Optimization of Truss Structures Using Genetic Algorithms with Domain Trimming (GADT)

[Samer Barakat and Omar Nassif]

Abstract—**This paper presents evolutionary-based least-weight optimization procedure for designing truss structures. A modified version of Genetic Algorithm with Domain Trimming (GADT) is developed and presented herein. The DADT is used for solving the nonlinear constrained optimization problems. In this optimum design formulation, the objective function is the material weight of the truss; the design variables are the cross-sections of the truss members; the constraints are the stresses in members and the displacements of the joints. The constraints were handled using non-stationary dynamically modified penalty functions. One classical truss optimization example is presented herein to demonstrate the efficiency of the GADT algorithm. The test problem includes a 10-bar planar truss subjected to a given load condition. The result shows that the GADT method is very efficient in finding the best discovered optimal solutions, which are better of the results of other structural optimization methods.**

Keywords—**Truss Structural Optimization, Genetic Algorithm, Domain Trimming, Constraint Handling.**

I. Introduction

Obtaining optimal designs that satisfy multiple conflicting criteria, such as minimum cost and maximum performance, is one of the most influential factors in modern structural design. Most structural designs are considered constrained optimization problems that can be solved to identify the design values of structural performance. The optimum solutions might be linearly and/or nonlinearly constrained in the design space. In the presence of multiple optima and non-smooth constraints in the design variable space, it is difficult to obtain a set of optimum values using local optimization techniques. On the other hand, this difficulty has geared the research towards relatively new and innovative evolutionary based optimization techniques such as the Genetic Algorithm (GA) [1], Ant Colony Optimization (ACO) [2], Particle Swarm Optimizer (PSO) [3, Shuffled Complex Evolution (SCE) [4], Harmony Search [5], and Hybrid Methods [6]-[8]. These approaches are investigated and used in recent years for optimizing structural designs and are proven superior to local search techniques. Many structural optimization problems involve problem-specific constraints applicable to the solutions limiting the feasible search space Compared to other constraint handling techniques the use of penalty functions is relatively simple and easy to implement.

This study presents the development and implementation of the GADT to achieve superior optimization. The capabilities of the developed optimization tool are demonstrated on two classical truss optimization problems being challenging with unknown global and multiple local minima.

II. Genetic Algorithm with Domain Trimming (GADT)

A. *Development of the GADT Optimization Technique*

To begin GA optimization, a population of solution alternatives Np (population size) are randomly generated using a uniform probability distribution; each solution of the GA consists of a combination of variables (x1, x2, x3, …, xn) which has its own fitness value. In cases where the optimization is performed to find the minimum weight for a given problem, a function F*j(X) = total mass + penalty has to be minimized. Solution alternatives that yield low F*j(X) values for the objective function would have better fitness as long as they are not violating the problem constraints. Populations of solutions are represented by chromosomes. The design variables stored in the chromosome can be either discrete (selected from a pool of defined values) or continuous (selected from a continuous range of variables). In this research, the vector of variables contains continuous values. Once F*j(X) for every solution j in the initial population is computed, a fitness value is assigned to each solution j using Eq.(1), Solutions with F*j(X) less than F*ave of the population are considered unfit and are eliminated by assigning them a fitness value of zero:

$$F_j(\boldsymbol{X}) = \begin{cases} \mathrm{F}^*{}_{\mathrm{ave}} - F^*{}_j(\boldsymbol{X}) & \text{for } F^*{}_j(\boldsymbol{X}) < \mathrm{F}^*{}_{\mathrm{ave}} \\ 0 & \text{for } F^*{}_j(\boldsymbol{X}) \geq \mathrm{F}^*{}_{\mathrm{ave}} \end{cases}$$
$$\mathrm{j} = 1,2,\ldots,\mathrm{N_p} \qquad (1)$$

where (***X***) is the vector of the design variables.

The three basic operations of a GA, reproduction, crossover, and mutation, are used to improve the fitness of each population from one generation (iteration) to the next. The reproduction operation selects the better fit designs, copies them, and places them into a mating pool allowing each to mate and reproduce. The roulette wheel selection method is used in this study for its simplicity and popularity.

After the reproduction operation is performed, the crossover operation mates the selected designs to create more fit offspring solutions. The uniform crossover operation is used to combine genetic information between two parent solutions. Uniform crossover selects two parent solutions at a time from the mating pool and swaps variables corresponding to zeros in a binary vector known as a mask. The mask is the same length as all variable vectors and consists of a preselected percentage of randomly arranged zeros (%c). This percentage has an impact on the speed of

Samer Barakat

Dept. of Civil and Environmental Engineering/University of Sharjah
UAE

Omar Nassif

Department of Civil Engineering/University of Tennessee
USA

convergence to an optimum solution. Each mask within a population is different, so the number of unique, randomly generated masks is equal to half of the number of solutions (parents) in the population multiplied by the total number of generations (populations).

Since GA mimics the natural selection of the fittest treats through multiple generations, it is inherently vulnerable to *Genetic Drift* (continuous survival of "unfit" but "lucky" individuals, and their genes, from one generation to the next). The mutation operation is used to minimize the effect of genetic drift and add diversity to the search space by randomly changing a variable in a design solution. During mutation the value of any chromosome variable may be changed to a randomly selected variable. Another counter measure of genetic drift used for continuous variables is to use blending techniques [12], a general blending formula would be of the form:

$$x_{new} = \beta x_{mn} + (1-\beta)x_{dn} \qquad (2)$$

Where, x_{new} = the nth variable of the offspring chromosome of the crossover, β = random number on the interval [0,1], x_{mn} = the nth in the mother chromosome, and x_{dn} = the nth in the father chromosome.

One of the main advantages for the GA optimization is its relative insensitivity to local minima (not very susceptible to being trapper in local minima). However, this may also be considered as one of its limitations: being a *"low-resolution"* technique especially if the population size is relatively limited. i.e. convergence may sometime occur to points in the neighborhood of, but not exactly at, global minima. The reason for this is that initially, the solution variables are randomly selected from a pre-specified range (domain); those variables do not change (except for mutation, which occurs at small probability) but rather change place from one solution to another. In addition, a poor choice of the solution domain (e.g.: [1, 1000] while the optimal value is at a value of 2) further aggravates the issue. Blending techniques, while acting to reduce this disadvantage, introduce a new random parameter (β), it may also have a negative effect as it may negate the strong traits of the parents.

In this study, a new technique is developed to alleviate this inherent limitation of GA. The technique involves trimming the domain then re-initiating the GA so that the probability of selecting the optimal solution is improved. For example, trimming a domain from [1, 1000] to [1, 10] then re-initiating GA would increase the initial probability of selection of the optimal solution by 200 times, when everything being constant. Trimming is done as a percentage of original domain size, and continuous trimming then GA re-initiation goes on until the optimum solution is found. One can think of trimming in terms of evolution theory as when a catastrophe occurs in nature eliminating the majority of the population leaving only the elite survivals to restart the evolution process. Fig.4 simulates this technique on a single variable chromosome (taking trimming as 90% of domain size):

1- After the GA converged to a certain objective function value, the elite 10 chromosomes (in this case, variables) are taken regardless of which generation they're in, since the range between the minimum and maximum variable is more than 90%, no trimming occurs.

2- As the GA converged for the second time, the range of values are now less than 90% and trimming will happen, but rather than being centered, the trimming range is biased toward the mean by a ratio: (max-mean)/(max-min), the trimmed domain is now less than 90% of the original as some of the trimming range lies outside the original domain.

3,4,5 - Because the domain has been trimmed, the chances of getting to the global optimum are higher, resulting in the elite values getting closer to each other, trimming will stop when range of values exceeds 90% of the domain or if the GA found the optimum solution to a satisfactory precision.

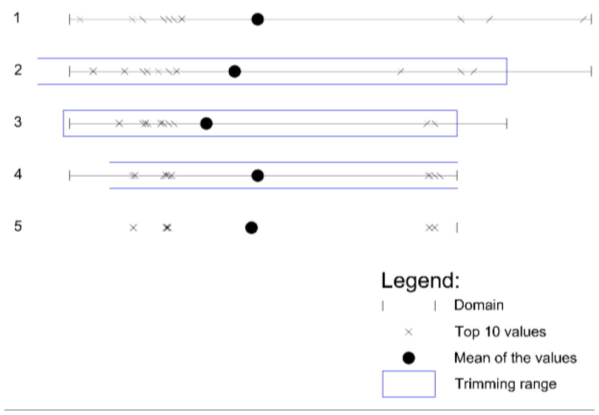


Fig. 4: Illustration of the domain trimming method on a single variable (trimming is 90%)

A question arises that how can it be sure that the optimal solution is inside the trimmed domain; the answer is that it doesn't guarantee that, however, the worst case scenario is that it will get results comparable to conventional GA since trimming doesn't happen until GA converges. Experimenting on sample problems showed that the technique gives better results than conventional GA. Fig. 5a-d illustrates the logical steps for the GADT technique via flowcharts.

To demonstrate the GADT algorithm performance, one of the standard test functions in optimization problems is considered: the Six-Hump Camelback function problem:

$$f(x,y) = \left(4 - 2.1x^2 + x^{\frac{4}{3}}\right)x^2 + xy + (-4 + 4y^2)y^2 \qquad (3)$$

With boundaries: x ∈ [-1.5, 1.5] and y ∈ [-2, 2]. The function takes the form shown in Fig. 6; it has six minima, two of them are global minima with a value of: -1.0316 located at the two points: (-0.0898, 0.7126) and (0.0898, -0.7126). Initially, the GA starts off with the domain provided by the user and iterates until there is a convergence, i.e. the fitness values are close to each other and the improvement in best fitness in subsequent iterations is less than a specified value, say, 1%. At this stage, the GA has identified a number of local minima and further iterations will have little effect on finding the global minimum. Convergence to a series of *"not necessarily most fit"* solutions means that the GA is approaching the vicinity of the global optimum but cannot discover the fittest solution. This may be due to elimination of some of the fittest solutions as they are combined with lesser-fit solutions, or inclusion of residual unfit solutions within the available solution domain.

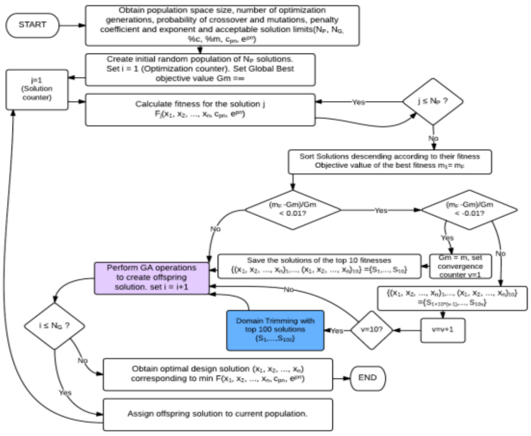

START
Obtain population space size, number of optimization generations, probability of crossover and mutations, penalty coefficient and exponent and acceptable solution limits(N_P, N_G, %c, %m, c_{pn}, e^{pn})
Create initial random population of N_P solutions. Set i = 1 (Optimization counter). Set Global Best objective value Gm =∞
j=1 (Solution counter)
Calculate fitness for the solution j $F_j(x_1, x_2, ..., x_n, c_{pn}, e^{pn})$
Yes
j ≤ N_P ?
No
Sort Solutions descending according to their fitness Objective valtue of the best fitness m_1= m_F
$(m_F$ -Gm)/Gm < 0.01?
Yes
$(m_F$ -Gm)/Gm < -0.01?
No
Yes
No
Perform GA operations to create offspring solution. set i = i+1
Save the solutions of the top 10 fitnesses $\{(x_1, x_2, ..., x_n)_1,..., (x_1, x_2, ..., x_n)_{10}\} =\{S_1,..., S_{10}\}$
Gm = m, set convergence counter v=1
$\{(x_1, x_2, ..., x_n)_1,..., (x_1, x_2, ..., x_n)_{10}\} =\{S_{1+10*(v-1)},..., S_{10v}\}$
No
i ≤ N_G ?
Domain Trimming with top 100 solutions $\{S_1,...,S_{100}\}$
Yes
v=10?
v=v+1
No
Obtain optimal design solution $(x_1, x_2, ..., x_n)$ corresponding to min $F(x_1, x_2, ..., x_n, c_{pn}, e^{pn})$
END
Yes
Assign offspring solution to current population.

Fig. 5a: Flow chart of general GADT optimization

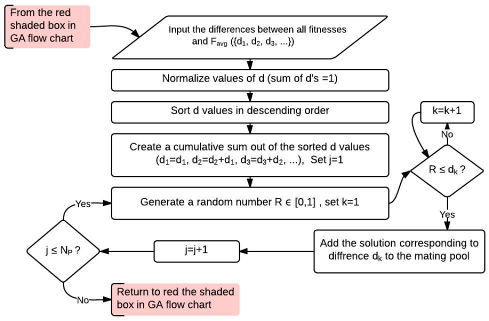

From the red shaded box in GA flow chart
Input the differences between all fitnesses and F_{avg} $(\{d_1, d_2, d_3, ...\})$
Normalize values of d (sum of d's =1)
Sort d values in descending order
k=k+1
No
Create a cumulative sum out of the sorted d values $(d_1=d_1, d_2=d_2+d_1, d_3=d_3+d_2, ...)$, Set j=1
R ≤ d_k ?
Yes
Generate a random number R ϵ [0,1] , set k=1
Yes
j ≤ N_P ?
j=j+1
Add the solution corresponding to diffrence d_k to the mating pool
No
Return to red the shaded box in GA flow chart

Fig. 5b: Flow chart of Reproduction, Crossover and Mutation

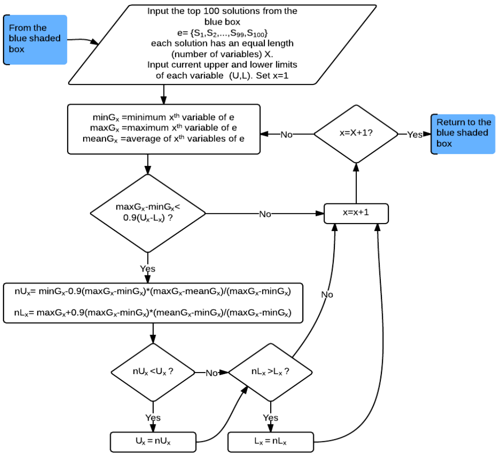

From the blue shaded box
Input the top 100 solutions from the blue box $e= \{S_1,S_2,...,S_{99},S_{100}\}$ each solution has an equal length (number of variables) X. Input current upper and lower limits of each variable (U,L). Set x=1
$minG_x$ =minimum x^{th} variable of e
$maxG_x$ =maximum x^{th} variable of e
$meanG_x$ =average of x^{th} variables of e
No
x=X+1?
Yes
Return to the blue shaded box
$maxG_x-minG_x< 0.9(U_x-L_x)$?
No
x=x+1
Yes
$nU_x= minG_x-0.9(maxG_x-minG_x)*(maxG_x-meanG_x)/(maxG_x-minG_x)$
$nL_x= maxG_x+0.9(maxG_x-minG_x)*(meanG_x-minG_x)/(maxG_x-minG_x)$
No
$nU_x <U_x$?
No
$nL_x >L_x$?
Yes
Yes
$U_x = nU_x$
$L_x = nL_x$

Fig. 5c: Flow chart of Roulette wheel selection

At this point, refinement of the available solution domain improves the chances of discover the fittest solution. GADT technique removes the most unfit solutions from the domain and re-initiates GA within the trimmed domain. Occasionally, some fittest solutions are dismissed as unfit as result of being combined with lesser-fit solutions. To minimize the probability of this potential pitfall, two measures are incorporated in the GATD: (a) the initial domain considers a large enough population to allow higher probability for discovering the most combinatory possibilities, and (b) the domain trimming limited to a high percentage (e.g. 90%) of the domain from the previous step. This means that while some values are clearly identified as unfit, they are still retained in the domain for the subsequent GA re-initiation. The motivation for this retention it exhaust all possibilities of producing fit solutions before completely eliminating part of the domain. The impact on speed of discovery of optimum solutions is obvious. However, it is justified by the significantly improved successful discovery rate. The method can also be helpful to identify what range of variables to look for in subsequent searches.

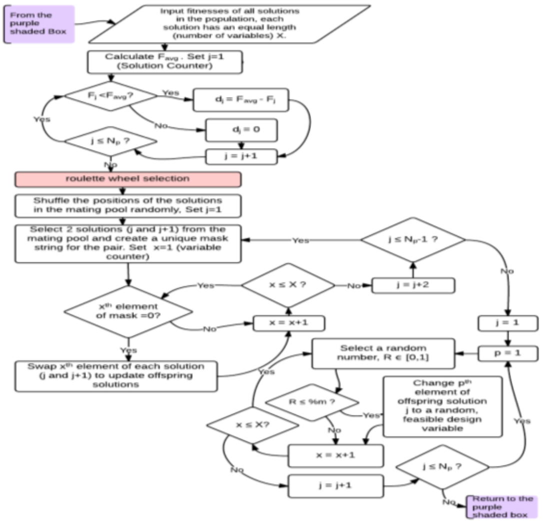

From the purple shaded Box
Input fitnesses of all solutions in the population, each solution has an equal length (number of variables) X.
Calculate F_{avg} . Set j=1 (Solution Counter)
F_j <F_{avg}?
Yes
$d_j = F_{avg} - F_j$
Yes
No
d_j = 0
j ≤ N_p ?
j = j+1
No
roulette wheel selection
Shuffle the positions of the solutions in the mating pool randomly, Set j=1
Select 2 solutions (j and j+1) from the mating pool and create a unique mask string for the pair. Set x=1 (variable counter)
Yes
j ≤ N_p-1 ?
No
Yes
x ≤ X ?
No
j = j+2
x^{th} element of mask =0?
No
x = x+1
j = 1
Yes
Select a random number, R ϵ [0,1]
p = 1
Swap x^{th} element of each solution (j and j+1) to update offspring solutions
Yes
R ≤ %m ?
Yes
Change p^{th} element of offspring solution j to a random, feasible design variable
Yes
x ≤ X?
No
x = x+1
No
j ≤ N_p ?
j = j+1
No
Return to the purple shaded box

Fig. 5d: Flowchart of Domain Trimming

Fig.7 shows the fittest 100 overall solutions throughout the GADT; the box drawn on the contour map represents the updated (trimmed) domain, eliminating 10% of the original domain at each step. The continuation of domain trimming results in identifying new local minima. As more local minima are identified, they begin to compete with each other. Eventually, the trimming technique will exclude the values of low fitted local minima and will deem them as unfit results. This helps the GA to focus more on only the top-fitted solutions ultimately enhance the discovery rate (precision) of global optima solutions. Fig.7d shows the final domain after 20 domain trimming iterations; it is noted that the domain cannot get trimmed further as the two global minima reside on the trimmed boundaries of the last step. It is worth mentioning that the final domain is only 3.5% of the initial full domain in more precise (virtually exact) optimum solution: f(x, y) = -1.0316284533464 (0.000000014% error).

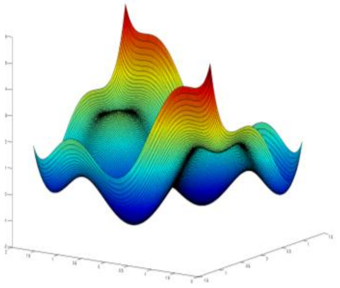
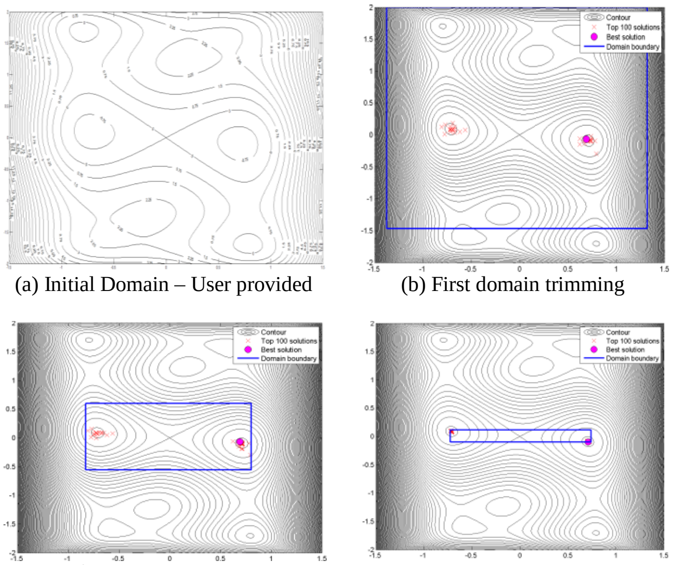


Fig. 6: Six-hump Camelback function. a) 3D plot b) Contour plot

B. *GADT Technique Robustness*

To ensure GADT robustness, domain trimming is subject to further criteria preventing any potential adverse consequences. Domain trimming will not resume in any of the following scenarios: (a) if the range of values that yield



(a) Initial Domain – User provided

(b) First domain trimming

(c) after 10 domain trimming

(d) Final solution domain

Fig.7: Illustration of the GADT algorithm applied to the Six-hump Camelback function at different iterations of domain trimming.

fit results in the current domain is more than 90% of the previous domain; in other words, domain trimming will not commence until the feasible values get condensed into less than 90% of the domain.

And (b) if after trimming, the GA yields solutions worse than previous GA initiation. Although the latter was not encountered during any of the algorithm testing sessions for any of the presented problems here, the criterion is set in place for potential future problem-specific complications. The effect of initial domain size is discussed in a subsequent section.

C. *Constraints Handling*

The use of penalty functions is very popular in handling constraints enabling the solution of constrained problems as unconstrained. The solutions that violate any constraints are penalized in order to characterize non-feasible solutions by high objective function values. Non-stationary (dynamic) penalty functions typically exhibit superior performance to stationary (static) penalty functions. In its generality, a non-stationary penalty function is defined as:

$$f(X) = F(X) + p(X, c_{pn}, e_{pn}) \tag{4}$$

Where *F(x)* is the original objective function of the constrained optimization problem; *p(•)* is a dynamically modified penalty value, defined as:

$$p(X, c_{pn}, e_{pn}) = \begin{cases} (c_{pn} r_i)^{e_{pn}}, & r \geq 1 \\ 0, & r < 1 \end{cases} \tag{5}$$

Where, $\boldsymbol{r_i}$ is the individual member's performance criteria (in a structural design problem, this is often taken as the demand-to-capacity ratio, or utilization ratio to the satisfaction of the relevant design code); while $\boldsymbol{c_{pn}}$ and $\boldsymbol{e_{pn}}$ are the penalty coefficient and exponent, respectively. As their name implies, they provide means to penalize the optimization objective if the $\boldsymbol{r_i}$ exceeds unity. Both $\boldsymbol{c_{pn}}$ and $\boldsymbol{e_{pn}}$, with different severity, will penalize unfit solutions minimizing their probability of re-appearing subsequent generation and feasible solution domain.

D. *Truss Structural Optimization*

The mathematical form of the optimization problem for truss structure can be expressed as follows:

Find $A^T = \{A_1, \ A_2, \ .... \ A_n\}$ (6)

To Minimize $F = W(A) = \rho \sum_{i=1}^{n} L_i A_i$ (7)

Subject to $g_j^L \leq g_j(A) \leq g_j^U \quad j = 1,2,...m$ (8)

and $A_i^{\min} \leq A_i \leq A_i^{\max} \quad i = 1,2,...n$ (9)

Where A_i = the design variable *i* (member *i* cross-sectional area), n = the number of the design variables, *W(A)* = the objective function (the structural weight), ρ = the material density, L_i = the member length, m = the number of inequality constraints (*g*), $A_i^{\min}$ and $A_i^{\max}$ are the lower and the upper bound of the i^{th} variable respectively. The lower and upper bounds posed by Eq.(8) on the constraints include truss member stresses and joint displacements.

E. *Example: Cantilever 10-Bar Planar Truss Structure*

The GADT is tested against a classical global optimization problem: the *Cantilever 10-Bar Planar Truss Structure* optimization problem. This 10-dimensional problem has been investigated by many researchers, and has been well-established as an optimization benchmark problem known for being challenging with unknown global and multiple local minima. A schematic of the cantilever 10-bar planar truss structure can be found in Fig.8. The assumed material density is 0.1 lb/in^3 (2767.990 kg/m^3), the length L is 360 in (914.4 cm) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The stress and deflection limitations on the members are ±25.0 ksi (172.375 MPa) and ± 2.00 in (5.08 cm), respectively. Cross-sectional areas are allowed to vary between 0.1 in^2 and 35 in^2 (0.6452 cm^2 and 225.806 cm^2). No member grouping is utilized, resulting in each member having a potentially unique cross-sectional area (A_1 to A_{10}). One loading case is studied when P_1 = 100 kips (444.8 kN) and P_2 = 0. Table 1 gives the best discovered optimum solutions along with the corresponding minimum weight benchmarked against optimal designs by other published studies. It should be noted that the best discovered solution was found after 10000 iterations and the initial population size, prior to trimming, is taken as 500 solutions. The optimal solutions found by the GADT meet all of the problem constraints and the comparisons in Tables 1 show that the GADT provides superior results. Fig.9 shows the convergence history to the optimum solution for GA with and without domain trimming for a population size of 500 going through 2000 iterations. Notice that convergence plateaus after around 800 iterations; at which point, the trimming has progressed such that it has little further effect on the GA signifying the elimination of most or all unfit values. Also, notice that in some subsequent iterations, the conventional GA identifies a higher minimum weight as the best discovered solution, as compared to a previous iteration.

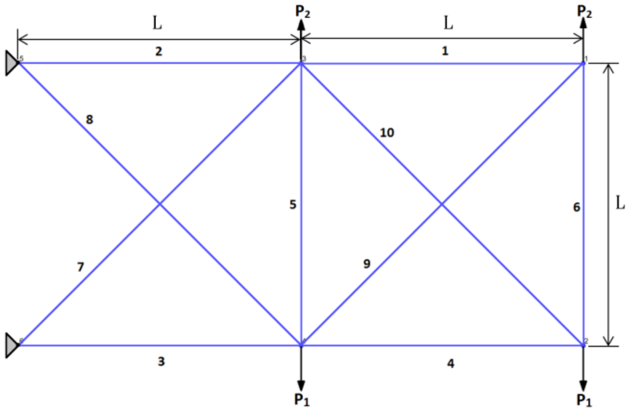


Fig.8: 10-Bar planar cantilever truss model

This primarily a result of the mutation process, which eventually is filtered out with enough iterations. This effect is not exhibited in presence of domain trimming since the mutation is unable to select the excluded/trimmed unfit values.

Since GA optimization is an evolutionary method, it relies on generating populations of solutions in order to find the optimum amongst them. To promote better discovery of optimum results, an adequate population size should be selected. Unfortunately, this parameter is problem-specific. The number of variables in a solution, diversity of values and initial domain selection are some of the factors determining the appropriate population size.

Table 2: Optimization results for the 10-bar planar truss

Member	Optimal cross-sectional areas								
Areas- cm^2 in^2	This work	[4]	[6]	[7]			[8]	[5]	[9]
A1	195.49	196.58	196.65	198.12	197.67	195.56	193.99	194.52	197.87
	30.30	30.47	30.48	30.71	30.64	30.31	30.07	30.15	30.67
A2	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.66	0.65
	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
A3	147.76	149.63	150.05	154.28	148.78	150.07	149.72	146.52	153.29
	22.90	23.19	23.26	23.91	23.06	23.26	23.21	22.71	23.76
A4	97.47	98.27	97.97	95.01	96.90	98.26	97.86	98.52	94.13
	15.11	15.23	15.19	14.73	15.02	15.23	15.17	15.27	14.59
A5	0.65	0.65	0.65	0.65	0.65	0.65	0.65	0.66	0.65
	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
A6	3.87	3.37	3.53	0.65	3.81	3.55	3.46	3.51	0.65
	0.59	0.52	0.55	0.10	0.59	0.55	0.54	0.54	0.10
A7	137.84	135.60	48.08	55.14	48.33	48.26	48.14	139.10	55.34
	21.37	21.02	7.45	8.55	7.49	7.48	7.46	21.56	8.58
A8	48.42	48.42	13.20	135.55	136.18	139.43	136.95	48.65	135.94
	7.51	7.45	2.05	21.01	21.11	20.92	21.23	7.54	21.07
A9	0.65	0.65	138.99	134.26	139.13	139.43	139.55	0.65	135.23
	0.10	0.10	21.54	20.81	21.57	21.61	21.63	0.10	20.96
A10	137.80	139.26	0.65	0.65	0.65	0.65	0.65	138.39	0.65
	21.36	21.59	0.10	0.10	0.10	0.10	0.10	21.45	0.10
Weight kN	22.46	22.51	22.51	22.58	22.51	22.50	22.51	22.50	22.58
lb	5049.83	5060.85	5060.87	5076.68	5061.40	5055.00	5057.37	5057.88	5076.85
σ_{max} Mpa	172.37	172.37	172.37	140.53	172.11	172.37	172.37	172.37	140.30
ksi	25.0	25.0	25.0	20.4	25.0	25.0	25.0	25.0	20.3
Δ_{max} cm	5.08	5.08	5.079995	5.079999	5.079925	5.08	5.08	5.08	5.0797
in	2 .0	2 .0	1.999998	1.999999	1.99997	2 .0	2 .0	2 .0	1.99989

III. Summary and Conclusions

A modified version of Genetic Algorithm with Domain Trimming (GADT) is developed and presented in this study. The innovative technique is used to solve a least-weight optimization problem in the design of truss structures. Then, the GADT is tested against a well-established challenging benchmark problem; the Cantilever 10-Bar Planar Truss Structure optimization problem.

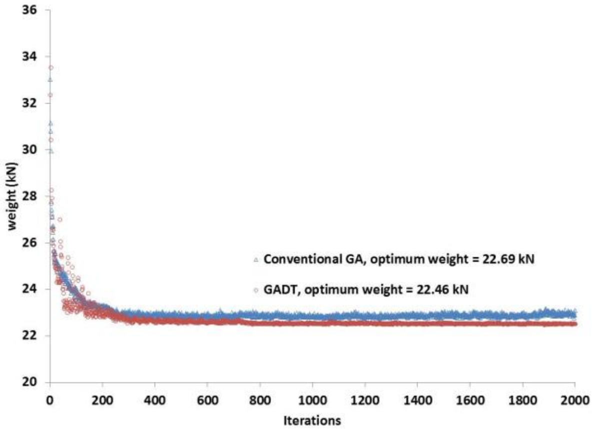


Fig.10: Path to optimum solution for the 10-bar truss by GA and GADT.

This benchmark problem is a 10-dimensional optimization problem with unknown local and global minima. The GADT handles the problem-specified constraints using ‘non-stationary penalty functions’ method. The results show that the (GADT) method is efficient in finding the best discovered optimal solution. The optimal solutions found by the GADT meet all of the problem constraints and the comparisons with the published literature show that the GADT provides superior results.

References

[1] Balling RJ, Briggs RR, Gillman K. Multiple Optimum Size/Shape/Topology Designs for Skeletal Structures Using a Genetic Algorithm. J Struct Eng. American Society of Civil Engineers; 2006; 132(7):1158–65.

[2] Perez RE, Behdinan K. Particle swarm approach for structural design optimization. Comput Struct. Pergamon Press, Inc.; 2007; 85(19-20):1579–88.

[3] Luh G-C, Lin C-Y. Structural topology optimization using ant colony optimization algorithm. Appl Soft Comput. Elsevier Science Publishers B. V.; 2009; 9(4):1343–53.

[4] Barakat, Samer A. "Shuffled complex evolution optimizer for truss structure optimization." Computing in Civil and Building Engineering, Proceedings of the International Conference. Vol. 30. 2010.

[5] Lee, Kang Seok, and Zong Woo Geem. "A new structural optimization method based on the harmony search algorithm." Computers & Structures 82.9, 781-798, 2004.

[6] Kaveh, A., and S. Malakouti Rad. "Hybrid genetic algorithm and particle swarm optimization for the force method-based simultaneous analysis and design."Iranian Journal of Science and Technology, Transaction B: Engineering 34.B1, 15-34, 2010.

[7] Csébfalvi, Anikó. "A hybrid meta-heuristic method for continuous engineering optimization." Civil Engineering 53.2, 93-100, 2009.

[8] Kaveh, A., and S. Talatahari. "A hybrid particle swarm and ant colony optimization for design of truss structures." Asian Journal of Civil Engineering 9.4, 329-348, 2008.

[9] Schmit Jr LA, Miura H, "Approximation concepts for efficient structural synthesis," NASA CR-2552, Washington, DC: NASA, 1976.