Constraint-Handling Method for Multi-objective Function Optimization: Pareto Descent Repair Operator

Ken Harada, Jun Sakuma, Isao Ono, and Shigenobu Kobayashi

Department of Computational Intelligence and Systems Science,
Tokyo Institute of Technology,
4259 Nagatsuta-cho Midori-ku Yokohama-shi Kanagawa-ken 226-8502, Japan
{ken,jun}@fe.dis.titech.ac.jp, {isao,kobayasi}@dis.titech.ac.jp
http://www.fe.dis.titech.ac.jp/

Abstract. Among the multi-objective optimization methods proposed so far, Genetic Algorithms (GA) have been shown to be more effective in recent decades. Most of such methods were developed to solve primarily unconstrained problems. However, many real-world problems are constrained, which necessitates appropriate handling of constraints. Despite much effort devoted to the studies of constraint-handling methods, it has been reported that each of them has certain limitations. Hence, further studies for designing more effective constraint-handling methods are needed.

For this reason, we investigated the guidelines for a method to effectively handle constraints. Based on these guidelines, we designed a new constraint-handling method, Pareto Descent Repair operator (PDR), in which ideas derived from multi-objective local search and gradient projection method are incorporated. An experiment comparing GA that use PDR and some of the existing constraint-handling methods confirmed the effectiveness of PDR.

1 Introduction

Multi-objective optimization (MOO) has many real-world applications, e.g. portfolio optimization, for which multiple conflicting objective functions are to be simultaneously optimized. MOO whose variables are real-valued is called multi-objective function optimization, which is the subject of this paper. Genetic Algorithms (GA) are known to be relatively efficient and effective MOO methods [1]. GA applies crossover and selection to a set of solutions and converge them to entire Pareto-optimal solutions. Selection for MOO consists of *ranking*, which brings solutions closer to Pareto-optimal solutions, and *sharing*, which enhances the diversity of solutions.

Most MOO methods, including GA, were designed for solving primarily unconstrained problems. However, real-world problems often have constraints, and the handling of them can substantially influence the performance of the optimization methods. When GA is applied to constrained problems, two major difficulties arise.

One of them is that some GA require feasible solutions to start with. The most naive way of obtaining feasible solutions is to randomly generate solutions until a prespecified number of them are found. However, this approach fails when the probability of obtaining a feasible solution in such a way is very low. Therefore, feasible solutions must be explicitly searched for, which is one role that constraint-handling methods play.

The other difficulty is that, on problems whose Pareto-optimal solutions lie on feasible region boundaries (boundaries hereafter), GA may not be able to obtain solutions close to the Pareto-optimal solutions. The most commonly used constraint-handling method in GA is death penalty (DP), which simply discards infeasible solutions. The solutions that GA generates can be mostly infeasible on problems whose Pareto-optimal solutions lie on boundaries. Extreme examples of such problems are ZDT1 and ZDT2 [1] whose Pareto-optimal solutions form line segments at which 29 constraint boundaries intersect perpendicularly. When the solutions that GA maintains come near the Pareto-optimal solutions, most of the solutions that GA generates are infeasible and discarded by DP, which implies that GA cannot obtain solutions close to the Pareto-optimal solutions. Therefore, effective constraint-handling methods which facilitate searching for Pareto-optimal solutions on boundaries are necessary.

One class of constraint-handling methods modify solution representation and/or crossover so that infeasible solutions can never be generated [2]. However, these methods are not applicable to general problems. Another class of methods attempt to search for feasible solutions from infeasible solutions by reducing constraint violations. The existing methods of this kind are known to have certain limitations as described in Sect. 2.2.

In order to design an effective constraint-handling method, we first investigate the guidelines for a method to effectively handle constraints. We then explain the concepts and calculations necessary to meet these guidelines and propose them as Pareto Descent Repair operator (PDR).

Section 2 formulates constrained multi-objective function optimization, explains Pareto-optimality, and reviews existing constraint-handling methods. Section 3 presents the guidelines for effective constraint handling and explains the details of PDR. To demonstrate the effectiveness of PDR, Sect. 4 shows the results of experiments comparing PDR and other constraint-handling methods when they are used in GA. Lastly, Sect. 5 summarizes this paper.

2 Constraint Handling in Multi-objective Function Optimization

2.1 Constrained Multi-objective Function Optimization

Formulation. Constrained multi-objective function optimization problem can generally be formulated as

$$\text{Minimize } \boldsymbol{f}(\boldsymbol{x}) \text{ subject to } \boldsymbol{x} \in S \ , \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^N$, and $\boldsymbol{f} = (f_1, f_2, \ldots, f_M)^T$ is a vector of M objective functions. *Feasible region* S is the region that satisfies inequality constraints $\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}$, where $\boldsymbol{g} = (g_1, g_2, \ldots, g_P)^T$ is a vector of P constraint functions. Solutions that satisfy all constraints are said to be *feasible*, and those that do not, *infeasible*. Objective functions are defined for arbitrary feasible solutions, and constraint functions, for arbitrary solutions. Constraint functions are assumed to be continuously differentiable in this paper, since there are a considerable number of problems for which analytical or approximate gradients of constraint functions are available and continuous.

If $g_j(\boldsymbol{x}) = 0$ holds for solution $\boldsymbol{x}$, $\boldsymbol{x}$ lies on the *boundary* of the corresponding constraint, and the constraint is said to be *active* at $\boldsymbol{x}$. If a direction $\boldsymbol{d} \in \mathbb{R}^N$ satisfies $\boldsymbol{d} \cdot \nabla g_j(\boldsymbol{x}) \leq 0$, $\boldsymbol{d}$ is said to be *feasible* w.r.t. the active constraint. The *constraint violation* of constraint $g_j(\boldsymbol{x}) \leq 0$ at $\boldsymbol{x}$ can be defined as $g_j^+(\boldsymbol{x}) = \max(g_j(\boldsymbol{x}), 0)$. By reducing the positive components of $\boldsymbol{g}^+ = (g_1^+, g_2^+, \ldots, g_P^+)^T$, feasible solutions can be searched for.

Pareto-Optimality and the Objective of MOO methods. If, for $\boldsymbol{x_1}, \boldsymbol{x_2} \in S$,

$$\forall i \in \{1, 2, \ldots, M\},\ f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2) \wedge \exists i \in \{1, 2, \ldots, M\},\ f_i(\boldsymbol{x}_1) < f_i(\boldsymbol{x}_2)$$

holds, $\boldsymbol{x_1}$ is said to be *superior* to $\boldsymbol{x_2}$, which is denoted by $\boldsymbol{x_1} \succ \boldsymbol{x_2}$. If a solution $\boldsymbol{x}$ in a set of solutions is not inferior to any other solution in the set, $\boldsymbol{x}$ is said to be *non-inferior* within the set. If $\boldsymbol{x}' \in S$ such that $\boldsymbol{x}' \succ \boldsymbol{x}$ does not exist, $\boldsymbol{x}$ is said to be *Pareto-optimal*. If a solution $\boldsymbol{x}'$ such that $\boldsymbol{x}' \succ \boldsymbol{x}$ does not exist in the feasible ε-vicinity of $\boldsymbol{x}$, $\boldsymbol{x}$ is said to be *locally Pareto-optimal*. There are often multiple Pareto-optimal and locally Pareto-optimal solutions.

The objective of MOO methods is to find a set of solutions which are close to Pareto-optimal solutions (*proximity*) and which evenly cover entire Pareto-optimal solutions (*diversity*) [1,3].

2.2 Existing Constraint-Handling Methods

This section reviews prominent constraint-handling methods which search feasible solutions by reducing constraint violations, and explains their drawbacks regarding the abilities to find feasible solutions and to search Pareto-optimal solutions on boundaries.

Penalty Methods. A vector of penalty functions $\boldsymbol{P}(\boldsymbol{x}) = (P_1(\boldsymbol{x}), P_2(\boldsymbol{x}), \ldots, P_M(\boldsymbol{x}))^T$, each of whose components represents the degree of overall constraint violation at a solution, is defined, and the unconstrained optimization problem

$$\text{Minimize } \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{P}(\boldsymbol{x})$$

is solved. It has been pointed out that it is difficult to design appropriate penalty functions [2]. In addition, penalty methods cannot be used when there are infeasible solutions for which objective functions are undefined.

Objectivization of Constraint Violations. Constraint violations are regarded as additional objective functions, and the unconstrained problem

$$\text{Minimize } \tilde{\boldsymbol{f}}(\boldsymbol{x}), \text{ where } \tilde{\boldsymbol{f}} = (f_1, f_2, \ldots, f_M, g_1^+, g_2^+, \ldots, g_P^+)^T \ , \tag{2}$$

is solved [2]. Many methods that use GA for the optimization of (2) have been proposed [4,2]. We call such methods OCV(e) since constraint violations and objective functions are treated equally. It has also been proposed to redefine superiority relationship in OCV so that the regular superiority relationship is used for feasible solutions, the multi-objective superiority relationship w.r.t constraint violations is used for infeasible solutions, and feasible solutions are always superior to infeasible solutions [5,6]. Such variants of OCV are called OCV(ne) in this paper.

Regardless of whether or not constraint violations and objective functions are treated equally, OCV may not be able to find feasible solutions. Consider the 2-variable-2^2-constraint problem shown in Fig. 1. Assume that the feasible region is sufficiently small, and, when solutions are randomly generated, at least one solution is inside each of the 2^2 dark-shaded areas which violate two constraints. In this situation, at least one non-inferior solution w.r.t. constraint violations exists in each of the dark-shaded areas, and there are at least 2^2 of them in total. On a similar problem with N variables, the number of such solutions is at least 2^N. When N is big, almost all of the randomly-generated solutions are non-inferior, for which ranking does not function. Sharing, in the meantime, attempts to increase the diversity of solutions and, as a result, disperses the solutions. Hence, OCV may not find feasible solutions, which will be demonstrated in Sect. 4.

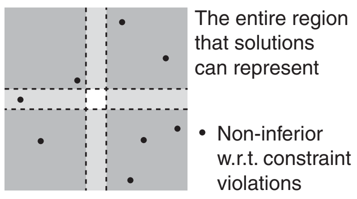
The entire region that solutions can represent
• Non-inferior w.r.t. constraint violations

Fig. 1. A problem on which OCV fails when the dimension of the variable space becomes big. Dashed lines denote constraint boundaries. Infeasible regions are shaded, and, the more constraints they violate, the darker they are shaded.

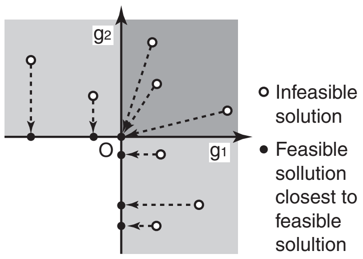

Fig. 2. Feasible solutions closest to infeasible solutions in the constraint function space

Another drawback of OCV(e) is that, on problems whose Pareto-optimal solutions lie on boundaries, infeasible solutions remain in the set of solutions throughout the entire search [2]. Furthermore, in some cases, not even one feasible solution may be found, as demonstrated in Sect. 4. In addition, OCV(e) itself is infeasible when there are some infeasible solutions for which objective functions are undefined. Although OCV(ne) does not have these drawbacks, it can practically reduce to DP: when most of the offspring solutions that GA generates are infeasible, they are simply discarded since their parent solutions are feasible and superior to them. Therefore, OCV(ne) cannot facilitate searching for Pareto-optimal solutions on boundaries, either.

Repair Operators. Repair operators for function optimization search for feasible solutions by reducing constraint violations, *without* considering objective functions. Such repair operators have a great potential since they are applicable to any function optimization problems. However, there have been a very small number of such studies [2]. Although GENOCOP III [7] is proposed as a repair operator, it cannot be used to search for feasible solutions since it assumes that some feasible solutions are available.

3 Pareto Descent Repair Operator

3.1 Guidelines for Effective Constraint Handling

This section gives the guidelines for designing an effective constraint-handling method that circumvents the problems pointed out in Sect. 2.2.

Guideline 1: Take the Repair Operator Approach. It is difficult to define appropriate penalty functions for penalty methods, and OCV may find no feasible solutions, as pointed out in the previous section. In addition, some of these methods themselves are infeasible if objective functions are undefined for some infeasible solutions. These imply that the approach of repair operators is more promising.

Guideline 2: Monotonically Decrease the Number of Violated Constraints and Constraint Violations. A feasible solution can be searched for by reducing constraint violations, as mentioned earlier. Since there are multiple constraint violations, it can be regarded as an MOO problem. Note that, since constraint functions are assumed to be continuous, there is a region, surrounding each feasible region, in which constraint functions can be regarded as unimodal. In fact, when constraint functions are linear or quadratic, constraint functions are unimodal in the entire infeasible region. Note also that infeasible solutions generated during GA's search are often near feasible regions, and constraint functions can be regarded as unimodal around the infeasible solutions. Being able to repair infeasible solutions in such regions is important in terms of both improving the probability of obtaining initial feasible solutions and facilitating GA's search on problems whose Pareto-optimal solutions lie on boundaries. In order to repair infeasible solutions in such regions, it is appropriate to monotonically decrease both the number of violated constraints and constraint violations.

Guideline 3: Search for the Feasible Solution Closest in the Constraint Function Space. Since constraint violations represent the degrees of violation of constraints, it is reasonable to search for the feasible solution at which violated constraint functions are as close to zero as possible, that is, the feasible solution closest to the infeasible solution in the constraint function space. This repairing approach in the case of two constraints is shown schematically in Fig. 2.

3.2 Strategies for Meeting the Guidelines

Guideline 1 implies the use of repair operators for constraint handling. This section explains what are necessary for meeting Guidelines 2 and 3.

Decrease Constraint Violations using Multi-objective Local Search. To monotonically decrease constraint violations, a multi-objective local search can be used with violated constraint functions regarded as objective functions. In this paper, Pareto Descent Method (PDM) [8,9] is used, which, as mentioned in the appendix, calculates appropriate Pareto descent directions and descent directions with relatively small computational cost and efficiently decreases all objective functions simultaneously. PDM consists of search direction calculation and linear search, and a repair operator based on it has a similar structure.

Search for Feasible Solutions on Boundaries. When no violated constraints have been satisfied yet, there is no active unviolated constraints (active constraints hereafter), and the search direction should be a Pareto descent direction of violated constraint functions so that they are decreased efficiently. When there are active constraints, violated constraint functions have to be reduced on the boundaries of the active constraints, since the feasible solutions closest to infeasible solutions in the constraint function space are on the boundaries of initially violated constraints. For this purpose, we can draw on the ideas of gradient projection method [10]. In the constraint-handling context, the search direction must be in the null-space of the gradients of active constraint functions. In order to decrease constraint violations in the null-space, the search direction should be a Pareto descent direction of the violated constraint functions in the null-space if such a direction exists, and a descent direction in the null-space otherwise. Linear search must be conducted while moving solutions in the search direction back onto the boundaries of active constraints.

Even when there are no descent directions in the null-space, the number of violated constraints and constraint violations may be further reduced by regarding some of the active constraints as inactive (*inactivation*). Consider the 2-variable-3-constraint problem shown in Fig. 3. Since there are two active constraints at $\boldsymbol{x}_1$, no descent directions of c_3 exist in the null-space. When c_1 is considered inactive, there are descent directions of c_3 in the null-space that are feasible w.r.t. c_1. Hence, feasible solution $\boldsymbol{x}_2$ can be obtained by inactivating c_1. Note that, when some active constraints are inactivated, not all violated constraint functions can be zero at the resulting feasible solution.

The details of these direction calculations and linear search are described in the following sections.

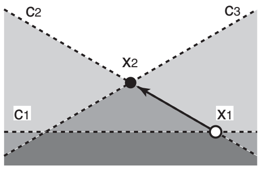

Fig. 3. Inactivation. At infeasible solution $\boldsymbol{x}_1$, there are no descent directions of the violated constraint functions in the null-space of the gradients of the active constraint functions. Inactivation of c_1 allows for obtaining feasible solution $\boldsymbol{x}_2$.

3.3 Search Direction Calculation

When No Active Constraints Exist. Pareto descent directions of violated constraint functions can be obtained using PDM, if they exist. If they do not, PDM detects it [8,9]. In this case, descent directions do not exist either, which implies that constraint violations cannot be locally decreased any further.

When Active Constraints Exist. Denote the active constraint functions by $\hat{g}_j^{\mathrm{u}}$ $(j = 1, 2, \ldots, \hat{P}^{\mathrm{u}})$ and those of violated constraints by g_j^{v} $(j = 1, 2, \ldots, P^{\mathrm{v}})$. In order for a search direction $\boldsymbol{d} \in \mathrm{I\!R}^N$ to be a descent direction of the violated constraint functions in the null-space of the gradients of the active constraint functions, $\boldsymbol{d}$ has to satisfy

$$\hat{G}^{\mathrm{u}T}\boldsymbol{d} = \boldsymbol{0} \ , \text{ where } \hat{G}^{\mathrm{u}} = [\nabla\hat{g}_1^{\mathrm{u}}, \ldots, \nabla\hat{g}_{\hat{P}^{\mathrm{u}}}^{\mathrm{u}}] \text{ and} \tag{3}$$

$$G^{\mathrm{v}T}\boldsymbol{d} \leq \boldsymbol{0} \ , \text{ where } G^{\mathrm{v}} = [\nabla g_1^{\mathrm{v}}, \ldots, \nabla g_{P^{\mathrm{v}}}^{\mathrm{v}}] \ . \tag{4}$$

The following sections detail the calculations of Pareto descent directions and descent directions in the null-space.

Pareto Descent Directions in the Null-space. The condition for a descent direction $\boldsymbol{d}$ in the null-space to be a Pareto descent direction is that there exists a convex combination weight $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{P^{\mathrm{v}}})^T \in \mathrm{I\!R}_+^{P^{\mathrm{v}}}$, where $\mathrm{I\!R}_+$ is the set of non-negative real numbers, such that

$$\boldsymbol{d} = -G^{\mathrm{v}}\boldsymbol{\alpha} \ . \tag{5}$$

Substituting this into (3) gives

$$G\boldsymbol{\alpha} = \boldsymbol{0} \ , \text{ where } G = -\hat{G}^{\mathrm{u}T}G^{\mathrm{v}} \ . \tag{6}$$

Denote the rank of G by $r(G)$. When $r(G) = P^{\mathrm{v}}$, the sole solution $\boldsymbol{\alpha} = \boldsymbol{0}$ of (6) represents $\boldsymbol{d} = \boldsymbol{0}$, which implies that no Pareto descent directions exist. When $r(G) = 0$, the gradients of violated constraint functions are already in the null-space, and the search direction should simply be a Pareto descent direction of the violated constraint functions. When $0 < r(G) < P^{\mathrm{v}}$, (6) implies that $\boldsymbol{\alpha}$ exists in a subspace of dimension $P^{\mathrm{v}} - r(G)$. Denote the basis vectors of the subspace by $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{P^{\mathrm{v}}-r(G)} \in \mathbb{R}^{P^{\mathrm{v}}}$ and the coordinates of $\boldsymbol{\alpha}$ in the subspace by $\boldsymbol{\beta} \in \mathbb{R}^{P^{\mathrm{v}}-r(G)}$. Now $\boldsymbol{\alpha}$ can be expressed as $\boldsymbol{\alpha} = U\boldsymbol{\beta}$, where $U = \left[\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{P^{\mathrm{v}}-r(G)}\right]$. Substituting this into (5) and then substituting the result into (4) gives

$$-G^{\mathrm{v}T} G^{\mathrm{v}} U\boldsymbol{\beta} \leq \mathbf{0} \ . \tag{7}$$

The constraint $U\boldsymbol{\beta} \geq \mathbf{0}$ that each component of $\boldsymbol{\alpha}$ is non-negative and (7) are homogeneous linear inequalities of $\boldsymbol{\beta}$. This is of the same form as that for calculating Pareto descent directions in PDM. PDM calculates $\boldsymbol{\alpha}$ that maximizes α_i for each $i = 1, 2, \ldots, M$ to obtain Pareto descent directions. Similarly, $\boldsymbol{\beta}$ that maximizes α_i for each $i = 1, 2, \ldots, M$ can be calculated, which give Pareto descent directions in the null-space.[1]

Descent Directions in the Null-space. When $r(\hat{G}^{\mathrm{u}T}) = N$, the sole solution $\boldsymbol{d} = \mathbf{0}$ of (3) implies that no descent directions exist. When $r(\hat{G}^{\mathrm{u}T}) < N$, (3) implies that $\boldsymbol{d}$ exists in a subspace of dimension $N - r(\hat{G}^{\mathrm{u}T})$. Denote the basis vectors of the subspace by $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{N-r(\hat{G}^{\mathrm{u}T})} \in \mathbb{R}^N$ and the coordinates of $\boldsymbol{d}$ in the subspace by $\boldsymbol{\gamma} \in \mathbb{R}^{N-r(\hat{G}^{\mathrm{u}T})}$. Now $\boldsymbol{d}$ can be expressed as $\boldsymbol{d} = E\boldsymbol{\gamma}$, where $E = \left[\boldsymbol{e}_1, \boldsymbol{e}_2 \ldots, \boldsymbol{e}_{N-r(\hat{G}^{\mathrm{u}T})}\right]$. Substituting this into (4) gives

$$G^{\mathrm{v}T} E\boldsymbol{\gamma} \leq \mathbf{0} \ . \tag{8}$$

This is a homogeneous linear inequality of $\boldsymbol{\gamma}$ and is the same form as that for calculating descent directions in PDM. Therefore, descent directions in the null-space can be obtained as descent directions are calculated in PDM.

Inactivation. Denote the set of active constraints by $\hat{C}^{\mathrm{u}}$, its subset by $\acute{C}^{\mathrm{u}}$, and the constraint functions of constraints in $\acute{C}^{\mathrm{u}}$ by $\acute{g}^{\mathrm{u}}_j$ ($j = 1, 2, \ldots, \acute{P}^{\mathrm{u}}$). In order for Guideline 2 to be satisfied when $\acute{C}^{\mathrm{u}}$ is inactivated, there must exist descent directions in the null-space of the gradients of the constraint functions of the constraints in $\hat{C}^{\mathrm{u}} \backslash \acute{C}^{\mathrm{u}}$ that are feasible w.r.t. $\acute{C}^{\mathrm{u}}$, i.e.,

$$\acute{G}^{\mathrm{u}T}\boldsymbol{d} \leq \mathbf{0} \ , \text{ where } \acute{G}^{\mathrm{u}} = [\nabla \acute{g}^{\mathrm{u}}_1, \ldots, \nabla \acute{g}^{\mathrm{u}}_{\acute{P}^{\mathrm{u}}}] \ . \tag{9}$$

Equation (9) can be incorporated into the above-mentioned calculations of descent directions and Pareto descent directions, and their existence can be tested by PDM. Hence, the possibility of inactivation of $\acute{C}^{\mathrm{u}}$ can be determined using PDM.[2]

When multiple subsets of $\hat{C}^{\mathrm{u}}$ can be inactivated, the one to be inactivated should be chosen based on the following rules according to Guideline 3:

[1] Some of the thus found Pareto descent directions may be redundant. Such redundant directions can be identified and removed as done in the calculation of descent directions in PDM [8,9].

[2] In order to find subsets that can be inactivated, every subset of $\hat{C}^{\mathrm{u}}$ must be examined. When the cardinality of $\hat{C}^{\mathrm{u}}$ is big, however, not all subsets can be examined, and some compromise has to be made.

1. Choose the subset with the smallest cardinality, and
2. If there are more than one such subsets, choose the one for which there exist Pareto descent directions of the violated constraints in the null-space of the gradients of the active constraint functions that are feasible w.r.t. the inactivated constraints.

3.4 Linear Search over Active Constraint Boundaries

Moving Solutions back onto Active Constraint Boundaries. In order to move a solution $\boldsymbol{y}$ in a search direction back onto active constraint boundaries, we can search for a solution that satisfies active constraints by small margins using $\boldsymbol{y}$ as the initial solution.[3] Consider using golden section method for the linear search in the optimization. The length of closed linear search interval and the number of iterations determine the maximum error ε that can transpire in the linear search. Minimizing $\sum_{j=1}^{\hat{P}^{\mathrm{u}}} \left(d_{\hat{g}_j^{\mathrm{u}}}(\boldsymbol{x}) - (-\varepsilon) \right)^2$ gives a solution that satisfies active constraints by the distance of at most 2ε, where $\hat{g}_j^{\mathrm{u}}(\boldsymbol{x})$ $(j = 1, 2, \ldots, \hat{P}^{\mathrm{u}})$ are active constraint functions, and $d_{\hat{g}_j^{\mathrm{u}}}(\boldsymbol{x})$ is the signed distance of $\boldsymbol{x}$ from the j-th boundary. Since $d_{\hat{g}_j^{\mathrm{u}}}(\boldsymbol{x})$ cannot usually be calculated precisely in practice, it has to be approximated. Applying Taylor expansion to $\hat{g}_j^{\mathrm{u}}(\boldsymbol{x})$ and ignoring the terms of order greater than two, $d_{\hat{g}_j^{\mathrm{u}}}(\boldsymbol{x})$ can be approximated [11] by

$$\tilde{d}_{\hat{g}_j^{\mathrm{u}}}(\boldsymbol{x}) = \frac{\nabla \hat{g}_j^{\mathrm{u}}(\boldsymbol{x}) \cdot \boldsymbol{x} + \hat{g}_j^{\mathrm{u}}(\boldsymbol{x})}{||\nabla \hat{g}_j^{\mathrm{u}}(\boldsymbol{x})||} \quad . \tag{10}$$

Linear Search. Since the number of violated constraints and constraint violations must be monotonically decreased according to Guideline 2, the step-size must be chosen so that the solution in the search direction is 1) just before any of the unviolated constraints is violated or 2) just before any of the violated constraint functions increases. Additionally, the step-size must be chosen so that the solution in the search direction is 3) just after any of the violated constraints is satisfied, since the next iteration searches over the boundaries of active constraints including the one just satisfied.

3.5 Proposal of Pareto Descent Repair Operator

We propose the repair operator consisting of the above-mentioned search direction calculations and linear search as Pareto Descent Repair operator (PDR). PDR efficiently decreases constraint violations by calculating an appropriate search direction for each case it may encounter: active constraints may or may not exist, and Pareto descent directions and descent directions may or may not exist. The most computationally intense part of PDR is that of solving linear

[3] We can alternatively search for a solution which minimizes the distance to each boundary. The optimum, however, may violate the active constraints by small margins since the linear search used in that optimization always transpires a small error.

programming problems for direction calculations. Since computationally efficient linear programming solvers such as simplex method [12] can be used, the computational complexity of PDR is accordingly small.

3.6 Use of PDR in GA

When a repair operator is used in GA, infeasible solutions can be replaced by their corresponding feasible solutions (Lamarckian). They can also be stored and used in crossover, and their corresponding feasible solutions are used for evaluations of objective functions (non-Lamarckian) [2]. Consider solving a problem with a single feasible region using a crossover operator such as UNDX [13] which generates offspring solutions around the center of mass of parent solutions. Note that, when GA generates infeasible offspring solutions, Pareto-optimal solutions are likely to lie on boundaries. When Lamarckian PDR (PDR(l) hereafter) is used, parent solutions are either inside the feasible region or on boundaries, and their offspring solutions therefore are inside the feasible region. When non-Lamarckian PDR (PDR(nl) hereafter) is used, parent solutions are both inside and outside the feasible region, and their offspring solutions are more likely to be generated near the boundaries. This difference becomes prominent on problems such as ZDT2 on which a number of boundaries intersect at the Pareto-optimal solutions. Therefore, Pareto-optimal solutions on boundaries are expected to be obtained with higher precision when PDR(nl) is used than when PDR(l) is used. Even if there are multiple feasible regions, a similar argument applies when mating restriction is imposed so that solutions close to each other, which often belong to the same feasible region, are chosen for mating.

4 Experiments

In order to verify the effectiveness of PDR, GA that use PDR(l), PDR(nl), OCV(e), and OCV(ne) are compared on some well-known multi-objective benchmark problems. The results of death penalty (DP) will also be shown just for a reference, since it is the standard constraint-handling method for GA.

4.1 Experiment Setup

Performance Metrics. In order to evaluate the proximity and diversity of solutions, we use generational distance (GD) and D1R, which are used in many existing studies. GD is defined as the mean of the distances from each solution to its nearest Pareto-optimal solution in the normalized objective space [1] and measures proximity. D1R is defined as the mean of the distances from each Pareto-optimal solution to its nearest solution in the normalized objective space [14] and measures both proximity and diversity.

Pareto-optimal solutions are necessary to evaluate GD and D1R. We assume that the solutions obtained by running GA with a large population size and many generations are Pareto-optimal, as existing studies do. Note that, when OCV is used, the set of solutions may contain infeasible solutions. Since the number of non-inferior solutions, which are necessarily feasible, is sometimes used as a performance metric [1], obtaining more feasible solutions is better than obtaining less. Since infeasible solutions only deteriorate both proximity and diversity, GD and D1R are calculated using all the solutions in the solution set.

Benchmark Problems. Since the methods being compared are applicable to problems with arbitrary numbers of objective functions and feasible regions, the benchmark problems in Table 1 are used, each of which has two objective functions and a single feasible region. These problems with relatively simple constraints were chosen so that the behaviors of the constraint-handling methods can be examined in detail.

Table 1. The properties of the benchmark problems used in the experiment [1]

Name	N	M	**# const.**	**Linearity of const.**	**Local Pareto-opt.**	**Pareto-optimal solutions in the variable space**
BNH	2	2	6	Linear and non-linear	No	A kinked line partly on a boundary
TNK	2	2	2	Non-linear	Yes	Multiple curves on a boundary
ZDT2	30	2	60	Linear	No	A line at which 29 boundaries intersect

GA. Population size is 100, which is commonly used for MOO. Initial solutions are generated uniformly at random in $[-100, 100]^N$. For DP, however, initial solutions are generated uniformly at random in feasible regions. 50 parent pairs are formed at each generation. Since it has been reported that proximity is improved as the number of offspring solutions for each parent pair is increased [15,16], 20 offspring solutions are generated for each pair. Since it has also been reported in [15,16] that, although the best-performing crossover is problem dependent, UNDX [13] performs relatively well on many problems, UNDX is used in the experiment. SPEA2 [17] is known to exhibit good performance as a survival selection [18]. However, since the original SPEA2 requires substantial computation and memory space, modified SPEA2 [15,16] is used, which approximates crowdedness around a solution with the Euclidean distance from the solution to the other solution nearest to it in the normalized objective space.

PDR. Gradients are approximated by forward difference with the difference of 10^{-4}. To move a solution in a search direction back onto active constraint boundaries, steepest descent method is used. For linear search, golden section method is used, with the closed linear search interval length of 10^{-2}, the maximum number of extension of the interval of 20, and the basic number of iterations of 20. When active constraints are $\hat{C}^{\mathrm{u}} = \{\hat{c}^{\mathrm{u}}_1, \ldots, \hat{c}^{\mathrm{u}}_{\hat{P}^{\mathrm{u}}}\}$, $\hat{C}^{\mathrm{u}}$ and $\{\hat{c}^{\mathrm{u}}_i\}$ for each $i = 1, 2, \ldots, \hat{P}^{\mathrm{u}}$ are considered for inactivation. In order to accommodate a solution violating 30 linear constraints, search direction calculation and linear search are applied at most 30 times. Unrepairable infeasible solutions are discarded. Infeasible initial solutions are repaired in the Lamarckian way for PDR(nl) as well.

OCV. Since diversity w.r.t. constraint violations is unnecessary, sharing is applied in the original objective space.

4.2 Results

Figure 4 shows the transitions of GD and D1R, averaged over 10 trials, against the number of objective function evaluations when GA with constraint-handing methods are applied to the benchmark problems. For the methods that diverged GD, the lines of both GD and D1R are omitted.

BNH and TNK. Regarding GD, PDR(nl) performed the best, followed by PDR(l). PDR(nl) performed better because part or all of the Pareto-optimal solutions of these problems lie on boundaries, and GA's search for the Pareto-optimal solutions was better facilitated by PDR(nl), as explained in Sect. 3.6. No performance difference regarding D1R can be observed between PDR(l) and PDR(nl).

DP and OCV(ne) performed worse than PDR w.r.t. GD since it is difficult for DP to search for solutions on boundaries, and OCV(ne) behaves practically the same as DP, as explained in Sect. 2.2. Regarding D1R, PDR performed no worse than DP and OCV(ne).

On BNH, OCV(e) performed the worst. This is because OCV(e) maintains infeasible solutions throughout the entire search. OCV(e) performed poorly in D1R because its GD is not good. OCV(e) diverged GD on TNK. The entire third quadrant of TNK is Pareto-optimal w.r.t. constraint violations and objective functions, and solutions in the second and fourth quadrants can also be non-inferior. Therefore, ranking did not function on TNK, and crossover and sharing dispersed solutions.

ZDT2. Again, PDR(nl) performed the best regarding both GD and D1R since PDR(nl) better facilitates the search of the Pareto-optimal solutions on boundaries than DP and PDR(l) do, as explained in Sect. 1 and Sect. 3.6, respectively. OCV diverged GD as predicted in Sect. 2.2. Since this was observed despite the strong interpolating property of UNDX, similar results are expected to be observed when other less interpolative crossovers are used.

On the Whole. Experimental results confirmed that GA performs the best when PDR is used, which requires additional computational complexity comparable to that of linear programming solvers. They have also shown that OCV(ne) exhibits performance similar to that of DP on low dimensional problems, and OCV(e) and OCV(ne) can disperse solutions on problems with many constraints. In addition, it has been confirmed that, on problems whose Pareto-optimal solutions lie on boundaries, GA's search is better facilitated and solutions are obtained with higher precision when PDR is applied in the non-Lamarckian way.

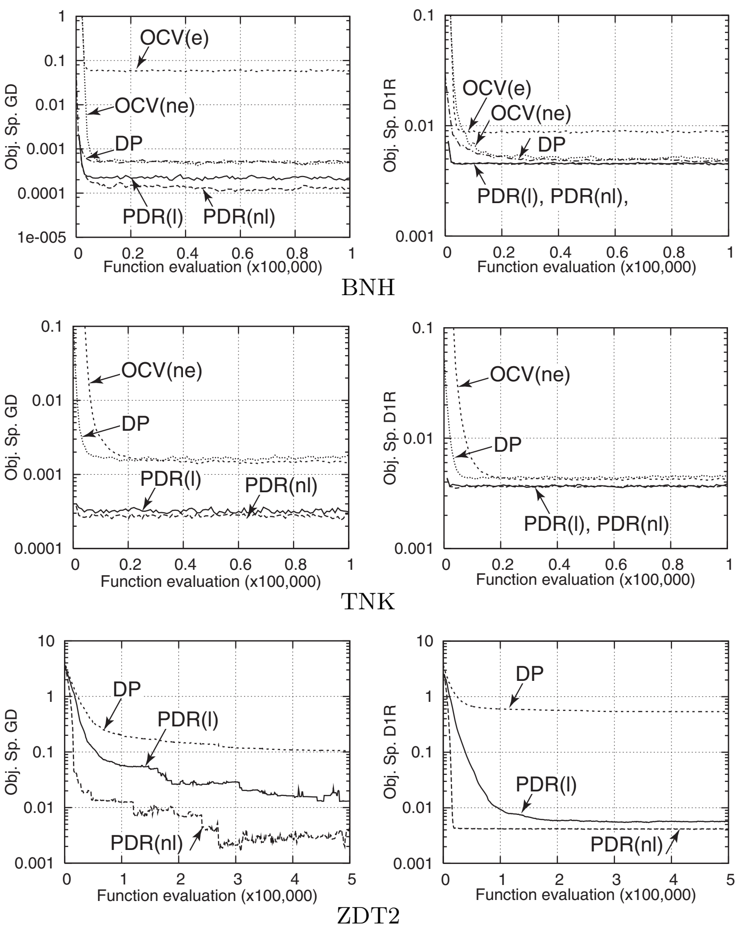

Fig. 4. Transitions of GD and D1R, averaged over 10 trials, when GA combined with constraint-handling methods are applied to the benchmark problems

5 Conclusions

This paper first presented the guidelines for designing effective constraint-handling methods. It then proposed Pareto Descent Repair operator (PDR) that meets these guidelines. PDR's effectiveness was verified through experiments comparing it with other constraint-handling methods. It was also confirmed that Pareto-optimal solutions on the boundaries are obtained with higher precision when PDR is applied in the non-Lamarckian way.

Although this paper proposed PDR as a repair operator for MOO, it can also be applied to single-objective optimization problems. Hence, it remains to investigate the effectiveness of PDR on single-objective optimization problems.

References

1. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
2. Coello, C.A.C.: Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering **191**(11-12) (2002) 1245–1287
3. Knowles, J.D., Corne, D.W.: Memetic algorithms for multiobjective optimization: issues, methods and prospects. In Krasnogor, N., Smith, J.E., Hart, W.E., eds.: Recent Advances in Memetic Algorithms. Springer (2004) 313–352
4. Coello, C.A.C.: Treating constraints as objectives for single-objective evolutionary optimization. Engineering Optimization **32**(3) (2000) 275–308
5. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186** (2000) 311–338
6. Oyama, A., Shimoyama, K., Fujii, K.: New constraint-handling method for multi-objective multi-constraint evolutionary optimization and its application to space plane design. In Schilling, R., Haase, W., Periaux, J., Baier, H., Bugeda, G., eds.: Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2005). (2005) 416–428
7. Michalewicz, Z., Nazhiyath, G.: Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In: Proceedings of the 2nd IEEE International Conference on Evolutionary Computation. Volume 2. (1995) 647–651
8. Harada, K., Sakuma, J., Ikeda, K., Ono, I., Kobayashi, S.: Local search for multiobjective function optimization: Pareto descent method (in Japanese). Transactions of the Japanese Society for Artificial Intelligence **21**(4) (2006) 340–350
9. Harada, K., Sakuma, J., Ikeda, K., Ono, I., Kobayashi, S.: Local search for multiobjective function optimization: Pareto descent method. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006), New York, NY, ACM Press (2006) 659–666
10. Luenberger, D.G.: Linear and Nonlinear Programming. Second edn. Addison-Wesley, Reading, MA (1984)
11. Gellert, W., Gottwald, S., Hellwich, M., Kästner, H., Künstner, H., eds.: VNR Concise Encyclopedia of Mathematics. Van Nostrand Reinhold, New York (1989)
12. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. Second edn. MIT Press (2001)
13. Ono, I., Kobayashi, S.: A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. In: 7th International Conference on Genetic Algorithms (ICGA7). (1997) 246–253
14. Knowles, J.D., Corne, D.W.: On metrics for comparing non-dominated sets. In: Proceedings of the 2002 Congress on Evolutionary Computation Conference (CEC02), IEEE Press (2002) 711–716
15. Harada, K., Sakuma, J., Kobayashi, S., Ikeda, K., Ono, I.: Hybridization of genetic algorithm and local search in multiobjective function optimization: Recommendation of GA then LS. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2006), New York, NY, ACM Press (2006) 667–674
16. Harada, K., Ikeda, K., Sakuma, J., Ono, I., Kobayashi, S.: Hybridization of genetic algorithm with local search in multiobjective function optimization: Recommendation of GA then LS (in Japanese). Transactions of the Japanese Society for Artificial Intelligence **21**(6) (2006) 482–492

17. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In Giannakoglou, K., et al., eds.: EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems. (2001) 12–21
18. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the Congress on Evolutionary Computation (CEC-2002). (2002) 825–830
19. Fliege, J., Svaiter, B.F.: Steepest descent methods for multicriteria optimization. Mathematical Methods of Operations Research **51**(3) (2000) 479–494

Appendix: Pareto Descent Method

Denote the normalized gradients of objective functions at solution $\boldsymbol{x}$ by $\bar{\nabla} f_i(\boldsymbol{x})$ $(i = 1, 2, \ldots, M)$. If a direction $\boldsymbol{d} \in \mathbb{R}^N$ satisfies

$$\boldsymbol{d} \cdot (-\bar{\nabla} f_i(\boldsymbol{x})) \geq 0 \ (i = 1, 2, \ldots, M) \ , \tag{11}$$

all objective functions can be simultaneously decreased by moving $\boldsymbol{x}$ in direction $\boldsymbol{d}$. Such directions are called *descent directions* for MOO. There are often multiple descent directions. The descent directions to which no other descent directions are superior in improving all objective functions are called *Pareto descent directions* [8,9]. There are often multiple Pareto descent directions. A descent direction is a Pareto descent direction if it can be expressed as a convex combination of the steepest descent directions of objective functions. Descent directions and Pareto descent directions of a 2-variable-2-objective problem are shown in Fig. 5.

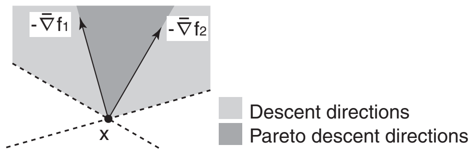

Fig. 5. Descent directions and Pareto descent directions of a 2-variable-2-objective problem

Since objective functions can be efficiently decreased by searching in Pareto-descent directions, several methods that calculates such directions were proposed in recent years, which include Multi-objective Steepest Descent Method (MSDM) [19] and Pareto Descent Method (PDM) [8,9]. PDM calculates feasible Pareto descent directions or descent directions, as appropriate, by solving linear programming problems, which has less computational complexity than MSDM does. Therefore, PDM can both effectively and efficiently decrease all objective functions simultaneously.