Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Applied Soft Computing 3 (2003) 221–235

Applied Soft Computing

www.elsevier.com/locate/asoc

# Design optimization of N-shaped roof trusses using reactive taboo search

Karim Hamza, Haitham Mahmoud, Kazuhiro Saitou*

*Department of Mechanical Engineering, University of Michigan, 3211 EECS, 2350 Hayward St., Ann Arbor, MI 48109-2102, USA*

Received 22 May 2002; accepted 15 May 2003

**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 27 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 (OF) and constraints, multimodal objective function and costly CPU time per objective function evaluation. Three implementations of general-purpose genetic algorithms (GAs) are tested for this problem, along with a version of taboo search called reactive taboo search (RTS). In this study, the raw version of RTS exhibited better performance than the tested versions of GA but lacks some of the GA capabilities to span the search space. A modification of RTS that uses a population-based exploitation of the search history is proposed. The optimization results show that the introduced modification can further improve the performance of RTS.


*Keywords:* Truss optimization; Reactive taboo search; Heuristic global optimization

## 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 [3]. 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 [11]. 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 [19]. 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 [2,14]. Combining the categories has also been performed. Gil and Andreu [7] combined the configuration and sizing problems. Deb and Gulati [5] combined topology and sizing through real-coded genetic algorithms (GA). A fully connected ground structure is taken as a start, then during optimization, members having close to zero cross-sectional areas are then deleted.

* Corresponding author. Tel.: +1-734-763-0036; fax: +1-734-647-3170.
*E-mail addresses:* khamza@engin.umich.edu (K. Hamza), ham@engin.umich.edu (H. Mahmoud), kazu@engin.umich.edu (K. Saitou).

1568-4946/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1568-4946(03)00036-X

Optimization methods applied for the truss optimization problem included gradient-based methods such as the research work of Taylor and Rossow [20] and Kirsch [15], simulated annealing by Moh and Chiang [18] in addition to genetic algorithms [3–6,14]. Analytical methods have generally been limited by approximations that are always introduced due to the complexity of the real-world problem that is non-linear and often has no closed-form objective function (OF) or constrains.

To the best of the authors' knowledge, most of the 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 are tested as well as reactive taboo search (RTS), which also seems to be an attractive global optimizer [1]. A previous study by the authors [13] revealed a superior performance of RTS, which although has less capability to span the search space than GA, is capable of efficiently nailing down the local optima then escaping them to find news ones. This study proposes a modification of the escape mechanisms of RTS to further increase its effectiveness at finding new better optima. The obtained results show improved performance when implementing the proposed modifications. 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. A simple example demonstrating the proposed modification of the escape mechanism of RTS is presented then an actual real-life truss is used as a benchmark problem to compare the performance of the optimizers.

## 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*): This is a plane truss (Fig. 1) that has a certain general shape resembling the letter "N".
- *Upper chord*: These 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*: These 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*: These are (as the name suggests) the vertical members in the truss (Fig. 2).
- *Diagonal members*: These are those internal inclined members (Fig. 2).

Fig. 1. Photo of actual N-shaped truss.

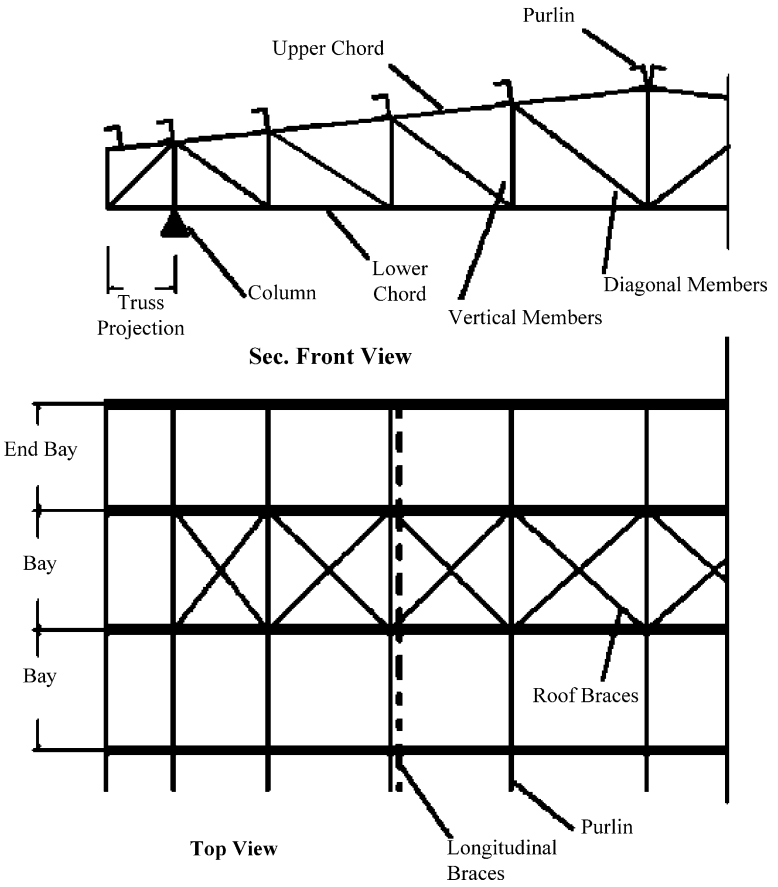



Fig. 2. A typical N-shaped truss.

- *Truss projection*: This is the distance the truss protrudes after the centerline of the carrying column (Fig. 2).
- *Bays*: These are the spans between the trusses in the top view (Fig. 2).
- *End bay*: This is a last bay in a building.
- *Purlins*: These are light members positioned across the bays and are carried on top of the upper chord (Figs. 2 and 3). Purlins, in turn carry the roof cladding.
- *Roof braces*: These are X-shaped sets of members (Fig. 2) that are present in some bays in order to increase the overall structure stiffness.
- *Longitudinal braces*: These are sets of members across the bays that are included to increase the overall rigidity of the structure (Figs. 2 and 3).

### 2.2. Design variables

Twenty-seven variables that a designer can modify are used as design variables in this parametric model.

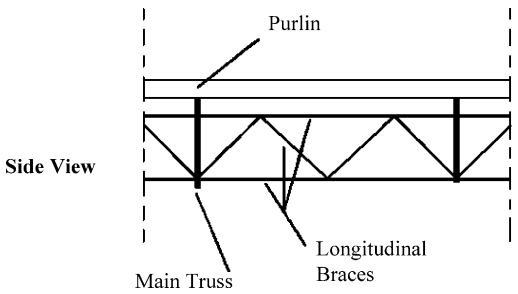



Fig. 3. Longitudinal braces.

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 follows.

### 2.2.1. Topology and configuration variables

(X1) This is an integer number that defines the selected roof layout plan (from up to five user-defined choices), this variable subsequently sets the number of main bays' lengths and end bays' lengths. In the particular considered truss however, only one roof layout was allowed, making this variable fixed.

(X2) This is the length of the vertical member directly on top of the support. Normally, this variable is continuous, but it is discretized into about 20 steps in this model to avoid the necessity of using mixed integer/continuous optimizers. However, discretization does not impose much deviation from practicality, since the fabrication often favors "rounded-off" and similar dimensions.

(X3) This 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) This is the number of subdivided 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) (Fig. 3), longitudinal braces 6 generally improves the angle of the diagonal members which in turns gives better distribution of the axial forces in the members.

(X5) This 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) This 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) This 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) This is the total number of longitudinal braces lines across the roof (both type-1 and type-2).

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

(X10) This is the number of nodes (equal to number of cells minus one) on type-1 longitudinal braces.

(X11) This is the number of nodes on type-2 longitudinal braces.

### 2.2.2. Member sizing variables

X12–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, three groups of verticals, four groups of diagonals, longitudinal braces two groups of chords, two groups of verticals and two groups of diagonals. These variables dictate the selection of the truss members from within a database of standard cross-sections. The database contains 48 different choices from among standard sizes of I-, C- and L-sections.

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 the following:

- Load cases include dead load, live load and wind load. Load case combinations are dead load + live

load, dead load + wind load and dead load + live load + wind load.
- 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.

A good review of constraint handling approaches is provided in [16]. In this paper, constraints are enforced through adaptive penalty [4]. 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:

- When no constraints are violated.
  In this case, $\mathrm{OF} = W_t$.
- When one or some of the constraints are violated.
  In this case, $\mathrm{OF} = \max(W_t, W_b) \times C_{Pen} \times C_{Pen}$, where $W_t$ is the total weight of the considered structure, $W_b$ the total weight of the best feasible structure encountered so far during the optimization, $C_{Pen}$ is a penalty constant, and $I_{Pen}$ 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 (GA1) tested in this paper implements variable storage as integer variables, four 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 as in [12] and are translated to their equivalent binary strings when needed during some of the crossover and mutation operators.

#### 3.1.1. Crossover operators

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

$$\mathrm{ChildVal1} = \mathrm{Round}(\alpha \times \mathrm{ParentVal1} + (1-\alpha) \times \mathrm{ParentVal2});$$

$$\text{ChildVal2} = \text{Round}((1-\alpha) \times \text{ParentVal1} + \alpha \times \text{ParentVal2}).$$

- *Outer crossover* (*adopted from real-coded GA*): The new variable value is computed as ChildVal1 = Round(StrongerParentVal + $\alpha$(StrongerParentVal − WeakerParentVal)) where $\alpha$ in both cases is a randomly generated number between 0 and 1.
- *Uniform crossover* [17]: In which the variables are unchanged, but exchanged between the parents with a 50% probability of exchange.

#### 3.1.2. 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 fitness.

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*: This 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 (GA2) is the same as GA1, but all evaluations of objective function are stored. Thus, when performing population members OF evaluation, only unexplored 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 (GA3) is the same as GA2, 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 [8–10] 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 [1] 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 follows:

1. Begin at a starting point.
2. Examine non-tabooed neighboring points and move to the best of them.

3. If new point has 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.

A 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 [1] 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 $\pm 1$. 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 to prevent moving into it again until some other moves are completed. 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. Thus, the tabooing does not always prevent cycling back into the domain of attraction of a local optimum, which occasionally calls for executing the “quick escape” mechanism.

### 4.4. Quick escape mechanism

RTS senses the existence of a large basin of attraction in the objective function that tabooing is not enough to overcome when the average cycle length of going back to a previously marked local optimum exceeds a threshold value. The threshold value is specified as a fraction of the maximum taboo list length. In the previous study [13], the threshold value was taken as 90% of the maximum taboo list length. However, further experimentation revealed that the 90% value takes too long to trigger the escape mechanism. In this study, the escape mechanism is triggered when the

average cycle length exceeds 50% of the maximum taboo list length.

Quick escape is performed by randomly changing the values of some of the variables of the current point. It is simply like re-starting the search at new starting point, which may or may not have been visited before, by "mutating" the current design. The point obtained after the quick escape jump, is thus not entirely random, but is in some sense related to the point from which the jump is made.

### 4.5. Modified escape mechanism

The main reasons for using an escape mechanism in RTS is to escape large basins of attraction and to diversify the search into unexplored regions of the search space. The escape mechanism described in Section 4.4, which was proposed by Battiti and Tecchiolli [1] can be viewed as a mutation of the local optimum point. Such mechanism introduces an element of randomness into RTS, which is otherwise completely deterministic. The modified escape mechanism follows the following steps:

1. Form a population using some of the already explored points.
2. Evolve the population in a manner encouraging an up-hill climb.
3. Select one point of the up-hill population that is as far as possible from the mean of the explored space (analogous to speciation in GA).
4. Perform the same sort of mutation as in normal escape mechanism on the selected point (instead of on the local optimum).

The proposed modification of the escape mechanism thus, serves to increase the overall randomness of the process and encourages the escape mechanism to jump into regions that have not been explored before.

The underlying assumptions that justify the use of the modified escape mechanism are as follows:

- The new optima to be found are pretty far from the currently found ones, so moving away from the found local optima before making the jump is better than making the random jump from the local optima themselves.
- The actual objective function evaluation is expensive in terms of computational time, so the population search among the previously explored points (which does not require new objective function evaluations) is not too much of an added computational expense on the overall search process whose dominant computational expense is the evaluation of new points.

Mixing both normal and modified escape mechanisms is also an interesting possibility to make use of the advantages of both.

## 5. A demonstrative example

A simple example involving multi global optima is used to examine the effect of the different escape mechanisms on the reactive taboo search capability to find many of the optima.

The objective function is given as

$$\min f = \sum_{i=1}^{N} x_i^2.$$

Subject to the constraints, $-5 \leq x_i \leq$ and $x_i$ are integers for $i = 1$ to $N$, where $N$ is the number of variables.

The already known solution of this problem is that all corner points (having all combinations of $x_i = \pm 5$)

Table 1
Number of local optima found in the demonstrative example when employing the normal, modified and mixed quick escape mechanisms

| $N$ | Number of existing optima | Number of optima found | | | | | |
|---|---|---|---|---|---|---|---|
| | | Within the first 5000 objective function evaluations | | | Within the first 8000 objective function evaluations | | |
| | | Normal | Modified | Mixed | Normal | Modified | Mixed |
| 6 | 64 | 27 | 48 | 40 | 34 | 52 | 59 |
| 16 | 65536 | 4 | 8 | 8 | 4 | 14 | 12 |

are global optima. The number of existing optima is thus $2^N$.

Table 1 shows the number of global found by RTS in a typical set of optimization runs while implementing the normal escape mechanism, the modified one and a mix of both. The mixed escape mechanism simply uses the normal escape mechanism to perform the first escape, then the modified mechanism in the following escape, then the normal one for the next escape and keeps alternating the escape mechanisms.

This example represents an extreme case where all the optima a very far from each other and thus the modified escape mechanism easily outperforms the normal one. This may not be the case in real-life problems. It is interesting though to see that the mixed escape mechanism is nearly performing just as good as the modified one. The next section examines the performance of the different implementations of algorithms for optimizing a real N-shaped truss.

## 6. Algorithms application

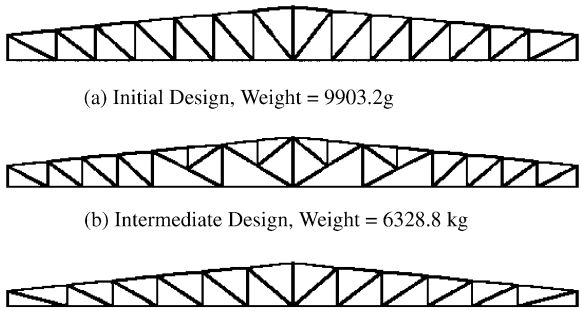
### 6.1. Truss data

Data of a real N-shaped truss is used as a starting point for the optimization algorithms. The truss data is given in Table 2. 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. The truss member cross-sections are given in Table 3.

Table 2
Truss data

| | |
|---|---|
| Number of main bays | 2 |
| Building clear-span (m) | 21.0 |
| Material Young's modulus (GPa) | 207 |
| Allowed stress (MPa) | 140 |
| Maximum slenderness (compression members) | 180 |
| Maximum slenderness (all members) | 300 |
| Maximum deflection under live load | 1/300 of span |
| Live load (kg/m$^2$) | 50 |
| Wind pressure (kg/m$^2$) | 50 |
| Dead load | Weight + 20 kg/m$^2$ |

Available database contains L-sections (LPN), C-sections (UPN and CFC) and I-sections (IPN and IPE).



(a) Initial Design, Weight = 9903.2g

(b) Intermediate Design, Weight = 6328.8 kg

(c) Best Obtained Design, Weight = 5655.3

Fig. 4. Truss topology and configuration.

### 6.2. GA parameters

Among the 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.
- *Maximum number of OF evaluations*: Tested several.

Table 3
Chosen truss member groups cross-sections

| Variable | Designs | | |
|---|---|---|---|
| | Initial | Intermediate | Final best |
| X12 | CFC[a] 140x4 | CFC 140x3 | CFC 140x3 |
| X13 | 2xLPN[a] 70x7 | 2xLPN 70x7 | 2xUPN[a] 65 |
| X14 | 2xLPN 70x7 | 2xLPN 70x7 | 2xIPN[a] 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 (kg) | 9903.2 | 6328.8 | 5655.3 |

[a] CFC: cold-formed C-section, LPN: standard L-section, UPN: hot-rolled C-section, IPN: standard I-section.

- *Overall crossover probability*: 0.9.
- Equal probability for different crossover operators.
- *Overall mutation probability*: 0.25.
- Equal probability for different mutation operators.
- *Fitness scaling constant*: 1.6.

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

### 6.3. 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.
- *Maximum number of OF evaluations*: Tested several.

### 6.4. Results and discussion

Each of the design variables concerned with truss member sizing has 48 possible choice options, variables concerning configuration and topology range between 3 and 20 options. The total search space (all possible combinations of variables) is $1.58814 \times 10^{37}$. 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 $6.3 \times 10^{-34}$ of the total search space. For moderate size trusses as the one considered in this paper, one OF evaluation consumes about 150 ms on an 800 MHz PC, which allows for making multiple runs, each consuming about 20 min of computer time.

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 3. The intermediate design is shown as a demonstration of topology change as well as sizing.

Since RTS has less stochastic content compared to GA, ten optimization runs are used as a representative of RTS. Twenty runs are performed for each of GA1, GA2 and GA3 using four different population sizes (five runs for each population size). The results of the optimization runs performed in the previous study [13] are summarized in Table 4 and plotted in Figs. 5 and 6. The RTS runs of [13] are referred to as the ones with "Delayed Escapes" as their escapes are not triggered until the average cycle length reaches 90% of the maximum taboo list length.

The results shown are for the number of new objective function evaluations, thus caching in GA2 and

Table 4
Optimization results

| No. of OF evaluation | Objective function value | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Average | | | | | Best run | | | | |
| | GA1 | GA2 | GA3 | RTS-D'lyd | RTS-Norm | GA1 | GA2 | GA3 | RTS-D'lyd | RTS-Norm |
| 500 | 9518 | 9487 | 9034 | 7780 | 7781 | 8197 | 7703 | 7534 | 7781 | 7781 |
| 1000 | 9346 | 9209 | 8825 | 6687 | 6688 | 7599 | 7656 | 7534 | 6688 | 6688 |
| 1500 | 9216 | 8973 | 8775 | 6491 | 6491 | 7599 | 7656 | 7534 | 6491 | 6491 |
| 2000 | 9088 | 8929 | 8759 | 6430 | 6430 | 7599 | 7656 | 7534 | 6430 | 6430 |
| 2500 | 8987 | 8929 | 8706 | 6430 | 6430 | 7599 | 7656 | 7534 | 6430 | 6430 |
| 3000 | 8866 | 8840 | 8658 | 6430 | 6430 | 7599 | 7656 | 7259 | 6430 | 6430 |
| 4000 | 8754 | 8616 | 8538 | 6430 | 6430 | 7270 | 7236 | 7259 | 6430 | 6430 |
| 5000 | 8664 | 8591 | 8463 | 6430 | 6430 | 7270 | 7236 | 7259 | 6430 | 6430 |
| 6000 | 8513 | 8293 | 8427 | 6430 | 6283 | 7270 | 7236 | 7259 | 6430 | 5868 |
| 7000 | 8436 | 8226 | 8358 | 6430 | 6152 | 7270 | 7194 | 7259 | 6430 | 5649 |
| 8000 | 8396 | 8115 | 8255 | 6280 | 6128 | 7174 | 7034 | 7259 | 5677 | 5649 |
| 9000 | 8263 | 7998 | 8150 | 6249 | 6113 | 7174 | 7034 | 7259 | 5649 | 5649 |
| 10000 | 8213 | 7935 | 7969 | 6249 | 6113 | 7174 | 7034 | 7259 | 5649 | 5649 |

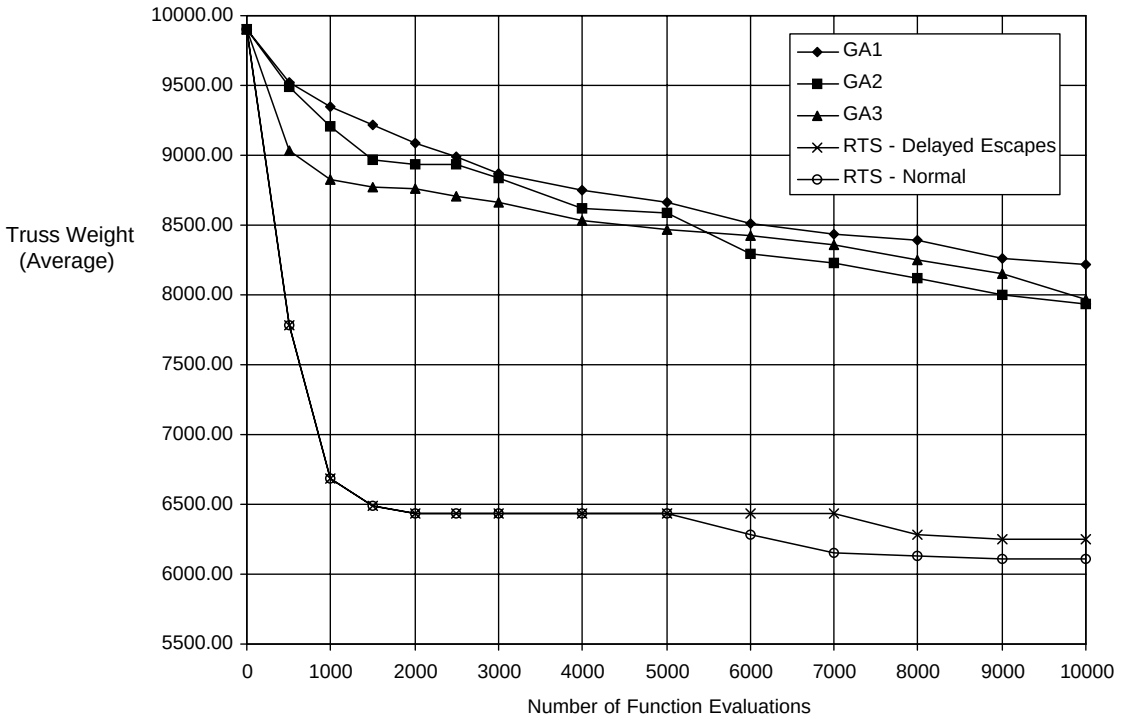



Fig. 5. Optimization progress—average truss weight in performed runs.

GA3 resulted in improvement of the performance over the traditional GA1. Furthermore having the initial population normally distributed about the starting point in GA3 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. GA3 however has little or no advantage over GA2 towards the end of the search. Examination of Figs. 5 and 6 and Table 4 also shows an appreciably better performance of RTS over GA. Possible reasons for RTS being better suited for the examined optimization problem than the implemented forms of GA are as follows.

GA relies on having several points that are distributed over the search space (population) to achieve diversification. According to the schemata theory [12], 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 [6]. Mutation is generally used to increase diversification, especially when the whole population gets too closely attracted to a certain region.

The main weakness of GA when the problem has large dimensionality is that a moderate population size (100–200 members) becomes insufficient to achieve enough diversification over the search space and a sufficient schemata pool. 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 introduces a member which is much better than the others, thus may lead to pre-mature convergence and thus may further decrease 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

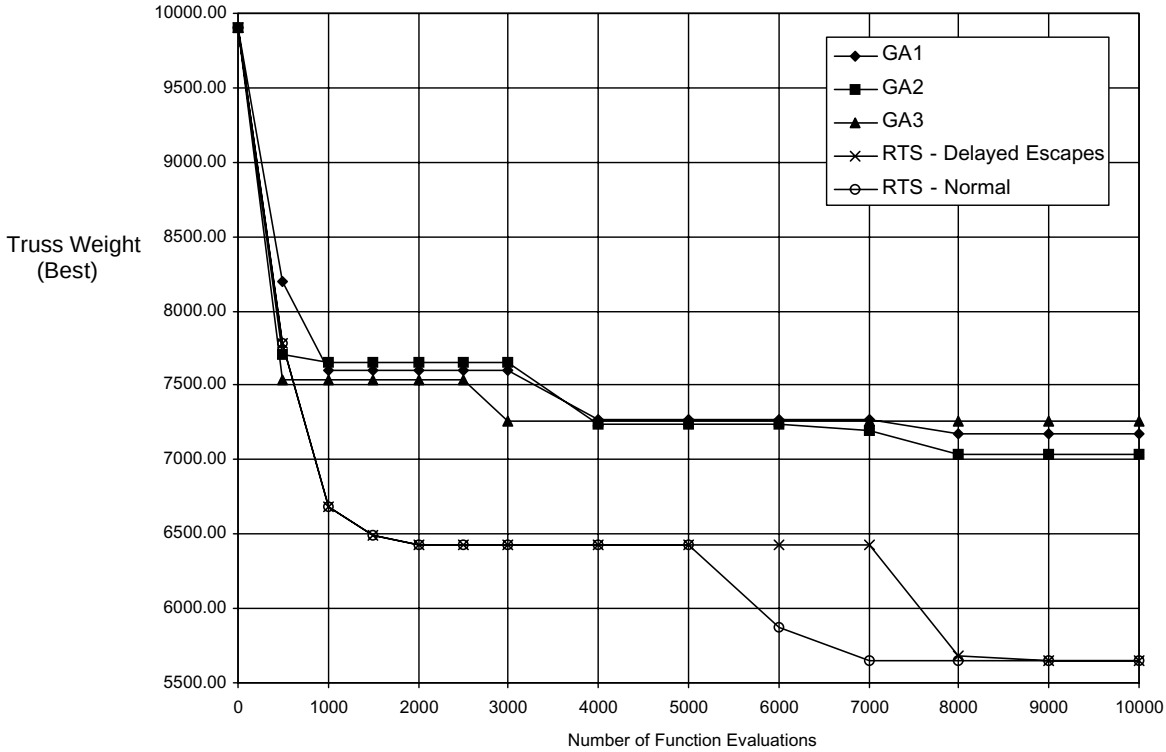



Fig. 6. Optimization progress—truss weight in best of performed runs.

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 and 6. Upon reaching a local optimum, imposing taboo conditions switches RTS to diversification by temporarily preventing the revisit of 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 and 6 as well as Table 4 that after the good start, the Delayed Escape RTS remained incapable of finding any better designs for quite a long period. Thus, decreasing the threshold for triggering quick escapes in the current study from 90 to 50% of the maximum taboo list length proved beneficial. Thus, the threshold of 50% taboo list length is simply termed "normal escape mechanism" and is used again as a reference in the rest of this study in Table 5 and Figs. 7 and 8.

The fact that the normal escape mechanism of RTS merely "hopes" that its jump falls into a better objective function basin of attraction motivates introducing some analogies of GA into RTS in order to enhance its diversification capabilities. The proposed modification of the RTS escape mechanism is an attempt at accomplishing such better diversification without additional objective function evaluations.

As a comparison of RTS with different escape mechanisms, Table 5 and Figs. 7 and 8 show the optimization results using the normal, modified and mixed escape. It is observed for this type of problem that the normal escape mechanism is almost always better than the modified one. This may imply that the optima are not scattered in locations far from each other and thus there is no extra benefit in using the modified escape mechanism. However, it is observed that the mixed escape mechanism, which repetitively alternates between the two types, gets the benefit of both more diversification as well as better exploitation

Table 5
Optimization results

| No. of OF evaluation | Objective function value | | | | | |
|---|---|---|---|---|---|---|
| | Average | | | Best run | | |
| | RTS-Norm | RTS-Mod | RTS-Mix | RTS-Norm | RTS-Mod | RTS-Mix |
| 500 | 7781 | 7781 | 7781 | 7781 | 7781 | 7781 |
| 1000 | 6688 | 6688 | 6688 | 6688 | 6688 | 6688 |
| 1500 | 6491 | 6491 | 6491 | 6491 | 6491 | 6491 |
| 2000 | 6430 | 6430 | 6430 | 6430 | 6430 | 6430 |
| 2500 | 6430 | 6430 | 6430 | 6430 | 6430 | 6430 |
| 3000 | 6430 | 6430 | 6430 | 6430 | 6430 | 6430 |
| 4000 | 6430 | 6430 | 6430 | 6430 | 6430 | 6430 |
| 5000 | 6430 | 6430 | 6430 | 6430 | 6430 | 6430 |
| 6000 | 6283 | 6430 | 6283 | 5868 | 6430 | 5868 |
| 7000 | 6152 | 6430 | 6253 | 5649 | 6430 | 5649 |
| 8000 | 6128 | 6430 | 6220 | 5649 | 6430 | 5536 |
| 9000 | 6113 | 6413 | 6173 | 5649 | 6255 | 5536 |
| 10000 | 6113 | 6345 | 6128 | 5649 | 5911 | 5511 |
| 12000 | 6075 | 6079 | 6060 | 5618 | 5638 | 5511 |
| 15000 | 6066 | 6042 | 6005 | 5527 | 5638 | 5511 |
| 20000 | 6048 | 6022 | 5914 | 5527 | 5638 | 5511 |

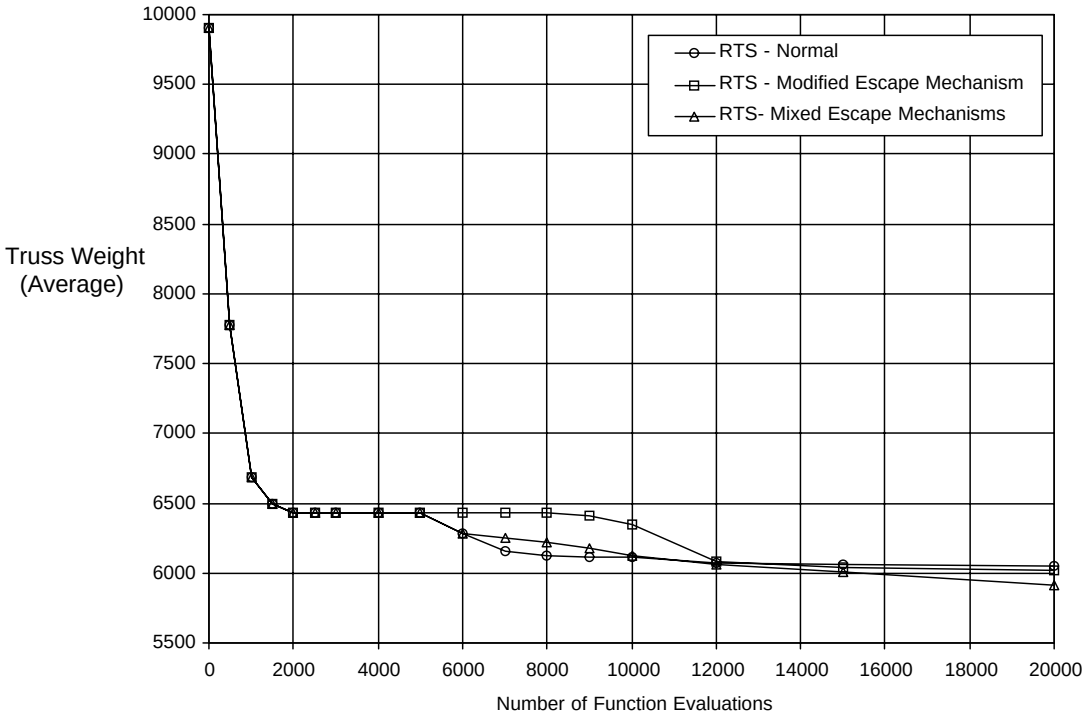



Fig. 7. Optimization progress—average truss weight in performed runs.

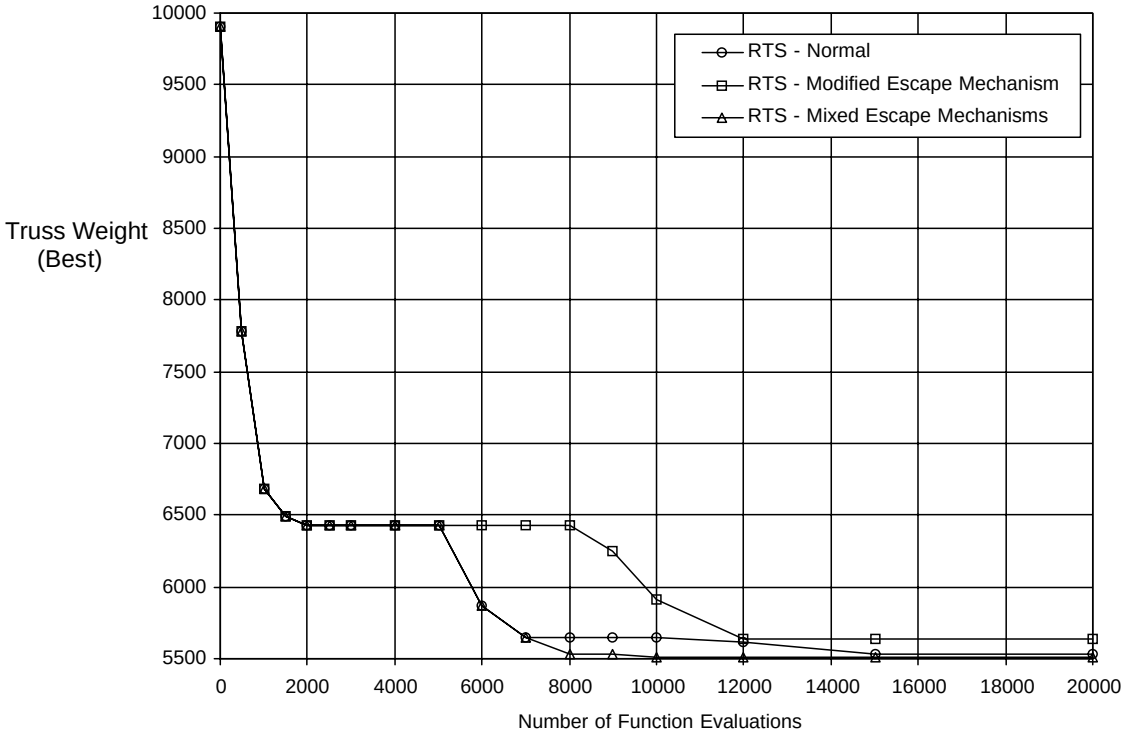



Fig. 8. Optimization progress—truss weight in best of performed runs.

of zones having optima that are not too far from each other. The observed performance of RTS employing the mixed escape mechanism in Table 5 and Figs. 7 and 8 seem to be better than having one of the two mechanisms on its own.

## 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 a challenging task 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 in improving the existing truss design that is actually erected. While utilizing a number of objective function evaluations that is only a small fraction of the total search space, both GA and RTS succeeded in coming up with better designs. RTS performed better than GA thought it has less diversification capabilities, which motivated a modification to the mechanism RTS uses to escape large attraction basins in the objective function. The modified escape mechanism is a population-based search within the previously evaluated points. The population search is directed to find regions that have not been explored before. Since the population search is done on pre-computed points, it has no additional cost in terms of objective function evaluation. The performed study shows that alternating between the normal and modified escape mechanisms gives an overall better performance.

## Acknowledgements

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


[1] R. Battiti, G. Tecchiolli, The reactive taboo search, ORSA J. Comput. 6 (1994) 126–140.

[2] M. Bendose, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Comput. Meth. Appl. Mech. Eng. 71 (1988) 197–224.

[3] C. Chapman, K. Saitou, M. Jakiela, Genetic algorithms as an approach to configuration and topology design, Adv. Design Automat. 65 (1993) 485–498.

[4] S. Chen, An approach for impact structure optimization using the robust genetic algorithm, Finite Elements Anal. Design 37 (2001) 431–446.

[5] K. Deb, S. Gulati, Design of truss-structures for minimum weight using genetic algorithms, Finite Elements Anal. Design 37 (2001) 447–465.

[6] F. Erbatur, O. Hasancebi, Layout optimization using GAs and SA, in: Optimal Structural Design Workshop, GECCO-2001, San Francisco, 2001, pp. 102–107.

[7] L. Gil, A. Andreu, Shape and cross-section optimization of a truss structure, Comput. Struct. 79 (2001) 681–689.

[8] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Operations Res. 13 (5) (1986) 533–549.

[9] F. Glover, Taboo search—Part I, ORSA J. Comput. 1 (1989) 190–206.

[10] F. Glover, Taboo search—Part II, ORSA J. Comput. 2 (1990) 4–32.

[11] D.E. Goldberg, M. Samtani, Engineering optimization via genetic algorithms, in: Proceedings of the 9th Conference on Electronic Computations, ASCE, Birmingham, 1986, pp. 471–482.

[12] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, MA, 1989.

[13] K. Hamza, H. Mahmoud, K. Saitou, Design optimization of N-shaped roof trusses, in: GECCO-2002, Morgan Kaufmann, New York, 2002, pp. 1089–1096.

[14] M. Jakiela, C. Chapman, J. Duda, A. Adewuya, K. Saitou, Continuum structural topology design with genetic algorithms, Comput. Meth. Appl. Mech. Eng. 186 (2) (2000) 339–356.

[15] U. Kirsch, Optimal design of trusses by approximate compatibility, Comput. Struct. 12 (1979) 93–98.

[16] A. Kurpati, S. Azarm, J. Wu, Constraint handling improvements for multi-objective genetic algorithms, Struct. Multidis. Optim. 23 (2002) 204–213.

[17] Z. Liang-Jie, M. Zhi-Hong, L. Yan-Da, Mathematical analysis of crossover operator in genetic algorithms and its improved strategy, in: Proceedings of the IEEE Conference on Evolutionary Computation, vol. 1, IEEE, Piscataway, NJ, 1995, pp. 412–417.

[18] J. Moh, D. Chiang, Improved simulated annealing search for structural optimization, AIAA J. 38 (2000) 1965–1973.

[19] S. Rajeev, C.S. Krishnamoorthy, Discrete optimization of structures using genetic algorithms, J. Struct. Eng. 118 (5) (1992) 1233–1250.

[20] J. Taylor, M. Rossow, An optimal structural design using optimality criteria, in: Proceedings of the 13th Annual Meeting on Advances in Engineering Science, NASA, Hampton, 1976, pp. 521–530.