A Dimensionally-Aware Genetic Programming Architecture for Automated Innovization

Sunith Bandaru and Kalyanmoy Deb

KanGAL Report Number 2012015

Abstract Automated innovization is an unsupervised machine learning technique for extracting useful design knowledge from Pareto-optimal solutions in the form of mathematical relationships of a certain structure. These relationships are known as design principles. Past studies have shown the applicability of automated innovization on a number of engineering design optimization problems using a multiplicative form for the design principles. In this paper, we generalize the structure of the obtained principles using a tree-based genetic programming framework. While the underlying innovization algorithm remains the same, evolving multiple trees, each representing a different design principle, is a challenging task. We also propose a method for introducing dimensionality information in the search process to produce design principles that are not just empirical in nature, but also meaningful to the user. The procedure is illustrated for three engineering design problems: two-bar truss design, welded-beam design and metal-cutting process optimization.

1 Introduction

In recent years there has been a growing interest in the field of post-optimality analysis. In a single objective scenario, this usually concerns the optimality, sensitivity and robustness studies on the obtained solution. Multi-objective optimization on the other hand, poses an additional challenge in that there are a multitude of possible solutions (when the objectives are conflicting) which are all said to be Pareto-optimal. The data-mining of Pareto-optimal solutions has received particular attention as it can reveal certain characteristic features exclusive to these solutions, which can often be generalized to the entire Pareto-optimal front. In practical problem solving,

Sunith Bandaru and Kalyanmoy Deb
Kanpur Genetic Algorithms Laboratory,
Indian Institute of Technology Kanpur, Kanpur 208016, India
e-mail: `{sunithb,deb}@iitk.ac.in`

the knowledge of these features can give the designer a better understanding of the problem structure. Most studies in this direction rely on visual means of identifying the features. For example, a heatmap of the Pareto-optimal variable values can reveal the general distribution of these solutions in the decision space [18, 25]. The use of parallel coordinate plots [8], ANOVA, decision trees [21] and clustering in objective [22] and decision space has also been proposed. For example, the use of self-organizing maps (SOMs) to condense the information embedded in the multi-dimensional objective and decision spaces a lower-dimensional (generally two) map was proposed in [15]. A SOM preserves the topology of the higher-dimensional space, meaning that nearby points in the input space are mapped to nearby units in SOM. Thus it can serve as a cluster analyzing tool for high-dimensional data [15]. The method though unsupervised still requires the users to deduce knowledge, such as the role of design variables in trade-offs, visually through these maps.

Deb and Srinivasan [7] describe the concept of *innovization* by defining the special features as commonalities among the Pareto-optimal solutions. These commonalities (or invariants) are given mathematical forms, called *design principles*, by performing regression between variables and/or objectives using appropriate functions. However, regression can only be performed when a correlation is observed between the regressed entities. Innovization, in its original form, required users to identify this correlation visually through two and three dimensional plots. This *manual innovization* task is therefore limited to features of Pareto-optimal solutions present in humanly perceivable dimensions.

Automated innovization [2] is an unsupervised machine learning technique that can identify correlations in any multi-dimensional space formed by variables, objectives, etc. specified by the user and subsequently performs a selective regression on the correlated part of the Pareto-optimal dataset to obtain the design principle $\psi(\mathbf{x})$. The procedure was later extended [1] so that design principles hidden in all possible Euclidean spaces formed by the variables and objectives (and any other user-defined functions) can be obtained simultaneously without any human interaction. The regression assumes the following mathematical structure for the design principle,

$$\psi(\mathbf{x}) = \prod_{j=1}^{N} \phi_j(\mathbf{x})^{a_j b_j}, \tag{1}$$

where ϕ_j's are N basis functions (variables, objectives functions, constraints etc.) specified by the user which can have Boolean exponents a_j and real-valued exponents b_j. It has been argued that since many natural, physical, biological and man-made processes are governed by formulae with the same structure (power laws [14]), most correlations are expected to be mathematically captured by it. By definition, $\psi(\mathbf{x})$ is a design principle if it is invariant, i.e. $\prod_{j=1}^{N} \phi_j(\mathbf{x})^{a_j b_j} = c$ is true for a majority of the Pareto-optimal solutions, for some constant c. However, due to the approximate nature of Pareto-optimal datasets, the equality relation may not hold strictly and hence the extent of commonality of a design principle $\psi(\mathbf{x})$ is obtained by clustering the set of c-values. The minimization of equal-weighted sum of (i) number of clusters ($\mathcal{C}$), and (ii) percentage coefficient of variance ($c_v = \sigma/\mu$)

within these clusters, has been proposed in [2] to obtain design principles. An optimization problem that uses this weighted objective function,

$$\text{Minimize} \quad \mathcal{C} + \sum_{k=1}^{\mathcal{C}} c_v^{(k)} \times 100\% \qquad \text{where } c_v^{(k)} = \frac{\sigma_c}{\mu_c} \quad \forall\, c \in k\text{-th cluster}, \tag{2}$$

is formulated with a_j's represented by an N-bit binary variable string and b_j's as N real variables. The algorithmic calculation of the objective function requires the use of a derivative-free optimization method like genetic algorithms (GA). The population based approach of GA also enables obtaining multiple design principles simultaneously using a niching strategy [1]. Given enough GA generations, the final population will contain all possible design principles that fit the form in Eq. (1).

In this paper, we generalize the mathematical structure of the design principles in Eq. (1) using parse tree representation. The overall automated innovization problem is solved using a system that integrates a GA with a genetic programming algorithm for handling such parse trees. The rest of the paper is organized as follows. Sect. 2 introduces genetic programming in general and discusses various evolutionary operators, namely initialization, fitness calculation, selection, crossover and mutation, in the context of automated innovization. In Sect 3 we describe the procedure for introducing dimensional-awareness in the proposed system to obtain meaningful design principles. Results on three standard engineering design problems are presented in Sect.4.

2 Genetic Programming for Automated Innovization

At an abstract level, genetic programming (GP) is a *weak* search algorithm for automatically generating computer programs to perform specified tasks [12]. Weak search methods do not require the user to know or specify the form or structure of the solution in advance [10]. Most GP implementations employ an evolutionary algorithm as the main search engine. However, simulated annealing, hill climbing approaches and estimation of distribution algorithms (EDAs) have also been used in literature [17]. Like other evolutionary computation techniques, a typical GP starts with a population of randomly created individuals, which in this case are programs. The fitness for each individual is determined by running the program. High fitness individuals are selected to form the mating pool, on which primary genetic operations, namely crossover and mutation, are applied to create a new population of programs. The process is repeated until some stopping criterion (like maximum number of generations) is met.

Most GP systems evolve programs in a domain-specific language specified by primitives called *functions* and *terminals*. The terminal set ($\mathcal{T}$) may consist of the program's external inputs, ephemeral random constants, and nullary (zero-argument) functions/operators where as the function set ($\mathcal{F}$) may contain operators (arithmetic, Boolean, conditional, etc.), mathematical functions and constructs (loops, for example) that are defined in the language being used. Computer programs are traditionally represented in the memory as parse trees made up of such primitives. Other common ways of expressing programs include linear and graph-based representations.

The most common application of GP has been to the process of induction of mathematical models based on observations. This process is known by the names model induction, system identification and symbolic regression depending on the purpose. The power of GP algorithms to evolve models in a symbolic form without assuming the functional form of the underlying relationship can also be applied to automated innovization. In this paper, we generalize the mathematical structure of the design principles in Eq. (1) by representing $\psi(\mathbf{x})$ using parse trees composed of the N basis functions and real-valued ephemeral constants as terminals, i.e $\mathcal{T} = \{\phi_1, \phi_2, \ldots, \phi_N, \mathcal{R}\}$ and a user-specified function set $\mathcal{F}$. Fig. 1 shows two examples of parse trees and their corresponding $\psi(\mathbf{x})$ expressions obtained by inorder depth-first tree traversal. By starting with a population of such trees and using the objective function in (2), it is possible to evolve design principles of a generic form using a GP system. In the following sections we discuss each step of the (GA + SmallGP [16]) system used in this work. Some of these steps are standard and hence described only briefly, while others which have been modified to suit the requirements of automated innovization are explained in more detail.

2.1 Initialization

Two initialization methods are very common in GP, the `Full` method and the `Grow` method [17]. The `Full` method always generates trees in which all leaves (end nodes) are at the same user-specified $MAXDEPTH$ value. This is achieved by randomly selecting nodes only from the $\mathcal{F}$ set until the depth limit is reached, at which nodes are selected from the $\mathcal{T}$ set. On the other hand, the `Grow` method creates trees of varied sizes and shapes by randomly selecting nodes from the full primitive set ($\mathcal{F}+\mathcal{T}$) for all nodes until the depth limit is reached, at which nodes are chosen from $\mathcal{T}$ as in the case of `Full` method. In Fig. 1, the tree on the left could have been the result of either the `Full` or the `Grow` method, but the one on the right can only be created by the latter. SmallGP uses a mix of `Full` and `Grow` methods. When selecting from the full primitive set the probability of choosing from the terminal set is,

$$p(\mathcal{T}) = \frac{|\mathcal{T}|}{|\mathcal{F}|+|\mathcal{T}|} \times SF,$$

where $|.|$ denotes the set size and SF is the scaling factor which scales initialization between `Full` (when $SF = 0$) and `Grow` (when $SF = 1$). It is to be noted that the ephemeral constants only contribute one virtual terminal symbol to the $\mathcal{T}$ set so that $|\mathcal{T}| = N+1$. The initialization also takes into account the maximum program (tree) length $MAXLEN$ which can also be specified by the user.

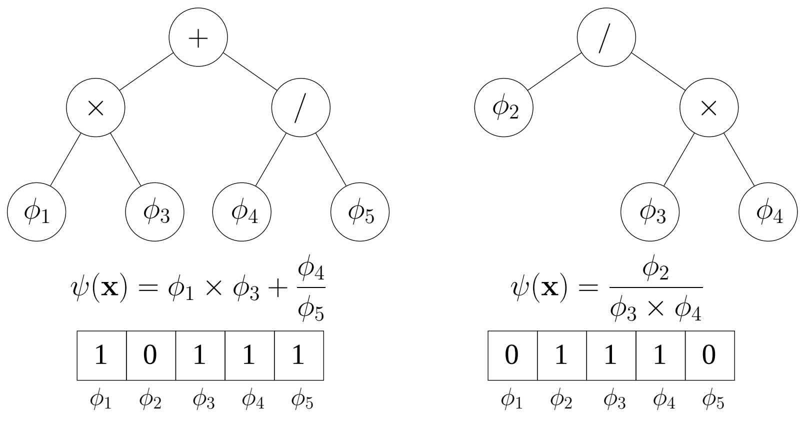
+
×
/
ϕ_1
ϕ_3
ϕ_4
ϕ_5
$\psi(\mathbf{x}) = \phi_1 \times \phi_3 + \frac{\phi_4}{\phi_5}$
1 0 1 1 1
ϕ_1 ϕ_2 ϕ_3 ϕ_4 ϕ_5
/
ϕ_2
×
ϕ_3
ϕ_4
$\psi(\mathbf{x}) = \frac{\phi_2}{\phi_3 \times \phi_4}$
0 1 1 1 0
ϕ_1 ϕ_2 ϕ_3 ϕ_4 ϕ_5

Fig. 1: Two examples of parse trees obtained from the `Full` (left) and `Grow` (right) methods for $MAXDEPTH = 2$. The corresponding $\psi(\mathbf{x})$ expressions and binary strings are also shown.

2.2 *Fitness and Constraint Evaluation*

This step involves the use of a grid-based clustering algorithm for evaluating the fitness function given in (2) for the trees created above. Each tree is decoded to obtain the design principle $\psi(\mathbf{x})$ that it represents. The resulting mathematical expression is evaluated for all m trade-off solutions provided as input to the GP algorithm to obtain the corresponding c-values. Grid-based clustering [2] involves sorting these c-values into a set $\mathbf{C}$ and dividing their range into d equal divisions (a parameter of the clustering routine). Elements in $\mathbf{C}$ which belong to divisions with less than $\lfloor m/d \rfloor$ c-values are categorized as *unclustered*. Adjacent divisions with more than $\lfloor m/d \rfloor$ c-values are merged to form clusters. Thus, the number of clusters $\mathcal{C}$ and the number of unclustered points $\mathcal{U}$ can be obtained and used in (2) to calculate the fitness for any given tree. Instead of asking the user to choose the parameter value for d, it is evolved alongside the GP trees using a GA. Therefore, each population member of the proposed system consists of a GP tree variable for $\psi(\mathbf{x})$ and an integer variable for d, which is also initialized (in the range $[1,m]$) in the previous step. This is the reason for integrating GA with SmallGP in this paper.

It has been suggested in [2] that for obtaining the most accurate design principles, the constraint

$$\mathcal{U} = 0, \tag{3}$$

should be imposed during clustering. This forces unclustered c-values to form one-element clusters by increasing the value of d, which in turn causes $\mathcal{C}$ and c_v within clusters to increase. The optimization of the weighted objective compensates for this by producing more accurate design principles.

A measure for the degree of commonality of $\psi(\mathbf{x})$ is also calculated in this step. This measure is called the significance S of the design principle and is defined as,

$$S = \frac{(m - \mathcal{U}')}{m} \times 100\%. \tag{4}$$

Here $\mathcal{U}'$ is the total number of elements in **C** which belong to divisions with less than $(\lfloor m/d \rfloor + \varepsilon)$ c-values. By choosing a small integer value for ε, c-values which barely formed clusters due to imposition of (3) can be identified.

2.3 Niched-Tournament Selection

In order to maintain multiple design principles in the population, [1] proposes the use of niched-tournament selection for creating the mating pool. We adopt a similar niching technique in the present paper. A binary string of length N is associated with each GP tree. If the l-th basis function is present in a tree then the l-th bit of the corresponding binary string is assigned a value of 1, else it takes a value of 0. Tournaments are only allowed between trees which have exactly the same binary string. This allows different *species* of design principles to co-exist in the population, while still promoting the better individual (fitness wise) when the trees have exactly the same basis functions. Fig. 1 shows such binary strings for the two trees. Since they are different, both individuals are equally competent irrespective of their fitness values and tournament selection is not performed between them.

2.4 Subtree Crossover and Discrete SBX

SmallGP uses a size-safe subtree crossover [17] to recombine trees in the mating pool. It ensures that the created children do not exceed the maximum tree length $MAXLEN$. This is accomplished by first determining the number of nodes by which the smaller parent tree can be extended. Then, a random subtree satisfying this requirement is cut from the larger parent tree. Similarly, a random subtree (whose maximum size is determined by taking the new size of the larger tree into account) is cut from the smaller tree. The two subtrees are exchanged at the cut locations to produce the offspring trees. Implementation details can be found in the manual [16].

The GA variable d is recombined using the discrete version of simulated binary crossover (SBX) [5].

2.5 Point Mutation and Discrete Polynomial Mutation

Point mutation [17] is the simplest form of mutation in GP where a random node is selected and the primitive at that node is replaced with a different random primitive of the same kind (function or terminal) and arity to maintain the closure property

[12]. Like the bit-flip mutation in GAs, point mutation is applied on a per-node basis, thus allowing multiple nodes to be mutated independently.

The GA variable d is mutated using the discrete version of polynomial mutation suggested in [5].

3 Dimensional Awareness

GP systems are known to produce exceptionally good models in symbolic regression applications, given that an appropriate set of primitives is provided. This has inspired the use of GP for scientific knowledge discovery from datasets obtained through physical processes, experiments, phenomena, etc. Computer-aided scientific knowledge discovery differs from standard symbolic regression in that the obtained model, in addition to fitting the data well, is also expected to be novel, interesting, plausible and understandable [24]. The key to achieving this is to incorporate the semantic content that is encapsulated in the data into the search process. Notable examples are the hypothesis-led discovery process of Robot Scientist Adam [11], the partial-derivative-pair metric based discoveries made by Eureqa [20], and other such discovery programs mentioned in [23].

The foremost application of automated innovization has been for engineering problems [2, 3]. It may be beneficial in these cases to extract design principles which are not just empirical in nature but also meaningful to the designer. In GP this is usually achieved by constraining the tree structures [17]. For example, if a model is known to be periodic *a priori*, then the search may be constrained to models that take the form $a \times \sin(b \times t)$ through strong typing [13] or grammar-based constraints [9]. When no such domain-specific information is available, one can still generate meaningful and syntactically correct tree structures by taking into account the most basic requirement for relationships governing all physical systems, namely *dimensional consistency* or *commensurability*, which states:

- Only commensurable quantities (quantities with the same dimensions) may be compared, equated, added, or subtracted.
- One may take ratios of incommensurable quantities (quantities with different dimensions), and multiply or divide them.

Previous work on dimensionally-aware genetic programming proposed a *weakly typed* or *implicit casting* approach [10] where dimensionality is not enforced, but promoted through an additional objective. We incorporate a similar strategy in the proposed (GA + SmallGP) system using constraints for penalising dimensional inconsistency.

Datasets obtained from multi-objective optimization of engineering systems consist of Pareto-optimal values of variables, constraints and objectives, for all of which the dimensions are known *a priori*. Each basis function $\phi(\mathbf{x})$ provided as input to the proposed system can therefore also carry its dimensionality information. For a given tree the dimensional consistency of $\psi(\mathbf{x})$ is checked using the following procedure.

Table 1: Transformed terminal operations for calculating the exponent E from the exponents e_i and e_j of two terminals. Note that Z can be set to any value greater than E_{max}.

Operation	Transformed Operation
$T_i + T_j$	$E = e_i$, if $e_i = e_j$ $E = Z$, otherwise
$T_i - T_j$	$E = e_i$, if $e_i = e_j$ $E = Z$, otherwise
$T_i \times T_j$	$E = Z$, if $\max(\lvert e_i\rvert, \lvert e_j\rvert) > E_{max}$ $E = e_i + e_j$, otherwise
$\frac{T_i}{T_j}$	$E = Z$, if $\max(\lvert e_i\rvert, \lvert e_j\rvert) > E_{max}$ $E = e_i - e_j$, otherwise
$T_i^{T_j}$	$E = e_i T_j$, if $\lvert e_i\rvert \leq E_{max}$ & $e_j = 0$ $E = Z$, otherwise

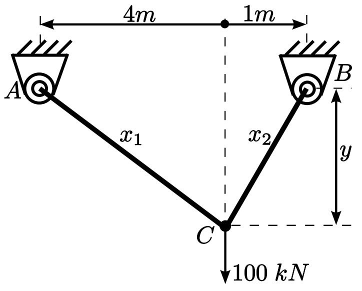

Fig. 2: Two-bar truss configuration showing the cross-sectional areas x_1 and x_2 and the vertical length y. A load of $F = 100$ kN is applied.

Noting that the largest possible absolute value of the exponent for any fundamental dimension (mass, length, time, etc.) among all known physical quantities is four, it can be established that in a tree of depth $MAXDEPTH$ the maximum absolute value that a dimension exponent can have in the corresponding $\psi(\mathbf{x})$ is $E_{max} = 4 \times 2^{MAXDEPTH}$. For two terminals T_i and T_j having dimension exponents e_i and e_j, undergoing various operations, the resulting exponent E is calculated using the transformed terminal operations shown in Table 1. Note that when adding or subtracting incommensurable quantities, E is assigned an arbitrary value Z greater than E_{max}. This indicates to subsequent operations that the tree being evaluated (or $\psi(\mathbf{x})$) is already dimensionally inconsistent. The absolute value of the exponent E obtained after completely evaluating the tree is constrained to be at or below E_{max} for all basic dimensions, thus imposing dimensional consistency.

The optimization problem for (GA + SmallGP) based automated innovization can now be formulated as,

$$\begin{aligned}
&\underset{\{\psi(\mathbf{x}),d\}}{\text{Minimize}}\ TS\left(\mathcal{C} + \sum_{k=1}^{\mathcal{C}} c_v^{(k)} \times 100\%\right) \quad \text{where} \quad c_v^{(k)} = \frac{\sigma_c}{\mu_c} \quad \forall\, c \in k\text{-th cluster},\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{and} \quad c = \psi(\mathbf{x}) \quad \forall\, m\\
&\text{Subject to} \left\{1 \leq d \leq m;\ \mathcal{U} = 0; S \geq S_{reqd}; |E| \leq E_{max}\ \ \forall \text{ basic dimensions}\right\}
\end{aligned} \tag{5}$$

The tree-size TS (number of nodes in the tree) is multiplied to the weighted objective function in (2) in order to promote smaller trees. This helps in obtaining simple relationships. The minimum significance required by the user is specified by S_{reqd}. We recommend starting with a high value of S_{reqd} (say 90%) and gradually reducing it until some design principles are obtained.

4 Results

We now illustrate the working of the proposed algorithm on three engineering design problems. In all cases the design optimization problem is solved using NSGA-II [6] with the following parameters:

1. Population size = 500 (truss and welded-beam), 1000 (metal-cutting)
2. Number of generations = 500,
3. SBX for real variables with $p_c = 0.9$ and $\eta_c = 10$,
4. Polynomial mutation for real variables with $p_m = 0.05$ and $\eta_m = 50$,

4.1 Two-bar Truss Design

The two-bar truss design problem involves three variables as shown in Fig. 2. The bi-objective formulation is,

$$\begin{array}{ll} \text{Minimize} & f_1(\mathbf{x}) = \text{ Volume } (V) = x_1\sqrt{16+y^2} + x_2\sqrt{1+y^2}, \\ \text{Minimize} & f_2(\mathbf{x}) = \text{ Max. Stress } (S) = \max(\sigma_{AC}, \sigma_{BC}), \\ \text{Subject to} & \left\{ S \leq 10^5 \text{ kPa}; 0 \leq x_1, x_2 \leq 0.01 \text{ m}^2 \text{ and } 1 \leq y \leq 3 \text{ m} \right\} \end{array} \tag{6}$$

NSGA-II gives $m = 500$ non-dominated solutions at the end 500 generations. For automated innovization we choose the function set $\mathcal{F} = \{+, -, \times, \%, ^\wedge\}$ where % represents protected division and $^\wedge$ represents the power function. The objectives and the variables are chosen as the basis functions, i.e, $\Phi = \{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \} = \{V, S, x_1, x_2, y\}$. and so the terminal set is $\mathcal{T} = \{V, S, x_1, x_2, y, \mathcal{R}\}$. The following parameters are used in the proposed (GA + SmallGP) algorithm to solve the optimization problem in (5):

1. Population size = 1000,
2. Number of generations = 100,
3. Discrete SBX for variable d with $p_c = 0.9$ and $\eta_c = 10$,
4. Discrete polynomial mutation for variable d with $p_m = 0.05$ and $\eta_m = 50$,
5. SmallGP crossover probability = 0.9, mutation probability (per node) = 0.2,
6. Maximum program depth ($MAXDEPTH$) = 10 and length ($MAXLEN$) =10,
7. Ephemeral constants, $\mathcal{R} = \{-10.0, -9.5, -9.0, \ldots, 9.0, 9.5, 10.0\}$,
8. Threshold significance $S_{reqd} = 80\%$, Clustering constant $\varepsilon = 3$.

Table 2 shows the obtained design principles, their significance values and the exponents of their basic dimensions. A total of 26 principles were obtained, which were symbolically simplified in MATLAB and only the unique ones are presented here.

The truss design problem can be mathematically solved using the identical resource allocation strategy in order to verify the obtained design principles. Increasing the cross-sectional area of one member reduces the stress induced in it and so the second objective takes the other member into account at some point. But since

Table 2: Design principles obtained using (GA + SmallGP) algorithm for truss design problem.

Notation	Design Principle (DP) $\psi(\mathbf{x}) = constant$	Significance	Basic Dimensions		
			Mass	Length	Time
DP1	$y = constant$	86.60%	0.0	1.0	0.0
DP2	$S \times V = constant$	87.00%	1.0	2.0	-2.0
DP3	$S \times x_1 = constant$	85.00%	1.0	1.0	-2.0
DP4	$S \times V \times y = constant$	87.00%	1.0	3.0	-2.0
DP5	$(V \times y)/x_2 = constant$	86.20%	0.0	2.0	0.0
DP6	$(V \times y)/x_1 = constant$	88.20%	0.0	2.0	0.0
DP7	$V/x_1 = constant$	86.40%	0.0	1.0	0.0
DP8	$V/(S \times x_1 \times x_2) = constant$	87.20%	-1.0	0.0	2.0
DP9	$V^2/(x_1 \times x_2) = constant$	87.40%	0.0	2.0	0.0
DP10	$y/(S \times x_1) = constant$	88.00%	-1.0	0.0	2.0
DP11	$x_2/x_1 = constant$	83.80%	0.0	0.0	0.0
DP12	$(S \times V \times x_2 \times y)/x_1 = constant$	88.00%	1.0	3.0	-2.0
DP13	$V/x_2 = constant$	86.80%	0.0	1.0	0.0
DP14	$(S \times V^2 \times y)/x_1 = constant$	87.20%	1.0	4.0	-2.0
DP15	$(x_2 \times y)/x_1 = constant$	86.40%	0.0	1.0	0.0
DP16	$x_2/(S \times x_1^2) = constant$	86.40%	-1.0	-1.0	2.0
DP17	$V^2/(x_1 \times x_2 \times y) = constant$	91.40%	0.0	1.0	0.0
DP18	$(S \times V^2)/x_2 = constant$	87.20%	1.0	3.0	-2.0
DP19	$S \times x_2 \times y = constant$	87.00%	1.0	2.0	-2.0
DP20	$(x_2 \times y)/(S \times x_1^2) = constant$	86.80%	-1.0	0.0	2.0

both the objectives are equally important, this cannot be allowed. A balance can be obtained only when the stresses in both the members are equal.

$$S = \sigma_{AC} = \sigma_{BC} \Rightarrow S = \frac{100}{5}\frac{\sqrt{16+y^2}}{yx_1} = \frac{4 \times 100}{5}\frac{\sqrt{1+y^2}}{yx_2}. \tag{7}$$

Following a similar argument for the volumes we get,

$$V = 2 \times x_1\sqrt{16+y^2} = 2 \times x_2\sqrt{1+y^2}. \tag{8}$$

Solving (7) and (8) gives the following relationships, all of which must be true for Pareto-optimality,

$$y = 2, \quad x_2 = 2x_1, \quad V = 4\sqrt{5}x_1 = 2\sqrt{5}x_2 \quad Sx_1 = 20\sqrt{5}, \quad Sx_2 = 40\sqrt{5}. \tag{9}$$

All design principles obtained by our approach conform to the above relationships. On the other hand, a dimensionally *unaware* GP produced relationships such as,

$$0.5 - (x_2 - S)^y = constant, S \times (V - x_2) = constant, (x_1 \times S) + x_2 = constant, \tag{10}$$

which, although numerically satisfy the requirements of a design principle, are of no practical value to the designer.

The next question to investigate is whether the 20 design principles can be reduced to the few shown in (9). To answer this, we first need to look at the *c*-value

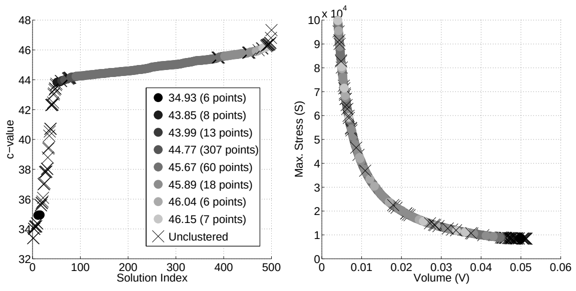

Fig. 3 Cluster plot (left) and the mapping of clusters in the objective space (right) for DP3. 425 out of 500 (85.00%) c-values obtained from $\psi(\mathbf{x}) = S \times x_1$ form eight clusters shown in shades of gray. The largest cluster has 307 points and an average c-value of 44.77.

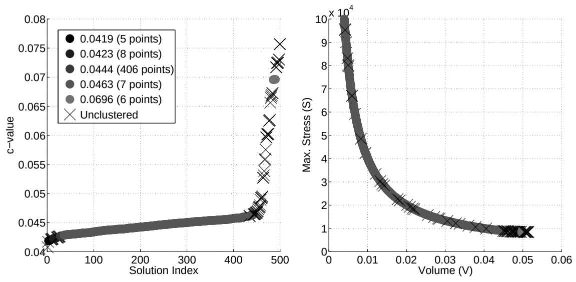

Fig. 4 Cluster plot (left) and the mapping of clusters in the objective space (right) for DP16. 432 out of 500 (86.40%) c-values obtained from $\psi(\mathbf{x}) = x_2/(S \times x_1^2)$ form five clusters shown in shades of gray. The largest cluster has 406 points and an average c-value of 0.0444.

cluster plots of all design principles. For illustration let us consider DP3 and DP16. In each case, $\psi(\mathbf{x})$ is evaluated for all $m = 500$ trade-off solutions. The resulting c-values are sorted and plotted as shown in Figs. 3 and 4.

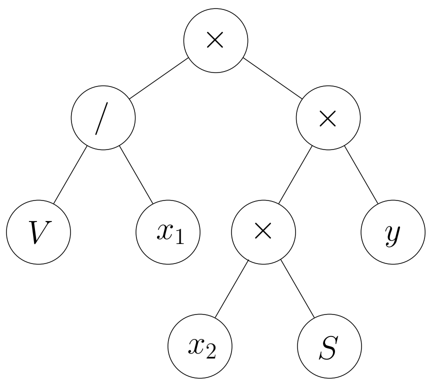

Fig. 5: GP tree for DP12 of truss design problem. Tree depth = 3 and size (or length) = 9.

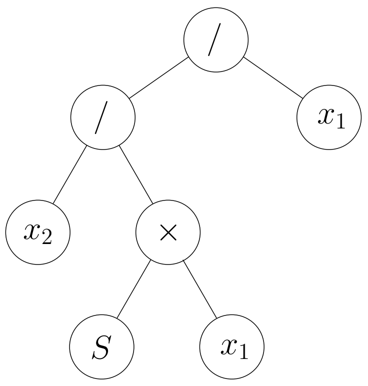

Fig. 6: GP tree for DP16 of truss design problem. Tree depth = 3 and size (or length) = 7.

The right side plot in each figure shows that both design principles are applicable on (approximately) the same part of the trade-off front, indicating that they can be combined. Indeed reducing DP16 with DP3 results in $x_2/x_1 = constant$ which in itself is another design principle (DP11). By considering the largest clusters of DP16 and DP3, the approximate value of the constant in DP11 is found to be $0.0444 \times 44.77 = 1.99 \approx 2$, which agrees with the second relationship in (9). In fact, all design principles in Table 2 form clusters in the same part of the trade-off front and hence they can be combined in any way to eliminate the redundant ones.

For illustration, the tree structures of DP12 and DP16 are shown in Figs. 5 and 6.

4.2 Welded Beam Design

As our next example, we consider the bi-objective welded-beam design problem. It involves the minimization of welding cost C and end deflection D of a welded cantilever beam carrying an end load of 6000 lb. The design variables (in inches) are: beam thickness b, beam width t, length of the weld l and weld thickness h. The allowable bending stress (σ), shear stress (τ) and buckling force (P_c) are limited by constraints. The problem formulation [7] is:

$$\begin{array}{ll} \text{Minimize} & f_1(\mathbf{x}) = C = 1.10471h^2l + 0.04811tb(14.0 + l), \\ \text{Minimize} & f_2(\mathbf{x}) = D = \frac{2.1952}{t^3b}, \\ \text{Subject to} & \left\{ \begin{array}{l} \tau(\mathbf{x}) \leq 13,600 \text{ psi}; \sigma(\mathbf{x}) \leq 30,000 \text{ psi}; b \geq h; P_c(\mathbf{x}) \geq 6,000 \text{ lb} \\ 0.125 \leq h, b \leq 5.0 \text{ in.}; 0.1 \leq l, t \leq 10.0 \text{ in.} \end{array} \right\} \end{array} \tag{11}$$

$$\begin{array}{ll} \text{where} & \tau(\mathbf{x}) = \sqrt{(\tau')^2 + (\tau'')^2 + (l\tau'\tau'')/\sqrt{0.25(l^2 + (h+t)^2)}}, \\ & \tau' = \frac{6,000}{\sqrt{2}hl}, \qquad \tau'' = \frac{6,000(14+0.5l)\sqrt{0.25(l^2+(h+t)^2)}}{2[0.707hl(l^2/12+0.25(h+t)^2)]}, \\ & \sigma(\mathbf{x}) = \frac{504,000}{t^2b}, \\ & P_c(\mathbf{x}) = 64,746.022(1 - 0.0282346t)tb^3. \end{array}$$

The problem is solved using NSGA-II to obtain $m = 500$ trade-off solutions. For automated innovization, we choose the same parameters as before, except the population size which is increased to 3000. The function set for GP remains the same. The terminal set is chosen as $\mathcal{T} = \{C, D, b, t, l, h, \sigma, \tau, P_c, \mathcal{R}\}$. An extra ‘Cost’ dimension is added to the basic dimensions’ set for providing the dimensional information of C to the algorithm. Table 3 shows the obtained design principles for $S_{reqd} = 90\%$.

All relationships are found to be applicable over approximately the same part of the trade-off front and therefore some of them are redundant. The independent design principles DP2, DP3 and DP5 have previously been shown to be true [7, 1].

DP1 requires special attention because it shows a limitation of the current approach. The deflection D is of the order of 10^{-4} whereas the cluster plot of DP2 in Fig. 7 reveals that the value of t is clustered around 10 in. for most solutions. This leads to an ambiguity where $D + t$ numerically satisfies the requirement of a design principle. The current approach does not handle these situations. A possible remedy for future analysis could be to normalize each basis function (ϕ) with its order of magnitude.

Table 3: Design principles obtained using (GA + SmallGP) algorithm for welded-beam problem.

Notation	Design Principle (DP) $\psi(\mathbf{x}) = constant$	Significance	Basic Dimensions			
			Mass	Length	Time	Cost
DP1	$(D+t) = constant$	95.20%	0.0	1.0	0.0	0.0
DP2	$t = constant$	95.60%	0.0	1.0	0.0	0.0
DP3	$D \times b = constant$	95.00%	0.0	2.0	0.0	0.0
DP4	$D \times b \times t = constant$	95.60%	0.0	3.0	0.0	0.0
DP5	$\sigma \times b = constant$	94.80%	1.0	0.0	-2.0	0.0
DP6	$\sigma \times b \times t = constant$	95.60%	1.0	1.0	-2.0	0.0
DP7	$D/\sigma = constant$	95.60%	-1.0	2.0	2.0	0.0
DP8	$D/(\sigma \times t) = constant$	95.60%	-1.0	1.0	2.0	0.0

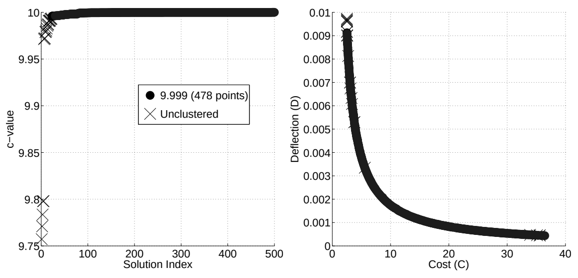

Fig. 7 Cluster plot (left) and the mapping of clusters in the objective space (right) for DP2. 478 out of 500 (95.60%) c-values obtained from $\psi(\mathbf{x}) = t$ form a single clusters with an average c-value of 9.999.

4.3 Metal-Cutting Process Optimization

Next, we consider the metal-cutting process optimization problem described in [19]. A steel bar is to be machined using a carbide tool of nose radius $r_n = 0.8$ mm on a lathe with $P^{max} = 10$ kW rated motor to remove 219912 mm^3 of material. A maximum cutting force of $F_c^{max} = 5000$ N is allowed. The motor has a transmission efficiency $\eta = 75\%$. The total operation time (T_p) and the used tool life (ξ) are to be minimized by optimizing the cutting speed (v), the feed rate (f) and the depth of cut (a) while maintaining a surface roughness of $R^{max} = 50\mu m$. The problem is formulated as,

$$\begin{array}{ll}
\text{Minimize} & f_1(\mathbf{x}) = T_p(\mathbf{x}) = 0.15 + 219912\left(\frac{1+\frac{0.20}{T(\mathbf{x})}}{MRR(\mathbf{x})}\right) + 0.05 \quad \text{min} \\
\text{Minimize} & f_2(\mathbf{x}) = \xi(\mathbf{x}) = \frac{219912}{MRR(\mathbf{x})T(\mathbf{x})} \times 100\% \\
\text{Subject to} & \left\{ \begin{array}{l} P(\mathbf{x}) \le \eta P^{max}; F_c(\mathbf{x}) \le F_c^{max}; R(\mathbf{x}) \le R^{max} \\ 250 \le v \le 400 \text{ m/min}; 0.15 \le f \le 0.55 \text{ mm/rev}; 0.5 \le a \le 6 \text{ mm} \end{array} \right\}
\end{array} \qquad (12)$$

$$\text{where} \quad T(\mathbf{x}) = \frac{5.48\times10^9}{v^{3.46} f^{0.696} a^{0.460}}, \qquad MRR(\mathbf{x}) = 1000vfa$$

$$P(\mathbf{x}) = \frac{vF_c(\mathbf{x})}{60000}, \qquad F_c(\mathbf{x}) = \frac{6.56\times10^3 f^{0.917} a^{1.10}}{v^{0.286}}, \qquad R(\mathbf{x}) = \frac{125 f^2}{r_n}.$$

NSGA-II results in 1000 trade-off solutions. (GA + SmallGP) is used with the same parameters as for truss design problem for $S_{reqd} = 70\%$ to obtain the design principles shown in Table 4. A new dimension ‘Life’ is introduced to denote used

Table 4: Design principles obtained using (GA + SmallGP) algorithm for metal-cutting problem.

Notation	Design Principle (DP) $\psi(\mathbf{x}) = constant$	Significance	Basic Dimensions			
			Mass	Length	Time	Life
DP1	$v/(f^2 \times \xi) = constant$	72.70%	0.0	-1.0	-1.0	-1.0
DP2	$(a \times v)/f = constant$	74.60%	0.0	1.0	-1.0	0.0
DP3	$v/(f^2 \times T_p \times \xi) = constant$	73.40%	0.0	-1.0	-2.0	-1.0
DP4	$f = constant$	72.90%	0.0	1.0	0.0	0.0
DP5	$a/(f \times T_p) = constant$	72.90%	0.0	0.0	-1.0	0.0
DP6	$(a^{5.5} \times f \times \xi)/T_p = constant$	77.50%	0.0	6.5	-1.0	1.0
DP7	$(a \times T_p \times v)/f = constant$	74.20%	0.0	1.0	0.0	0.0
DP8	$a^{5.5} \times T_p \times \xi = constant$	82.60%	0.0	5.5	1.0	1.0
DP9	$a \times T_p \times v = constant$	74.10%	0.0	2.0	0.0	0.0
DP10	$(a^2 \times T_p \times \xi)/v = constant$	74.40%	0.0	1.0	2.0	1.0
DP11	$(a^2 \times \xi)/v = constant$	76.00%	0.0	1.0	1.0	1.0
DP12	$a^{5.5} \times f \times \xi = constant$	76.80%	0.0	6.5	0.0	1.0

tool life which is expressed as a percentage of total tool life. Empirical (and more accurate) forms of DP4 and DP9 have previously been reported in [4]. Here, we sacrifice the accuracy of the design principle in favour of ease of interpretability for the designer, by making use of a dimensionally-aware GP. The problem with magnitudes of basis functions is again observed here when DP6 is reduced using DP4, giving rise to a relationship which does not agree with DP8.

The tree structures of two of the design principles, DP6 and DP11, are shown in Figs. 8 and 9. Their cluster plots in Figs. 10 and 11 show that even with such complex structures, they are valid design principles found using the proposed algorithm.

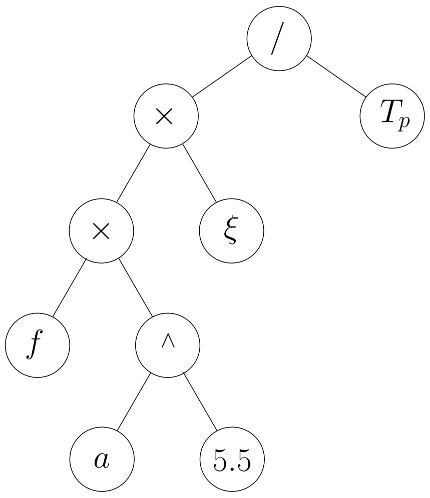

Fig. 8: GP tree for DP6 of metal-cutting problem. Tree depth = 4 and size (or length) = 9.

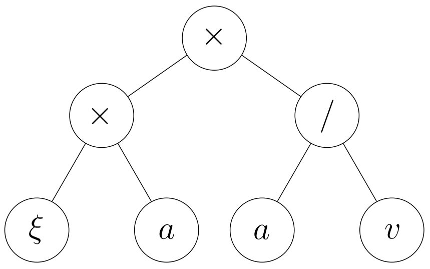

Fig. 9: GP tree for DP11 of metal-cutting problem. Tree depth = 2 and size (or length) = 7.

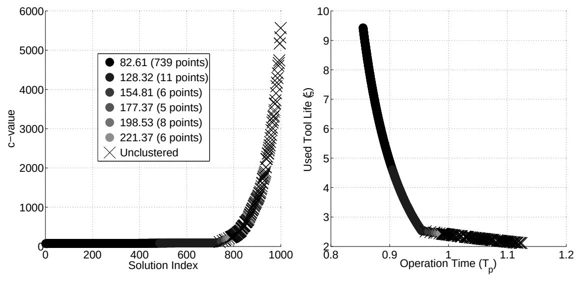

Fig. 10 Cluster plot (left) and the mapping of clusters in the objective space (right) for DP6. 775 out of 1000 (77.50%) *c*-values obtained from $\psi(\mathbf{x}) = (a^{5.5} \times f \times \xi)/T_p$ form six clusters shown in shades of gray. The largest cluster has 739 points and an average *c*-value of 82.61.

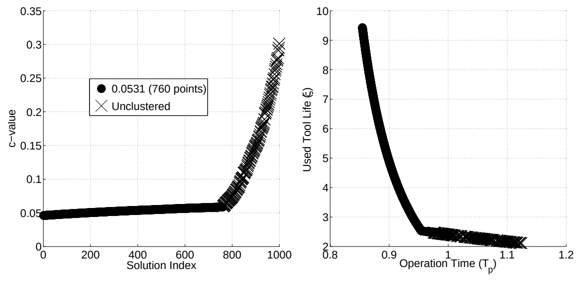

Fig. 11 Cluster plot (left) and the mapping of clusters in the objective space (right) for DP11. 760 out of 1000 (76.00%) *c*-values obtained from $\psi(\mathbf{x}) = (a^2 \times \xi)/v$ form a single cluster with average *c*-value of 0.0531.

5 Conclusions

This paper introduced a generalization to the automated innovization framework proposed previously by the authors. A tree-based representation is used to evolve generic design principles for extracting knowledge from multi-objective trade-off datasets. The underlying algorithm for automated innovization remains the same, but introduction of the parse tree representation required the use of a genetic programming system. We integrated the SmallGP system with a standard genetic algorithm for evolving the design principles. In order to obtain only physically meaningful design principles, we made the (GA + SmallGP) system dimensionally aware by penalising operations performed between incommensurable quantities. The dimensional consistency is checked at each step of tree evaluation for all fundamental dimensions. The proposed algorithm was tested on three engineering design problems. While syntactically correct principles were obtained in both cases, a limitation concerning different magnitudes of basis functions was identified.

References

1. Bandaru, S., Deb, K.: Automated innovization for simultaneous discovery of multiple rules in bi-objective problems. In: Proceedings of the 6th international conference on Evolutionary multi-criterion optimization, EMO'11, pp. 1–15. Springer-Verlag, Berlin, Heidelberg (2011)
2. Bandaru, S., Deb, K.: Towards automating the discovery of certain innovative design principles through a clustering-based optimization technique. Engineering Optimization **43**(9), 911–941 (2011)

3. Deb, K., Bandaru, S., Greiner, D., Gaspar-Cunha, A., Tutum, C.: An integrated approach to automated innovization for discovering useful design principles: Three engineering case studies. Indian Institute of Technology Kanpur, India, KanGAL Technical Report 2012001 (2012)
4. Deb, K., Datta, R.: Hybrid evolutionary multi-objective optimization of machining parameters. Tech. rep., Indian Institute of Technology Kanpur (2011). URL `http://www.iitk.ac.in/kangal/papers/k2011005.pdf`
5. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Computer Science and Informatics **26**, 30–45 (1996)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
7. Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: GECCO '06 - Proceedings of the 8th annual conference on genetic and evolutionary computation, pp. 1629–1636. New York: ACM (2006)
8. Fonseca, C., Fleming, P., et al.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of the fifth international conference on genetic algorithms, vol. 1, p. 416. Citeseer (1993)
9. Gruau, F.: On using syntactic constraints with genetic programming. In: Advances in genetic programming, pp. 377–394. MIT Press (1996)
10. Keijzer, M., Babovic, V.: Dimensionally aware genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1069–1076 (1999)
11. King, R., Rowland, J., Oliver, S., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L., et al.: The automation of science. Science **324**(5923), 85–89 (2009)
12. Koza, J.: Genetic Programming, On the Programming of Computers by Means of Natural Selection. A Bradford Book. MIT Press, Cambridge, MA, USA (1992)
13. Montana, D.: Strongly typed genetic programming. Evolutionary computation **3**(2), 199–230 (1995)
14. Newman, M.: Power laws, pareto distributions and zipf's law. Contemporary physics **46**(5), 323–351 (2005)
15. Obayashi, S., Sasaki, D.: Visualization and data mining of Pareto solutions using self-organizing map. In: Evolutionary Multi-Criterion Optimization, pp. 796–809. Springer (2003)
16. Planatscher, H.: SmallGP - A lean genetic programming system for symbolic regression (2004). Available from: `http://planatscher.net/smallgp.html`
17. Poli, R., Langdon, W., McPhee, N., Koza, J.: Genetic programming: An introductory tutorial and a survey of techniques and applications. University of Essex, UK, Tech. Rep. CES-475 (2007)
18. Pryke, A., Mostaghim, S., Nazemi, A.: Heatmap visualization of population based multi objective algorithms. In: Evolutionary Multi-Criterion Optimization, pp. 361–375. Springer (2007)
19. Quiza Sardiñas, R., Rivas Santana, M., Alfonso Brindis, E.: Genetic algorithm-based multi-objective optimization of cutting parameters in turning processes. Engineering Applications of Artificial Intelligence **19**(2), 127–133 (2006)
20. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. science **324**(5923), 81–85 (2009)
21. Sugimura, K., Obayashi, S., Jeong, S.: Multi-objective optimization and design rule mining for an aerodynamically efficient and stable centrifugal impeller with a vaned diffuser. Engineering Optimization **42**(3), 271–293 (2010)
22. Taboada, H., Coit, D.: Data mining techniques to facilitate the analysis of the Pareto-optimal set for multiple objective problems. In: Proceedings of the 2006 Industrial Engineering Research Conference (CD-ROM) (2006)
23. Valdés-Pérez, R.: Discovery tools for science apps. Communications of the ACM **42**(11), 37–41 (1999)
24. Valdés-Pérez, R.: Principles of humancomputer collaboration for knowledge discovery in science. Artificial Intelligence **107**(2), 335–346 (1999)
25. Walker, D., Everson, R., Fieldsend, J.: Visualisation and ordering of many-objective populations. In: 2010 IEEE Congress on Evolutionary Computation (IEEE-CEC), pp. 1–8 (2010)