Recombination of Similar Parents in EMO Algorithms

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 examines the effect of crossover operations on the performance of EMO algorithms through computational experiments on knapsack problems and flowshop scheduling problems using the NSGA-II algorithm. We focus on the relation between the performance of the NSGA-II algorithm and the similarity of recombined parent solutions. First we show the necessity of crossover operations through computational experiments with various specifications of crossover and mutation probabilities. Next we examine the relation between the performance of the NSGA-II algorithm and the similarity of recombined parent solutions. It is shown that the quality of obtained solution sets is improved by recombining similar parents. Then we examine the effect of increasing the selection pressure (i.e., increasing the tournament size) on the similarity of recombined parent solutions. An interesting observation is that the increase in the tournament size leads to the recombination of dissimilar parents, improves the diversity of solutions, and degrades the convergence performance of the NSGA-II algorithm.

1 Introduction

Since Schaffer's study [21], various evolutionary multiobjective optimization (EMO) algorithms have been proposed to find well-distributed Pareto-optimal or near Pareto-optimal solutions of multiobjective optimization problems (Coello et al. [2] and Deb [4]). Recent EMO algorithms usually share some common ideas such as elitism, fitness sharing and Pareto ranking. While mating restriction has been often discussed in the literature, it has not been used in many EMO algorithms as pointed out in some reviews on EMO algorithms [7, 23, 26]. In this paper, we examine the effect of recombining similar parents on the performance of EMO algorithms to find well-distributed Pareto-optimal or near Pareto-optimal solutions.

Mating restriction was suggested by Goldberg [8] for single-objective genetic algorithms. Hajela & Lin [9] and Fonseca & Fleming [6] used it in their EMO algorithms. The basic idea of mating restriction is to ban the recombination of dissimilar parents from which good offspring are not likely to be generated. In the implementation of mating restriction, a user-definable parameter σ_{mating} called the mating radius is usually used for banning the recombination of two parents whose distance is larger than σ_{mating}. The distance between two parents is measured in the decision space or the objective space. The necessity of mating restriction in EMO

algorithms was also stressed by Jaszkiewicz [17] and Kim et al. [18]. In the parallelization of EMO algorithms, mating restriction is implicitly realized since similar individuals are likely to be assigned to the same processor or the same island (e.g., see Branke et al. [1]). On the other hand, Zitzler & Thiele [25] reported that no improvement was achieved by mating restriction in their computational experiments. Van Veldhuizen & Lamont [23] mentioned that the empirical evidence presented in the literature could be interpreted as an argument either for or against the use of mating restriction. Moreover, there was also an argument for the selection of dissimilar parents. Horn et al. [10] argued that information from very different types of tradeoffs could be combined to yield other kinds of good tradeoffs. Schaffer [21] examined the selection of dissimilar parents but observed no improvement.

A similarity-based mating scheme was proposed in Ishibuchi & Shibata [13] to examine positive and negative effects of mating restriction on the performance of EMO algorithms. In their mating scheme, one parent (say Parent A) was chosen by the standard fitness-based binary tournament scheme while its mate (say Parent B) was chosen among a pre-specified number of candidates (say β candidates) based on their similarity or dissimilarity to Parent A. To find β candidates, the standard fitness-based binary tournament selection was iterated β times. Almost the same idea was independently proposed in Huang [11] where Parent B was chosen from two candidates (i.e., the value of β was fixed as $\beta = 2$). Ishibuchi & Shibata [14] extended their similarity-based mating scheme as shown in Fig. 1. That is, first a pre-specified number of candidates (say α candidates) were selected by iterating the standard fitness-based binary tournament selection α times. Next the average vector of those candidates was calculated in the objective space. The most dissimilar candidate to the average vector was chosen as Parent A. On the other hand, the most similar one to Parent A among β candidates was chosen as Parent B. Furthermore, it was demonstrated in [15] that the diversity-convergence balance can be dynamically adjusted by controlling the values of the two parameters α and β.

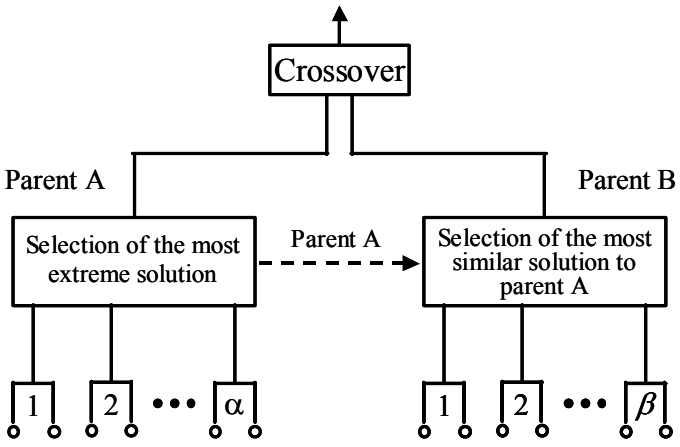


Fig. 1. Mating scheme in Ishibuchi & Shibata [14].

In this paper, we examine the effect of crossover operations on the performance of EMO algorithms through computational experiments on knapsack problems and flowshop scheduling problems using the NSGA-II algorithm of Deb et al. [5]. We focus on the relation between the performance of the NSGA-II algorithm and the similarity of recombined parents. First we show the necessity of crossover operations through computational experiments with various specifications of crossover and mutation probabilities. It is shown that crossover operations play an important role in the NSGA-II algorithm while its performance is not very sensitive to the crossover probability if compared with the mutation probability. Next we examine the relation between the performance of the NSGA-II algorithm and the similarity of recombined parents. We use the similarity-based mating scheme in Fig. 1 to choose dissimilar parents as well as similar parents. That is, the most similar or dissimilar solution to Parent A among β candidates is chosen as Parent B in our computational experiments where the value of α is fixed as $\alpha = 1$ (i.e., Parent A is selected by the standard fitness-based binary tournament selection). The similarity is measured in the decision space and the objective space. It is shown that the performance of the NSGA-II algorithm can be improved by recombining similar parents. Then we examine the effect of increasing the selection pressure (i.e., increasing the tournament size) on the similarity of recombined parents. An interesting observation is that the increase in the tournament size leads to the recombination of dissimilar parents, improves the diversity of solutions, and degrades the convergence performance of the NSGA-II algorithm on some test problems.

2 Test Problems

Multiobjective 0/1 knapsack problems with k knapsacks (i.e., k objectives and k constraints) and n items in Zitzler & Thiele [26] can be written as follows:

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

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

$$\text{where } f_i(\mathbf{x}) = \sum_{j=1}^{n} p_{ij} x_j, \quad i = 1,2, ..., k. \tag{3}$$

In this formulation, $\mathbf{x}$ is an n-dimensional binary vector (i.e., $(x_1, x_2, ..., x_n) \in \{0, 1\}^n$), 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 in EMO algorithms. The k-objective n-item knapsack problem is referred to as a k-n knapsack problem in this paper.

Zitzler & Thiele [26] examined the performance of several EMO algorithms using nine test problems with two, three, four objectives and 250, 500, 750 items. In this paper, we use the three 500-item test problems (i.e., 2-500, 3-500, and 4-500 knapsack problems) while we can only report a part of experimental results due to the

page limitation.

We also generate two-objective and three-objective 20-machine 80-job flowshop scheduling problems in the same manner as Ishibuchi et al. [12, 16]. These test problems are denoted as 2-20-80 and 3-20-80 scheduling problems, respectively. The makespan and the maximum tardiness are considered as two objectives to be minimized in the 2-20-80 scheduling problem. In addition to these two objectives, the minimization of the total flowtime is considered in the 3-20-80 scheduling problem. Each solution of these two test problems is represented as a permutation of 80 jobs.

In the case of the knapsack problems, the distance between two solutions (i.e., two binary strings) is measured by the Hamming distance in the decision space. On the other hand, the distance of two solutions (i.e., two permutations of 80 jobs) is calculated as the sum of the distance between the positions of each job. The calculation of the distance between two permutation-type strings is illustrated in Fig. 2. The distance between the positions of Job 1 (denoted by J1 in Fig. 2) is 4 since it is placed in the first position of String 1 and the fifth position of String 2. The distance between the positions of the other jobs is calculated in the same manner (i.e., 1 for Job 2, 0 for Job 3 and Job 4, and 3 for Job 5). Thus the distance between the two strings in Fig. 2 is calculated as 8 (i.e., $4+1+0+0+3$).

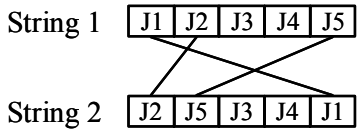


Fig. 2. Distance between two strings for five-job flowshop scheduling problems.

The distance between two solutions in the objective space is calculated by the Euclidean distance in both the knapsack problems and the scheduling problems. That is, the distance between two solutions **x** and **y** is calculated in the objective space as

$$|\mathbf{f}(\mathbf{x})-\mathbf{f}(\mathbf{y})| = \sqrt{|f_1(\mathbf{x})-f_1(\mathbf{y})|^2 + \cdots + |f_k(\mathbf{x})-f_k(\mathbf{y})|^2}\,, \tag{4}$$

where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$ is the k-dimensional objective vector corresponding to the solution **x**.

3 Performance Measures

Various performance measures have been proposed in the literature to evaluate a non-dominated solution set. As explained in Knowles & Corne [19], Okabe et al. [20], and Zitzler et al. [27], no single performance measure can simultaneously evaluate various aspects of a non-dominated solution set (e.g., convergence and diversity). Moreover, some performance measures are not designed to simultaneously compare many solution sets but to compare only two solution sets with each other. For various

performance measures, see [2, 4, 19, 20, 27].

In this paper, we use the following performance measures to simultaneously compare a number of solution sets:

1. Generational distance (GD)
2. $D1_R$ measure ($D1_R$)
3. Spread measure (Spread)
4. Hypervolume measure (Hypervolume)
5. Ratio of non-dominated solutions (Ratio)

Let S and S^* be a set of non-dominated solutions and the set of all Pareto-optimal solutions. The generational distance [22] is the average distance from each solution in S to its nearest Pareto-optimal solution in S^*. The distance is measured in the objective space using the Euclidean distance. On the other hand, the $D1_R$ measure is the average distance from each Pareto-optimal solution in S^* to its nearest solution in S. This measure was used in Czyzak & Jaszkiewicz [3]. The generational distance and the $D1_R$ measure need all Pareto-optimal solutions of each test problem. Since true Pareto-optimal solutions are not available for each test problem except for the 2-500 knapsack problem, we use as S^* a set of near Pareto-optimal solutions obtained for each test problem using much longer CPU time and much larger memory storage than computational experiments reported in this paper.

The spread measure is calculated for the solution set S as follows:

$$Spread = \sum_{i=1}^{k} [\max_{\mathbf{x} \in S} \{f_i(\mathbf{x})\} - \min_{\mathbf{x} \in S} \{f_i(\mathbf{x})\}] . \quad (5)$$

This measure is similar to the maximum spread of Zitzler [24]. The hypervolume measure [25] calculates the volume of the dominated region by the solution set S in the objective space.

The ratio of non-dominated solutions is calculated for a solution set with respect to other solution sets. We used this measure in our former studies to compare multiple non-dominate rule sets [12, 16]. This measure is similar to the coverage measure of Zitzler & Thiele [25], which was proposed to compare two non-dominated rule sets. Let us assume that we have m solution sets S_1, S_2, ..., S_m. By merging these solution sets, we construct another solution set S as $S = S_1 \cup S_2 \cup \ldots \cup S_m$. Let S_{ND} be the set of non-dominated solutions in S. The ratio of non-dominated solutions is calculated for each solution set S_i as $|S_i \cap S_{ND}| / |S_i|$ where $|S_i|$ denotes the cardinality of S_i (i.e., the number of solutions in S_i).

4 Conditions of Computational Experiments

Our computational experiments on the knapsack problems are performed using the NSGA-II algorithm under the following parameter specifications:

Crossover probability (one-point crossover): 0.8,
Mutation probability (bit-flip mutation): 1/500 (per bit),

Population size: 200 (2-500 problem), 250 (3-500 problem), 300 (4-500 problem),
Stopping condition: 500 generations.

The average value of each performance measure is calculated over 50 runs from different initial populations for each knapsack problem.

On the other hand, we use the following parameter specifications for the scheduling problems:

Crossover probability (two-point order crossover): 0.8,
Mutation probability (shift mutation): 0.5 (per string),
Population size: 200,
Stopping condition: 500 generations.

The average value of each performance measure is calculated over 20 runs with different initial populations for each scheduling problem.

It should be noted that the above-mentioned parameter values are basic settings in our computational experiments. Various specifications of the crossover and mutation probabilities are examined in the next section.

5 Effects of Crossover Operations

For examining the necessity of crossover operations in the NSGA-II algorithm, we apply it to each knapsack problem using the following specifications of the crossover and mutation probabilities:

Crossover probability (P_C) : 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,
Mutation probability (P_M) : 0.0001, 0.0002, 0.0005, 0.001, 0.002, ..., 0.1 (per bit).

In the application of the NSGA-II algorithm to each scheduling problem, the following parameter specifications are examined:

Crossover probability (P_C) : 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,
Mutation probability (P_M) : 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 (per string).

Experimental results on the 3-500 knapsack problem are shown in Fig. 3. In these experimental results, we can not observe the necessity of the crossover operation. That is, good results are obtained even in the case of no crossover (i.e., $P_C = 0$). The performance of the NSGA-II algorithm in Fig. 3 is sensitive to the mutation probability (P_M) and insensitive to the crossover probability (P_C).

While the crossover operation seems to be unnecessary in Fig. 3, its necessity is clearly shown by the average ratio of non-dominated solutions in Fig. 4 (a) where the 10×11 combinations of the crossover and mutation probabilities are compared. That is, 10×11 solution sets obtained from these combinations are compared to calculate the ratio of non-dominated solutions. From Fig. 4 (a), we can see that all solutions obtained from the NSGA-II with no crossover (i.e., $P_C = 0$) are dominated by other solutions obtained from that with crossover (i.e., $P_C > 0$). The same observation is obtained from computational experiments on the 3-20-80 scheduling problem in Fig.

4 (b). The performance of the NSGA-II algorithm for the 3-20-80 scheduling problem is not sensitive to the crossover probability as shown in Fig. 5. Similar results are obtained for all the five test problems in this paper.

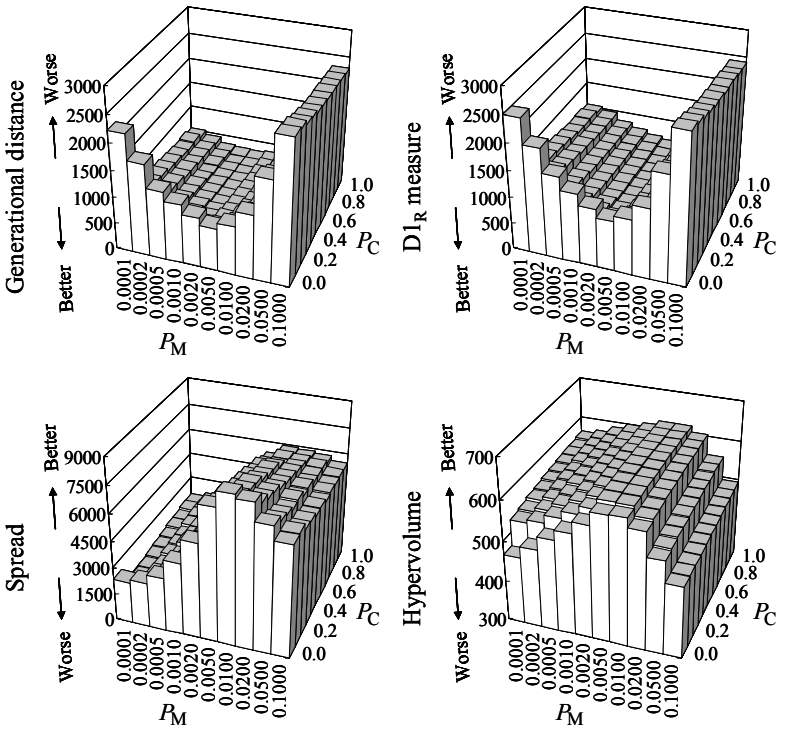
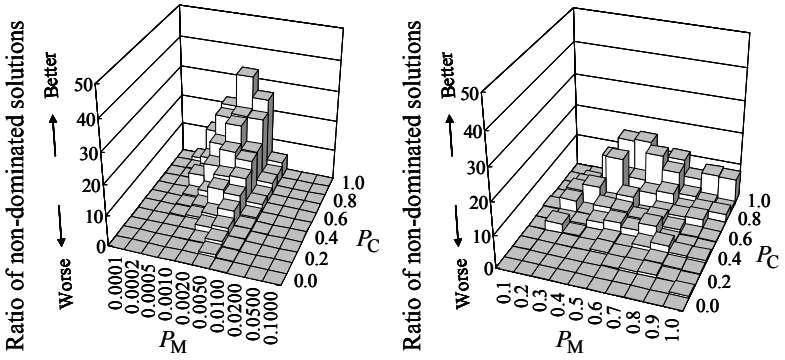


Fig. 3. Experimental results on the 3-500 knapsack problem.



(a) 3-500 knapsack problem (b) 3-20-80 scheduling problem

Fig. 4. Average ratio of non-dominated solutions.

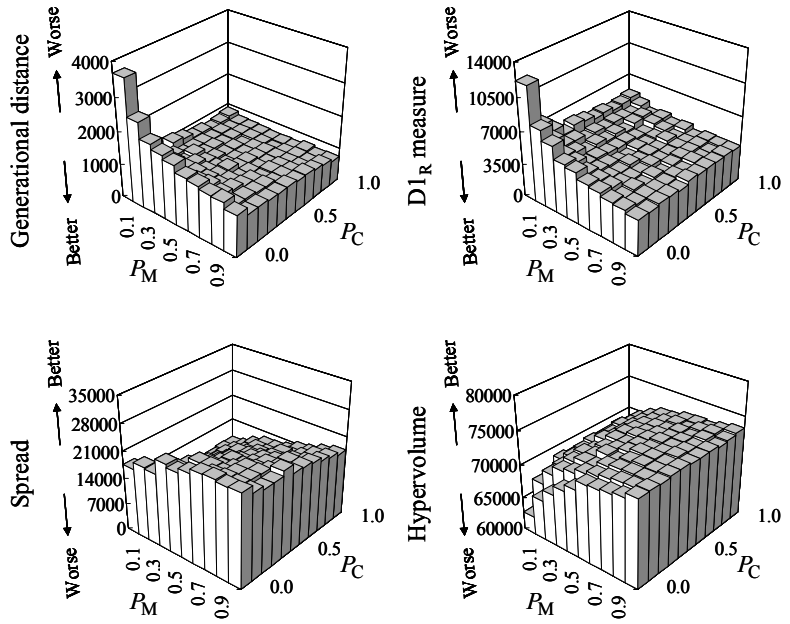
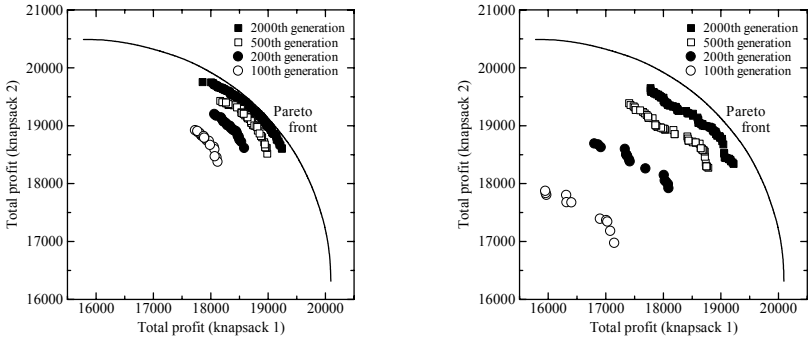


Fig. 5. Experimental results on the 3-20-80 scheduling problem.

From careful observations of Fig. 3 and Fig. 5, we can see that the use of crossover operations improved the convergence of solutions to the Pareto front and degraded the diversity of solutions. Such effects of crossover operations are visually demonstrated in Fig. 6 for the 2-500 knapsack problem. Fig. 6 shows non-dominated solutions at each generation of a single run of the NSGA-II algorithm with crossover (Fig. 6 (a)) and without crossover (Fig. 6 (b)). The mutation rate was specified as 1/500 (i.e., its basic setting) in Fig. 6.



(a) NSGA-II with crossover. (a) NSGA-II without crossover.

Fig. 6. Non-dominated solutions at each generation for the 2-500 knapsack problem.

6 Effects of Recombination of Similar Parents

We examine the effect of recombining similar parents using the mating scheme in Fig. 1. When a pair of parents is to be chosen, one parent (say Parent A) is selected by the standard fitness-based binary tournament selection in the same manner as the NSGA-II algorithm. That is, the value of α in Fig. 1 is fixed as $\alpha = 1$ in order to focus on the effect of recombining similar parents. Next we iterate the standard fitness-based binary tournament selection β times to find β candidates for the selection of the other parent (say Parent B). The most similar candidate to Parent A is chosen as Parent B (i.e., the mate of Parent A). For comparison, we also examine the choice of the most dissimilar parent to Parent A. This mating scheme is exactly the same as the standard fitness-based binary tournament selection in the NSGA-II algorithm when the values of α and β are specified as $\alpha = 1$ and $\beta = 1$. A large value of β means a strong bias toward the choice of similar (or dissimilar) parents.

We apply the NSGA-II algorithm with the mating scheme in Fig. 1 to each test problem using various values of β. Experimental results on the 3-500 knapsack problem are summarized in Fig. 7. We can see from Fig. 7 that the performance of the NSGA-II algorithm is improved by recombining similar parents. We can also see that similar results are obtained independent of the choice between the objective space and the decision space where the similarity of solutions is measured. The average distance between recombined parents is shown in Fig. 8. The improvement in the performance of the NSGA-II algorithm is also observed in computational experiments on the 3-20-80 scheduling problem in Fig. 9 where we show average results over 50 runs (Fig. 4 (b) and Fig. 5 were average results over 20 runs on the 3-20-80 scheduling problem).

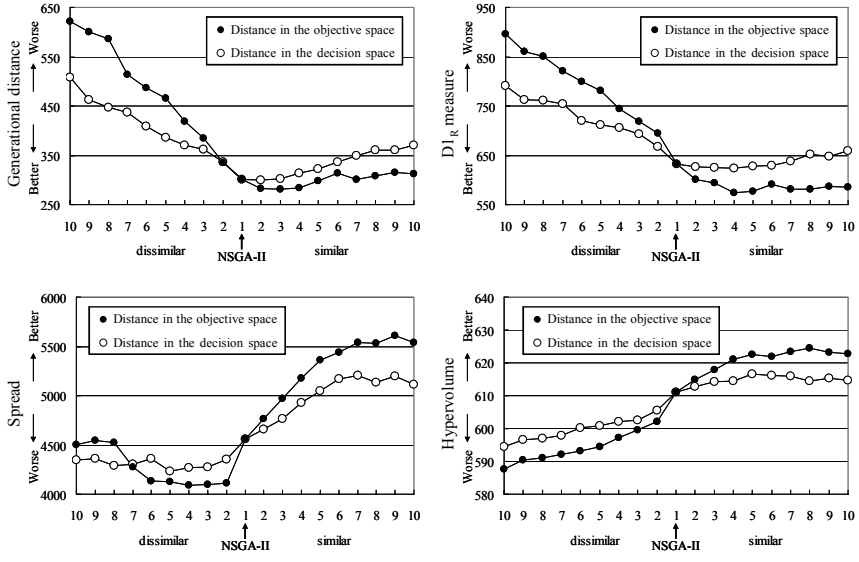
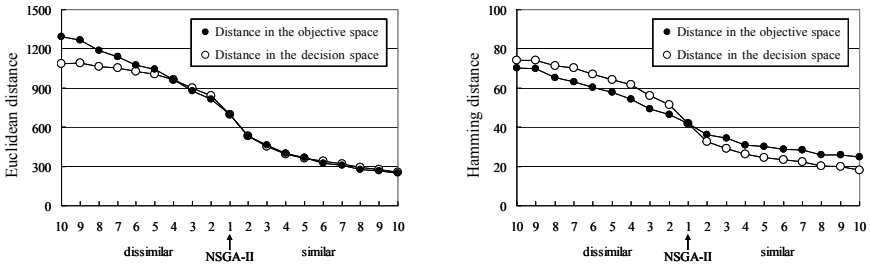


Fig. 7. Effects of recombining similar parents for the 3-500 knapsack problem.



(a) Objective space (b) Decision space

Fig. 8. Average distance between recombined parents for the 3-500 knapsack problem.

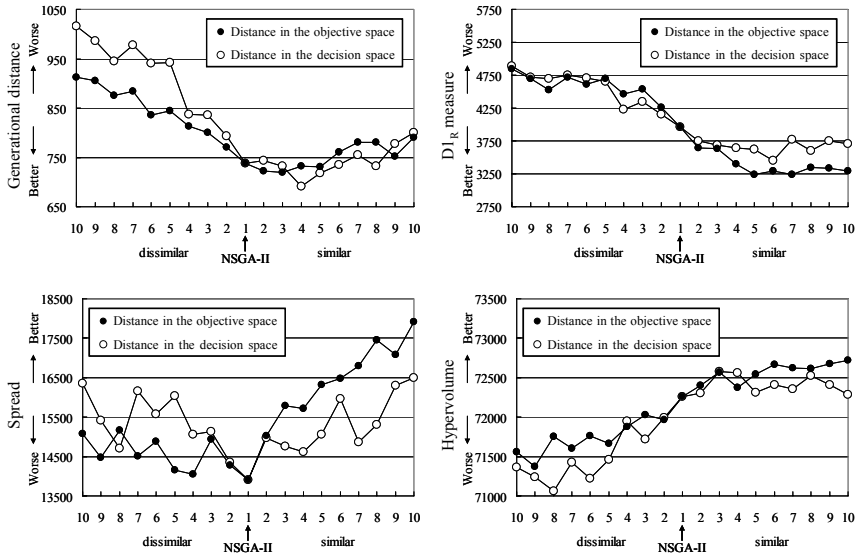
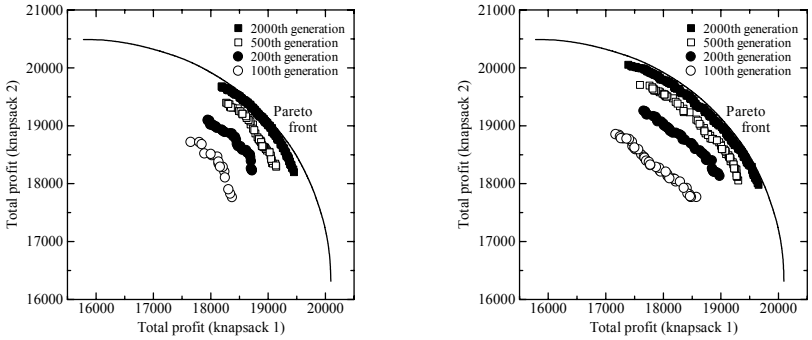


Fig. 9. Effects of recombining similar parents for the 3-20-80 scheduling problem.

We visually show the effect of choosing similar parents on the performance of the NSGA-II algorithm for the 2-500 knapsack problem in Fig. 10 (a) where β was specified as $\beta = 5$. From the comparison between Fig. 10 (a) and Fig. 6 (a) by the original NSGA-II algorithm, we can see that the recombination of similar parents improved the performance of the NSGA-II algorithm (especially with respect to the diversity of solutions). Such improvement was observed in Fig. 7 and Fig. 9. The effect of our similarity-based mating scheme in Fig. 1 becomes much clearer if we choose extreme and similar parents using the two parameters α and β. We visually show the effect of choosing extreme and similar parents in Fig. 10 (b) where the values of the two parameters α and β were specified as $\alpha = 5$ and $\beta = 5$. We can clearly see that the diversity of solutions was improved in Fig. 10 (b) from Fig. 6 (a).



(a) NSGA-II with $\alpha = 1$ and $\beta = 5$. (b) NSGA-II with $\alpha = 5$ and $\beta = 5$.

Fig. 10. Effects of our meting scheme on the performance of the NSGA-II algorithm.

7 Effects of High Selection Pressure

We examine the performance of the NSGA-II algorithm on each test problem using various specifications of the tournament size. Experimental results on the 3-500 knapsack problem are shown in Fig. 11. Each closed circle in Fig. 11 shows the average result over 50 runs for each specification of the tournament size. The standard deviation is also shown in Fig. 11 as the radius of each interval (i.e., the distance between the closed circle and each edge of the interval). It should be noted in Fig. 11 that the tournament size 1 means the random selection from the current population. Even in this case, the NSGA-II algorithm still has a somewhat strong selection pressure since the best individuals are chosen from the parent population and the offspring population in the generation update phase. We can see from Fig. 11 that the increase in the tournament size improves the diversity of solutions (i.e., the spread and hypervolume measures) and degrades the convergence of solutions to the Pareto front (i.e., the generational distance and the $D1_R$ measure). This observation may be counterintuitive for some readers because the strong selection pressure does not lead to the improvement of the convergence but the improvement of the diversity in Fig. 11. During the computational experiments in Fig. 11, we also measure the distance between recombined parents. The average distance between recombined parents is shown in Fig. 12. From this figure, we can see that the increase in the tournament size leads to the recombination of dissimilar parents. Similar results are also obtained for the 3-20-80 scheduling problem in Fig. 13 and Fig. 14. Average results over 50 runs are shown in these figures.

In Fig. 15, we visually show the effect of increasing the tournament size on the performance of the NSGA-II algorithm for the 2-500 knapsack problem. From the comparison of Fig. 15 with Fig. 6 (a) by the binary tournament selection, we can see that the increase in the tournament size (i.e., the use of a higher selection pressure) improved the diversity of solutions while it degraded the convergence of solutions to the Pareto front. Such observations were also obtained from Fig. 11 and Fig. 14.

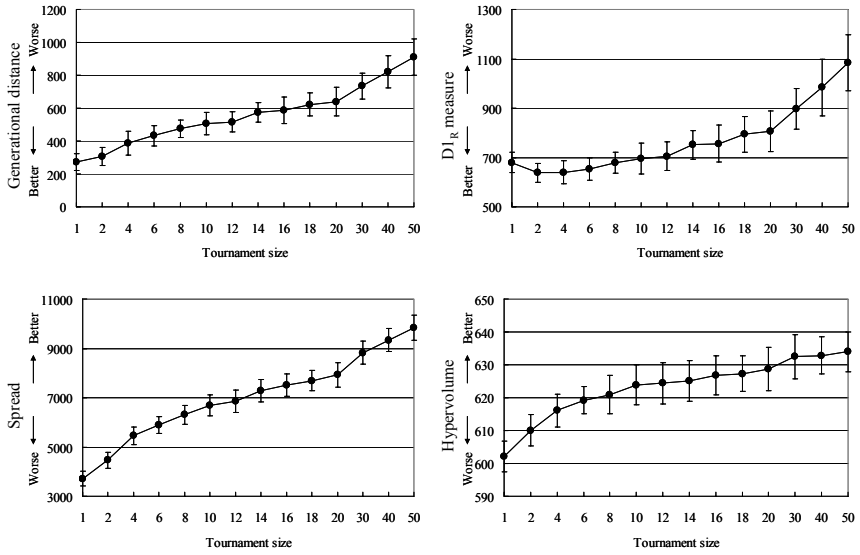
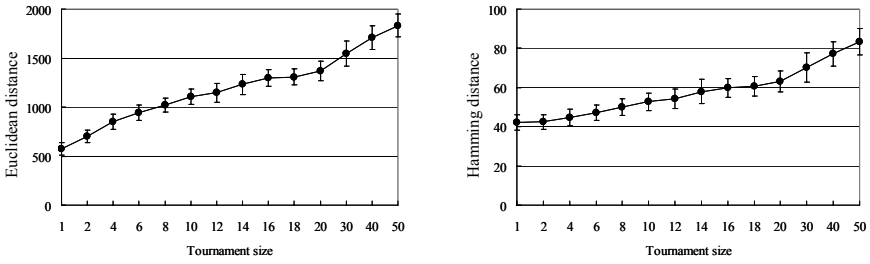
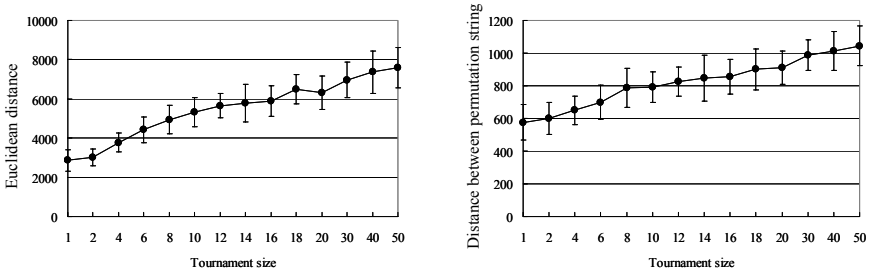


Fig. 11. Effects of the tournament size for the 3-500 knapsack problem.



(a) Objective space (b) Decision space

Fig. 12. Average distance between recombined parents for the 3-500 knapsack problem.



(a) Objective space (b) Decision space

Fig. 13. Average distance between recombined parents for the 3-20-80 scheduling problem.

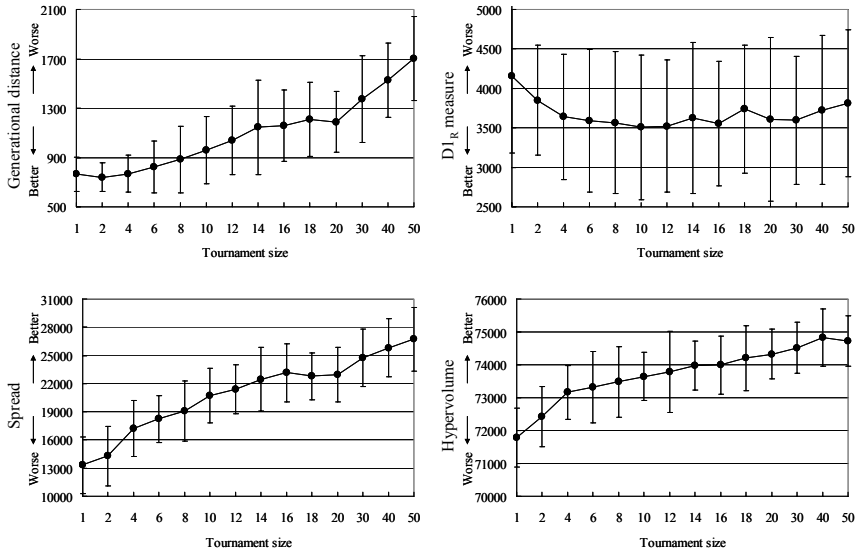
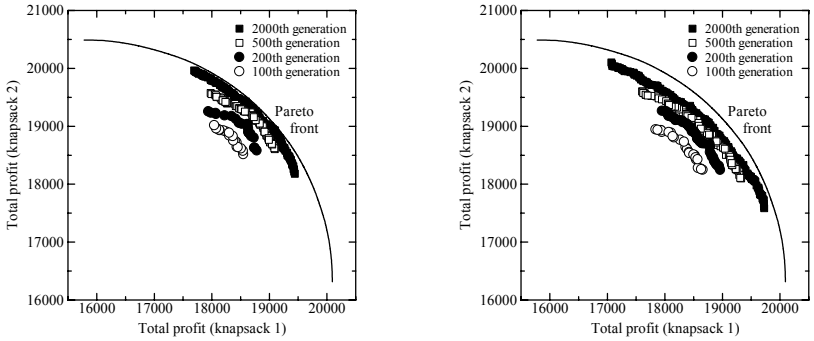


Fig. 14. Effects of the tournament size for the 3-20-80 scheduling problem.



(a) Tournament size is 10. (b) Tournament size is 50.

Fig. 15. Effects of the tournament size on the performance of the NSGA-II algorithm.

8 Concluding Remarks

Through computational experiments on multiobjective 0/1 knapsack problems and multiobjective flowshop scheduling problems, we examined the effect of crossover operations on the performance of the NSGA-II algorithm. First we showed that crossover operations played an important role while mutation operations seemed to have a larger effect on the performance of the NSGA-II algorithm. Next we empirically demonstrated that the recombination of similar parents improved the

performance of the NSGA-II algorithm. Then we examined the relation between the size of the tournament selection and the similarity of recombined parents. Experimental results showed that the use of a higher selection pressure (i.e., the increase in the tournament size) decreased the similarity of recombined solutions, improved the diversity of solutions, and degraded the convergence of solutions to the Pareto front. These effects of a higher selection pressure should be further examined since they are somewhat counterintuitive. That is, one may think that the use of a higher selection pressure may lead to the improvement in the convergence of solutions to the Pareto front and the deterioration in the diversity of solutions.

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

References

1. Branke, J., Schmeck, H., Deb, K., and Reddy, S. M.: Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation. Proc. of 2004 Congress on Evolutionary Computation (2004) 1952-1957.
2. Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Boston (2002).
3. Czyzak, P. and Jaszkiewicz, A.: Pareto-Simulated Annealing – A Metaheuristic Technique for Multi-Objective Combinatorial Optimization. Journal of Multi-Criteria Decision Analysis **7** (1998) 34-47.
4. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001).
5. 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.
6. Fonseca, C. M. and Fleming, P. J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. Proc. of 5th International Conference on Genetic Algorithms (1993) 416-423.
7. Fonseca, C. M. and Fleming, P. J.: An Overview of Evolutionary Algorithms in Multiobjective Optimization. Evolutionary Computation **3** (1995) 1-16.
8. Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989).
9. Hajela, P. and Lin, C. Y.: Genetic Search Strategies in Multicriterion Optimal Design. Structural Optimization **4** (1992) 99-107.
10. Horn, J., Nafpliotis, N., and Goldberg, D. E.: A Niched Pareto Genetic Algorithm for Multi-Objective Optimization. Proc. of 1st IEEE International Conference on Evolutionary Computation (1994) 82-87.
11. Huang, C. F.: Using an Immune System Model to Explore Mate Selection in Genetic Algorithms. Lecture Notes in Computer Science, Vol. 2723 (Proc. of GECCO 2003), Springer, Berlin (2003) 1041-1052.
12. 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.
13. Ishibuchi, H. and Shibata, Y.: An Empirical Study on the Effect of Mating Restriction on

the Search Ability of EMO Algorithms. Lecture Notes in Computer Science, Vol. 2632 (Proc. of EMO 2003), Springer, Berlin (2003) 433-447.
14. Ishibuchi, H. and Shibata, Y.: A Similarity-based Mating Scheme for Evolutionary Multiobjective Optimization. Lecture Notes in Computer Science, Vol. 2723 (Proc. of GECCO 2003), Springer, Berlin (2003) 1065-1076.
15. Ishibuchi, H. and Shibata, Y.: Mating Scheme for Controlling the Diversity-Convergence Balance for Multiobjective Optimization. Lecture Notes in Computer Science, Vol. 3102 (Proc. of GECCO 2004), Springer, Berlin (2004) 1259-1271.
16. 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.
17. 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.
18. Kim M., Hiroyasu, T., Miki, M., and Watanabe, S.: SPEA2+: Improving the Performance of the Strength Pareto Evolutionary Algorithm 2. Lecture Notes in Computer Science, Vol. 3242 (Proc. of PPSN VIII), Springer, Berlin (2004) 742-751.
19. Knowles, J. D. and Corne, D. W.: On Metrics for Comparing Non-dominated Sets. Proc. of 2002 Congress on Evolutionary Computation (2002) 711-716.
20. Okabe, T., Jin, Y., and Sendhoff, B.: A Critical Survey of Performance Indices for Multi-Objective Optimization. Proc. of 2003 Congress on Evolutionary Computation (2003) 878-885.
21. 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.
22. Van Veldhuizen, D. A.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ph. D dissertation, Air Force Institute of Technology (1999).
23. Van Veldhuizen, D. A. and Lamont, G. B.: Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. Evolutionary Computation **8** (2000) 125-147.
24. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph. D dissertation, Shaker Verlag, Aachen (1999).
25. Zitzler, E. and Thiele, L.: Multiobjective Optimization using Evolutionary Algorithms – A Comparative Case Study. Proc. of 5th International Conference on Parallel Problem Solving from Nature (1998) 292-301.
26. Zitzler, E. and Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation **3** (1999) 257-271.
27. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Trans. on Evolutionary Computation **7** (2003) 117-132.