Automated Innovization for Simultaneous Discovery of Multiple Rules in Engineering Problems

Sunith Bandaru and Kalyanmoy Deb
Indian Institute of Technology Kanpur, Kanpur, UP 208016, India
Email: {sunithb,deb}@iitk.ac.in
URL: http://www.iitk.ac.in/kangal

KanGAL Report Number 2010009

October 2, 2010

Abstract

The trade-off solutions of a multi-objective optimization problem, as a whole, often hold crucial information in the form of rules. These rules, if predominantly present in most trade-off solutions, can be considered as the characteristic features of optimal solutions. Knowledge of such features, in addition to providing better insights to the problem at hand, enables the designer to handcraft solutions for other optimization tasks which are structurally similar to it; thus eliminating the need to actually optimize. *Innovization* is the process of extracting these so called *design rules*. This paper proposes to move a step closer towards the complete *automation* of the innovization process using a niched clustering based optimization technique. The focus is on obtaining multiple design rules in a single knowledge discovery step using a niching strategy.

1 Introduction: A Motivating Example

The goal in multi-objective problem solving is always to find solutions which are as close to the true Pareto-optimal front as possible with as much diversity in functional space as possible. Numerous algorithms proposed over the years have been successful in achieving this with varying degrees. Considering the amount of time, resources and research effort that has gone into developing these algorithms, it is ironic that practically only a single (or utmost a few) optimal solution(s) actually get implemented in most problems. The accuracy and diversity attained with respect to other solutions can be put to good use if they can somehow be used to gain interesting knowledge about the problem.

Consider the bi-objective design problem of a two-bar truss. The configuration is shown in Figure 1. The problem requires that the total volume V of the truss structure be minimized along with the minimization of the maximum stress S induced in either of the bars. Geometrical constraints restrict the cross-sectional areas x_1 and x_2 of the bars the dimension y. The induced stress should remain below the elastic strength S_y of the material used, which gives rise to a third constraint. The optimization formulation thus becomes,

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

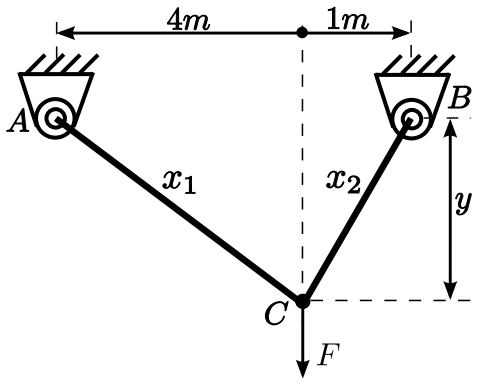

Figure 1: Two bar truss configuration.

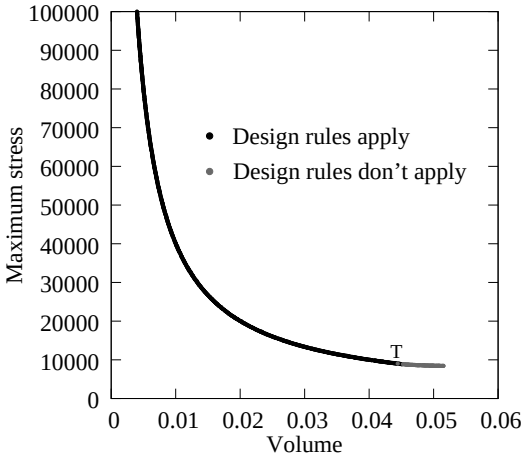

Figure 2: Pareto front of the truss problem.

It is possible to analytically derive solutions for some special multi-objective problems. Methods usually involve the use of Fritz-John conditions or Karush-Kuhn-Tucker optimality criteria and are generally used for convex multi-objective problems. For solving (1) however, the identical resource allocation strategy can be used. Increasing the cross-sectional area of one bar element reduces the stress induced in it and so the second objective takes the other bar element into account at some point. But since both the objectives are equally important, this cannot be allowed. A balance can be obtained only when the stresses in both the bars are equal. Thus,

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

Also, since sum of two positive quantities is minimum only when they are equal,

$$x_1\sqrt{16+y^2} = x_2\sqrt{1+y^2}. \tag{3}$$

Solving (2) and (3) gives the following relationships, where the '$*$' emphasizes the fact that they represent optimality.

$$y^* = 2, \quad x_2^* = 2x_1^*, \quad V^* = 4\sqrt{5}x_1^* = 2\sqrt{5}x_2^*. \tag{4}$$

Note that these relationships will hold irrespective of the material and loading. They are thus the generic "design rules" of the truss problem in (1). Designers can remember them as guidelines when optimizing any such structure and easily handcraft an optimal solution. Of course the solution has to be checked for feasibility before actual implementation because the derivation of (4) did not involve the use of constraints. The gray points in Figure 2 indicate the solutions to which these rules do not apply.

2 Related Work

The simple example discussed above shows that interesting knowledge in the form of design rules exist in multi-objective scenarios and that they are the distinguishing features of the Pareto-optimal solutions of a problem. Analytical solutions to optimization problems, especially multi-objective problems, are however rarely easy to obtain. Some alternatives exist to avoid actually solving the problems to decipher design rules. For example, monotonicity analysis [1] is capable of obtaining them directly from the problem [2] provided that the latter satisfies certain conditions of monotonicity.

In general however, there seem to be three major obstacles in deriving design rules such as those in (4): (i) most problems are not analytically solvable, (ii) methods require the problems to

have a certain form or satisfy certain criteria, and (iii) other methods can only produce implicit or semantic rules. The first two of these obstacles can be overcome by employing data-mining and machine learning techniques on the solutions obtained by solving the problem through a numerical optimization algorithm. Obayashi and Sasaki [3] used self-organizing maps (SOMs) to generate clusters of design variables that indicate their role in making design improvements. A multi-objective design exploration (MODE) methodology was proposed [4] to reveal the structure of optimal design space. The study concluded that such design knowledge can be used to produce better designs. Hierarchical grouping of solutions in the form of a dendogram to identify strongly related variables is described in [5]. Recently, a data-mining technique called proper orthogonal decomposition has been used [6] to extract implicit design knowledge from the Pareto-optimal solutions.

While the above mentioned studies truly depict the importance of analyzing trade-off solutions, they fall short of providing a generic framework which can also overcome the third obstacle in discovering design rules; the ability to extract rules that are meaningful to the human designer. In other words, rules that have an explicit mathematical form are more useful and intuitive to a human for remembrance and future application to similar design problems. *Innovization* [7] addresses this, though at a simple level, through a manual graph plotting and regression analysis procedure. In the remaining sections we describe and extend the innovization procedure for complete automated discovery of multiple design rules from the Pareto-optimal solutions.

3 Discovering Design Rules through Innovization

The term innovization comes from *innov*ation through optim*ization*. As described above, it is a manual plotting-and-analysis process which can help reveal design rules hidden in Pareto-optimal datasets. Deb and Srinivasan [8] define it as "a new design methodology in the context of finding new and innovative design principles by means of optimization techniques". The basic procedure is simple and can be accomplished through the following steps:

1. Obtain the Pareto-optimal front for the multi-objective problem using any of the available population-based evolutionary algorithms [9].

2. Form the data-set containing the optimal variable values ($\mathbf{x}^*$), objective function values ($\mathbf{f}^*$) and corresponding values of any other function(s) $\phi_j(\mathbf{x}^*)$ (see below) for all the obtained trade-off solutions.

3. Consider various entity combinations: variable-variable, objective-objective, objective-variable, etc. and plot the corresponding values for all the trade-off solutions. This will visually reveal the existence of design rules when any of these combinations show a high correlation between the entities considered. Other functions (ϕ_j's) which the designer feels are significant to the design task may also be considered.

4. In case a non-linear correlation is observed, techniques like logarithmic transformation and curve-fitting are employed to come up with the closest rule.

While steps (1) and (2) can be easily integrated into a computer code, it is the human judgement required in step (3) that makes innovization, in this state, a tedious and time consuming approach to rule-finding. Nevertheless the process has been applied as such to various multi-objective problems [7, 8, 10] and even researchers in fields as diverse as architecture [11], virtual reality [12], robotics [13], etc. are realizing the need to identify the commonalities between solutions in the form of usable rules.

The truss design problem introduced previously can be used to illustrate the innovization approach. The required trade-off data is generated by solving (1) using NSGA-II [14]. A population

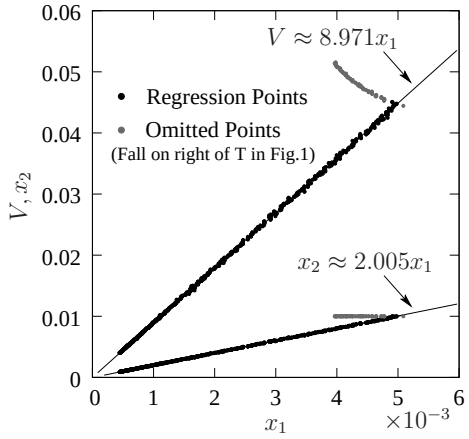

Figure 3: Innovization for V-x_1 and x_2-x_1.

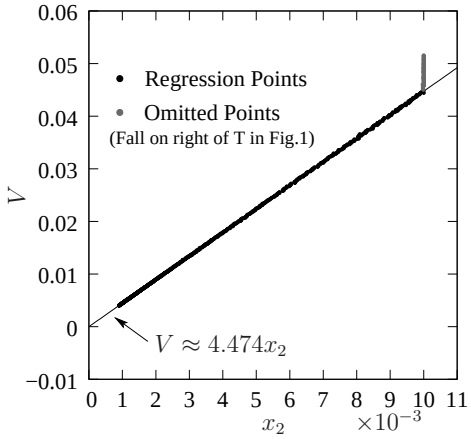

Figure 4: Innovization for V-x_2.

size of 1000 is used for the purpose so as to generate a well-represented trade-off front. F is taken to be 100 kN and S_y as 10^5 kPa. Figures 3 and 4 clearly show that V-x_1, x_2-x_1 and V-x_2 have a high correlation. The slopes of regression lines fitted to each of these combinations (by ignoring points which apparently show a change in relationship) gives the constant of proportionality. Note that,

$$V \approx 8.971x_1 \approx 4\sqrt{5}x_1, \quad x_2 \approx 2.005x_1 \approx 2x_1 \text{ and } \quad V = 4.474x_2 \approx 2\sqrt{5}x_2. \tag{5}$$

As it can be seen, a post-optimality procedure like innovization is capable of revealing design rules from the trade-off solutions. However, there are some obvious difficulties which prompt an automation of this approach. Firstly, as discussed above, manually choosing different combinations and checking for correlations is a time consuming process. Secondly, all solutions need not follow a particular design rule. Unlike in Figure 3 and 4, if the change in relationship is subtle, a blind regression analysis may lead to erroneous conclusions. Each design rule can thus be associated with a prominence level or *significance* depending on the percentage of the Pareto front that it applies to. It is crucial that rules with a low significance be filtered out by the automated algorithm. Lastly, there can be design rules which have different proportionality constants in different regions of the Pareto-optimal front [15, 16]. The design rule is then said to be *parametrically* varying across these regions. The automated algorithm should be able to tell apart solutions which parametrically satisfy the rule from the ones that do not satisfy it at all.

4 Proposed Automated Innovization Approach

Let $\phi_j(\mathbf{x})$ be the set of "basis functions" whose combinations are to be analyzed for the presence of design rules. The designer can specify N such functions. They also include the variables, objectives and constraints of the problem. Previous manual innovization studies [7, 8, 10] and recent proof-of-principle studies towards a possible automated innovization [15, 16] sufficiently show that most design rules take the generic form,

$$\Pi_{j=1}^{N} \phi_j(\mathbf{x})^{a_{ij} B_{ij}} = C_i \tag{6}$$

where C_i is the proportionality constant for the i-th design rule and B_{ij}'s are corresponding powers of the basis functions in that design rule. The Boolean variables a_{ij}'s, in a way, reflect the presence ($a_{ij} = 1$) or absence ($a_{ij} = 0$) of the j-th basis function in the i-th rule. In addition to being relatively easier for an automated algorithm to detect, the mathematical form of these relationships makes them more intuitive to the user. Note that B_{ij}'s in (6) can take any real value depending on the problem. In order to restrict the search space, the maximum absolute power

(among participating basis functions) is set to one by making the following transformation which keeps the design rule unaltered:

$$\begin{aligned}\left(\Pi_{j=1}^{N}\phi_j(\mathbf{x})^{a_{ij}B_{ij}}\right)^{\frac{1}{\{B_{ip}|p:(\max_p|a_{ip}B_{ip}|)\}}} &= C_i^{\frac{1}{\{B_{ip}|p:(\max_p|a_{ip}B_{ip}|)\}}} \\ &= c_i \quad \text{(say)}, \end{aligned} \tag{7}$$

which can be simply written as,

$$\Pi_{j=1}^{N}\phi_j(\mathbf{x})^{a_{ij}b_{ij}} = c_i, \quad \text{where } b_{ij} = \frac{B_{ij}}{\{B_{ip}|p:(\max_p|a_{ip}B_{ip}|)\}} \in [-1,1]. \tag{8}$$

Supposing that a_{ij}'s and b_{ij}'s are known for the i-th rule, the parameter c_i can easily be calculated for all trade-off solutions in the Pareto-optimal data-set. Solutions that parametrically satisfy the i-th rule will have the same (or nearly equal) values for c_i. The present algorithm uses a grid-based clustering technique to identify clusters of such solutions. If a solution yields a c_i value which is significantly different from that of others, then it remains unclustered.

4.1 Finding Optimal a_{ij}'s and b_{ij}'s

Each cluster mentioned above corresponds to one region of the Pareto front to which the i-th design rule applies parametrically. The c_i-values in each of these clusters should therefore be nearly equal. In other words, the spread of c_i-values in each cluster should be minimum. This condition can be used to obtain a_{ij}'s and b_{ij}'s. To ensure that a *narrow* distribution of c_i-values is obtained in the clusters, the coefficient of variation (c_v=variance/mean) of the c_i-values is simultaneously minimized in all of them. The weighted sum approach is used to have a computationally tractable approach. Since all clusters are equally important for the design rule to be valid, the c_v's are assigned equal weights. Note that c_v being a *normalized measure of variance* will have the same order of magnitude in all the clusters and therefore the weighted sum approach should suffice [16]. Thus, to find optimal a_{ij}'s and b_{ij}'s for the i-th rule the following optimization problem can be solved,

$$\begin{array}{ll} \text{Minimize} & \displaystyle\sum_{clusters} c_v^{(k)}, \qquad c_v^{(k)} = \frac{\sigma_{c_i}}{\mu_{c_i}} \quad \forall\, c_i \in k\text{-th cluster} \\ \text{Subject to} & -1.0 \le b_{ij} \le 1.0 \quad \forall\, j : a_{ij} = 1, \\ & |b_{ij}| \ge 0.1 \quad \forall\, j : a_{ij} = 1, \\ & \sum_j a_{ij} \ge 1, \\ & a_{ij}\text{'s are Boolean and } b_{ij}\text{'s are real.} \end{array} \tag{9}$$

Any basis function ϕ_j with a low magnitude power b_{ij} will hardly contribute to the design rule. Moreover, inclusion of zeroes in the search space will lead to a trivial solution where $c_i = 1$ for all trade-off solutions and so $c_v = 0$ for all clusters. The second set of constraints checks this by putting a lower bound on the magnitude of b_{ij} (see Section 5.3). The last constraint ensures that at least one basis function is used to form the design rule.

4.2 One-dimensional Grid-based Clustering

Grid-based clustering technique involves partitioning the data into a number of grids (or divisions) and merging them to form clusters. The number of clusters need not be predefined. The following steps describe the procedure:

Step 1: Sort all $\{c_i^{(1)}, c_i^{(2)}, \ldots, c_i^{(m)}\}$ obtained by evaluating (8) for all m trade-off solutions in the data-set.

Step 2: Divide the space $[c_{i,min}, c_{i,max}]$ into, say d_i divisions.

Step 3: Count the number of points in each division.

Step 4: Merge adjacent divisions which have more than or same as $\lfloor \frac{m}{d_i} \rfloor$ points (the average number of points per division) to form clusters.

Step 5: Count the number of clusters $\mathcal{C}_i$ and the total number of unclustered points $\mathcal{U}_i$ in all divisions with less than $\lfloor \frac{m}{d_i} \rfloor$ points.

There are two conflicting arguments for deciding the number of divisions d_i. It is desirable that the design rule be applicable to as many points as possible, so ideally $\mathcal{U}_i = 0$. This translates to a high value of d_i since each unclustered point can then form a one-element cluster. But this, in turn will increase the number of clusters. A good clustering algorithm should be able to find the simplest representation of the data (which points to a low d_i) while ensuring that points within a cluster are similar in some respect. In the present case, c_v's within the clusters define this similarity. Thus, finding the optimum number of divisions can be framed as an optimization problem,

$$\begin{array}{ll} \text{Minimize} & \mathcal{C}_i + \sum_{k=1}^{\mathcal{C}_i} c_v^{(k)} \times 100\%, \qquad c_v^{(k)} = \dfrac{\sigma_{c_i}}{\mu_{c_i}} \quad \forall\, c_i \in k\text{-th cluster}, \\ \text{Subject to} & \mathcal{U}_i = 0, \\ & 1 \leq d_i \leq m, \\ & d_i \text{ is an integer.} \end{array} \tag{10}$$

The limits on d_i are due to the clustering criterion of $\lfloor \frac{m}{d_i} \rfloor$ points. It is easy to see that any value of $d_i > m$ would yield the same result as $d_i = m$. The percentage coefficient of variation is used to approximately scale the c_v-values to the order of number of clusters ($\mathcal{C}_i$) which in turn allows the use of the weighted sum [16].

The clustering algorithm discussed above uses the a_{ij} and b_{ij} values obtained from the methodology described in Section 4.1 to calculate d_i. The latter in turn uses the clusters identified by the former to calculate the c_v's. The two optimization problems (9) and (10) therefore have to be solved simultaneously. The algorithm calculations involved in both of these and the non-availability of mathematical functions prevents the use of a classical optimization approach. The present paper uses a simple genetic algorithm (GA) instead. It has the added advantage that a_{ij}'s can simply be coded as the bits of a binary string whereas b_{ij}'s can be regarded as real variables. Each population member then acts as a design rule and only the best among these survive.

4.3 Significance of Design Rules

As discussed towards the end of Section 3, the discovered design rules should be significant for them to be useful for a designer. A direct measure of significance is the percentage of trade-off data-set that the rule applies to. To calculate the significance of each GA population member, the fourth step of the clustering algorithm in Section 4.2 is modified as,

Modified Step 4: Merge adjacent divisions which have more than or same as $\lfloor \frac{m}{d_i} \rfloor + \epsilon$ points to form clusters.

With a small integer value for ϵ, divisions which barely form part of the clusters can be identified. Let $\mathcal{C}_{i,MS}$ and $\mathcal{U}_{i,MS}$ respectively be the number of clusters and unclustered points calculated with the modified step. Note that the clustering itself need not be redone for this purpose. The significance S_i of the i-th design rule can now be given as,

$$S_i = \frac{m - \mathcal{U}_{i,MS}}{m} \times 100\%. \tag{11}$$

By placing a lower bound on S_i (say $S_{reqd} = 80\%$), designers can choose the minimum level of prominence for the design rules.

4.4 Niching for Multiple Design Rules

The discussion so far has been carried out with respect to the i-th design rule. However, solving a simple superposition of problems (9) and (10) with an additional constraint on significance (11) will only yield a single design rule because only the best population member will survive through the generations. The co-existence of multiple design rules can be promoted by introducing a *niched-tournament* selection operator [17] in the GA which allows a tournament to be played only between population members which use the same set of basis functions. The niching is implemented on top of the penalty-parameter-less approach to constraint handling [18]. The following criteria are used to determine the winner among two solutions u and v participating in a tournament:

1. If $a_{uj} = a_{vj} \quad \forall\, j = 1, 2, \ldots, N$, then u and v can be compared.
   (a) If one is feasible and the other is not then the feasible solution is preferred.
   (b) If both are feasible then the one with better objective value is preferred.
   (c) If both are infeasible then the one with lower constraint violation is preferred.
2. Else both u and v are competent.

With a GA that uses (i) this new niched-tournament selection operator for handling the constraints, (ii) one-point crossover and bit-wise mutation for the binary string of a_{ij} bits, (iii) simulated binary crossover (SBX) and polynomial mutation for b_{ij}'s and, (iv) a discrete version of SBX and polynomial mutation for the variable d_i, the combined optimization problem to be solved for extracting multiple design rules simultaneously is proposed as,

$$\begin{array}{ll} \text{Minimize} & \mathcal{C}_i + \sum_{k=1}^{\mathcal{C}_i} c_v^{(k)} \times 100\%, \qquad c_v^{(k)} = \dfrac{\sigma_{c_i}}{\mu_{c_i}} \quad \forall\, c_i \in k\text{-th cluster} \\ \text{Subject to} & -1.0 \le b_{ij} \le 1.0 \quad \forall\, j : a_{ij} = 1, \\ & |b_{ij}| \ge 0.1 \quad \forall\, j : a_{ij} = 1, \\ & \sum_j a_{ij} \ge 1, \\ & \mathcal{U}_i = 0, \quad 1 \le d_i \le m, \quad S_i \ge S_{reqd}, \\ & a_{ij}\text{'s are Boolean, } b_{ij}\text{'s are real and } d_i \text{ is an integer.} \end{array} \tag{12}$$

5 Results

Algorithm 1 summarizes the proposed automated innovization approach for discovering multiple design rules in a single knowledge discovery step. It is now applied in this form to two well-studied engineering design problems.

5.1 Truss Design Revisited

The trade-off solutions obtained in Section 3 for $m = 1,000$ points using NSGA-II are utilized here. All objectives and variables are chosen as the basis functions. Hence,

$$\phi_1 = V, \quad \phi_2 = S, \quad \phi_3 = x_1, \quad \phi_4 = x_2, \quad \phi_5 = y.$$

The following parameters are used for solving (12):

Population Size ($popsize$) = 400

Algorithm 1 Automated innovization for simultaneous multiple rule discovery.

1: Obtain a good set of m diverse and near-Pareto-optimal solutions of the multi-objective problem.
2: Choose the terminal set of N basis functions ϕ_j's.
3: Form the $m \times N$ data-set of ϕ_j's evaluated at all m trade-off solutions.
4: Initialize variables a_{ij}, b_{ij} and $d_i \; \forall \; i = popsize$ GA members.
5: $gen \leftarrow 1$
6: **while** $gen \leq maxgen$ **do**
7: **for** $i := 1$ to $popsize$ **do**
8: Transform $b_{ij} \leftarrow \dfrac{b_{ij}}{\{b_{ip} | p : (\max\limits_{p} |a_{ip} b_{ip}|)\}}$ to maintain $\max |a_{ij} b_{ij}| = 1$
9: Evaluate $c_i = \Pi_{j=1}^{N} \phi_j(\mathbf{x})^{a_{ij} b_{ij}} \; \forall \; m$ trade-off solutions.
10: Sort and cluster $\{c_i^{(1)}, c_i^{(2)}, \ldots, c_i^{(m)}\}$ using the grid-based clustering.
11: Evaluate the objective and constraints in (12).
12: **end for**
13: Perform niched-tournament selection on the GA population members.
14: Perform appropriate crossover operations.
15: Perform appropriate mutation operations.
16: Update GA population members.
17: $gen \leftarrow gen + 1$
18: **end while**
19: Report unique members of final GA population as the obtained design rules (**D**).

Maximum number of generation ($maxgen$) = 500

One-point crossover probability ($p_{c,binary}$) = 0.85

Bit-wise mutation probability ($p_{m,binary}$) = 0.15

Continuous and discrete SBX probability ($p_{c,real}$) = 0.95

Continuous and discrete polynomial mutation probability ($p_{m,real}$) = 0.05

Continuous and discrete SBX distribution index (η_c) = 10

Continuous and discrete polynomial mutation distribution index (η_m) = 50

Parameter for calculating $\mathcal{U}_{i,MS}$ (ϵ) = 3

Threshold significance for design rules (S_{reqd}) = 80%

Table 1 shows the *unique* solutions present in the GA population after 500 generations of the proposed algorithm. Notice how the use of niched-tournament selection and the constraint on significance together helped maintain diverse (in terms of basis functions) and yet significant design rules. For example, the $i = 20$-th design rule is as follows:

$$V^{1.0000000} S^{0.7911332} x_1^{-0.2102126} = c_{20}. \tag{13}$$

The essence of this design rule is not in the parametric constant c_{20} but in the fact that the left-hand side of this expression remains almost constant for at least $S_{reqd} = 80\%$ of the NSGA-II obtained data-set.

There is, however, a downside to discovering and presenting design rules in the form as in Table 1. A human designer would prefer having all the design rules in a compact form which can be intuitive to him/her. It is not difficult to see that there are some *redundant* rules in Table 1. The next logical step is therefore to condense and present them in a more compact form.

Table 1: Design rules ($\mathbf{D}$) obtained for the truss design problem.

i	$a^*_{ij}\ \forall\ j$	$a^*_{i1}b^*_{i1}$	$a^*_{i2}b^*_{i2}$	$a^*_{i3}b^*_{i3}$	$a^*_{i4}b^*_{i4}$	$a^*_{i5}b^*_{i5}$	d^*_i
1	10010	-0.9979552	0.0000000	0.0000000	1.0000000	0.0000000	869
2	01111	0.0000000	-0.9668656	-0.7030048	-0.2639445	1.0000000	869
3	01011	0.0000000	-0.7727935	0.0000000	-0.7749417	1.0000000	869
4	01101	0.0000000	-0.7048297	-0.7049160	0.0000000	1.0000000	870
5	00110	0.0000000	0.0000000	-0.9990348	1.0000000	0.0000000	869
6	00111	0.0000000	0.0000000	-0.9952743	0.9999844	1.0000000	869
7	00001	0.0000000	0.0000000	0.0000000	0.0000000	1.0000000	869
8	11111	0.7817370	-0.7754779	-0.7850468	-0.7734357	1.0000000	869
9	10011	0.8268434	0.0000000	0.0000000	-0.8259557	1.0000000	869
10	10101	0.8388214	0.0000000	-0.8391667	0.0000000	1.0000000	869
11	10111	0.8813441	0.0000000	-0.5827007	-0.3013669	1.0000000	869
12	11101	0.9486451	0.1086554	-0.8411865	0.0000000	1.0000000	856
13	11011	0.9962421	0.6705470	0.0000000	-0.3253438	1.0000000	908
14	11001	0.9989458	0.9984587	0.0000000	0.0000000	1.0000000	869
15	11000	0.9999623	1.0000000	0.0000000	0.0000000	0.0000000	869
16	11110	1.0000000	-0.5590224	-0.7850468	-0.7741561	0.0000000	869
17	10100	1.0000000	0.0000000	-0.9971116	0.0000000	0.0000000	869
18	10110	1.0000000	0.0000000	-0.7353538	-0.2647701	0.0000000	869
19	11010	1.0000000	0.6702390	0.0000000	-0.3300355	0.0000000	869
20	11100	1.0000000	0.7911332	-0.2102126	0.0000000	0.0000000	860

5.2 Reduced Row-Echelon Form

In linear algebra, reduced row-echelon forms (RREF) are used to identify linearly dependent rows of a matrix. By eliminating redundant variables using RREF, a system of linear equations can be solved easily. Here, the same technique is used to condense the design rules. However, since the original trade-off data-set is only near-Pareto-optimal, the obtained design rules can only be approximate. Therefore, a tolerance *tol* should be allowed during row operations. Algorithm 2 is used for this purpose.

Algorithm 2 Determination of *tol* and condensed design rules.

1: $l \leftarrow$ maximum number of significant digits in any element of $\mathbf{D}$.
2: **repeat**
3: $tol \leftarrow 10^{-l}$
4: $\mathbf{D}_{reduced} = \text{rref}(\mathbf{D}, tol)$ {The MATLAB® function rref() is used here.}
5: $l \leftarrow l - 1$
6: **until** $\text{rank}\,\mathbf{D}_{reduced} < N$
7: Identify insignificant relationships in $\mathbf{D}_{reduced}$ by performing grid-based clustering.
8: Report other relationships as the condensed design rules.

Table 2 shows the result of applying Algorithm 2 on the values in Table 1. It can be seen that

Table 2: Reduced design rules ($\mathbf{D}_{reduced}$) for the truss design problem, $tol = 0.01$.

i	$a^*_{i1}b^*_{i1}$	$a^*_{i2}b^*_{i2}$	$a^*_{i3}b^*_{i3}$	$a^*_{i4}b^*_{i4}$	$a^*_{i5}b^*_{i5}$	d^*_i	S_i
DR1	1.0000000	0.0000000	0.0000000	-1.0006158	0.0000000	520	88.2%
DR2	0.0000000	1.0000000	0.0000000	1.0005781	0.0000000	508	80.8%
DR3	0.0000000	0.0000000	1.0000000	-1.0009661	0.0000000	507	86.8%
DR4	0.0000000	0.0000000	0.0000000	0.0000000	1.0000000	511	87.2%

all the original rules in (4) are realizable from these condensed rules whose approximate forms

are summarized below:

$$\text{DR1: } \frac{V}{x_1} = c_{\text{DR1}}, \quad \text{DR2: } Sx_2 = c_{\text{DR2}}, \quad \text{DR3: } \frac{x_1}{x_2} = c_{\text{DR3}}, \quad \text{DR4: } y = c_{\text{DR4}}.$$

Further insight can be obtained by again performing grid-based clustering on the c_i-values of these four rules to determine the number of clusters ($\mathcal{C}_{i,MS}$) and unclustered points ($\mathcal{U}_{i,MS}$). The corresponding d_i^*'s and significance S_i are also shown in the table. Figures 5, 6, 7 and 8 show the distribution of c_i's, the horizontal broken lines being their cluster averages. Unclustered points are shown in gray.

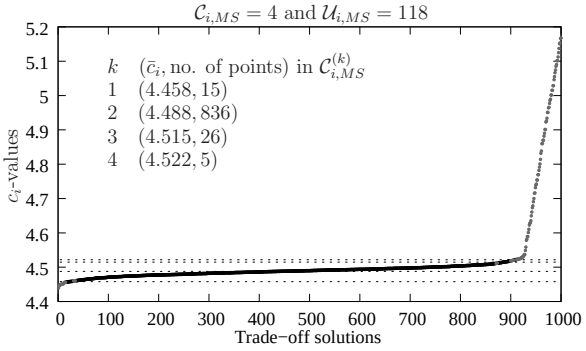

Figure 5: Design rule i = DR1 ($S_i = 88.2\%$) is equivalent to $V = 2\sqrt{5}x_2 = 4.472x_2$.

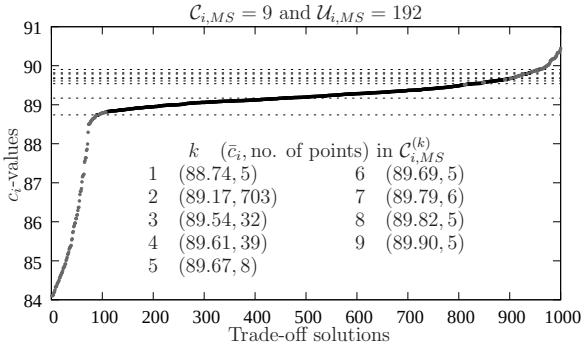

Figure 6: Design rule i = DR2 ($S_i = 80.8\%$) is equivalent to $Sx_2 = 200/\sqrt{5} = 89.44$.

The figures and the corresponding relationships show how the proposed automated algorithm is capable of successfully deciphering important rules for a design problem directly from the trade-off data-set. Specifically, for this problem it can be seen that indeed the obtained rules and the associated c_i's are approximately similar to those derived theoretically (and manually) earlier (equation 4). In a complex design scenario, such rules generated automatically will be extremely useful for the designer.

5.3 Welded Beam Design

This problem involves the minimization of cost (C) and end deflection (D) of a welded cantilever beam carrying a given maximum load. The design variables are the thickness of the beam b, width of the beam t, length of the weld l and weld thickness h. Constraints are used to limit the allowable bending stress (σ), shear stress (τ) and buckling force (P_c). The multi-objective formulation can be found in [18]. The trade-off front is obtained for $m = 300$ population size

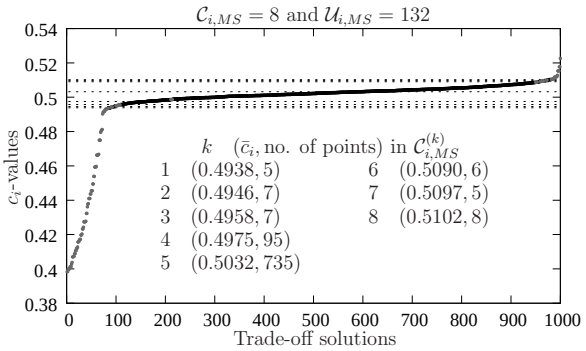

Figure 7: Design rule i = DR3 ($S_i = 86.8\%$) is equivalent to $x_2 = 2x_1$ or $x_1 = 0.5x_2$.

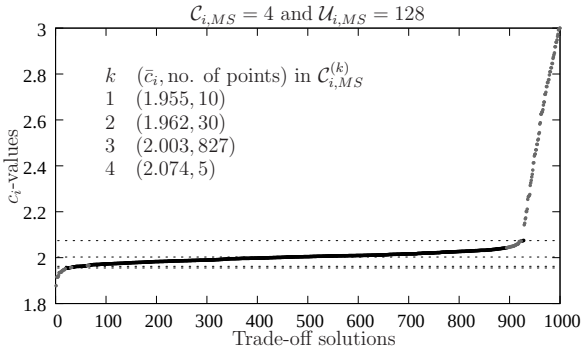

Figure 8: Design rule i = DR4 ($S_i = 87.2\%$) is equivalent to $y = 2$.

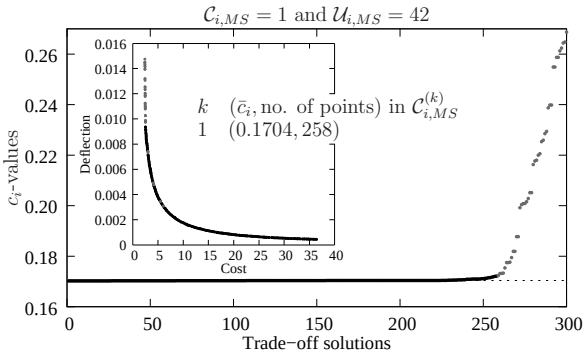

Figure 9: Design rule i = DR2 and the corresponding mapping on the Pareto front.

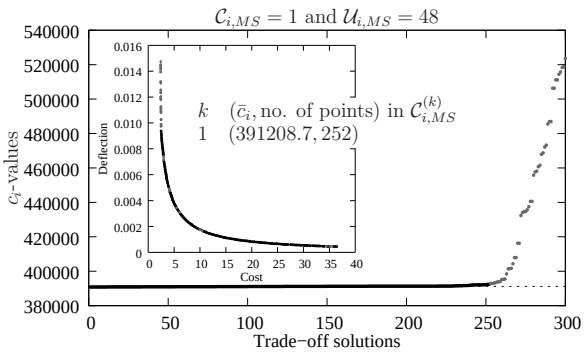

Figure 10: Design rule i = DR7 and the corresponding mapping on the Pareto front.

using NSGA-II. Next, the proposed algorithm is used on this data-set with $popsize = 600$ and $maxgen = 800$. All other parameters are same as in the truss design problem. The following basis functions are considered:

$$\phi_1 = C, \quad \phi_2 = D, \quad \phi_3 = b, \quad \phi_4 = t, \quad \phi_5 = l, \quad \phi_6 = h \quad \phi_7 = \sigma \quad \phi_8 = P_c.$$

The $\mathbf{D}$ matrix contains 213 unique design rules which reduce to just the seven relationships shown in Table 3. The following relations which were shown in a previous innovization study [8] to be the

Table 3: Reduced design rules ($\mathbf{D}_{reduced}$) for the welded beam problem, $tol = 0.01$.

i	$a_{i1}^* b_{i1}^*$	$a_{i2}^* b_{i2}^*$	$a_{i3}^* b_{i3}^*$	$a_{i4}^* b_{i4}^*$	$a_{i5}^* b_{i5}^*$	$a_{i6}^* b_{i6}^*$	$a_{i7}^* b_{i7}^*$	$a_{i8}^* b_{i8}^*$	d_i^*	S_i
DR1	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	-0.2983906	152	34.7%
DR2	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.3334565	154	86.0%
DR3	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	0.00000	-0.3328624	157	88.0%
DR4	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.00000	-0.0000215	155	83.3%
DR5	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00000	0.2031956	158	34.0%
DR6	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	-0.1844011	151	49.3%
DR7	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.3334762	167	84.0%

design rules of this problem are the approximate forms of these automatically obtained condensed rules:

$$\text{DR2: } DP_c^{0.333} = c_{\text{DR2}}, \text{DR3: } \frac{b}{P_c^{0.333}} = c_{\text{DR3}}, \text{DR4: } t = c_{\text{DR4}}, \text{DR7: } \sigma P_c^{0.333} = c_{\text{DR7}}.$$

The original study discovered a set of four design rules which can be derived from the above four rules. For example, DR2 and DR3 give $Db = constant$. Similarly, DR2 and DR7 together suggest that $D \propto \sigma$ and so on.

These rules can be very handy for a designer. As an example, consider DR2 which indicates that the beam deflection $D \propto \frac{1}{P_c^{0.333}}$, the proportionality constant being $\bar{c}_i = 0.1704$ as shown in Figure 9. A designer can convert a similar bi-objective problem into a single objective problem and still be able to create most of the trade-off solutions simply by using this rule. Similarly, the constraint on the σ can be safely eliminated by noting that $\sigma = \frac{391208.7}{P_c^{0.333}}$ applies to 252 out of the 300 trade-off solutions as shown in Figure 10. To check whether the two design rules are applicable to the same region of the trade-off front, the clustered and unclustered points from both can be mapped to the front as shown in the insets of the two figures.

Table 3 also shows three relations that do not satisfy the criterion of being applicable to at least 80% of the data-set, namely, DR1, DR5 and DR6. They occur despite the constraint on significance because of two reasons: (i) the constraint on the magnitudes of b_{ij}'s in (12) sometimes

causes spurious (yet significant) relationships to creep into the **D** matrix which carry on into $\mathbf{D}_{reduced}$ but lose their significance during the row-echelon transformation. Fortunately, they can be identified by again performing grid-based clustering as done here. In fact, the above problem was solved with 0.01 instead of 0.1 as the lower bound on magnitude since the latter resulted in DR3 having a significance of only 46% with $a^*_{i8}b^*_{i8} = -0.3314016$. This difficulty can be alleviated, if a different lower bound can be set adaptively for each $|b_{ij}|$. (ii) the reduction to echelon form causes accumulation of the errors in b_{ij}'s which in turn are due to the near-Pareto-optimal nature of the data-set. While the latter can be controlled by ensuring proximity to the Pareto-front using local search methods, work is in progress at Kanpur Genetic Algorithms Laboratory (KanGAL) to take care of the former.

6 Conclusions and Future Work

The authors' earlier work [16] on automatically deciphering design principles or rules requires that the user choose various combinations of basis functions; thus discovering each relationship one at a time. The proposed algorithm addresses the problem of finding multiple such rules simultaneously in a single optimization run by automatically eliminating unwanted basis functions from the provided set. It integrates a clustering based optimization approach with a niched-tournament selection operator to allow multiple design rules to co-exist in a GA population. Additional constraints are used to discourage insignificant rules. The algorithm is demonstrated on the truss and welded beam design problems successfully and reveals interesting information about the optimal solutions in both cases. Intuitive and easy-to-use design rules are obtained by condensing the original rules using the reduced row-echelon form. A systematic procedure for determining the tolerance required during row operations is also developed. Using a well-optimized Pareto-optimal data-set, the algorithm, as a whole, provides compact design rules which can be easily stored and retrieved for future design tasks of similar nature.

This study can be extended in a number of ways towards achieving a complete automation of the innovization process. Firstly, as seen in the welded beam problem, constraining the magnitude of b_{ij} values to a lower bound caused certain unwanted relationships to appear as rules. Though they could be identified after a row-echelon reduction and subsequent grid-based clustering, the original algorithm itself can be modified to adaptively change this lower bound. One likely approach is to set it to a value below which the corresponding ϕ_j is incapable of amounting to a threshold variation in c_i-values.

Another extension of the algorithm can be made for discrete variable problems. The limited values which a variable can take in such problems may drive the present algorithm towards trivial design rules (such as, x_i =constant), each satisfying a significant fraction of available data. If all the variables are discrete, the row-echelon form may never have a rank less than N with a reasonable tolerance. Further studies are needed to modify the current algorithm.

Rules in combinatorial optimization problems may be different than the ones obtained here. It will be an interesting study to modify the current algorithm for such problems as well. It would be also an interesting task to extend the proposed approach in more than two-objective problems.

References

[1] P Y Papalambros and D J Wilde. *Principles of optimal design: Modeling and computation.* Cambridge: Cambridge University Press, 2000.

[2] K. Deb and A. Srinivasan. Monotonicity analysis, discovery of design principles, and theoretically accurate evolutionary multi-objective optimization. *Journal of Universal Computer Science*, 13(7):955–970, 2007.

[3] S. Obayashi and D. Sasaki. Visualization and data mining of pareto solutions using self-organizing map. In *Evolutionary Multi-Criterion Optimization*, pages 71–71. Springer, 2003.

[4] S. Obayashi, S. Jeong, and K. Chiba. Multi-objective design exploration for aerodynamic configurations. *AIAA Paper*, 4666:2005, 2005.

[5] Tamara Ulrich, Dimo Brockhoff, and Eckart Zitzler. Pattern identification in Pareto-set approximations. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 737–744, New York, NY, USA, 2008. ACM.

[6] A. Oyama, T. Nonomura, and K. Fujii. Data mining of pareto-optimal transonic airfoil shapes using proper orthogonal decomposition. *AIAA Paper AIAA-2009-4000, AIAA, Reston, VA*, 2009.

[7] K. Deb. Unveiling innovative design principles by means of multiple conflicting objectives. *Engineering Optimization*, 35(5):445–470, 2003.

[8] K. Deb and A. Srinivasan. Innovization: Innovating design principles through optimization. In *Proceedings of the 8th annual conference on genetic and evolutionary computation, GECCO '06*, pages 1629–1636. New York: ACM, 2006.

[9] K. Deb. *Multi-objective optimization using evolutionary algorithms*. New York: Wiley, 2001.

[10] D. Datta, K. Deb, and C. Fonseca. Multi-objective evolutionary algorithms for resource allocation problems. In *Evolutionary Multi-Criterion Optimization*, pages 401–416. Springer, 2007.

[11] Michael Bitterman. Personal communication, July 2010.

[12] E. Madetoja, H. Ruotsalainen, and V.M. Mönkkönen. New visualization aspects related to intelligent solution procedure in papermaking optimization. In *EngOpt 2008International Conference on Engineering Optimization*, 2008.

[13] S. Doncieux, J.B. Mouret, and N. Bredeche. Exploring new horizons in evolutionary design of robots. In *Workshop on Exploring new horizons in Evolutionary Design of Robots at IROS*, volume 2009, pages 5–12, 2009.

[14] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[15] S. Bandaru and K. Deb. Automated discovery of vital knowledge from pareto-optimal solutions: First results from engineering design. In *IEEE Congress on Evolutionary Computation (CEC-2010)*, pages 1224–1231. IEEE Press, 2010.

[16] S. Bandaru and K. Deb. Towards automating the discovery of certain innovative design principles through a clustering based optimization technique. *Engineering optimization*, in press.

[17] C. K. Oei, D. E. Goldberg, and S.-J. Chang. Tournament selection, niching, and the preservation of divrsity. IlliGAL Report No. 91011, Urbana, IL: University of Illinois at Urbana-Champaign, 1991.

[18] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):311–338, 2000.