Some Issues on the Implementation of Local Search in Evolutionary Multiobjective Optimization

Hisao Ishibuchi and Kaname Narukawa

Department of Industrial Engineering, Osaka Prefecture University,
1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan
{hisaoi, kaname}@ie.osakafu-u.ac.jp

Abstract. This paper discusses the implementation of local search in evolutionary multiobjective optimization (EMO) algorithms for the design of a simple but powerful memetic EMO algorithm. First we propose a basic framework of our memetic EMO algorithm, which is a hybrid algorithm of the NSGA-II and local search. In the generation update procedure of our memetic EMO algorithm, the next population is constructed from three populations: the current population, its offspring population generated by genetic operations, and an improved population obtained from the offspring population by local search. We use Pareto ranking and the concept of crowding in the same manner as in the NSGA-II for choosing good solutions to construct the next population from these three populations. For implementing local search in our memetic EMO algorithm, we examine two approaches, which have been often used in the literature: One is based on Pareto ranking, and the other is based on a weighted scalar fitness function. The main difficulty of the Pareto ranking approach is that the movable area of the current solution by local search is very small. On the other hand, the main difficulty of the weighted scalar approach is that the offspring population can be degraded by local search. These difficulties are clearly demonstrated through computational experiments on multiobjective knapsack problems using our memetic EMO algorithm. Our experimental results show that better results are obtained from the weighted scalar approach than the Pareto ranking approach. For further improving the weighted scalar approach, we examine some tricks that can be used for overcoming its difficulty.

1 Introduction

Evolutionary multiobjective optimization (EMO) algorithms have been applied to various problems for efficiently finding their Pareto-optimal or near Pareto-optimal solutions. Recent EMO algorithms usually share some common ideas such as elitism, fitness sharing and Pareto ranking for improving both the diversity of solutions and the convergence to the Pareto front (e.g., see Coello et al. [1] and Deb [2]). In some studies, local search was combined for further improving the search ability of EMO algorithms [4]-[12].

Hybridization of EMO algorithms with local search is often referred to as MOGLS (multiobjective genetic local search) algorithms. Such a hybrid algorithm is also called a memetic EMO algorithm. It is clearly shown by Jaszkiewicz [9] that memetic EMO algorithms have higher search ability than pure EMO algorithms. Memetic EMO algorithms can be roughly classified into two categories according to their solution evaluation mechanisms in local search: One uses a weighted scalar fitness function, and the other uses Pareto ranking. A memetic EMO algorithm based on the weighted scalar fitness function with random weights was proposed by Ishibuchi & Murata [6], improved by Jaszkiewicz [10] and Ishibuchi et al. [7], and simplified by Ishibuchi & Kaige [5]. On the other hand, Knowles & Corne [11] proposed a memetic EMO algorithm called M-PAES (memetic Pareto archived evolution strategy) where each solution was evaluated based on Pareto ranking. Some Pareto ranking-based acceptance rules of local search moves were examined in Ishibuchi et al. [7] and Murata et al. [13]. The MOGLS of Jaszkiewicz [10] and the M-PAES of Knowles & Corne [11], which are well-known memetic EMO algorithms with high search ability, have been compared with each other in some comparative studies [4], [8], [9], [12].

Our goal in this paper is to design a simple but powerful memetic EMO algorithm. For achieving this goal, we discuss some issues related to the implementation of local search in EMO algorithms through computational experiments on multiobjective knapsack problems in Zitzler & Thiele [14]. Our computational experiments are performed using a framework of a simple MOGLS algorithm, which is proposed in this paper by combining the NSGA-II [3] with local search. In order to emphasize its simplicity, we refer to our MOGLS algorithm as the simple MOGLS (i.e., S-MOGLS) algorithm in this paper. As in the NSGA-II, we use Pareto ranking and the concept of crowding for generation update in our S-MOGLS algorithm. One characteristic feature of our generation update procedure is the use of three populations for generating the next population: the current population, its offspring population generated by genetic operations, and an improved population obtained from the offspring population by local search. In the existing MOGLS algorithms [5]-[10], the offspring population that had been improved by local search was not used in their generation update procedures. In this sense, our S-MOGLS can be viewed as an improved version of the S-MOGLS of Ishibuchi & Kaige [5].

Through computational experiments on multiobjective knapsack problems using our S-MOGLS algorithm, we compare the above-mentioned two approaches to the implementation of local search. We show that better results are obtained by the weighted scalar approach than the Pareto ranking approach. We also demonstrate a serious difficulty of the weighted scalar approach: Local search often degrades the offspring population generated by genetic operations. For overcoming this difficulty, we examine the effectiveness of the following tricks:

(1) The use of the three populations in the generation update procedure in our S-MOGLS algorithm. Generation update procedures with/without the offspring population are compared with each other.
(2) The choice of good initial solutions for local search from the offspring population. The tournament selection of initial solutions based on the weighted scalar fitness function is examined using various specifications of the tournament size.
(3) The modification of the acceptance rule of local search moves. We examine a modified acceptance rule in the weighted scalar approach.

This paper is organized as follows. In Section 2, we propose a basic framework of our S-MOGLS algorithm. While we try to maximize the search ability of our S-MOGLS algorithm, we also try to minimize its algorithmic complexity so that it can be easily understood, easily implemented and efficiently executed using small memory storage within short CPU time. In Section 3, we show various variants of our S-MOGLS algorithm. In one variant, the weighted scalar fitness function is used in the selection of parent solutions and the local search for their offspring. In another variant, Pareto ranking instead of the weighted scalar fitness function is used for both the parent selection and the local search. Of course, there exist many intermediate variants between these two extremes. In Section 4, we show experimental results on multiobjective knapsack problems using some variants of our S-MOGLS algorithm. Finally Section 5 concludes this paper.

2 Basic Framework of Our S-MOGLS Algorithm

Let us consider the following k-objective maximization problem:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x}))\,, \tag{1}$$
$$\text{subject to } \mathbf{x} \in \mathbf{X}\,, \tag{2}$$

where $\mathbf{f}(\mathbf{x})$ 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. When the following two conditions are satisfied, a feasible solution $\mathbf{x} \in \mathbf{X}$ is said to be dominated by another feasible solution $\mathbf{y} \in \mathbf{X}$ (i.e., $\mathbf{y}$ dominates $\mathbf{x}$: $\mathbf{y}$ is better than $\mathbf{x}$):

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

If there is no feasible solution $\mathbf{y}$ that dominates $\mathbf{x}$, $\mathbf{x}$ is said to be a Pareto-optimal solution of the multiobjective optimization problem. The task of EMO algorithms is to find Pareto-optimal or near Pareto-optimal solutions as many as possible.

The following weighted scalar fitness function was used in the MOGLS algorithms of Ishibuchi et al. [5]-[7] and Jaszkiewicz [8]-[10]:

$$f(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x})\,, \tag{4}$$

where

$$\forall i\ ,\ \lambda_i \geq 0 \ \text{ and } \ \sum_{i=1}^{k} \lambda_i = 1 . \tag{5}$$

In those MOGLS algorithms, the weight vector $\lambda = (\lambda_1, ..., \lambda_k)$ was randomly specified whenever a pair of parent solutions was to be selected. The roulette wheel selection was used in the original MOGLS algorithm [6]. In Ishibuchi et al. [7], better results were obtained by the tournament selection than the roulette wheel selection. On the other hand, a pair of parent solutions was randomly chosen from the best *K* solutions in the current population with respect to the weighted scalar fitness function in Jaszkiewicz [8]-[10]. The same weighted scalar fitness function with the current weight vector, which had been used for choosing a pair of parent solutions, was also used in local search for their offspring generated by genetic operations.

In our S-MOGLS algorithm, we use the weighted scalar fitness function in (4) for choosing a pair of parent solutions from the current population. Since the tournament selection needs less CPU time than the random selection from the best *K* solutions, we use the tournament selection in our S-MOGLS algorithm (more specifically the binary tournament slection). The selected parents are recombined to generate their offspring to which a mutation operation is applied. The selection, recombination and mutation are iterated to generate the offspring population (see Fig. 1). Local search is applied only to good solutions in the offspring population. For choosing initial solutions for local search from the offspring population, we use the tournament selection based on the weighed scalar fitness function in (4). Whenever an initial solution is chosen, the weight vector is randomly updated. Local search is probabilistically applied to the selected initial solution. The idea of choosing good initial solutions was proposed in Ishibuchi et al. [7]. Only when the initial solution is updated (i.e., improved) by local search, the improved solution is added to the improved population in Fig. 1.

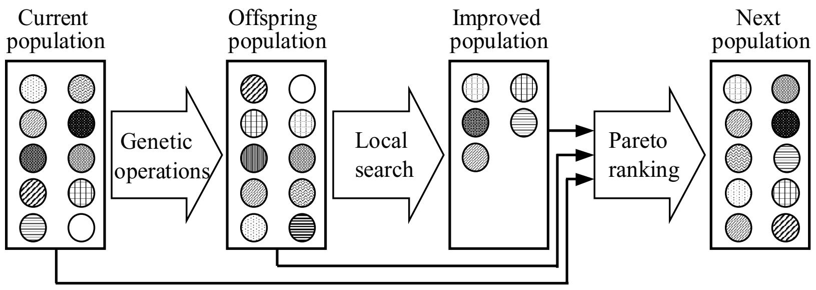


Fig. 1. Generation update mechanism in our S-MOGLS algorithm.

Recently it has been widely recognized in the EMO community that some form of elitism is necessary for designing EMO and memetic EMO algorithms with high search ability (e.g., see Deb [2]). One straightforward implementation of elitism is to store non-dominated solutions in the secondary population separately from the main (i.e., current) population. For example, the secondary population was used in the MOGLS algorithms of Ishibuchi et al. [6]-[7] and Jaszkiewicz [8]-[10], the M-PAES [11] and the SPEA [14]. While the use of the secondary population significantly improves the search ability of EMO and memetic EMO algorithms, it also increases algorithmic complexity, memory storage and CPU time. Thus we do not use the secondary population in our S-MOGLS algorithm. Instead of the secondary population, we use the generation update scheme of the NSGA-II [3] for choosing good solutions from the three populations in Fig. 1 (i.e., current, offspring and improved populations). That is, each solution in the three populations is evaluated by Pareto ranking and the concept of crowding in the same manner as in the NSGA-II [3].

The outline of our S-MOGLS algorithm is written as follows:

Basic Framework of Our S-MOGLS Algorithm:

Step 1 *(Initialization):* Generate an initial population with N_{pop} solutions where N_{pop} is the population size.

Step 2 *(Genetic operations):* Generate an offspring population by iterating the following procedures N_{pop} times:
- (1) Randomly specify the weight vector.
- (2) Choose a pair of parent solutions from the current population using the binary tournament selection based on the weighted scalar fitness function with the current weight vector.
- (3) Generate an offspring from the selected parents by crossover and mutation.

Step 3 *(Local search):* Generate an improved population by iterating the following procedures N_{pop} times:
- (1) Randomly specify the weight vector.
- (2) Choose an initial solution for local search from the offspring population using the binary tournament selection based on the weighted scalar fitness function with the current weight vector.
- (3) Apply a local search procedure based on the weighted scalar fitness function with the current weight vector to the selected initial solution with the local search application probability P_{LS}. Only when the initial solution is updated by the local search, the final solution at which the local search is terminated is added to the improved population.

Step 4 *(Generation update):* Construct the next population from the current, offspring and improved populations by choosing good solutions based on Pareto ranking and the concept of crowding in the same manner as in the NSGA-II.

Step 5 *(Termination test):* If the pre-specified stopping condition is not satisfied, return to Step 2. Otherwise terminate the execution of the algorithm.

3 Several Variants of Our S-MOGLS Algorithm

Various variants of our S-MOGLS algorithm can be implemented. In this section, we briefly describe those variants of our S-MOGLS algorithm. Some of them are used in computational experiments in the next section for discussing the implementation of local search in memetic EMO algorithms.

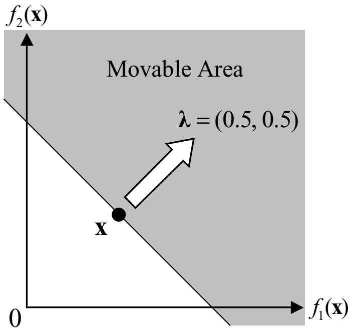
Selection of Parent Solutions: In our S-MOGLS algorithm, we use the weighted scalar fitness function for parent selection. This is to choose similar parents in the objective space, from which good offspring are likely to be generated. Of course, we can use other selection schemes. For example, we can use the parent selection scheme of the NSGA-II [3] (i.e., Pareto ranking and the concept of crowding).

Selection of Initial Solutions for Local Search: In our S-MOGLS algorithm, we choose good initial solutions from the offspring population for local search. It is also possible to probabilistically apply local search to offspring solutions independent of their performance. Moreover it is possible to apply local search to all offspring as in some memetic EMO algorithms (e.g., the S-MOGLS algorithms in [6]-[10]).

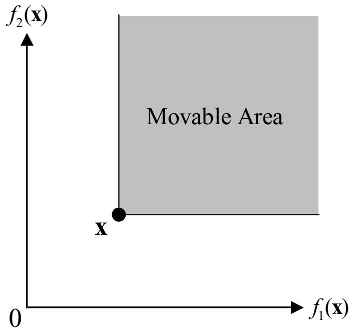
Local Search: In our S-MOGLS algorithm, we use the weighted scalar fitness function in local search as well as parent selection. We can also use Pareto ranking in local search. When we use Pareto ranking, the current solution $\mathbf{x}$ is replaced with its neighboring solution $\mathbf{y}$ (i.e., the local search move from $\mathbf{x}$ to $\mathbf{y}$ is accepted) only when $\mathbf{y}$ dominates $\mathbf{x}$ (i.e., $\mathbf{y}$ is better than $\mathbf{x}$: see (3)). That is, the local search move is rejected when $\mathbf{x}$ and $\mathbf{y}$ are non-dominated with each other. In the M-PAES [11], a more sophisticated acceptance rule was used for handling the situation where $\mathbf{y}$ and $\mathbf{x}$ are incomparable with each other. The acceptance rule in [11] involves not only the comparison between the current solution $\mathbf{x}$ and the candidate solution $\mathbf{y}$, but also the comparison with other solutions. This may somewhat degrade the inherent advantage of local search: simplicity. Thus we do not use the local search procedure in [11].

Let us further discuss the weighted scalar approach and the Pareto ranking approach to the implementation of local search. In Fig. 2, we show the movable area of the current solution $\mathbf{x}$ by each approach in the case of a two-objective maximization problem. In Fig. 2 (a), the weight vector was specified as $\boldsymbol{\lambda} = (\lambda_1, \lambda_2) = (0.5,\ 0.5)$. As shown in Fig. 2, we can see that the movable area in the Pareto ranking approach is much smaller than the weighted scalar approach. This difference exponentially increases with the number of objectives because the movable area in the Pareto ranking approach is $1/2^k$ of the k-dimensional objective space while it is always 1/2 in the case of the weighted scalar approach.

As shown by computational experiments in the next section (and other studies [7], [13]), too small movable areas in the Pareto ranking approach prevent local search from efficiently searching for good solutions. On the other hand, too large movable areas in the weighted scalar approach sometimes lead to the deterioration of the offspring population. Thus we examine a simple modification of the weighted scalar approach for decreasing the movable areas as shown in Fig. 3 (a). More specifically, the local search move from the current solution **x** to the candidate solution **y** is accepted when the inequality relation $d / r < a$ holds in Fig. 3 (b) where a is a user-definable parameter. Of course, the candidate solution **y** should be better than the current solution **x** with respect to the weighted scalar fitness function. It should be noted that the weighted scalar approach corresponds to the case of $a = \infty$.

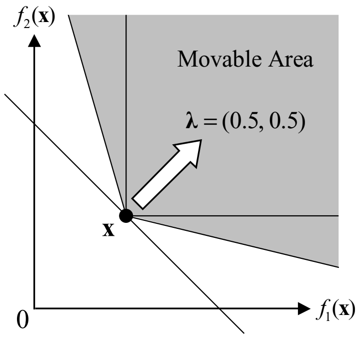


(a) Weighed scalar approach.

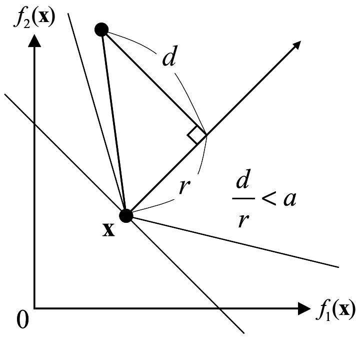


(b) Pareto ranking approach.

Fig. 2. Movable area of the current solution **x** in the two-dimensional objective space.



(a) Movable area in the modified approach.



(b) Definition of the acceptance rule.

Fig. 3. Modification of the weighted scalar approach.

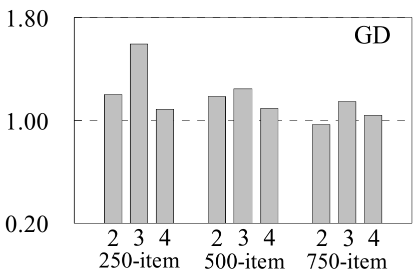
4 Computational Experiments

In our computational experiments, we used nine knapsack problems in Zitzler & Thiele [14]: 2-250, 2-500, 2-750, 3-250, 3-500, 3-750, 4-250, 4-500, 4-750 where "*k*-*m*" means a *k*-objective *m*-item problem. Our computational experiments were performed in the same manner as in other comparative studies (e.g., Ishibuchi & Kaige [4], [5], Jaszkiewicz [9], and Knowles & Corne [12], and Zitzler & Thiele

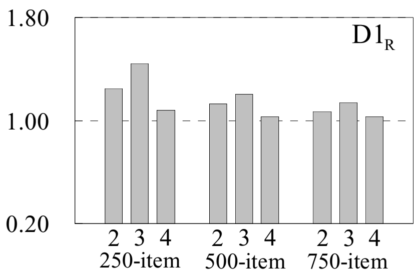
[14]). We used the same parameter specifications as the NSGA-II in those comparative studies for the EMO part of our S-MOGLS algorithm and as the M-PAES and the MOGLS in those studies for the local search part.

For examining the performance of obtained non-dominated solution sets by our S-MOGLS algorithm, we use the generational distance (GD) and the $\mathrm{D1_R}$ measure (see Coello [1] and Deb [2] for various performance measures). These measures evaluate the quality of a non-dominated solution set using a reference solution set. The reference solution set is a set of Pareto-optimal or near Pareto-optimal solutions. In our computational experiments, the reference solution set for each test problem was constructed by choosing non-dominated solutions among all solutions obtained in our previous computational experiments. The GD measure is the average distance from each solution in the obtained solution set to its nearest reference solution. This measure evaluates the convergence to the Pareto front. On the other hand, the $\mathrm{D1_R}$ measure is the average distance from each reference solution to its nearest solution in the obtained solution set. This measure evaluates both the convergence and the diversity of obtained solutions. In our computational experiments, the average values of these measures were calculated over 30 runs for each test problem.

Comparison between two approaches: We first compared the Pareto ranking approach to the weighted scalar approach using the S-MOGLS algorithm in Section 2. The local search application probability P_{LS} was specified as $P_{\mathrm{LS}} = 0.1$. The relative performance of the Pareto ranking approach with respect to the weighted scalar approach was calculated for each test problem. Experimental results are summarized in Fig. 4. In these figures, the horizontal axis shows the nine test problems. From Fig. 4, we can see that the relative performance of the Pareto ranking approach is larger than 1.00 (which is the relative performance of the weighted scalar approach) for many test problems. This means that the Pareto ranking approach is inferior to the weighted scalar approach because the GD and $\mathrm{D1_R}$ measures should be minimized.



(a) The generational distance.



(b) The $\mathrm{D1_R}$ measure.

Fig. 4. Relative performance of the Pareto ranking approach with respect to the weighted scalar approach for each test problem.

While better results were obtained from the weighted scalar approach, it has a serious difficulty: The offspring population can be degraded by the local search. This difficulty is illustrated in Fig. 5. In Fig. 5 (a), we show an offspring population and its improved population. These populations are intermediate results during a computational experiment using our S-MOGLS algorithm on the 2-500 problem. Special parameter values were used in this computational experiment for illustration purpose (e.g., a small population size and a large local search application probability). Fig. 5 (b) shows why the offspring population can be degraded by the local search. As shown in Fig. 5 (b), the offspring population is likely to be degraded when the local search direction for each solution is not appropriate. Ideally the local search direction should be vertical to the non-dominated front of the offspring population. In such an ideal case, the offspring population will not be degraded (i.e., the local search moves in Fig. 5 (b) won't happen). In the remaining of this section, we examine some tricks for overcoming the above-mentioned difficulty of the weighted scalar approach.

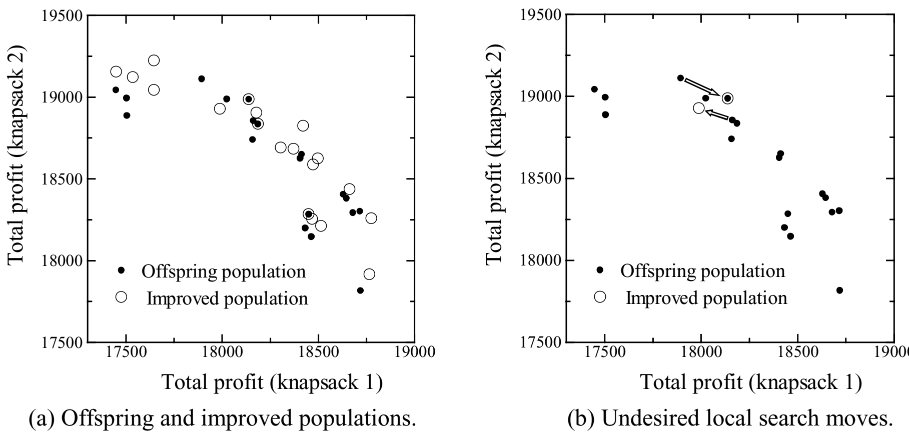
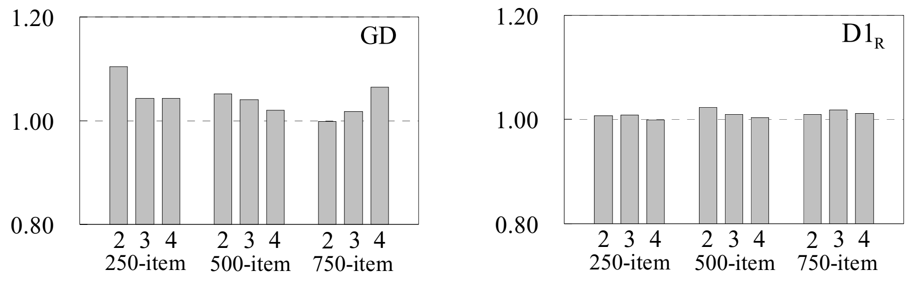


Fig. 5. Illustration of the deterioration of the offspring population by the local search based on the weighted scalar fitness function.

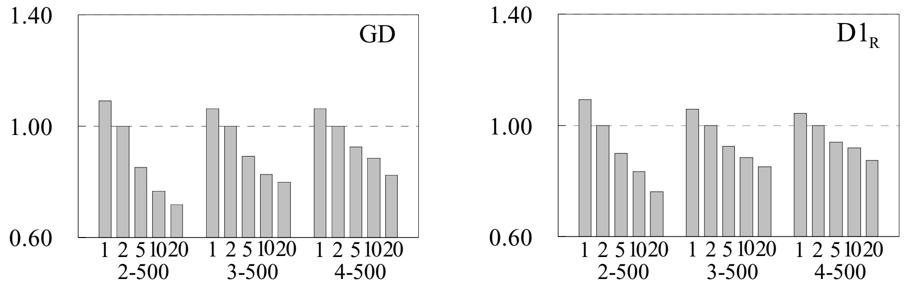
Use of three populations for generation update: A straightforward remedy for the above-mentioned difficulty is to use the offspring population as well as the improved population in the generation update procedure. For examining the effectiveness of this generation update scheme, we examined the performance of a variant of our S-MOGLS algorithm where offspring solutions were not used for generating the next population when they were updated by local search. This variant is exactly the same as the S-MOGLS algorithm of Ishibuchi & Kaige [5]. The relative performance of this variant with respect to our S-MOGLS algorithm in Section 2 is summarized in Fig. 6 in the same manner as Fig. 4. From this figure, we can see that the performance of the S-MOGLS algorithm was degraded by modifying its generation update scheme with respect to the GD measure.

Increase in the selection pressure of initial solutions: As we have already discussed, the inappropriate specification of the local search direction causes the deterioration of offspring populations by local search. Since we use the tournament selection based on the weighted scalar fitness function for choosing initial solutions for local search, the increase in the selection pressure of initial solutions (i.e., the increase in the tournament size) may lead to the selection of an appropriate initial solution for the current weight vector. That is, the tournament selection with a large tournament size is likely to choose a very good offspring solution with respect to the current weight vector. Such an offspring solution is likely to locate near the non-dominated front of the offspring population. Moreover the current weight vector is likely to be vertical to the non-dominated front around the selected initial solution.



(a) The generational distance. (b) The $D1_R$ measure.

Fig. 6. Relative performance of a variant of our S-MOGLS algorithm where intermediate offspring solutions were not used for generating the next population.



(a) The generational distance. (b) The $D1_R$ measure.

Fig. 7. Effect of the specification of the tournament size. Relative performance of each specification was calculated with respect to the case of the binary tournament selection.

We examined various specifications of the tournament size for the selection of initial solutions in our S-MOGLS algorithm. Experimental results are summarized in Fig. 7 where horizontal axis shows the tournament size (i.e., 1, 2, 5, 10, 20). In this figure, the relative performance was calculated with respect to the binary tournament selection. Thus the relative performance is always 1.00 in this figure when the tournament size is 2. From this figure, we can see that the performance of our S-MOGLS algorithm was improved by increasing the selection pressure of initial solutions for local search. When the tournament size was specified as 1, initial solutions were randomly chosen from the offspring population. In this case, the performance of our S-MOGLS algorithm was degraded by the following two reasons: One is the inappropriate specification of the local search direction for each initial solution, and the other is the selection of poor offspring solutions as initial solutions. On the other hand, good solutions together with appropriate local search directions are chosen as initial solutions for local search when the selection pressure is high. As a result, the performance of our S-MOGLS algorithm was improved by increasing the tournament size in Fig. 7.

Modification of the acceptance rule: We also examined the effectiveness of the modification of the acceptance rule illustrated in Fig. 3. We examined various specifications of the user-definable parameter a using our S-MOGLS algorithm with the binary tournament selection. The relative performance of each specification of a was calculated with respect to the case of the weighted scalar approach (i.e., $a = \infty$). Experimental results are summarized in Fig. 8 for $a = 0.5$, 1, 2, 5, ∞. In Fig. 8, we can see that the convergence to the Pareto front (i.e., the GD measure in Fig. 8 (a)) was improved by the modification of the acceptance rule from the case of $a = \infty$ to $\alpha = 0.5$, 1, 2, 5. This modification, however, may have a negative effect on the diversity of solutions because the $\mathrm{D1_R}$ measure was not improved in Fig. 8 (b).

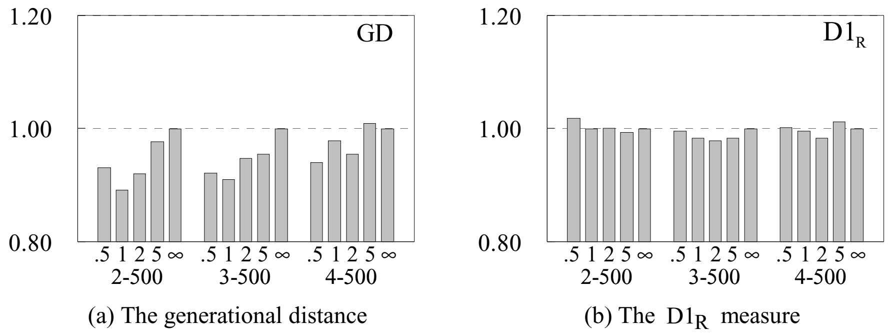


Fig. 8. Effect of the modification of the acceptance rule of the local search move. Relative performance of each specification was calculated with respect to the case of the weighted scalar approach (i.e., $a = \infty$).

5 Concluding Remarks

We proposed a new memetic EMO algorithm called S-MOGLS by combining local search with the NSGA-II [3]. Our intention is to implement a simple but powerful memetic EMO algorithm, which can be easily understood, easily implemented, and efficiently executed using small memory storage within short CPU time. Using our S-MOGLS algorithm, we examined several issues related to the implementation of local search in memetic EMO algorithms. Through computational experiments, we showed that the weighted scalar approach outperforms the Pareto ranking approach. We also showed that the weighted scalar approach has a serious difficulty: The offspring population can be degraded by local search. We implemented three remedies for this difficulty in our S-MOGLS algorithm: the use of three populations (i.e., parent, offspring and improved populations) in generation update, the choice of good initial solutions for local search, and the modification of the acceptance rule. The effectiveness of these tricks was demonstrated by our experimental results.

The authors would like to thank the financial support from Kayamori Foundation of Information Science Advancement, and Japan Society for the Promotion of Science (JSPS) through Grand-in-Aid for Scientific Research (B): KAKENHI (14380194).

References

1. Coello, C. A. C., Van Veldhuizen, D. A., and Lamont, G. B.: *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, Boston (2002).
2. Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester (2001).
3. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Trans. on Evolutionary Computation* 6 (2002) 182 – 197.
4. Ishibuchi, H., and Kaige, S.: Effects of Repair Procedures on the Performance of EMO Algorithms for Multiobjective 0/1 Knapsack Problems, *Proc. of 2003 Congress on Evolutionary Computation* (2003) 2254-2261.
5. Ishibuchi, H., and Kaige, S.: Implementation of Simple Multiobjective Memetic Algorithms and Its Application to Knapsack Problems, *International Journal of Hybrid Intelligent System* 1 (2004) 22-35.
6. Ishibuchi, H., and 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.
7. Ishibuchi, H., Yoshida, T., and 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.
8. Jaszkiewicz, A.: Comparison of Local Search-Based Metaheuristics on the Multiple Objective Knapsack Problem, *Foundations of Computing and Decision Sciences* 26 (2001) 99-120.
9. 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.
10. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization, *European Journal of Operational Research* 137 (2002) 50-71.
11. Knowles, J. D., and Corne, D. W.: M-PAES: A Memetic Algorithm for Multiobjective Optimization, *Proc. of 2000 Congress on Evolutionary Computation* (2000) 325-332.

12. Knowles, J. D., and Corne, D. W.: A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization, *Proc. of 2000 Genetic and Evolutionary Computation Conference Workshop Program: WOMA I* (2000) 103-108.
13. Murata, T., Kaige, S., and Ishibuchi, H.: Generalization of Dominance Relation-Based Replacement Rules for Memetic EMO Algorithms, *Lecture Notes in Computer Sciences* 2723 (2003) 1234-1245. *Proc. of 2003 Genetic and Evolutionary Computation Conference.*
14. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation* 3 (1999) 257-271.