Lamarckian Repair and Darwinian Repair in EMO Algorithms for Multiobjective 0/1 Knapsack Problems

Shiori Kaige, Kaname Narukawa, and Hisao Ishibuchi

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

Abstract. Multiobjective 0/1 knapsack problems have been frequently used as test problems to examine the performance of evolutionary multiobjective optimization algorithms in the literature. It has been reported that their performance strongly depends on the choice of a constraint handling method. In this paper, we examine two implementation schemes of greedy repair: Lamarckian and Darwinian. In the Lamarckian implementation of greedy repair, a feasible solution is generated from an unfeasible one by removing items until all the constraint conditions are satisfied. That is, the genetic information of the unfeasible solution is modified. On the other hand, the genetic information of the unfeasible solution is not changed in the Darwinian implementation where greedy repair is used only to evaluate the fitness value of each solution. We compare these two implementation schemes with each other through computational experiments. We also compare greedy repair-based methods with a penalty function approach.

1 Introduction

Evolutionary multiobjective optimization (EMO) is a very active research area in the field of evolutionary computation (see, for example, Coello et al. [1] and Deb [2]). Since the study of Zitzler & Thiele [18], multiobjective 0/1 knapsack problems have been frequently used in computational experiments to examine the performance of various EMO algorithms (e.g., Ishibuchi et al. [6], [7], Jaszkiewicz [9], [10], Knowles & Corne [12], [13], and Zitzler et al. [17]). When EMO algorithms are applied to multiobjective 0/1 knapsack problems, unfeasible solutions are often generated by genetic operations. That is, generated solutions do not always satisfy the constraint conditions. Thus several constraint handling methods have been examined in the application of EMO algorithms to multiobjective 0/1 knapsack problems (e.g., Ishibuchi & Kaige [4], Mumford [15], and Zydallis & Lamont [20]). Constraint handling methods can be roughly classified into the following three categories:

Greedy Repair: An unfeasible solution is repaired by removing items until all the constraint conditions are satisfied. The order in which items are removed is pre-

specified based on a heuristic evaluation measure.
Penalty Function: The objective function related to each knapsack is penalized when the constraint condition with respect to that knapsack is violated.
Permutation Cording: Each solution is not represented by a binary string but a permutation of items. That is, the order of items is used as a string to represent each solution. A feasible solution is obtained from each permutation-type string by adding items to the knapsacks in the order specified by that string.

In this paper, we concentrate on the comparison between two implementation schemes of greedy repair: Lamarckian and Darwinian. In the Lamarckian implementation, a feasible solution is generated from an unfeasible one by removing items until all the constraint conditions are satisfied. That is, the genetic information of the unfeasible solution is modified by greedy repair as shown in Fig. 1 where the fifth and sixth items are removed from the unfeasible solution. Fig. 1 illustrates how a newly generated infeasible solution is repaired before it is inserted in the next population. As a result, the current population always consists of feasible solutions.

On the other hand, the genetic information of an unfeasible solution is not changed in the Darwinian implementation where greedy repair is used only to evaluate the fitness value of each solution. As shown in Fig. 2, the same feasible solution as in Fig. 1 is generated from the unfeasible solution by greedy repair. This feasible solution is used only to assign the fitness value to the unfeasible solution. As a result, the current population becomes a mixture of feasible and unfeasible solutions in the case of the Darwinian implementation of greedy repair.

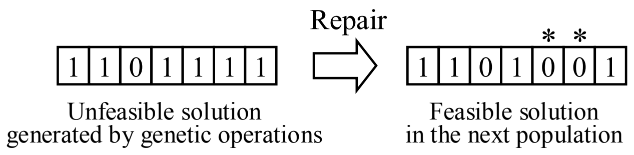


Fig. 1. Illustration of the Lamarckian implementation of greedy repair.

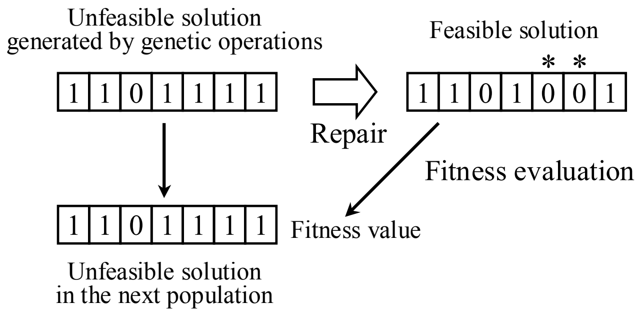


Fig. 2. Illustration of the Darwinian implementation of greedy repair.

In this paper, we first briefly explain multiobjective 0/1 knapsack problems and two repair methods examined in Ishibuchi & Kaige [4] and Zydallis & Lamont [20]. The two repair methods are the maximum ratio repair and the weighted scalar repair. Then we examine the two implementation schemes (i.e., Lamarckian and Darwinian) of these repair methods. The two implementation schemes are compared with each other through computational experiments on multiobjective 0/1 knapsack problems in Zitzler & Thiele [18] using the NSGA-II algorithm of Deb et al. [3]. We also evaluate the performance of the repair methods in comparison with a penalty function approach where the objective function with respect to each knapsack is penalized when the corresponding constraint condition is violated.

2 Multiobjective 0/1 Knapsack Problems

Multiobjective 0/1 knapsack problems with k knapsacks (i.e., k objectives) and n items in Zitzler & Thiele [18] can be written as follows:

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

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

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.

3 Repair Methods

Zitzler & Thiele [18] used a greedy repair method where items were removed in the ascending order of the maximum profit/weight ratio q_j over all knapsacks:

$$q_j = \max\{p_{ij} / w_{ij} \mid i = 1,2,...,k\}\,,\ j = 1,2,...,n\,. \tag{4}$$

The maximum profit/weight ratio q_j in (4) has been used in many studies on EMO algorithms [6], [7], [12], [13], [17], [18]. In this paper, we refer to this repair method as ***maximum ratio repair***.

While Pareto ranking was used to evaluate each solution in many EMO algorithms, the following weighted scalar fitness function was used in some EMO algorithms (e.g., Ishibuchi et al. [5], [8] and Jaszkiewicz [9]-[11]):

$$f(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{k} \lambda_i f_i(\mathbf{x}), \tag{5}$$

where

$$\forall i \;\; \lambda_i \geq 0 \;\; \text{and} \;\; \sum_{i=1}^{k} \lambda_i = 1. \tag{6}$$

In the MOGLS of Jaszkiewicz [11], the weighted scalar fitness function in (5) was used in the following manner. When a pair of parent solutions is to be selected, first the weight vector $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_k)$ is randomly specified. Next the best K solutions are selected from the current population using the weighted scalar fitness function with the current weight vector. Then a pair of parent solutions is randomly chosen from those K solutions in order to generate an offspring by genetic operations from the selected pair. The same weighted scalar fitness function with the current weight vector is used in the repair for the generated offspring where items were removed in the ascending order of the following ratio:

$$q_j = \sum_{i=1}^{k} \lambda_i p_{ij} \Bigg/ \sum_{i=1}^{k} w_{ij}, \quad j = 1,2,...,n. \tag{7}$$

We refer to this repair method as ***weighted scalar repair***. A local search procedure is applied to the repaired offspring using the same scalar fitness function with the current weight vector. The weighted scalar repair is also used in the local search phase.

It should be noted that the weighted scalar repair is directly applicable only to the MOGLS with the weighted scalar fitness function. In the application to the NSGA-II algorithm [3] in this paper, we use the weighted scalar repair by randomly updating the weight vector $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_k)$ for each unfeasible solution. That is, a different weight vector is assigned to each unfeasible solution.

Each repair method is implemented in the two frameworks: Lamarckian and Darwinian. This means that we examine the four combinations of the two repair methods and the two implementation schemes.

Just for comparison, we also examine the performance of a penalty function approach. Using a positive constant α representing a unit penalty with respect to the violation of each constraint condition, we formulate the following k-objective optimization problem with no constraint conditions from the original k-objective 0/1 knapsack problem in (1)-(3):

$$\text{Maximize } \mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), ..., g_k(\mathbf{x})), \tag{8}$$

where

$$g_i(\mathbf{x}) = f_i(\mathbf{x}) - \alpha \cdot \max\left(0, \sum_{j=1}^{n} w_{ij} x_j - c_i\right), \quad i = 1, 2, ..., k. \tag{9}$$

In this formulation, each objective $f_i(\mathbf{x})$ is penalized when the corresponding constraint condition is violated. The application of EMO algorithms to the *k*-objective optimization problem in (8)-(9) is straightforward because no constraint conditions are involved. When we evaluate the performance of an EMO algorithm with the penalty function method, we only examine feasible solutions in the final solution set obtained by each run of the EMO algorithm (i.e., unfeasible solutions of the original knapsack problem are not taken into account in the performance evaluation).

4 Computational Experiments

4.1 Conditions of Computational Experiments

We incorporated each of the four combinations of the two repair methods and the two implementation schemes into the NSGA-II algorithm [3]. As test problems, we used three knapsack problems in Zitzler & Thiele [18]: 2-250 (i.e., two-objective 250-item), 2-500, and 2-750 problems. Each solution for an *n*-item problem was coded as a binary string of length *n*. The four variants of the NSGA-II algorithm were applied to the three knapsack problems under the following parameter specifications:

Crossover probability (one-point crossover): 0.8,
Mutation probability (bit-flip mutation): $4/n$,
Population size: 150 (2-250 problem), 200 (2-500 problem), 250 (2-750 problem),
Stopping condition: 500 generations.

The average performance of each variant on each test problem was calculated over 30 runs with different initial populations.

A number of performance measures have been proposed for evaluating a set of non-dominated solutions in the literature. As explained in Knowles & Corne [14], Okabe et al. [16], and Zitzler et al. [19], no performance measure can simultaneously evaluate various aspects of a solution set. In this paper, we use three performance measures that are applicable to simultaneous comparison of many solution sets. Let *S* be a solution set obtained by an EMO algorithm. A simple performance measure to evaluate the diversity of solutions in the solution set *S* is the width of the solution set *S* in the objective space. The width of the solution set *S* is measured for each objective $f_i(\mathbf{x})$ as

$$width_i(S) = \max\{f_i(\mathbf{x}) \mid \mathbf{x} \in S\} - \min\{f_i(\mathbf{x}) \mid \mathbf{x} \in S\} . \qquad (10)$$

The sum of the widths over the *k* objectives is calculated as

$$width(S) = \sum_{i=1}^{k} width_i(S) . \qquad (11)$$

On the other hand, the proximity of the solution set *S* to the Pareto front is evaluated by the generational distance defined as follows:

$$\mathrm{GD}(S) = \frac{1}{|S|} \sum_{\mathbf{x} \in S} \min\{d_{\mathbf{xy}} \mid \mathbf{y} \in S^*\} , \tag{12}$$

where S^* is a reference solution set (i.e., the set of Pareto-optimal solutions) and $d_{\mathbf{xy}}$ is the distance between a solution $\mathbf{x}$ and a reference solution $\mathbf{y}$ in the k-dimensional objective space:

$$d_{\mathbf{xy}} = \sqrt{(f_1(\mathbf{x}) - f_1(\mathbf{y}))^2 + \cdots + (f_k(\mathbf{x}) - f_k(\mathbf{y}))^2} . \tag{13}$$

For evaluating both the diversity of solutions in the solution set S and the convergence speed to the Pareto front, we calculate the $\mathrm{D1_R}$ measure defined as follows:

$$\mathrm{D1_R}(S) = \frac{1}{|S^*|} \sum_{\mathbf{y} \in S^*} \min\{d_{\mathbf{xy}} \mid \mathbf{x} \in S\} . \tag{14}$$

It should be noted that $\mathrm{D1_R}(S)$ in (14) is the average distance from each reference solution $\mathbf{y}$ in S^* to its nearest solution in S while $\mathrm{GD}(S)$ in (12) is the average distance from each solution $\mathbf{x}$ in S to its nearest reference solution in S^* . The generational distance $\mathrm{GD}(S)$ measures the proximity of the solution set S to the reference solution set S^* . On the other hand, $\mathrm{D1_R}$ evaluates how well the solution set S approximates the reference solution set S^* .

Since all Pareto-optimal solutions are known for the 2-250 and 2-500 test problems, we can use them as the reference solution set S^* for each test problem. On the other hand, the performance of the four variants of the NSGA-II algorithm on the 2-750 test problem is visually examined in its two-dimensional objective space.

4.2 Comparison of Two Implementation Schemes: Lamarckian and Darwinian

Average results over 30 runs are summarized in Table 1 for the maximum ratio repair and Table 2 for the weighted scalar repair. In these tables, the average value of each performance measure is shown together with the corresponding standard deviation in the parentheses. The better result between the two implementation schemes (i.e., Lamarckian and Darwinian) is shown by boldface. It should be noted that smaller values of the generational distance and the $\mathrm{D1_R}$ measure mean better results while larger values of the width measure mean better results.

From Table 1 and Table 2, we can see that the Darwinian implementation consistently outperforms the Lamarckian implementation independent of the setting of the other factors (i.e., in all combinations of the two repair methods, the two test problems, and the three performance measures). We can also see from the comparison between Table 1 and Table 2 that the weighted scalar repair consistently outperforms the maximum ratio repair in all combinations of the two implementation schemes, the two test problems, and the three performance measures.

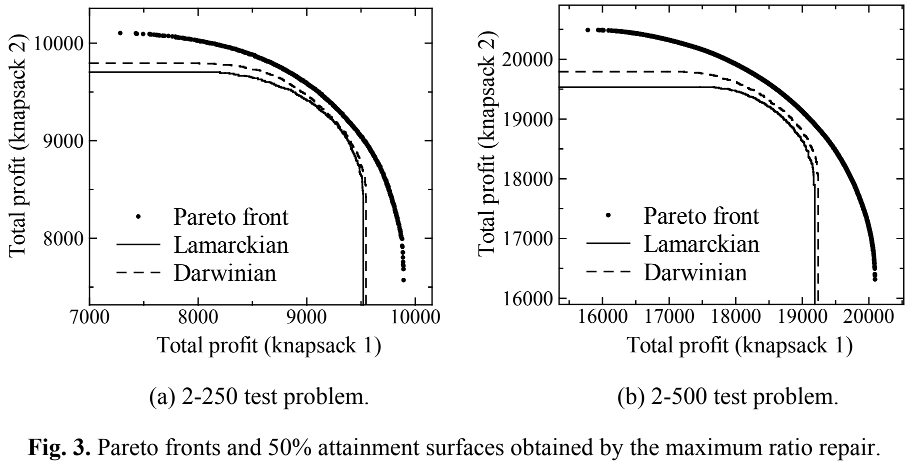
Table 1. Average results using the maximum ratio repair.

Problem	Generational distance		$D1_R$ measure		Width	
	Lamarckian	Darwinian	Lamarckian	Darwinian	Lamarckian	Darwinian
2-250	140 (15)	**105** (17)	262 (27)	**212** (27)	2173 (233)	**2414** (289)
2-500	324 (32)	**239** (24)	632 (42)	**502** (47)	2758 (322)	**3324** (371)

Table 2. Average results using the weighted scalar repair.

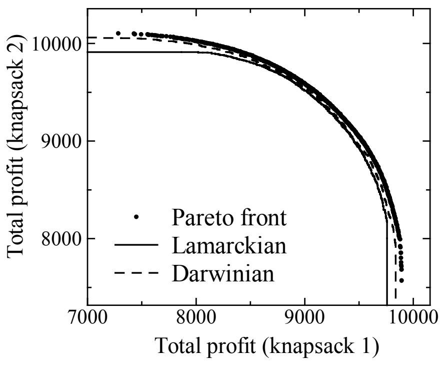
Problem	Generational distance		$D1_R$ measure		Width	
	Lamarckian	Darwinian	Lamarckian	Darwinian	Lamarckian	Darwinian
2-250	57 (6)	**36** (5)	94 (15)	**40** (5)	3336 (273)	**4518** (186)
2-500	155 (17)	**100** (19)	269 (26)	**103** (18)	4717 (326)	**7114** (228)

In order to visually compare the two implementation schemes, we calculated the 50% attainment surface using 30 solution sets obtained from 30 independent runs for each case. Experimental results are shown in Fig. 3 for the maximum ratio repair and Fig. 4 for the weighted scalar repair. In these figures, all the Pareto-optimal solutions (i.e., Pareto front) of each test problem are also shown. These Pareto-optimal solutions were used to calculate the generational distance and the $D1_R$ measure in Table 1 and Table 2. Fig. 3 and Fig. 4 visually show the superiority of the Darwinian implementation over the Lamarckian implementation.

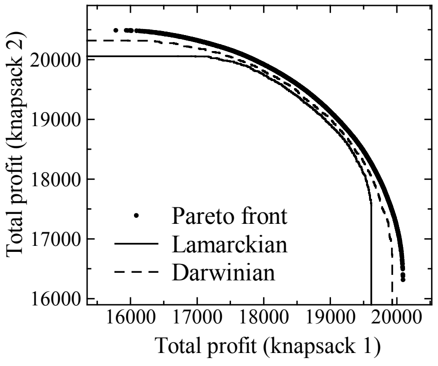


(a) 2-250 test problem.
(b) 2-500 test problem.

Fig. 3. Pareto fronts and 50% attainment surfaces obtained by the maximum ratio repair.

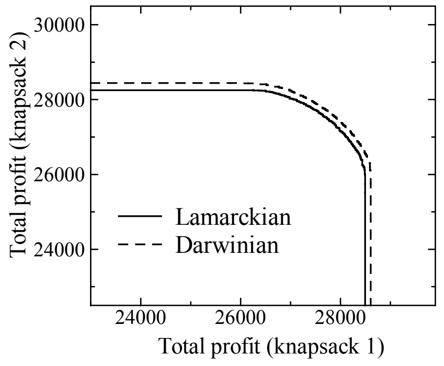


(a) 2-250 test problem.

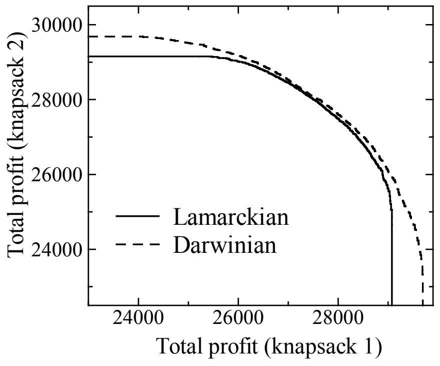


(b) 2-500 test problem.

Fig. 4. Pareto fronts and 50% attainment surfaces obtained by the weighted scalar repair.



(a) Maximum ratio repair.



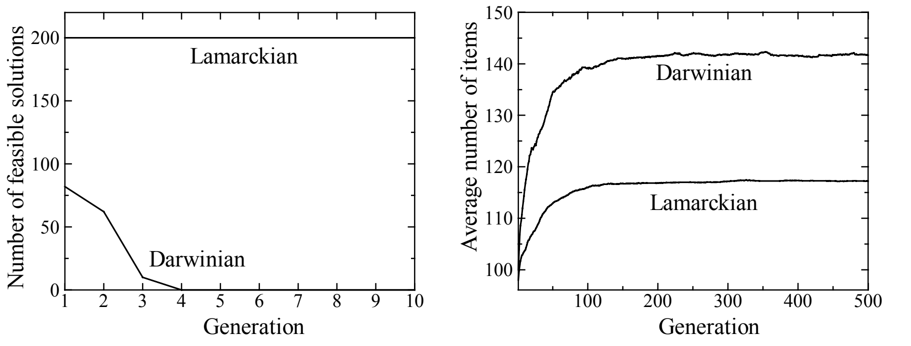
(b) Weighted scalar repair.

Fig. 5. 50% attainment surfaces obtained for the 2-750 test problem.

We also calculated the 50% attainment surface for the 2-750 test problem. Experimental results are summarized in Fig. 5. As in the above-mentioned experimental results on the 2-250 and 2-500 test problems, better results were obtained by the Darwinian implementation than the Lamarckian implementation. We can also see that the weighted scalar repair outperforms the maximum ratio repair in Fig. 5 as in Fig. 3 and Fig. 4.

In order to further demonstrate the difference between the two implementation schemes, we monitored the number of feasible solutions in each generation during the execution of the four variants of the NSGA-II algorithm on the three test problems. In Fig. 6 (a), we show experimental results of a single run on the 2-500 test problem using the maximum ratio repair. As we have already explained, all the solutions at each generation were always feasible in the case of the Lamarckian implementation.

On the contrary, the number of feasible solutions rapidly decreased to zero in the early stage of evolution in the case of the Darwinian implementation in Fig. 6 (a). We also monitored the average number of items included in each solution during the same execution as in Fig. 6 (a). Experimental results are summarized in Fig. 6 (b). From Fig. 6 (b), we can see that more items were included in each solution in the case of the Darwinian implementation than the Lamarckian implementation. We can also see that the increase in the average number of items was very slow after the 100th generation even in the case of the Darwinian implementation where all solutions were infeasible after the 4th generation.

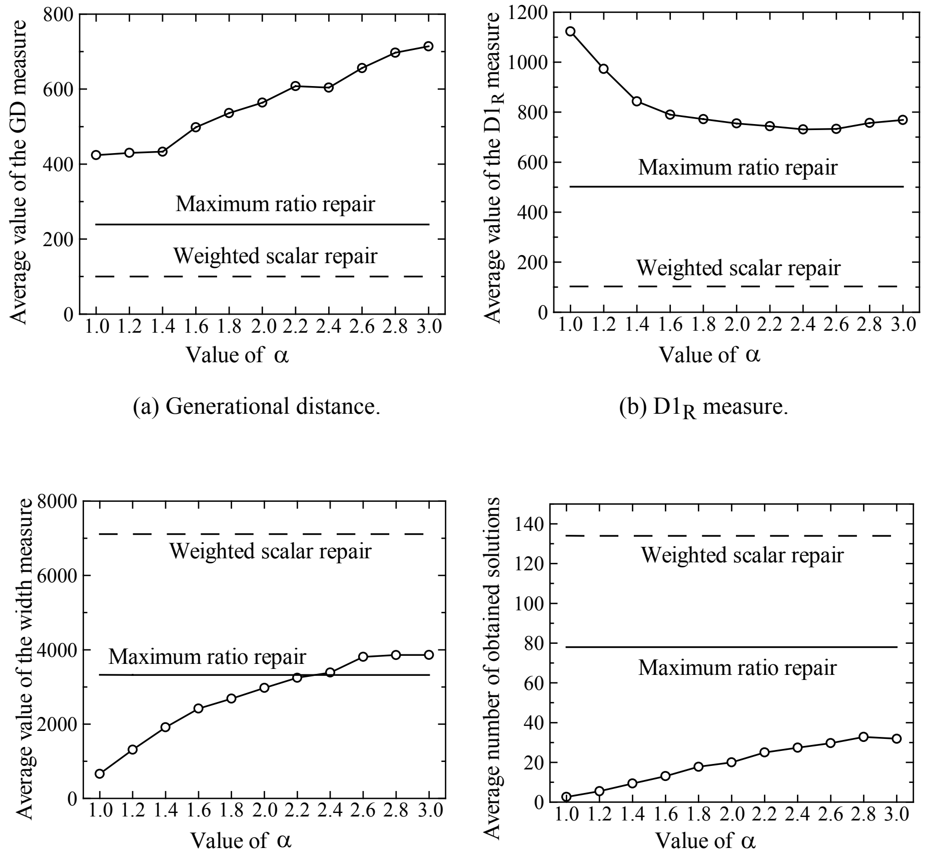


(a) Number of feasible solutions. (b) Average number of items in each solution.

Fig. 6. Experimental results of a single run on the 2-500 test problem.

4.3 Comparison with Penalty Function Approach

In the same manner as in Subsection 4.2, we applied the NSGA-II algorithm to each test problem 30 times using the penalty function approach in (8)-(9). We examined the various specifications of the unit penalty α : $\alpha = 1.0, 1.2, 1.4, ..., 3.0$. Average results over 30 runs on the 2-500 test problem are depicted by open circles in Fig. 7 where we also show the average results by the Darwinian implementation of the two repair methods in Subsection 4.2. Fig. 7 (a)-(c) show the average results with respect to the three performance measures: the generational distance, the $D1_R$ measure, and the width measure, respectively. We also show the average number of obtained solutions in Fig. 7 (d). It should be noted that smaller values mean better results in Fig. 7 (a) and Fig. 7 (b) while larger values mean better results in Fig. 7 (c) and Fig. 7 (d). From Fig. 7, we can see that good results were not obtained by the penalty function approach while we examined a wide range of parameter values. More specifically, we can see from Fig. 7 (a) that the convergence to the Pareto front degraded as the unit penalty α increased. On the other hand, we can see from Fig. 7 (b)-(d) that the diversity of obtained solutions degraded as the unit penalty α decreased. When the unit penalty α was very small (e.g., $\alpha = 1.0$), only a few solutions were obtained from each run using the penalty function approach as shown in Fig. 7 (d).



(a) Generational distance.

(b) $D1_R$ measure.

(c) Width measure.

(d) Number of obtained solutions.

Fig. 7. Average results by the penalty function approach on the 2-500 test problem.

5 Concluding Remarks

We examined two implementation schemes (i.e., Lamarckian and Darwinian) of greedy repair through computational experiments on multiobjective 0/1 knapsack problems. We also examined two repair methods: maximum ratio repair and weighted scalar repair. While the Lamarckian implementation has been mainly used in the application of EMO algorithms to multiobjective 0/1 knapsack problems in the literature, better results were obtained by the Darwinian implementation in this paper.

The main contribution of this paper is that the superiority of the Darwinian implementation over the Lamarckian implementation was clearly demonstrated through computational experiments on multiobjective 0/1 knapsack problems.

We also showed that better results were obtained by the weighted scalar repair than the maximum ratio repair. This observation is consistent with some reported results in the literature. Finally we demonstrated that good results were not obtained by the penalty function approach in comparison with greedy repair. This observation is also consistent with some reported results in the literature.

While we empirically showed the superiority of the Darwinian implementation over the Lamarckian implementation through computational experiments on multiobjective 0/1 knapsack problems, we did not explain why the Darwinian implementation outperformed the Lamarckian implementation in the application of the NSGA-II algorithm to the three test problems in this paper. We did not examine the performance of the two implementation schemes for other test problems, either. Further empirical studies as well as theoretical studies are left for future research with respect to the comparison between the two implementation schemes of greedy repair for multiobjective optimization problems.

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

References

1. Coello Coello, C. A., 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 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.
6. 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* 2632 (2003) 433-447. *Proc. of Second International Conference on Evolutionary Multi-Criterion Optimization*.
7. Ishibuchi, H., and Shibata, Y.: A Similarity-Based Mating Scheme for Evolutionary Multiobjective Optimization, *Lecture Notes in Computer Sciences* 2723 (2003) 1065-1076. *Proc. of 2003 Genetic and Evolutionary Computation Conference.*
8. 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.

9. Jaszkiewicz, A.: Comparison of Local Search-Based Metaheuristics on the Multiple Objective Knapsack Problem, *Foundations of Computing and Decision Sciences* 26 (2001) 99-120.
10. 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.
11. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization, *European Journal of Operational Research* 137 (2002) 50-71.
12. 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.
13. 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.
14. Knowles, J. D., and Corne, D. W.: On Metrics for Comparing Non-Dominated Sets, *Proc. of 2002 Congress on Evolutionary Computation* (2002) 711-716.
15. Mumford, C. L.: Comparing Representations and Recombination Operators for the Multi-Objective 0/1 Knapsack Problem, *Proc. of 2003 Congress on Evolutionary Computation* (2003) 854-861.
16. 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.
17. Zitzler, E., Laumanns, M., and Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm, *TIK-Report* 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (2001).
18. Zitzler, E., and Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Trans. on Evolutionary Computation* 3 (1999) 257-271.
19. 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.
20. Zydallis, J. B., and Lamont, G. B.: Explicit Building-Block Multiobjective Evolutionary Algorithms for NPC Problems, *Proc. of 2003 Congress on Evolutionary Computation* (2003) 2685-2695.