H. Ishibuchi, T. Doi, and Y. Nojima, “Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms,” *Lecture Notes in Computer Science 4193: Parallel Problem Solving from Nature - PPSN IX*, pp. 493-502, Springer, Berlin, September 2006.

Incorporation of Scalarizing Fitness Functions into Evolutionary Multiobjective Optimization Algorithms

Hisao Ishibuchi, Tsutomu Doi, and Yusuke Nojima

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, doi@ci.cs.osakafu-u.ac.jp
nojima@cs.osakafu-u.ac.jp
http://www.ie.osakafu-u.ac.jp/~hisaoi/ci_lab_e

Abstract. This paper proposes an idea of probabilistically using a scalarizing fitness function in evolutionary multiobjective optimization (EMO) algorithms. We introduce two probabilities to specify how often the scalarizing fitness function is used for parent selection and generation update in EMO algorithms. Through computational experiments on multiobjective 0/1 knapsack problems with two, three and four objectives, we show that the probabilistic use of the scalarizing fitness function improves the performance of EMO algorithms. In a special case, our idea can be viewed as the probabilistic use of an EMO scheme in single-objective evolutionary algorithms (SOEAs). From this point of view, we examine the effectiveness of our idea. Experimental results show that our idea improves not only the performance of EMO algorithms for multiobjective problems but also that of SOEAs for single-objective problems.

1 Introduction

Evolutionary multiobjective optimization (EMO) is one of the most active research areas in the field of evolutionary computation. EMO algorithms have been successfully applied to various application areas [2]. Most EMO algorithms use Pareto ranking to evaluate the fitness of each solution. Pareto ranking-based EMO algorithms, however, do not work well on many-objective problems (e.g., see [5], [6], [8], [12], [16]). This is because solutions rarely dominate other solutions in the presence of many objectives. Hughes [6] showed that multiple runs of single-objective evolutionary algorithms (SOEAs) outperformed a single run of EMO algorithms in their applications to many-objective problems. Similar results were also reported in [8], [12]. Whereas EMO algorithms do not work well on many-objective problems, usually they work very well on two-objective problems. In some cases, EMO algorithms can outperform SOEAs even when they are used to solve single-objective problems. It was reported in some studies [13], [18] that better results were obtained by transforming single-objective problems into multi-objective ones.

These experimental results suggest that SOEAs and EMO algorithms have their own advantages and disadvantages. In this paper, we hybridize them into a single algorithm in order to simultaneously utilize their advantages. More specifically, we

propose an idea of probabilistically using a scalarizing fitness function for parent selection and generation update in EMO algorithms. Following this idea, we implement a hybrid algorithm using NSGA-II [3] and a weighted sum fitness function. The weighted sum fitness function is probabilistically used for parent selection and generation update in NSGA-II. We introduce two probabilities to specify how often the weighted sum fitness function is used for parent selection and generation update.

We use NSGA-II because it is one of the most frequently-used EMO algorithms in the literature. The use of the weighted sum fitness function is due to its simplicity. Of course, other scalarizing fitness functions can be used in our hybrid algorithm. A scalarizing fitness function-based EMO algorithm was proposed by Hughes [5] in a general form. The weighted sum fitness function was successfully used in multiobjective genetic local search (MOGLS) algorithms [7], [9], [10]. High performance of MOGLS of Jaszkiewicz [10] was reported [1], [11], [14]. The weighted sum fitness function was also used in a two-stage EMO algorithm of Mumford [14].

The main feature of our hybrid algorithm is the probabilistic use of the weighted sum fitness function. When the probability of its use is very low, our hybrid algorithm is almost the same as NSGA-II. The increase in the probability of its use intensifies the flavor of weighted sum-based algorithms. Another feature is the flexibility in the specification of the weight vector in the weighted sum fitness function. We can use a set of uniformly distributed weight vectors for multiobjective optimization as well as a single weight vector for single-objective optimization. In this paper, we first explain our hybrid algorithm in Section 2. Then we examine its performance as single-objective and multiobjective algorithms in Section 3 and Section 4, respectively.

2 Implementation of a Hybrid Algorithm

In this section, we implement a hybrid algorithm by incorporating a weighted sum fitness function into NSGA-II [3]. We introduce two probabilities P_{PS} and P_{GU}, which specify how often the weighted sum fitness function is used for parent selection and generation update, respectively.

Let us consider the following k-objective maximization problem:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ ...,\ f_k(\mathbf{x})) \text{ subject to } \mathbf{x} \in \mathbf{X}, \tag{1}$$

where $\mathbf{f}(\mathbf{x})$ is the k-dimensional objective vector, $\mathbf{x}$ is the decision vector, and $\mathbf{X}$ is the feasible region in the decision space. When the following relation holds between two feasible solutions $\mathbf{x}$ and $\mathbf{y}$, $\mathbf{x}$ is said to be dominated by $\mathbf{y}$ (i.e., $\mathbf{y}$ is better than $\mathbf{x}$):

$$\forall i,\ f_i(\mathbf{x}) \le f_i(\mathbf{y}) \quad \text{and} \quad \exists j,\ f_j(\mathbf{x}) < f_j(\mathbf{y}). \tag{2}$$

When there is no feasible solution $\mathbf{y}$ in $\mathbf{X}$ that dominates $\mathbf{x}$, $\mathbf{x}$ is referred to as a Pareto-optimal solution. Usually multiobjective optimization problems have a large number of Pareto-optimal solutions. The set of objective vectors corresponding to all Pareto-optimal solutions is referred to as Pareto front.

2.1 Description of NSGA-II as a General Evolutionary Algorithm

NSGA-II of Deb et al. [3] is an elitist EMO algorithm with the $(\mu + \lambda)$-ES generation update mechanism. The outline of NSGA-II can be written as follows:

[NSGA-II]
Step 1: P = Initialize(P)
Step 2: While the stopping condition is not satisfied, do
Step 3: P' = Parent Selection(P)
Step 4: P'' = Genetic Operations(P')
Step 5: P = Generation Update($P \cup P''$)
Step 6: End while
Step 7: Return Non-dominated(P)

In NSGA-II, each solution in the current population is evaluated using Pareto ranking and a crowding measure in the following manner for parent selection in Step 3. First the best rank (i.e., Rank 1) is assigned to all the non-dominated solutions in the current population. Solutions with Rank 1 are tentatively removed from the current population. Next the second best rank (i.e., Rank 2) is assigned to all the non-dominated solutions in the remaining population. Solutions with Rank 2 are tentatively removed from the remaining population. In this manner, ranks are assigned to all solutions in the current population. Solutions with smaller rank values are viewed as being better than those with larger rank values. A crowding measure is used to compare solutions with the same rank. Roughly and informally speaking for two-objective problems, the crowding measure of a solution is the Manhattan distance between its two adjacent solutions in the objective space (for details, see [2], [3]). When two solutions have the same rank, one solution with a larger value of the crowding measure is viewed as being better than the other with a smaller value.

A prespecified number of pairs of parent solutions are selected from the current population by binary tournament selection to form a parent population P' in Step 3. An offspring solution is generated from each pair of parent solutions by crossover and mutation to form an offspring population P'' in Step 4. The current population and the offspring population are merged to form an enlarged population. Each solution in the enlarged population is evaluated by Pareto ranking and the crowding measure as in the parent selection phase. A prespecified number of the best solutions are chosen from the enlarged population as the next population P in Step 5.

2.2 Weighted Sum Fitness Function

The weighted sum fitness function of the k objectives in (1) is written as follows:

$$fitness(\mathbf{x}) = w_1 \cdot f_1(\mathbf{x}) + w_2 \cdot f_2(\mathbf{x}) + \ \ldots \ + w_k \cdot f_k(\mathbf{x}), \tag{3}$$

where w_i is a non-negative weight value.

One important issue is the specification of the weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_k)$. We examine the following three versions in our hybrid algorithm.

Version I: The weight vector is always specified as $\mathbf{w} = (1, 1, ..., 1)$. That is, we always use the following scalarizing fitness function:

$$fitness(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ ... \ + f_k(\mathbf{x}). \tag{4}$$

Version II: A different weight vector is randomly chosen from the $(2^k - 1)$ binary vectors excluding the zero vector (0, 0, ..., 0). For example, a weight vector is randomly chosen from the three binary vectors (1, 1), (1, 0) and (0, 1) in the case of $k = 2$ (i.e., two-objective problems). We have 7 and 15 non-zero binary vectors for three-objective and four-objective problems, respectively.

Version III: A different vector is randomly chosen from a set of non-negative integer vectors satisfying the following relation: $w_1 + w_2 + \ ... \ + w_k = d$ where d is a prespecified integer. In this paper, d is specified as $d = 4$ (Other values should be examined in the future study). In the case of two-objective problems, a weight vector is randomly chosen from the five integer vectors (4, 0), (3, 1), (2, 2), (1, 3) and (0, 4). 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 problems, we have 35 integer vectors: (4, 0, 0, 0), (3, 1, 0, 0), ..., (0, 0, 0, 4). The same idea of the weight vector specification was used in [12], [14], [15].

2.3 Hybrid Algorithm

Our hybrid algorithm is the same as NSGA-II except for parent selection in Step 3 and generation update in Step 5. When a pair of parent solutions are to be selected from the current population, the weighted sum fitness function and the NSGA-II fitness evaluation mechanism are used with the probabilities P_{PS} and $(1 - P_{PS})$, respectively. When another pair of parent solutions are to be selected, the probabilistic choice between the two fitness evaluation schemes is performed again. As in NSGA-II, we always use binary tournament selection independent of the chosen fitness evaluation scheme. It should be noted that we use a randomly chosen weight vector in Version II and Version III of our hybrid algorithm.

As in the parent selection phase, we probabilistically use the weighted sum fitness function in the generation update phase. When one solution is to be chosen from the enlarged population and added to the next population, the weighted sum fitness function and the NSGA-II fitness evaluation mechanism are used with the probabilities P_{GU} and $(1 - P_{GU})$, respectively. When another solution is to be chosen, the probabilistic choice between the two fitness evaluation schemes is performed again.

One extreme case of our hybrid algorithm with $P_{PS} = P_{GU} = 0.0$ is exactly the same as the pure NSGA-II since the weighted sum fitness function is never used. Another extreme case with $P_{PS} = P_{GU} = 1.0$ is a weighted sum-based genetic algorithm with the $(\mu + \lambda)$-ES generation update mechanism. In this case, Version I algorithm is a single-objective genetic algorithm (SOGA) since the scalarizing fitness function in (4) is always used. Version II and Version III algorithms with $P_{PS} = P_{GU} = 1.0$ are EMO algorithms, which are somewhat similar to VEGA of Schaffer [17].

3 Single-Objective Optimization by Our Hybrid Algorithm

In this section, we examine the performance of our hybrid algorithm as a single-objective optimization algorithm for maximizing the sum of the k-objectives (i.e., the scalarizing fitness function in (4)). We use Version I of our hybrid algorithm where the scalarizing fitness function in (4) is used with the probabilities P_{PS} and P_{GU} for parent selection and generation update, respectively. As test problems, we use three 500-item knapsack problems with two, three and four objectives in [19]. These test problems are denoted as 2-500, 3-500 and 4-500, respectively. Our hybrid algorithm is applied to each test problem using the following parameter specifications:

Population size: 200 (i.e., $\mu = \lambda = 200$),
Crossover probability: 0.8 (uniform crossover),
Mutation probability: 0.002 (bit-flip mutation),
Probability P_{PS}: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,
Probability P_{GU}: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,
Termination condition: 2000 generations.

We examine the 11×11 combinations of the 11 values of P_{PS} and P_{GU}. Our hybrid algorithm with each combination of P_{PS} and P_{GU} is applied to each test problem 50 times. Average results over 50 runs are summarized in Fig. 1 for the 2-500 problem. In this figure, the performance of the pure SOGA with $P_{PS} = P_{GU} = 1.0$ at the top-right corner is improved by probabilistically using the NSGA-II fitness evaluation mechanism for generation update (i.e., by decreasing P_{GU} from $P_{GU} = 1.0$ to $P_{GU} < 1.0$). It is interesting to observe that even the pure NSGA-II with $P_{PS} = P_{GU} = 0.0$ at the bottom-left corner outperforms the pure SOGA with $P_{PS} = P_{GU} = 1.0$. This observation supports the idea of using EMO algorithms for single-objective optimization to escape from local optima [13], [18].

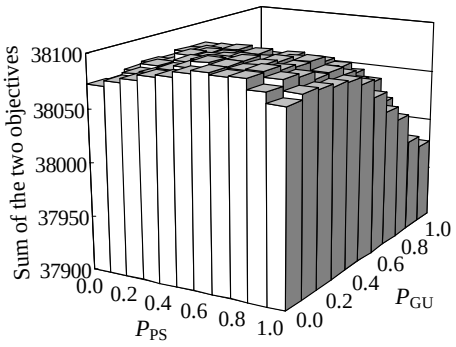


Fig. 1. Experimental results by Version I of our hybrid algorithm on the 2-500 problem.

In order to further examine the performance of the three algorithms (i.e., our hybrid algorithm and its two extreme cases: NSGA-II and SOGA), each algorithm is applied to the 2-500 knapsack problem 100 times. In our hybrid algorithm, P_{PS} and

P_{GU} are specified as P_{PS} = 0.5 and P_{GU} = 0.0. Fig. 2 shows the histograms of the obtained 100 values of the scalarizing fitness function in (4) by each algorithm. From Fig. 2 (and also from Fig. 1), we can see that NSGA-II and our hybrid algorithm outperform SOGA even when they are evaluated as single-objective optimization algorithms for maximizing the sum of the two objectives of the 2-500 problem.

The performance of SOGA is also improved by probabilistically using the NSGA-II fitness evaluation mechanism for the knapsack problems with three and four objectives (i.e., 3-500 and 4-500). Experimental results are shown in Fig. 3. The search ability of the pure SOGA with P_{PS} = 1.0 and P_{GU} = 1.0 at the top-right corner is improved by using the NSGA-II fitness evaluation mechanism for generation update (i.e., by decreasing P_{GU}). The pure NSGA-II with P_{PS} = 0.0 and P_{GU} = 0.0 at the bottom-left corner, however, cannot find good solutions with respect to the scalarizing fitness function. From the comparison between Fig. 1 and Fig. 3, we can see that the convergence ability of NSGA-II is degraded by increasing the number of objectives. This observation coincides with some studies on the performance of Pareto ranking-based EMO algorithms for many-objective optimization [6], [8], [12], [16].

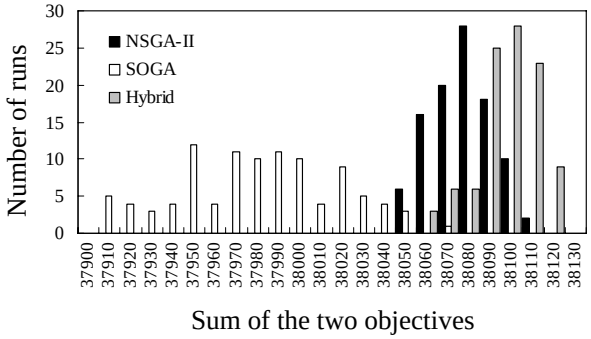
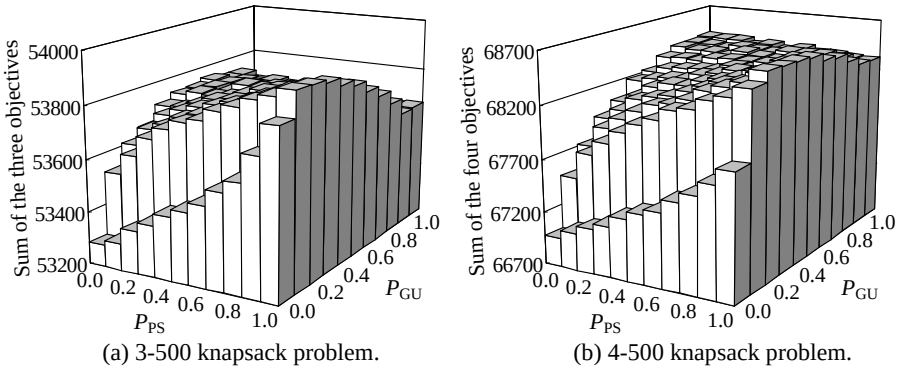


Fig. 2. Histograms of the obtained 100 values of the scalarizing fitness function by each of NSGA-II, SOGA and our Version I hybrid algorithm with P_{PS} = 0.5 and P_{GU} = 0.0.



(a) 3-500 knapsack problem. (b) 4-500 knapsack problem.

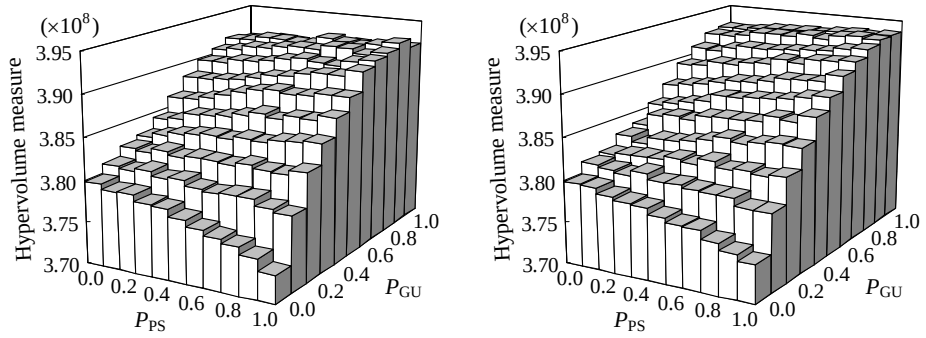
Fig. 3. Experimental results of our Version I hybrid algorithm.

4 Multi-Objective Optimization by Our Hybrid Algorithm

In this section, we examine the performance of our hybrid algorithm as a multi-objective optimization algorithm. As in the previous section, we use the three 500-item knapsack problems as test problems. Version II and Version III of our hybrid algorithm are applied to each test problem using the same parameter specifications as in the previous section. Each version of our hybrid algorithm is applied to each test problem 50 times using each of the 11×11 combinations of P_{PS} and P_{GU}.

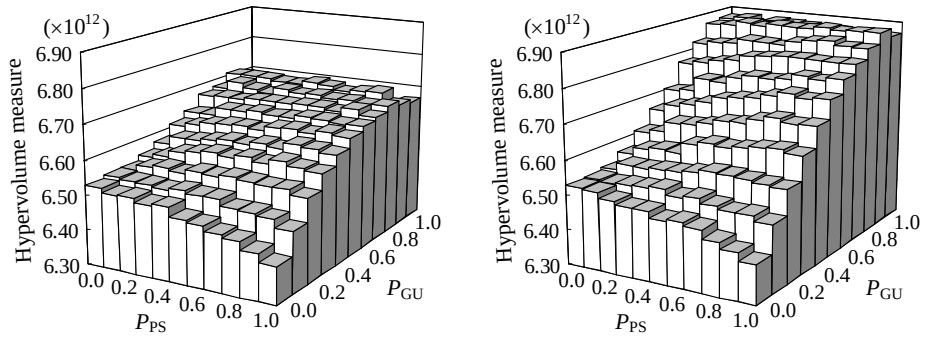
In each run of our hybrid algorithm (i.e., Version II and Version III), we calculate the hypervolume measure (e.g., see Deb [2]) after the 2000th generation. Average results over 50 runs are summarized in Figs. 4-6.

The choice of a performance measure is very difficult. Whereas we only use the hypervolume (due to the page limitation), it is not necessarily the best choice [8]. We may need other performance measures in addition to the hypervolume. For example, Jaszkiewicz [12] proposed an idea of using achievement scalarizing functions. For the 2-500 problem, we also show the 50% attainment surface [4] later.



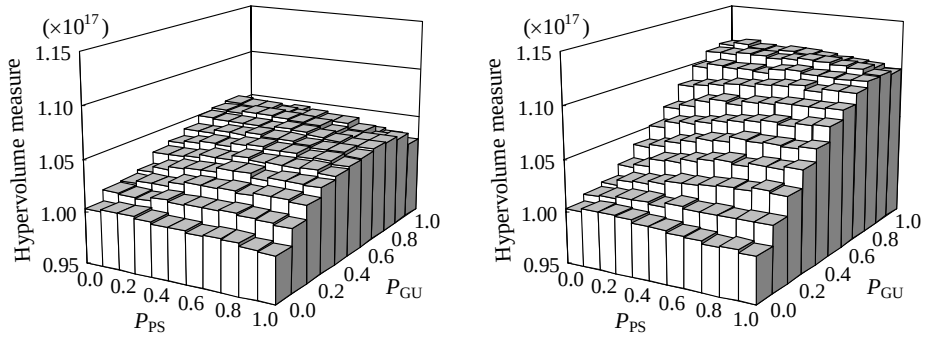
(a) Version II with binary weight vectors. (b) Version III with integer weight vectors.

Fig. 4. Average values of the hypervolume measure for the 2-500 knapsack problem.



(a) Version II with binary weight vectors. (b) Version III with integer weight vectors.

Fig. 5. Average values of the hypervolume measure for the 3-500 knapsack problem.



(a) Version II with binary weight vectors. (b) Version III with integer weight vectors.

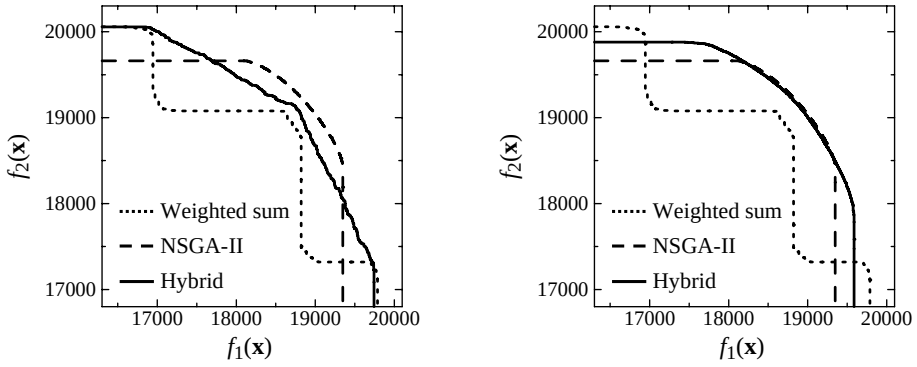
Fig. 6. Average values of the hypervolume measure for the 4-500 knapsack problem.

In Figs. 4-6, the performance of the pure NSGA-II at the bottom-left corner with $P_{PS} = P_{GU} = 0.0$ is improved by probabilistically using the weighted sum fitness function for generation update (i.e., by increasing P_{GU} from $P_{GU} = 0.0$ to $P_{GU} > 0.0$). Good results are also obtained when the weighted sum fitness function is used with high probabilities for both parent selection and generation update (i.e., around the top-right corner). An interesting observation is that the use of the weighted sum fitness function only for parent selection (i.e., $P_{PS} > 0.0$ and $P_{GU} = 0.0$: experimental results along the bottom row) clearly degrades the performance of NSGA-II especially for the 2-500 problem in Fig. 4 and the 3-500 problem in Fig. 5.

In all the six plots in Figs. 4-6, good results are obtained by the weighted sum-based EMO algorithm at the top-right corner with $P_{PS} = P_{GU} = 1.0$. Its performance, however, can be further improved by appropriately specifying the two probabilities P_{PS} and P_{GU}. For example, better results are obtained in Fig. 6 around the top-left corner with a small value of P_{PS} and a large value of P_{GU} than the top-right corner with $P_{PS} = P_{GU} = 1.0$. For all the three test problems, better results are obtained from Version III with integer weight vectors than Version II with binary vectors.

In Fig. 7, we show the 50% attainment surface [4] over 50 runs of our hybrid algorithm on the 2-500 knapsack problem for some combinations of P_{PS} and P_{GU} including the two extreme cases (i.e., the pure NSGA-II and the pure weighted sum-based EMO algorithm). We use our Version II hybrid algorithm in Fig. 7 with the three binary weight vectors (1, 1), (1, 0) and (0, 1). As a result, the 50% attainment surface has three peaks in the case of the pure weighted sum-based EMO algorithm as shown by the dotted line labeled as “Weighted sum” in Fig. 7. Whereas NSGA-II can find better solutions than the center peak of the pure weighted sum-based EMO algorithm, it cannot find extreme solutions around the other two peaks. Depending on the specifications of the two probabilities P_{PS} and P_{GU}, our hybrid algorithm finds different solution sets. In Fig. 7 (a), the 50% attainment surface by our hybrid algorithm with $P_{PS} = 1.0$ and $P_{GU} = 0.9$ is similar to but clearly better than that of the pure weighted sum-based EMO algorithm. On the other hand, the 50% attainment surface by our hybrid algorithm with $P_{PS} = 0.5$ and $P_{GU} = 0.5$ in Fig. 7 (b) is similar to but has larger diversity than that of NSGA-II.

From Fig. 7, we can see that the probabilistic use of the weighted sum fitness function increases the diversity of obtained non-dominated solutions. Such an increase in the diversity leads to the improvement in the hypervolume measure for the 2-500 problem in Fig. 4. Not only the diversity improvement but also the convergence improvement, however, contributes to the improvement in the hypervolume measure in Fig. 5 for the 3-500 problem and Fig. 6 for the 4-500 problem. Actually, the convergence performance of NSGA-II is improved by the probabilistic use of the weighted sum fitness function in all the four combinations of the two versions (i.e., Version II and Version III) and the two test problems (i.e., 3-500 and 4-500). Such improvement has already been demonstrated for the case of our Version I hybrid algorithm in Fig. 3 for the 3-500 and 4-500 problems.



(a) $P_{PS} = 1.0$ and $P_{GU} = 0.9$. (b) $P_{PS} = 0.5$ and $P_{GU} = 0.5$.

Fig. 7. 50% attainment surface over 50 runs by our Version II hybrid algorithm for the 2-500 problem. Experimental results by the two extreme cases (i.e., the pure NSGA-II and the pure weighted sum-based EMO algorithm) are also shown for comparison.

5 Conclusions

In this paper, we proposed an idea of probabilistically using a scalarizing fitness function for parent selection and generation update in EMO algorithms. Following the proposed idea, we implemented a hybrid algorithm by incorporating the weighted sum fitness function into NSGA-II. When our hybrid algorithm was used for single-objective optimization, it outperformed SOGA. That is, the probabilistic use of the NSGA-II fitness evaluation mechanism improved the performance of SOGA. On the other hand, when our hybrid algorithm was used for multiobjective optimization, it outperformed NSGA-II in terms of the hypervolume measure. That is, the probabilistic use of the weighted sum fitness function improved the performance of NSGA-II. Future work includes the comparison of our hybrid algorithm with other approaches to many-objective optimization problems (e.g., [5], [6], [12], [14]).

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

References

1. Colombo, G., Mumford, C. L.: Comparing Algorithms, Representations and Operators for the Multi-objective Knapsack Problem. Proc. of 2005 Congress on Evolutionary Computation (2005) 2241-2247
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6 (2002) 182-197
4. Fonseca, C. M., Fleming, P. J.: On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. Lecture Notes in Computer Science, Vol. 114: Parallel Problem Solving from Nature - PPSN IV. Springer, Berlin (1996) 584-593
5. Hughes, E. J.: Multiple Single Objective Sampling. Proc. of 2003 Congress on Evolutionary Computation (2003) 2678-2684
6. Hughes, E. J.: Evolutionary Many-objective Optimization: Many Once or One Many? Proc. of 2005 Congress on Evolutionary Computation (2005) 222-227
7. Ishibuchi, H., Murata, T.: A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews 28 (1998) 392-403
8. Ishibuchi, H., Nojima, Y., Doi, T.: Comparison between Single-objective and Multi-objective Genetic Algorithms: Performance Comparison and Performance Measures. Proc. of 2006 Congress on Evolutionary Computation (2006) (in press)
9. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling. IEEE Trans. on Evolutionary Computation 7 (2003) 204-223
10. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization. European Journal of Operational Research 137 (2002) 50-71
11. Jaszkiewicz, A.: On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem - A Comparative Experiment. IEEE Trans. on Evolutionary Computation 6 (2002) 402-412
12. Jaszkiewicz, A.: On the Computational Efficiency of Multiple Objective Metaheuristics: The Knapsack Problem Case Study. European Journal of Operational Research 158 (2004) 418-433
13. Knowles, J. D., Watson, R. A., Corne, D. W.: Reducing Local Optima in Single-Objective Problems by Multi-Objectivization. Lecture Notes in Computer Science, Vol. 1993: Evolutionary Multi-Criterion Optimization - EMO 2001. Springer, Berlin (2001) 269-283
14. Mumford, C. L.: A Hierarchical Solve-and-Merge Framework for Multi-Objective Optimization. Proc. of 2005 Congress on Evolutionary Computation (2005) 2241-2247
15. Murata, T., Ishibuchi, H., Gen, M.: Specification of Genetic Search Directions in Cellular Multi-Objective Genetic Algorithms. Lecture Notes in Computer Science, Vol. 1993: Evolutionary Multi-Criterion Optimization - EMO 2001, Springer, Berlin (2001) 82-95
16. Purshouse, R. C., Fleming, P. J.: Evolutionary Many-Objective Optimization: An Exploratory Analysis. Proc. of 2003 Congress on Evolutionary Computation (2003) 2066-2073
17. Schaffer, J. D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Proc. of 1st International Conference on Genetic Algorithms and Their Applications (1985) 93-100
18. Watanabe, S., Sakakibara K.: Multi-Objective Approaches in a Single-Objective Optimization Environment. Proc. of 2005 Congress on Evolutionary Computation (2005) 1714-1721
19. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans. on Evolutionary Computation 3 (1999) 257-271