DESIGN OPTIMIZATION OF N-SHAPED ROOF TRUSSES

Karim Hamza
Ph.D. Pre-Candidate
University of Michigan
Ann Arbor, MI 48109-2102
khamza@engin.umich.edu

Haitham Mahmoud
Ph.D. Pre-Candidate
University of Michigan
Ann Arbor, MI 48109-2102
ham@engin.umich.edu

Kazuhiro Saitou
Assistant Professor
University of Michigan
Ann Arbor, MI 48109-2102
kazu@engin.umich.edu

Abstract

Design optimization of a class of plane trusses called the N-Shaped Truss (NST) is addressed. The parametric model of NST presented is intended for real-world application, avoiding simplifications of the design details that compromise the applicability. The model, which includes twenty-seven discrete variables concerning topology, configuration and sizing of the truss, presents a challenging optimization problem. Aspects of such challenge include large search space dimensionality, absence of a closed-form objective function and constraints, multi-modal objective function and costly CPU time per objective function evaluation. Three implementations of general-purpose genetic algorithms (GA) are tested for this problem, along with a version of taboo search called reactive taboo search (RTS). The RTS exhibited better performance than the tested versions of GA. Performance study of the algorithms provides some good insight to some weaknesses in GA and RTS as well as future prospective combination of them to gain better performance.

1 INTRODUCTION

Truss structure optimization is a problem that is attractive due to its direct applicability in design of structures. Optimization of trusses can be classified into three main categories i) sizing, ii) configuration and iii) topology. This classification is slightly different from that of continuum structures, given in (Chapman *et. al.*, 1993) In the sizing optimization, cross-sectional areas of members in the truss are design variables and the coordinates of the nodes and connectivity are held constant (Goldberg 1986). The sizing problem is made even more interesting and practical through restricting the choice of truss members to a discrete set of available standard cross-sections (Rajeev and Krishnamoorthy, 1992). In configuration optimization, the member cross-sections and connectivity (*i.e*. topology) remain constant, but the nodal position locations are the design variables. In topology optimization, the connectivity is the objective of the optimization (Bendose and Kikuchi 1988) and (Jakiela *et. al.*, 2000). Combining the categories has also been performed. Gil and Andreu (2001) combined the configuration and sizing problems. Deb and Gulati (2001) combined topology and sizing through real coded genetic algorithms. A fully connected ground structure is taken as a start, then during optimization, members having close to zero cross-sectional areas are then deleted.

Optimization methods applied included gradient-based methods such as the work of (Taylor and Rossow 1976) and (Kirsch 1979), simulated annealing (Moh and Chiang 2000) and genetic algorithms. Analytical methods have generally been limited by approximations due to the complexity of the real-world problem, which is nonlinear and often has no closed form objective function and constrains.

To the best of the authors' knowledge, most previous work was directed to developing optimization models for general trusses rather on a "high-level," without going deep into the design details of the truss. In this paper, a particular class of plane trusses (N-Shaped) is considered. While restricted to that class of trusses, the parametric model formulated goes deep into the design details and combines all truss optimization categories of sizing, configuration and topology. The optimization problem has a large search space which makes direct exhaustive search methods totally impractical. In addition, structural optimization problems are known to have many local optima, which encourages the use of heuristic global optimizers. Three implementations of genetic algorithms (GA) are tested as well as reactive taboo search (RTS) which also seems to be an attractive global optimizer (Battiti and Tecchiolli 1994).

The paper starts with a review of truss optimization then proceeds to describe the parametric model of the N-shaped truss. Following the description of the parametric model, the implemented GA and RTS are presented, then an actual real-life truss is used as a bench-mark problem to compare the performance of the optimizers. Results and discussion are then presented.

2 PARAMETRIC MODEL OF NST

2.1 TERMINOLOGY

Some of the terminology used in practice for the design of trusses is to be used in this paper. The following is a quick summery of such terminology:

- An N-Shaped Truss (NST): is a plane truss (Fig.1) that has a certain general shape resembling the letter "N."
- Upper Chord: are all the inclined members on the top part of the truss (Fig. 2). All upper chord members of an N- Shaped Truss form one straight line.
- Lower Chord: are all the horizontal members on the lower part of the truss (Fig. 2). All lower chord members of an N- Shaped Truss form one horizontal straight line.
- Vertical Members: are (as the name suggests), the vertical members in the truss (Fig. 2).
- Diagonal Members: are those internal inclined members (Fig. 2).
- Truss Projection: is the distance the truss protrudes after the centerline of the carrying column (Fig. 2).
- Bays: Are the spans between the trusses in the top view (Fig. 2).
- End Bay: is a last bay in a building.
- Purlins: are light members positioned across the bays and are carried on top of the upper chord (Figs. 2-3). Purlins, in turn carry the roof cladding.
- Roof braces: are X-shaped sets of members (Fig. 2) that are present in some bays in order to increase the overall structure stiffness.
- Longitudinal Braces: are sets of members across the bays that are included to increase the overall rigidity of the structure (Figs. 2-3).

Figure 1: Photo of Actual N-Shaped Truss

2.2 DESIGN VARIABLES

Twenty seven variables that a designer can modify are used as design variables in this parametric model. The design variables are categorized into i) variables concerned with topology and configuration and ii) variables concerned with sizing of the truss members. The design variables are given as:

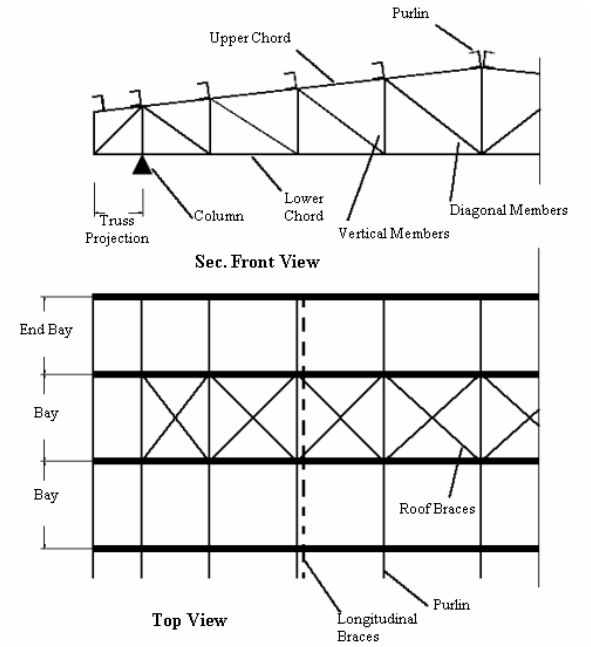


Figure 2: Typical N-Shaped Truss

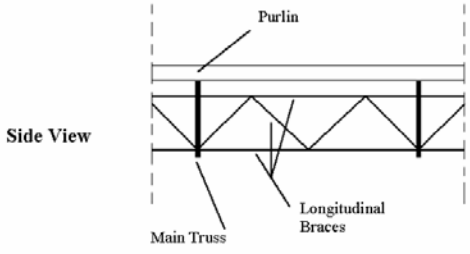


Figure 3: Longitudinal Braces

Topology and Configuration Variables:

X1: is an integer number that defines the selected roof layout plan (from up to 5 user-defined choices), this variable subsequently sets the number of main bays, bays' lengths and end-bays' lengths.

X2: is the length of the vertical member directly on top of the support. Normally, this variable is continuous, but it is discretized in this model to avoid the necessity of using mixed integer/continuous optimizers. However, discretization doesn't impose much deviation from practicality, since the fabrication often favors "rounded-off" and similar dimensions.

X3: is the number of purlins on top of the truss. This normally dictates the general truss topology, since every purlin must have a vertical member in the truss underneath it. The space between two purlins (or their two verticals) will be referred to as a truss "cell".

X4: is the number of sub-divided truss cells near the support. Subdividing the cells near the support (the portion which has low truss depth as opposed to the middle part of the truss), generally improves the angle of the diagonal members which in turns gives better distribution of the axial forces in the members.

X5: is the number of truss cells which have reinforced diagonal members. Normally, the diagonals closer to the support are subjected to higher axial loads, therefore it is often efficient to choose a different cross section for the first one or few diagonal members.

X6: is the number of merged cells near the middle of the truss. The purpose of merging cells at the portion of bigger truss depth is also to improve the angle of the diagonals to give better stress distribution.

X7: is the number of verticals that are nearer to the column and are taking a different cross-section than the rest of the verticals.

The model also allows for two different configurations of longitudinal braces to be used, thus the longitudinal braces passing near the mid-span (with higher depth) may be different from those passing above the support.

X8: is the total number of longitudinal braces lines across the roof.

X9: is half the number of longitudinal braces lines close to the support (Type-1).

X10: is the number of nodes (equal to number of cells minus one) on Type-1 Longitudinal Braces.

X11: is the number of nodes on Type-2 Longitudinal Braces.

Member Sizing Variables:

X12 to X27 are integer variables defining the selected standard cross-section from the available database for 16 groups of truss members. The truss member groups are: purlins, main truss upper chord, lower chord, 3 groups of verticals, 4 groups of diagonals, longitudinal braces 2 groups of chords, 2 groups of verticals and 2 groups of diagonals.

It should be noted that the truss members grouping employed in this parametric model keeps the number of design variables fixed, but the number of truss members is variable. Also, all variables being integer allows for pure-integer GA and RTS to be used in optimization, without the loss of practicality of the model.

2.3 CONSTRAINTS

Constraint evaluation is the main costly event in terms of CPU time. It involves generating a finite element (FE) mesh of the truss, solving the FE model for different load cases then performing safety check on truss members. The safety constraints involve:

- Load Cases include Dead load, Live Load and Wind Load. Load Case Combinations are Dead Load + Live Load (DL), Dead Load + Wind Load (DW) and Dead Load + Live Load + Wind Load (DLW)
- Mild steel members subjected to tension must not exceed allowed under any of the load case combinations.
- Members subjected to compression must not exceed allowed compressive stress under any of the load case combinations. Allowed compressive stress depends on member slenderness.
- Bending stresses in Purlins must be safe under all load cases and also capable of carrying a specified concentrated load in its mid-span.
- Depth of the beam cross-section chosen for Purlins should not be less than a certain portion of its length.
- Deflection under live load is not to exceed a certain amount.
- Slenderness of all members subjected to compression is not to exceed a certain value.
- Slenderness of any member is not to exceed a certain value.
- Purlin spacing should be within a certain range.
- Diagonal members angle from horizontal should be within a certain range.

Constraints are enforced through adaptive penalty (Chen 2001). To ensure that the search converges to a feasible design, additional cost is added to the objective function to make the cost of any infeasible design more than that of the current best feasible design. The penalty cost also depends upon the amount of violation. Typically, the penalty cost is high at the beginning of the search and is then gradually lowered as better feasible designs are found. A crucial matter for efficient employment of adaptive penalty is to have a feasible initial design.

2.4 OBJECTIVE FUNCTION

In many applied cases, truss optimization is a multi-objective process regarding issues such as weight, cost, stiffness and natural frequencies. However, the particular class of trusses considered finds its main domain of application in industrial and commercial clear-span buildings. For such applications, there is usually the single objective of minimizing the overall cost. In many practical cases, the overall cost is directly associated with the total steel weight. Thus for the current study, the objective is to minimize the overall weight. Such weight includes the main truss members, longitudinal bracing members, purlins and estimates of all connection plates (by empirical formulas in terms of other truss parameters).

The objective function (OF) combines the truss total weight plus a penalty term to prevent constraints violation. There are two cases for the objective function:

- **No constraints are violated**

In this case: $\mathrm{OF} = W_{\mathrm{t}}$

- **One or some of the constraints are violated**

In this case: $\mathrm{OF} = \max(W_{\mathrm{t}}, W_{\mathrm{b}}) \times C_{\mathrm{Pen}} \times I_{\mathrm{Pen}}$

Where:

W_t : is the total weight of the considered structure

W_b : is the total weight of the best feasible structure encountered so far during the optimization

C_{Pen} : is a penalty constant

I_{Pen} : is the number of truss members that violate the safety constraints

A key implemented feature is the adaptive penalty which aims at preventing "over-penalizing" the infeasible designs while making sure that no infeasible design has a better OF value than the best feasible encountered design.

3 GENETIC ALGORITHM

3.1 GENERAL PURPOSE GA

The general purpose genetic algorithm (GA_1) tested in this paper implements variable storage as integer variables, 4 crossover operators, 12 mutation operators, fitness scaling, population distribution, roulette wheel selection along with elitist selection.

Integer Storage: For efficiency of storage, variables are stored directly as integers rather than binary strings (Goldberg 1989) and are translated to their equivalent binary strings when need during crossover and mutation.

Crossover Operators:

- Binary string crossover.
- Inner Crossover (adopted from real coded GA). The new variable values are computed as:

$ChildVal_1 = Round\ (\alpha\ ParentVal_1 + (1 - \alpha)\ ParentVal_2)$

$ChildVal_2 = Round\ (\ (1 - \alpha)\ ParentVal_1 + \alpha\ ParentVal_2)$

Where α is a randomly generated number between 0 and 1

- Outer Crossover (adopted from real coded GA). The new variable value is computed as:

$ChildVal = Round\ (StrongerParentVal + \alpha\ (StrongerParentVal - WeakerParentVal)\ )$

- Uniform crossover (Liang-Jie et al., 1995). In which the variables are unchanged, but exchanged between the parents with a 50% probability of exchange.

Mutation Operators:

- Binary bit flipping.
- Binary bit shift left.
- Binary bit shift right.
- Binary bit inversion.
- Shifting value to nearest boundary.
- New random number generation.

Another similar set of mutation operators is also used that only act if the member fitness is below average.

An overall probability for crossover and mutation is specified for a search. For each mutation or crossover operation of mating members, selection of which operator to use is performed randomly according to an assigned probability of use for each operator.

Fitness Scaling: linear fitness scaling is implemented to give a fair survival chance for strong population members.

Speciation: members further away from population average get a fitness bonus to encourage diversification.

Roulette Wheel Selection: is used for selecting members of old population for mating and producing new members of next population.

Elitist Selection: one copy of best member in a population passes unchanged to the next population to ensure that any optimized value is no worse than the best previously attained. And the rest of the new population is filled by the traditional selection, crossover and mutation.

Seeding: one feasible point is included in the initial population and rest of the population is chosen randomly. Due to the nature of the problem, a purely random initial population may end up with a population of all-infeasible designs. Such an initial population will cause failure of the adaptive penalty strategy, as it requires knowing the OF value of some feasible design.

3.2 GA WITH CACHING

The second implementation of GA tested in this paper (GA_2) is the same as GA_1, but all evaluations of objective function are stored. Thus, when performing population members OF evaluation, only un-explored regions of the search space will require the FE solution of the truss.

By nature, OF caching is inherent in RTS and is one of the strong points in favor of it. Therefore history storage is implemented into GA in order to even up the advantage RTS has and allow for a better comparison.

3.3 GA WITH NORMALLY DISTRIBUTED INITIAL POPULATION

RTS benefits from a good starting point, so an interesting study would be to have a biased initial population. Thus, the third implementation of GA (GA_3) is the same as GA_2, but has its all members of the initial population normally distributed about the initial feasible design.

4 REACTIVE TABOO SEARCH

4.1 GENERAL SCHEME

Reactive taboo search is a heuristic global optimization technique that has less stochastic content than genetic algorithm. In fact, save for a small portion of the algorithm, it is almost completely deterministic. The basic idea in taboo search (Glover 1986, 1989, 1990) is to make use of previously evaluated points within the search space

to direct the future sampling and prevent entrapment at a local minimum by applying taboo conditions. Reactive taboo search (Battiti and Tecchiolli 1994) proposes a scheme for adaptively varying the way the taboo conditions are applied based on the objective function history, thus the search "reacts" to the objective function behavior. Pseudo-code of RTS is given as:

```
1   Begin at a starting point
2   Examine Non-Tabooed Neighboring Points
    and move to the best of them
3   If new point has been not been visited before
4           Goto 2
5   Else If cycling is not "excessive"
6           Put a taboo condition upon point
7           Goto 2
8       Else perform "quick escape" and Goto 2
```

The single starting point in the search space is set as the "current point". RTS then evaluates the entire neighborhood of the current point and moves to the best point in it which then becomes the new current point. An important feature in RTS, is that all the previously evaluated points are stored in the memory, this leads to lots of savings in computational time when evaluating the neighborhood of the new point. Memorizing all evaluated points is costly in terms of required storage resources since the total memory required for the algorithm grows linearly as more points are being evaluated, however, such memorizing saves a lot of computational time if the OF is costly in terms of CPU evaluation time.

At the start of the search RTS, simply behaves like a steepest descent search until it hits a local minimum. Whereas steepest descent stops upon reaching a local minimum, RTS continues to search the neighborhood of the current point and move to best point within it even if it is worse than the current point. To prevent infinite cycling back and forth around a local minimum, TS imposes a taboo condition upon the last visited point, that is, "a previously visited point cannot be visited again until a certain number of iterations is completed", and such number of iterations is typically referred to as the "taboo list length".

In RTS, the taboo list length is adaptively changed according to the search behavior within a minimum and a maximum value. If the search still gets stuck in a large basin of attraction of the objective function, which the maximum taboo list length is not enough to overcome, a "quick escape" is performed.

The search is typically stopped after performing a specified number of moves or objective function evaluations. The best point encountered is returned.

4.2 NEIGHBORHOOD EVALUATION

RTS performs a complete neighborhood evaluation. Unlike the version of RTS proposed by Battiti and Tecchiolli (1994) where all variables were either zero or one, the implemented version in this paper uses integer values for the variables. The neighborhood is defined as the set of points that have all their variables equal to those of the current point except for one variable, which is different by a value of ±1. Thus, the number of points in the neighborhood is twice the number of variables (or less for points touching the upper and lower limits of the variable ranges).

4.3 RTS REACTION TO SEARCH BEHAVIOR

At each move (iteration), RTS places a taboo condition on the previous point, the taboo condition lasts a number of iterations equal to the current taboo list length. RTS also keeps track of when was each point visited, and the number of visits. If a point is visited twice, the taboo list length is increased. Thus, near a local minimum, the taboo list length keeps increasing until it is enough to explore regions further away. If a number of iterations pass without any cycles occurring (visiting the same point several times), the taboo list length is decreased.

Typically, a maximum taboo list length is specified. It is generally not beneficial to have the maximum taboo list length greater than the number of points in the neighborhood, because it can lead to a situation when all the points in the neighborhood are tabooed. When such a situation arises, the taboo conditions are relaxed, and the new current point is chosen as the last visited point in the neighborhood.

Sometimes if a large basin of attraction exists in the objective function, there could be a situation when taboo conditions are not enough to overcome the domain of the local minimum and that is when the "quick escape" is performed.

4.4 QUICK ESCAPE MECHANISM

RTS keeps a record of the average cycle length. When it approaches the maximum taboo list length, this indicates that tabooing is not enough to overcome the current basin of attraction, and quick escape is necessary. Quick escape is performed by randomly changing the values of some of the variables of the current point. It is simply just like re-starting the search at new starting point that is not entirely random.

5 APPLICATION

5.1 TRUSS DATA

Data of a real N-shaped truss is used as a starting point for the optimization algorithms. The truss data is given in Table 1.

Table 1 Truss Data

Number of Main Bays	2
Building Clear Span	21.0 m
Material Young's Modulus	207 GPa
Allowed Stress	140 MPa
Max. Slenderness (Compression Members)	180
Max. Slenderness (All Members)	300
Max. Deflection under live load	1/300 of Span
Live Load	50 kg/m^2
Wind Pressure	50 kg/m^2
Dead Load	Weight + 20 kg/m^2
Available Database Contains L-sections (LPN), C-sections (UPN & C.F.) and I-sections (IPN & IPE)	

A photo of the actual truss during erection procedure is given in Fig. 1. This design (topology, configuration and sizing) is used as the starting point for optimization. Topology and configuration are shown in Fig. 4. Truss member cross-sections are given in Table 2.

5.2 GA PARAMETERS

Among the several available tuning options for the implemented GA, the following settings are chosen:

- Population Size: 100, 150, 200 and 250
- Number of Generations: (Unlimited), search stops when maximum number of objective function evaluations is reached.
- Max. number of OF evaluations: (Tested Several)
- Overall crossover probability: 0.9
- Equal probability for different crossover operators
- Overall mutation probability: 0.25
- Equal probability for different operators
- Fitness scaling constant: 1.6

Choice of the search parameters was based on practical published values and the available computational resources. Further tuning is possible.

5.3 RTS PARAMETERS

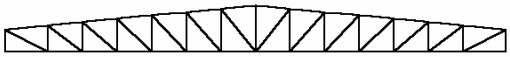
RTS has less tuning parameters than GA. The following settings are chosen:

- Number of moves: (Unlimited), search stops when maximum number of objective function evaluations is reached.
- Max. number of OF evaluations: (Tested Several)

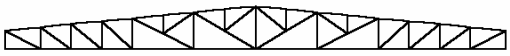
5.4 RESULTS

Each of the design variables concerned with truss member sizing has 48 possible choice options, variables concerning configuration and topology range between 3 to 20 options. The total search space (all possible combinations of variables) is 1.58814×10^{37}. Practicality limits for reasonable CPU time made it preferable to limit the comparison of optimization algorithms to 10,000 OF evaluations. Some reasonably good results are obtained even though 10,000 OF evaluations comprise only 6.3×10^{-34} of the total search space.

Topology and configuration of the initial design, an intermediate design during optimization and final best obtained design are shown in Fig. 4. A listing of the chosen cross-sections for truss member groups and overall design weight is given in Table 2. The intermediate design is shown as a demonstration of topology change as well as sizing.



a) Initial Design, Weight = 9903.2 kg



b) Intermediate Design, Weight = 6328.8 kg

c) Best Obtained Design, Weight = 5655.3 kg

Figure 4: Truss Topology and Configuration

Table 2 Chosen Truss Member Groups Cross-sections

Variable	**Designs**		
	Initial	**Intermediate**	**Final Best**
X12	C.F. C140x4	C.F. C140x3	C.F. C140x3
X13	2xLPN 70x7	2xLPN 70x7	2xUPN 65
X14	2xLPN 70x7	2xLPN 70x7	2xIPN 80
X15	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X16	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X17	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X18	2xLPN 50x5	2xLPN 50x5	2xIPN 80
X19	2xLPN 50x5	2xLPN 50x5	2xLPN 40x4
X20	2xLPN 50x5	2xLPN 50x5	2xLPN 40x4
X21	2xLPN 50x5	2xLPN 50x5	2xLPN 40x4
X22	2xLPN 70x7	2xLPN 70x7	2xLPN 50x5
X23	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X24	2xLPN 60x6	2xLPN 60x6	2xLPN 30x3
X25	2xLPN 60x6	2xLPN 60x6	2xLPN 50x5
X26	2xLPN 50x5	2xLPN 50x5	2xLPN 30x3
X27	2xLPN 50x5	2xLPN 50x5	2xLPN 30x3
Truss Weight	**9903.2 kg**	**6328.8 kg**	**5655.3 kg**

Table 3 Optimization Results

# of OF Eval.	Objective Function Value							Standard Deviation		
	RTS	Avg. of 20 Runs			Best of 20 Runs					
		GA_1	GA_2	GA_3	GA_1	GA_2	GA_3	GA_1	GA_2	GA_3
500	7781	9518	9487	9034	8197	7703	7534	587	574	474
1000	6687	9346	9209	8825	7599	7656	7534	666	725	443
1500	6491	9216	8973	8775	7599	7656	7534	673	687	432
2000	6430	9088	8929	8759	7599	7656	7534	697	691	432
2500	6430	8987	8929	8706	7599	7656	7534	690	691	452
3000	6430	8866	8840	8658	7599	7656	7259	619	673	545
4000	6430	8754	8616	8538	7270	7236	7259	665	734	555
5000	6430	8664	8591	8463	7270	7236	7259	594	731	569
6000	6430	8513	8293	8427	7270	7236	7259	604	640	602
7000	6430	8436	8226	8358	7270	7194	7259	571	633	594
8000	5704	8396	8115	8255	7174	7034	7259	581	639	554
9000	5655	8263	7998	8150	7174	7034	7259	496	534	523
10000	5655	8213	7935	7969	7174	7034	7259	479	547	479

Since RTS has very little stochastic content compared with GA, only one optimization run is used as a representative of RTS. Twenty runs are performed for each of GA_1, GA_2 and GA_3 using four different population sizes (five runs for each population size). The results of optimization performance are summarized in Table 3 and plotted in Figures 5 – 6.

The results in Table 2 and Fig. 5 are for the number of *new* objective function evaluations, thus caching in GA_2 and GA_3 resulted in improvement of the performance over the traditional GA_1. Furthermore having the initial population normally distributed about the starting point in GA_3 improves the consistency of the search (as seen in the standard deviation of the 20 runs) and results in a quicker descent of the objective function at the start of search. GA_3 however has little or no advantage over GA_2 towards the end of the search.

Further examination of Figs 5 – 6 and Table 2 shows an appreciably better performance of RTS over GA. To analyze possible reasons for RTS being better suited for the examined optimization problem than the implemented forms of GA.

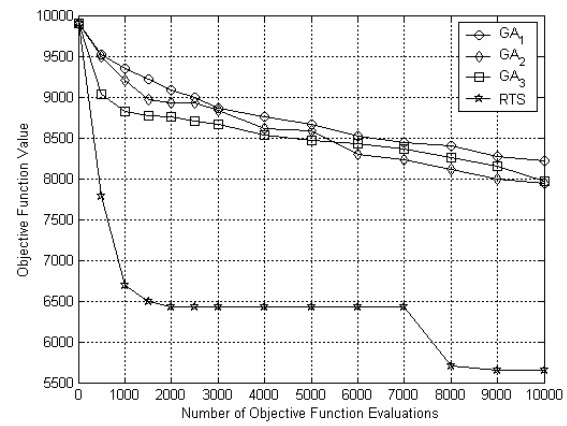


Figure 5 Optimization Progress – Average of GA Runs

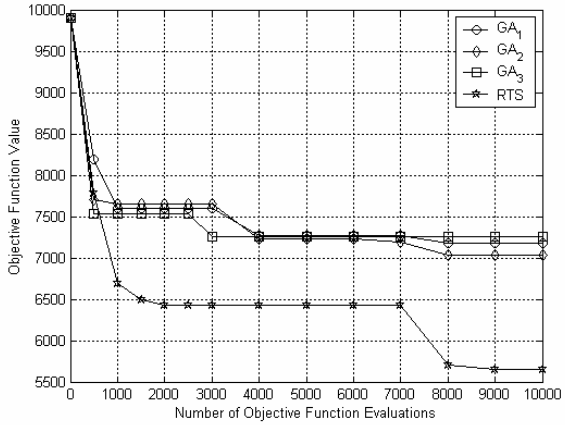


Figure 6: Optimization Progress – Best of GA Runs

6 DISCUSSION

GA relies on having several points that are distributed over the search space (population) to achieve diversification. According to the schemata theory (Goldberg 1989), selection along with crossover provides intensification by attracting the population points to zones of higher fitness. Eventually the whole population gets attracted to the global optimum. In general, the intensification properties of GA are not as good as those of local optimizers (Erbatur and Hasancebi 2001). Mutation is generally used to increase diversification, especially when the whole population gets too closely attracted to a certain region.

The main weakness GA suffers when the problem has large dimensionality is that a moderate population size (100 to 200 members) becomes insufficient to achieve enough diversification over the search space and insufficient schemata pool, which also confounds the intensification. Increasing the population size beyond certain limits is on the other hand very costly in terms of the number of objective function evaluations.

Another problem that GA encounters is due to the complexity of the constraints which makes GA unable to converge without seeding with an initial feasible point. Seeding itself decreases the GA efficiency.

RTS has separate mechanisms for intensification and diversification. For intensification, RTS relies on a local optimizer that nails down the local optimum. Thus, finding the local optimum is fast, efficient and has less sensitivity to large dimensionality than GA. This accounts for the fast descent of the OF value encountered at the beginning of the RTS search in Figs. 5 – 6. Upon reaching a local optimum, RTS switches to diversification by imposing taboo conditions to prevent moving to already explored points. If the taboo conditions are not enough to escape a large basin of attraction, RTS performs its quick escape move and "hopes" it will be enough to escape the current basin of attraction. It can be seen in Figs. 5 – 6 as well as Table 3 that after the good start, RST remained incapable of finding any better designs for a long period.

Given N number of objective function evaluations, the memory requirement is constant for the traditional GA (GA_1), but of order N for RTS, GA_2 and GA_3 because of caching. Caching also incurrs additional computational effort of order less than N^2 but such computational effort has little overall effect when the OF is costly to evaluate.

It is seen in this study that RTS has better capabilities for intensification as well as exploiting a good starting point while GA has better diversification. Future research aspects may include combining both to get even better. One such possibility would be to use RTS, but perform large OF attraction basin detection, once the quick escape mechanism becomes inefficient, the search may be switched to a population-based search until a new basin of attraction is found, then switch back to RTS.

7 CONCLUSIONS

Design optimization of a real-world class of plane trusses is considered. A parametric model of the truss is developed, which takes into account most of the practical aspects for design applicability. Optimization of the model is pretty challenging since it involves sizing, configuration and topology, large dimensionality and costly objective function. Three implementations of general purpose GA as well as RTS are tested to see if they can come up with better designs than an actual erected design. Through a number of objective function evaluations that is only a very small fraction of the total search space, both GA and RTS succeeded in coming up with better designs. Although RTS performed better, observation reveals that RTS has better intensification, while GA has better diversification. This motivates future work for combining aspects of GA and RTS.

Acknowledgments

This work is an extension of a course project of ME558 Discrete Design Optimization, offered in Fall 2001 at the University of Michigan, Ann Arbor. MECO, Modern Egyptian Contracting provided the data of the previously erected truss, used as starting point in this paper.

References

R. Battiti and G. Tecchiolli (1994), "The Reactive Tabu Search," ORSA Journal on Computing, V 6, pp. 126-140.

M.P. Bendose and N. Kikuchi (1988), "Generating Optimal Topologies in Structural Design using a Homogenization Method," Computer Methods in Applied Mechanics and Engineering, V 71, pp. 197-224.

C. Chapman, K. Saitou and M. Jakiela (1993), "Genetic Algorithms as an Approach to Configuration and Topology Design," Advances in Design Automation, V 65, pp. 485-498.

S. Y. Chen (2001), "An approach for impact structure optimization using the robust genetic algorithm," Finite Elements in Analysis and Design, V 37, pp. 431-446.

K. Deb and S. Gulati (2001), "Design of truss-structures for minimum weight using genetic algorithms," Finite Elements in Analysis and Design, V 37, pp. 447-465.

F. Erbatur and O. Hasancebi (2001), "Layout optimization using GAs and SA," Optimal Structural Design Workshop, GECCO-2001, pp. 102-107.

L. Gil and A. Andreu (2001), "Shape and cross-section optimization of a truss structure," Computers and Structures, V 79, pp. 681-689.

F. Glover (1986), "Future Paths for Intege Programming and Links to Artificial Intelligence," Computers and Operations Research, V 13, No. 5, pp. 533-549.

F. Glover (1989), "Tabu Search – Part I," ORSA Journal on Computing, V 1, pp. 190-206.

F. Glover (1990), "Tabu Search – Part II," ORSA Journal on Computing, V 1, pp. 4-32.

D. Goldberg and M. Samtani (1986), "Engineering Optimization via Genetic Algorithms," Proceeding of the 9th Conf. on Electronic Computations, ASCE, Birmingham, pp. 471-482.

D. Goldberg (1989), "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley.

M. Jakiela, C. Chapman, J. Duda, A. Adewuya, and K. Saitou (2000), "Continuum structural topology design with genetic algorithms," Computer Methods in Applied Mechanics and Engineering, V 186, No. 2, p 339—356.

U. Kirsch (1979), "Optimal Design of Trusses by Approximate Compatibility," Computers and Structures, V 12, pp. 93-98.

Z. Liang-Jie, M. Zhi-Hong and L. Yan-Da (1995), "Mathematical analysis of crossover operator in genetic algorithms and its improved strategy," Proceedings of the IEEE Conference on Evolutionary Computation, V 1, pp. 412-417.

J. Moh and D. Chiang (2000), "Improved Simulated Annealing Search for Structural optimization," AIAA Journal, V 38, pp. 1965-1973.

S. Rajeev and C.S. Krishnamoorthy (1992), "Discrete Optimization of Structures using Genetic Algorithms," Journal of Structural Engineering, V 118, No. 5, pp. 1233-1250.

J.E. Taylor and M.P. Rossow (1976), "An Optimal Structural Design using Optimality Criteria," Advances in Engineering Science, 13th Annaul Meeting, Hampton, VA, pp. 521-530.