Novel Evolutionary Algorithm with Set Representation Scheme for Truss Design

A. Isaacs, T. Ray, W. Smith

Abstract—Presented in this paper is a novel scheme of representation for truss geometry. Trusses are represented as a set of elements having a collection of properties (e.g., cross-sectional area, type of material). These sets can be of varying cardinality representing truss structures with different numbers of elements and hence distinctly different topologies. A recombination operator to handle a set representation that can generate offspring topologies that can be different from the parents is also proposed. Depending on the physical problem being solved, one can introduce specific operators which will aid the optimization process. One such mutation operator is used for the truss design to reduce the number of elements, hence finding the smallest feasible topology for the truss structure. Another mutation operator perturbs the properties of the elements using the Gaussian mutation.

I. INTRODUCTION

The design of trusses involves sizing, configuration and topology optimization. In sizing optimization of trusses, the cross-sectional areas of the truss members are design variables and the coordinates of the nodes and the connectivity of the elements are kept fixed. In configuration optimization, the coordinates of the nodes are the design variables with connectivity and the cross-sectional areas of the truss elements remaining the same. Topology optimization deals with varying the connectivity of the elements.

A characteristic of topology optimization is that discrete variables are used to represent various topologies. In many cases, the candidate areas of the truss members are provided as discrete values. Genetic Algorithms (GAs) are heuristic, combinatorial search methods that can handle discrete variables easily and they have been found well suited for structural design problem involving topology optimization. Discrete values are typically represented using binary or integer coding in GAs.

Many researchers have applied genetic algorithms to solve the problem of truss design using simultaneous sizing, configuration and topology optimization. Wu and Chow [1], Kaveh and Kalatjari [2], and Coello [3] have all used binary-coded variables to represent the cross-sectional areas. In binary coding, discrete values are represented by \( n \) bits (each bit taking a value of 0 or 1), giving rise to \( 2^n \) possible combinations. Discrete values that are not powers of 2 cannot be represented exactly. In [3], there are 42 discrete values of cross-sectional areas, binary coded using 6-bits. This corresponds to a total of 64 combinations with the first 42 representing prescribed cross-section area values and the remaining 22 values are randomly selected from the discrete values. Thus, some of the area values are repeated. In the case of Ghasemi, Hinton and Wood [4], the remaining 22 values repeat the first 22 values. Kaveh and Kalatjari [2] use an additional bit per truss member to represent the presence (1) or absence (0) of the truss members in the structure.

Real-coded variables are used by Deb [5], [6], to represent the cross-sectional area; negative values of the cross-sectional areas are used to represent the absence of that truss member. For discrete values of cross-sectional area, discrete versions of simulated binary crossover (SBX) and mutation operators are used to ensure that recombination and mutation generate only discrete values as prescribed. Turkkan [7] uses the real-coding of the area variables using the real recombination and mutation operators. The area computed from recombination and mutation is mapped using the integer part of that value into an array of discrete values.

Tang, Tong and Gu [8] use mixed representation; integer coding to represent the discrete values of the cross-sectional area and binary-coding to represent the topological variables (one for each of the truss members). They also propose the recombination and mutation operators to work with the integer coding. Topology is represented using additional binary-coded variables. Rajeev and Krishnamoorthy [9] have used variable length representation in Variable String Length Genetic Algorithm (VGA) to represent different topologies for truss design. They use the cut-and-splice operator as in messy GA [10] to generate offspring with differing chromosome length from their parents.

All the cited techniques use variations of binary, integer or real coding of variables to represent discrete values of variables or different topologies. One of the drawbacks of the genetic algorithms with fixed-length coding is an inability to represent different topologies. One needs to either augment the design variables with topological variables or use clever techniques like using negative areas to represent missing truss members. Even though messy genetic algorithms with variable-length coding address this issue, it is not always possible to create topologies that are viable using a cut-and-splice operator.

Presented in this paper is a novel representation using sets for coding variables with either continuous or discrete values, and recombination and mutation operators that work using set representation. Truss structures can be very easily translated to a collection of elements having certain properties. An evolutionary algorithm using set representation is applied to the design of a minimum weight structure for a 10-member truss. The results obtained are promising, confirming the
validity and the potential of the approach.

II. REPRESENTATION

Structural entities such as a truss can be represented as a collection of truss elements. The proposed representation uses the abstraction of a set (of elements) to represent the geometric entities made up of simpler elements. A structural entity or an object can be represented by a set $S$ made up of finite elements $E_1$ to $E_n$. Cardinality of a set $|S|$ is the count of the number of elements ($n = |S|$) in that set. Sets with differing cardinality represent truss structures with varying numbers of elements.

If the ground structure for a particular truss has 10 elements, the set $S_G$ can be defined as

$$S_G = \{E_1, E_2, \ldots, E_{10}\}.$$  

Now a truss $S_1$ having 8 elements can be represented as

$$S_1 \subseteq S_G, \quad |S_1| = 8$$

Elements are characterized by a finite set of properties, $P_i$. Each property can represent a physical quantity (e.g. length, cross-sectional area), and a material property (e.g. density, Young’s modulus).

$$E_i := \{P_1, P_2, \ldots, P_k\}$$

A property can either have a fixed value or it can be a variable. Material properties (e.g. density) are represented as fixed value properties, whereas physical properties need not have fixed value. Possible values of variable properties can be defined by a range or a collection.

- **Range** - Properties such as cross-sectional area can be represented as a range of possible values. Integer ranges are also possible such as the number of gear teeth on a wheel.

  Area $\equiv P_5 \in [10.0, 20.0]$ cm$^2$

- **Collection** - Properties such as different materials can be represented as a collection of possible densities. A collection is a set of discrete values, real or integer.

  Density $\equiv P_2 \in \{0.1, 0.12, 0.2, 0.22\}$ lb/in$^3$

A random collection of truss elements from the ground structure (truss structure with all possible members included) can generate truss structures that are not viable. Analysis of such structures can waste the function evaluations budget. To weed out such structures at the time of construction, a number of constraints can be specified. A constraint can be represented as a set of elements, a subset of which have to present in the constructed set. Consider a constraint $C_1$ as a set of elements $E_1$, $E_3$, $E_7$. In a constructed set $S_1$ one or more elements of $C_1$ have to be present. Mathematically it can be represented as a non-null intersection of $S_1$ and $C_1$.

$$C_1 := \{E_1, E_3, E_7\}, \quad S_1 \cap C_1 \neq \emptyset$$

Now any truss structure can be represented as a set of elements (subset of the elements of the ground structure) with properties - Cross-sectional area, density. Cardinality of this set itself can be variable. Density can be a fixed property whereas cross-sectional area can be a variable and it can take values from a range or a collection. A number of constraints can be specified to enforce connectivity requirements. A valid structure is the one that satisfies all the constraints.

$$S_1 \subseteq S_G, \quad \forall j \ C_j \cap S_1 \neq \emptyset$$

III. ALGORITHM

The main steps of an evolutionary algorithm are depicted in Algorithm 1. The algorithm starts with initializing the population $P_1$ of the first generation.

**Algorithm 1 Toplevel Algorithm**

**Require:** $N > 1$ \{Number of Generations\}

**Require:** $N_p > 0$ \{Population size\}

1: $P_1 = $ Initialize()
2: Evaluate($P_1$)
3: for $i = 2$ to $N$ do
4:  Rank($P_{i-1}$)
5:  repeat
6:  $p_1, p_2, p_3 = $ Select($P_{i-1}$)
7:  $c_1, c_2, c_3 = $ Recombine($p_1, p_2, p_3$)
8:  Mutate($c_1$)
9:  Mutate($c_2$)
10: Mutate($c_3$)
11: Add $c_1, c_2, c_3$ to $C_{i-1}$
12: until $C_{i-1}$ has $N_p$ children
13: Evaluate($C_{i-1}$)
14: $P_i = $ Reduce($P_{i-1} + C_{i-1}$)
15: end for

The other major steps of the evolutionary algorithm are:

- **Evaluate** - For all the individuals of a population calculate the objective and the constraint functions.
- **Rank** - Rank all the candidate solutions in the population to identify good candidate solutions.
- **Select, Recombine, Mutate** - Using Selection, Recombination and Mutation operators a child population is evolved from the parent population.
- **Reduce** - From parent population and child population select better individuals (using ranks) to form the next generation of individuals.

A. Initialization

Each individual of the population is a subset $S$ of the ground set $S_G$. The cardinality of this set can be constrained by prescribing limits as either a fixed number ($|S| = 7$) or an integer range ($|S| = [6, 8]$). When choosing the elements of the subset $S$, it is made sure that any constraints specified are satisfied. Outlined in Algorithm 2 is the backtracking algorithm which constructs various subset combinations by exchanging elements. The algorithm starts by selecting a random subset $S$ (i.e. the elements are picked
up in random order) from the ground set \( S_G \). The function \( \text{VerifyConstraints}(S) \) checks if the subset \( S \) satisfies all the specified constraints. If the subset \( S \) does not satisfy the constraints, a new subset \( S_t \) is formed by exchanging the elements from \( S \) and \( S_G - S \). This iterative step methodically swaps elements and can run through all the possible combinations of \( S \).

**Algorithm 2 Subset Construction**

**Require:** \( S_G \) (Ground set)

**Require:** Count(\( S \)) = \([n_1, n_2]\), \( n_1 < n_2 \leq |S_G| \)

1: \( n = \text{random}[n_1, n_2] \)
2: Select \( S \subseteq S_G, |S| = n \)
3: if VerifyConstraints(\( S \)) then
4: STOP
5: end if
6: for \( p \in S \) do
7: \( S_t = S_G - S \) (Remaining elements)
8: for \( q \in S_t \) do
9: \( S_1 = (S - \{p\}) \cup \{q\} \) (New subset)
10: if VerifyConstraints(\( S_1 \)) then
11: \( S = S_1 \) (Found valid subset)
12: STOP
13: end if
14: end for
15: end if

**Ensure:** \( S \subseteq S_G, n_1 \leq |S| \leq n_2 \)

Once a valid subset \( S \) is found, properties of all the elements in the subset \( S \) are initialized. Only variable properties need to be initialized. For each property, a value is randomly chosen from a range or a collection depending on the kind of the property.

**B. Ranking**

Ranking of the individuals is used to identify the better individuals and it is based on the value of the objective and the constraints. For unconstrained problems, the individual with the better objective function value has a higher rank. For constrained problems a feasible solution has a higher rank than an infeasible solution. Within feasible solutions the ranks are computed based on the objective function value. Within infeasible solutions the ranks are based on the maximum constraint violation. Individuals with higher maximum constraint violations are ranked lower.

**C. Selection**

Parents are selected from the population and undergo recombination to create offspring. Parents are selected using binary tournament. To select a parent, two individuals are picked up from the population randomly and the individual with lower rank is selected as a parent. Since the recombination operator requires 3 parents, 6 distinct individuals are chosen from the population and 3 parents are chosen using binary tournament.

**D. Recombination**

Recombination is used to create offspring from parents. The \( \text{REDISTRIBUTE} \) operator outlined in Algorithm 3 is used for recombination. \( \text{Recombination Probability} \) determines if the offspring are generated from the parents or created randomly. The random creation step helps create diversity in the population.

**Algorithm 3 REDISTRIBUTE Operator**

**Require:** \( S_1, S_2, S_3 \) [3 Parents chosen for recombination]

**Require:** \( P_r > 0 \) [Recombination Probability]

**Require:** \( P_r > 0 \) [Resizing Probability]

**Require:** Count(\( S_i \)) \( \in \) \([n_1, n_2]\), \( 0 < n_1 \leq |S_i| \leq n_2 \leq |S_G| \)

1: if random[0, 1] < \( P_r \) then
2: if random[0, 1] < \( P_r \) then
3: \( m_1, m_2, m_3 = \text{random}[n_1, n_2] \)
4: add_missing = 1
5: else
6: \( m_1 = |S_1|, m_2 = |S_2|, m_3 = |S_3| \)
7: add_missing = 0
8: end if
9: \( S_n = S_1 \cap S_2 \cap S_3 \) (Common Elements)
10: \( V = \{\forall k P_k \in E_i, \forall j E_j \in S_k\} \) (Collect properties)
11: \( \text{PCX}(V) \) (PCX operator for common properties)
12: \( S_n = S_1 + S_2 + S_3 \)
13: if add_missing = 1 then
14: \( S_n = S_t + (S_G - S_t) \)
15: end if
16: \( S'_1, S'_2, S'_3 = \text{Subset}(S_n) \)
17: else \{No Recombination\}
18: \( S'_1, S'_2, S'_3 = \text{Subset}(S_G) \)
19: end if

The main step involves redistribution of elements of all the three parents \( (S_1, S_2, S_3) \) to create offspring \( (S'_1, S'_2, S'_3) \). The cardinality of each of the offspring is decided by the \( \text{Resizing Probability} \) \( (P_r) \). If the random number is less than the resizing probability, cardinality of the offspring is the same as that of the corresponding parent \( |S'_1| = |S_1|, |S'_2| = |S_2|, |S'_3| = |S_3| \), otherwise cardinality of the offspring is randomly selected from the allowed range.

A pool \( (S_3) \) of all the elements from three parents is created. This pool will have multiple copies of elements as the parents can have common elements (either the same or different properties). If the offspring cardinality are different from that of the parents, it may not be possible to create offsprings using only the elements since there may not be sufficient distinct elements. In such cases all the missing elements are added to the pool (add_missing = 1). Offsprings are now created by picking subsets from the pool using Algorithm 2.

If there are common elements in all the parents, then the properties of all such common elements are grouped together \( (V) \) and the parent centric recombination (PCX) [11] operator is used to create properties with new values. Subsequently, the new values are used for the properties of the common
elements.

E. Mutation

The mutation operator (MUTATE) perturbs the values of the randomly selected properties of randomly selected elements from the set. For the real continuous values, Gaussian mutation operator [12] is used to modify the values. For properties with discrete values, a new value is randomly selected from the possible values.

For the problem of truss design, the design is driven by the stress and displacement constraints. It is easier for the algorithm to pick up sets with a large number of elements as they will satisfy the constraints easily. To ensure that the final structure has fewer elements, an operator is required that will keep removing the elements from the structure. In addition to the standard mutation operator, a DELETE operator is used. This is a mutation operator which randomly deletes an element from the structure.

Either, the DELETE operator or the MUTATE operator is used as the mutation operator. In the problem definition the probabilities for DELETE and MUTATE operators are specified. This is in addition to the mutation probability. Mutation probability determines if a particular candidate solution will undergo mutation or not. The operator probability determines which mutation operator (DELETE or MUTATE) is used for mutation.

IV. Case Studies

A. Problem Description

A 6-node, 10-member ground structure is shown in Figure 1. The problem is the design of the minimum weight truss structure using simultaneous sizing and topology optimization.

![Ground structure for 6-Node, 10-Member Truss](image)

Fig. 1. Ground structure for 6-Node, 10-Member Truss

The structure is loaded with a force \( F \) of 100000 lbf at nodes 2 and 4. The dimensions of the structure and the density of the material are given in Table I. The maximum allowed stress in compression or tension is 25 kpsi. The maximum displacement allowed at nodes 2 and 4 is 2 in.

For sizing optimization, member cross-sectional areas are the design variables. Depending on the values of the areas used, the following three separate problems are defined.

| The first problem uses a continuous variation in the cross-sectional area of each member, the other two problems use two different sets of discrete values for the allowable areas. |
| Problem 1: Continuous area variation from 1 in\(^2\) to 35 in\(^2\) |
| Problem 2: Discrete area variation from 1 in\(^2\) to 35 in\(^2\) in steps of 1 in\(^2\) |
| Problem 3: Discrete area variation from available member sizes taken from the \textit{American Institute of Steel Construction Manual} [9]. |
| \( A = (1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50) \) All the values are in square inches. |

B. Problem Representation

This truss design problem can be described by the set representation shown in Table II. The problem description is organized in sections. The [Algorithm] section in Table II defines the parameters controlling the optimization process. Parameters \textit{Population} and \textit{Generations} determine the size of the population and number of generations for which the evolutionary algorithm runs. \textit{Fitness} defines the user-defined function to be used for calculating the objective function and the constraints. \textit{Objectclasses} parameter lists the object classes that will define the physical problem. In this case there is single object class \textit{Beams}, which defines the truss structure.

The [Beams] section in Table II identifies that a set type representation is to be used for this object. The elements belonging to the ground structure are given as B1, \ldots, B10, defined by parameter \textit{Elements}. Each object can have from 5 elements to 10 elements as defined by the \textit{Count} (Cardinality) parameter. This range can be shrunk to reduce the size of the design space. Constraints of the construction of the set are defined by parameters \textit{Constraint}1, \ldots, \textit{Constraint}4. At least one of the elements from each constraint should be present in any valid object.

The elements are described next, in separate sections. In Table II, the [B1] section defines the properties for element B1. The \textit{Area} parameter lists the range of cross-sectional areas, from 1 in\(^2\) to 35 in\(^2\). The density is given as 0.1 lb/in\(^3\).

If all the properties of each element are identical, then
TABLE II
OPTIMIZATION PROBLEM DESCRIPTION OF 10-MEMBER TRUSS

<table>
<thead>
<tr>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population = 100</td>
</tr>
<tr>
<td>Generations = 100</td>
</tr>
<tr>
<td>Fitness = Truss_6node</td>
</tr>
<tr>
<td>ObjectiveClasses = {Beams}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Beams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = Set</td>
</tr>
<tr>
<td>Elements = {B1,B2,B3,B4,B5,B6,B7,B8,B9,B10}</td>
</tr>
<tr>
<td>Count = [5,10]</td>
</tr>
<tr>
<td>Constraint_1 = {B1, B7}</td>
</tr>
<tr>
<td>Constraint_2 = {B3, B8}</td>
</tr>
<tr>
<td>Constraint_3 = {B5, B7, B10}</td>
</tr>
<tr>
<td>Constraint_4 = {B6, B9}</td>
</tr>
<tr>
<td>Mutation_Operators = {MUTATE:0.8, DELETE:0.2}</td>
</tr>
<tr>
<td>Recombination_Operators = {REDISTRIBUTE:1.0}</td>
</tr>
<tr>
<td>Recombination_Probability = 0.8</td>
</tr>
<tr>
<td>Mutation_Probability = 0.2</td>
</tr>
</tbody>
</table>

[B1] Area = [1.0, 35.0] density = 0.1

[B2] inherit: B1

instead of defining all the properties for each element repeatedly, the inherit parameter can be used. This allows the element [B2] to inherit all the properties (i.e. area and density) from the element [B1].

C. Experimental Setup

All the design problems (Problems 1, 2 & 3) are solved with a population size of 100 and maximum 400 generations. Each problem is solved using 6 runs with variations in mutation probability and random seed. Recombination probability (P_r) is fixed at 0.8. The PCX parameters \(\sigma_\eta\) and \(\sigma_\zeta\) are fixed at 0.1 each. Resizing probability (P_s) is set to 0.2. Operator probability for MUTATE operator and DELETE operator is kept fixed at 0.8 and 0.2 respectively. Increasing the probability of DELETE operator affects the ability of the algorithm to find feasible structures, and reducing the probability results in a larger number of generations to obtain the optimum structure. Distribution Index for the mutation operator is set at 20.

Objective function (weight of the truss) and the constraint functions (stress in truss members, displacement at nodes 2 and 4) are evaluated using the finite element method. ANSYS (Version 10.0) is used as the finite element solver. The evolutionary algorithm generates an ANSYS macro file for each of the candidate solution to be evaluated, invokes ANSYS, and extracts the stresses and displacements from the outputs generated by ANSYS.

D. Results

The best truss design obtained is shown in Figure 2 and is made up of 6 members connected to 5 nodes. The algorithm is able to find the same topology (with members 1,3,4,7,8, and 9) for all the three problems. This topology is the best known solution for the 10-member truss design problem.

![Optimum structure for 6-Node, 10-Member Truss](image)

Fig. 2. Optimum structure for 6-Node, 10-Member Truss

1) Problem 1: The results obtained by the proposed algorithm and the results published by Deb and Gulati [6] are given in Table III. The results by Deb and Gulati are based on a population size of 220 and obtained after 225 generations. The SBX operator is used for recombination and polynomial probability based operator is used for mutation. The recombination probability is 0.9 and mutation probability used is 0.1. Proposed algorithm using set representation is able to obtain the result using less number of function evaluations.

![Comparison of results for Problem 1](image)

TABLE III
COMPARISON OF RESULTS FOR PROBLEM 1

<table>
<thead>
<tr>
<th>Member</th>
<th>Current Work Area (in^2)</th>
<th>Deb &amp; Gulati Area (in^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24.525</td>
<td>29.68</td>
</tr>
<tr>
<td>3</td>
<td>25.557</td>
<td>22.07</td>
</tr>
<tr>
<td>4</td>
<td>17.906</td>
<td>15.30</td>
</tr>
<tr>
<td>7</td>
<td>6.202</td>
<td>6.09</td>
</tr>
<tr>
<td>8</td>
<td>25.162</td>
<td>21.44</td>
</tr>
<tr>
<td>9</td>
<td>19.724</td>
<td>21.29</td>
</tr>
<tr>
<td>Weight (lb)</td>
<td>5048.49</td>
<td>4899.15</td>
</tr>
</tbody>
</table>

2) Problem 2: Listed in Table IV are the results obtained by the set representation algorithm and the results published by Deb and Gulati [6]. The results by Deb and Gulati are using same parameters as mentioned earlier (population size of 220 ran for 225 generations).

![Comparison of results for Problem 2](image)

TABLE IV
COMPARISON OF RESULTS FOR PROBLEM 2

<table>
<thead>
<tr>
<th>Member</th>
<th>Current Work Area (in^2)</th>
<th>Deb &amp; Gulati Area (in^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>31</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>Weight (lb)</td>
<td>5029.53</td>
<td>4912.85</td>
</tr>
</tbody>
</table>

The weight variation versus generations is shown in Figure 3. The topology evolution for Problem 2 is shown in
Figure 4 and it is seen that the algorithm finds the best topology very early in the evolution.

It is worth noting that the proposed algorithm uses a small population size of 100 and 40,000 function evaluations as compared to population size of 225 and 49,500 function evaluations in [6]. It is also not clear how the recombination probability of 0.9 and mutation probability of 0.1 have been selected in [6].

<table>
<thead>
<tr>
<th>Member</th>
<th>Current Work Areas (in$^2$)</th>
<th>Kaveh &amp; Kalatjari Areas (in$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.00</td>
<td>30.00</td>
</tr>
<tr>
<td>3</td>
<td>26.50</td>
<td>19.90</td>
</tr>
<tr>
<td>4</td>
<td>14.20</td>
<td>15.50</td>
</tr>
<tr>
<td>7</td>
<td>7.22</td>
<td>7.22</td>
</tr>
<tr>
<td>8</td>
<td>18.80</td>
<td>22.00</td>
</tr>
<tr>
<td>9</td>
<td>22.90</td>
<td>22.00</td>
</tr>
</tbody>
</table>

Weight (lb) 5035.74 4962.1

3) Problem 3: Listed in Table V are the results for problem 3 obtained using the proposed algorithm and published by Kaveh and Kalatjari [2] using binary-coded GA. They have used recombination probability of 0.9, mutation probability for area variables of 0.1 and mutation probability for topological variables is 0.001. The algorithm runs in multiple stages, each stage composed of 100 generations of evolution using population size of 50. The best structure reported is obtained in stage 4.

Problem 3 is also solved using sizing optimization keeping the topology fixed to the best topology obtained. The results obtained are shown in Figure 5 in the form of weight variation.

The algorithm is run with population size of 24 for 200 generations. Only the MUTATE operator was used for mutation as the topology is fixed. The figure plots the weights for best feasible solutions in each generation. The proposed algorithm recovers the best solution for the truss design with the weight of 4962.1 lb as reported in [2].

V. CONCLUSION

A novel representation based on a set is proposed. The set representation allows very easy definition of the varying topology and the discrete values for variables. Use of properties allows flexibility in specifying number of design variables of different types. Continuous variables and discrete variables can be mixed in the problem definition without worrying about how those variables need to be coded.

A new representation usually requires redefining the existing operators or defining the new ones. We have proposed new operators; a recombination operator that can create topologically different structures and a context specific mutation operator that tries to obtain the smallest possible structure. Gaussian mutation operator is adapted to the set representation to perturb the properties of the elements of the structure. Further work is required to improve the efficiency of the operators, to improve convergence and reduce function evaluations.

A modular structure for the optimization problem definition allows easy reconfiguration of the optimization problem without changing the algorithm. It is possible in the future, to create new object classes with abstract representation that can be easily mapped to various physical domains. A framework is in place to handle multiple object classes definition simul-
taneously and to support mixed variable coding strategies used in the genetic algorithms.

The results for the 10-member truss design included in the paper are preliminary results obtained from a few experimental runs. Random seed and mutation rates have been varied to achieve the results. For the simultaneous sizing and topology optimization problems, the weights obtained using the set representation are very close to the best reported in the literature. For the sizing optimization problem the same optimum design as reported in [2] was obtained using the population size of 24 and allowing it to evolve over 200 generations which highlights the efficiency of the operators in place.

REFERENCES


