2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Comparison between Single-Objective and Multi-Objective Genetic Algorithms: Performance Comparison and Performance Measures

Hisao Ishibuchi, *Member*, *IEEE*, Yusuke Nojima, *Member*, *IEEE*, and Tsutomu Doi, *Student Member*, *IEEE*

Abstract— **We compare single-objective genetic algorithms (SOGAs) with multi-objective genetic algorithms (MOGAs) in their applications to multi-objective knapsack problems. First we discuss difficulties in comparing a single solution by SOGAs with a solution set by MOGAs. We also discuss difficulties in comparing several solutions from multiple runs of SOGAs with a large number of solutions from a single run of MOGAs. It is shown that existing performance measures are not necessarily suitable for such comparison. Then we compare SOGAs with MOGAs through computational experiments on multi-objective knapsack problems. Experimental results on two-objective problems show that MOGAs outperform SOGAs even when they are evaluated with respect to a scalar fitness function used in SOGAs. This is because MOGAs are more likely to escape from local optima. On the other hand, experimental results on four-objective problems show that the search ability of MOGAs is degraded by the increase in the number of objectives. Finally we suggest a framework of hybrid algorithms where a scalar fitness function in SOGAs is probabilistically used in MOGAs to improve the convergence of solutions to the Pareto front.**

I. Introduction

Evolutionary multi-objective optimization (EMO) is one of the most active research areas in the field of evolutionary computation. Recently developed EMO algorithms usually share the three common features: Pareto ranking, diversity preserving, and elitism. While EMO algorithms have been successfully applied to various application areas [1]-[3], Pareto ranking-based EMO algorithms do not work well on many-objective optimization problems (e.g., see Hughes [4]). This is because solutions rarely dominate other solutions in the presence of many objectives. Hughes [4] showed that multiple runs of single-objective evolutionary algorithms outperformed a single run of EMO algorithms.

Whereas EMO algorithms do not work well on many-objective optimization problems, they work very well on two-objective problems. In some cases, EMO algorithms can outperform single-objective evolutionary algorithms even when they are used to solve single-objective problems. It was reported in some studies [5], [6] that better results were obtained by transforming single-objective problems into multi-objective ones.

This work was partially supported by Japan Society for the Promotion of Science (JSPS) through Grand-in-Aid for Scientific Research (B): KAKENHI (17300075).

Hisao Ishibuchi, Yusuke Nojima and Tsutomu Doi are with the Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan (hisaoi@cs.osakafu-u.ac.jp, nojima@cs.osakafu-u.ac.jp, doi@ci.cs.osakafu-u.ac.jp).

In this paper, we demonstrate the above-mentioned advantages and disadvantages of EMO algorithms through computational experiments on multi-objective knapsack problems [7] using one of the most frequently-used multi-objective genetic algorithms (MOGAs): NSGA-II [8]. For comparison, we also implement a single-objective genetic algorithm (SOGA) with the same architecture as NSGA-II. Our experimental results show that NSGA-II is outperformed by its single-objective version (i.e., SOGA) with a scalar fitness function when they are applied to four-objective knapsack problems. It is also shown that NSGA-II outperforms SOGA in their applications to two-objective knapsack problems even when they are evaluated with respect to a scalar fitness function defined by the sum of two objectives used in SOGA. Based on these experimental results, we suggest a framework of hybrid algorithms where a scalar fitness function in SOGA is probabilistically used in NSGA-II for parent selection and generation update in order to improve the convergence of solutions to the Pareto front.

This paper is organized as follows. First we explain some basic concepts in multiobjective optimization, MOGAs, and SOGAs in Section II. Next we discuss difficulties in comparing a single solution by SOGAs with a solution set by MOGAs in Section III. We also discuss difficulties in comparing several solutions from multiple runs of SOGAs with a large number of solutions from a single run of MOGAs. It is explained that existing performance measures are not necessarily suitable for such comparison. For example, the hypervolume measure [9] is sometimes misleading when we compare SOGAs with MOGAs. Then we evaluate the search ability of NSGA-II in comparison with its single-objective version through computational experiments on multi-objective knapsack problems with two, three and four objectives [8] in Section IV. A framework for hybridizing SOGAs into MOGAs is suggested in Section V. Finally we conclude this paper in Section VI.

II. Multiobjective Optimization

A. Multiobjective Optimization Problems

Let us consider the following k-objective maximization problem:

$$\text{Maximize } \mathbf{z} = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})), \tag{1}$$

$$\text{subject to } \mathbf{x} \in \mathbf{X}, \tag{2}$$

where $\mathbf{z}$ is the objective vector, $f_i(\mathbf{x})$ is the i-th objective to be maximized, $\mathbf{x}$ is the decision vector, and $\mathbf{X}$ is the feasible region in the decision space.

Let **x** and **y** be two feasible solutions of the *k*-objective maximization problem in (1)-(2). If the following conditions hold, **y** can be viewed as being better than **x**:

$$\forall\, i\,,\, f_i(\mathbf{x}) \le f_i(\mathbf{y}) \;\text{ and }\; \exists\, j\,,\, f_j(\mathbf{x}) < f_j(\mathbf{y})\,. \qquad (3)$$

In this case, we say that **y** dominates **x** (equivalently **x** is dominated by **y**).

When **x** is not dominated by any other feasible solutions (i.e., when there exists no feasible solution **y** that dominates **x**), the solution **x** is referred to as a Pareto-optimal solution of the *k*-objective maximization problem in (1)-(2). The set of all Pareto-optimal solutions forms the tradeoff surface in the objective space. This tradeoff surface is referred to as the Pareto front. Various EMO algorithms have been proposed to efficiently search for Pareto-optimal solutions [1]-[3].

B. Multi-Objective Genetic Algorithms: MOGAs

As a representative EMO algorithm, we use one of the most frequently-used MOGAs: NSGA-II of Deb et al. [8]. Let P and N_{pop} be the current population in NSGA-II and the population size, respectively (i.e., $N_{\text{pop}} = |P|$). Then the outline of NSGA-II can be written as follows:

```
Step 1: P := Initialize (P)
Step 2: while a termination condition is not satisfied, do
Step 3:     P' := Selection (P)
Step 4:     P'' := Genetic Operations (P')
Step 5:     P := Replace (P ∪ P'')
Step 6: end while
Step 7: return (non-dominated solutions (P))
```

First N_{pop} solutions are randomly generated to form an initial population P in Step 1 in the same manner as in standard GAs (i.e., SOGAs). Next N_{pop} pairs of parent solutions are selected from the current population P to form a parent population P' in Step 3. Then an offspring population P'' is constructed in Step 4 by generating a single offspring solution from each pair of parent solutions in P' by crossover and mutation. Genetic operations in Step 4 are the same as those in SOGAs. The next population is constructed in Step 5 by choosing the best N_{pop} solutions from the $2 \cdot N_{\text{pop}}$ solutions in the current population P and the offspring population P''. Parent selection in Step 3 and generation update in Step 5 of NSGA-II are different from SOGAs. Pareto ranking and a crowding measure are used to evaluate each solution in each step. The binary tournament selection is used in Step 3 to choose parent solutions. For details of NSGA-II, see Deb [1] and Deb et al. [8].

C. Single-Objective Genetic Algorithms: SOGAs

By using a scalar fitness function in Step 3 and Step 5 instead of Pareto ranking and a crowding measure, a SOGA algorithm can be implemented from NSGA-II. We use the following weighted sum scalar fitness function:

$$f(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \cdots + w_k f_k(\mathbf{x})\,, \qquad (4)$$

where $\mathbf{w} = (w_1, w_2, ..., w_k)$ is a non-negative weight vector.

The outline of SOGA is the same as NSGA-II in the previous subsection. The difference between NSGA-II and SOGA is that the scalar fitness function in (4) is used for parent selection and generation update in SOGA while Pareto ranking and a crowding measure are used in NSGA-II. That is, each solution is evaluated by the scalar fitness function in SOGA while the multiobjective evaluation of each solution in NSGA-II is based on Pareto ranking and a crowding measure.

When we search for a single solution by SOGA, we use a single weight vector. That is, a single solution is obtained from a single run of SOGA. On the other hand, we execute SOGA several times using different weight vectors when we search for multiple solutions by SOGA. That is, multiple solutions are obtained from multiple runs of SOGA.

In our computational experiments, we use three variants of SOGA: SOGA-1, SOGA-2 and SOGA-3. In SOGA-1, a single solution is obtained from a single run of SOGA with $\mathbf{w} = (1, 1, ..., 1)$. That is, the sum of the k objectives is used in SOGA-1 as the scalar fitness function. In SOGA-2 and SOGA-3, multiple solutions are obtained from multiple runs of SOGA. These two variants of SOGA are different from each other in the specification of the weight vector. In SOGA-2, we use $(2^k - 1)$ binary vectors excluding the zero vector (0, 0, ..., 0). For example, we use the three binary vectors (1, 0), (0, 1) and (1, 1) for two-objective problems. On the other hand, we use integer vectors in SOGA-3, which are generated from the following relations [10]:

$$w_1 + w_2 + \cdots + w_k = d\,, \qquad (5)$$

$$w_i \in \{0, 1, ..., d\} \text{ for } i = 1, 2, ..., k\,, \qquad (6)$$

where d is a prespecified integer. In this paper, we specify the value of d as $d = 4$. When $d = 4$, we have five integer vectors (4, 0), (3, 1), (2, 2), (1, 3), (0, 4) for two-objective problems. For three-objective problems, we have 15 integer vectors: (4, 0, 0), (3, 1, 0), (2, 2, 0), ..., (0, 1, 3), (0, 0, 4). For four-objective, we have 35 integer vectors: (4, 0, 0, 0), (3, 1, 0, 0), ..., (0, 0, 0, 4).

In order to compare a single run of NSGA-II with multiple runs of SOGA under the same computation load, we use the total number of evaluated solutions as a stopping condition of each algorithm. In preliminary computational experiments, we examined two settings for the execution of NSGA-II and SOGA under the same computation load. In one setting, we decreased the number of generations when SOGA was executed several times. In the other setting, we decreased the population size. Because better results were obtained from the latter setting, we adjust the population size in the case of multiple runs of SOGA in this paper.

Table I summarizes the population size and the number of weight vectors in the execution of the three variants of SOGA for multi-objective optimization problems with two, three, and four objectives. The number of weight vectors in each variant is the same as that of its runs. It should be noted that the population size of NSGA-II is the same as SOGA-1. In Table I, the product of the population size and the number of weight vectors is constant (i.e., 210) so that each algorithm examines exactly the same number of solutions.

TABLE I
THE NUMBER OF WEIGHT VECTORS AND POPULATION SIZE IN SOGA.

Problems	Population Size			Number of Weight Vectors		
	SOGA-1	SOGA-2	SOGA-3	SOGA-1	SOGA-2	SOGA-3
2-Objective	210	70	42	1	3	5
3-Objective	210	30	14	1	7	15
4-Objective	210	14	6	1	15	35

III. PERFORMANCE MEASURES

A single solution is obtained from a single run of SOGA while a large number of non-dominated solutions are usually obtained from a single run of MOGA. In this section, we discuss the comparison between a single solution by SOGA and a non-dominated solution set by MOGA. We also discuss the comparison between several solutions obtained by multiple runs of SOGAs and a large number of non-dominated solutions obtained by a single run of MOGA.

A. Dominance Relation between Solutions and Solution Sets

The relation between a single solution from a single run of SOGA and a set of non-dominated solutions from a single run of MOGA can be classified into four cases as shown in Fig. 1. In Fig. 1 (a), the SOGA solution S is dominated by some solutions in the MOGA solution set. In this case, we can say that MOGA outperforms SOGA since the inclusion of the SOGA solution S does not improve the quality of the MOGA solution set. On the other hand, we can say that SOGA outperforms MOGA in Fig. 1 (d) since the SOGA solution dominates all solutions in the MOGA solution set.

In Fig. 1 (b), the SOGA solution S is a non-dominated solution in the MOGA solution set. While the inclusion of the SOGA solution S somewhat improves the quality of the MOGA solution set, we cannot say that SOGA outperforms MOGA since the SOGA solution S dominates no MOGA solutions. We may intuitively say that MOGA outperforms SOGA in the case of Fig. 1 (b).

It is very difficult to say which is better between SOGA and MOGA in Fig. 1 (c) where the SOGA solution S dominates some MOGA solutions. If we use performance measures that are strongly related to the convergence of solutions (e.g., generational distance [11]), SOGA may be evaluated as being better than MOGA. On the contrary, if we use performance measures that are strongly related to the diversity of solutions (e.g., maximum spread [12]), MOGA may be evaluated as being better than SOGA.

B. Difficulties in Performance Evaluation

The hypervolume measure [9] has often been used to evaluate non-dominated solution sets since it is related to both the convergence and the diversity of solutions. In Fig. 2, the hypervolume for the SOGA solution S is the sum of the two areas B and D while it is the sum of the three areas A, B and C for the MOGA solution set. Thus the comparison of SOGA with MOGA is reduced to that of the area D with the sum of the two areas A and C. Let us explain that the hypervolume measure strongly depends on the location of the origin. For example, if the origin is close to the MOGA solution set as in Fig. 3 (a), SOGA may be viewed as being better than MOGA. On the contrary, MOGA may be viewed as being better than SOGA in Fig. 3 (b).

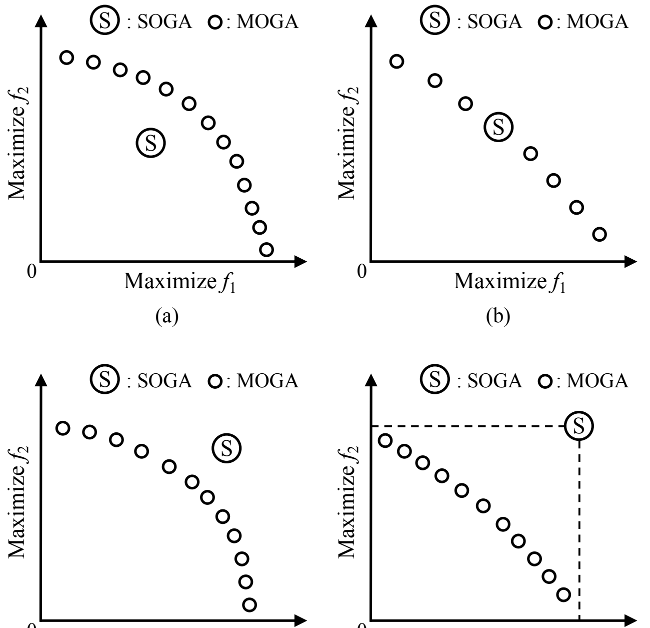

Fig. 1. Four cases of the relation between a single SOGA solution and a MOGA solution set of non-dominated solutions.

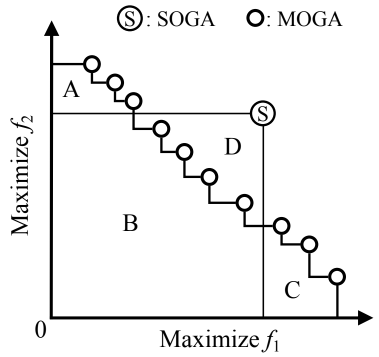

Fig. 2. Illustration of the hypervolume measure.

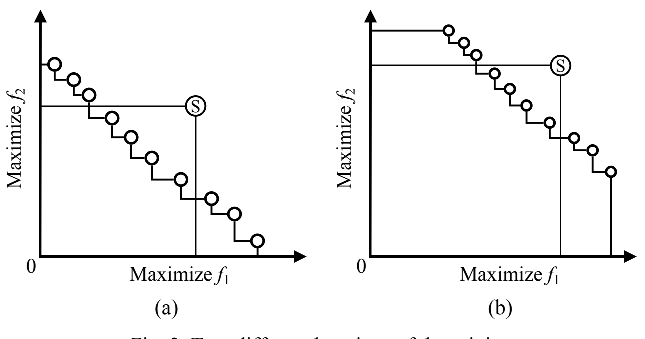

Fig. 3. Two different locations of the origin.

Several solutions can be obtained from multiple runs of SOGA. Let us assume that we have three SOGA solutions and a set of MOGA solutions as shown in Fig. 4. When we use the hypervolume measure, the comparison of the SOGA solution set with the MOGA solution set in Fig. 4 is that of the sum of the three areas A, C and E with the sum of the two areas B and D. As shown in Fig. 5 (a), MOGA may be viewed as being better than SOGA when the origin is close to the solution sets. On the contrary, SOGA may be viewed as being better than MOGA in Fig. 5 (b).

From these discussions, we can see that it is not easy to compare SOGA with MOGA. Some performance measures, which have been proposed to compare non-dominated solution sets with each other, are not necessarily suitable for comparing a single solution (or several solutions) by SOGA with a large number of non-dominated solutions by MOGA.

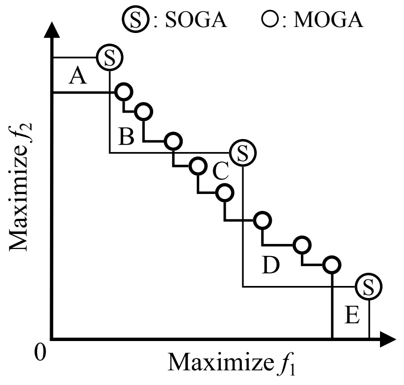

Fig. 4. A SOGA solution set and a MOGA solution set.

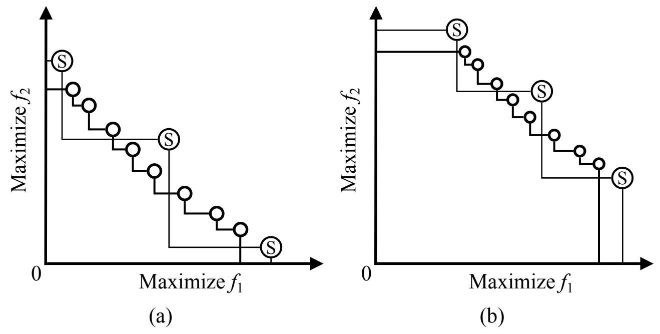

Fig. 5. Two different locations of the origin.

IV. Computational Experiments

A. Test Problems

As test problems, we use multi-objective 0/1 knapsack problems of Zitzler & Thiele [7]. In general, a k-objective n-item 0/1 knapsack problem (k-n test problem) in [7] can be written as follows:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})), \quad (7)$$

$$\text{subject to } \sum_{j=1}^{n} w_{ij} x_j \le c_i, \ i = 1, 2, ..., k, \quad (8)$$

$$x_j = 0 \text{ or } 1, \ j = 1, 2, ..., n, \quad (9)$$

$$\text{where } f_i(\mathbf{x}) = \sum_{j=1}^{n} p_{ij} x_j, \ i = 1, 2, ..., k. \quad (10)$$

In this formulation, $\mathbf{x}$ is an n-dimensional binary vector, p_{ij} is the profit of item j according to knapsack i, w_{ij} is the weight of item j according to knapsack i, and c_i is the capacity of knapsack i. Each solution $\mathbf{x}$ is handled as a binary string of length n.

When SOGA and MOGA are applied to the multi-objective knapsack problem in (7)-(10), genetic operations often generate infeasible solutions that do not satisfy the constraint conditions in (8). We use a repair method based on a maximum profit/weight ratio as in Zitzler & Thiele [7]. When an infeasible solution is generated, a feasible solution is created by removing items in the ascending order of the maximum profit/weight ratio defined as follows:

$$q_j = \max\{p_{ij} / w_{ij} \mid i = 1, 2, ..., k\}, \ j = 1, 2, ..., n. \quad (11)$$

B. Settings of Computational Experiments

We apply NSGA-II and the three variants of SOGA to the 500-item knapsack problems with two, three and four objectives (i.e., 2-500, 3-500 and 4-500 problems). The population size is specified as shown in Table I in Section II. In all computational experiments, we use the uniform crossover with the crossover probability 0.8 and the bit-flip mutation with the mutation probability 0.002 (i.e., 1/500). While we use various stopping conditions (i.e., various specifications of the total number of generations), the four algorithms are compared with one another under the same computation load in terms of the total number of examined solutions.

C. Results on the Two-Objective Test Problem

In this subsection, we report experimental results on the 2-500 problem. First we compare NSGA-II with SOGA-1 with respect to the following fitness function:

$$f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}), \quad (12)$$

which is used as a scalar fitness function in SOGA-1. In the case of NSGA-II, the best solution with respect to the scalar fitness function in (12) is chosen as the final solution from the non-dominated solution set obtained by its single run. Each algorithm is applied to the 2-500 problem 100 times by specifying the total number of generation as 2000 (i.e., the stopping condition of each algorithm is 2000 generations). Fig. 6 shows the histograms of 100 solutions obtained by each algorithm. It should be noted that SOGA-1 and NSGA-II are executed with the same population size (i.e., 210, see Table I in Section II). From Fig. 6, we can see that NSGA-II outperforms SOGA-1 even when they are compared with each other in terms of the scalar fitness function. This is because MOGAs are more likely to escape from local optima as pointed out by some studies [5], [6].

While NSGA-II outperforms SOGA-1 in terms of the convergence of solutions to the Pareto front, the diversity of obtained non-dominated solutions is not large if compared with the entire Pareto front. In Fig. 7, we show the entire

Pareto front together with the 50% attainment surfaces [13] over 100 solutions sets obtained by NSGA-II and SOGA-3. As in Fig. 6, the stopping condition of each algorithm is 2000 generations in Fig. 7. It should be noted in Fig. 7 that a single run of NSGA-II uses the same computation load as five runs of SOGA-3.

Fig. 8 shows how the average value of the hypervolume measure is increased by each algorithm over 100 runs. The hypervolume is measured using the origin of the objective space. From Fig. 8, we can see that SOGA-2 and SOGA-3 outperform NSGA-II in the long run. We can also see that better results are obtained by NSGA-II than SOGA-2 and SOGA-3 in early generations.

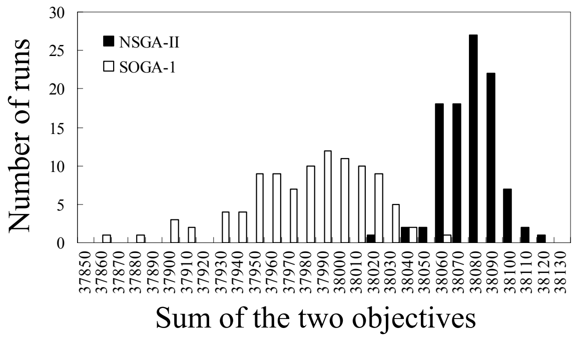

Fig. 6. Histograms of 100 solutions for the 2-500 problem.

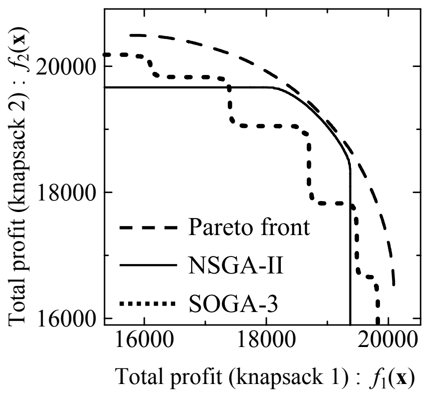

Fig. 7. A solution set by NSGA-II and five solutions by SOGA-3.

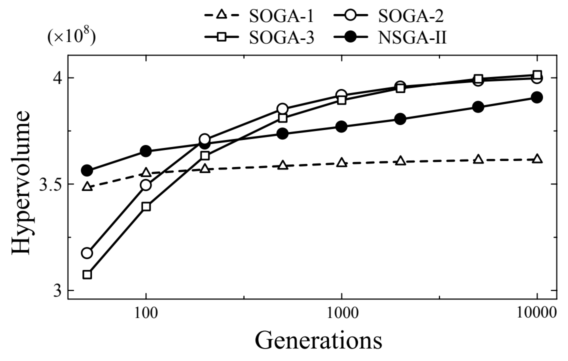

Fig. 8. Comparison using the hypervolume measure on the 2-500 problem.

We also compare NSGA-II with each of the three SOGA variants using the dominance relation. For example, in the comparison between NSGA-II and SOGA-3, we count the number of NSGA-II solutions that dominate at least one SOGA-3 solution. We also count the number of NSGA-II solutions that are dominated by at least one SOGA-3 solution. Average results over 100 runs are summarized in Table II. For example, the column labeled "SOGA-2" under "# of dominating solutions" shows the average number of NSGA-II solutions that dominate at least one SOGA-2 solution. From the last three columns of Table II, we can see that no NSGA-II solutions are dominated by any SOGA solutions after 2000 generations. We can also see from Table II that many NSGA-II solutions dominate SOGA solutions. These observations show the high search ability of NSGA-II in terms of the convergence of solutions to the Pareto front.

TABLE II
THE AVERAGE NUMBER OF NSGA-II SOLUTIONS DOMINATING SOGA SOLUTIONS AND DOMINATED BY SOGA SOLUTIONS (2-500 PROBLEM).

Generations	# of dominating solutions			# of dominated solutions		
	SOGA-1	SOGA-2	SOGA-3	SOGA-1	SOGA-2	SOGA-3
200	1.38	35.10	43.49	1.92	0	0
400	4.83	35.01	51.78	0.50	0	0
600	6.80	34.09	51.40	0.06	0	0
800	8.09	34.88	50.82	0.01	0	0
1000	8.79	33.73	49.77	0.04	0	0
1500	9.66	30.90	44.05	0.01	0	0
2000	9.86	26.65	40.27	0	0	0

D. Results on the Three-Objective Test Problem

In the same manner as in Fig. 6, we compare NSGA-II with SOGA-1 by applying them to the 3-500 problem 100 times. Experimental results are shown in Fig. 9. From Fig. 6 and Fig. 9, we can see that the search ability of NSGA-II in terms of the convergence of solutions to the Pareto front is deteriorated by the increase in the number of objectives.

The performance of the four algorithms on the 3-500 problem is compared in Fig. 10 using the average value of the hypervolume measure over 100 runs of each algorithm. As in Fig. 8, we can see that SOGA-2 and SOGA-3 outperform NSGA-II in the long run while better results are obtained by NSGA-II in early generations.

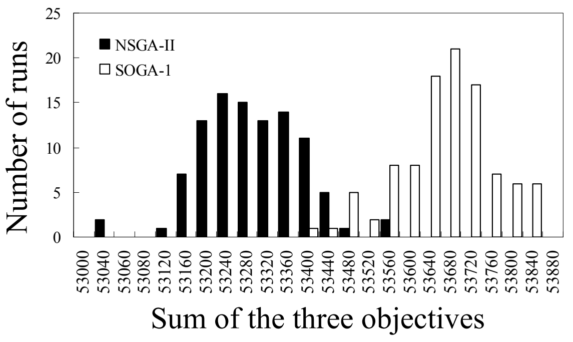

Fig. 9. Histograms of 100 solutions for the 3-500 problem.

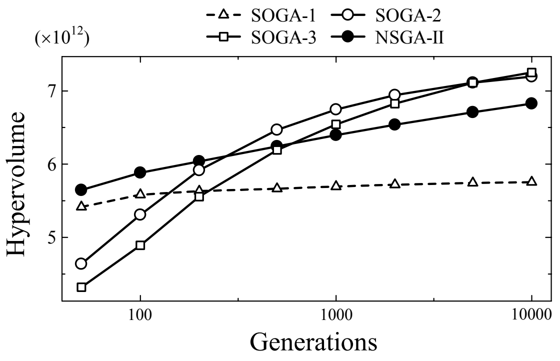

Fig. 10. Comparison using the hypervolume measure on the 3-500 problem.

Poor performance of SOGA-2 and SOGA-3 in early generations may be due to their small population size. As shown in Table I in Section II, each run of SOGA-2 and SOGA-3 is executed using a small population size. Thus SOGA-2 and SOGA-3 need a larger number of generations than NSGA-II to obtain good solutions. Since SOGA-3 uses a smaller population than SOGA-2 (see Table I), SOGA-3 is inferior to SOGA-2 in early generations in Fig. 7 and Fig. 10. In the long run, SOGA-3 outperforms SOGA-2 with respect to the hypervolume measure since more solutions are obtained by SOGA-3 (i.e., more weight vectors are used in SOGA-3).

Poor performance of NSGA-II in Fig. 10 in the long run is mainly due to the lack of the diversity in obtained non-dominated solutions while their convergence to the Pareto front is not enough either as shown in Fig. 9. For the case of the 2-500 problem, the lack of the diversity is demonstrated in Fig. 6. In order to visually examine the diversity of non-dominated solutions obtained by a single run of NSGA-II, we show their projections onto the two-objective space with the first two objectives in Fig. 11. For comparison, we also show the projections of obtained solutions by SOGA-3. From Fig. 11, we can see that NSGA-II does not find non-dominated solutions in a wide region of the objective space if compared with SOGA-3.

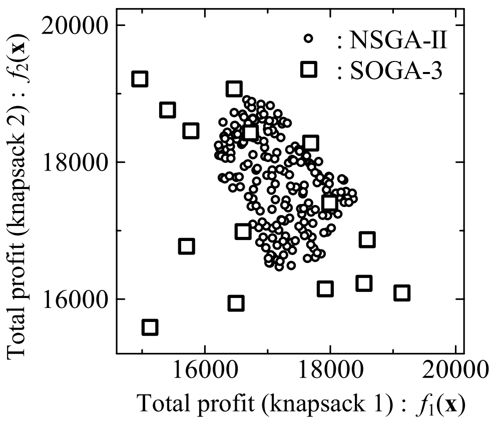

Fig. 11. Projections of solutions obtained by SOGA-3 and NSGA-II for the 3-500 problem onto the two-dimensional space with the first two objectives.

In the same manner as Table II, we compare NSGA-II with each of the three SOGA variants using the dominance relation. That is, we count the number of NSGA-II solutions dominating at least one SOGA solution. We also count the number of NSGA-II solutions that are dominated by at least one SOGA solution. Average results over 100 runs are summarized in Table III. We can see that many NSGA-II solutions are dominated by SOGA-1 solutions (see the fifth column labeled as SOGA-1). We can also see that no NSGA-II solutions dominate SOGA-1 solutions (see the second column). These observations show that the search ability of NSGA-II is deteriorated by the increase in the number of objectives (see Fig. 9).

TABLE III
THE AVERAGE NUMBER OF NSGA-II SOLUTIONS DOMINATING SOGA SOLUTIONS AND DOMINATED BY SOGA SOLUTIONS (3-500 PROBLEM).

Generations	# of dominating solutions			# of dominated solutions		
	SOGA-1	SOGA-2	SOGA-3	SOGA-1	SOGA-2	SOGA-3
200	0	179.14	187.41	18.41	0	0
400	0.03	102.33	194.94	10.80	0	0
600	0	50.37	187.77	10.28	0	0
800	0	24.93	157.68	10.54	0	0
1000	0	13.20	125.29	11.79	0.08	0
1500	0	2.90	63.89	12.53	0.24	0
2000	0	1.12	35.15	12.80	0.87	0.06

E. Results on the Four-Objective Test Problem

Experimental results on the 4-500 problem are shown in Fig. 12. The search ability of NSGA-II in terms of the convergence of solutions to the Pareto front is further degraded by the increase in the number of objectives. The comparison among Fig. 6, Fig. 9 and Fig. 12 clearly shows how the increase in the number of objectives degrades the convergence of solutions to the Pareto front.

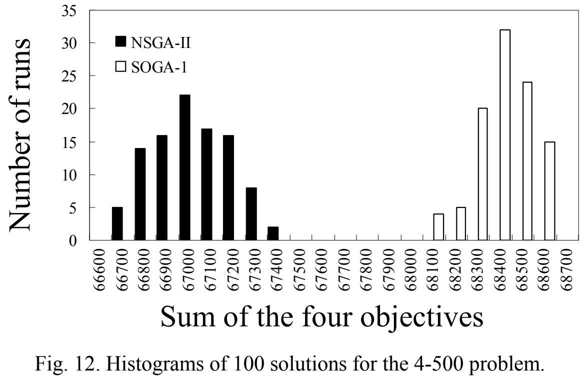

Fig. 12. Histograms of 100 solutions for the 4-500 problem.

The four algorithms are compared with one another in Fig. 13 for the 4-500 problem using the hypervolume measure in the same manner as in Fig. 8 and Fig. 10. We can obtain the same observation from the three figures: SOGA-2 and SOGA-3 outperform NSGA-II in the long run while better results are obtained by NSGA-II in early generations.

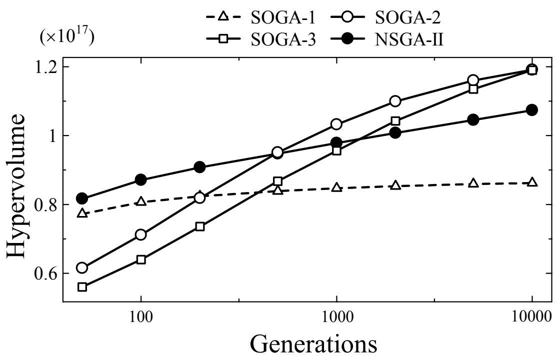

Fig. 13. Comparison using the hypervolume measure on the 4-500 problem.

In the same manner as Table II and Table III, we compare NSGA-II with each of the three SOGA variants using the dominance relation. Average results over 100 runs are summarized in Table IV. We can see that many NSGA-II solutions are dominated by SOGA solutions (see the last three columns). We can also see that no NSGA-II solutions dominate SOGA-1 solutions (see the second column). From the comparison between the two columns labeled as SOGA-2 (i.e., the third and sixth columns), we can see that more SOGA-2 solutions come to dominate NSGA-II solutions as the number of generations increases. This observation coincides with the relative performance of NSGA-II in Fig. 13.

TABLE IV
THE AVERAGE NUMBER OF NSGA-II SOLUTIONS DOMINATING SOGA SOLUTIONS AND DOMINATED BY SOGA SOLUTIONS (4-500 PROBLEM).

Generations	# of dominating solutions			# of dominated solutions		
	SOGA-1	SOGA-2	SOGA-3	SOGA-1	SOGA-2	SOGA-3
200	0	201.85	201.88	47.69	0	0
400	0	160.35	202.80	44.16	0	0
600	0	77.47	202.45	43.19	0	0
800	0	34.71	196.54	41.53	0.14	0
1000	0	15.53	175.55	40.56	0.45	0
1500	0	2.15	108.51	39.03	2.11	0.06
2000	0	0.41	61.11	38.58	4.51	0.36

V. HYBRID ALGORITHMS

A. Basic Idea

From our experimental results in the previous section, we can obtain the following observations:

(1) NSGA-II can outperform SOGA in their applications to single-objective optimization problems. That is, NSGA-II is more likely to escape from local optima.

(2) SOGA can outperform NSGA-II in their applications to many-objective optimization problems in terms of the convergence of solutions to the Pareto front. The search ability of NSGA-II is deteriorated by the increase in the number of objectives.

(3) Multiple runs of SOGA can outperform a single run of NSGA-II with respect to the diversity of solutions.

That is, SOGA and NSGA-II have their own advantages and disadvantages. This suggests that the hybridization of these two algorithms seems to be a promising idea.

As we have already explained, SOGA and NSGA-II are different from each other only in their solution evaluation mechanisms for parent selection and generation update. Whereas the multi-objective evaluation based on Pareto ranking and a crowding measure is used in NSGA-II, the single-objective evaluation based on a scalar fitness function is used in SOGA. A simple approach to the hybridization of these two algorithms is to probabilistically use the multi-objective evaluation and the single-objective evaluation for parent selection and generation update.

Let P_{PS} be the probability of using the single-objective evaluation for parent selection. When a pair of parents are to be selected from the current population, the single-objective evaluation and the multi-objective evaluation are used with the probabilities P_{PS} and $(1-P_{PS})$, respectively. Two parents recombined by crossover are selected by the same evaluation mechanism. When another pair of parents are to be selected, the single-objective or multiobjective evaluation is probabilistically chosen in the same manner.

Generation update can be also performed in a similar probabilistic manner. Let P_{GS} be the probability of using the single-objective evaluation for generation update. When a solution is to be chosen from the current and offspring populations for generation update, the single-objective evaluation and the multi-objective evaluation are used with the probabilities P_{GS} and $(1-P_{GS})$, respectively. Each solution in the next population is chosen in this manner.

NSGA-II and SOGA correspond to the two extreme cases with $P_{PS} = P_{GS} = 0$ and $P_{PS} = P_{GS} = 1$, respectively. Our idea is to implement a hybrid algorithm with high search ability by appropriately specifying the two probabilities P_{PS} and P_{GS}. It should be noted that we can use an arbitrary single-objective evaluation mechanism in our hybrid algorithm. For example, we can use a weighted sum fitness function with a prespecified weight vector such as SOGA-1. We can also use a set of multiple scalar fitness functions. For example, we can randomly or systematically use weighted sum fitness functions with different weight vectors.

B. Preliminary Results

Using the above-mentioned idea, we implement a hybrid algorithm of NSGA-II and SOGA-1. That is, we use the sum of the k objectives as a scalar fitness function with the probabilities P_{PS} and P_{GS} for parent selection and generation update in NSGA-II, respectively. Experimental results on the 2-500 and 4-500 problems are shown in Fig. 14 and Fig. 15 where NSGA-II, SOGA-1 and the hybrid algorithm are compared with one another in terms of the scalar fitness function (i.e., the sum of the k objectives). Different values of P_{PS} and P_{GS} are used for each problem as shown in the caption of each figure (e.g., $P_{PS} = 0.5$ and $P_{GS} = 0.0$ for the 2-500 problem). Those parameter values are specified after examining several combinations.

In these figures, the hybrid algorithm works well on the test problems. The hybrid algorithm is more likely to escape from local optima than SOGA-1 in their applications to the 2-500 problem (see Fig. 14). That is, the hybrid algorithm shares the global search ability with NSGA-II. On the other hand, the hybrid algorithm has good convergence property as SOGA-1 in their applications to the 4-500 problems (see Fig. 15). Similar results to Fig. 15 are obtained for the 3-500 test problem whereas we do not show them due to the page limitation. These observations suggest that the hybridization of MOGAs and SOGAs is a promising approach to multiobjective optimization. Of course, further studies are needed to evaluate the performance of the hybrid algorithm in comparison with various EMO algorithms using more reliable performance measures such as the attainment function [13], [14]. Parameter specifications in the hybrid algorithm should be also examined further.

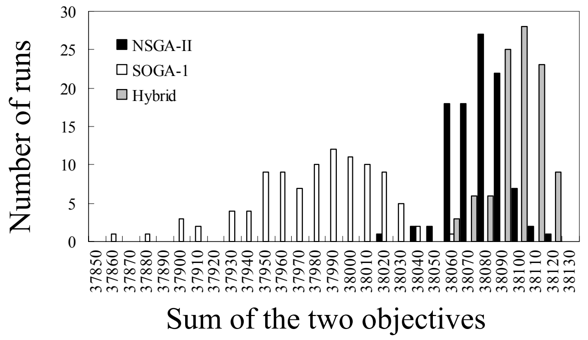

Fig. 14. Histograms for the 2-500 problem ($P_{PS} = 0.5$ and $P_{GS} = 0.0$).

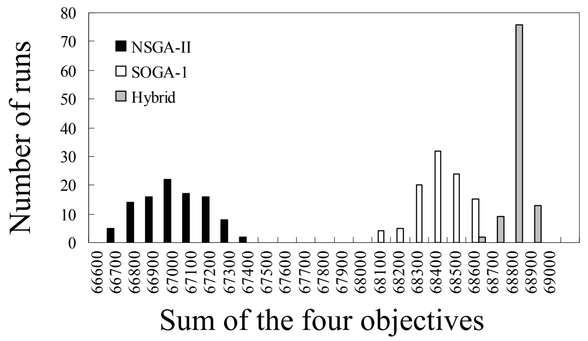

Fig. 16. Histograms for the 4-500 problem ($P_{PS} = 0.8$ and $P_{GS} = 0.1$).

VI. Conclusions

In this paper, we clearly demonstrated advantages and disadvantages of SOGA and NSGA-II. Better results were obtained from NSGA-II than SOGA in their applications to the 2-500 problem when they were evaluated by a scalar fitness function. This observation supports the use of MOGAs in single-objective optimization problems in order to escape from local optima. The search ability of NSGA-II was degraded in terms of the convergence of solutions to the Pareto front by the increase in the number of objectives. This observation coincides with reported results in the literature : EMO algorithms did not work well on many-objective optimization problems (e.g., see Jaszkiewicz [15] for experimental results on multiobjective knapsack problems). We also showed that multiple runs of SOGA outperformed NSGA-II. This is mainly due to the lack of the diversity of non-dominated solutions obtained by NSGA-II. Based on these observations, we suggest an idea of hybridizing SOGA into NSGA-II. Our idea is much simpler than other proposals of hybrid algorithms such as a hierarchical algorithm [16].

In addition to these empirical examinations, we also discussed the difficulty in comparing a single solution (or several solutions) by SOGA with a large number of non-dominated solutions by NSGA-II.

References

[1] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, 2001.

[2] C. A. Coello Coello, D. A. van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, Boston, 2002.

[3] C. A. Coello Coello and G. B. Lamont, *Applications of Multi-Objective Evolutionary Algorithms*, World Scientific, Singapore, 2004.

[4] E. J. Hughes, "Evolutionary many-objective optimization: many once or one many?," *Proc. of 2005 IEEE Congress on Evolutionary Computation*, pp. 222-227, Edinburgh, UK, September 2-5, 2005.

[5] J. D. Knowles, R. A. Watson, and D. W. Corne, "Reducing local optima in single-objective problems by multi-objectivization," *Lecture Notes in Computer Science*, vol. 1993, pp. 269-283, Springer, Berlin, March 2001.

[6] S. Watanabe and K. Sakakibara, "Multi-objective approaches in a single-objective optimization environment," *Proc. of 2005 IEEE Congress on Evolutionary Computation*, pp. 1714-1721, Edinburgh, UK, September 2-5, 2005.

[7] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, November 1999.

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

[9] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms – A comparative case study," *Lecture Notes in Computer Science*, vol. 1498, pp. 292-301, Springer, Berlin, September 1998.

[10] T. Murata, H. Ishibuchi, and M. Gen, "Specification of genetic search directions in cellular multi-objective genetic algorithms," *Lecture Notes in Computer Science*, vol. 1993, pp. 82-95, Springer, Berlin, March 2001.

[11] D. A. Van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, Ph. D dissertation, Air Force Institute of Technology, Dayton, 1999.

[12] E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, Ph. D dissertation, Swiss Federal Institute of Technology, Zurich, Shaker Verlag, Aachen, 1999.

[13] C. M. Fonseca and P. J. Fleming, "On the performance assessment and comparison of stochastic multiobjective optimizers," *Lecture Notes in Computer Science*, vol. 1141, pp. 584-593, Springer, Berlin, June 1996.

[14] V. G. da Fonseca, C. M. Fonseca, and A. O. Hall, "Inferential performance assessment of stochastic optimizers and the attainment function," *Lecture Notes in Computer Science*, vol. 1993, pp. 213-225, Springer, Berlin, March 2001.

[15] A. Jaszkiewicz, "On the computational efficiency of multiple objective metaheuristics: The knapsack problem case study," *European Journal of Operational Research*, vol. 158, no. 2, pp. 418-433, October 2004.

[16] C. L. Mumford, "A hierarchical solve-and-merge framework for multi-objective optimization," *Proc. of 2005 IEEE Congress on Evolutionary Computation*, pp. 2241-2247, Edinburgh, UK, September 2-5, 2005.