Neurocomputing 248 (2017) 41–56

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A method to learn high-performing and novel product layouts and its application to vehicle design

Victor Parque[a,b,*], Tomoyuki Miyashita[a]

[a] Department of Modern Mechanical Engineering, Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
[b] Department of Mechatronics and Robotics, Egypt-Japan University of Science and Technology, Qesm Borg Al Arab, Alexandria, Egypt

A R T I C L E   I N F O

Article history:
Received 30 June 2016
Revised 29 November 2016
Accepted 3 December 2016
Available online 8 March 2017

Keywords:
Design
Vehicle
Optimization
Genetic programming
Gradient-free optimization

A B S T R A C T

In this paper we aim at tackling the problem of searching for novel and high-performing product designs. Generally speaking, the conventional schemes usually optimize a (multi) objective function on a dynamic model/simulation, then perform a number of representative real-world experiments to validate and test the accuracy of the some product performance metric. However, in a number of scenarios involving complex product configuration, e.g. optimum vehicle design and large-scale spacecraft layout design, the conventional schemes using simulations and experiments are restrictive, inaccurate and expensive.

In this paper, in order to guide/complement the conventional schemes, we propose a new approach to search for novel and high-performing product designs by optimizing not only a proposed novelty metric, but also a performance function which is learned from historical data. Rigorous computational experiments using more than twenty thousand vehicle models over the last thirty years and a relevant set of well-known gradient-free optimization algorithms shows the feasibility and usefulness to obtain novel and high performing vehicle layouts under tight and relaxed search scenarios.

The promising results of the proposed method opens new possibilities to build unique and high-performing systems in a wider set of design engineering problems.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

A central question in the field of Design Engineering is how to build optimal product design layouts. Tackling this question implies not only understanding, but also developing the effective search heuristics and knowledge representations that lead to relevant designing. Thus, developing such heuristics implies the realization of concrete procedures that map key features from the space of needs and functions to product configuration. How to build such heuristics?

Often, developing effective algorithms for product design have focused on building articulated rules that link preferential and functional requirements to concrete product parameters [2–4,10,12,13]. Also, a natural answer to the above question comes from practical fields: artisans use (1) *metaphoring*, which is the process of mimicking the functions and features of pre-existing referential entities, and (2) *tooling*, which is the process of building systems by using modularity principles to enable scalability and hierarchical entities. Both *metaphoring* and *tooling* principles rely on search heuristics to find the relevant set of articulated rules that map the space of needs and functions to the key systems having specific parametric representations. Here, search heuristics are either performed artificially by using optimization algorithms or performed naturally by the artisan him(her)self.

However, considering matters of practical use and effective sampling of the design search space, it is cumbersome to perform the above effectively and efficiently. The main reason is that in certain design applications either (1) real-world experiments are expensive and dangerous, or (2) simulations are inaccurate to model and represent the real-world invariants reasonably well. Examples of this scenario include the design of the most complex existing systems in the world including vehicles, airplanes, large-space spacecrafts, and robots for critical environments.

In order to tackle the above limitations in the current approaches for product designs, in this paper we introduce a simple and complementary approach for tasks involving the optimal configuration of parametric product design layouts. Basically, we tackle the problem of how to learn the optimal mappings from functional and user requirements to concrete product definitions by using historical performance data and a newly applied novelty metric; so

* Corresponding author.
E-mail addresses: parque@aoni.waseda.jp, victor.parque@ejust.edu.eg (V. Parque), tomo.miyashita@waseda.jp (T. Miyashita).

http://dx.doi.org/10.1016/j.neucom.2016.12.082
0925-2312/© 2017 Elsevier B.V. All rights reserved.

that the whole task of designing a product entity optimizes a set of user-defined and pre-stated goals in performance and uniqueness.

1.1. Related works

Generally speaking, inspired by the work of artisans whom usually build a relevant rule set by interaction, iteration, analysis and synthesis, it is natural to develop algorithms for product design that enable some form of *metaphoring* [2] and *tooling* [1]. These algorithms replicate, in a digital and conceptual sense, not only the process of mimicking the functions and features of pre-existing referents, but also the process of building modularity to scale toward sophisticated hierarchical, ontological entities.

However, the above artisan-inspired approaches to product design layout are resource-consuming: either time-consuming iterations in the realization of the product design are needed, or costly real-world experimentations become essential. Representative examples of research in this direction include achievements in the last decade to design new chemical reactors for failing batteries [6], and novel inhibitors for a new type of influenza [7].

Then, improving the efficiency in resource usage during the task of designing product layouts becomes essential. In that sense, one would like to use and learn the most optimal mappings from functional and user requirements to concrete product design definitions efficiently. Then, the optimal mapping aids the holistic task of designing to optimize a set of pre-stated goals. An example of research in this direction include the work of Wang et al. whom use evolutionary computation for product design and manufacturing [5]; and the work of Akai et al. whom use Simulated Annealing for the dynamic deployment of a product family under commonalization-oriented modularity and a discrete choice-based market model [28].

It is straightforward to use optimization heuristics to realize the optimal mappings between functional and user requirements to concrete design definitions. However, as described before, the direct use of such techniques requires using either real-world experimental tests or simulation experiments to evaluate a set of predefined metrics of performance. And, in the field of Design Engineering, approaches involving experiments followed by simulations have been used widely in a plethora of applications. Some examples include the use of Simulated Annealing to find optimal composition of parameters in chemical compounds through reactions modeled in directed graphs [6]. Also, Dynamic Programming and Gradient-based algorithms were used to optimize inhibitors against H1N1 influenza [7]. Evolutionary Algorithms were used to optimize not only the product design but also the manufacturing process [8]. Genetic Algorithms were used to learn the relationships between firm and market factors to enable rapid product design and development [9]. Furthermore, Fuzzy Logic helped modeling rules to facilitate the product screening to decide new product configurations [10,11]. Data Envelopment Analysis was used to evaluate multiple factors during the evolutionary-based generation of ideas [3]. Clustering methods were used to build modular mechatronic systems [12]. Reinforcement Learning was used to map the function requirements to means realizations in a vehicle design problem [13].

1.2. Our results

In order to tackle the above problems, we propose a simple approach to search for optimal product designs under scenarios having restrictive simulations and experimentations. Basically, we use historical data in order to approximate a representative surrogate function that not only models, but also explains the performance of observed past design variables reasonably. Then, by using the learned surrogate performance function, along with a proposed novelty function, we optimize the product design layouts. The main goal and advantage of our approach is not only to map the representative approximations of the off-line learning (which is based on historical data) to the global refinements in parametric optimization (both in continuous space), so that the overall search heuristic focuses on sampling over the most meaningful (novel and high-performing) areas of the product layout search space. Our proposed approach aims at contributing towards the holistic design of product systems.

In line of the above, neither real-world experiments nor simulation tests consider novelty metrics in a meaningful way, such as the work of Grignon and Fadel [14]. Basically, novelty is an important factor in design because it enables the meaningful, the complete and the uniform sampling of the design search space; where novelty is defined as the maximum dissimilarity of the sampled product design compared to the convex hull of observed (historical) clusters.

Computational experiments using historical data of more than twenty thousand vehicles and rigorous computational experiments using Genetic Programming and eight representative gradient-free optimization algorithms, described subsequently, show that our proposed scheme works well: (1) a competitive performance function is generated within 10 h, (2) the breadth of Genetic Programming models is a key parameter to learn vehicle performance functions, and (3) it is possible to obtain unique and high-performing vehicle layouts when the search space is both constrained and unconstrained by history.

The main contributions in this paper are summarized as follows:

- We provide a simple and effective approach for optimizing the product design layout using a performance function, learned from historical data, and newly proposed novelty metric.
- Time for generating vehicle design layouts is feasible experimentally, and the possibility for generating unique and high-performing vehicle layouts are confirmed by rigorous and extensive computational experiments involving more than twenty thousand vehicle layouts and representative gradient-free algorithms in the literature.

1.3. Extension

The basic ideas and building blocks developed in this paper were firstly introduced at ICONIP 15' [26]. In this paper, we further develop our proposed approach in the following points:

- Inclusion of constraints to consider technical constraints in sheet thickness, vehicle lengths, movable free-space inside the vehicle and feasible bounds on vehicle mileage performance.
- Rigorous sensitivity analysis on a number of relevant Genetic Programming parameters when learning vehicle performance functions, that is a rigorous study on the convergence time and the quality of the learned functions with regard to the number of multi-trees, population size, and tree depth.
- Rigorous study of the vehicle layout problem, under tight and relaxed box-bounding constraints, by using significant gradient-free optimization algorithms that consider multimodality, parameter adaptation, search memory, selection pressure, probability distribution, neighborhood concepts, search space partitioning and adaptive meshing.

The rest of this paper is organized as follows. Section 2 describes the problem. Section 3 shows a case study using real world data for a vehicle design layout problem and Section 4 concludes the paper.

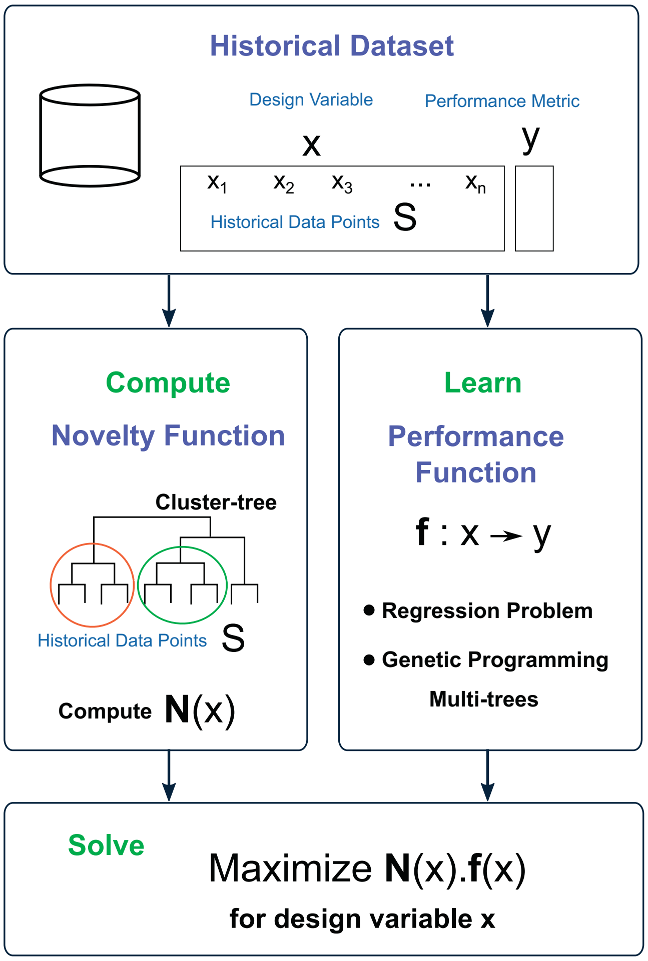


Fig. 1. Summary of the proposed approach.

2. The proposed method

2.1. Main problem

In this paper, we formulate the problem of product design layout as follows:

$$\text{Maximize } N(x).f(x) \quad x \in \mathbf{X} \tag{1}$$

where x is the product design variable, $\mathbf{X}$ is the box-bounding restriction on the variables, $N(x)$ is the novelty function of the design variable x, and $f(x)$ is the performance of the design variable $x \in R^n$. Then, we aim at tackling the above problem, for which the coming sub-sections describe both how to compute the novelty factor $N(x)$ and the performance factor $f(x)$.

2.2. Summary

Fig. 1 introduces the main steps to tackle Eq. (1). Basically, our proposed approach consists of the following main steps:

- *Input*. Given a historical dataset of design variables x and performance function y of a product, where a set of historical data points of design variables is denotes by S.
- *Step 1*. Compute the novelty function $N(x)$ as a function of the bounds of the tree clusters resulting from the historical data points S.
- *Step 2*. Learn the performance function f: $x \to y$, which is a regression problem solved by a nonlinear function learner (e.g. Genetic Programming with Multi-trees).
- *Step 3*. Solve the problem stated in Eq. (1) wherein x is the product design variable and given that both the novelty function $N(x)$ and performance function $f(x)$ are computed above.

2.3. Computing novelty

2.3.1. Convex hull approach

For $x_0 \in R^n$ being a design vector, in this subsection, we explain how to compute the novelty factor $N(x)$ w.r.t a referential set S with points.

Our approach to compute the novelty factor is the minimum distance to the convex hull of the given input data S. Concretely speaking, our scheme is as follows:

$$N(x_0) = M \quad in\ ||\mathbf{I}x - x_0||^2 \quad Ax \le b \quad A_{eq}x = b_{eq} \tag{2}$$

where $\mathbf{I}$ is the identity matrix, $||.||$ is the norm function, and A, b, A_{eq} and b_{eq} are the linear constraints defining the *convex hull* $Conv(S)$ of the points in the set S so that any point $x \in Conv(S)$ satisfies the following: $Ax \le b$ and $A_{eq}x = b_{eq}$. Clearly, the above is a constrained linear least squares problem and convergence to the optima is guaranteed. Solutions of the problem can be computed using any convex optimization approach.

Basically, the minimum distance to the convex hull is a simple heuristic to compute how novel x is compared to the set S. However, the reader may find Eq. (2) unintuitive: Doesn't novelty imply a maximum (not minimum) distance from S?

Note that Eq. (2) is the minimum distance of a point x_0 to a polygon representing the convex hull Conv(S) of the set of points S. Taking the maximum distance renders an incorrect metric for novelty, since the maximum distance overestimates the dissimilarity of the point x_0 to set of points S. Rather, the minimum distance (lower bound) is a more accurate estimation of such dissimilarity. Maximizing the dissimilarity (or maximizing distance) of the sampled points to the set of points in S is achieved by the solution of the problem in Eq. (1).

2.3.2. Unique points of convex hull approach

Our proposal to compute the novelty metric is unique in the sense that it offers accuracy, efficiency and robustness when measuring dissimilarity degree.

On the other hand, there exists a variety of distance metrics in the literature which can be used as dissimilarity proxies, e.g. the Euclidean or the Manhattan approaches to the mean or centroid of a referential set S. One is able to take the distance of x_0 to each point in S and take the mean (or any statistic) to compute a representative novelty metric for x_0.

However, in design engineering, if our interest is to learn novel design configurations for practical use, the conventional distance metrics using a single referential point have problems in efficiency and accuracy:

- *Inefficiency*: the bounded time complexity is $O(|S|)$ per sampled point x_0, which is sufficiently large for problems involving very rich historical data S. For instance, consider a sampling algorithm where the dataset size $|S|$ is equal to the population size. In such setting, computing the Euclidean distance among all sampled points in the population is expected to be of quadratic order ($O(|S|^2)$) [27], which is confirmed by computational experiments [18].
- *Inaccuracy*: the distance of any sampled point x_0 which is sufficiently close to any point in the historical dataset S brings no information on how dissimilar is the sampled point from the existing clusters in the historical data, thus giving an incomplete metric for novelty. Consider the example in Fig 2. Here,

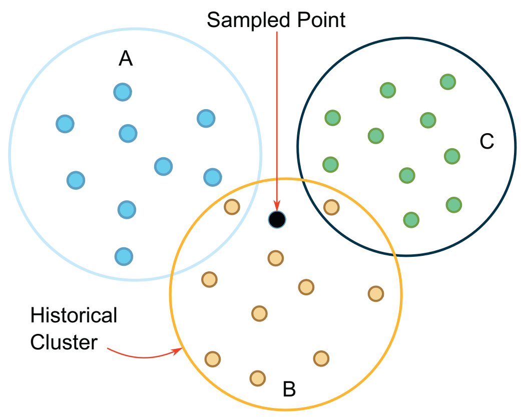


Fig. 2. Example of a sampled point which is sufficiently close to a point in the dataset.

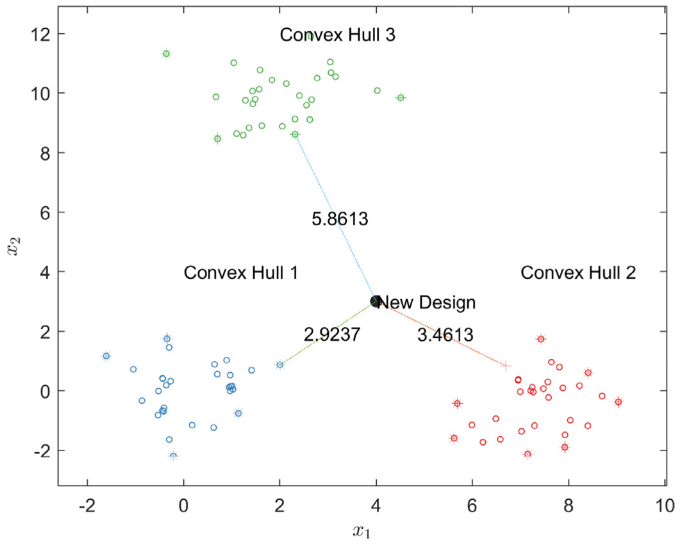


Fig. 3. Example of Measuring Novelty between a *New Design* point and three referential sets of points. The convex hull points are represented with '*', and any other point is represented with 'o'.

the sampled point is sufficiently close to some arbitrary point in the dataset. Then, the closest distance to historical points is to provide an innacurate notion of novelty, since the sampled datappoint belongs to the historical cluster *B*. Similar notions are to occur when the sampled points are in areas between clusters $A-B$, as well as $B-C$.

Thus, our approach to measure novelty by using the convex hull *Conv*(*S*) is advantageous to improve the accuracy and the robustness when measuring dissimilarity from a referential historical set *S*.

Fig. 3(a) shows an example of three sets of data and the minimum distances from a design variable x_0 to the convex hulls. In this figure, the design variable is two-dimensional $x \in R^2$, as well as the three point sets of referential arbitrary data (left, right and top). If one is interested in knowing how novel the *New Design* x_0 is, one needs to compute the minimum distance to the convex hulls. In Fig. 3, the minimum distance to the *Convex Hull 1* is 2.9237, the minimum distance to the *Convex Hull 2* is 3.4613, and the minimum distance to the *Convex Hull 3* is 5.8613. The reader may not that the points minimizing the distance to the Convex Hulls are not necessarily the vertices of the Convex Hull.

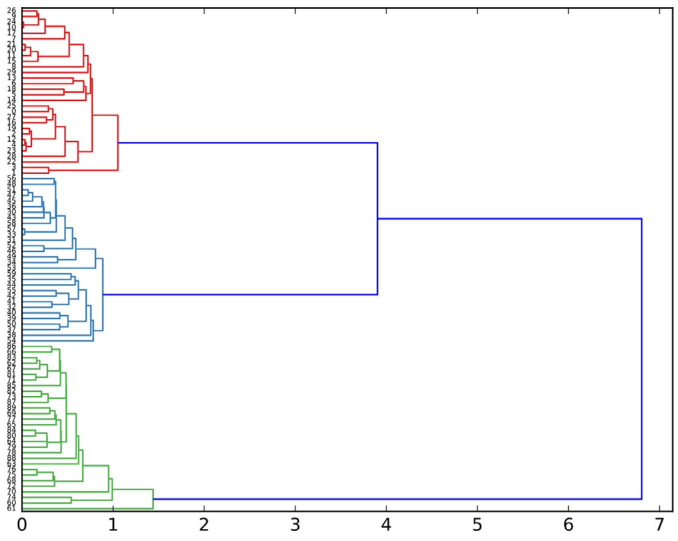


Fig. 4. Tree generated from the set of arbitrary points. Note colors preserve clustering topology with respect to the set of points shown in Fig. 3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

2.3.3. Convex hull and hierarchical clustering

How to deal with unstructured and arbitrary historical data points? A simple answer to this question is to use a clustering algorithm based on the K-means strategy. However, the K-means approach requires a-priori knowledge of the number of clusters, which is difficult to obtain in instances of incomplete and partially known history.

Instead of using the K-means approach, we use the hierarchical clustering algorithm with single (minimum) linkage. Here, the convex hull can be easily computed by using the subtrees generated from cutting the main hierarchical tree. Concretely speaking, we perform the following two steps:

- First, cluster the set *S*, using the single-linkage hierarchical clustering with single (minimum) linkage. The result is a tree T_o.
- Second, cut the tree T_o using the threshold *TH*. Here, the result is a set of trees *T*, each of which defines a subset of points of *S*.

Important to our scope, note that |*T*| is unknown a-priori, it is a consequence of choosing and appropriate threshold *TH*. Also, in design engineering applications, it is important to note that the threshold *TH* captures the tolerance for dissimilarity in the existing historical product clusters. Then, smaller values of the *TH* implies increasing the granularity of the observed clusters.

In order to exemplify our assertions, Fig. 4 shows an example of the tree T_o generated by the hierarchical clustering of the set of points displayed in Fig. 3. In this figure, *x*-axis we plot the minimum distance among clusters, and in *y*-axis we plot we plot the ordering of the points. Thus, if one decides to use some threshold *TH*= 2 (cutting line) to separate the clusters, it is straightforward to obtain three trees *T* (clusters) automatically. Then, the points on the leaves of the sub-trees *T* can be readily used to compute their respective convex hulls. Naturally, for the sake of the example, Fig. 4 shows sub-trees maintaining the colors of the observed clusters in Fig. 3.

2.3.4. Surrogate to convex hull

Computing the Convex Hull and the Hierarchical Clustering are time-consuming tasks. Indeed, considering a set of data inputs with *m* observations in R^n ($m=|S|$), computing the Convex Hulls using the *gift wrapping* algorithm takes time complexity $O(m^{n+1})$ [17]. Though, the *quick hull* algorithm provides reasonable approximations running with $O(m\log(m))$ time [16]. In the same line, the hierarchical clustering algorithm with single linkage runs with $O(m^2)$ time [15]. Thus, given that the time complexity of the above algorithms is polynomial in the number of observations m, we propose an alternative method to the novelty factor of a design variable x with respect to the set of observed points in the set of clusters T. Concretely speaking, our formulation is as follows:

$$N(x) = min\ V_i(x) \tag{3}$$

$$V_i(x) = \left[1 - \frac{1}{n}\sum_{j=1}^{n} v_{i,j}(x)\right]^{1/a} \tag{4}$$

$$v_{i,j}(x) = \begin{cases} 1 & \text{if } x \in [L_{i,j}, U_{i,j}] \\ 0 & otherwise \end{cases} \tag{5}$$

$$\begin{array}{c} L_{i,j} = min\ (T_{i,j}) \\ Ui, j = max\ (T_{i,j}) \\ i \in [1,k], j \in [1,n] \end{array} \tag{6}$$

where i and j are index suffixes for clusters and dimensions, respectively; $L_{i,\,j}$ is the lower bound in the j-th dimension of the i-th cluster T_i; $U_{i,\,j}$ is the upper bound in the j-th dimension of the i-th cluster T_i; $v_{i,\,j}(x)$ is an n-dimensional binary variable indicating whether x is inside or outside the boundaries of the i-th cluster T_i; $V_i(x)$ is the novelty degree of variable x with respect to the i-th cluster T_i; a is the novelty normalization constant; and n is the dimensionality of x.

In the above formulation, computing the lower and upper bounds $[L_{i,\,j}, U_{i,\,j}]$ for each cluster $i \in [1, k]$, and each dimension $j \in [1, n]$ is fast and prone to be improved by parallel computation. For $\gamma = \max|T_i|$, the time complexity is bounded by $\mathcal{O}(n\gamma)$ in case of using a single processor, and $\mathcal{O}(n \log \gamma)$ in case of using $\mathcal{O}(\gamma / \log \gamma)$ processors.[1] These complexities are considerably lower compared to the polynomial bounds of convex hull algorithms. Thus, the above formulations realize the practical efficiency while keeping the novelty metric within the same principles as stated in Eq. (2). That is, measuring the dissimilarity degree of the variable x with respect to the bounds of an existing referential point set; where more dissimilar elements per dimension in the variable x, make x to be more unique in comparison to each observed cluster. The main role of using the *min* function in Eq. (3) is due to our intention to minimize the worst expected case in regard to the computed distance to each cluster.

2.4. Computing performance

2.4.1. Motivation

As discussed in Section 1, real-world experimental tests and simulation experiments are the widely used approaches to measure the performance of a design variable x.

However, in a number of circumstances not only the real-world experiments are expensive and dangerous, but also the simulations are inaccurate to consider the real-world granularity and noise. Thus, we propose an alternative approach based on historical data to approximate a surrogate function that explains the variances of the real-world performances of already existing design variables.

2.4.2. Learning the performance function

Concretely speaking, we use the following formulation:

$$Find\ \ f : S \to Y \tag{7}$$

where S is the set of historical data (observed points) of the design variables, Y is the metric with historical performance associated with the set of observed points in S, and f is the function that approximates the mapping between the design variable $x \in S$ to its performance metric Y.

Basically, for simplicity and without loss of generality, the above statement can be formulated as the following regression problem:

$$\begin{array}{c} Minimize\ \sqrt{\dfrac{1}{|S|}\sum\limits_{x \in S}[f(x) - Y]^2} \\ f \in \mathbf{F} \end{array} \tag{8}$$

where $\mathbf{F}$ is the space of function encodings.

The current literature provides with a number of suitable heuristics being able to compute reasonable approximations of the function f, depending on the kind of space $\mathbf{F}$ being used. For example, if one considers the space $\mathbf{F}$ as kernel hyperplanes, we may use Support Vector Machines. If one considers the space $\mathbf{F}$ as directed graphs, we may use Neural Networks or Genetic Programming. If one considers the space $\mathbf{F}$ as density functions, we may use Probability based approaches.

2.4.3. Genetic network programming with multi-trees

Without having a-priori knowledge of the convexity of the function f, and without losing the expression ability of the learned functions, we use Genetic Programming due to its feature to offer understandability of the modeled/learned functions. Concretely speaking, to model the function f we used Genetic Programming with multi-trees as follows:

$$f(x) = w_0 + \sum_{i=1}^{nGP} w_i . t_i(x) \tag{9}$$

where w_0 is the bias term, t_i represents a tree in Genetic Programming, w_i is the weight of the i-th tree used in the linear combination, and nGP is the maximum number of trees set by the user.

The main reason behind using Genetic Programming with multi trees is to enable the ensemble-based approach to learning nonlinear functions, thus reducing the risk of over-fitting due to bloated local trees. The above mechanism is simple and well-known in the machine learning community. Investigating different complementary approaches to reduce the risk of over-fitting to the historical training data is left for future work.

3. Computational experiments

In this section we describe the computational experiments performed to evaluate the effectiveness and usefulness of our method.

3.1. Motivation

In line of the above, the optimum vehicle design is chosen to be included in the experiments section, since simulations and experiments are restrictive, inaccurate and expensive.

- Dynamic simulations and experiments are restrictive in the sense of not being able to generalize to a wide spectrum of vehicle models. A large number of dynamic models or large number of real-world experimental trials is needed to evaluate the performance function of diverse vehicle layouts, implying the expensive use of resources; e.g. time and capital.
- Dynamic simulation is inaccurate in estimating the performance of vehicle design layouts since it implies solving a large number of interrelated dynamic equations influenced by uncertainty and noise. Often, to alleviate this problem, dynamic simulations are aided by expensive real-world experiments, e.g. car-in-the-loop, to fine-tune and validate the models/assumptions.

[1] To ensure work-efficiency in parallel cores according to the Brent's Theorem.

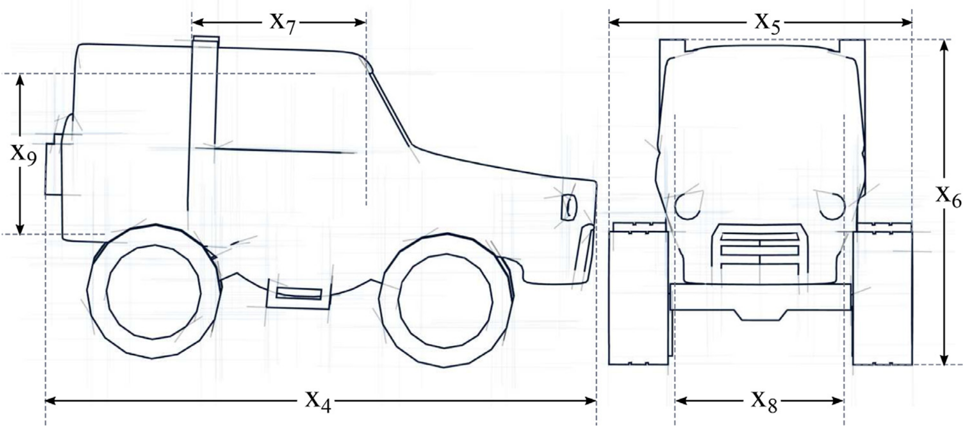


Fig. 5. Design variables in the vehicle design layout problem.

Table 1
Main constraints used in the vehicle layout problem.

Constraint	Description
$x_4 > x_7$, $x_5 > x_8$, $x_6 > x_9$	Exterior lengths are wider than interior lengths.
$x_7, x_8, x_9 > 1$	Minimum interior free-space.
$x_4 - x_7 \in (1, 2)$	Length needed for engine and powertrain.
$x_6 - x_9 \in (0.25, 0.75)$	Thickness of vehicle base.
$x_5 - x_8 \in (0.07, 0.15)$	Thickness of vehicle ceil and wall.
$f(x) \in (0, 200)$	Performance function of vehicle.

Table 2
List of design variables in the vehicle layout design problem.

Variable	Variable name	Units	Min	Max	Std
x_1	Torque	Nm	4	92.4	8.474
x_2	Maximum output	Ps	28	550	59.259
x_3	Engine displacement	cc	0.533	4.996	0.751
x_4	Total length	m	2.735	5.460	0.456
x_5	Full width	m	1.390	1.970	0.096
x_6	Height	m	1.040	2.285	0.184
x_7	Interior length	m	0.630	4.250	0.424
x_8	Interior width	m	1.150	1.730	0.089
x_9	Interior height	m	0.935	1.615	0.079
Y	Fuel efficiency	km/l	0	61	4.181

Table 3
Main parameters in the Genetic Programming algorithm.

Name (Symbol)	Values
Number of GP trees (nGP)	{10, 20, 40}
Population size (P)	{250, 500, 1000}
Maxdepth (D)	{5, 10}
Operators	+ − *, *pdiv, psqrt, plog, tanh, iflte, exp*
Generations	1000
Tournament selection size	15
Mutation robability	0.1
Crossover robability	0.85
Elite raction	0.05

On the other hand, our proposed approach requires of heavy use of historical observations of design variables to tackle the problem of optimum vehicle design; and can be used as a guide/complement to improve/optimize simulations and experiments.

Thus, due to the above considerations and in the context of complex product configuration, such as those in vehicle and large-scale spacecraft layout design, we chose the optimum vehicle design as the computational experiment platform where acquisition of historical data is widely available.

3.2. Vehicle design layout

To this extent, we consider the problem of finding high-performing and novel vehicle layouts, in line of Eq. (1). Historical data involves 23,599 observations of car models between 1982 and 2013 and considers the vehicle design variables and the vehicle performance metric as shown in Table 2. In order to give glimpse of the tackled problem, Fig. 5 exemplifies a number of design variables in a conceptual vehicle layout.

Furthermore, in order to ensure feasibility when querying the space of vehicle layouts, the constraints as displayed in Table 1 are used. These constraints represent a meaningful set of definitions in order to allow a reasonable degree of feasible vehicle structures. The constraint $f(x) \in (0, 200)$ has the specific role to avoid over-fitted issues of Genetic Programming when learning the performance function $f(x)$ when using historical data. Also, whenever any of the above constraint is not satisfied, the objective function is replaced by its natural logarithm.

Therefore, our aim is to search for the feasible vehicle design layouts x being both novel and high performing, in the sense that the novelty function $N(x)$ is computed by Eq. (3) and the performance function $f(x)$ is computed by the ratio of vehicle fuel efficiency.

3.3. Clustering and number trees

Furthermore, in order to compute the novelty metric, the tree generated by the Hierarchical Clustering algorithm with single linkage is shown in Fig. 7, and using the threshold $TH = 10$ the generated set T consists of 17 subtrees. The reason for using $TH = 10$ is due to our intention to have a reasonable number of clusters to compute the novelty metric. Since small (large) values of TH renders large (small) number of tree-clusters as shown in Fig. 6, choosing $TH = 5$ implies rendering 49 tree-clusters, choosing $TH = 20$ implies rendering 8 tree-clusters, and choosing $TH = 40$ implies rendering 4 tree-clusters. Thus, for a reasonable and close to real approximation (preferably in the range $|T| \in [10, 30]$) of the number of tree-clusters, we used $TH = 10$ to generate a set T consisting of 17 tree-clusters.

3.4. Learned performance function

In order to generate the performance function $f(x)$ by solving Eq. (8), we used Genetic Programming with the parameters shown by Table 3. Here, the number of GP trees (nGP), the Population size

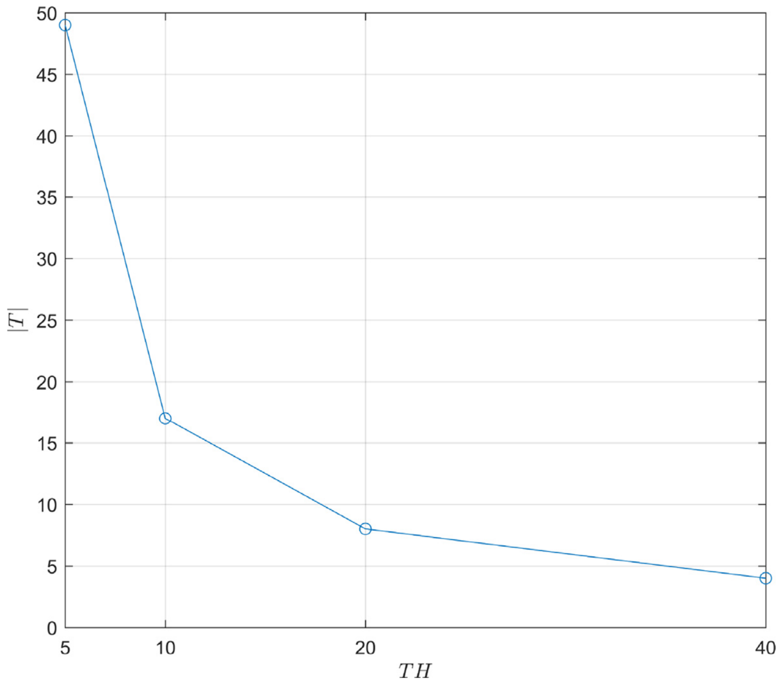


Fig. 6. Number of trees $|T|$ as a function of threshold TH.

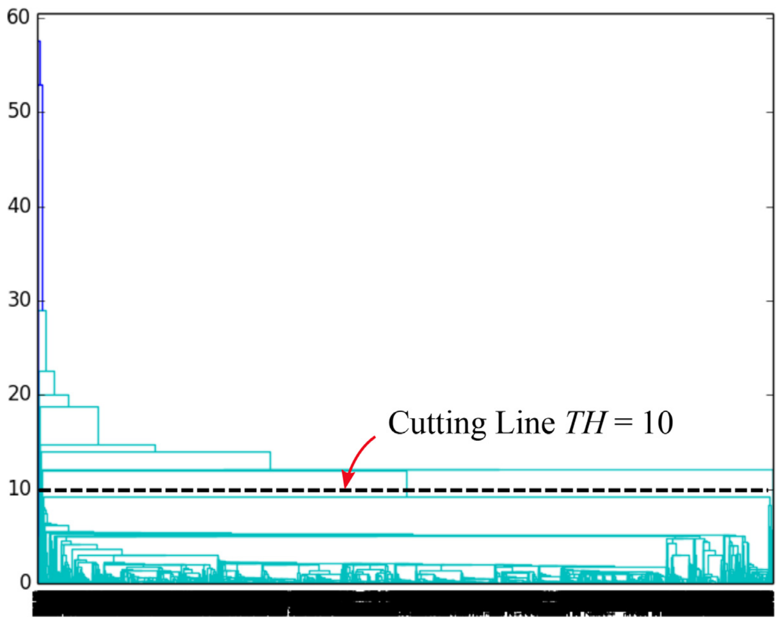


Fig. 7. Hierarchical Clustering and threshold used to obtain a subset of trees T.

(P) and Maxdepth (D) are set to different values in order to study the convergence and quality of the learned functions.

3.4.1. *Function operators*

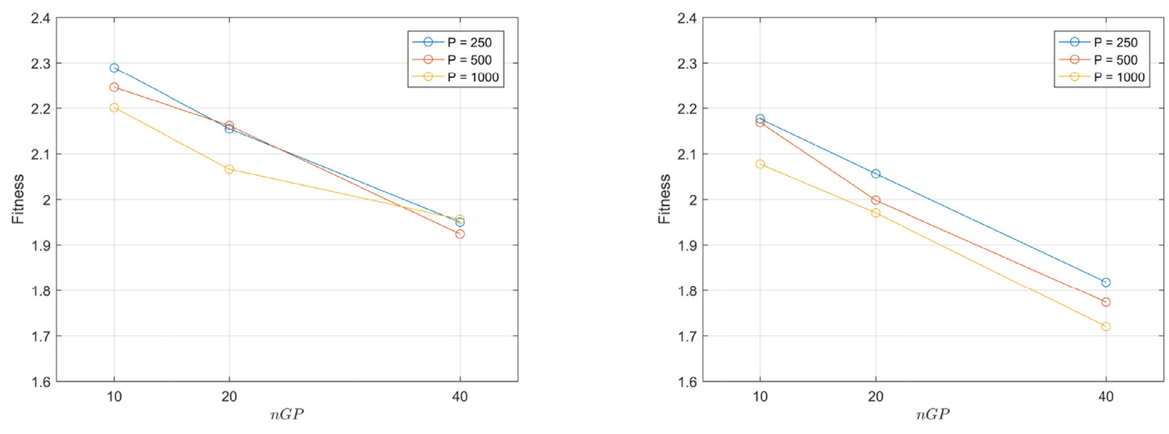
Also, besides common operators such as + , − , *, *tanh* and *exp*, other special operators involve the following definitions: *iflte*$(a, b, c, d) = if(a <= b)$ then c else d; *pdiv*$(a, b) = if(b = 0)$ then 0 else a/b; *psqrt*$(x) = \sqrt{|x|}$; and *plog* $= log|x|$. These above mentioned represent a general and meaningful set to enable the adequate and reasonable search of the space of functions representable by trees. Investigating the role of more specific functions is left for future work.

3.4.2. *Parameters in Genetic Programming*

Furthermore, other parameters in Genetic Programming include a reasonable number of generations G =1000, small values of of Tournament selection size, Mutation Probability, and Elite Fraction; and a large value of Crossover Probability 0.85. The above settings is due to enable a reasonable balance of both elitism (exploitation) and diversity (exploration) in Genetic Programming. Since fine tuning of these parameters will induce into trees being over-fitted to historical training data, we leave fine-tuning out of the scope of this paper.

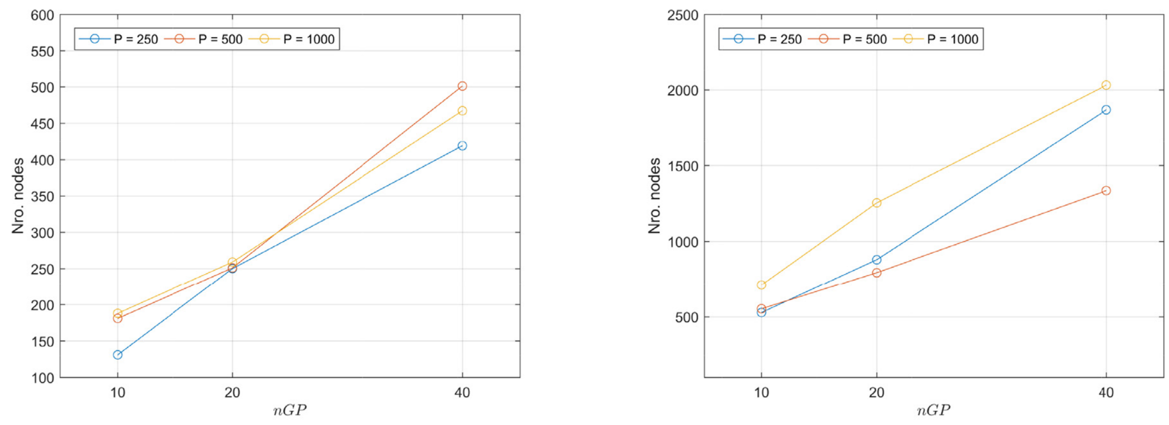
3.4.3. *Breadth, depth and population size in Genetic Programming*

In order to give a glimpse of quality of the converged solutions to solve Eq. (8), Fig. 8 shows the converged values of the fitness



(a) $D = 5$ (b) $D = 10$

Fig. 8. Converged fitness values of Genetic Programming for different Population size P and different MaxDepth of Genetic Programming trees D. Fitness is defined by root mean square (Eq. (8)).



(a) $D = 5$ (b) $D = 10$

Fig. 9. Total number of nodes in Genetic Programming for different population size P and different MaxDepth of Genetic Programming trees D.

when using (a) $D = 5$ and (b) $D = 10$ as MaxDepth in the Genetic Programming trees. These figures show that (1) for any tree depth, increasing the number of multi trees (breadth) improves the fitness performance, and (2) only for higher tree depth, increasing population size improves the fitness consistently. This occurs due to both tree breadth and depth increasing the search space of the tree-based representable functions.

In order to give a glimpse of the size of converged tree solutions, Fig. 9 shows the total number of nodes when using (a) $D = 5$ and (b) $D = 10$ as MaxDepth in the Genetic Programming trees. These figures show that, when increasing the allowable breadth and depth of trees, bloating occurs in our models, which is represented by the increased number of nodes of the converged trees. However, these figures also show that it is possible to control the bloat problem reasonably by using small values of max tree depth ($D = 5$).

Then, in order to show the running behavior when running the Genetic Programming algorithms, Fig. 10 shows the total time in hours to run the $G = 1000$ generations. Clearly, for small depth ($D = 5$) and both any breadth ($nGP < 40$) and any population sizes ($P < 1000$), trees can be obtained within 10 h. However, for large depth ($D = 10$), trees can be obtained within 50 h. Moreover, Fig. 10 shows that for any breadth, depth and for smaller population size, time follows approximately a linear behavior. Investigating both the experimental and theoretical complexity in wider scenarios is another key theme in our agenda for future work.

Then, in order to investigate the trade-off between size and fitness of the evolved trees in Genetic Programming, Figs. 11 and 12 show both the size and fitness for different depth, breadth and population size. These figures show that (1) increasing population and breadth improves the fitness values, and (2) increasing the breadth of trees induces obtaining tree populations having different size yet uniform fitness performance. This result holds independent from population size used during the evolution process. The above results imply that, in the vehicle design layout problem, (1) the learned trees having smaller size are able to achieve the same fitness performance compared to the learned trees having larger size, and (2) increasing the population size is not as advantageous as increasing the tree breadth (nGP) during the learning process.

Thus, due to the above observations, our study uses the best tree obtained by evolving Genetic Programming with depth $D = 5$ and breadth $nGP = 20$. Then, in order to show the type of function that the evolved tree is able to represent after solving Eq. (8), Fig. 13 shows the best learned function f where the weights for each subtree are highlighted in bold. For clarity, the function is not simplified in order to show the weights w_i, in bold, the bias term w_0 and the functions represented by each tree t_i for

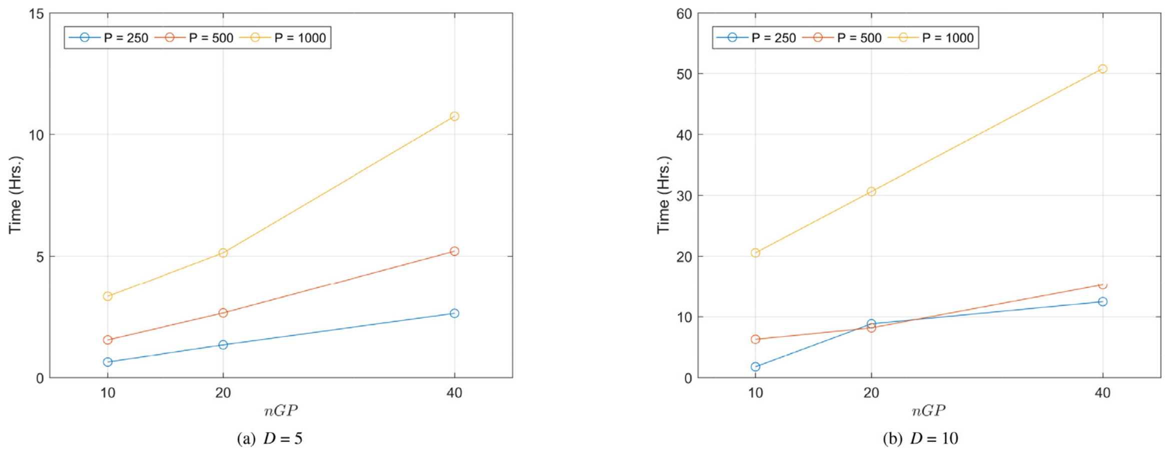


Fig. 10. Total time to evolve Genetic Programming in $G = 1000$ generations for different Population size P and different MaxDepth of Genetic Programming trees D.

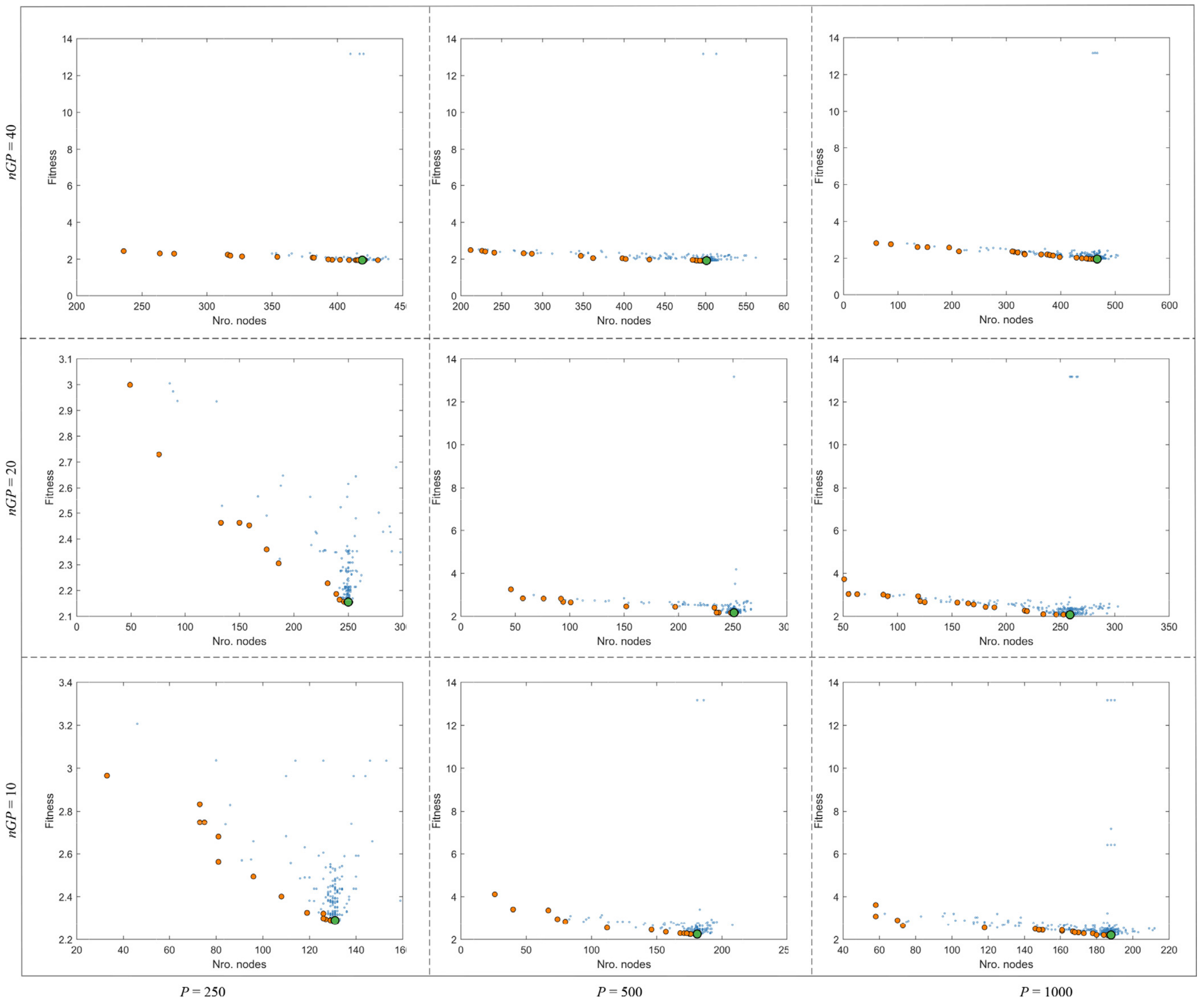


Fig. 11. Size and fitness in tree population for MaxDepth $D = 5$.

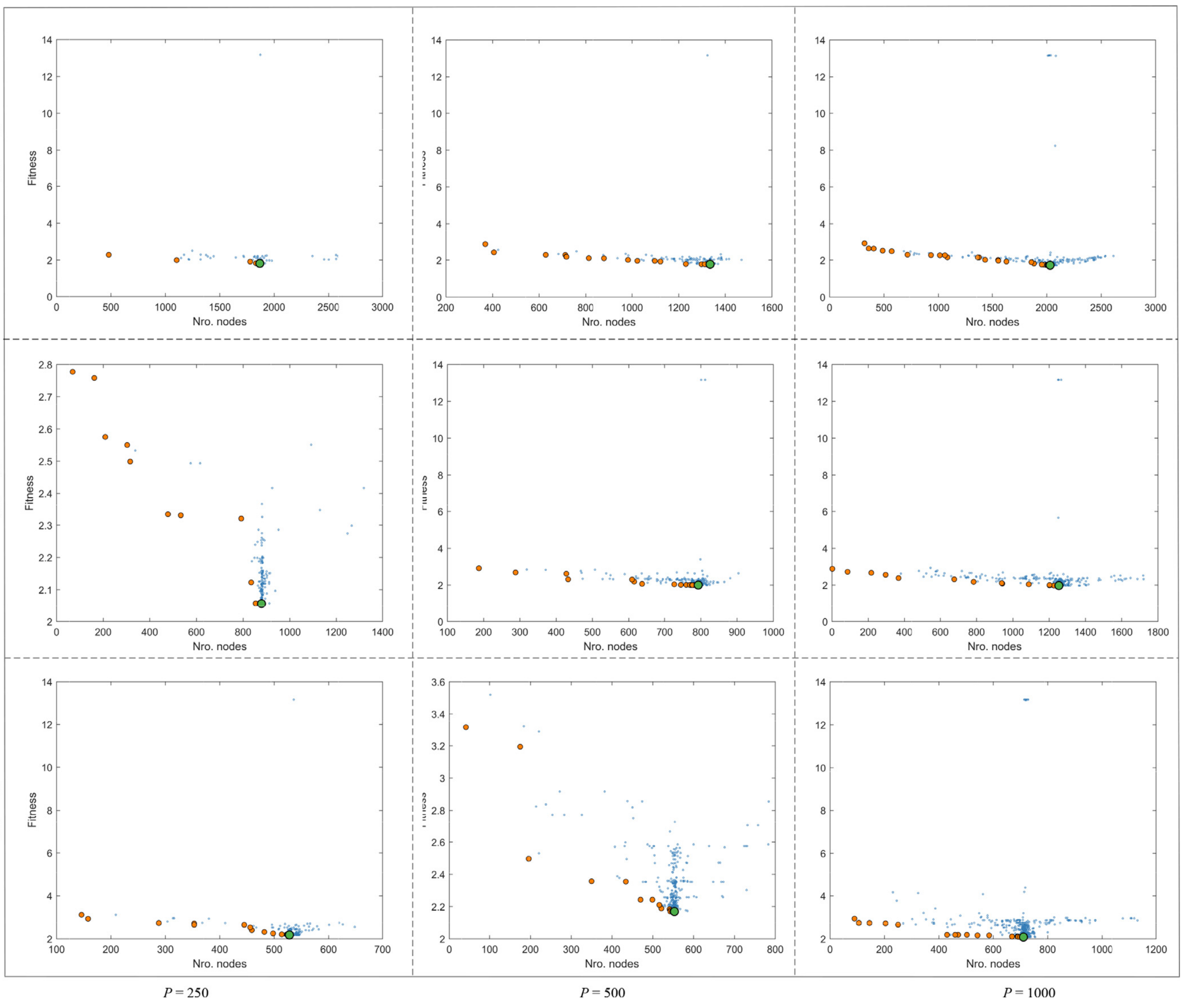


Fig. 12. Size and fitness in tree population for MaxDepth $D = 10$.

$i \in [1, nGP]$. Note that the number of terms in the function is 21, involving $nGP = 20$ trees and one bias term.

3.5. Optimization problem

3.5.1. Gradient-free optimization algorithms

In this section we describe our computational experiments to solve Eq. (1). Here, we use the following gradient-free optimization algorithms:

- NPSO: Particle Swarm Optimization with Niching Properties [18]. The key components of this algorithm integrates the Fitness-Euclidean-Ratio Particle Swarm Optimization (FERPSO) and a Local Search (LS) procedure. The FERPSO has the role of searching around the close solution neighbors with high relative fitness performance, whereas the LS procedures has the role of updating the *personal best* guided by the close *personal best* of neighbor solutions. The role of integrating FERPSO and LS aims at improving the accuracy of converged solutions in multimodal problems.
- DESPS: Differential Evolution-Successful Parent Selection-DE/BEST/1/BIN [19]. This algorithm uses (1) a complementary archive that stores the successful trials and (2) an accumulator that counts the number of unsuccessful trials. Thus, whenever the accumulator is within a user-defined threshold, new trials are computed from the population, otherwise new trials are computed from the archive. The main role of using the archive and the accumulator is to avoid stagnation in which Differential Evolution is unable to generate better solutions. Here, the archive has the role of guiding the exploration towards promising regions of the search space.
- RBDE, Real-Binary Differential Evolution (RBDE) [20]. This algorithm uses the Whitley distribution to generate reference vectors (*parents*) for mutation and crossover in Differential Evolution algorithm. The selection based on the Whitley distribution uses a sorted population and a user-defined parameter to fine-tune for exploration desirability. Compared to the conventional Differential Evolution, where parent selection is equiprobable in the population, RBDE is more exploitative in the sense that the Whitley distribution induces selecting parents with high fitness with better chance.
- SHADE, Success History Parameter Adaptation for Differential Evolution [21]. This algorithm uses an archive of key control parameters (crossover probability and scaling factor) in Differential Evolution algorithm. At every iteration, the control param-

```
-  3.4245
- 0.27*plog(plog(pdivide(minus(17.37,x1),pdivide(x4,x3)))) - 0.01*pdivide(plus(exp(psqroot(x1)),exp(exp(x8))),minus(plus(times(x6,x7),exp(x3)),exp(x8)))
- 58.39*pdivide(psqroot(x4),times(x5,x6)) - 55.56*psqroot(plus(iflte(plus(x1,x1),x2,plog(x9),exp(x8)),minus(minus(x2,x3),plus(x3,x5))))
+ 0.21*pdivide(pdivide(iflte(x7,exp(x3),plog(x9),plus(x3,16.81)),minus(pdivide(x1,x8),exp(x5))),times(minus(pdivide(x2,x4),plus(x4,17.61)),minus(x3,x5)))
- 82.99*exp(exp(tanh(tanh(x7)))) - 8.29*times(plus(plus(x8,tanh(x5)),times(tanh(x5),x5)),plus(plus(x6,x6),tanh(x4)))
+ 0.16*times(plus(plog(minus(x8,x3)),pdivide(plus(x7,x2),exp(x8))),pdivide(minus(plus(17.71,17.87),pdivide(x1,x8)),times(exp(x5),times(x5,x5))))
+ 54.51*pdivide(plog(minus(x1,x3)),times(tanh(x5),x5)) + 5.54*plog(iflte(exp(x8),plus(psqroot(x1),x3),psqroot(plus(x4,x6)),pdivide(pdivide(x4,x3),x9)))
+ 0.18*exp(x7)  - 9.88*plus(pdivide(plog(minus(x8,x3)),times(exp(x3),exp(x3))),x3)
- 1.33*pdivide(exp(pdivide(x7,x9)),iflte(x4,plus(exp(x3),tanh(x7)),plog(x4),pdivide(pdivide(x8,x3),x9))) + 257.56*exp(tanh(x7))
+ 55.02*psqroot(plus(x2,pdivide(x1,x8))) + 1.4*pdivide(iflte(plus(x5,psqroot(x8)),plus(x6,x6),exp(x4),plus(x1,plus(x1,x1))),exp(psqroot(pdivide(x1,x9))))
+ 62.26*plus(plus(x8,x5),x5) - 61.41*plog(pdivide(plus(plog(x1),plus(x1,x1)),tanh(tanh(x7))))
+ 39.32*times(times(psqroot(minus(x2,17.21)),plog(x6)),pdivide(x6,plus(17.21,tanh(x3))))
- 0.17*times(x3,pdivide(minus(-49.89,plus(17.37,x2)),times(times(x5,x5),2x6))))
```

Fig. 13. The best function f minimizing Eq. (8) for depth $D = 5$ and breadth $nGP = 20$.

eters are updated by a normal distribution that uses an arbitrary mean chosen from the successful parameters over a user-defined learning period. Compared to the conventional Differential Evolution, where crossover probability and scaling factor are user-defined constants, SHADE is self-adaptive in the sense of using past successful trials to update future parameters. Thus, the main role of adapting the control parameters in Differential Evolution is to robustly guide the search process by using a diverse set of successful parameters in a historical learning period, each of which is updated at every iteration.

- DEGL, Differential Evolution using a Neighborhood-based Mutation [22]. This algorithm uses global and local interpolation vectors in the mutation process of Differential Evolution. Here, the global (local) interpolation vector is a solution in the direction towards the global best (local best) of the the population (assuming a ring-based neighborhood topology). The role of considering global and local interpolation vectors is to balance the exploration over local neighborhoods (useful in multimodal problems) with the exploitation of the entire search space (global best).
- NOMAD, Nonlinear Optimization with the MADS Algorithm [23]. This algorithm implements the Mesh Adaptive Direct Search in which the search process occurs at four stages in every iteration: (1) search over a finite number of vectors inside in lattices (meshes), (2) if the previous step failed, search over the local neighborhood that consists of vectors in the direction of a finite set of poll directions and vectors having close fitness performance, (3) if the previous steps failed, search over the local neighborhood consisting of dominant vectors, and (4) mesh coarsening occurs if successful non-dominated vector vector is found. The unique point of NOMAD is to extend the Generalized Pattern Search method to include a balance of exploration and exploitation through lattice coarsening, neighborhood and dominance concepts.
- DIRECT, Direct Global Optimization Algorithm [24]. This algorithm implements the DIviding RECTangles concept, which samples vectors at the center of hypercubes and subdivides potentially optimal hypercubes recursively. The main advantage of DIRECT is to allow for both equally and locally distributed sampled vectors over the search space.
- BIPOPCMAES, BI-Population Covariance Matrix Adaptation Evolution Strategy [25]. This algorithm samples vectors from a multi-variate normal distribution with zero mean and a covariance matrix which is updated with best sampled vectors. Also, multi-start occurs using population size of either increasing size or varying size, subject to a budget of total number of function evaluations. The purpose of combining multi-start regimes of different regimes is to first allow exploration (using large population size), followed by exploitation (smaller population size).

The reason behind using the above algorithmic set is to rigorously tackle the studied nonlinear optimization problem using a representative class of gradient-free algorithms. That is an algorithmic set tackling nonlinear optimization problems while considering multimodality, parameter adaptation, search memory, selection pressure, probability distribution, neighborhood concepts, search space partitioning, and (adaptive) meshing. The parameters used for each algorithm are default and described in the references. Fine tuning the local algorithmic parameters to solve Eq. (1) is out of the scope of this paper. The datasets and source codes are to be open-source upon publication of this manuscript.

3.5.2. Search scenarios

Furthermore, in order to evaluate a wide space of vehicle layouts, we considered the following scenarios in our computational experiments:

- *Scenario A*, a tight scenario, in which the search space is restricted to the boundaries of the observed historical points, that is the box-bounding constraint $\mathbf{X}$ is set to $[min(S), max(S)]$.
- *Scenario B*, a relaxed scenario, in which the search space is allowed to expand by some small factor, that is the box-bounding constraint $\mathbf{X}$ is set to $[\frac{min(S)}{2}, \frac{3max(S)}{2}]$.

The above experimental settings are chosen mainly due to our purpose of evaluating and comparing the kind of vehicle models able to be found by the above described algorithms when the search space is both constrained and unconstrained by history. Evaluating other scenarios such as focusing on a certain type of manufacturers, shapes and user preferences is part of our agenda in future work.

3.5.3. Results

Then, we show in Figs. 14 and 15 the convergence behavior over 30 independent runs up to 10,000 evaluations in all the algorithms and all scenarios described above. These figures show that, compared to the Scenario 2, the algorithms running in the Scenario 1 converge to numerous local optima and have difficulty obtaining uniform convergence throughout all the running instances. We believe this occurs due to the highly nonlinear nature of the vehicle design problem and the narrow search space (since bounds in Scenario 1 are set to historical bounds). Thus, as confirmed by the results in Scenario 2, it is possible to achieve better converged objective functions mainly due to the widened search space.

The following presents some discussions on the assessed gradient-free optimization algorithms on the tight/relaxed search space.

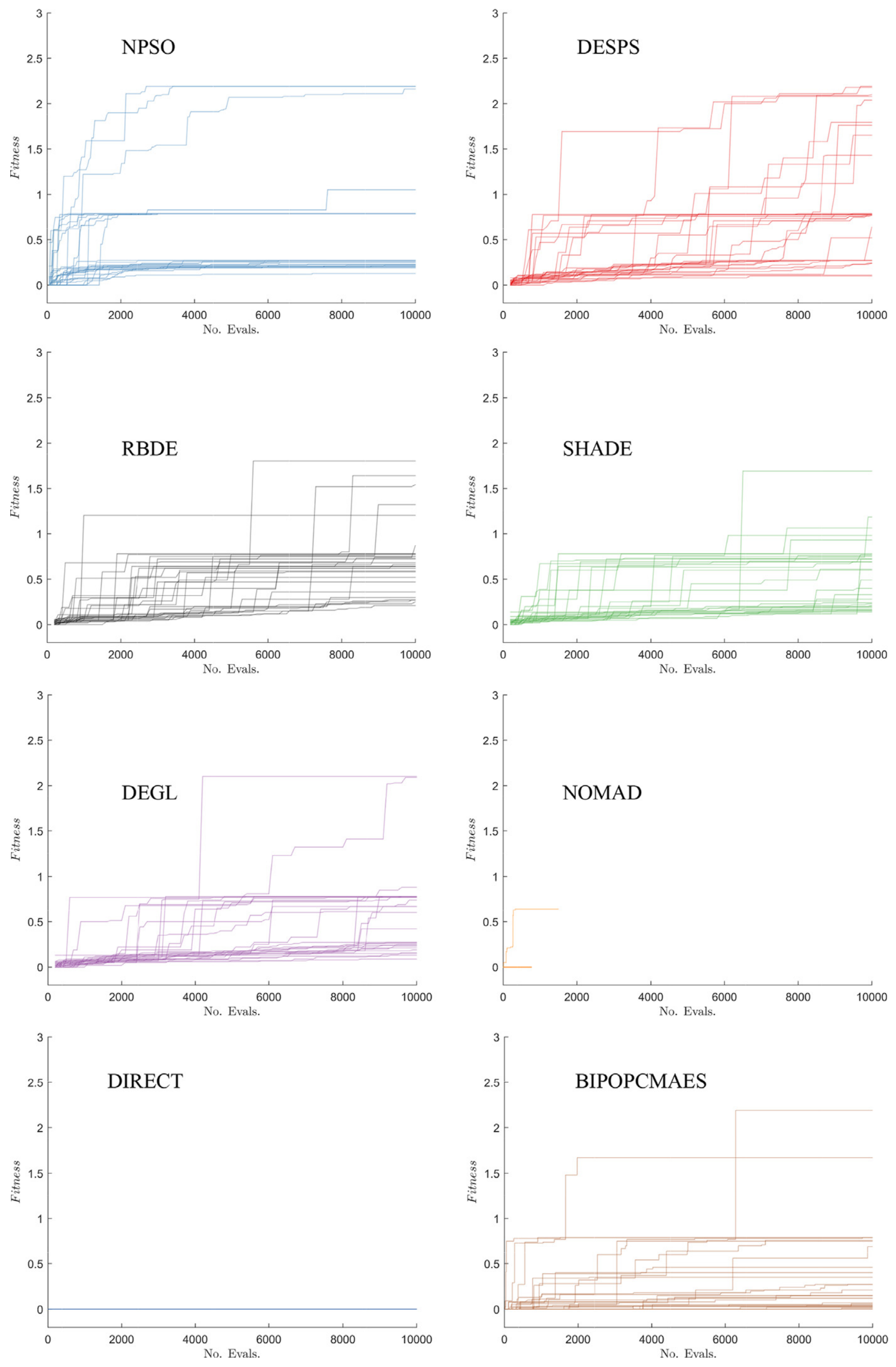


Fig. 14. Convergence of vehicle design problem for Scenario 1.

- NPSO
  Under *tight search space*, Scenario 1, the high number of stagnation cases in local optima can be explained due to the fact of having the concept of closeness to guided the search towards the *personal best* in the neighborhood of sampled vectors. Thus, under the *relaxed search space*, Scenario 2, it becomes more likely to find high-performing and novel vehicle designs, and less likely to stagnate in local optima.
- DESPS
  Under *tight search space*, Scenario 1, the common stagnation in local optima can be explained due to (1) the hardness of finding high-performing and novel vehicle designs considering historical bounds, and (2) the small threshold value, 128, for the accumulator for unsuccessful trials and small number of population size, 100. These facts are conducive to think that the archive has higher chance to store solutions with low diversity and almost equal fitness. In line of the above considerations, under the *relaxed search space*, Scenario 2, the algorithm is able to generate good solutions with better chance. Then, this observation implies that the archive in the algorithm is able to store more diverse solutions when broadening the search space; thus, the algorithm is better guided to explore towards promising regions of the search space; and less likely to incur in stagnation of local optima.

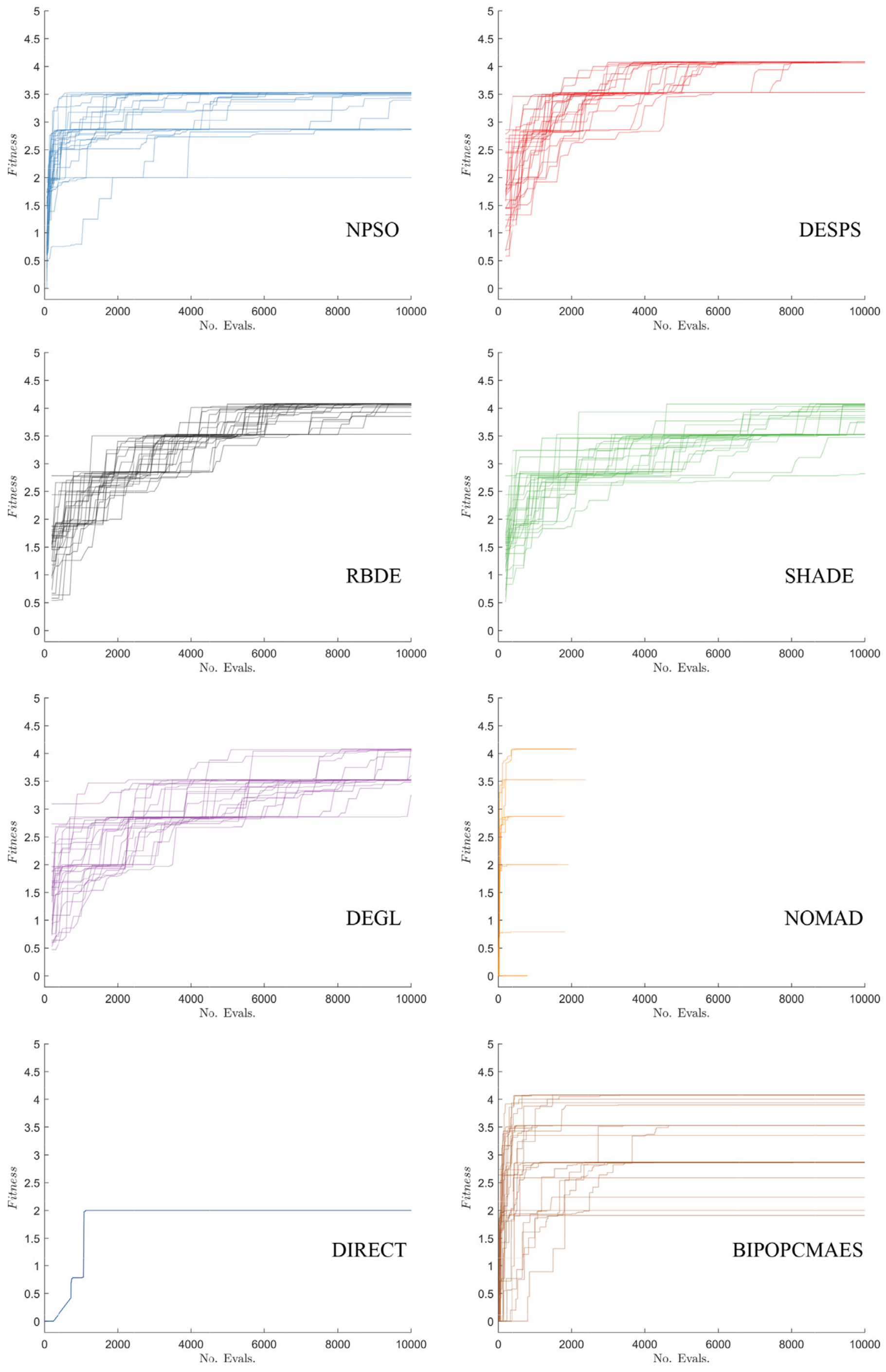


Fig. 15. Convergence of vehicle design problem for Scenario 2.

- RBDE
  Under *tight search space*, Scenario 1, the high number of local-optima stagnation cases can be explained due to (1) the hardness of finding high-performing and novel vehicle designs considering historical bounds, and (2) the large value of the pre-defined parameter to fine-tune for exploration desirability, which biases towards high explorative behavior of the search space. These observations imply that exploration is less useful when the search space is tight, as the one having historical bounds, and induces to have a high number of unsuccessful trials. However, under the *relaxed search space*, Scenario 2, the algorithm is able to find good solutions, comparable to the performance of DESPS. This fact implies that the exploration ability of the algorithm is useful when the search space is broadened by some factor, and that the Whitley distribution is useful to guide the selection of promising solutions.
- SHADE
  Under *tight search space*, Scenario 1, the high number of local-optima stagnation cases can be explained due to (1) the hardness of finding high-performing and novel vehicle designs considering historical bounds, and (2) the use of normal distribution over an averaged historical learning period to update the crossover probability and scaling factor. These observations is conducive to think that self-adaptation considering the

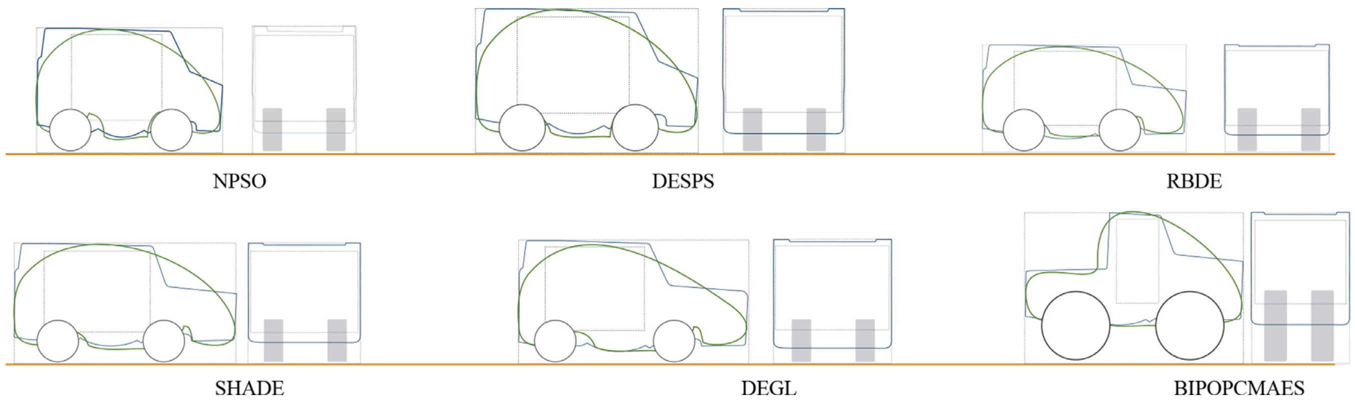


Fig. 16. Lateral and Front view of best solutions for the vehicle design problem in Scenario 1.

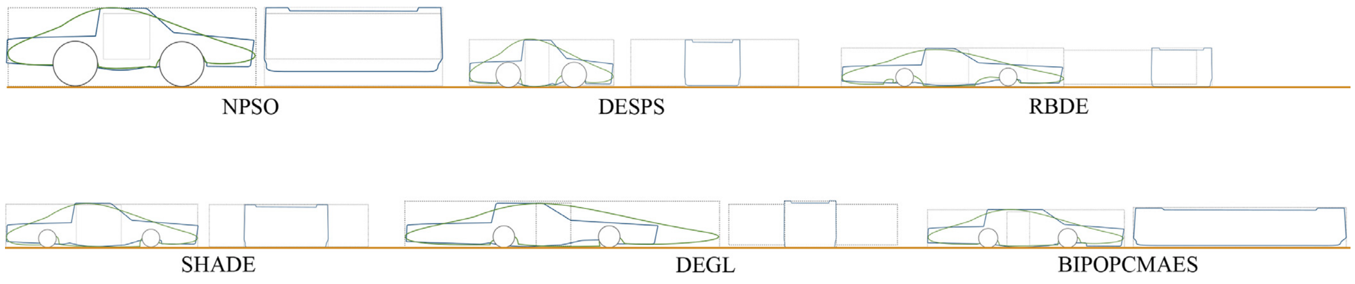


Fig. 17. Lateral and Front view of best solutions for the vehicle design problem in Scenario 2.

historical learning period is irrelevant in the tight search space. In line of the above, under the *relaxed search space*, Scenario 2, the algorithm is able to find good solutions with better chance. This observation implies that the usefulness of self-adaptation relies on the ability to find successful trials, where a relaxed search space induces in useful crossover probability and scaling factors.

- DEGL
  Under *tight search space*, Scenario 1, the high number of local-optima stagnation cases can be explained due to (1) the hardness of finding high-performing and novel vehicle designs considering historical bounds, and (2) the small weights used in the computation of the global (local) the interpolation vectors and (3) the large value of crossover probability encouraging a highly-explorative behavior. This observation induces to sample solutions which are closer to local optima. In line of the above considerations, under the *relaxed search space*, Scenario 2, the algorithm has higher chance to find good solutions. The reason for this behavior is due to crossover helping to explore over a widened search space; thus exploration provides a diverse set from which the next promising solutions are sampled to form a closer region in reference to the global/local the interpolation vectors.
- NOMAD
  Under *tight search space*, Scenario 1, the high number of local-optima stagnation cases can be explained due to (1) the hardness of finding high-performing and novel vehicle designs considering historical bounds, and (2) the fixed approach to sample solutions in the lattices (meshes) and (3) the space complexity that is involved in mesh coarsening while sampling a large number of solutions. This observation imply that the algorithm is less likely to have a diverse set of promising solutions. Under the *relaxed search space*, Scenario 2, the algorithm is able to find better solutions, some of which are of comparable performance to DESPS and RBDE. The main reason for early stopping is the upper bound for allowable cache memory (2000MB). These observations imply that mesh coarsening and sampling the neighborhood consisting of solutions with similar and dominant performance is useful for fast convergence to find high-performing and novel vehicle designs.
- DIRECT
  Under *tight search space*, Scenario 1, and *relaxed search space*, Scenario 2, the algorithm has the worst performance due to the fixed approach to sample solutions in the hypercubes, which mainly depends on the lower/upper bounds of the search space (which is fixed in both scenarios). Compared to NOMAD, wherein mesh coarsening occurs when successful solutions are found, the DIRECT algorithm has no form of self-adaptation in hypercube formation.
- BIPOPCMAES
  Under *tight search space*, Scenario 1, and the *relaxed search space*, Scenario 2, the high number of local-optima stagnation cases can be explained due to (1) the hardness of finding high-performing and novel vehicle designs considering historical bounds, (2) the use of a normal distribution that uses a covariance matrix updated from averaged best previous solutions. These observations are indications that the normal distribution using a historical covariance is less useful as a balancing mechanism to first allow exploration (using large population size), followed by exploitation (smaller population size).

3.5.4. Examples of vehicle layouts

Furthermore, in order to exemplify the kind of vehicle layout solutions we are able to obtain, we show the best solutions over 30 independent runs in Tables 4 and 5. And, to compare performance of the objective functions, we also provide the estimated performance function $f(x)$ (vehicle fuel efficiency) and the objective function $N(x).f(x)$ (Eq. (1)) in Tables 4 and 5.

In order to have a suitable conceptualization of the type of vehicle layouts which all the algorithms are able to find, Figs. 16 and 17 show vehicle layout estimations based on the optimized principal lengths x_4, x_5, ..., x_9 (described in Table 2). For simplicity

Table 4
Best optimized variables and objective performance over 30 independent runs for Scenario 1.

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	$f(x)$	$N(x).f(x)$
NPSO	55.58	527.45	0.54	2.77	1.55	1.87	1.35	1.47	1.28	200	2.19
DESPS	72.06	526.06	0.54	3.29	1.8	2.13	1.66	1.72	1.43	199.66	2.19
RBDE	69.7	466.96	0.62	3.01	1.64	1.59	1.51	1.52	1.11	164	1.8
SHADE	68.05	503.57	0.59	3.27	1.66	1.78	1.57	1.59	1.2	154.22	1.69
DEGL	92.35	383.68	0.54	3.4	1.75	1.83	1.47	1.68	1.24	191.65	2.1
NOMAD	4	550	0.53	2.74	1.39	2.29	4.25	1.15	1.62	173.26	0.64
DIRECT	48.2	289	2.76	4.1	1.68	1.66	2.44	1.44	1.27	13.84	0
BIPOPCMAES	89.87	414.96	0.53	3.23	1.45	2.23	0.63	1.35	1.24	199.95	0.79

Table 5
Best optimized variables and objective performance over 30 independent runs for Scenario 2.

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	$f(x)$	$N(x).f(x)$
NPSO	111.37	425.15	0.52	3.41	2.45	1.08	0.64	2.41	0.63	200	3.53
DESPS	127.04	670.54	0.35	1.99	0.76	0.63	0.33	2.31	0.64	199.97	4.08
RBDE	118	270.59	0.48	3.06	0.84	0.53	0.59	2.33	0.47	199.89	4.08
SHADE	137.83	460.4	0.41	2.64	1.16	0.6	0.62	2.19	0.58	199.69	4.08
DEGL	113.85	568.14	0.38	4.33	0.71	0.64	0.48	2.33	0.57	199.57	4.08
NOMAD	54.18	339.72	0.36	1.37	2.93	0.55	0.32	0.6	0.66	200	4.08
DIRECT	124.27	419.5	6.29	6.04	1.07	1	0.63	2.56	2.32	200	2
BIPOPCMAES	70.72	825	0.27	2.71	2.96	0.52	0.32	0.57	0.47	200	4.08

and without loss of generality, both curved and non-curved vehicle shape estimations are provided. Then, from the above descriptions, we can observe that (1) it is possible to obtain vehicle layouts being both unique and high performing when the search space of vehicle layouts is bounded by history, and (2) it is possible to achieve improved novelty and performance when the search space of vehicle layouts is widened by a small factor. It is desirable to contrast the realizability of the generated models by using further vehicle powertrain modeling and experimentation, which is in our future agenda.

4. Conclusion

In this paper we propose a new approach to search for unique and high-performing vehicle design layouts given observations of historical data. The basic idea in our proposal is to first learn a surrogate vehicle performance function that approximates the real-world features from past historical data, and then optimize the learned function, along with a newly propose novelty metric, to enable high-performance and uniqueness. The proposed novelty function is defined as the dissimilarity degree with respect to historical clusters, and we believe it may be of interest to the community.

Our application involving real world data of more than twenty thousand vehicles from the last 30 years, along with rigorous and extensive computational experiments based on Genetic Programming and a eight relevant gradient-free optimization algorithms, has shown the feasibility and usefulness of our proposed approach. More specifically, our results show that (1) it is possible to learn vehicle performance functions being succinct and economical with the lowest estimation error (where error is comparable to large models), (2) increasing tree breadth is effective to reduce the risk of bloating in Genetic Programming, (3) it is possible yet hard to obtain novel car layouts with fairly competitive fuel efficiency ratios under historical boundaries, and (4) it is possible and simple to obtain unique vehicle layouts with much improved fuel-efficiency ratios compared to the historical boundaries.

The future work in our agenda aims at exploring the generalizable abilities of our proposed ideas to tackle nonlinear design problems, as well as investigating the trade-off between novelty and performance in the vehicle design problem. We believe that further advances on learning and pattern recognition algorithms for real world product design will bring further insights to build unique and high-performing systems.

Acknowledgment

We acknowledge the support from Kakenhi No. 15K18095 to fund this work.

References

[1] Y. Kitamura, R. Mizoguchi, Ontology-based systematization of functional knowledge, J. Eng. Des. 15 (4) (2004) 327–351.
[2] F.E.H. Tay, J. Gu, A methodology for evolutionary product design, Eng. Comput. 19 (2–3) (2003) 160–173.
[3] T. Miyashita, D. Satoh, An improvement method of conceptual design ideas using data envelopment analysis, in: Proceedings of International Conference on Engineering Design, ICED, 2009, pp. 6.13–6.22.
[4] Karniel, Y. Reich, Multi-level modelling and simulation of new product development processes, J. Eng. Des. 24 (3) (2013) 185–210.
[5] L. Wang, H.C. Amos, K. Deb, Multi-objective Evolutionary Optimisation for Product Design and Manufacturing, SpringerLink, 2011.
[6] M. Kowalik, C.M. Gothard, A.M. Drews, N.A. Gothard, A. Weckiewicz, P.E. Fuller, B.A. Grzybowski, K.J.M. Bishop, Parallel optimization of synthetic pathways within the network of organic chemistry, Angew. Chem. 51 (32) (2012) 7928–7932.
[7] T.A. Whitehead, A. Chevalier, Y. Song, C. Dreyfus, S.J. Fleishman, C.D. Mattos, C.A. Myers, H. Kamisetty, P. Blair, I.A. Wilson, D. Baker, Optimization of affinity, specificity and function of designed influenza inhibitors using deep sequencing, Nat. Biotechnol. 30 (2012) 543–548.
[8] S. Vajna, S. Clementa, A. Jordan, Autogenetic design theory: an approach to optimise both the design process and the product, in: Proceedings of ASM Conference Proceedings, Quebec, Canada, 2002, pp. 51–57.
[9] S. Achiche, F.P. Appio, T.C. McAloone, A.D. Minin, Fuzzy decision support for tools selection in the core front end activities of new product development, Res. Eng. Des. 24 (1) (2013) 1–18.
[10] E. Ostrosi, S.T. Bi, Generalised design for optimal product configuration, Int. J. Adv. Manuf. Technol. 43 (1–4) (2010) 13–25.
[11] C.T. Lin, C.T. Chen, A fuzzy-logic-based approach for new product go/nogo decision at the front end, IEEE Trans. Syst. Man Cybern. Part A – Syst. Hum. 34 (1) (2004) 132–142.
[12] Z. Li, Z. Cheng, Y. Feng, J. Yang, An integrated method for flexible platform modular architecture design, J. Eng. Des. 24 (1) (2013) 25–44.
[13] V. Parque, M. Kobayashi, M. Higashi, Reinforced explorit on optimizing vehicle powertrains, Neural Information Processing, Lecture Notes in Computer Science, 8227, Springer, Berlin, Heidelberg, 2013, pp. 579–586.
[14] P.M. Grignon, G.M. Fadel, Configuration design optimization method, in: Proceedings of ASME Design Engineering Technical Conferences, DETC, September 12–15, Las Vegas, Nevada, 1999.
[15] R. Sibson, SLINK: an optimally efficient algorithm for the single-link cluster method, Comput. J. 16 (1) (1973) 30–34.
[16] C.B. Barber, D.P. Dobkin, H. Huhdanpaa, The quick hull algorithm for convex hulls, ACM Trans. Math. Softw. 22 (4) (1996) 469–483.

[17] S.S. Skiena, Convex hull, The Algorithm Design Manual, Springer-Verlag, New York, 1997, pp. 351–354.

[18] B.Y. Qu, J.J. Liang, P.N. Suganthan, Niching particle swarm optimization with local search for multi-modal optimization, Inf. Sci. 197 (2012) 131–143.

[19] S.-M. Guo, C.-C. Yang, P.-H. Hsu, J.S. Tsai, Improving differential evolution with a successful-parent-selecting framework, IEEE Trans. Evol. Comput. 19 (2015) 717–730.

[20] A.M. Sutton, et al., Differential evolution and non-separability: using selective pressure to focus search, in: Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference, GECCO, 2007, pp. 1428–1435.

[21] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Proceedings of IEEE Congress Evolutionary Computation, Cancun, Mexico, 2013, pp. 71–78.

[22] S. Das, et al., Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.

[23] S.L. Digabel, Algorithm 909: NOMAD: nonlinear optimization with the MADS algorithm, ACM Trans. Math. Softw. 37 (4) (2011) 44:1–44:15.

[24] D.R. Jones, Direct global optimization algorithm, in: Encyclopedia of Optimization, Kluwer Academic Publishers, 1999.

[25] N. Hansen, Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed, in: Proceedings of Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference, ACM, 2009, pp. 2389–2395.

[26] V. Parque, T. Miyashita, Learning the optimal product design through history, Proceedings of the 22nd International Conference on Neural Information Processing, pp. 382–389, Turkey.

[27] X. Li, A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio, in: Proceedings of Genetic and Evolutionary Computation Conference, GECCO '07, 2007, pp. 78–185.

[28] R. Akai, H. Amaya, K. Fujita, Product family deployment through optimal resource allocation under market system, in: Proceedings of ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Canada, 2010, pp. 277–290.

Victor Parque is Assistant Professor at the Department of Modern Mechanical Engineering, Waseda University, as well as Associate Professor at Egypt–Japan University of Science and Technology. After joining Engineering Complex in 2003, he obtained the MBA degree from Esan University in 2009, and obtained the doctoral degree from Waseda University in 2011. He was a PostDoctoral Fellow at Toyota Technological Institute from 2012 to 2014. His research interests include Learning and Intelligent Systems and its applications to Design Engineering and Control.

Tomoyuki Miyashita is Professor at the Department of Modern Mechanical Engineering, Waseda University. After joining Nippon Steel Co. Ltd. in 1992, He obtained the doctoral degree from Waseda University in 2000. He was Research Associate at Waseda University and Ibaraki University since 2000 and 2002, respectively. He became Associate Professor and Professor at Waseda University in 2005 and 2007, respectively. His research interests are Design Process, Design Optimization, Organ Deformation.