Hyeongmin Han
Industrial and Enterprise Systems Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: hhan19@illinois.edu

Sehyun Chang
Global R&D Master, Automotive R&D Division,
Hyundai Motor Company,
Namyang-eup,
Hwaseong-si 18280,
Gyeonggi-do, Korea
e-mail: suschang@hyundai.com

Harrison Kim[1]
Professor
Industrial and Enterprise Systems Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: hmkim@illinois.edu

Multiple Target Exploration Approach for Design Exploration Using a Swarm Intelligence and Clustering

In engineering design problems, performance functions evaluate the quality of designs. Among the designs, some of them are classified as good designs if responses from performance functions satisfy a target point or range. An infinite set of good designs in the design space is defined as a solution space of the design problem. In practice, since the performance functions are analytical models or black-box simulations which are computationally expensive, it is difficult to obtain a complete solution space. In this paper, a method that finds a finite set of good designs, which is included in a solution space, is proposed. The method formulates the problem as optimization problems and utilizes gray wolf optimizer (GWO) in the way of design exploration. Target points of the exploration process are defined by clustering intermediate solutions for every iteration. The method is tested with a simple two-dimensional problem and an automotive vehicle design problem to validate and check the quality of solution points. [DOI: 10.1115/1.4043201]

1 Introduction

Engineering design problems can be formulated as traditional optimization problems which consist of design variables, constraints, and an objective function (or objective functions for multi-objective optimization problems). In the problems, performance functions that evaluate the quality of the products can be involved in the constraints or objective functions. For example, in product family design, the objectives in the problem are often maximizing performance and minimizing costs. The platform-design problem can be formulated as one comprehensive optimization problem with one objective where the multiple objectives are combined into one function [1]. There can be multiple objective functions in product family design to obtain a set of Pareto-optimal solutions [2]. For the engineering design problems that have multiple disciplines, they are decomposed into subproblems that represent different disciplines. The distributed problems can be obtained from multidisciplinary design optimization (MDO) techniques, such as collaborative optimization [3], bilevel integrated system synthesis [4], and analytical target cascading [5]. However, in the early stage of an engineering design process, the target of performance functions and the boundary of design variables are not certain. They can be changed during the design process because of the uncertainty in real-world situations. Also, designers may focus on the feasible region of the design space rather than finding one optimal solution.

Instead of the point-based design, which finds the best design point in iterative processes, set-based design can be adapted to answer the challenge. When the parameters and the target for the design process are assured, the point-based design can be used. However, in practice, the design optimization problem is uncertain because of the variability of the parameters in the system. With set-based design, designers can keep multiple solutions and narrow down the set of candidate designs by removing unnecessary solutions [6–8]. Since designers can interact during the design process, they can gain insight from the intermediate solutions of the process and have a better understanding of the system. Designers can even adjust the parameters of the problem based on the information [9]. This concept has been adopted to optimization problems with various applications, such as MDO [10], aerospace industry [11], and high-rise building structures [12]. In this paper, we propose a method that finds a finite set of feasible designs using design exploration which would be a part of set-based design approaches. K-means clustering and gray wolf optimizer (GWO) are utilized in the way of design exploration. The related research is discussed in Secs. 1.1 and 1.2.

1.1 Design Exploration.

As the set-based design starts from finding the feasible designs and solution space in the design space, we reviewed the literature about design exploration in this section. The solution space is a region in the design variable space that satisfies the target range and constraints, which is the result of design-exploration processes. Instead of having one best point, designers will get an infinite set of solutions so that they can easily adapt to a new target. There are incomplete and complete methods of having a solution space. Obviously, complete methods will provide more solutions than incomplete methods, but it takes a much longer time to solve the problem.

One of the ways to achieve the solution spaces is from algorithms for constraint satisfactory problem (CSP). CSP is a mathematical problem that consists of variables, domain, and constraints. The goal of the problem is to find values for the variables that are in the domain and satisfy the constraints. In most cases, researchers have focused on CSP with discrete variables [13]. However, for design optimization problems, continuous variables are mostly used. In order to use the discrete CSP, Lottaz et al. [14] transformed the continuous CSP into discrete CSP and used discrete CSP techniques directly. Also, Yannou et al. [15] combined a CSP technique and Monte Carlo simulation which is inefficient when the system is highly constrained. Instead of applying Monte Carlo directly, the authors used the CSP technique first with incomplete approach since obtaining the complete set of the solution space is computationally expensive. After getting the incomplete solution space, they performed Monte Carlo simulation. One of the algorithms for continuous CSP is an interval partitioning method [16]. This algorithm partitions the domains of variables into smaller boxes so that they can be determined if feasible or infeasible. For CSP techniques, the functions in the problem formulation should be formed as analytical expressions. Moreover, it is difficult to solve the problem with high-dimensional functions.

In engineering design problems, Pareto frontiers can be used to get the solution space. Mattson and Messac [17] proposed a

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received April 8, 2018; final manuscript received March 8, 2019; published online April 22, 2019. Assoc. Editor: Mian Li.

method to find Pareto frontiers of multi-objective problems. With this method, the authors can obtain a set of solutions that maximize performance. The formulation in this paper has a lower and upper bound of responses, while the performance is maximized in Ref. [17]. Another method, isoperformance [18], is a systematic method to obtain the solution space and select the best design from the solution space. The authors not only maximize the performance but also consider other criteria such as cost and risk. There are three steps in the method: obtaining a performance-invariant set of solutions using a gradient-based approach, eliminating some of the solutions based on noninferiority criteria (i.e., only Pareto-optimal points remain), and discussing the designs in the solution space to finalize the design. However, the step size in the first step would be too small to explore a wide range of space. Also, they formulated the problem with a target point, but we focus on the problem that contains a target range. Monte Carlo simulation is another method that the designers can use. Eichstetter et al. [19] proposed a method that provides a box-shaped solution space. With the step by step process, designers can find an incomplete set of good designs. The set is a Cartesian product of lower and upper bounds, which creates a box-shaped space. Since the final solution of the method is a box with lower and upper bounds, the solution cannot cover two different islands in the solution space. Also, if the shape of the feasible region is not convex, the box-shaped solution could be missing many solutions.

1.2 Gray Wolf Optimizer. In this paper, a method is proposed that utilizes a derivative-free method to find good design points that satisfy the target range. Evolutionary algorithms (EA) are one of the widely used algorithms for exploration and optimization. The algorithms are inspired by biological evolution. It starts with a random population, and the quality of each individual is evaluated by a fitness function. For each iteration, the population is evolved using crossover and mutation. EA includes genetic algorithm (GA) [20], genetic programming [21], and evolutionary strategy [22]. Another method is swarm intelligence. It imitates biological systems such as bird swarms and ant colonies. Some of the methods include ant colony optimization [23], particle swarm optimization (PSO) [24], and GWO [25].

In the proposed method, GWO is chosen because of its iteration process. Different from other methods, we do not have to wait until it converges. The algorithm starts with a fixed number of agents and maximum iterations, and it stops at the maximum iteration. Therefore, we can calculate the exact number of function evaluations with GWO.

Algorithm 1 GWO

Input: Number of search agents (n), maximum number of GWO iterations ($t_{\max}$)
procedure GWO(n, $t_{\max}$)
  Initialize $\mathbf{X}_i(1)$ ($i = 1, 2, \ldots, n$)
  Calculate the fitness for each $\mathbf{X}_i(1)$
  $\mathbf{X}_\alpha \leftarrow$ the best agent
  $\mathbf{X}_\beta \leftarrow$ the second best agent
  $\mathbf{X}_\delta \leftarrow$ the third best agent
  $t \leftarrow 1$
  while $t \le t_{max}$ **do**
    for $i = 1, \ldots, n$ **do**
      Update the agent as in Eq. (7)
      Calculate the fitness for $\mathbf{X}_i(t+1)$
    end for
    Update $\mathbf{a}$, $\mathbf{A}$, and $\mathbf{C}$
    Update $\mathbf{X}_\alpha$, $\mathbf{X}_\beta$, and $\mathbf{X}_\delta$
    $t \leftarrow t + 1$
  end while
  return $\mathbf{X}_\alpha$
end procedure

GWO is a metaheuristic method that computationally simulates gray wolf hunting for the prey [25]. The social hierarchy of wolves is mathematically modeled by categorizing all the gray wolves into four types of wolves. The wolf who has the best fitness value is denoted as alpha (α). The second and third best wolves are named as beta (β) and delta (δ), respectively. The rest of the wolves are considered as omega (ω). In GWO, the search process of the ω wolves is guided by α, β, and δ. The group hunting process that encircles the prey (the optimal solution of an optimization problem) is modeled as follows:

$$\mathbf{D} = \mathbf{C} \circ \mathbf{X_p}(t) - \mathbf{X}(t) \tag{1}$$

$$\mathbf{X}(t+1) = \mathbf{X_p}(t) - \mathbf{A} \circ \mathbf{D} \tag{2}$$

where $\circ$ indicates the Hadamard product, t represents the current GWO iteration, $\mathbf{X_p}$ represents the position vector of the prey, and $\mathbf{X}$ denotes the position vector of a gray wolf. The vectors $\mathbf{A}$, $\mathbf{C}$, and $\mathbf{D}$ are intermediate vectors that imply randomness of the process. The intermediate vectors have the same dimension as the dimension of $\mathbf{X_p}$ and $\mathbf{X}$. The vectors $\mathbf{A}$ and $\mathbf{C}$ are defined as follows:

$$\mathbf{A} = 2\mathbf{a} \circ \mathbf{r_1} - \mathbf{a} \tag{3}$$

$$\mathbf{C} = 2\mathbf{r_2} \tag{4}$$

where $\mathbf{a}$ is a vector of which the elements are linearly decreased from 2 to 0 during the optimization process, and $\mathbf{r_1}$ and $\mathbf{r_2}$ are the random vectors whose elements are in [0, 1]. As it is shown in Eq. (2), wolves are updated toward the prey. However, the prey is unknown during the search process in mathematical models. The prey is estimated by α, β, and δ since it is assumed that the three best solutions have better knowledge about the location of the prey [25]. The updating scheme of each wolf is as follows. Also, the algorithm of GWO is described in Algorithm 1.

$$\mathbf{D}_\alpha = \mathbf{C_1} \circ \mathbf{X}_\alpha - \mathbf{X}, \quad \mathbf{D}_\beta = \mathbf{C_2} \circ \mathbf{X}_\beta - \mathbf{X}, \quad \mathbf{D}_\delta = \mathbf{C_3} \circ \mathbf{X}_\delta - \mathbf{X} \tag{5}$$

$$\mathbf{X_1} = \mathbf{X}_\alpha - \mathbf{A_1} \circ \mathbf{D}_\alpha, \quad \mathbf{X_2} = \mathbf{X}_\beta - \mathbf{A_2} \circ \mathbf{D}_\beta, \quad \mathbf{X_3} = \mathbf{X}_\delta - \mathbf{A_3} \circ \mathbf{D}_\delta \tag{6}$$

$$\mathbf{X}(t+1) = \frac{\mathbf{X_1} + \mathbf{X_2} + \mathbf{X_3}}{3} \tag{7}$$

In this paper, a concept that provides a finite set of good designs is proposed. Numerical experiments for a small problem and an automotive vehicle design are performed to test the proposed method. The rest of the paper is organized as follows. In Sec. 2, the proposed method is explained in detail. In Sec. 3, a two-dimensional problem and a bicycle model are tested for the numerical experiment. Finally, we present the discussion and conclusion in Secs. 4 and 5, respectively.

2 Methods

2.1 Problem Statement. The main goal of this paper is to introduce a new method that provides a finite set of good designs. If a design point in the design space satisfies the boundary of design variables and the response of the point is in the target range, we define it as a "good" design. Otherwise, it is a "bad" design. A set of feasible designs will give designers a wide range of alternatives for the final decision. The proposed method can solve engineering design problems with a special structure, in which responses of the performance functions have two-sided

bounds. The purpose of the method is obtaining a set of feasible designs, which satisfies solution space $\mathcal{F}$ which is as follows:

$$\mathcal{F} = \{\mathbf{x} \mid \mathbf{x}^{\mathbf{L}} \le \mathbf{x} \le \mathbf{x}^{\mathbf{U}}, \quad \mathbf{T}^{\mathbf{L}} \le \mathbf{R}(\mathbf{x}) \le \mathbf{T}^{\mathbf{U}}\} \tag{8}$$

In Eq. (8), $\mathbf{x}$ is a vector of design variables and $\mathbf{R(x)}$ represents the performance functions. There are N number of design variables and M number of performance functions in the problem. A lower and upper bound of the design variables are represented as $\mathbf{x}^{\mathbf{L}}$ and $\mathbf{x}^{\mathbf{U}}$, and they are called a boundary of the design variables. Also, lower and upper bounds of the responses are denoted as $\mathbf{T}^{\mathbf{L}}$ and $\mathbf{T}^{\mathbf{U}}$, which are called a target range in this paper.

2.2 Proposed Methods: Multiple Target Exploration. The proposed method has three features that make GWO sample good design points efficiently. First, in order to use GWO as a sampling technique, a filtering process for every intermediate solution is added to the algorithm. Second, we modified existing updating scheme in GWO for better exploration. Lastly, a clustering method is used to obtain target points for design exploration and define fitness functions of the algorithm. In the method, penalty function, $\phi(\mathbf{y})$, is used as a fitness function of the problem. Either the constraints for the boundary of the design variables, $\mathbf{x} \ge \mathbf{x}^{\mathbf{L}}$ and $\mathbf{x} \le \mathbf{x}^{\mathbf{U}}$, or the target range of responses, $\mathbf{y} \ge \mathbf{T}^{\mathbf{L}}$ and $\mathbf{y} \le \mathbf{T}^{\mathbf{U}}$, can be relaxed to define the penalty function. The constraints for the boundary are hard constraints that must be satisfied in the problem because these constraints have physical meanings in most cases. However, the constraints $\mathbf{y} \ge \mathbf{T}^{\mathbf{L}}$ and $\mathbf{y} \le \mathbf{T}^{\mathbf{U}}$ can be considered as soft constraints. As responses of performance functions represent a quality of design and the target range is determined by designers, the target range can be violated. An objective function, or a fitness function with relaxed constraints, is as follows:

$$\phi(\mathbf{R}(\mathbf{x}); \mathbf{T}^{\mathbf{L}}, \mathbf{T}^{\mathbf{U}}) = \sum_{j}^{M} \max\{\mathbf{R}_j(\mathbf{x}) - \mathbf{T}_j^{\mathbf{U}}, \mathbf{T}_j^{\mathbf{L}} - \mathbf{R}_j(\mathbf{x}), 0\} \tag{9}$$

In the proposed method, however, different penalty functions other than Eq. (9) are used. Instead of using $\mathbf{T}^{\mathbf{L}}$ and $\mathbf{T}^{\mathbf{U}}$ to define the penalty function, a target point, $\mathbf{T}$, that is in the target range ($\mathbf{T}^{\mathbf{L}} \le \mathbf{T} \le \mathbf{T}^{\mathbf{U}}$) is selected to construct a penalty function as in Eq. (10). The target points are obtained by using a clustering method.

$$\phi(\mathbf{R}(\mathbf{x}); \mathbf{T}) = |\mathbf{R}(\mathbf{x}) - \mathbf{T}| \tag{10}$$

With the new penalty function, GWO keeps updating toward the target point even if the agents are in the target range. When Eq. (9) is used as a fitness function, three best agents are not updated once it finds three good designs because $\mathbf{X}_\alpha$, $\mathbf{X}_\beta$, and $\mathbf{X}_\delta$ are updated only if there is an agent that has lower fitness value than them. They cannot become different points once they are in the target range (i.e., fitness values equal to 0). However, with the new penalty function, they keep being updated. The detailed explanation for three important features of the method is as follows:

(1) *Adding the filtering process for every intermediate solution.* We consider all the intermediate points during the optimization process rather than merely taking the final solution. In the method, the searching process of the optimization problem is considered as a sampling process that finds multiple feasible points in $\mathcal{F}$. For example, when GWO solves an optimization problem which has Eq. (9) as an objective function, it evaluates all the fitness values for agents. During the GWO iteration process, if the fitness is 0 (the point is in the target range), we add the point to the set of good designs, $\mathcal{G}$, which has points that satisfy all the target ranges. Otherwise, the point is added to the set of bad designs, $\mathcal{B}$. Therefore, the final results of the proposed methods are two sets of points. Algorithm 2 is the GWO that is modified to obtain two sets of points that contain good and bad designs, respectively.

Algorithm 2 GWO for design exploration 1 (GWODE1)

Input: Number of search agents (n), maximum number of GWO iterations ($t_{\max}$)
procedure GWODE1 (n, $t_{\max}$)
  $\mathcal{G} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$
  Initialize $\mathbf{X}_i(1)$ ($i = 1, 2, \ldots, n$)
  Calculate the fitness for each $\mathbf{X}_i(1)$ as in Eq. (9)
  $\mathbf{X}_\alpha \leftarrow$ the best agent
  $\mathbf{X}_\beta \leftarrow$ the second best agent
  $\mathbf{X}_\delta \leftarrow$ the third best agent
  $t \leftarrow 1$
  while $t \le t_{max}$ **do**
    for $i = 1, \ldots, n$ **do**
      if $\mathbf{T}^{\mathbf{L}} \le \mathbf{R}(\mathbf{X}_i(t)) \le \mathbf{T}^{\mathbf{U}}$ **then**
        $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathbf{X}_i(t)\}$
      else
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{X}_i(t)\}$
      end if
      Update the agent as in Eq. (7)
      Calculate the fitness for $\mathbf{X}_i(t+1)$ as in Eq. (9)
    end for
    Update $\mathbf{a}$, $\mathbf{A}$, and $\mathbf{C}$
    Update $\mathbf{X}_\alpha$, $\mathbf{X}_\beta$, and $\mathbf{X}_\delta$
    $t \leftarrow t + 1$
  end while
  return $\mathcal{G}$, $\mathcal{B}$
end procedure

(2) *Modifying updating scheme of vectors in GWO.* The updating scheme for vectors in GWO is modified for the design exploration process in two ways. First, the vector $\mathbf{a}$ decreases only if certain conditions are satisfied. In the original GWO, elements of the vector $\mathbf{a}$ are linearly decreased from 2 to 0. For the vector $\mathbf{A}$, if $|\mathbf{A}| > \mathbf{1}$, agents diverge from the prey; otherwise, they gather toward the prey. Since the agents in design exploration do not want to gather into a single point, elements in the vector $\mathbf{a}$ do not have to be decreased to 0. We added a conditional statement for updating the vectors and parameters. With a parameter s, shrink tolerance, if the number of good agents in a GWO iteration is greater than or equal to the total number of agents times s and the elements in the vector $\mathbf{a}$ are less than or equal to 1, the vectors and parameters are not updated. For the iterations with 10 agents and $s = 0.9$, if the number of good agents in a GWO iteration is greater than or equal to 9 and the elements in $\mathbf{a}$ are less than or equal to 1, then $\mathbf{A}$, $\mathbf{a}$, $\mathbf{C}$, $\mathbf{X}_\alpha$, $\mathbf{X}_\beta$, and $\mathbf{X}_\delta$ are not updated. It means that we do not have to shrink the searching area anymore and sample the points by using the current intermediate vectors. Figure 1 shows a comparison between the proposed updating scheme (top) and the original GWO (bottom). In the figure, dots (•) and crosses (X) represent good points and bad points, respectively. With the original updating scheme, points converge toward a point while points in the upper figure do not.

The second modification for the updating scheme is about Eq. (7). The updated agent from Eq. (7) can violate the lower bound or upper bound of design variables. In the original GWO, for each dimension, if the value of the element is less than the lower bound, it is set as x_i^L. If it is greater than the upper bound, the value is set as x_i^U. That means that many borderline points are sampled when we use the original updating scheme. In the modification, we accepted the updated agent only if it is in the boundary of the design

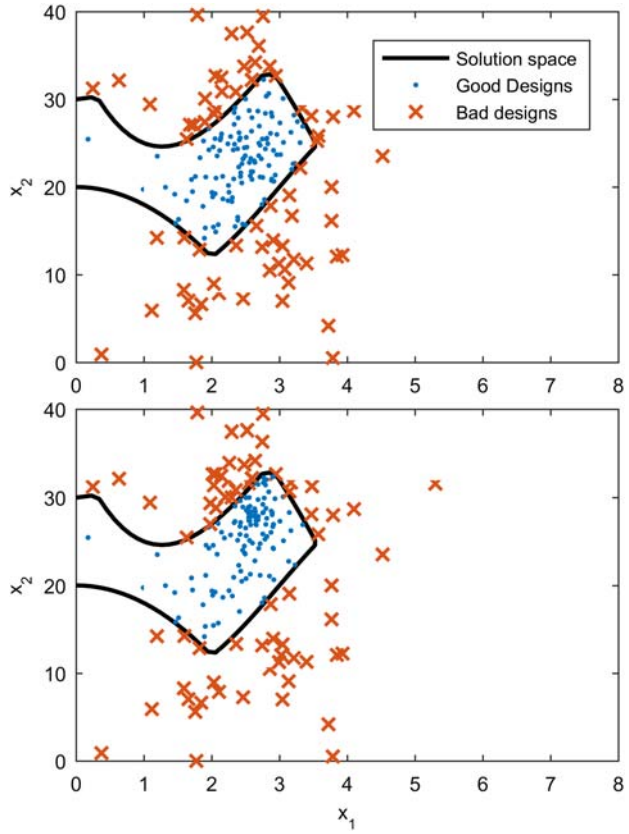

Fig. 1 The figures on top and bottom show intermediate points from the proposed method (top) and GWO (bottom), respectively

variables. For each dimension, corresponding elements of the vectors $\mathbf{r_1}$ and $\mathbf{r_2}$ in Eqs. (3) and (4) are re-evaluated until the updated value does not violate the bounds. Checking the feasibility of design variables is computationally much cheaper than evaluating fitness which includes performance functions. The algorithm with modified updating scheme is presented in Algorithm 3.

Algorithm 3 GWO for design exploration 2 (GWODE2)

Input: Number of search agents (n), maximum number of GWO iterations ($t_{\max}$), shrink tolerance (s)
procedure GWODE2 (n, $t_{\max}$,s)
  $\mathcal{G} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$
  Initialize $\mathbf{X}_i(1)$ ($i = 1, 2, \ldots, n$)
  Calculate the fitness for each $\mathbf{X}_i(1)$
  $\mathbf{X}_\alpha \leftarrow$ the best agent
  $\mathbf{X}_\beta \leftarrow$ the second best agent
  $\mathbf{X}_\delta \leftarrow$ the third best agent
  $t \leftarrow 1$
  while $t \le t_{max}$ **do**
    for $i = 1, \ldots, n$ **do**
      if $\mathbf{T}^{\mathbf{L}} \le \mathbf{R}(\mathbf{X}_i(t)) \le \mathbf{T}^{U}$ **then**
        $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathbf{X}_i(t)\}$
        $g \leftarrow g + 1$
      else
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{X}_i(t)\}$
      end if
      while $\mathbf{X}_i(t+1) \le \mathbf{x}^{\mathbf{U}}$ **or** $\mathbf{X}_i(t+1) \ge \mathbf{x}^{\mathbf{L}}$ **do**
        Update the agent as in Eq. (7)
      end while
      Calculate the fitness for $\mathbf{X}_i(t+1)$
    end for
    if $g < n \times s$ OR $\mathbf{a} > \mathbf{1}$ **then**
      Update $\mathbf{a}$, $\mathbf{A}$, and $\mathbf{C}$
      Update $\mathbf{X}_\alpha$, $\mathbf{X}_\beta$, and $\mathbf{X}_\delta$
    end if
  $t \leftarrow t + 1$
  end while
  return $\mathcal{G}$, $\mathcal{B}$
end procedure

(3) *Finding different target points by k-means clustering.*

For the first optimization problem, the target range defined by $\mathbf{T}^{\mathbf{L}}$ and $\mathbf{T}^{\mathbf{U}}$ is used for a fitness function as an initialization. From the initialization process, we can get a set of good points, $\mathcal{G}$. Points in $\mathcal{G}$ are clustered by k-means clustering and the centers of the clusters become the targets for the next iteration. Since the centers of the clusters exist in design space, we need to evaluate the responses for the centers, and the responses are set as target points.

K-means clustering is a data mining method that partitions observations into k clusters [26]. It evaluates the sum of point-to-centroid distances for each cluster and minimizes the sum of the evaluations. In order to validate the result from k-means clustering, a silhouette method is used [27]. Since the performance of the clustering from the k-means method depends on k, the number of clusters, the algorithm tries 1 to $k_{\max}$ clusters and evaluates the silhouette mean to find the best cluster. The target point, $\mathbf{T}$, is defined as the centers of the best clusters from k-means clustering and the silhouette method.

Algorithm 4 GWOMTE

Input: Number of search agents (n), maximum number of GWO iterations ($t_{\max}$), shrink tolerance (s), maximum number of explored target points ($p_{\max}$), target convergence tolerance (c), maximum number of clusters ($k_{\max}$)
procedure GWOMTE (n, $t_{\max}$, s, $p_{\max}$, c, $k_{\max}$)
  Define the fitness using $\mathbf{T}^{\mathbf{L}}$ and $\mathbf{T}^{\mathbf{L}}$ as in Eq. (9)
  $[\mathcal{G}, \mathcal{B}] \leftarrow$GWODE2($n$, t_{max}, s)
  $l \leftarrow 1$
  $p \leftarrow 2$
  while $\sum_j \frac{|\mathbf{T}^{\mathbf{x}}_{:j}(l-1) - \mathbf{T}^{\mathbf{x}}_{:j}(l)|}{|\mathbf{x}^{\mathbf{U}} - \mathbf{x}^{\mathbf{L}}|} \ge c$ **do**
    for $k = 1, \ldots, k_{\max}$ **do**
      $[\mathbf{I}^k, \mathbf{T}^k] \leftarrow$KMEANS($\mathcal{G}$, k)
      $ms^k \leftarrow$mean(SILHOUETTE($\mathcal{G}$, $\mathbf{I}^k$))
      $ms^{max}(l) \leftarrow argmax_k\{ms^k\}$
    end for
    $\mathbf{T}^{\mathbf{x}}(l) \leftarrow \mathbf{T}^{ms^{max(l)}}$
    Define $\mathbf{T}^{\mathbf{LIST}}(l)$ as responses of $\mathbf{T}^{\mathbf{x}}(l)$
    for $j = 1, \ldots, ms_{\max(l)}$ **do**
      if $p \ge p_{\max}$ **then**
        Stop GWOMTE
      end if
      Get the fitness using $\mathbf{T}^{\mathbf{LIST}}_{:j}(l)$ as in Eq. (10)
      $[\mathcal{G}, \mathcal{B}] \leftarrow [\mathcal{G}, \mathcal{B}] \cup$GWODE2($n$, $t_{\max}$, s)
      $p \leftarrow p + 1$
    end for
    $l \leftarrow l + 1$
  end while
  return $\mathcal{G}$, $\mathcal{B}$
end procedure

With the modifications, the proposed algorithm, GWO for multiple target exploration (GWOMTE), is presented in Algorithm 4. There are six parameters in the algorithm. The number of search agents (n) and the maximum number of GWO iterations ($t_{\max}$) are parameters for the original GWO. Shrink tolerance (s) defines the exploration and exploitation in the updating scheme. The maximum number of explored target points ($p_{\max}$) and target

convergence tolerance (c) are needed for stopping criteria. Also, the parameter $p_{\max}$ defines the maximum time resource that is allowed to designers because we can calculate the maximum number of function evaluations. Since the algorithm evaluates performance functions $n \times t_{\max}$ number of times for every target point, $n \times t_{\max} \times p_{\max}$ is the maximum number of function evaluations. Lastly, the maximum number of clusters ($k_{\max}$) determines a quality of k-means clustering.

The algorithm starts with GWODE2 where the fitness function is defined with the target range. The set of good design points is clustered by k-means clustering with different numbers of clusters, k, and the silhouette method is used to find the best cluster among $k_{\max}$ different clustering results. The centers of the best clusters are defined as target points. The target list of the lth iteration is defined by evaluating all the centers with the performance functions, $\mathbf{R(x)}$. For each target point, it defines the fitness function and GWODE2 is applied. There are two different stopping criteria. If the centers of clusters converge to certain points, the algorithm stops. The test of convergence is performed only if the number of clusters is the same as the number of clusters in the previous iteration. Also, if the number of target points that are used to define fitness functions is greater than $p_{\max}$, it stops.

3 Numerical Experiments

Numerical experiments are performed to test the proposed methods. Two different problems are tested in this section. The first problem is a two-dimensional problem with polynomial performance functions. The good points are displayed in graphs to see the spread of the solutions. The second problem is a two-degree-of-freedom bicycle model. There are seven design variables and five performance functions. For performance functions, closed-form expressions are used.

The purpose of the numerical evaluation is verifying GWOMTE and comparing with GWODE1 by measuring the quality of good points in the set $\mathcal{G}$. To evaluate the quality of points, four metrics are used: the rate of good designs (r_g), the rate of unique designs (r_u), spread (E_0), and spread of standardized data ($E_{0,\text{std}}$). r_g counts the number of good designs which is easily obtained from the cardinality of $\mathcal{G}$. For r_u, unique designs are defined as good design points that are not close to other good design points. In this numerical experiment, if the two points are in the same hypercube where its minimum distance between two vertices is 0.1, they are considered as the same points. That means we ignore all the digits after the second decimal number and remove redundant points. Spread is proposed by Willerton [28]. It measures the size of metric spaces. The spread $E_0(X)$ is defined by as follows:

$$E_0(Y) = \sum_{y \in Y} \frac{1}{\sum_{y' \in Y} \mathrm{e}^{-d(y,y')}} \tag{11}$$

In the definition, Y is a finite set and d is a metric. In this case, we use Euclidean distance as the metric. According to the paper by Willerton, the spread has the following properties [28]:

- $1 \le E_0(Y) \le |Y|$
- $E_0(tY)$ is increasing in t
- $E_0(tY) \to 1 \quad \text{as} \quad t \to 0$
- $E_0(tY) \to |Y| \quad \text{as} \quad t \to \infty$
- $E_0(Y) \le e^{\text{diam}(Y)}$

As it is shown in the first property, the lower bound of spread is 1 and the upper bound depends on the number of points in Y. $E_{0,\text{std}}$ also uses Eq. (11), but the good design points are standardized with x^L and x^U.

The numerical experiment starts with solving the problem using GWOMTE. With being solved by GWOMTE, we can get the number of explored target points, p^*, that shows how many times GWODE2 is used for GWOMTE. After GWOMTE is tested, we run GWODE1 for p^* times. By doing so, we evaluate the same number of points for both GWOMTE and GWODE1, which is equal to $(n \times t_{\max} \times p^*)$. For the results from GWOMTE and GWODE1, four metrics (r_g, r_u, E_0, and $E_{0,\text{std}}$) are calculated to compare the quality of solutions.

3.1 Problem 1: Two-Dimensional Problem. A simple two-dimensional problem is tested to visualize the set of good designs. The solution space of the problem is as follows:

$$\begin{aligned} \mathcal{F}^1 = \{\mathbf{x} \mid\ & T_1^L \le R_1(x_1, x_2) \le T_1^U, \\ & T_2^L \le R_2(x_1, x_2) \le T_2^U, \\ & T_3^L \le R_3(x_1, x_2) \le T_3^U, \\ & x_1^L \le x_1 \le x_1^U, \\ & x_2^L \le x_2 \le x_2^U\} \\ \text{where}\quad & R_1(x_1, x_2) = -x_1 + x_2 \\ & R_2(x_1, x_2) = 2x_1^2 + x_2 \\ & R_3(x_1, x_2) = x_1(x_1 - 3)(x_1 - 6) + x_2 \end{aligned} \tag{12}$$

There are two design variables, x_1 and x_2, and three performance functions, R_1, R_2, and R_3. Performance functions are polynomial functions which are linear, quadratic, and cubic. For the numerical experiments, the boundary of the design variables is fixed as $x_1^L = 0$, $x_1^U = 10$, $x_2^L = 0$, and $x_2^U = 150$. Three different target ranges are tested, and the problems with different target ranges are denoted as P1A, P1B, and P1C. The target range for each problem is shown in Eq. (13).

The solution space of P1A has one connected set, P1B has two disconnected sets, and P1C has three disconnected sets. We set the parameters as $n = 10$, $t_{\max} = 20$, $p_{\max} = 500$, $c = 2 \times 10^{-3}$, and $k_{\max} = 10$. For the shrink tolerance s, 11 different values are tested (0, 0.1, 0.2, … , 1). The results are summarized in Table 1. For each shrink tolerance, we tested 100 different random seeds and the results shown in the table are the averages of 100 outputs.

$$\begin{aligned} \text{P1A:}\quad & T_1^L = 10, \quad T_1^U = 30, \quad T_2^L = 20, \quad T_2^U = 50, \\ & T_3^L = 20, \quad T_3^U = 35 \\ \text{P1B:}\quad & T_1^L = 10, \quad T_1^U = 25, \quad T_2^L = 20, \quad T_2^U = 120, \\ & T_3^L = 20, \quad T_3^U = 35 \\ \text{P1C:}\quad & T_1^L = 20, \quad T_1^U = 25, \quad T_2^L = 20, \quad T_2^U = 120, \\ & T_3^L = 25, \quad T_3^U = 30 \end{aligned} \tag{13}$$

Table 1 shows comparisons between GWOMTE and GWODE1. The bold values in the table represent the better performance between the two methods. In the results, both GWOMTE and GWODE1 performed better than brute-force search. r_u values of brute-force search for P1A, P1B, and P1C were 0.0245, 0.0286, and 0.0092, respectively. For the parameter s, as it decreased, r_g and r_u decreased but E_0 increased. As s gets close to 0, the points spread more from the target point rather than converge. That means smaller s sacrifices the probability of getting good points for the spread of the points. In terms of the spread metrics (E_0 and $E_{0,\text{std}}$), GWODE1 was better than GWOMTE. We think this is because of the stopping criteria of GWOMTE. Since the algorithm stops when the target points are not much changed, the points are sampled toward similar target points at the last stage of the algorithm. About the r_u, it is the highest when $s = 0.8$ for all the cases. For these problems, 0.8 is the best value for s where the algorithm obtains additional good points by using redundant points.

Figures 2 and 3 show a good example of GWOMTE. Only good points are shown in the figures. Each marker in the figure represents the number of the cluster. For example, in the upper figure in Fig. 2, there are two different markers ("1" and "2") which tell the cluster number that the design points belongs to. The plus signs (+) represent the centers of clusters. Figure 2 shows the first and second

Table 1 Result of the two-dimensional problem

			GWOMTE				GWODE1			
	s	p	r_g	r_u	E_0	$E_{0,\text{std}}$	r_g	r_u	E_0	$E_{0,\text{std}}$
P1A	1	9.96	**0.6033**	**0.4212**	10.7690	1.0843	0.5173	0.4028	**12.8870**	**1.1123**
	0.9	10.43	**0.5986**	**0.4376**	10.9580	1.0847	0.5170	0.3998	**12.8700**	**1.1112**
	0.8	10.57	**0.5851**	**0.4426**	11.2490	1.0873	0.5171	0.4002	**12.9230**	**1.1116**
	0.7	10.60	**0.5688**	**0.4385**	11.3810	1.0860	0.5166	0.3996	**12.8840**	**1.1108**
	0.6	10.68	**0.5552**	**0.4357**	11.4440	1.0854	0.5154	0.3980	**12.8040**	**1.1107**
	0.5	11.23	**0.5406**	**0.4231**	11.4870	1.0861	0.5170	0.3962	**12.9390**	**1.1115**
	0.4	11.04	**0.5311**	**0.4216**	11.5270	1.0859	0.5169	0.3968	**12.8400**	**1.1108**
	0.3	11.19	**0.5290**	**0.4221**	11.6060	1.0868	0.5169	0.3970	**12.8840**	**1.1111**
	0.2	11.50	**0.5289**	**0.4196**	11.6220	1.0862	0.5177	0.3949	**12.9830**	**1.1118**
	0.1	11.21	**0.5279**	**0.4211**	11.6080	1.0861	0.5173	0.3969	**12.9410**	**1.1113**
	0	11.10	**0.5294**	**0.4225**	11.5940	1.0855	0.5165	0.3967	**12.9340**	**1.1111**
P1B	1	11.93	**0.4768**	**0.3491**	13.1300	**1.1694**	0.4005	0.3253	**14.1570**	1.1683
	0.9	12.11	**0.4666**	**0.3583**	13.3930	**1.1729**	0.4004	0.3245	**14.1680**	1.1680
	0.8	11.93	**0.4624**	**0.3641**	13.3880	**1.1708**	0.4016	0.3262	**14.1160**	1.1667
	0.7	11.83	**0.4458**	**0.3569**	13.6270	**1.1722**	0.3981	0.3236	**14.1870**	1.1690
	0.6	12.13	**0.4383**	**0.3560**	13.6970	**1.1706**	0.3995	0.3238	**14.1730**	1.1678
	0.5	13.04	**0.4227**	**0.3443**	14.0790	**1.1754**	0.4008	0.3217	**14.2070**	1.1660
	0.4	12.5	**0.4129**	**0.3436**	14.1100	**1.1706**	0.4022	0.3251	**14.2210**	1.1677
	0.3	12.41	**0.4012**	**0.3350**	**14.2840**	**1.1690**	0.4008	0.3248	14.1350	1.1661
	0.2	12.62	0.3935	**0.3279**	**14.3310**	1.1667	**0.4034**	0.3261	14.2990	**1.1680**
	0.1	13.01	0.3855	0.3224	**14.5030**	1.1618	**0.4024**	**0.3240**	14.1770	**1.1657**
	0	12.46	0.3799	0.3183	14.2310	1.1515	**0.4023**	**0.3257**	**14.2550**	**1.1663**
P1C	1	10.12	**0.2721**	**0.1850**	6.2002	**1.1404**	0.2401	**0.1733**	**7.3907**	1.1661
	0.9	9.88	**0.2713**	**0.1877**	6.2145	**1.1396**	0.2403	**0.1747**	**7.4013**	1.1664
	0.8	9.79	**0.2665**	**0.1886**	6.1024	**1.1321**	0.2382	**0.1736**	**7.3210**	1.1641
	0.7	9.59	**0.2595**	**0.1873**	6.0083	**1.1273**	0.2395	**0.1755**	**7.3266**	1.1643
	0.6	9.96	**0.2491**	**0.1828**	6.1818	**1.1328**	0.2363	**0.1742**	**7.3644**	1.1641
	0.5	10.69	**0.2402**	**0.1775**	6.4258	**1.1416**	0.2400	**0.1724**	**7.4454**	1.1661
	0.4	10.67	0.2218	0.1693	6.6037	1.1468	**0.2423**	**0.1726**	**7.4642**	**1.1680**
	0.3	11.02	0.1994	0.1560	6.7211	1.1486	**0.2415**	**0.1719**	**7.4225**	**1.1650**
	0.2	10.76	0.1795	0.1441	6.6990	1.1454	**0.2416**	**0.1727**	**7.4960**	**1.1663**
	0.1	11.74	0.1570	0.1283	6.9472	1.1510	**0.2415**	**0.1701**	**7.5599**	**1.1682**
	0	11.14	0.1394	0.1171	6.8922	1.1448	**0.2403**	**0.1707**	**7.4629**	**1.1666**

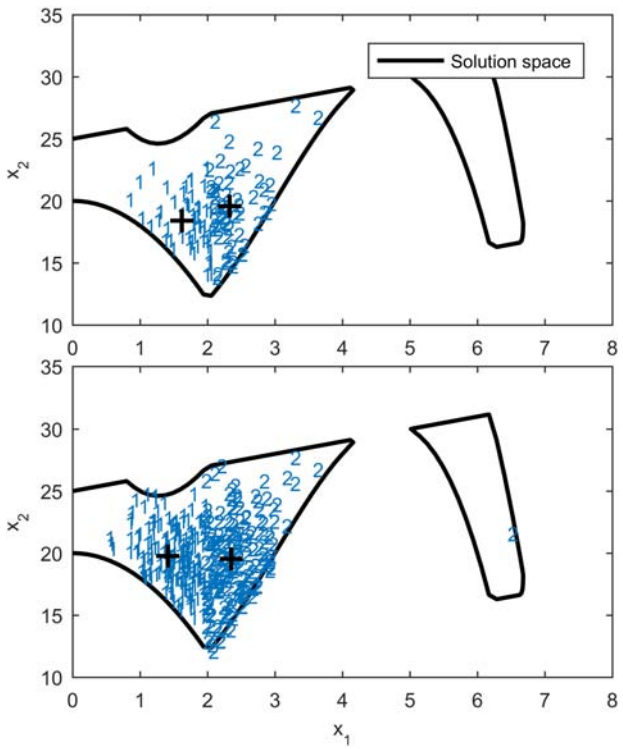

Fig. 2 A good example of GWOMTE for P1B (the first and second iterations)

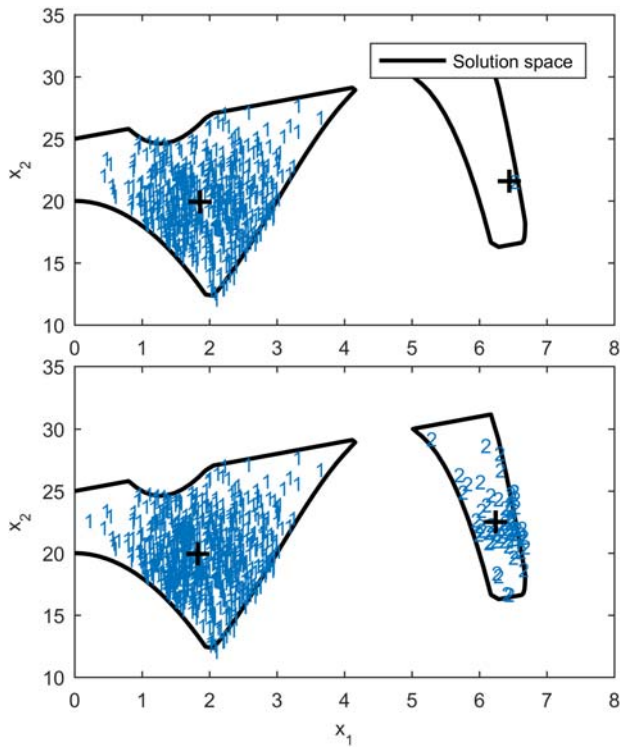

Fig. 3 A good example of GWOMTE for P1B (the third and fourth iterations)

iterations of the algorithm and Fig. 3 shows the third and fourth iterations. The algorithm did not find any point in the right set of the solution space in the first iteration. In the second iteration, one solution is found in the right set. However, both centers of clusters are in the left set. As more design points are sampled in the third iteration, one of the centers moved to the right set. In the fourth iteration, the algorithm started to find good design points in the right set.

3.2 Problem 2: Automotive Vehicle Design

3.2.1 Problem Description: Bicycle Model. In the bicycle model, lateral and yaw dynamics formulate vehicle motions. Lateral and yaw dynamics are shown in Eq. (14) and Eq. (15), respectively. Also, the vehicle geometry is shown in Fig. 4.

$$ma_y = F_{yf} + F_{yr} \tag{14}$$

$$I_{zz}\dot{r} = \sum M_z \tag{15}$$

In Eq. (14), m represents the mass of a vehicle, and F_{yf} and F_{yr} are the tire forces at the front and rear axles. In Eq. (15), I_{zz} is the yaw moment of inertia, and M_z is the yaw moment due to tire forces. Both equations can be formulated with the slip angle and cornering stiffness. The front and rear cornering stiffness are denoted as $C_{\alpha fs}$ and $C_{\alpha rs}$, respectively. The front and rear slip angle are defined as follows:

$$\alpha_f = \delta_t - \frac{V + ar}{U} \tag{16}$$

$$\alpha_r = \frac{-V + br}{U} \tag{17}$$

In Eqs. (16) and (17), U is the longitudinal velocity, V is the lateral velocity, and r represents the yaw rate. By using the cornering stiffness and slip angle, lateral and yaw dynamics are formulated as follows:

$$F_{yf} + F_{yr} = C_{\alpha_f}\alpha_f + C_{\alpha_r}\alpha_r \tag{18}$$

$$\sum M_z = aF_f - bF_r = aC_{\alpha_f}\alpha_f + bC_{\alpha_r}\alpha_r \tag{19}$$

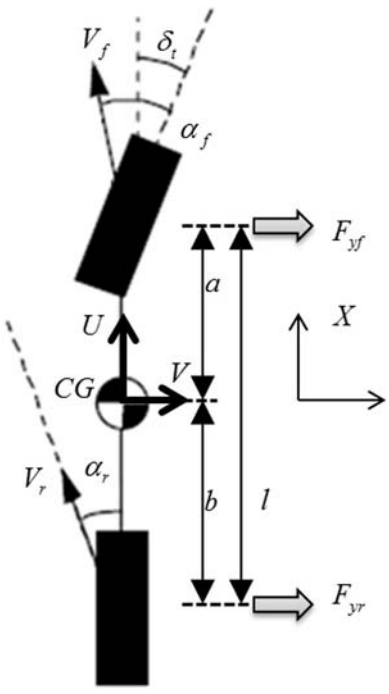

Fig. 4 A bicycle model

Here, $C_{\alpha_f} = 2C_{\alpha_f s}(180/\pi)$ and $C_{\alpha_r} = 2C_{\alpha_f r}(180/\pi)$. The final forms of the dynamic equations are obtained by setting the first derivatives of the state variables as 0. In the model, $U = 80$ and $g = 9.82$. The formulations are following.

$$\dot{V} = -\frac{(C_{\alpha_f} + C_{\alpha_r})}{m}\frac{V}{U} - \frac{(C_{\alpha_f}a + C_{\alpha_r}b + mU^2)}{m}\frac{r}{U} + \frac{C_{\alpha_f}}{m}\delta_t \tag{20}$$

$$\dot{r} = -\frac{(aC_{\alpha_f} - bC_{\alpha_r})}{I_{zz}}\frac{V}{U} - \frac{(a^2C_{\alpha_f} + b^2C_{\alpha_r})}{I_{zz}}\frac{r}{U} + \frac{aC_{\alpha_f}}{I_{zz}}\delta_t \tag{21}$$

3.2.2 Numerical Test. In this case study, seven design variables and five performance functions are used. Table 2 summarizes the design variables and performance functions. The solution space of the problem is following.

$$\begin{aligned}\mathcal{F}^2 = \{\mathbf{x} \mid\ & 50.3981 \le R_1(\mathbf{x}) \le 59.2831,\\ & 19.5470 \le R_2(\mathbf{x}) \le 22.9930,\\ & -4.4704 \le R_3(\mathbf{x}) \le -3.1263,\\ & -0.1598 \le R_4(\mathbf{x}) \le 0.2881,\\ & 1.1144 \le R_5(\mathbf{x}) \le 1.2277,\\ & 2250 \le x_1 \le 2750,\\ & 1350 \le x_2 \le 1650,\\ & 49.5 \le x_3 \le 60.5,\\ & 2250 \le x_4 \le 2750,\\ & 1080 \le x_5 \le 1320,\\ & 900 \le x_6 \le 1100,\\ & 14.4 \le x_7 \le 17.6\}\end{aligned} \tag{22}$$

where

$$R_1(\mathbf{x}) = \frac{U}{(a+b) - \dfrac{x_2(aC_{\alpha_f} - bC_{\alpha_r})U^2}{(a+b)C_{\alpha_f}C_{\alpha_r}}}\frac{100}{x_7}$$

$$R_2(\mathbf{x}) = \frac{U^2}{(a+b) - \dfrac{x_2(aC_{\alpha_f} - bC_{\alpha_r})U^2}{(a+b)C_{\alpha_f}C_{\alpha_r}}}\frac{100}{x_7}\frac{\pi}{180}$$

$$R_3(\mathbf{x}) = \frac{b(a+b)C_{\alpha_f}C_{\alpha_r} - x_2U^2aC_{\alpha_f}}{x_2x_4U^2\omega_n^2}\frac{100}{x_7}$$

$$R_4(\mathbf{x}) = \left(\frac{W_f}{C_{\alpha_f}} - \frac{W_r}{C_{\alpha_r}}\right)\frac{180}{\pi}$$

$$R_5(\mathbf{x}) = \frac{1}{2\pi}\omega_n$$

We set the parameters as $n = 20$, $t_{\max} = 40$, $p_{\max} = 500$, $c = 5 \times 10^{-3}$, and $k_{\max} = 10$. For the shrink tolerance s, 11 different values are tested (0, 0.1, 0.2, … , 1). For each shrink tolerance, we tested 20 different random seeds and the results shown in Table 3 are the averages of the outputs.

The results are similar to the results of the two-dimensional problem. r_g and r_u decreased as s got close to 0. However, r_u of GWODE1 was worse than r_u of brute-force search, which was 0.0698, while r_u of GWOMTE was better than r_u of brute-force search. Also, E_0 of GWOMTE was better than that of GWODE1. The spread metric, E_0, is not monotonically increasing as the number of points increases, but the maximum value that E_0 can get increases. As the number of good points for GWOMTE almost doubled that of GWODE1, E_0 was much higher for GWOMTE. The authors thought GWOMTE deserved the advantage because the algorithm got more good design points than GWODE1 from the same number of evaluations.

4 Discussion

The goal of the proposed methods is finding good and distinguishable design points as many as possible in a limited number

Table 2 Design variables and performance functions

	Symbol	Description	Unit
Design variables	x_1	Wheelbase	mm
	x_2	Total vehicle mass	kg
	x_3	Weight distribution ratio	%
	x_4	Yaw moment of inertia	kg m^2
	x_5	Front cornering stiffness	N/deg
	x_6	Rear cornering stiffness	N/deg
	x_7	Steering gear ratio	—
Performance functions	R_1	Steady-state yaw rate gain	1/s
	R_2	Steady-state lateral acceleration gain	(m/s^2)/deg
	R_3	Steady-state side slip	—
	R_4	Understeer gradient	deg/g
	R_5	Yaw rate natural frequency	Hz
Intermediate variables	L	$x_1/1000$	m
	a	$L(1-x_3/100)$	m
	b	$L-a$	m
	W_f	x_2gb/L	N
	W_r	x_2g-W_f	N
	$C_{\alpha f}$	$2x_5(180/\pi)$	N/rad
	$C_{\alpha r}$	$2x_6(180/\pi)$	N/rad
	ω_n	$\sqrt{\frac{1}{x_2x_4U^2}((a+b)^2C_{\alpha_f}C_{\alpha_r}+x_2U^2(-aC_{\alpha_f}+bC_{\alpha_r}))}$	rad/s
Parameters	U	Longitudinal velocity	kph
	g	Acceleration of gravity	m/s^2

Table 3 Result of the bicycle model

		GWOMTE				GWODE1			
s	p	r_g	r_u	E_0	$E_{0,\text{std}}$	r_g	r_u	E_0	$E_{0,\text{std}}$
1	102.50	**0.1049**	**0.1049**	**2146.1**	2.2472	0.0622	0.0613	1243.7	**3.0293**
0.9	102.65	**0.1051**	**0.1051**	**2153.2**	2.2452	0.0623	0.0613	1245.7	**3.0306**
0.8	100.15	**0.1049**	**0.1049**	**2101.5**	2.2451	0.0623	0.0614	1218.2	**3.0272**
0.7	92.95	**0.1049**	**0.1049**	**1951.3**	2.2448	0.0626	0.0617	1133.3	**3.0302**
0.6	98.55	**0.1040**	**0.1040**	**2051.0**	2.2424	0.0627	0.0618	1203.3	**3.0310**
0.5	84.70	**0.1036**	**0.1036**	**1755.6**	2.2451	0.0625	0.0616	1031.0	**3.0306**
0.4	96.50	**0.1036**	**0.1036**	**1997.2**	2.2521	0.0628	0.0619	1182.2	**3.0309**
0.3	99.70	**0.1016**	**0.1016**	**2022.1**	2.2570	0.0629	0.0620	1215.9	**3.0308**
0.2	95.30	**0.0949**	**0.0949**	**1806.4**	2.2966	0.0625	0.0616	1159.8	**3.0343**
0.1	107.15	**0.0840**	**0.0840**	**1805.5**	2.3568	0.0628	0.0619	1304.2	**3.0392**
0	113.20	**0.0770**	**0.0770**	**1744.6**	2.4060	0.0626	0.0617	1379.5	**3.0339**

of evaluations. The methods return a set of good designs, which is included in the solution space and a set of bad designs where the elements violate the target range. In the numerical experiments, the results of r_g and r_u were good for the size of the solution space. The high r_g values are meaningful for computationally expensive problems. Because of the complexity, the sampling or the design exploration process should be efficient, and the high r_g means the high probability of sampling a good design point. In terms of the spread metric, E_0, the results are different from r_u in the two-dimensional problems. It is because, for two different design points, E_0 increases if the distance between two points increases while r_u does not increase once they are not very close enough. r_u is a good metric for measuring the quality of distinguishable design points, but it is subjective because the user should define the value of the "close-enough distance."

In the experiment, GWOMTE performed well for the tested problems. Especially, for the two-dimensional problem, it performed much better than brute-force search. For the bicycle model, it still outperformed brute-force search, but the margin was smaller than the low-dimensional problem. Since the original GWO works best for benchmark problems and low-dimensional case-studies [29], further improvements and experiments should be done for high-dimensional problems. Also, we can use a different type of derivative-free method other than GWO, such as PSO or GA. GWO is used for the paper because we know how many evaluations are going to be done by the algorithm because of the stopping criterion of GWO. However, any other derivative-free method can be applied to this framework which is the concept of design exploration and setting the target points by k-means clustering. As the modified updating schemes are introduced specifically for GWO, the updating schemes for the method should be changed accordingly.

In the clustering stage of the proposed method, k-means and the silhouette method are used as they are the widely used clustering method, easy to implement, and efficient in time. However, k-means converges to a local minimum only (it converges to the global minimum in certain conditions [30]). In our method, we cannot guarantee or provide proof of convergence of the clustering stage. Instead, we added $p_{\max}$, the maximum number of explored target points, to use it for an additional stopping criterion. As one of the future works, we can fine-tune the proposed method by applying other clustering methods instead of k-means clustering. We can use the x-means clustering method [31], which is an extension of k-means clustering, where an algorithm automatically finds the number of clusters. By using x-means clustering, we do not have to use the silhouette method to determine the best number of

clusters. However, we still need to figure out if the clustering stage guarantees the optimality. Furthermore, we can use the time-series clustering method [32]. It is used when a series of datasets is given.

The outputs of proposed methods are two finite sets rather than infinite sets or feasible regions. If designers want to have an approximated solution space, the outputs, $\mathcal{G}$ and $\mathcal{B}$, can be used. By utilizing machine learning and data mining such as support vector machine [33], designers can perform a binary classification and build a prediction model that discriminate between good and bad designs. Since proposed methods sample the points near boundaries of the solution space, it will be effective to apply binary classification techniques.

Also, estimating the solution space or obtaining the finite set of good design points can be useful rather than having the best design by an optimization problem. For example, solution spaces from different target ranges can be combined to build a better design. For product family design, solution spaces from the same performance functions and different target ranges are sets of candidate solutions [19]. Designers can maximize the number of shared components by choosing design points that are in intersections of solution spaces.

5 Conclusion

In this paper, we introduced methods that can be used in the early stage of design processes. As target points explore the target range for performance functions, the objective function of the optimization is updated. The optimization problems are used as a means of sampling design points in the solution space. We modified the GWO for the sampling process, and it is used as a design exploration process. Target points are determined by clustering so that the centers of the clusters represent disconnected sets of the solution space. With the proposed methods, designers can obtain finite sets of design points, and those can be used to estimate solution space or design multiple products that satisfy different target ranges. Therefore, the proposed algorithms enable designers to explore design space and help them understand the solution space.

Acknowledgment

This material is based on work supported by Hyundai Motor Company. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of Hyundai Motor Company.

Nomenclature

c = target convergence tolerance
n = number of search agents
s = shrink tolerance
$\mathcal{B}$ = set of bad designs
$\mathcal{G}$ = set of good designs
M = number of performance functions
N = number of design variables
$k_{\max}$ = maximum number of clusters
$p_{\max}$ = maximum number of explored target points
r_g = rate of good designs
r_u = rate of unique designs
$t_{\max}$ = maximum number of GWO iterations
E_0 = spread
$E_{0,\text{std}}$ = spread of standardized data
$\mathbf{R}_j(\mathbf{x})$ = performance function ($j = 1, \ldots, M$)
$\mathbf{X}_i$ = position vector of ith agent
$\mathbf{x}^{\mathbf{L}}, \mathbf{x}^{\mathbf{U}} \in \mathbb{R}^N$ = boundary of design variables
$\mathbf{T}^{\mathbf{L}}, \mathbf{T}^{\mathbf{U}} \in \mathbb{R}^M$ = target range of responses form $\mathbf{R}(\mathbf{x})$
$\mathbf{T}^{\mathbf{LIST}}(l)$ = list of target points at lth iteration
$\mathbf{x} \in \mathbb{R}^N$ = design variables

References

[1] Chowdhury, S., Messac, A., and Khire, R. A., 2011, "Comprehensive Product Platform Planning (CP3) Framework," ASME J. Mech. Des., **133**(10), p. 101004.
[2] Simpson, T. W., and Dsouza, B. S., 2004, "Assessing Variable Levels of Platform Commonality Within a Product Family Using a Multiobjective Genetic Algorithm," Concurr. Eng., **12**(2), pp. 119–129.
[3] Braun, R. D., 1996, "Collaborative Optimization: An Architecture for Large-Scale Distributed Design," Ph.D. thesis, Stanford University, Stanford, CA.
[4] Sobieszczanski-Sobieski, J., Agte, J. S., and Sandusky, R. R., 2000, "Bilevel Integrated System Synthesis," AIAA J., **38**(1), pp. 164–172.
[5] Kim, H. M., Michelena, N. F., Papalambros, P. Y., and Jiang, T., 2003, "Target Cascading in Optimal System Design," ASME J. Mech. Des., **125**(3), pp. 474–480.
[6] Singer, D. J., Doerry, N., and Buckley, M. E., 2009, "What Is Set-Based Design?," Naval Eng. J., **121**(4), pp. 31–43.
[7] Sobek, D. K., Ward, A. C., and Liker, J. K., 1999, "Toyota's Principles of Set-Based Concurrent Engineering," Sloan Manage. Rev., **40**(2), p. 67.
[8] Bernstein, J. I., 1998, "Design Methods in the Aerospace Industry: Looking for Evidence of Set-Based Practices," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.
[9] Hakanen, J., and Knowles, J. D., 2017, "On Using Decision Maker Preferences With Parego," International Conference on Evolutionary Multi-Criterion Optimization, Münster, Germany, Mar. 19–22.
[10] Hannapel, S., and Vlahopoulos, N., 2014, "Implementation of Set-Based Design in Multidisciplinary Design Optimization," Struct. Multidiscipl. Optim., **50**(1), pp. 101–112.
[11] Al-Ashaab, A., Golob, M., Attia, U. M., Khan, M., Parsons, J., Andino, A., Perez, A., Guzman, P., Onecha, A., Kesavamoorthy, S., and Martinez, G., 2013, "The Transformation of Product Development Process Into Lean Environment Using Set-Based Concurrent Engineering: A Case Study From an Aerospace Industry," Concurr. Eng., **21**(4), pp. 268–285.
[12] Lee, S., Bae, J., and Cho, Y. S., 2012, "Efficiency Analysis of Set-Based Design With Structural Building Information Modeling (s-bim) on High-Rise Building Structures," Autom. Construct., **23**, pp. 20–32.
[13] Tsang, E., 2014, *Foundations of Constraint Satisfaction: The Classic Text*, Academic Press, New York (BoD—Books on Demand).
[14] Lottaz, C., Smith, I. F., Robert-Nicoud, Y., and Faltings, B., 2000, "Constraint-Based Support for Negotiation in Collaborative Design," Artif. Intell. Eng., **14**(3), pp. 261–280.
[15] Yannou, B., Moreno, F., Thevenot, H. J., and Simpson, T. W., 2005, "Faster Generation of Feasible Design Points," Proceedings of International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE2005), Hyatt Regency, Long Beach, CA, Sept. 24–28.
[16] Pedamallu, C. S., Özdamar, L., Csendes, T., and Vinkó, T., 2008, "Efficient Interval Partitioning for Constrained Global Optimization," J. Global Optim., **42**(3), pp. 369–384.
[17] Mattson, C. A., and Messac, A., 2003, "Concept Selection Using S-Pareto Frontiers," AIAA J., **41**(6), pp. 1190–1198.
[18] De Weck, O. L., and Jones, M. B., 2006, "Isoperformance: Analysis and Design of Complex Systems With Desired Outcomes," Syst. Eng., **9**(1), pp. 45–61.
[19] Eichstetter, M., Müller, S., and Zimmermann, M., 2015, "Product Family Design With Solution Spaces," ASME J. Mech. Des., **137**(12), pp. 121401.
[20] Davis, L., 1991, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.
[21] Koza, J. R., 1992, *Genetic Programming II, Automatic Discovery of Reusable Subprograms*, MIT Press, Cambridge, MA.
[22] Hansen, N., and Kern, S., 2004, "Evaluating the CMA Evolution Strategy on Multimodal Test Functions," International Conference on Parallel Problem Solving From Nature, Birmingham, UK, Sept. 18–22.
[23] Dorigo, M., and Gambardella, L. M., 1997, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Trans. Evol. Comput., **1**(1), pp. 53–66.
[24] Eberhart, R., and Kennedy, J., 1995, "A New Optimizer Using Particle Swarm Theory," Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS'95), Nagoya Municipal Industrial Research Institute, Oct. 4–6.
[25] Mirjalili, S., Mirjalili, S. M., and Lewis, A., 2014, "Grey Wolf Optimizer," Adv. Eng. Softw., **69**, pp. 46–61.
[26] Hartigan, J. A., and Wong, M. A., 1979, "Algorithm as 136: A K-Means Clustering Algorithm," J. R. Stat. Soc. Ser. C, **28**(1), pp. 100–108.
[27] Rousseeuw, P. J., 1987, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," J. Comput. Appl. Math., **20**, pp. 53–65.
[28] Willerton, S., 2015, "Spread: A Measure of the Size of Metric Spaces," Int. J. Comput. Geom. Appl., **25**(3), pp. 207–225.
[29] Faris, H., Aljarah, I., Al-Betar, M. A., and Mirjalili, S., 2017, "Grey Wolf Optimizer: A Review of Recent Variants and Applications," Neural Comput. Appl., **30**(2), pp. 413–435.
[30] Meilă, M., 2006, "The Uniqueness of a Good Optimum for K-Means," Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, USA, June 25– 29.
[31] Pelleg, D., and Moore, A. W., 2000, "X-Means: Extending K-Means With Efficient Estimation of the Number of Clusters," Proceedings of the 17th International Conference on Machine Learning, San Francisco, CA.
[32] Aghabozorgi, S., Shirkhorshidi, A. S., and Wah, T. Y., 2015, "Time-Series Clustering—A Decade Review," Inf. Syst., **53**, pp. 16–38.
[33] Cortes, C., and Vapnik, V., 1995, "Support-Vector Networks," Mach. Learn., **20**(3), pp. 273–297.