Comparison Between Lamarckian and Baldwinian Repair on Multiobjective 0/1 Knapsack Problems

Hisao Ishibuchi, Shiori Kaige, and Kaname Narukawa

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

Abstract. This paper examines two repair schemes (i.e., Lamarckian and Baldwinian) through computational experiments on multiobjective 0/1 knapsack problems. First we compare Lamarckian and Baldwinian with each other. Experimental results show that the Baldwinian repair outperforms the Lamarckian repair. It is also shown that these repair schemes outperform a penalty function approach. Then we examine partial Lamarckianism where the Lamarckian repair is applied to each individual with a prespecified probability. Experimental results show that a so-called 5% rule works well. Finally partial Lamarckianism is compared with an island model with two subpopulations where each island has a different repair scheme. Experimental results show that the island model slightly outperforms the standard single-population model with the 50% partial Lamarckian repair in terms of the diversity of solutions.

1 Introduction

Since 1990s, multiobjective 0/1 knapsack problems have been frequently used to evaluate the performance of various multiobjective metaheuristics including evolutionary multiobjective optimization (EMO) algorithms [4-7, 11, 16, 18]. 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 [11], and Zydallis & Lamont [18]). Constraint handling methods for multiobjective 0/1 knapsack problems 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 measure for evaluating each item.

Penalty Function: Objective functions are penalized when constraint conditions are violated.

Permutation Coding: 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.

C. A. Coello Coello et al. (Eds.): EMO 2005, LNCS 3410, pp. 370 – 385, 2005.

In this paper, we concentrate on the comparison between two implementation schemes of greedy repair: Lamarckian and Baldwinian. 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. As a result, each population includes no unfeasible solutions. Since Zitzler & Thiele [16], the Lamarckian implementation has been implicitly used in almost all computational experiments of EMO algorithms with greedy repair on multiobjective 0/1 knapsack problems [4-7, 11, 18].

On the other hand, the genetic information of an unfeasible solution is not changed in the Baldwinian 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, each population becomes a mixture of feasible and unfeasible solutions.

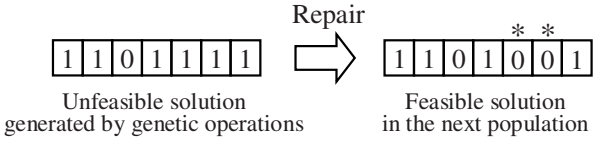


Fig. 1. Illustration of the Lamarckian implementation of greedy repair

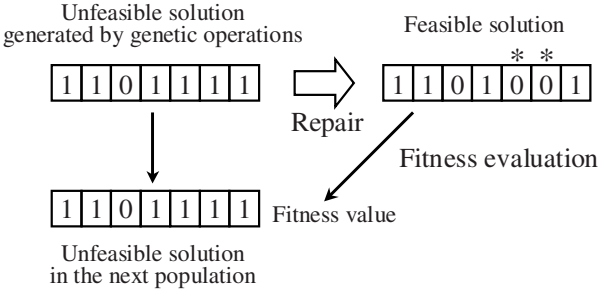


Fig. 2. Illustration of the Baldwinian implementation of greedy repair

In this paper, we first briefly explain multiobjective 0/1 knapsack problems and two repair methods examined in Ishibuchi & Kaige [4] and Zydallis & Lamont [18]. The two repair methods are the maximum ratio repair and the weighted scalar repair. Next we examine the two implementation schemes (i.e., Lamarckian and Baldwinian) 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 [16] using the NSGA-II algorithm of Deb et al. [2]. While better results can be obtained by memetic EMO algorithms (e.g., MOGLS [6]) for

multiobjective 0/1 knapsack problems, we use the NSGA-II algorithm because it seems to be the most frequently used EMO algorithm in the literature. Experimental results show that the Baldwinian repair outperforms the Lamarckian repair as in many other studies on single-objective combinatorial optimization problems (e.g., Liu et al. [9] and Orvosh & Davis [12,13]). We also evaluate the performance of the repair methods in comparison with a penalty function approach where objective functions are penalized when constraint conditions are violated. Then we examine the performance of partial Lamarckianism where the Lamarckian repair is applied to each unfeasible solution with a prespecified probability. When the Lamarckian repair is not applied, the unfeasible solution is handled by the Baldwinian repair.

Partial Lamarckianism has been examined in some studies on single-objective optimization problems. For example, Orvosh & Davis [12,13] found that good results were obtained for combinatorial optimization problems by the application of the Lamarckian repair to each unfeasible solution with a 5% probability while Michalewicz & Nazhiyath [10] used a 20% probability for continuous optimization problems. On the other hand, Houck et al. [3] showed that good results were obtained from 20% and 40% partial Lamarckianism search strategies on a number of test problems. Our experimental results show that a so-called 5% rule [12,13] works well on multiobjective 0/1 knapsack problems. Finally partial Lamarckianism is compared with an island model with two subpopulations where each island has a different repair scheme (i.e., Lamarckian or Baldwinian). Experimental results show that the island model slightly outperforms the standard single-population model with the 50% partial Lamarckian repair in terms of the diversity of solutions.

2 Multiobjective 0/1 Knapsack Problems

The following k-objective 0/1 knapsack problem with k knapsacks and n items was used in Zitzler & Thiele [16] where an objective function as well as a constraint condition was related to each knapsack:

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

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

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 by a binary string of length n in our computational experiments. As a test problem, we use a two-objective 500-item (i.e., 2-500) knapsack problem in [16].

3 Repair Methods

Zitzler & Thiele [16] used a greedy repair method where items were removed in the ascending order of the maximum profit/weight ratio q_j over all knapsacks:

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

The maximum profit/weight ratio q_j in (4) has been used in many studies on EMO algorithms (e.g., Knowles & Corne [7]). 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 (e.g., Deb et al. [2] and Zitzler & Thiele [16]), the following weighted scalar fitness function was used in some EMO algorithms (e.g., Jaszkiewicz [5, 6]):

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

where

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

In a multiobjective genetic local search (MOGLS) algorithm of Jaszkiewicz [5, 6], 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 are removed in the ascending order of the following ratio:

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

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 [2] 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.

For illustrating each repair method, we randomly generate an n-dimensional binary vector $\mathbf{x} = (x_1, ..., x_n)$ by assigning 0 with the probability 0.4 and 1 with the probability 0.6 to each x_j. Then we generate a feasible solution using one of the two repair methods if the randomly generated binary vector is unfeasible. When the

randomly generated binary vector is feasible, we return to the first step in order to generate another binary vector. The random generation of an unfeasible solution and the application of a repair method to the generated unfeasible solution are iterated to obtain a prespecified number of feasible solutions. We can draw the trajectory from each unfeasible solution to its repaired one in the objective space.

In Fig. 3 (a), we show the trajectories from 10 unfeasible solutions by the maximum ratio repair for the 2-500 test problem. In this figure, unfeasible and feasible solutions are denoted by open circles and closed circles, respectively. It should be noted that the same order of items specified by (4) is always used for all the 10 unfeasible solutions in the case of the maximum ration repair. Thus the directions of the trajectories are similar to each other in Fig. 3 (a). On the other hand, Fig. 3 (b) shows the trajectories from the same 10 unfeasible solutions by the weighted scalar repair. In the weighted scalar repair, a different order of items is used for each unfeasible solution because the weight vector $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_k)$ in (7) is randomly updated for each unfeasible solution. As a result, we observe various directions of the trajectories in Fig. 3 (b).

From the comparison between Fig. 3 (a) and Fig. 3 (b), one may think that the weighted scalar repair has a positive effect on the diversity of obtained solution sets by EMO algorithms. Our computational experiments in the next section show that better solution sets with larger diversity are actually obtained from the weighted scalar repair than the maximum ratio repair. Similar results have been reported with respect to the performance of the two repair methods in the literature [4, 18].

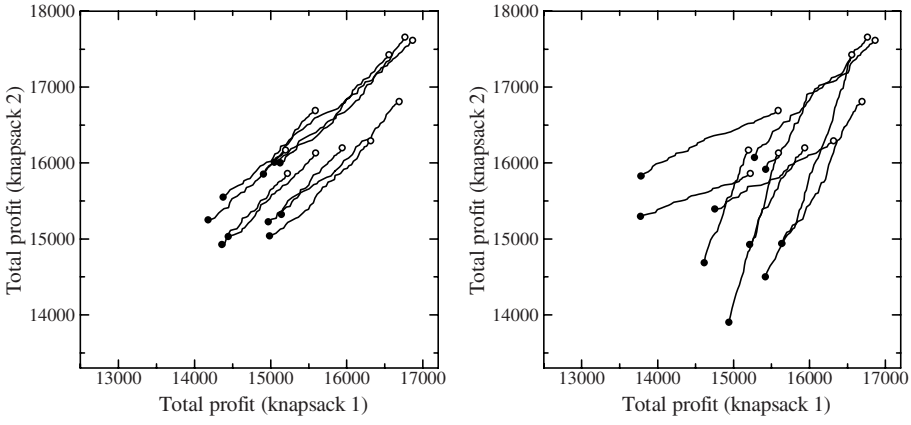


Fig. 3. Unfeasible solutions (open circles) and repaired solutions (closed circles)

Each repair method is implemented in the two implementation schemes: Lamarckian and Baldwinian. This means that we examine the four combinations of the two repair methods and the two implementation schemes. We also implement a

partial Lamarckian repair strategy, which can be viewed as a hybrid version of the Lamarckian and Baldwinian 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 approach, 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 incorporate each of the four combinations of the two repair methods and the two implementation schemes into the NSGA-II algorithm [2]. As a test problem, we use the two-objective 500-item knapsack problem (i.e., 2-500 test problem) in Zitzler & Thiele [16]. Each solution is coded as a binary string of length 500. Each of the four variants of the NSGA-II algorithm is applied to the test problem under the following parameter specifications:

Crossover probability (one-point crossover): 0.8,
Mutation probability (bit-flip mutation): 4/500,
Population size: 200,
Stopping condition: 500 generations.

The average performance of each variant 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 [8] and Zitzler et al. [17], no performance measure can simultaneously evaluate various aspects of a solution set. In this paper, we visually compare each variant of the NSGA-II algorithm by drawing the 50% attainment surface in the two-dimensional objective space over 30 runs. We also use two performance measures that are

applicable to simultaneous comparison of many solution sets. Let S be a solution set obtained by an EMO algorithm. The proximity of the solution set S to the Pareto front is evaluated by the generational distance (GD) as follows [15]:

$$\mathrm{GD}(S) = \frac{1}{|S|} \sum_{\mathbf{x} \in S} \min\{d_{\mathbf{xy}} \mid \mathbf{y} \in S^*\}, \tag{10}$$

where S^* is a reference solution set (i.e., the set of all 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 ($k = 2$ in our computational experiments):

$$d_{\mathbf{xy}} = \sqrt{(f_1(\mathbf{x}) - f_1(\mathbf{y}))^2 + \cdots + (f_k(\mathbf{x}) - f_k(\mathbf{y}))^2}. \tag{11}$$

For evaluating both the diversity of solutions in the solution set S and the convergence to the Pareto front, we calculate the $\mathrm{D1_R}$ measure as follows [1]:

$$\mathrm{D1_R}(S) = \frac{1}{|S^*|} \sum_{\mathbf{y} \in S^*} \min\{d_{\mathbf{xy}} \mid \mathbf{x} \in S\}. \tag{12}$$

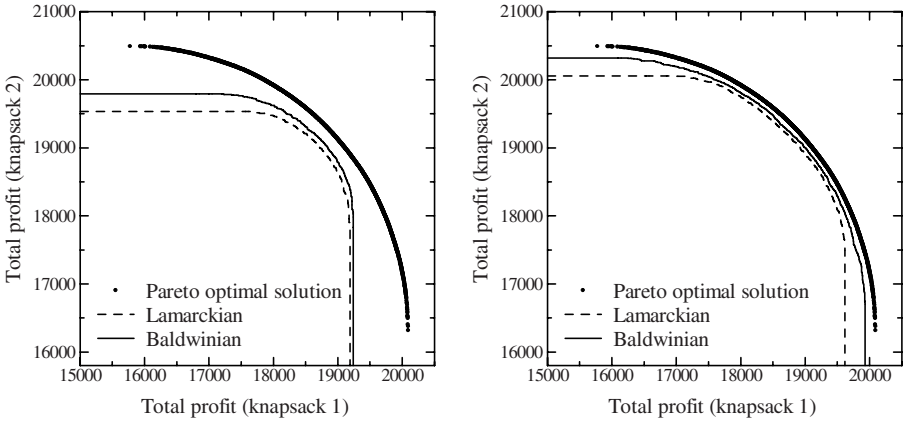
It should be noted that $\mathrm{D1_R}(S)$ is the average distance from each reference solution $\mathbf{y}$ in S^* to its nearest solution in S while $\mathrm{GD}(S)$ in (10) is the average distance from each solution $\mathbf{x}$ in S to its nearest reference solution in S^*. The generational distance evaluates the proximity of the solution set S to the reference solution set S^*. On the other hand, the $\mathrm{D1_R}$ measure evaluates how well the solution set S approximates the reference solution set S^*. Since all the Pareto-optimal solutions are known for the 2-500 test problem, we can use them as the reference solution set S^*.

4.2 Comparison of Two Implementation Schemes: Lamarckian and Baldwinian

Experimental results are summarized in Table 1 where the average value of each performance measure over 30 runs is shown together with the corresponding standard deviation in parentheses. It should be noted that smaller values of each performance measure mean better results. In Table 1, the better result between the two implementation schemes (i.e., Lamarckian and Baldwinian) is indicated by boldface for each combination of the two performance measures and the two repair methods. From Table 1, we can see that the Baldwinian implementation consistently outperforms the Lamarckian implementation for all the four combinations of the two performance measures and the two repair methods. We can also see from Table 1 that much better results are obtained from the weighted scalar repair than the maximum ratio repair as we expected from the trajectories by each repair method in Fig. 3. For visually demonstrating the above-mentioned observations, we show in Fig. 4 the 50% attainment surface over 30 runs for each combination of the two implementation schemes and the two repair methods. We can see from Fig. 4 that better results are obtained from the Baldwinian implementation scheme and the weighted scalar repair.

Table 1. Average value of each performance measure over 30 runs and the corresponding standard deviation in parentheses

Repair method	Generational distance (GD)		$D1_R$ measure	
	Lamarckian	Baldwinian	Lamarckian	Baldwinian
Maximum ratio	324 (32)	**239** (24)	632 (42)	**502** (47)
Weighted scalar	155 (17)	**100** (19)	269 (26)	**103** (18)



(a) Maximum ratio repair (b) Weighted scalar repair

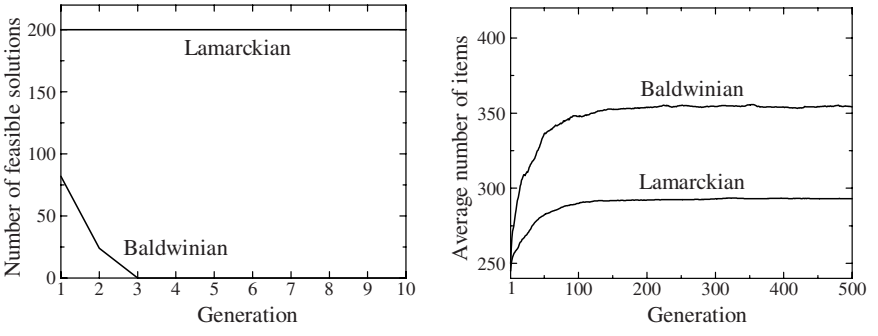
Fig. 4. Pareto front of the 2-500 test problem and the 50% attainment surface obtained from 30 runs using each combination of the two implementation schemes and the two repair methods

In order to further examine the two implementation schemes, we monitor the number of feasible solutions among 200 individuals in each generation. The number of feasible solutions in each generation is shown in Fig. 5 (a) for a single run with the maximum ratio repair. As we have already explained, all the 200 individuals at each generation are always feasible in the case of the Lamarckian implementation scheme. On the contrary, the number of feasible solutions rapidly decreases to zero in the early stage of evolution in the case of the Baldwinian implementation scheme in Fig. 5 (a).

We also monitor the average number of items in each solution during the same computational experiments as Fig. 5 (a). Experimental results are summarized in Fig. 5 (b). From Fig. 5 (b), we can see that much more items are included in each solution in the case of the Baldwinian repair than the Lamarckian repair. We can also see that the increase in the average number of items is very slow after the 100th generation even in the case of the Baldwinian repair where all solutions are infeasible after the 3rd generation as shown in Fig. 5 (a).

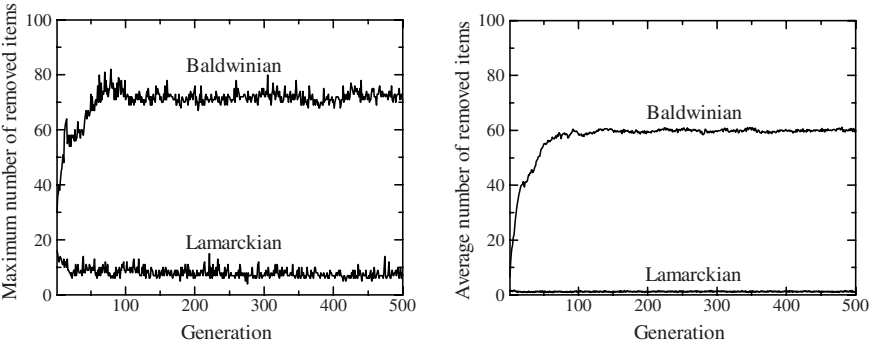
Furthermore, we monitor the number of removed items from each individual in each generation during the same computational experiments as Fig. 5. It should be

noted that excess items are not actually removed from individuals in the case of the Baldwinian repair. We monitor the number of excess items that are tentatively removed to evaluate each individual in the case of the Baldwinian repair. The maximum number and the average number of removed items over 200 individuals in each generation are shown in Fig. 6 (a) and Fig. 6 (b), respectively. From Fig. 6 (b), we can see that only a few items are removed in the case of the Lamarckian repair on average while each individual includes about 60 excess items in the case of the Baldwinian repair.



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

Fig. 5. Experimental results of a single run with the maximum ratio repair

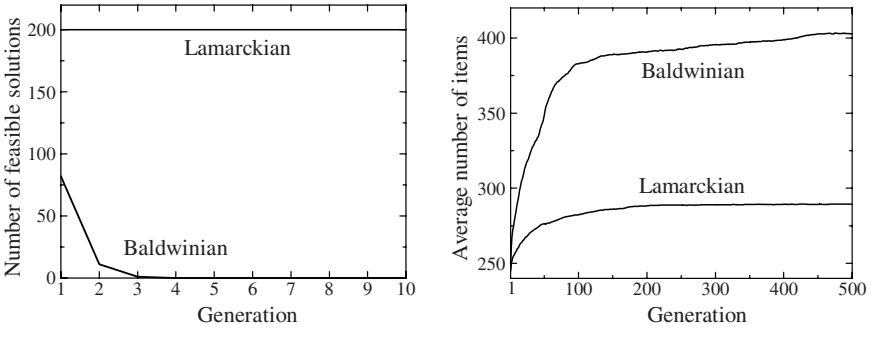


(a) Maximum number (b) Average number

Fig. 6. Maximum and average numbers of excess items removed by each implementation scheme of the maximum ratio repair

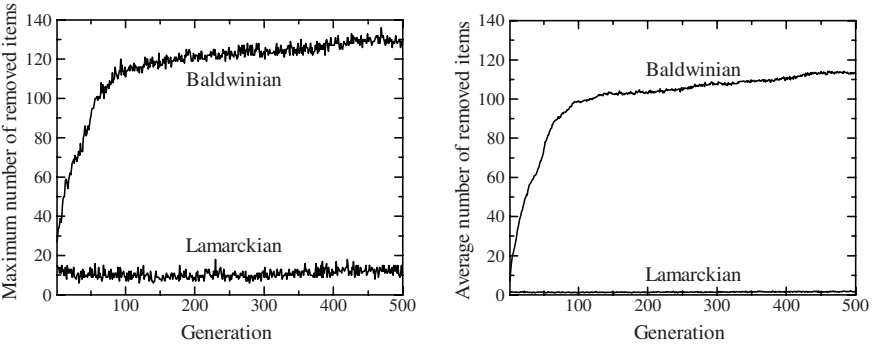
In the same manner as Fig. 5 and Fig. 6, experimental results by the weighted scalar repair are shown in Fig. 7 and Fig. 8. From Figs. 5-8, we can see that the number of excess items consistently increases over 500 generations only when we use

the Baldwinian implementation of the weighted scalar repair (see Fig. 8 (b)). As a result, much more items are included in each individual in this case. Even when the weighted scalar repair is used, only a few excess items are included in each individual in the case of the Lamarckian implementation (see Fig. 8 (b)). The difference in the average number of excess items between the two repair methods can be explained by the increase in the diversity of solutions by the weighted scalar repair. As shown in Fig. 3 (b), the weighted scalar repair spreads each individual over a wide range of the objective space. Thus the increase in the number of excess items contributes to the increase in the diversity of solutions in the objective space when we use the Baldwinian implementation of the weighted scalar repair. This explanation is consistent with the good result of the 50% attainment surface by the Baldwinian implementation of the weighted scalar repair in Fig. 4 (b).



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

Fig. 7. Experimental results of a single run with the weighted scalar repair

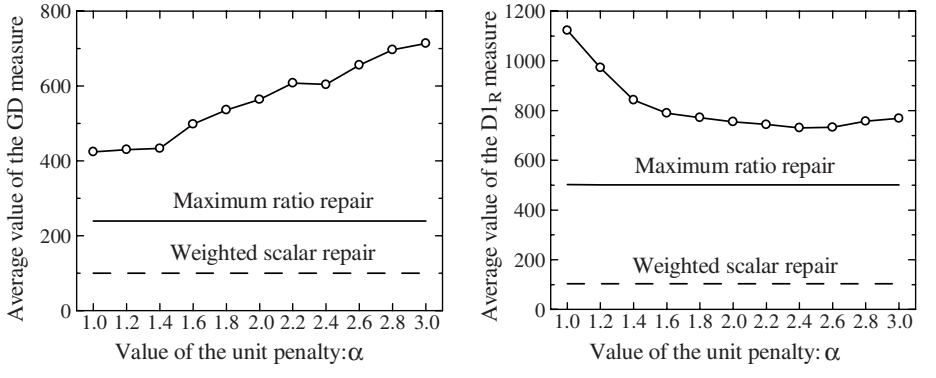


(a) Maximum number (b) Average number

Fig. 8. Maximum and average numbers of excess items removed by each implementation scheme of the weighted scalar repair

4.3 Comparison with Penalty Function Approach

In the same manner as in Subsection 4.2, we apply the NSGA-II algorithm to the 2-500 test problem 30 times using the penalty function approach in (8)-(9). We examine the various specifications of the unit penalty α: $\alpha = 1.0, 1.2, 1.4, ..., 3.0$. Average results over 30 runs are depicted by open circles in Fig. 9 where we also show the average results by the Baldwinian implementation of the two repair methods in Subsection 4.2. It should be noted that smaller values mean better results in Fig. 9. From Fig. 9, we can see that good results are not obtained by the penalty function approach while we examine a wide range of parameter values. More specifically, we can see from Fig. 9 (a) that the convergence to the Pareto front degrades as the unit penalty α increases. On the other hand, we can observe in Fig. 9 (b) that the diversity of obtained solutions degrades as the unit penalty α decreases.



(a) Generational distance (GD) (b) $D1_R$ measure

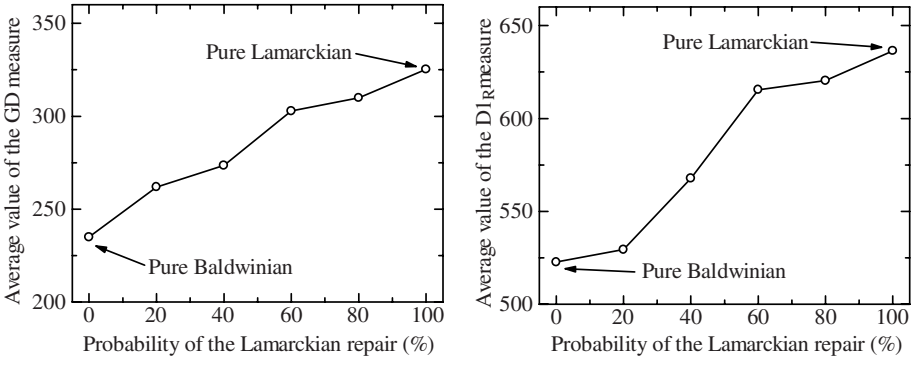
Fig. 9. Average results by the penalty function approach with various parameter values

4.4 Partial Lamarckianism

It has been demonstrated in the literature [3, 10, 12, 13] that a partial Lamarckianism search strategy outperforms both pure Lamarckian and pure Baldwinian. We examine the effect of the partial Lamarckian implementation where the Lamarckian repair is applied to each individual with a prespecified probability (say P_L %). When the Lamarckian repair is not applied, the individual is handled in the framework of the Baldwinian repair. We examine such a partial Lamarckian implementation for several specifications of the probability P_L of the Lamarckian repair: $P_L = 0$, 20, 40, 60, 80, 100 (%). It should be noted that $P_L = 0$ and $P_L = 100$ mean the pure Baldwinian repair and the pure Lamarckian repair, respectively. Experimental results are shown in Fig. 10 for the case of the maximum ratio repair. Fig. 10 (a) and Fig. 10 (b) show the average values of the generational distance (GD) and the $D1_R$ measure over 30 runs,

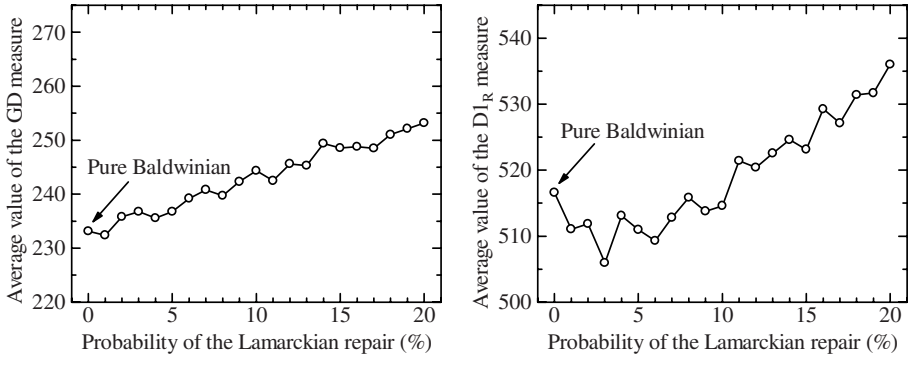
respectively. From Fig. 10, we can see that the quality of obtained solution sets is degraded by increasing the probability of the Lamarckian repair.

In order to examine the validity of a so-called 5% rule [12, 13], we perform exhaustive computational experiments using various values of the probability of the Lamarckian repair between 0% and 20%. Average results over 500 runs are shown in Fig. 11. From this figure, we can see that the partial Lamarckian repair improves the $D1_R$ measure of the pure Baldwinian repair when the probability of the Lamarckian repair is specified around 5%. That is, the diversity of obtained solutions is improved by invoking the Lamarckian repair with a low probability. This observation supports the validity of the 5% rule in the implementation of a partial Lamarckianism repair for multiobjective 0/1 knapsack problems.



(a) Generational distance (GD) (b) $D1_R$ measure

Fig. 10. Average results by the partial Lamarckian implementation of the maximum ratio repair over 30 runs



(a) Generational distance (GD) (b) $D1_R$ measure

Fig. 11. Further examination of the partial Lamarckian implementation of the maximum ratio repair through computational experiments with 500 runs

4.5 Comparison Between Partial Lamarckianism and Island Model

Recently Skolicki & De Jong [14] clearly demonstrated high search ability of multi-representation island models for single-objective continuous optimization problems where each island used a different coding method. We examine a similar idea using the two implementation schemes of greedy repair (i.e., Lamarckian and Baldwinian). That is, we examine an island model with two subpopulations where one island uses the Lamarckian repair and the other island uses the Baldwinian repair. A prespecified number of individuals are randomly chosen from non-dominated solutions in each island (i.e., each subpopulation) at intervals of a prespecified number of generations. Then their copies are inserted to the other island. This island model is executed using the following parameter specifications:

Subpopulation size: 100 (individuals),
Migration interval: 10, 50, 100 (generations),
Number of migrants: 10, 20, 50 (individuals).

The other parameter values are the same as the previous computational experiments. The size of each subpopulation is specified as 100 so that the total number of individuals (i.e., 2×100) is the same as the population size in the previous computational experiments. This specification is for comparing the island model with the standard single-population model with the 50% partial Lamarckian repair under the same computation load.

Experimental results by the island model are shown in Table 2 where the average values of the generational distance (GD) and the $D1_R$ measure are calculated over 30 runs. We also show average results by the standard single-population model with the 50% partial Lamarckian repair. The maximum ratio repair is used in Table 2. The best result for each performance measure is indicated by boldface. From this table, we can see that the island model improves the diversity of obtained solutions (see the fourth column labeled as $D1_R$) while it degrades the convergence to the Pareto front (see the third column labeled as GD).

Table 2. Average results over 30 runs by the island model with various parameter specifications. The maximum ratio repair is used in computational experiments in this table

Migration interval	Number of migrants	GD	$D1_R$
10	10	296	556
10	20	290	579
10	50	313	594
50	10	288	566
50	20	285	568
50	50	300	560
100	10	295	570
100	20	296	**549**
100	50	291	561
50% Lamarckian		**277**	586

5 Concluding Remarks

We examined two implementation schemes (i.e., Lamarckian and Baldwinian) of greedy repair through computational experiments on multiobjective 0/1 knapsack problems using the NSGA-II algorithm. While the Lamarckian implementation has been almost always used in the application of EMO algorithms to multiobjective 0/1 knapsack problems in the literature, better results were obtained by the Baldwinian implementation in this paper. The main contribution of this paper is that the superiority of the Baldwinian implementation over the Lamarckian implementation was clearly demonstrated through computational experiments on multiobjective 0/1 knapsack problems. This observation is consistent with some reported results on single-objective optimization problems [9, 10, 12, 13].

We also compared two greedy repair methods (i.e., maximum ratio repair and weighted scalar repair) with each other using the two implementation schemes. Independent of the choice of an implementation scheme, better results were obtained by the weighted scalar repair than the maximum ratio repair. This observation is consistent with some reported results [4, 5, 6, 18]. For comparison, we also examined the performance of the penalty function approach. In our computational experiments, better results could not be obtained from the penalty function approach in comparison with greedy repair.

In addition to pure Lamarckian and pure Baldwinian, we also examined their hybrid version (i.e., partial Lamarckianism search strategy) where the Lamarckian repair was applied to each individual with a prespecified probability. Some of our experimental results supported a so-called 5% rule where the Lamarckian repair and the Baldwinian repair were applied to each individual with 5% and 95% probabilities, respectively. That is, only 5% unfeasible solutions were actually repaired. Finally, we examined the performance of an island model with two subpopulations where each island used a different implementation scheme (i.e., Lamarckian or Baldwinian). In our computational experiments, we did not observe clear improvement by such an island model from the standard single-population model with the 50% partial Lamarckian repair. The use of the island model slightly improved the diversity of solutions.

While we empirically showed the superiority of the Baldwinian implementation over the Lamarckian implementation through computational experiments on multiobjective 0/1 knapsack problems, we could not clearly explain why the Baldwinian implementation outperformed the Lamarckian implementation in the application of EMO algorithms to multiobjective 0/1 knapsack problems. 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 for multiobjective optimization problems.

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. 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.
2. 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.
3. Houck, C. R., Joines, J. A., Kay, M. G., and Wilson, J. R.: Empirical Investigations of the Benefits of Partial Lamarckianism. Evolutionary Computation **5** (1997) 31-60.
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. Jaszkiewicz, A.: Comparison of Local Search-based Metaheuristics on the Multiple Objective Knapsack Problem. Foundations of Computing and Decision Sciences **26** (2001) 99-120.
6. 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.
7. 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 (2000) 103-108.
8. Knowles, J. D. and Corne, D. W.: On Metrics for Comparing Non-dominated Sets. Proc. of 2002 Congress on Evolutionary Computation (2002) 711-716.
9. Liu, B., Haftka, R. T., Akgun, M. A., and Todoroki, A.: Permutation Genetic Algorithm for Stacking Sequence Design of Composite Laminates. Computer Methods in Applied Mechanics and Engineering **186** (2000) 357-372.
10. Michalewicz, Z. and Nazhiyath, G.: Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints. Proc. of 2nd IEEE International Conference on Evolutionary Computation **2** (1995) 647-651.
11. 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.
12. Orvosh, D. and Davis, L.: Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints. Proc. of 5th International Conference on Genetic Algorithms (1993) 650.
13. Orvosh, D. and Davis, L.: Using Genetic Algorithms to Optimize Problems with Feasibility Constraints. Proc. of 1st IEEE Conference on Evolutionary Computation (1994) 548-553.
14. Skolicki, Z. and De Jong, K.: Improving Evolutionary Algorithms with Multi-representation Island Models. Lecture Notes in Computer Science, Vol. 3242 (Proc. of PPSN VIII), Springer, Berlin (2004) 420-429.
15. Van Veldhuizen, D. A.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ph. D dissertation, Air Force Institute of Technology (1999).
16. 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.

17. 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.
18. 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.