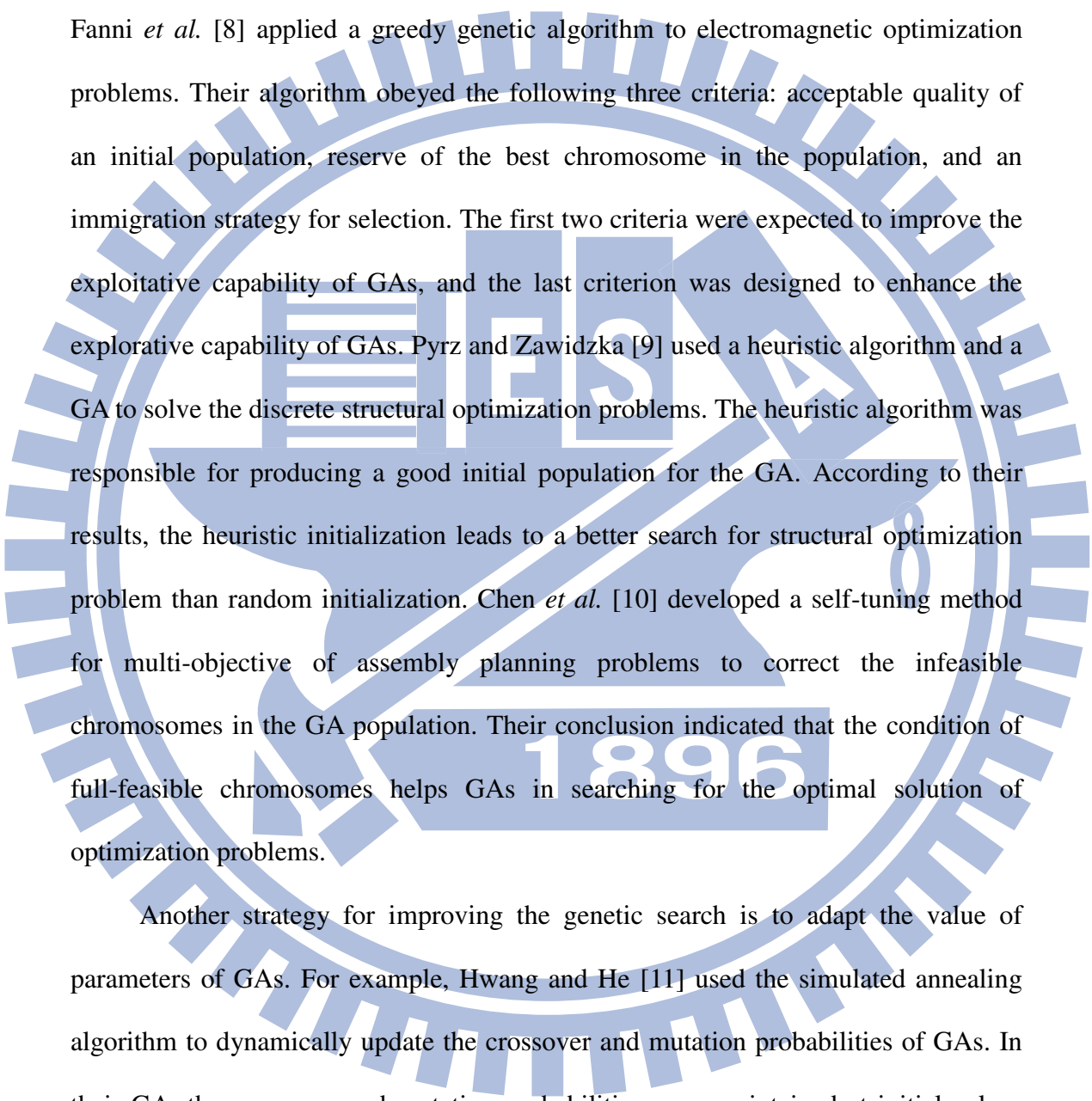


Chapter 1 Introduction

1.1. Background and motivation

The genetic algorithm (GA) is a stochastic optimization method based on selecting fitter solutions and exchanging information between selected solutions. In a simple GA, a solution is represented using a variable vector. Variable vectors are transformed into binary strings, called individuals or chromosomes. Each decision variable of solutions is linked to a substring of chromosomes, called a gene. Figure 1-1 shows the flow chart of a simple GA, and figure 1-2 shows the data flow of a simple GA. First, an even number of random variable vectors are generated to form an initial population. The fitness values of the variable vectors are calculated by a predefined fitness function, and the variable vectors are transformed into chromosomes. Second, a selection operator is adopted to copy the same number of chromosomes from the population. A chromosome with a large fitness values has high probability of being copied one or more times. Third, every two copied chromosomes reproduce two new chromosomes, called offspring, to form the next generation of a population. A crossover operator and a mutation operator play the major roles in reproduction. Fourth, the fitness values of offspring are calculated by the same fitness function used in step 1. If stopping criteria are met, then the computation of GA is stopped; otherwise, step 2, 3 and 4 are repeated.

The genetic algorithm (GA) is a commonly used method for solving discrete valued truss structural optimization problems [1-6]. However, GAs often spend a significant amount of computational time in searching for the optimal solution of discrete structural optimization problems, especially when the search space has enormous number of potential solutions. The reason is that the GA continuously



balances exploitation and exploration [7]. Exploitation is the search strategy of exploiting the best solution, and exploration is the search strategy of exploring the search space. According to the literatures, various strategies have been developed to strengthen GAs [8-15]. Among the strategies include the greedy notion. For example, Fanni *et al.* [8] applied a greedy genetic algorithm to electromagnetic optimization problems. Their algorithm obeyed the following three criteria: acceptable quality of an initial population, reserve of the best chromosome in the population, and an immigration strategy for selection. The first two criteria were expected to improve the exploitative capability of GAs, and the last criterion was designed to enhance the explorative capability of GAs. Pyrz and Zawidzka [9] used a heuristic algorithm and a GA to solve the discrete structural optimization problems. The heuristic algorithm was responsible for producing a good initial population for the GA. According to their results, the heuristic initialization leads to a better search for structural optimization problem than random initialization. Chen *et al.* [10] developed a self-tuning method for multi-objective of assembly planning problems to correct the infeasible chromosomes in the GA population. Their conclusion indicated that the condition of full-feasible chromosomes helps GAs in searching for the optimal solution of optimization problems.

Another strategy for improving the genetic search is to adapt the value of parameters of GAs. For example, Hwang and He [11] used the simulated annealing algorithm to dynamically update the crossover and mutation probabilities of GAs. In their GA, the crossover and mutation probabilities were maintained at initial values when the best chromosome was continuously improved. Otherwise, the probability ratios were gradually increased to extend the explorative capability of GAs. Velley [12] proposed using a dynamic population size for GAs to solve database query

optimization problems. His notion was based on the biology evolution concept. *The population size of a species is increased when the species fit an abundant environment.* Therefore, the dynamic population size GA increased its population size when the most chromosomes of populations had similar fitness values, and decreased the population size when a few chromosomes were obviously stronger than the others.

The third strategy for improving the genetic search is that GAs use the multi-population strategy to process the evolution. For example, Perry et. al. [13] applied a multi-population GA to structural identification problems. In their GA, all populations had the same population size; in addition, the fitter chromosomes were forced to migrate to a dominant population. Ma and Gong [14] proposed using a large core population and several small colony populations to process the evolution. The core population was responsible for exploiting the optimal solution, and the colony populations was responsible for exploring the search space for finding new promising search regions. The operation between the two kinds of populations was that the stronger chromosomes of the colony populations were continuously sent to the core population. The fourth strategy for improving the genetic search is to hybrid an improved model from GAs and local search algorithms. Lee et. al. [15] proposed using a greedy eugenics strategy to augment GAs. The augmented GA used the simulated annealing algorithm to refine the chromosomes of offspring populations. Their notion was that the locally optimal offspring provide valuable information to search for the optimal solution of optimization problems.

The fifth strategy for improving the genetic search is the technique of reduction of the search space. Perry et. al. [13] used statistic methods to reduce the search space for GAs. First, their GA ran within the small number of iterations. Second, the average value and standard deviation of variables of survival solutions were

calculated. Third, the sets of legal values of variables were narrowed down to some values that were near the average value of variables. Moreover, if the standard deviation of a variable is smaller than a threshold value, the variable will set to the average value. Finally, if there are two or more elements in any set of legal values of variables, the above three steps will repeat.

1.2. Objective

This work proposes a high performance GA by incorporating a hill-climbing strategy and two greedy notions to a simple GA, and is applied to the optimal design of truss structures with discrete sizing variables. The hill-climbing strategy is integrated into the GA to reduce the search space, and the greedy notions, including heuristic initialization and hybrid selection, are expected to help the GA to explore the search space for identifying the most promising search region. In the hill-climbing strategy, the best chromosome is assumed to be located at the center of the most promising search region. Notably, the best solution is used as the most important hit for the next search directions can also be found in another stochastic optimization method, particle swarm optimization [16]. The hill-climbing strategy obviously differs from Perry's reduction technique [13]. Reducing the search space of each variable depends on all variables in the hill-climbing strategy. Each variable uses the same reduction speed to narrow down its promising search range. However, the reduction of search space of each variable is independent of each other in the Perry's reduction method. Some variables may be converged too early when an approximated optimal solution is found.

1.3. Organization of the dissertation

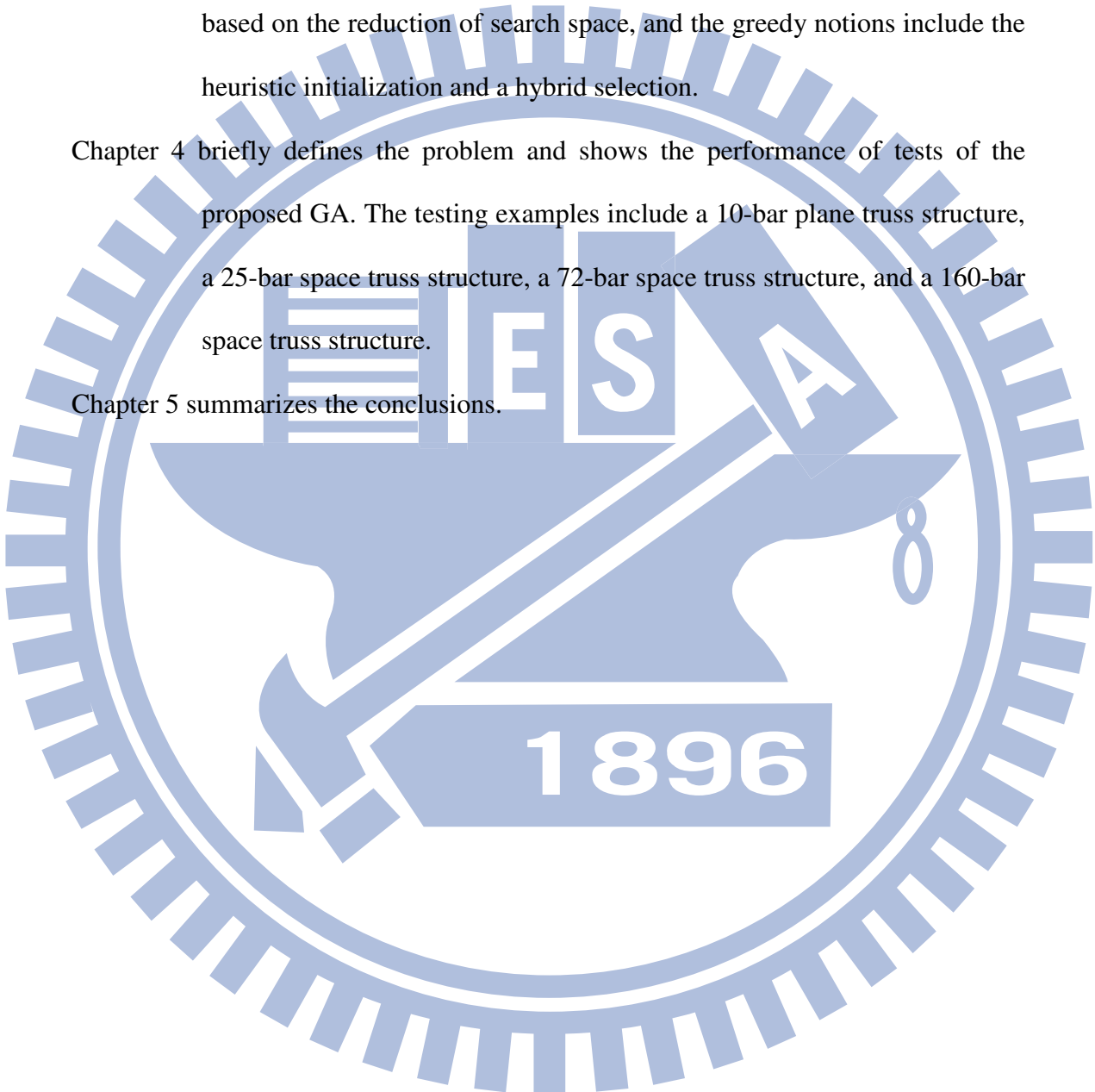
The rest of this dissertation is organized as follows:

Chapter 2: introduce the GAs, including a simple GA, greedy GAs, adaptive GAs and multi-population GAs.

Chapter 3 explains the hill-climbing and greedy GA. The hill-climbing strategy is based on the reduction of search space, and the greedy notions include the heuristic initialization and a hybrid selection.

Chapter 4 briefly defines the problem and shows the performance of tests of the proposed GA. The testing examples include a 10-bar plane truss structure, a 25-bar space truss structure, a 72-bar space truss structure, and a 160-bar space truss structure.

Chapter 5 summarizes the conclusions.



Chapter 2 Genetic algorithms

2.1. Simple GA

2.1.1. Chromosomes(or called individuals)

The genetic algorithm (GA) is a stochastic optimization method based on selecting fitter solutions and exchanging information between selected solutions. In a simple GA, a solution is represented using a variable vector. Variable vectors are transformed into binary strings, called individuals or chromosomes. Each decision variable of solutions is linked to a substring of chromosomes, called a gene. For example, the vector $(V[2], V[7], V[13], V[3])$ is a solution of an optimization problem, and V is a feasible value array. If the feasible value array contains 16 possible values, then the corresponding chromosome can be represented as follow:

$$(2, 7, 13, 3)_{10} \Rightarrow 0010_2 0111_2 1101_2 0011_2$$

Each chromosome has a fitness value that is estimated by a predefined fitness function. A chromosome with a large fitness value means that the chromosome has a higher probability to alive in the latter competing. On the other hand, a chromosome with a small fitness value means that it possibly dies in the latter competing.

2.1.2. Random initialization

The advantage of the genetic algorithm for optimization problems is that it can overcome the local minimum problem which is generally a challenge for other optimization methods. One reason is that, the genetic algorithm uses a population of start points to exploit the solutions. In the simple genetic algorithm, the initial population is composed of an even number (called population size) of random variable vectors.

2.1.3. Roulette wheel selection

Selection operators are designed for GAs to save strong chromosomes of parent

population so as to improve the offspring population. The basic notion is that a chromosome with a larger fitness value has the higher probability to be copied one or more times. The roulette wheel selection is the most commonly used selection method, and is illustrated in figure 2-1. The selection method randomly selects chromosomes from the previous generation, and the selection probability of any chromosome is positively proportional to its fitness value.

Suppose C_1, C_2, \dots, C_k are the chromosomes in the population, and their fitness values are f_1, f_2, \dots, f_k , respectively. First the selection probability of each chromosome is calculated as follows:

$$p(C_i) = \frac{f_i}{\sum_{j=1}^k f_j} \quad \text{for } i=1 \text{ to } k \quad (2-1)$$

Then k random values (r_1, r_2, \dots, r_k) in the interval of $[0, 1]$ are generated for selecting the chromosomes from the population. The selection rule is shown as follows:

$$\text{If } \sum_{i=1}^l p(C_i) \leq r_j < \sum_{i=1}^{l+1} p(C_i) \text{ then } C_l$$

For example, a population has 4 chromosomes C_1, C_2, C_3, C_4 , and their fitness values are 1, 9, 3, 7, respectively. Now, four random values 0.73, 0.24, 0.49, 0.64 are generated for the roulette wheel selection. The selection results are shown as follows:

Chromosomes in the population	C_1	C_2	C_3	C_4	
Fitness values	1	9	3	7	$\sum = 20$
Selection probability	0.05	0.45	0.15	0.35	
Accumulated probability	0.05	0.5	0.65	1.00	

Selection rules:

$$\text{IF } 0 \leq r < 0.05 \text{ then } C_1$$

$$\text{IF } 0.05 \leq r < 0.5 \text{ then } C_2$$

IF $0.5 \leq r < 0.65$ then C_3

IF $0.65 \leq r \leq 1.0$ then C_4

Selection results:

$r=0.73 \Rightarrow C_4$

$r=0.24 \Rightarrow C_2$

$r=0.49 \Rightarrow C_2$

$r=0.64 \Rightarrow C_3$

2.1.4. Crossover

After the selection procedure is finished, GAs use a crossover operator to rebuilt the copied chromosomes. Basically, every two copied chromosomes can produce two new chromosomes. Figure 2-2 illustrates three commonly used crossover operators, and the detail of the crossover operators is introduced in the following.

(1) Single-point crossover

For example, two chromosome instances are used for a single-point crossover operator. They are defined as follows:

$C_1=0110110011101000110100110$

$C_2=1001100101100101110111001$

First a random integer number is generated. Suppose the random number is 5, then the above two chromosomes are divided into two blocks at the location of 5, respectively.

$C_1=01101 \underline{10011101000110100110}$

$C_2=10011 \underline{00101100101110111001}$

Finally, the two chromosomes exchange their right blocks to produce two new chromosomes.

$C'_1=01101 \underline{00101100101110111001}$

$C'_2=10011 \underline{10011101000110100110}$

(2) Multi-point crossover

The chromosome instances used in (1) also used to explain the multi-point crossover operator. The multi-point crossover operator divided the chromosomes into three or more blocks. The two chromosomes exchange their even blocks to produce new chromosomes. Suppose 5, 13, 18 are randomly generated integer numbers. Chromosomes C_1 and C_2 are divided into four blocks, respectively.

$$C_1 = \underline{01101} \ \underline{10011101} \ \underline{00011} \ \underline{0100110}$$

$$C_2 = 10011 \ 00101100 \ 10111 \ 0111001$$

The two chromosomes exchanges their second and forth blocks to produce the new chromosome, and the new chromosomes are listed as follows:

$$C'_1 = \underline{01101} \ 00101100 \ \underline{00011} \ 0111001$$

$$C'_2 = 10011 \ \underline{10011101} \ 10111 \ \underline{0100110}$$

(3) Uniform crossover

The uniform crossover always exchange even positions of binary between two chromosomes. The chromosome instances used in (1) and (2) is adopted to explain the uniform crossover operator again. The results are listed as follows:

$$C'_1 = \underline{0011100111101101110101100}$$

$$C'_2 = \underline{1100110001100000110110011}$$

2.1.5. Mutation

The operator is designed to change some genes of individuals for preventing the population of individuals from becoming too similar to each other. Mutation operator is also a random operator, and is defined as follows:

$$I'(i) = \begin{cases} 1 & \text{(if } I(i) = 1 \text{ and } r > m_r \text{) or (if } I(i) = 0 \text{ } r \leq m_r \text{)} \\ 0 & \text{(if } I(i) = 0 \text{ and } r > m_r \text{) or (if } I(i) = 1 \text{ } r \leq m_r \text{)} \end{cases} \quad (2-2)$$

where $i=1$ to the length of a individual, r is a random number in the interval of $[0, 1]$,

and m_r is the mutation ratio.

For example, $m_r=0.1$

$I=0$	1	1	0	1	1	0
$r=0.70$	0.24	0.03	0.48	0.56	0.93	0.25
$I=0$	1	0	0	1	1	0

Generally, the mutation ratio should be small. If the mutation ratio is too large, the GA is like a random search algorithm.

2.1.6. Stop criteria

If stopping criteria are met, then the computation of GA is stopped; otherwise, selection, crossover and mutation operating are repeated. Basically, the stopping criteria of GA are decided by the users. The simplest stopping criterion is to stop a GA when the iteration number exceeds the upper bound. Another stopping criterion is that the GA is stopped if the best individual does not improved for a certain amount of iterations.

2.2. Greedy GA

The difference between a simple GA and a greedy GA is that the latter considers eugenics. In this section, three eugenic strategies for GAs are described.

2.2.1. Heuristic initialization

The heuristic initialization can speed up the GA to find solutions, especially for the large scale problems [7 and 9]. The notion of heuristic initialization is that the good start points can help GA in exploring the search space because their locations are generally far from valueless regions of the search space. The only drawback for applying the heuristic initialization to GAs is that the extra knowledge of problem for modifying random chromosome should exist.

2.2.2. Always save the best chromosome

According to eugenics, the best chromosome has the largest opportunity to produce the strongest chromosome. However, the roulette wheel selection is possible to lose the best chromosome due to its random feature. On the other hand, the $(\mu + \lambda)$ selection method based on eugenics ensures that the best chromosome is always saved [7]. The feature of the $(\mu + \lambda)$ selection method is that the best k chromosomes are selected from parent and offspring populations as both populations have k chromosomes, respectively. First the $2k$ chromosomes of parent and offspring populations are ranked according to their fitness values. Then the first k chromosomes are saved, and the others are died.

2.2.3. Immigration for population

If a GA uses eugenic strategies, inbreeding problems should be considered. The inbreeding problem means that the GA exploits solutions in some small regions of the search space only, and ignores the importance of exploring the other regions of the search space. The simplest way to avoid inbreeding is to add immigration into the population. The immigration can be generated in any way, but they should be independent on the parent population.

2.3. Adaptive GA

The simple GA is based on the Darwin's theory of evolution. Actually, biological evolution is very slow. Therefore, some GAs use dynamic working parameter to enhance their performances.

2.3.1. Dynamic population size

In biological word, a population grows up because the chromosome fit their environment and the environment is abundant in nature resource. Back to GAs, the

simple GA always uses the same number of population size. This design is far from the natural world. Therefore, using dynamic population sizes for GAs is reasonable. Velley [12] proposed the population size is increased when the most chromosomes of populations had similar fitness values, and is decreased when a few chromosomes were obviously stronger than the others (Figure 2-3). Velly's notion can be considered as an eugenic strategy. If a few chromosomes are obviously stronger than the others in a population, then the GA strongly limits the bad chromosomes to reproduce their offspring (Figure). In addition, Koumoussis and Katsaras [18] proposed that the GA uses a bigger population size and the smaller final population size in an evolution process (Figure 2-4). Their notation is that the selection probability of a chromosome is independent of the population size if the most chromosomes are very similar in a population.

2.3.2. Dynamic mutation ratio

The mutation ratios of GAs are generally small because the nature mutation probability is very low. A large mutation ratio may cause the GA to be like a random search algorithm. However, this situation seems to be a disadvantage but is sometimes useful. As the most chromosomes of a population are very similar, the selection and crossover operators are difficult to produce brand-new chromosomes. Increase the mutation ratio can force the GA to explore another region of the search space. Therefore, the GA can reduce the risk of the local minimum problem.

2.4. Multi-population GA

If a problem has a lot of local minimum over different region of the search space, then the multi-population GA can perform the better performance than a simple GA. There are several types of multi-population GA. Perry et. al. [13] emphasized that all

populations of a multi-population GA had the same population size, and the fitter chromosomes were forced to migrate to a dominant population (Figure 2-5). Their notion is that the GA keeps balance between exploration and exploitation in populations, except the dominant population. The dominant population is a eugenic population which is expected to produce the optimal solution. Ma and Gong [14] proposed using a large core population and several small colony populations to process the evolution (Figure 2-6). The core population should keep balance between exploration and exploitation. Therefore, the immigration is continuously from the colony populations. The number of colony populations can be large because each colony population consists of a few chromosomes. It is possible that each colony population locates at different region of the search space.

2.5. Hybrid GA

Local search algorithms are generally very effective for optimization problems if a good start point is given. Therefore, the notion of hybrid GA is that using GA as a global search method to find valuable regions of the search space, and using a local search algorithm to exploit solutions in the valuable regions which is found by the GA [7]. Moreover, the solution found by the local search algorithms can be used as the parent population for the GA to reproduce the offspring population [15]. Figure 2-7 shows two examples of hybrid GAs.

Chapter 3 Hill-climbing and greedy genetic algorithms (HGGA) for structural optimization problems

3.1. Statement of problems

According to design variables, structural optimization problems can be classified into three sub-problems. One sub-problem is the sizing structural optimization problem. The goal of the sizing structural optimization is to find the values of a set of cross-sectional areas of members that lead to the minimum weight of structures and satisfy all design constraints. Another sub-problem is the topology structural optimization problem. The topology structural optimization problem is similar to the sizing structural optimization problem, except for that the value of cross-sectional areas of members can not be zero in sizing structural optimization, yet can be zero in topology structural optimization. The other sub-problem is the shape optimization problem. The shape optimization problem simultaneously considers sizing and geometry variables to search for the minimum weight structure.

In this dissertation, only the sizing structural optimization problem is considered. The sizing structural optimization of truss structures with discrete sizing variables can be expressed as follows:

$$\begin{aligned} \text{Find} \quad & X \quad \text{where } x_i \in Z_i \\ \text{Minimize} \quad & F = \gamma \sum_{j=1}^m L_j A_j \quad (3-1) \\ \text{Subject to} \quad & G_{1,j}(X) \leq 0 \quad \text{for } j=1 \text{ to } m \\ & G_{2,k}(X) \leq 0 \quad \text{for } k=1 \text{ to } n \end{aligned}$$

where X is a design variable vector, x_i is the i -th element of X , Z_i is a set of predefined values (legal values) corresponding to x_i , F is the non-constraint objective function, γ is the unit weight of material, L_j and A_j are the length and

the cross-sectional area of j -th member, respectively, $G_{1,j}(X)$ is the stress constraint function of j -th member, $G_{2,k}(X)$ is the displacement constraint function of k -th coordinate, m is the number of members, and n is the number of degree of freedoms. Penalty function and Lagrange multipliers are two conventional methods of combining the non-constrained objective function and the constraints as a constrained objective function. Herein, a penalty function is employed to punish infeasible solutions for increasing the objective function. Additionally, equation (3-1) should work together with a structural analysis procedure. The structural analysis procedure is responsible for calculating the member forces and nodal displacements of potential structural designs. Figure 3-1 shows the flow chart of the structural optimization.

3.2. HGGA

3.2.1. Heuristic initialization

The optimal solution of truss structural optimization problems is always located on the boundary between feasible and infeasible search regions. Therefore, a randomly generated structural design can be refined simply by moving it directly to the boundary. The refining process has two levels of modifications: randomly generated truss structural designs to transition truss structural design and transition truss structural design to heuristically generated truss structural designs. The corresponding formulas are defined as follows:

$$a'_j = a_j^{(R)} \times R \quad \text{for } j=1 \text{ to } m \quad (3-2)$$

$$R = \text{Max} \left(\frac{d_1(X^{(R)})}{d_{1,a}}, \frac{d_2(X^{(R)})}{d_{2,a}}, \dots, \frac{d_n(X^{(R)})}{d_{n,a}} \right) \quad (3-3)$$

$$a_j^{(H)} = \begin{cases} a'_j & \text{if } \sigma_j(X^{(R)}) \times \frac{1}{R} \leq \sigma_{j,a}(a'_j) \\ a'_j \times \frac{\sigma_j(X^{(R)}) \times \frac{1}{R}}{\sigma_{j,a}(a'_j)} & \text{else} \end{cases} \quad \text{for } j=1 \text{ to } m \quad (3-4)$$

where a'_j is the j -th sizing variable of transition truss structural designs, $a_j^{(R)}$ is the j -th sizing variable of randomly generated truss structural designs $X^{(R)}$, R is a change ratio, $d_k(X^{(R)})$ is the nodal displacement of k -th coordinate corresponding to $X^{(R)}$, $d_{k,a}$ is the allowable displacement of k -th coordinate, $a_j^{(H)}$ is the j -th sizing variable of heuristically generated truss structural designs, $\sigma_j(X^{(R)})$ is the actual stress corresponding to $a_j^{(R)}$, and $\sigma_{j,a}(a'_j)$ is the allowable stress corresponding to a'_j .

Equations (3-2) and (3-3) are designed to modify randomly generated truss structural designs in order to correct their maximum displacements from any value to the allowable value. Equation (3-2) indicates that the same change ratio modifies all sizing variables. Utilizing the same change ratio to modify the cross-sectional areas of members is characterized by the fact that the member forces of modified structures are the same as those of original structures. Restated, if the values of sizing variable of randomly generated truss structural designs are magnified k times, the member stresses of transition truss structural designs are $1/k$ that of randomly generated truss structural designs. Therefore, the member stresses of transition truss structural designs can be accurately estimated directly from those of randomly generated truss structural designs. Equation (3-4) is expected to repair the stress constraints of transition truss structural designs. The calculation of equation (3-4) is required only when the sizing variables of transition truss structural designs correspond to a yielding situation.

Moreover, regardless of whether or not the allowable stresses of members are varied with the change of sizing variables, the refining of stress constraints runs for one time only. Notably, the fitness values of heuristically generated solutions consider only the structural weight when attempting to reduce the computational burden of structural analysis.

3.2.2. Encoding

In the GA, encoding is a process that transforms variable vectors into chromosomes. Several encoding methods are available for the mapping from the solution space to the chromosome space, including binary encoding, real number encoding, integral encoding and literal permutation encoding. The selection of encoding method generally depends on the desired problems to be solved. This work selects binary encoding for the hill-climbing and greedy GA. For example,

Variable vector: $X = (z_1(a_1), z_2(a_2), \dots)$

Assigned number vector: $ANX = (a_1, a_2, \dots)$

Chromosome: $CX = s_1 s_2 \dots$

where $z_i(a_i)$ represents the a_i -th element of set of legal values corresponding to the i -th variable of X , a_i is the assigned number of i -th variable of X , and s_i is the binary string corresponding to a_i .

3.2.3. Hybrid selection

Selection of GAs focused on gradually improving populations. The roulette wheel selection is a commonly used selection method. The selection method randomly selects chromosomes from the previous generation, and the selection probability of any chromosome is positively proportional to its fitness value. The weakness of the selection method is that the best chromosome is possible to be lost due to the random

selection scheme. The $(\mu + \lambda)$ selection is another conventionally adopted selection method. The selection method is characterized by its capability to always reserve the fitter chromosomes. However, valuable information for identifying the new promising search regions may be lost if the bad chromosomes are always died. This work uses hybrid methods to select chromosomes from the previous generation. The selection rules are listed in the following:

- 10% chromosomes copied from the best chromosome
- 70% chromosomes selected by roulette wheel selection
- 20% immigration

Chromosomes for immigration should be produced independently of the parent population. Random generation is the most commonly used method for producing immigration for GAs. However, this work uses heuristic initialization to produce immigration in the initial stage of the genetic search (first 20 iterations) and random immigration in a later stage.

3.2.4. Crossover and Mutation

After the selection is finished, crossover and mutation operators are used to generate next generation of populations. The crossover operator is used to exchange information between two chromosomes. The notion of crossover is that the new chromosome may be better than both of the parents if it takes the best characteristics from each of the parents. The mutation operator is used to add some changes to offspring chromosomes. The aim of mutation is to allow the GA to avoid local minima by preventing the population of chromosomes from becoming too similar to each other. In this work, the uniform crossover operator and the point mutation operator are adopted.

3.2.5. Hill-climbing strategy and modulus decoding method

Most GAs generally use the same coding method to encoding variable vectors to chromosomes and decoding chromosomes to variable vectors. However, this work uses different encoding and decoding methods because a hill-climbing strategy is integrated into the GA. Hill-climbing search methods emphasize extreme exploitation. The kind of search methods assume that the optimal solution is located at somewhere between the visiting location and the location of current best solution. Hill-climbing search methods always have an extremely high convergence speed because they ignore exploring the search space. However, the search results by the hill-climbing search methods highly depend on the start point.

The GA always keeps balance between exploitation and exploration. That is why the GA can overcome local minimum problems. Also, that is why the GA often requires a long computational time to search for the optimal solution of optimization problems. This work attempts to integrate a hill-climbing strategy into the GA to reduce the computational time for finding the optimal solution of discrete truss structural optimization problems. The hill-climbing strategy assumes that the center of the most promising search region is located at the location of current best solution; in addition, the most promising search region is gradually reduced whenever the best solution is improved. Figure 3-2 uses an example to further explain the hill-climbing strategy. Suppose a discrete optimization problem has p variables, and the i -th variable of the problem is linked to a set of 11 ranked discrete values (Z_i). The fat solid circle represents the search space, the thin solid circle represents the most promising search region at the k -iteration, and the dashed circle represents the most promising search region at the $(k+1)$ -iteration. Additionally, windows W_i^k and W_i^{k+1} are designed to produce the subset of discrete values corresponded to the i -th variable at the k -iteration and $(k+1)$ -iteration, respectively. At the k -iteration, the value of the

i -th variable is selected from $\{z_{i,2}, z_{i,3}, z_{i,4}, z_{i,5}, z_{i,6}, z_{i,7}\}$, where the middle element $z_{i,5}$ is corresponded to the value of the i -th variable of the best solution $(x_1^k, x_2^k, \dots, x_i^k = z_{i,5}, \dots, x_p^k)$, and the window size is 6 ($|W_i^k|^2 = |(0, 1, 1, 1, 1, 1, 0, 0, 0, 0)|^2 = 6$). If the best solution is moved from the old location to a new location $(x_1^{k+1}, x_2^{k+1}, \dots, x_i^{k+1} = z_{i,7}, x_p^{k+1})$ after the k -th iteration, the following subset $\{z_{i,5}, z_{i,6}, z_{i,7}, z_{i,8}, z_{i,9}\}$ is proposed for the i -th variable at the $(k+1)$ -iteration. The middle element $z_{i,7}$ of the new subset of discrete values is corresponded to the value of i -th variable of the new best solution, and the window size is decreased from 6 to 5 ($|W_i^{k+1}|^2 = |(0, 0, 0, 0, 1, 1, 1, 1, 0, 0)|^2 = 5$).

How to decrease the number of element of subsets for variables is an important issue for the hill-climbing GA. The solution may converge to a locally optimum if the hill-climbing strategy reduces the search space too fast. Herein, a three-level window size rule is proposed to handle the window size.

If iteration number of genetic search is smaller than a lower bound value (δ), then

$$Wsize(k) = Ws_{\max} \quad (3-5)$$

Else if $Round(\phi^k) \geq Ws_{\min}$, then

$$Wsize(k) = Round(\phi^k) \quad (3-6)$$

$$\phi^k = \begin{cases} \phi^{k-1} \times \alpha & \text{if the best solution is improved} \\ \phi^{k-1} \times \alpha & \text{else if the best solution is unchanged for } \beta \text{ generations,} \\ \phi^{k-1} & \text{else} \end{cases}$$

$$(\phi^1 = W_{\max}) \quad (3-7)$$

Else if $Round(\phi^k) < Ws_{\min}$, then

$$Wsize(k) = Ws_{\min} \quad (3-8)$$

where α is a real number in the interval of [0.9, 1], β is a positive integer number for forcing ϕ^k to be modified, Ws_{\max} is the maximum window size, and Ws_{\min} is the minimum window size. Basically, the more complex the optimization problems, the larger the value of α . By the three-level window size rule, the reduction speed for narrowing down search space can be easily adapted by changing the parameter α .

Due to the reduced search space, some chromosomes are possible to be outside the most promising search region. For example, a chromosome selected from the parent population is inside the previous most promising search region, but may be outside the new most promising search region. Moreover, the crossover and mutation operators are possible to produce offspring to be outside the most promising search region. To overcome the problem, a modulus decoding method is developed to map offspring from the chromosome space into the solution space. Figure 3-3 uses an example to illustrate the decoding method. The 2nd, 3rd and 4th elements are 1 and others are 0 in the window W_i . The set of legal values is ranked from small to large. The set of promising values is generated by W_i and the set of legal values. As indicated from figure 2, the $(2+3a)$ -th elements are set as $z_i(2)$, the $(3+3a)$ -th elements are set as $z_i(3)$, and the $(4+3a)$ -th elements are set as $z_i(4)$ in the set of promising values of i -th variable, where a is an integer. Therefore, no matter what value of i -th gene is, the gene is always mapped into the promising values. The following formulas are defined for modular decoding:

$$\text{Suppose that, } W_i(j) = \begin{cases} 0 & j = 1 \text{ to } w_{\text{left}} - 1 \\ 1 & j = w_{\text{left}} \text{ to } w_{\text{right}} \\ 0 & j = W_{\text{right}} + 1 \text{ to } pi \end{cases} \quad (3-9)$$

$$Z_i^{\text{promising}}(j) = Z_i(l) \quad (3-10)$$

$$l = w_{left} + \text{mod}(j - w_{left}, w_{right} - w_{left} + 1) \quad (3-11)$$

where w_{left} is the assigned number corresponded to the first element 1 of window W_i , w_{right} is the assigned number corresponded to the last element 1 of window W_i , p_i is the number legal values of i -th variable, $Z_i^{promising}$ is the set of promising values of i -th variable, Z_i is the set of legal values of i -th variable.

3.2.6. Flow chart of the hill-climbing and greedy GA

Figure 3-4 shows the flow chart of the hill-climbing and greedy GA. The hill-climbing and greedy GA starts from the process of heuristic initialization. The window size is then set to the maximum values, and an even number of heuristically generated truss structural designs forms an initial population. Next, the fitness values of heuristically generated truss structural designs are calculated by the evaluation process, and the heuristically generated truss structural designs are then transformed into chromosomes by the encoding process. The hybrid selection process produces the same number of chromosomes by selecting from the population and immigration. The survival chromosomes then exchange information by the crossover process, and some genes of modified chromosomes from the crossover process are changed by the mutation process. After the two processes are calculated, an offspring population is generated. The chromosomes of the offspring population should be transformed into variable vectors by the modulus decoding process to evaluate their fitness values. If the best solution is improved, the window size is adapted according to the three-level window size rule, and the centers of windows of variables are moved along with the value of variables corresponding to the best solution. However, if the best solution is unchanged for a certain number (β) of generations, the window adaption process is forced to update windows. The computation for searching the optimal solution will

stop if the stopping criteria are met. Otherwise, the offspring population is sent to the encoding process to repeat the next iteration of genetic search.



Chapter 4 Numerical examples

A greedy GA (GGA, greedy notions includes the heuristic initialization and the hybrid selection) and a hill-climbing and greedy GA (HGGA) were tested by several examples to discuss their search performance, and a simple GA (SGA) was used as a reference model to solve the same problems. Several working parameters were set to the same values for the three GAs. The maximum iteration number was 500; the population size was 50; the crossover probability is 100% for each chromosome; the mutation probability is 5% for each gene. Additionally, each GA was run 10 times by using 10 sets of independent initial weights to obtain more observations.

4.1. 10-bar plane truss structure

A 10-bar plane truss structure was adopted as an example for the performance test of the GGA and HGGA, and the example has been discussed in references [1, 5 and 17]. Figure 4-1 shows the topology of the truss structures. The elastic modulus was 10^4 Ksi ($6.89 \times 10^4 \text{ MPa}$); the weight per unit volume was $0.11b$ (2774 kgw/m^3); the allowable tension and compression stresses were 25 Ksi (172.25 MPa) and -25 Ksi (-172.25 MPa), respectively; the allowable displacements in both x and y directions were 0.0508 m (2 in); 100 kips (4.5 kN) concentrated loads were imposed on pin-2 and pin-4 in y -direction. Additionally, each member of the 10-bar plane truss structure was linked to an individual cross-sectional area. That is, this example has 10 sizing variables.

Case I:

The value of sizing variables was selected from the set $Z=\{0.1, 0.5, 1.0, 1.5, 2.0, \dots, 29.5, 30.0, 30.5, 31.0, 31.5\}$ (in^2). Therefore, the search space of this case contains 64^{10} data points. The working parameters of the hill-climbing strategy for this case were set as follows: the maximum window size was 32; the minimum

window size was 7; the values of α , β and δ for three-level window size rule are 0.95, 20 and 20, respectively. Table 1 lists the computational results of this example. The HGGA shows the best search performance. The weight of the best solution found by the HGGA is 5067.33 *lb*. However, the weights of the best solutions found by the SGA and GGA are 5221.86 *lb* and 5092.79 *lb*, respectively. The average weight and the standard deviation are used as tools to estimate the stability of the three GAs. The average weights of 10-run solutions corresponding to the SGA, GGA, and HGGA are 5635.40 *lb*, 5266.88 *lb* and 5077.64 *lb*, respectively. The standard deviations of 10-run solutions corresponding to the SGA, GGA, and HGGA are 359.08 *lb*, 186.50 *lb* and 4.58 *lb*, respectively. The HGGA obviously has the best stability. Figure 4-2 shows the comparison of convergence rates for this case. The three GAs have the similar convergence speed in the early stage of search. After 70 iterations, the search performance of the SGA is obviously slow down. On the other hand, the GGA and HGGA continuously improve their best solutions in the later stage of search. Table 2 shows the comparison of the optimal design results of this case. The solutions corresponding to the HGGA and reference [5] have the lightest weight.

Case II:

For obtaining more observations, a new set of legal discrete values was used for all variables to test the three GAs again. The new set was $Z^* = \{0.1, 0.5, 1.0, 3.0, 4.0, 5.5, 7.0, 7.5, 8.0, 8.5, 10.0, 12.0, 13.5, 14.0, 14.5, 15.5, 17.0, 19.0, 20.0, 20.5, 21.0, 21.5, 22.0, 22.5, 23.0, 23.5, 24.0, 24.5, 26.0, 29.0, 30.0, 31.0\}$. The number of data points of the search space in case I was 1024 times than that of the new search space, and the new search space still contains the best solution that was found in case I. The working parameters of the hill-climbing strategy for this case were set as follows: the maximum window size was 32; the minimum window size was 7; the values of α ,

β and δ for three-level window size rule are 0.95, 20 and 20, respectively. Table 1 and 2 also lists the computational results of this case. The best solutions found by the SGA, GGA and HGGGA are 5088.42 *lb*, 5089.70 *lb* and 5067.33 *lb*, respectively. The search result of the SGA is obviously better than that in case I. However, the SGA still has a poor stability, even if the search space is smaller. Figure 4-3 shows the comparison of convergence rates for this case. The weight curves corresponding to three GAs are similar each other.

4.2. 25-bar space truss structure

A 25-bar space truss structure shown in figure 4-4 was adopted as an example for the performance test of the GGA and HGGGA, and the example has been discussed in references [1, 2 and 17]. In the example, the elastic modulus was 10^4 *Ksi* (6.89×10^4 *MPa*); the weight per unit volume was 0.11*lb* (2774 kgw/m^3); the allowable tension and compression stresses were 40 *Ksi* (275.6 *MPa*) and -40 *Ksi* (-275.6 *MPa*), respectively; the allowable displacements in *x*, *y* and *z* directions were 0.35 *inch* (0.00889 *m*).

Case I:

Four joint loads were imposed to the 25-bar space truss structure. Joint loads: (1 *Kip*, -10 *Kips*, -10 *Kips*) at node 1, (0 *Kip*, -10 *Kips*, -10 *Kips*) at node 2, (0.5 *Kip*, 0 *Kip*, 0 *Kip*) at node 3, and (0.6 *Kip*, 0 *Kips*, 0 *Kips*) at node 6. Table 3 shows the sizing variables. The 25 structural members were divided into 8 groups. The value of sizing variables was selected from the set $Z = \{0.1, 0.2, 0.3, \dots, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4\}$ (*in*²). Therefore, the search space of this case contains 30^8 data points. The working parameters of the hill-climbing strategy for this case were set as follows: the maximum window size was 30; the minimum window size was 7; the values of α ,

β and δ for three-level window size rule are 0.95, 20 and 20, respectively. Table 4 lists the computational results of this case. The best solutions found by the SGA, GGA and HGGA are 487.07 *lb*, 485.05 *lb*, and 484.85 *lb*, respectively. The average weight of 10-run solutions corresponding to the SGA, GGA, and HGGA are 504.37 *lb*, 488.39 *lb* and 485.99 *lb*, respectively. The standard deviation of 10-run solutions corresponding to the SGA, GGA, and HGGA are 12.75 *lb*, 4.80 *lb* and 1.44 *lb*, respectively. The HGGA has the best search performance and stability. Figure 4-5 shows the comparison of convergence rates for this case. The convergence speeds of the GGA and HGGA are obviously better than that of the SGA. The GGA found an approximate optimal solution in the early stage search. The solution, however, was not improved in the later stage of search. On the other hand, the HGGA successfully found the optimal solution in the early stage of search. Table 5 shows the comparison of the optimal design results of this case. The solutions corresponding to the reference [17] and HGGA have the lightest weight.

Case II:

The 25-bar space truss structure was subjected to another load conditions. Load condition 1: (0 *Kip*, 20 *Kips*, -5 *Kips*) at node 1 and (0 *Kip*, -20 *Kips*, -5 *Kips*) at node 2; load condition 2: (1 *Kip*, 10 *Kips*, -5 *Kips*) at node 1, (0 *Kip*, 10 *Kips*, -5 *Kips*) at node 2, (0.5 *Kip*, 0 *Kip*, 0 *Kip*) at node 3, and (0.5 *Kip*, 0 *Kips*, 0 *Kips*) at node 6. The value of sizing variables is selected from the set $Z=\{0.01, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.8, 5.2, 5.6, 6.0\}$ (*in*²). Therefore, the search space of the example contains 16⁸ data points. The working parameters of the hill-climbing strategy for this case were set as follows: the maximum window size was 16; the minimum window size was 7; the values of α , β and δ for the three-level window size rule are 0.95, 20 and 20, respectively. Table 4 also lists the computational results of this example.

The best solutions found by the SGA, GGA and HGGA are 576.24 *lb*, 560.59 *lb*, and 560.59 *lb*, respectively. The average weights of 10-run solutions corresponding to the SGA, GGA, and HGGA are 650.91 *lb*, 564.25 *lb* and 561.40 *lb*, respectively. The GGA and HGGA found the same solution. The standard deviation of 10-run solutions corresponding to the SGA, GGA, and HGGA are 72.13 *lb*, 4.40 *lb* and 2.56 *lb*, respectively. The HGGA has the best stability. Figure 4-6 shows the comparison of convergence rates for this case. The convergence speeds of the GGA and HGGA are obviously better than that of the SGA. Table 5 also shows the comparison of the optimal design results of this case. The solutions corresponding to the reference [17], GGA and HGGA have the lightest weight.

Case III

This case considered the buckling problem. The load condition, the set of discrete legal values and the working parameters for the hill-climbing strategy were the same as that in case I. The buckling stress function was defined as follows:

$$\sigma_{j,cr} = \frac{-12.5A_j}{L_j^2} \quad j=1 \text{ to } 25 \quad (4-1)$$

Table 4 also lists the computational results of this case. The best solutions found by the SGA, GGA and HGGA are 525.11 *lb*, 514.30 *lb*, and 514.30 *lb*, respectively. The average weights of 10-run solutions corresponding to the SGA, GGA, and HGGA are 540.35 *lb*, 520.15 *lb* and 518.42 *lb*, respectively. The standard errors of 10-run solutions corresponding to the SGA, GGA, and HGGA are 12.76 *lb*, 4.20 *lb* and 4.28 *lb*, respectively. The search performance and stability of the GGA and HGGA are better than that of the SGA. Additionally, figure 4-7 shows the comparison of convergence rates for this case, and table 5 also lists the comparison of the optimal design results of this case.

4.3. 72-bar space truss structure

A 72-bar space truss structure shown in figure 4-8 was adopted as an example for the performance test of the GGA and HGGA, and the example has been discussed in the reference [17]. In the example, the elastic modulus was 10^4 Ksi (6.89×10^4 MPa); the weight per unit volume was 0.1 lb (2774 kgw/m^3); the allowable tension and compression stresses were 25 Ksi (172.25 MPa) and -25 Ksi (-172.25 MPa), respectively; the allowable displacements in x , y and z directions were 0.25 inch (0.00635 m). The 72-bar space truss structure was subjected to two load conditions. Load condition 1: (5 Kip, 5 Kips, -5 Kips) at node 17; load condition 2: (0 Kip, 0 Kips, -5 Kips) at node 17, 18, 19 and 20. Table 6 shows the sizing variables. The 72 structural members were divided into 16 groups. The value of sizing variables is selected from the set $Z = \{0.1, 0.2, 0.3, 0.4, 0.5, \dots, 2.8, 2.9, 3.0, 3.1, 3.2\}$ (in^2). Therefore, the search space of the example contains 32^{16} data points.

The working parameters of the hill-climbing strategy for this example were set as follows: the maximum window size was 32; the minimum window size was 7; the values of α , β and δ for the three-level window size rule are 0.95, 20 and 20, respectively. Table 7 lists the computational results of this example. The best solutions found by the SGA, GGA and HGGA are 406.02 lb, 388.08 lb, and 385.54 lb, respectively. The average weights of 10-run solutions corresponding to the SGA, GGA, and HGGA are 432.93 lb, 400.73 lb and 390.69 lb, respectively. The standard deviations of 10-run solutions corresponding to the SGA, GGA, and HGGA are 17.65 lb, 12.21 lb and 4.03 lb, respectively. The HGGA has the best search performance and stability. Figure 4-9 shows the comparison of convergence rates for this example. The weight curves corresponding to the GGA and HGGA are very much closed. Table 8

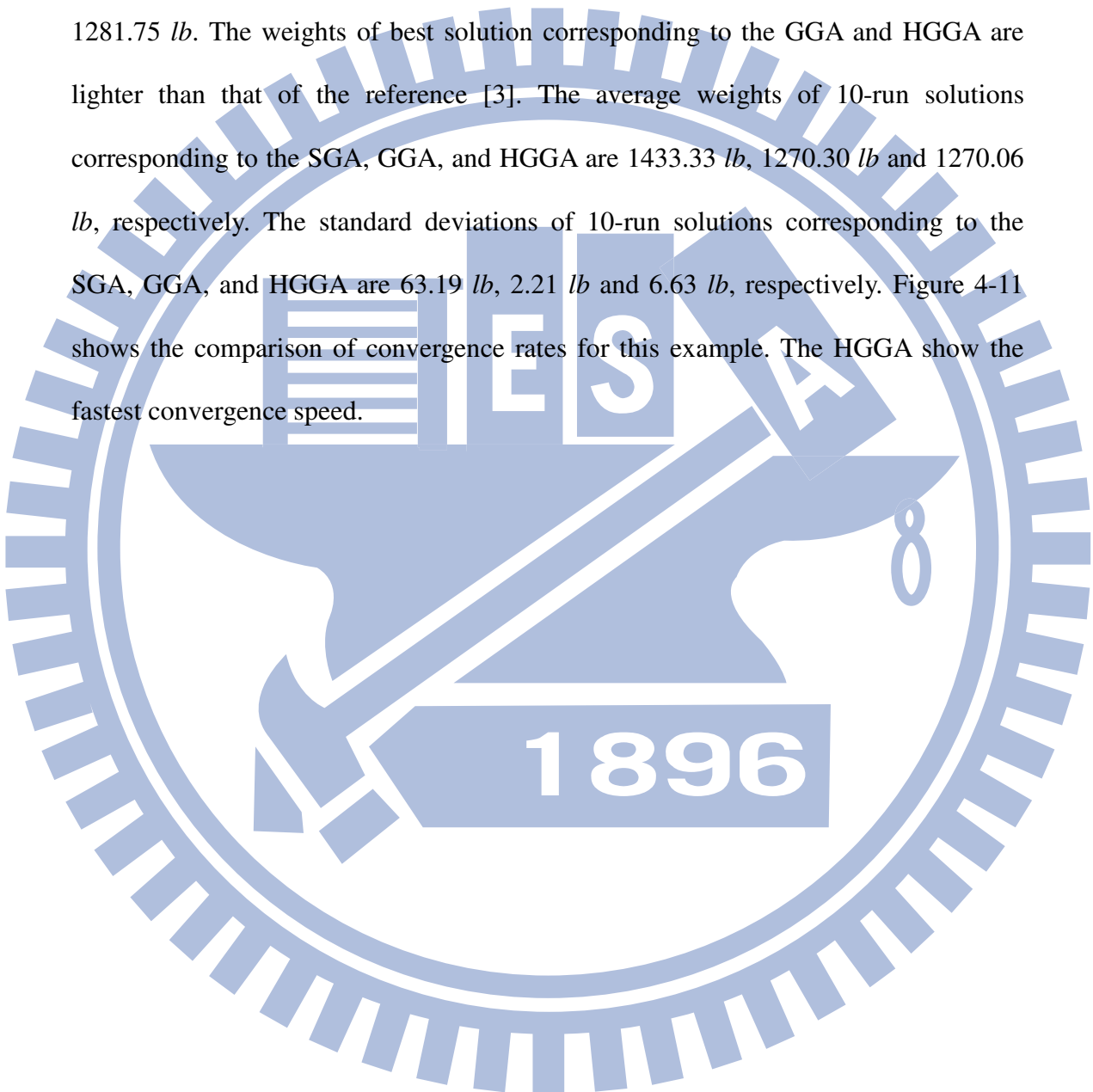
also shows the comparison of the optimal design results of this example. The solution corresponding to the HGGA has the lightest weight.

4.4. 160-bar space truss structure

A 160-bar space truss structure shown in figure 4-10 was adopted as the last example for the performance test of the GGA and HGGA, and the example has been discussed in the reference [3]. In the example, the elastic modulus was 3×10^4 Ksi (2.05×10^4 MPa); the weight per unit volume was 0.28 lb (7850 kgw/m^3); the allowable tension and compression stresses were 21.36 Ksi (147.15 MPa) and -21.36 Ksi (-147.15 MPa), respectively; a displacement limit of 3.94 in (0.1 m) was imposed on nodes E, F, G and D in x , y and z directions. The 160-bar space truss structure was subjected load as follows: (-2.4 Kip, 0 Kips, -1.2 Kips) at node E, (-2.4 Kip, 0 Kips, -1.2 Kips) at node F, (-2.2 Kip, 0 Kips, -1.2 Kips) at node G, (-1.9 Kip, 0 Kips, -1.1 Kips) at node D. The 160 structural members were classified into 16 groups. Group 1: the 24 members belonging to the four columns as shown in Figure 13 by AB; group 2: the 12 members belonging to the four columns (BC); group 3: the 12 members belonging to the four columns which make the cone of the tower; group 4: the 8 members which are the two wings; group 5: the 4 members belonging to the upper left wing; group 6~16: cross bars building-up each level. The value of sizing variables is selected from the set $Z = \{0.234, 0.266, 0.434, 0.484, 0.563, 0.517, 0.621, 0.688, 0.715, 0.813, 0.938, 1.15, 1.36, 0.902, 1.09, 1.19, 1.44, 1.46, 1.69, 1.73, 1.78, 1.94, 2.09, 2.11, 2.25, 2.40, 2.43, 2.48, 2.75, 2.86, 2.87, 3.25\}$ (in^2). Therefore, the search space of the example contains 32^{16} data points.

The working parameters of the hill-climbing strategy for this example were set as follows: the maximum window size was 32; the minimum window size was 7; the

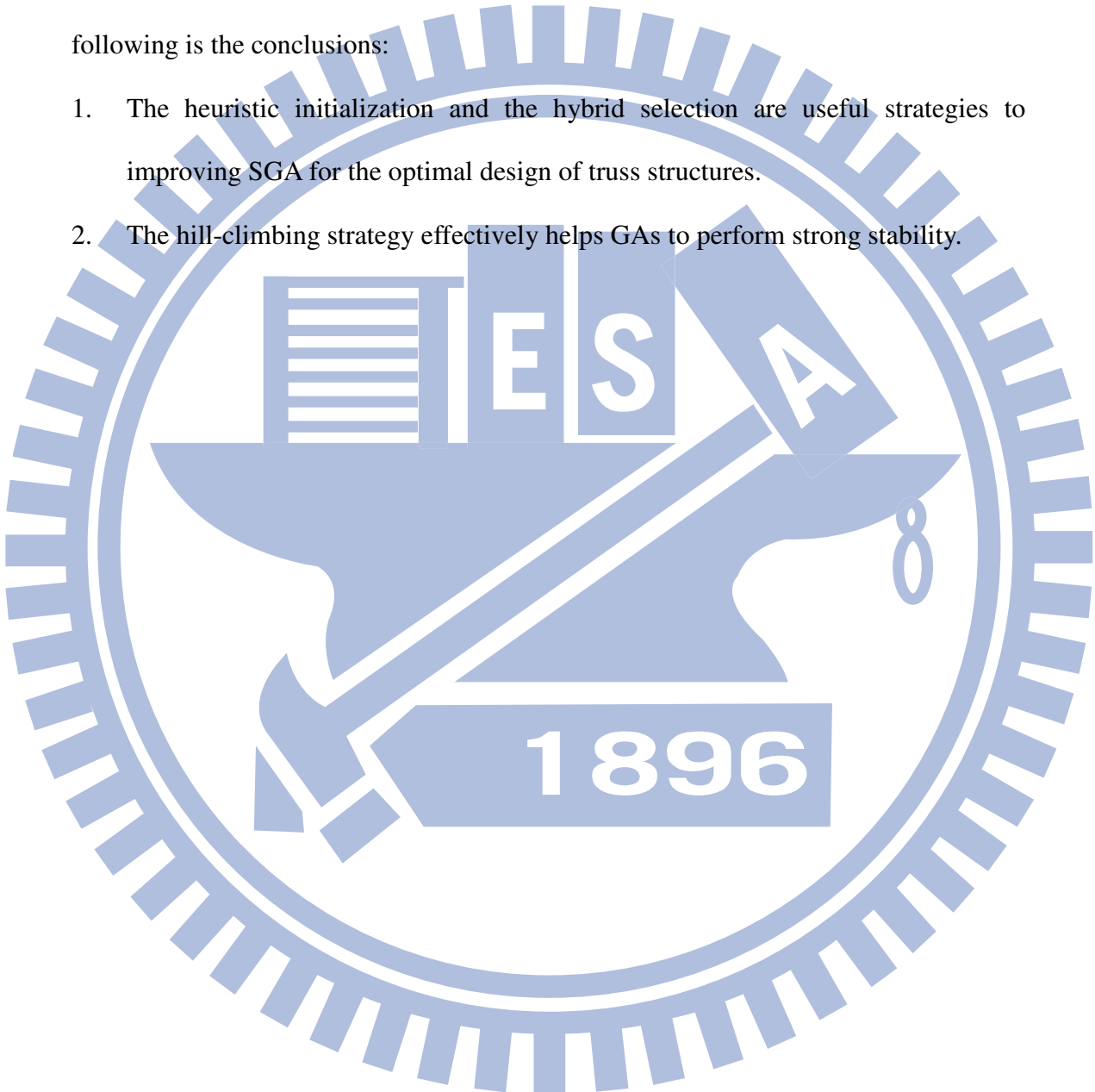
values of α , β and δ for the three-level window size rule are 0.95, 20 and 20, respectively. Table 9 lists the computational results of this example. The best solutions found by the SGA, GGA and HGGA are 1363.37 *lb*, 1265.05 *lb*, and 1265.05 *lb*, respectively. Galante's paper [3] indicated that their best solution had the weight of 1281.75 *lb*. The weights of best solution corresponding to the GGA and HGGA are lighter than that of the reference [3]. The average weights of 10-run solutions corresponding to the SGA, GGA, and HGGA are 1433.33 *lb*, 1270.30 *lb* and 1270.06 *lb*, respectively. The standard deviations of 10-run solutions corresponding to the SGA, GGA, and HGGA are 63.19 *lb*, 2.21 *lb* and 6.63 *lb*, respectively. Figure 4-11 shows the comparison of convergence rates for this example. The HGGA show the fastest convergence speed.



Chapter 5 Conclusions

This work presents a hill-climbing and greedy GA (HGGA) for the optimal design of truss structures with discrete sizing variables. Four benchmark problems have been used to test the performance of HGGA. Based on the above results, the following is the conclusions:

1. The heuristic initialization and the hybrid selection are useful strategies to improving SGA for the optimal design of truss structures.
2. The hill-climbing strategy effectively helps GAs to perform strong stability.



Reference

- [1] Rajeev S, Krishnamoorthy C. Discrete optimization of structures using genetic algorithm. *Journal of Structural Engineering* 1992; 118(5): 1233-50.
- [2] Wu S, Chow P. Integrated discrete and configuration optimization of truss using genetic algorithms, *Computers & Structures* 1995; 55(4): 695-702.
- [3] Galante M. Genetic algorithms as an approach to optimize real world trusses, *International Journal for Numerical Methods in Engineering* 1996; 39: 361-82.
- [4] Erbaturo F, Hasancebi O, Tutuncu I, Kilic H. Optimal design of plane and space structures with genetic algorithms, *Computers & Structures* 2000; 75(2000): 209-24.
- [5] Kaveh A, Kalatjari V. Genetic algorithm for discrete-sizing optimal design of trusses using the force method, *International Journal for Numerical Methods in Engineering* 2002; 55: 55-72.
- [6] Tang W, Tong L, Gu Y. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables, *International Journal for Numerical Methods in Engineering* 2005; 62: 1737-62.
- [7] Gen M, Cheng R, Lin L. *Network models and optimization multiobjective genetic algorithm approach*. London: Springer-Verlag, 2008.
- [8] Fanni A, Marchesi M, Serri A, Usai M. A Greedy genetic algorithm for continuous variables electromagnetic optimization problems, *IEEE Transactions on Magnetics* 1997; 33(2): 1900-3.
- [9] Pyrz M, Zawidzka J. Optimal discrete truss design using improved sequential and genetic algorithm, *Engineering Computations* 2001; 18(8): 1078-90.
- [10] Chen R, Lu K, Yu S. A hybrid genetic algorithm approach on multi-objective of assembly planning problem, *Engineering Applications of Artificial Intelligence*

2002, 15: 447–57.

- [11] Hwang D, He R. Improving real-parameter genetic algorithm with simulated annealing for engineering problems, *Advances in Engineering Software* 2006; 37: 406–18.
- [12] Vellev S. An a adaptive genetic algorithm with dynamic population size for optimizing join queries, *International Book Series Information Science and computing Book 2-Advanced Research in Artificial Intelligence*, ITHEA press, Sofia Bulgaria; 82-8.
- [13] Perry M, Koh C, Choo Y. Modified genetic algorithm strategy for structural identification, *Computers and Structures* 2006;84(2006): 529-40.
- [14] Ma H, Gong D. Research on modified shifting balance Genetic Algorithms, *Journal of China University of Mining & Technology* 2007; 17(2): 188-92.
- [15] Lee S, Su S, Lee C. Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics, *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics* 2003; 33(1): 113-21.
- [16] Kennedy J, Eberhart R. Particle swarm optimization. *IEEE international Conference on Neural Networks*, vol. 4, IEEE Press; 1995. 1942-8.
- [17] Li L, Huang Z, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers & Structures* 2009; 87(2009): 435-43.
- [18] Koumouis V, Katsaras C. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE transaction s on Evolutionary Computation* 2006; 10(1): 19-28.

Table 1 The computational results of three GAs for the 10-bar plane truss structure

Run 10 times	Case I			Case II		
	SGA	GGA	HGGA	SGA	GGA	HGGA
Best	5221.86	5092.79	5067.33	5088.42	5089.70	5067.33
Average	5635.40	5266.88	5077.64	5437.367	5289.793	5084.303
Standard deviation	359.08	186.50	4.58	393.65	270.07	13.55

The weight unit is *lb*.

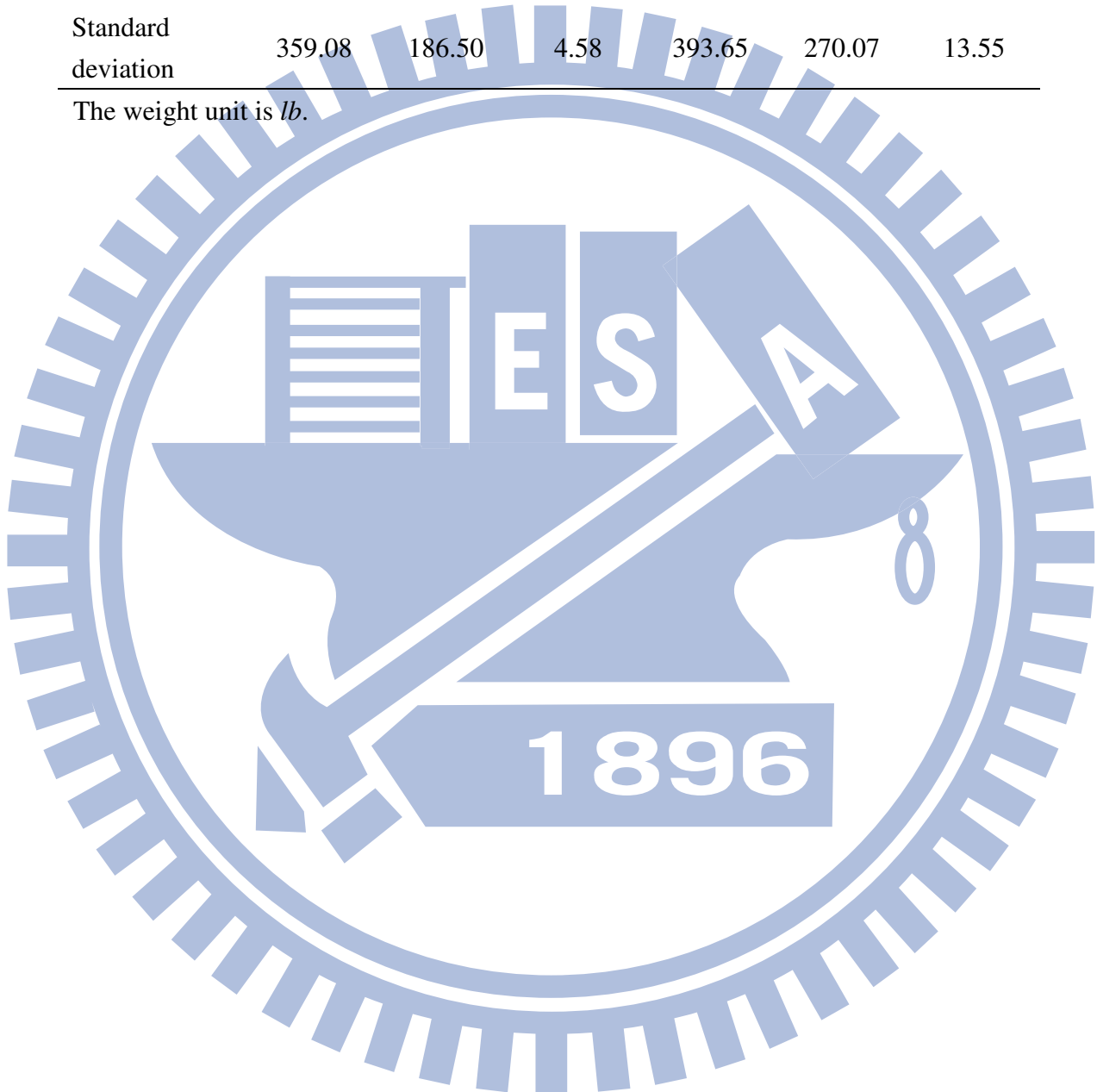


Table 2 The comparison of the optimal designs for the 10-bar plane truss structure

Sizing variable (in^2)	Case I				Case II	
	Reference		This work		This work	
	[5]	[17]	GGA	HGGA	GGA	HGGA
A_1	30.0	31.5	30.5	30.0	30.0	30.0
A_2	0.1	0.1	0.1	0.1	0.1	0.1
A_3	23.5	24.5	21.0	24.0	23.0	24.0
A_4	15.0	15.5	16.5	14.5	14.0	14.5
A_5	0.1	0.1	0.1	0.1	0.1	0.1
A_6	0.5	0.5	1.0	0.5	0.5	0.5
A_7	7.5	7.5	8.0	7.5	8.0	7.5
A_8	21.5	20.5	21.0	21.0	21.0	21.0
A_9	21.5	20.5	22.0	22.0	23.0	22.0
A_{10}	0.1	0.1	0.1	0.1	0.1	0.1
Weight (lb)	5067.33	5073.51	5092.79	5067.33	5089.70	5067.33

Table 3 The sizing variables for the 25-bar space truss structure

Sizing Variable	Members (End-End)
A_1	(1-2)
A_2	(1-4), (1-5), (2-3), (2-6)
A_3	(1-3), (1-6), (2-4), (2-5)
A_4	(3-6), (4-5)
A_5	(3-4), (5-6)
A_6	(3-10), (4-9), (5-8), (6-7)
A_7	(3-8), (4-7), (5-10), (6-9)
A_8	(3-7), (4-8), (5-9), (6-10)

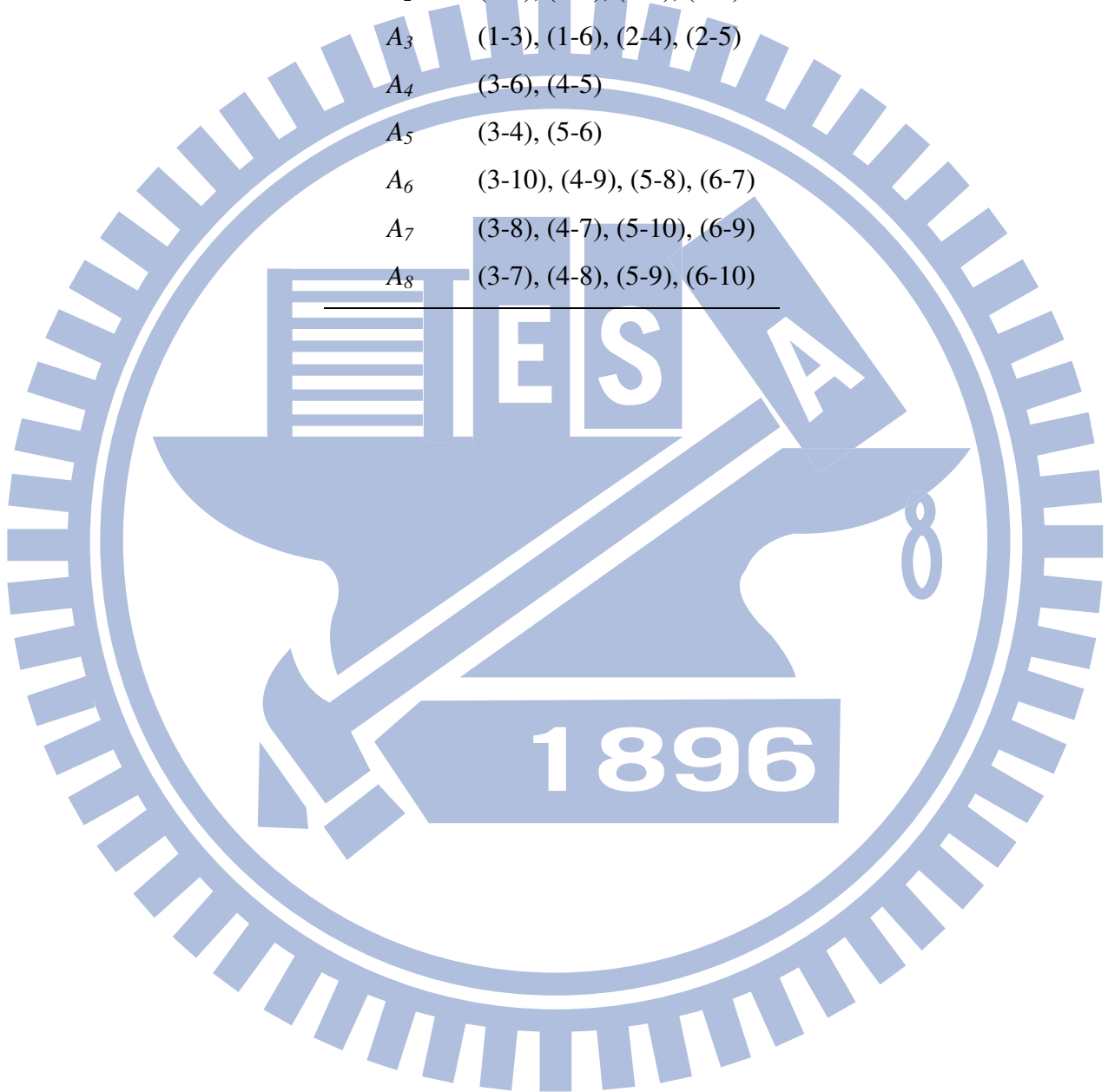


Table 4 The computational results of three GAs for the 25-bar space truss structure

Run 10 times	Case I			Case II			Case III		
	SGA	GGA	HGGA	SGA	GGA	HGGA	SGA	GGA	HGGA
Best	487.07	485.05	484.85	576.24	560.59	560.59	525.11	514.30	514.30
Average	504.37	488.39	485.99	650.91	564.25	561.40	540.35	520.15	518.42
Standard deviation	12.75	4.08	1.44	72.13	4.40	2.56	12.76	4.20	4.28

The weight unit is *lb*.

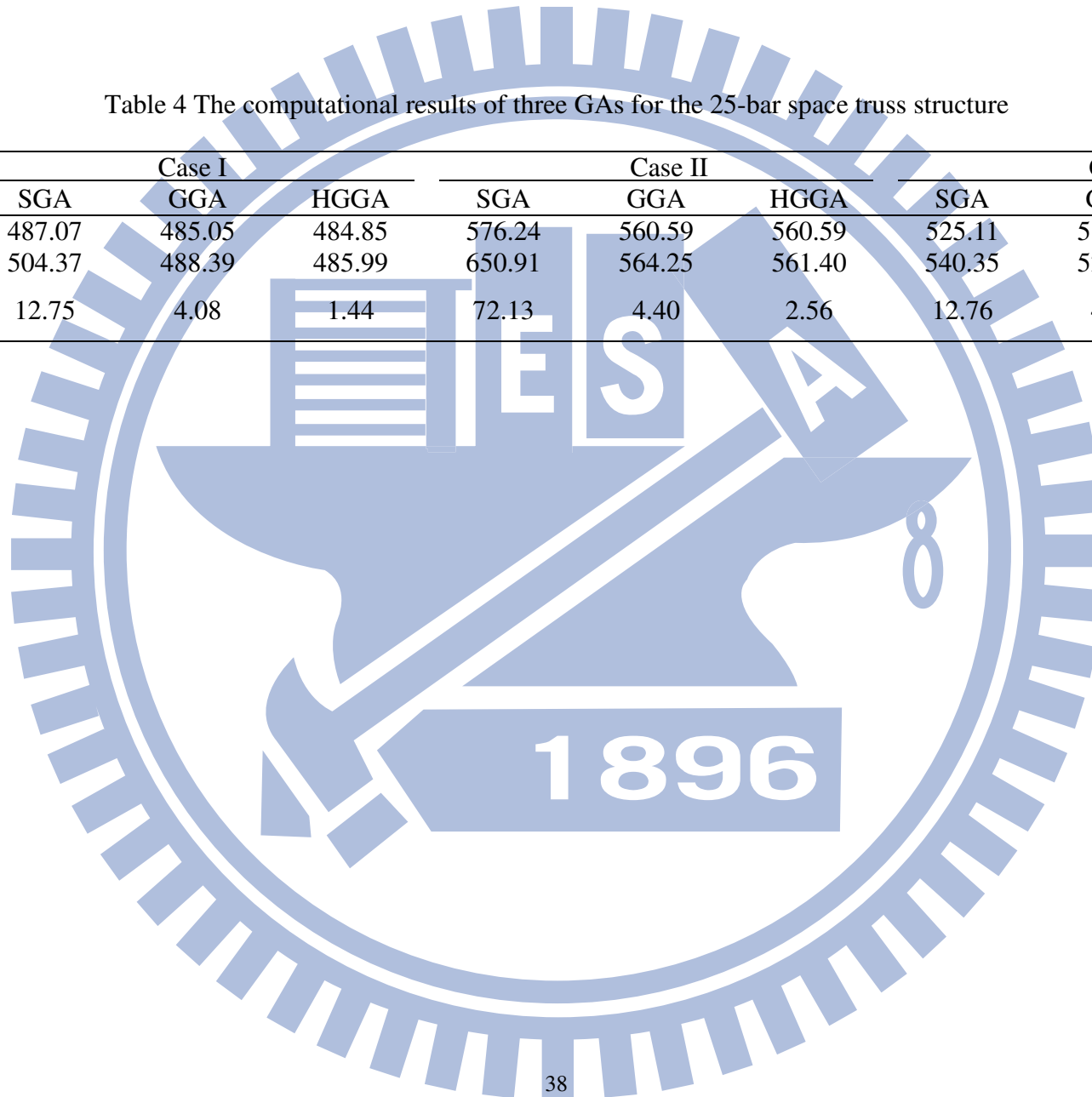


Table 5 The comparison of optimal designs for the 25-bar space truss structure

Sizing variable	Case I						Case II				Case III		
	Reference				This work		Reference		This work		Reference	This work	
	[1]	[4]	[5]	[17]	GGA	HGGA	[2]	[17]	GGA	HGGA	[2]	GGA	HGGA
A_1	0.1	0.1	0.1	0.1	0.1	0.1	0.4	0.01	0.01	0.01	0.3	0.1	0.1
A_2	1.8	1.2	0.4	0.3	0.5	0.3	2.0	2.0	2.0	2.0	0.9	0.7	0.7
A_3	2.3	3.2	3.4	3.4	3.4	3.4	3.6	3.6	3.6	3.6	3.0	3.2	3.2
A_4	0.2	0.1	0.1	0.1	0.1	0.1	0.01	0.01	0.01	0.01	0.4	0.1	0.1
A_5	0.1	1.1	2.2	2.1	1.9	2.1	0.01	0.01	0.01	0.01	1.0	0.8	0.8
A_6	0.8	0.9	1.0	1.0	1.0	1.0	0.8	0.8	0.8	0.8	1.1	1.1	1.1
A_7	1.8	0.4	0.4	0.5	0.4	0.5	2.0	1.6	1.6	1.6	1.2	1.2	1.2
A_8	3.0	3.4	3.4	3.4	3.4	3.4	2.4	2.4	2.4	2.4	3.0	3.0	3.0
Weight (<i>lb</i>)	546.01	493.80	484.33	484.85	485.05	484.85	563.52	560.59	560.59	560.59	525.20	514.30	514.30

Table 6 The sizing variables of the 72-bar space truss structure

Sizing Variable	Member (End-End)
<i>A1</i>	(1-5), (2-6), (3-7), (4-8)
<i>A2</i>	(1-6), (1-8), (2-5), (2-7), (3-6), (3-8), (4-7), (5-12)
<i>A3</i>	(5-6), (5-8), (6-7), (7-8)
<i>A4</i>	(5-7), (6-8)
<i>A5</i>	(5-9), (6-10), (7-11), (8-12)
<i>A6</i>	(5-10), (5-12), (6-9), (6-11), (7-10), (7-12), (8-11), (9-16)
<i>A7</i>	(9-10), (9-12), (10-11), (11-12)
<i>A8</i>	(9-11), (10-12)
<i>A9</i>	(9-13), (10-14), (11-15), (12-16)
<i>A10</i>	(9-14), (9-16), (10-13), (10-15), (11-14), (11-16), (12-15), (13-20)
<i>A11</i>	(13-14), (13-16), (14-15), (15-16)
<i>A12</i>	(13-15), (14-16)
<i>A13</i>	(13-17), (14-18), (15-19), (16-20)
<i>A14</i>	(13-18), (13-20), (14-17), (14-19), (15-18), (15-20), (16-19), (17-24)
<i>A15</i>	(17-18), (17-20), (18-19), (19-20)
<i>A16</i>	(17-19), (18-20)

Table 7 The computational results of three GAs for the 72-bar space truss structure

Run 10 times	SGA	GGA	HGGA
Best	406.02	388.08	385.54
Average	432.93	400.73	390.69
Standard deviation	17.65	12.21	4.03

The weight unit is *lb*.



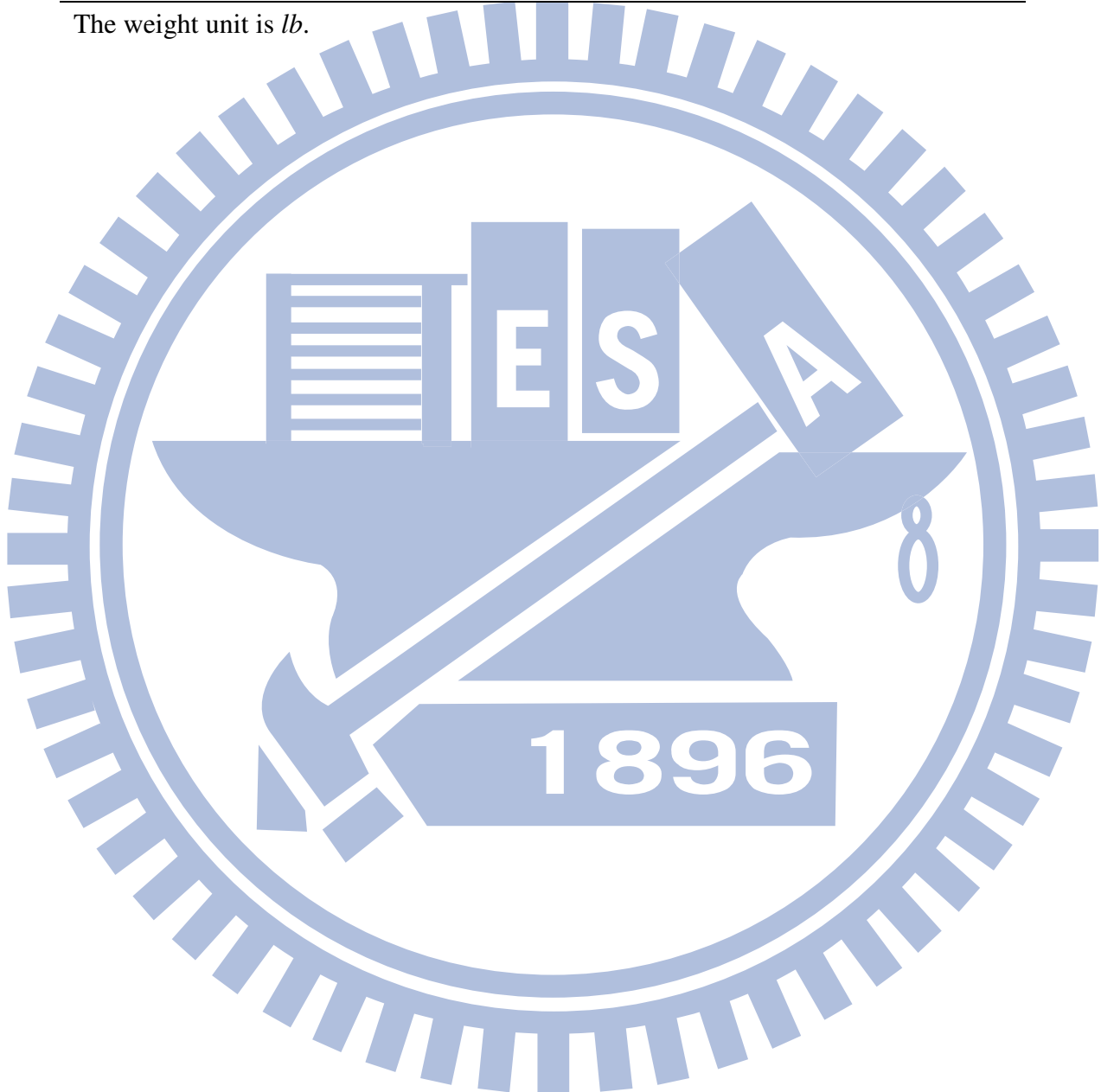
Table 8 The comparison of the optimal designs for the 72-bar truss structure

Sizing variable (in^2)	Reference		This work	
	[2]	[17]	GGA	HGGA
A_1	1.5	2.1	1.8	2.0
A_2	0.7	0.6	0.5	0.5
A_3	0.1	0.1	0.1	0.1
A_4	0.1	0.1	0.1	0.1
A_5	1.3	1.4	1.3	1.3
A_6	0.5	0.5	0.6	0.5
A_7	0.2	0.1	0.1	0.1
A_8	0.1	0.1	0.1	0.1
A_9	0.5	0.5	0.5	0.5
A_{10}	0.5	0.5	0.6	0.5
A_{11}	0.1	0.1	0.1	0.1
A_{12}	0.2	0.1	0.1	0.1
A_{13}	0.2	0.2	0.2	0.2
A_{14}	0.5	0.5	0.5	0.6
A_{15}	0.5	0.3	0.4	0.4
A_{16}	0.7	0.7	0.5	0.6
Weight (lb)	400.66	388.94	388.08	385.54

Table 9 The computational results of three GAs for the 160-bar space truss structure

Run 10 times	SGA	GGA	HGGA
Best	1363.67	1265.05	1265.05
Average	1433.33	1270.03	1270.26
Standard deviation	63.19	2.21	6.63

The weight unit is *lb*.



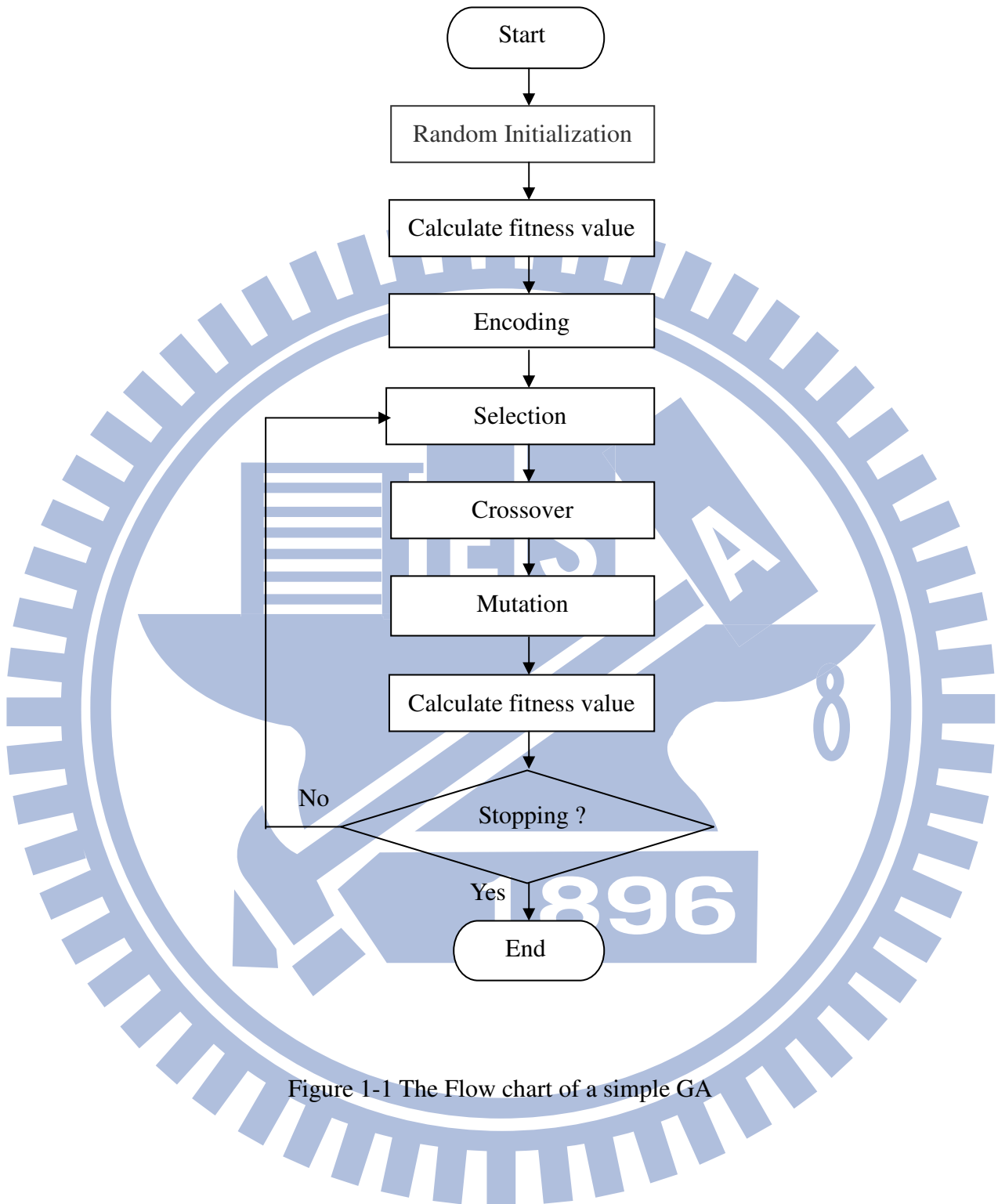


Figure 1-1 The Flow chart of a simple GA

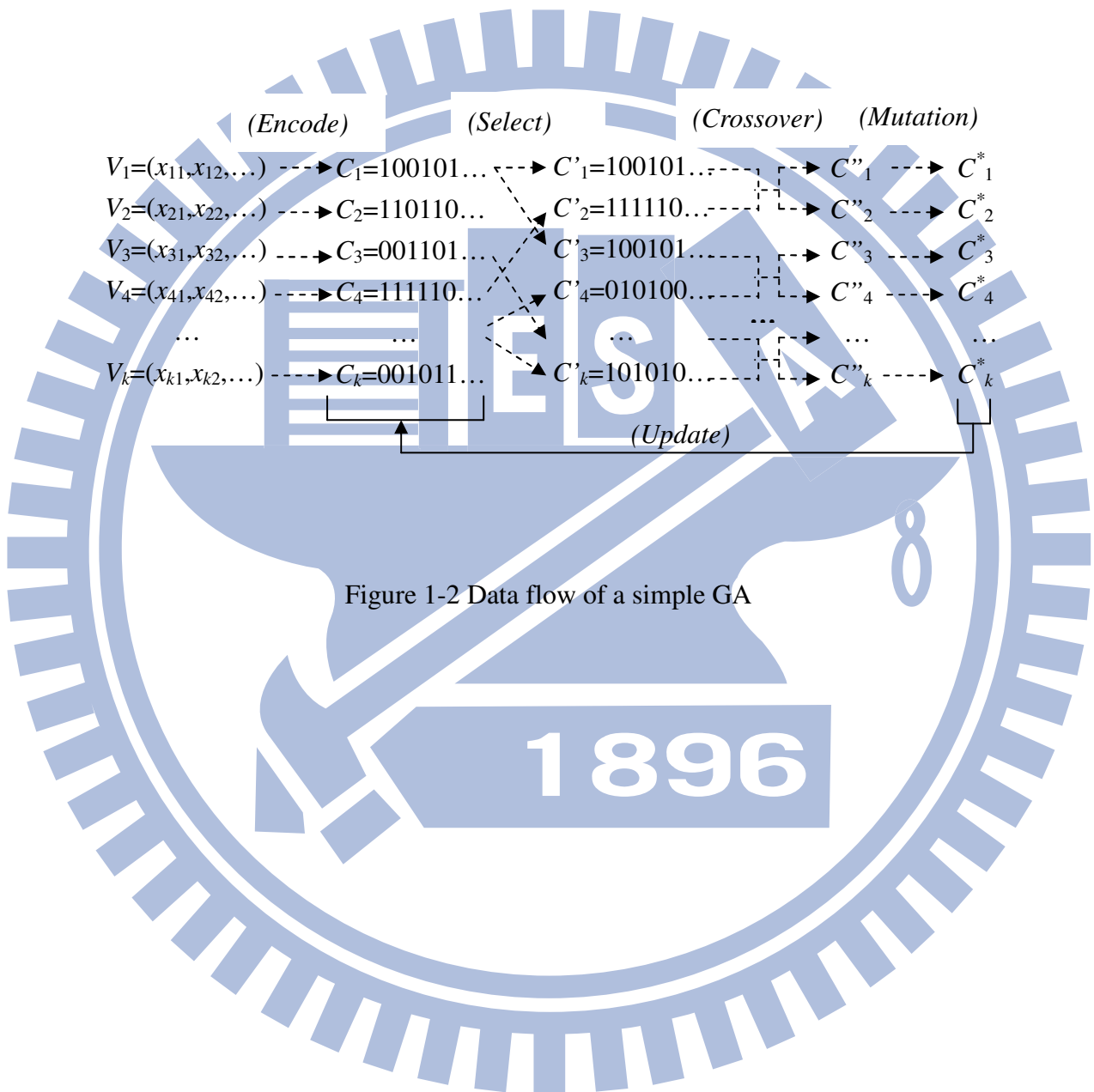


Figure 1-2 Data flow of a simple GA

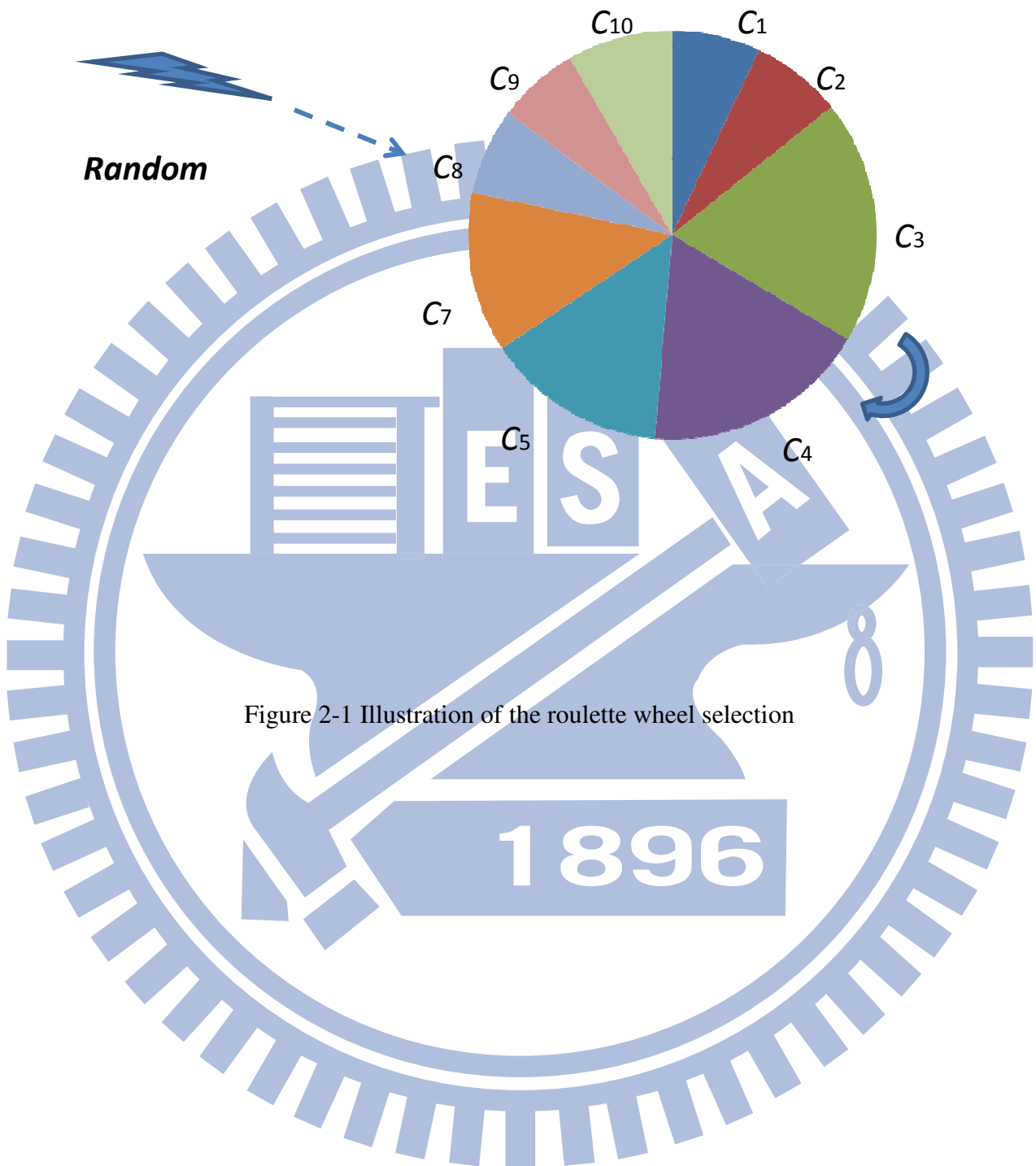


Figure 2-1 Illustration of the roulette wheel selection

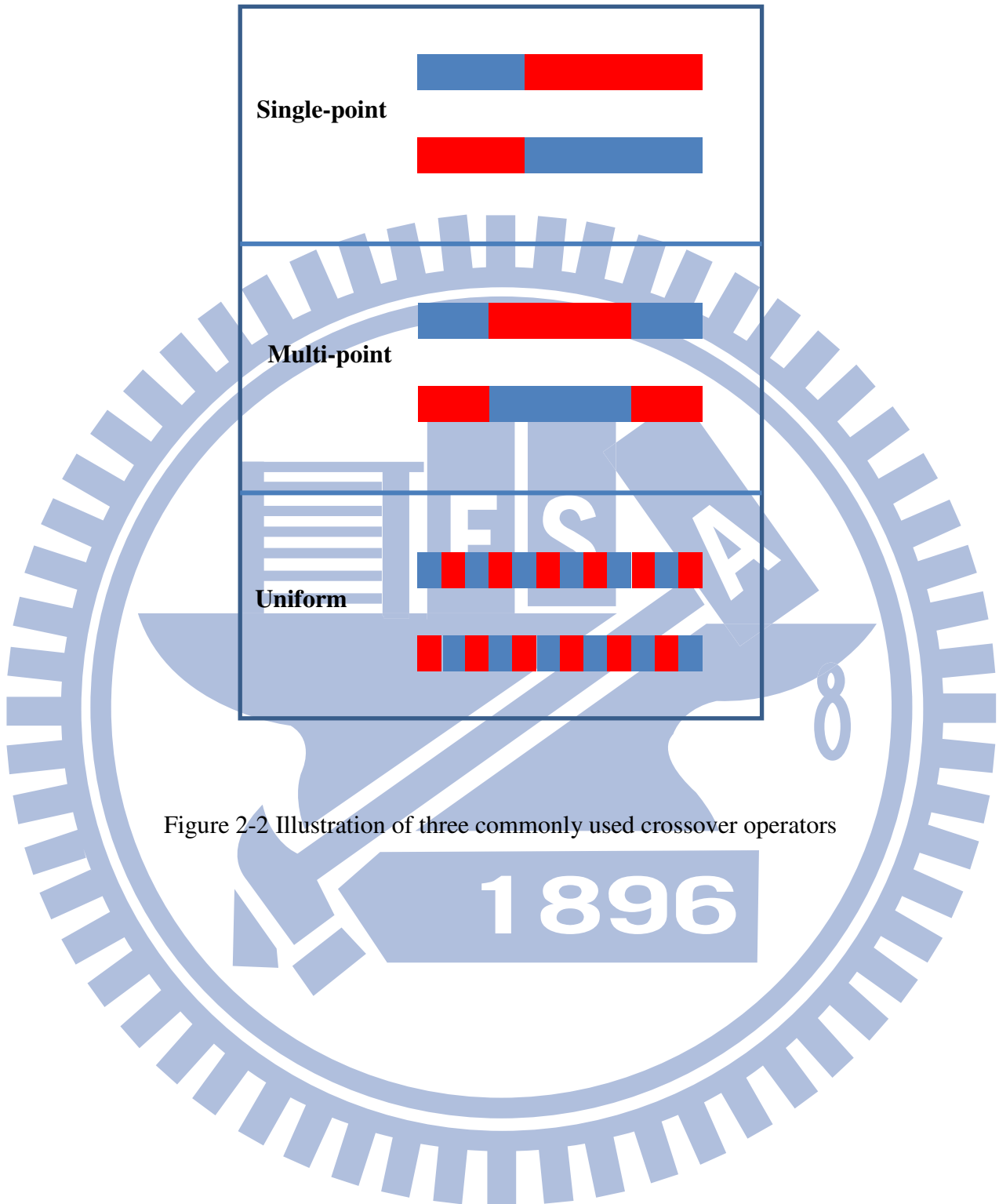


Figure 2-2 Illustration of three commonly used crossover operators

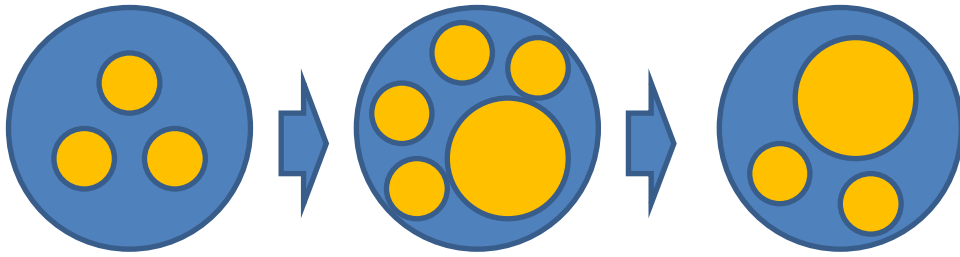


Figure 2-3 Illustration of Velly's dynamic population size GA

s

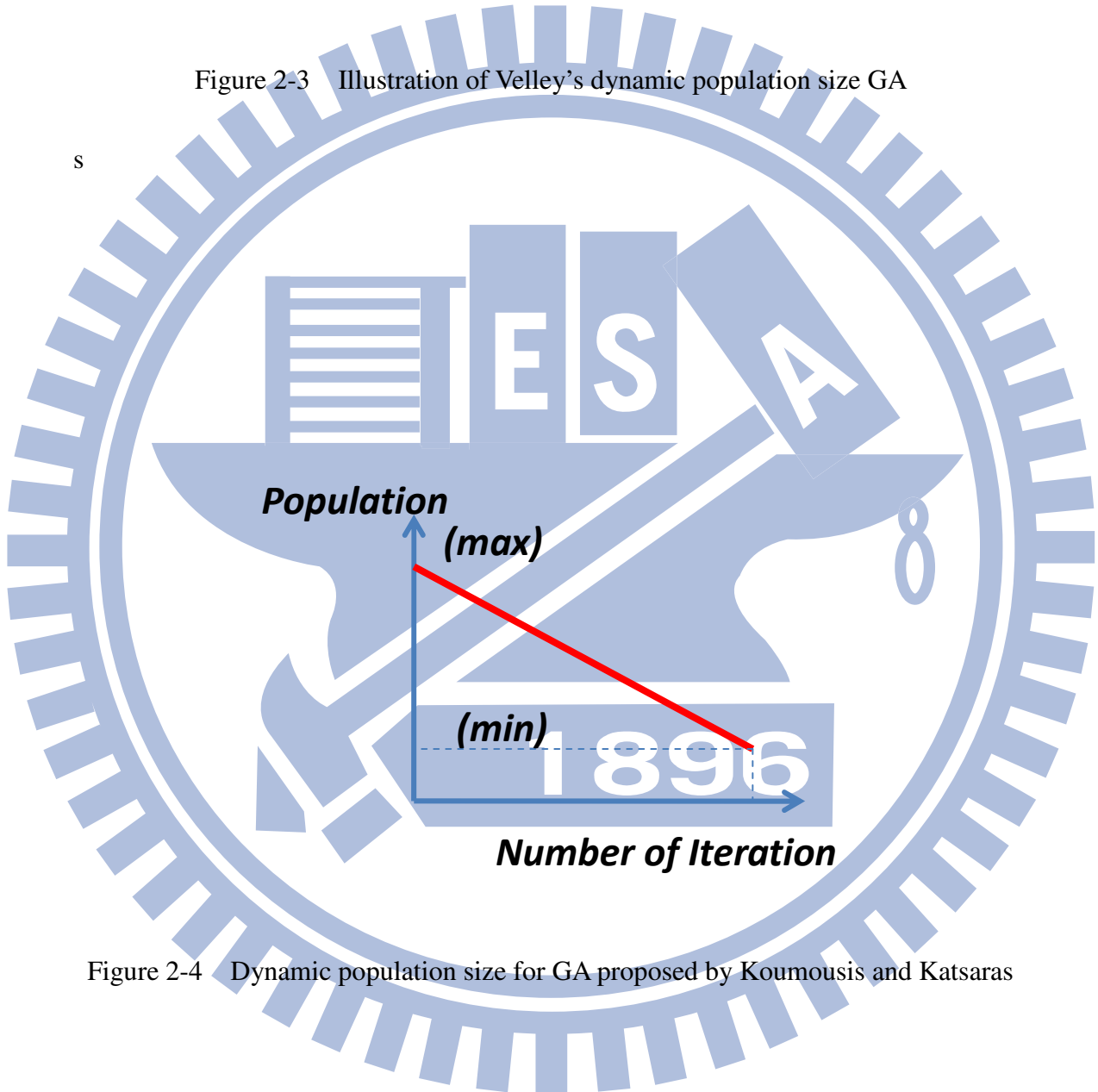


Figure 2-4 Dynamic population size for GA proposed by Koumouisis and Katsaras

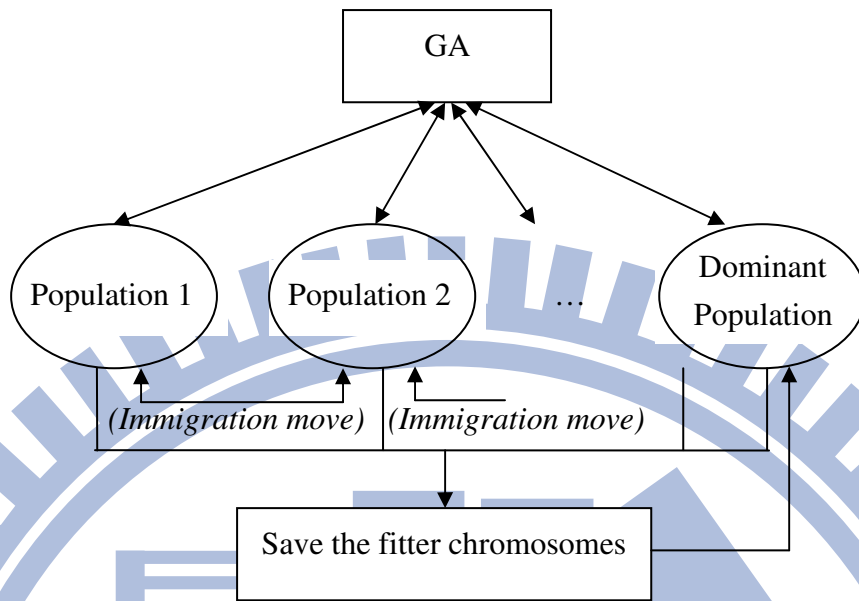


Figure 2-5 Illustration of the same size multi-population GA

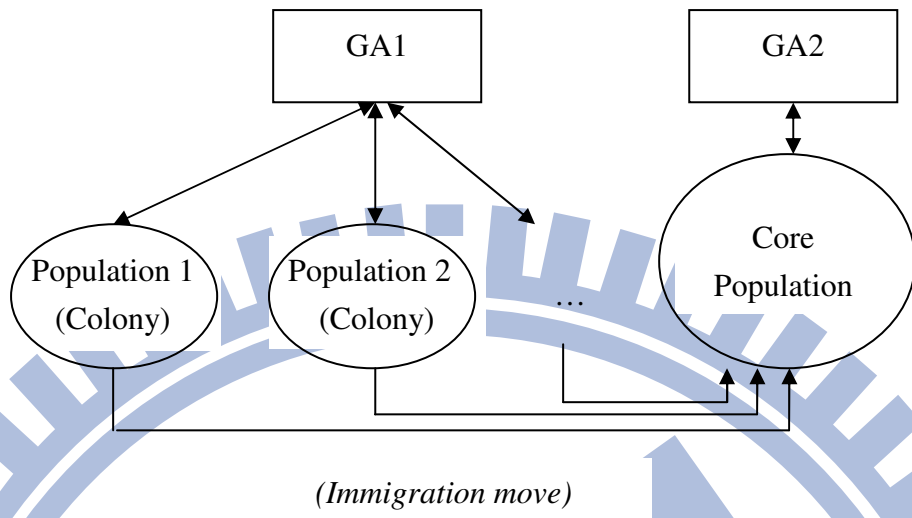


Figure 2-6 Illustration of the core/colony multi-population GA

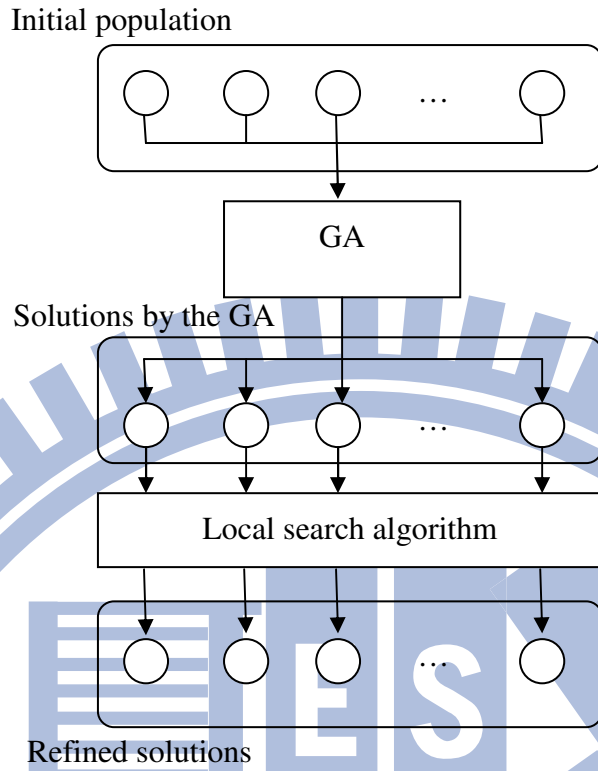


Figure 2-7a An example of the hybrid GA

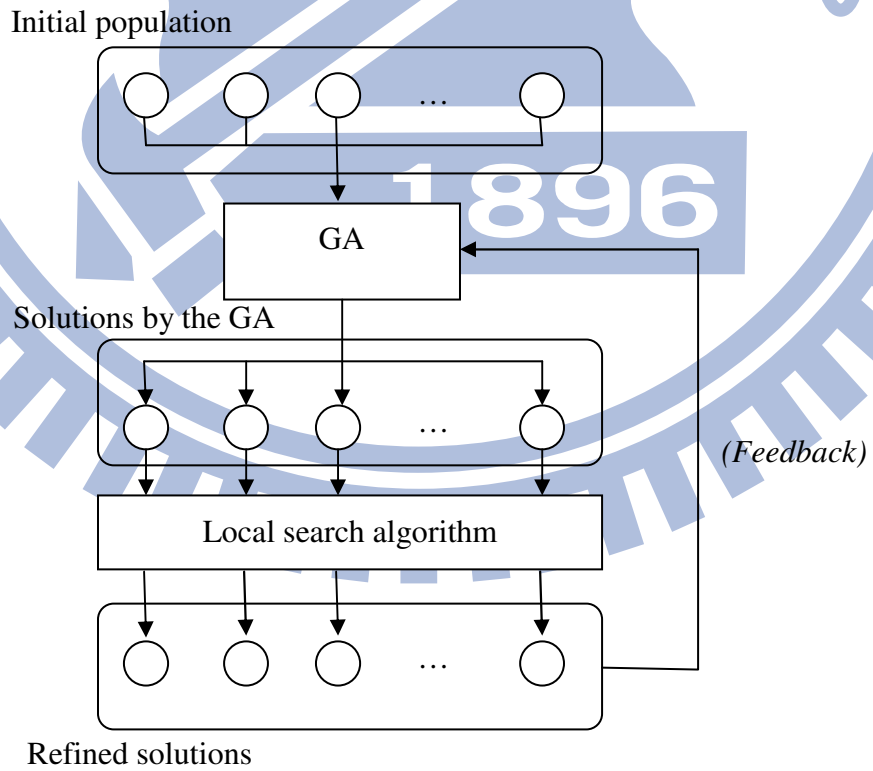


Figure 2-7b Another example of the hybrid GA

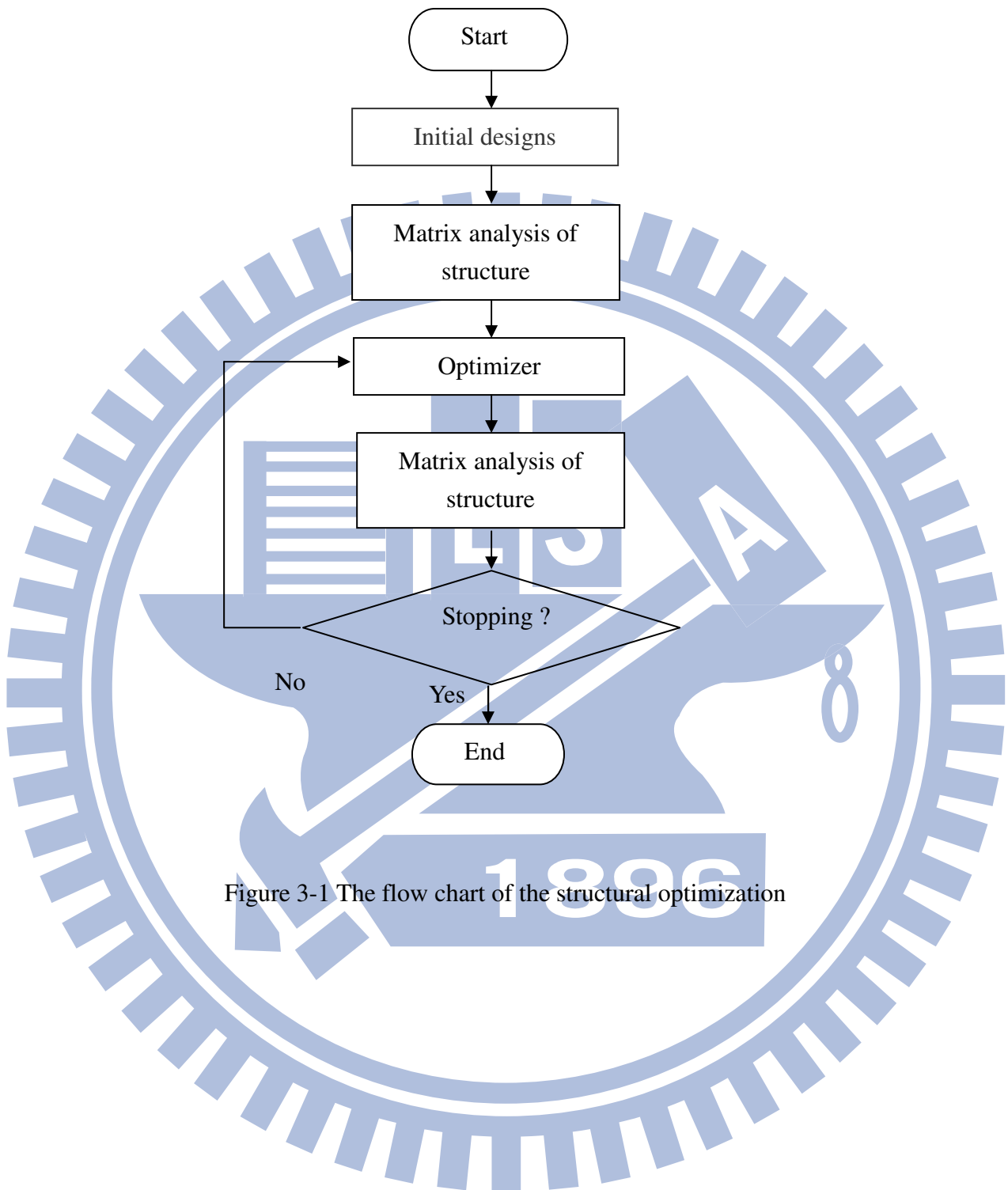
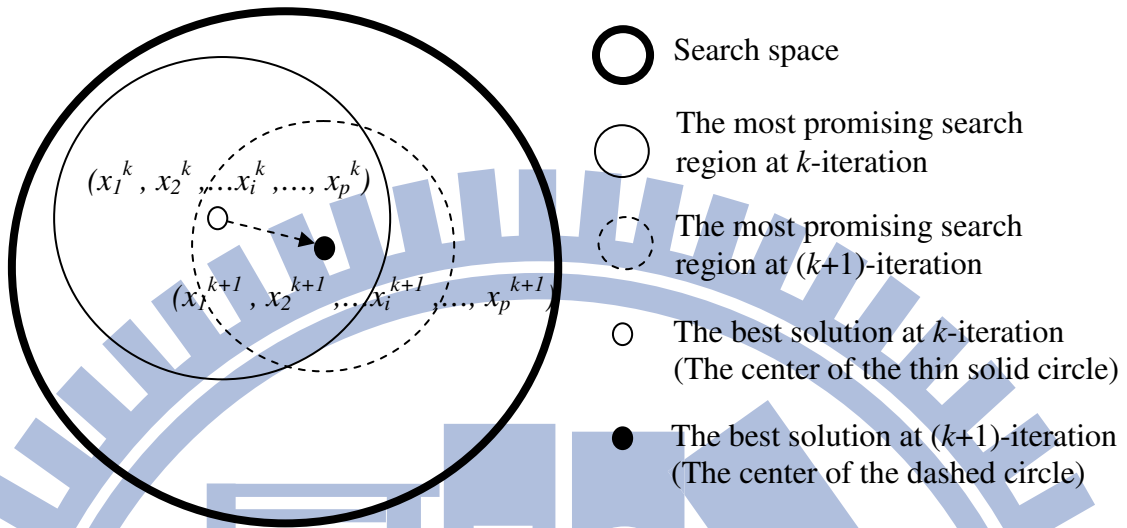


Figure 3-1 The flow chart of the structural optimization



		x_i^k		x_i^{k+1}							
Z_i	$z_{i(1)}$	$z_{i(2)}$	$z_{i(3)}$	$z_{i(4)}$	$z_{i(5)}$	$z_{i(6)}$	$z_{i(7)}$	$z_{i(8)}$	$z_{i(9)}$	$z_{i(10)}$	$z_{i(11)}$
W^k	0	1	1	1	1	1	1	0	0	0	0
W^{k+1}	0	0	0	0	1	1	1	1	1	0	0

Z_i A set of sequentially discrete values for x_i

W^k Window at k -iteration

W^{k+1} Window at $(k+1)$ -iteration

Figure 3-2 Illustration of the hill-climbing strategy

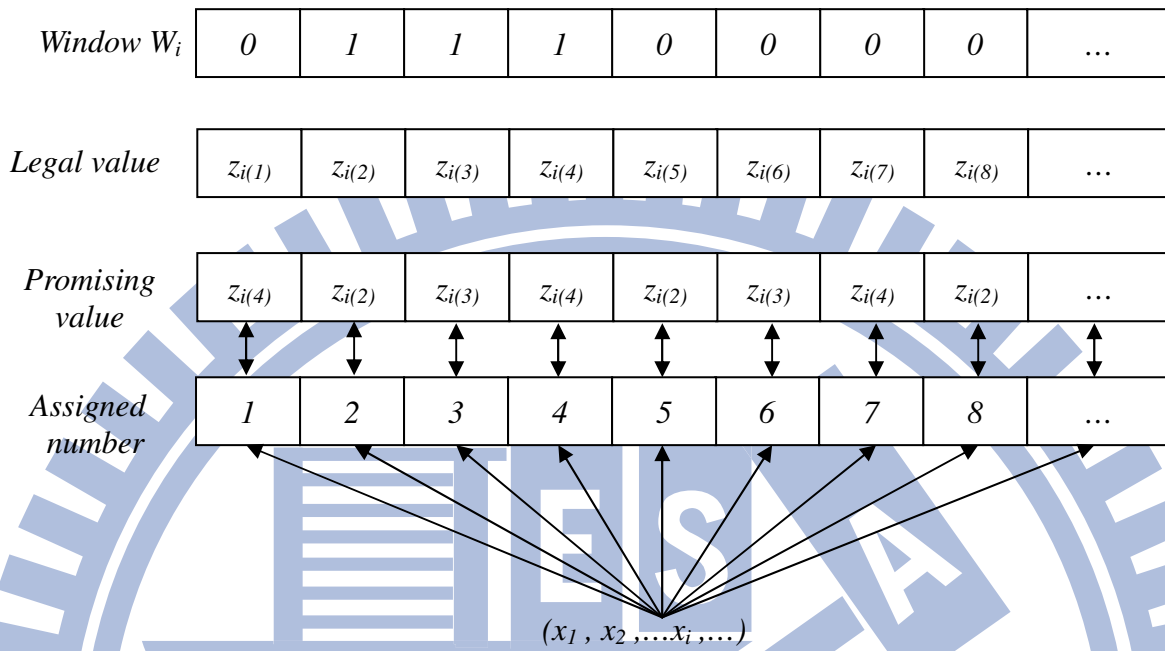


Figure 3-3 Illustration of modulus decoding

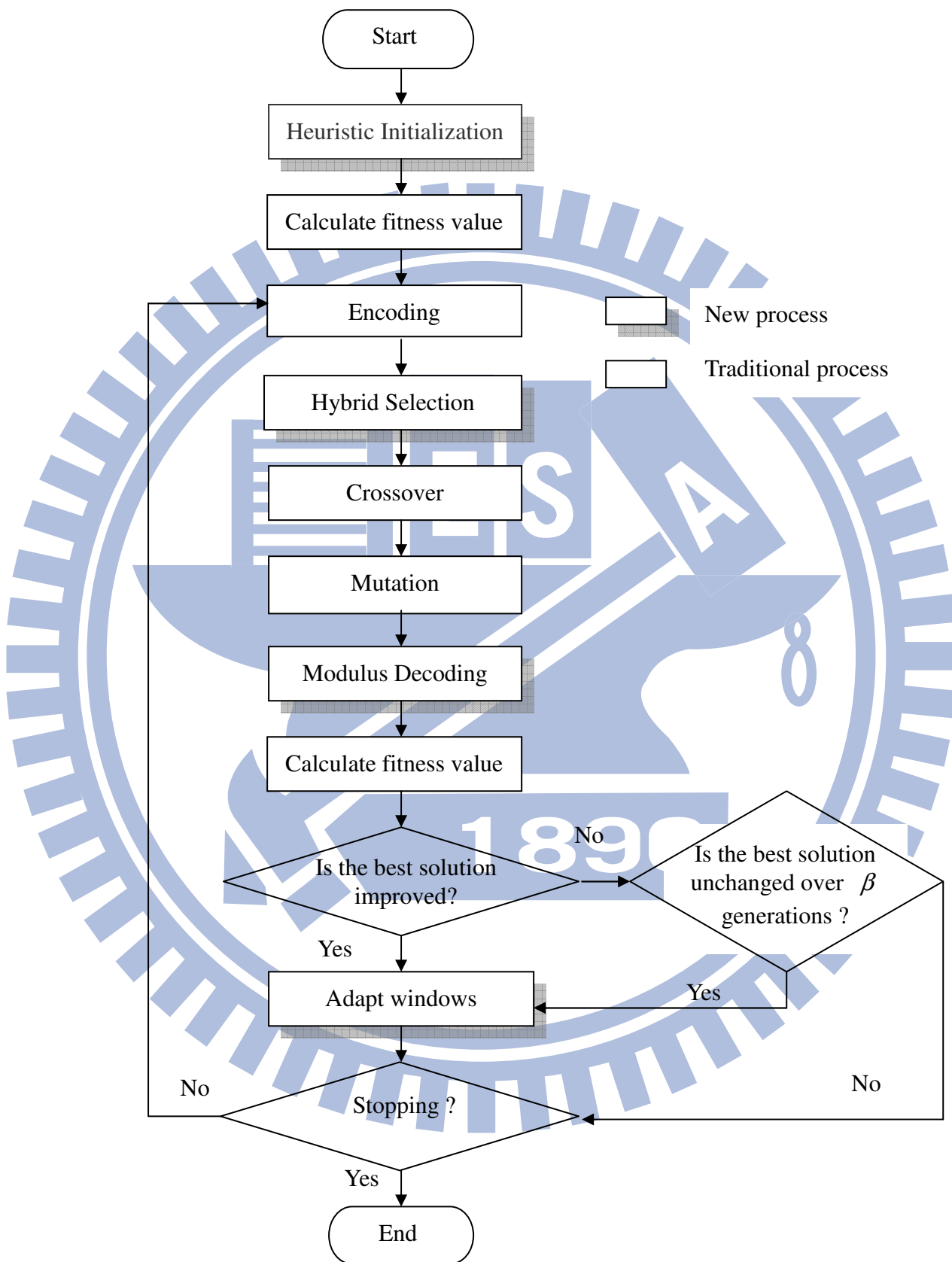


Figure 3-4 The flow chart of the hill-climbing and greedy genetic algorithm

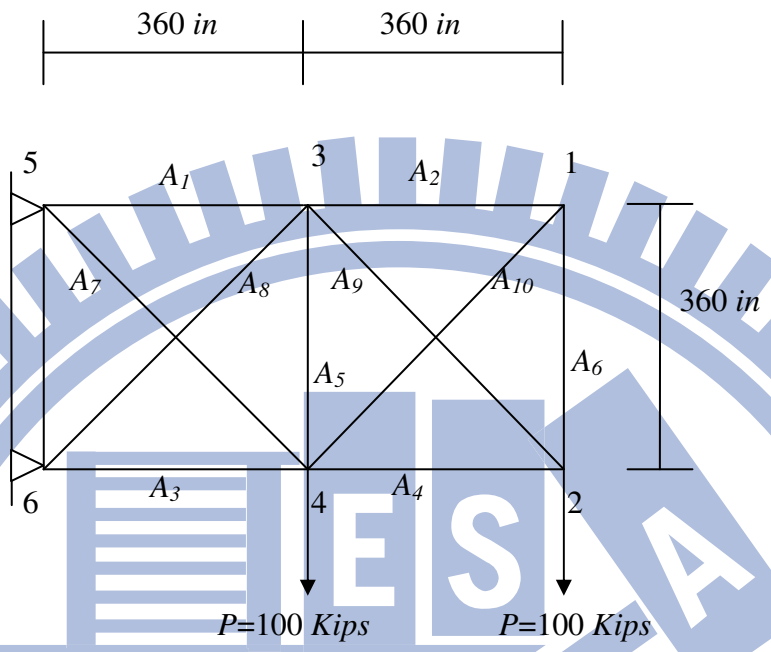


Figure 4-1 A 10-bar plane truss structure

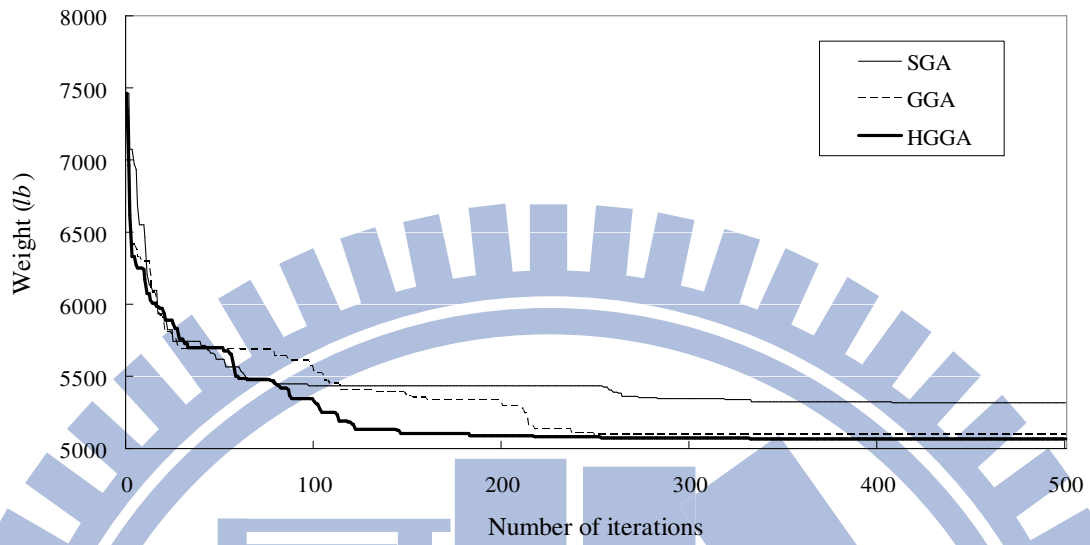


Figure 4-2 The comparison of convergence rates for the 10-bar plane truss structure (Case I)

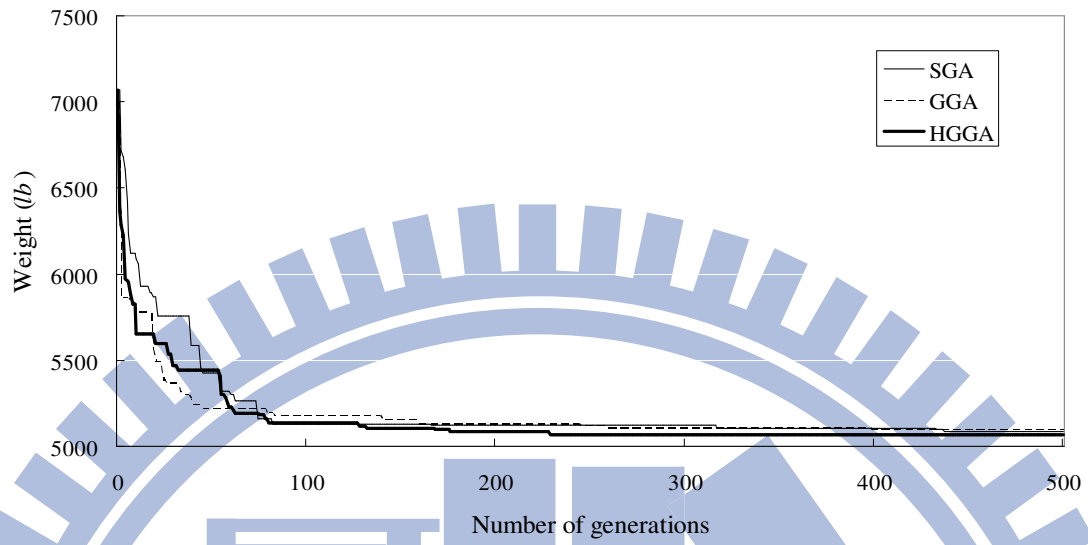


Figure 4-3 The comparison of convergence rates for the 10-bar plane truss structure (Case II)

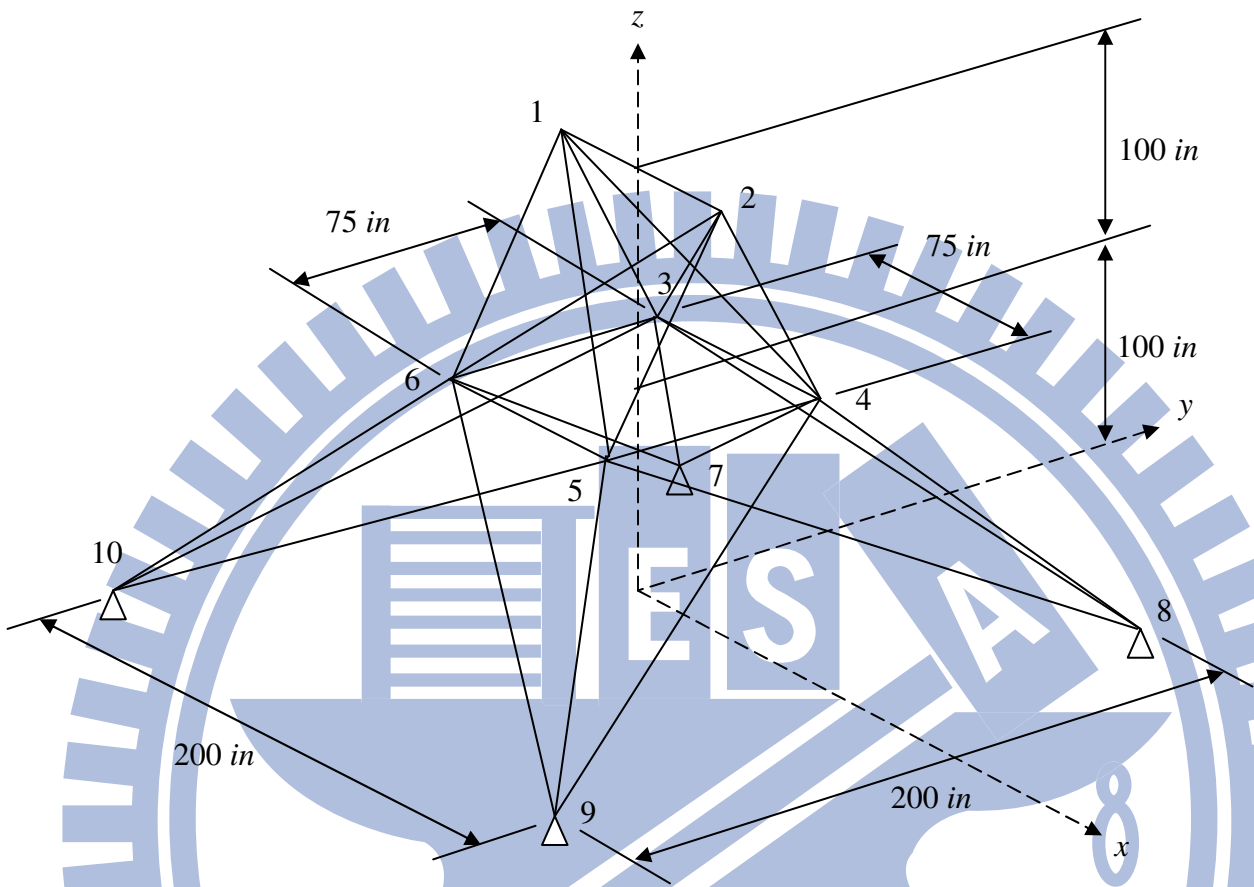


Figure 4-4 A 25-bar space truss structure

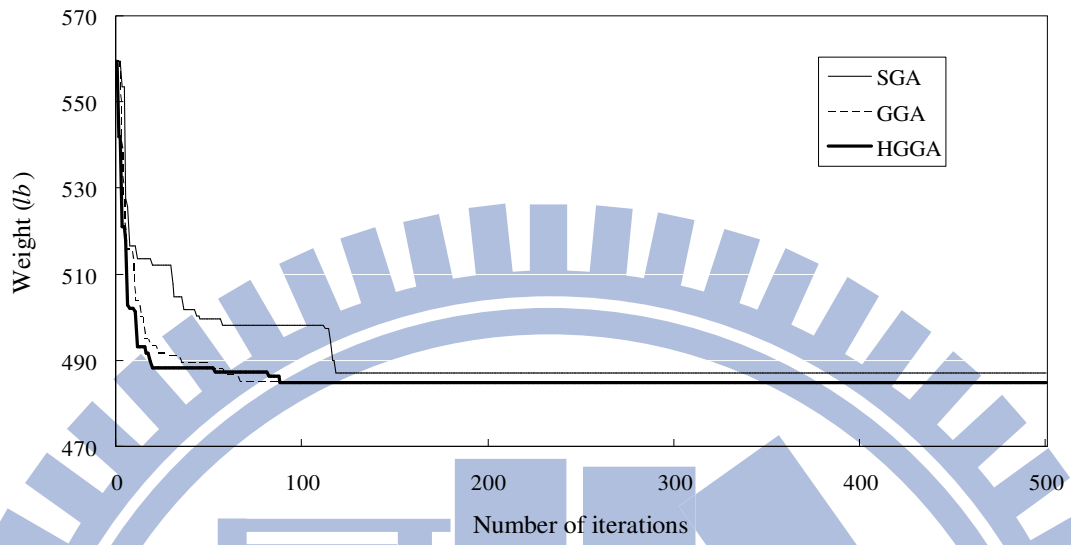


Figure 4-5 The comparison of convergence rates for the 25-bar space truss structure (case I)

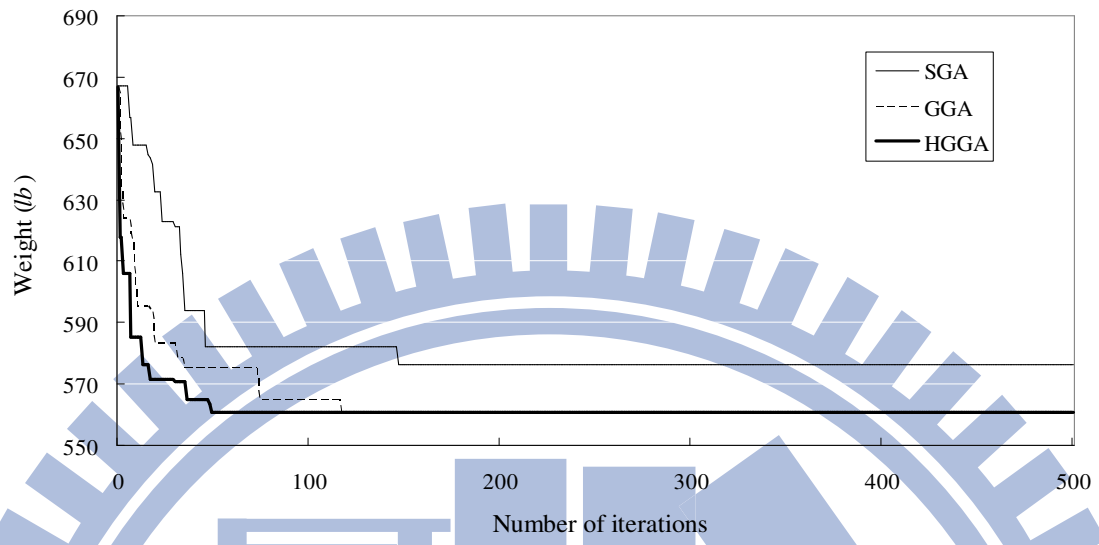


Figure 4-6 The comparison of convergence rates for the 25-bar space truss structure (case II)

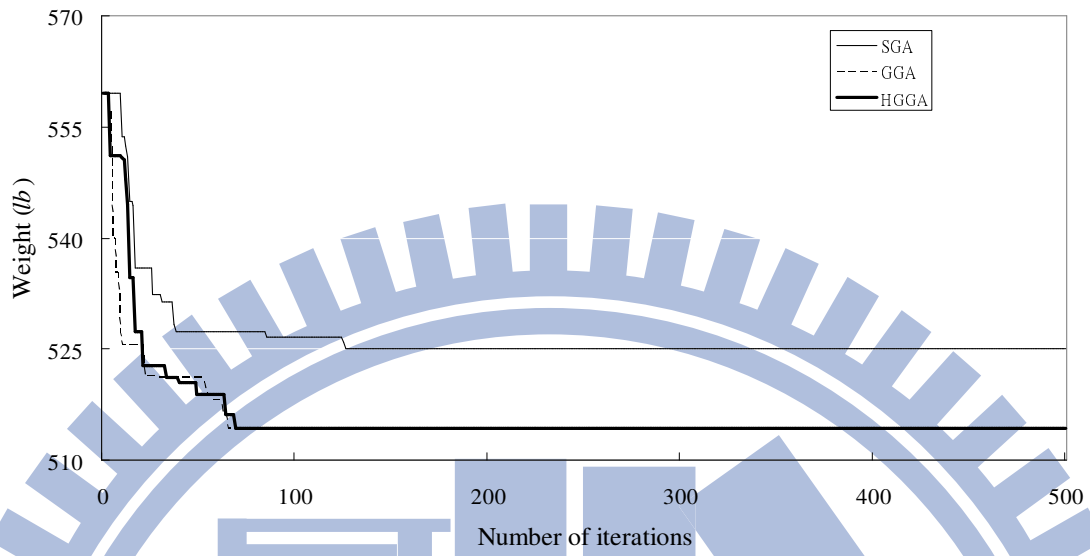


Figure 4-7 The comparison of convergence rates for the 25-bar space truss structure (case III)

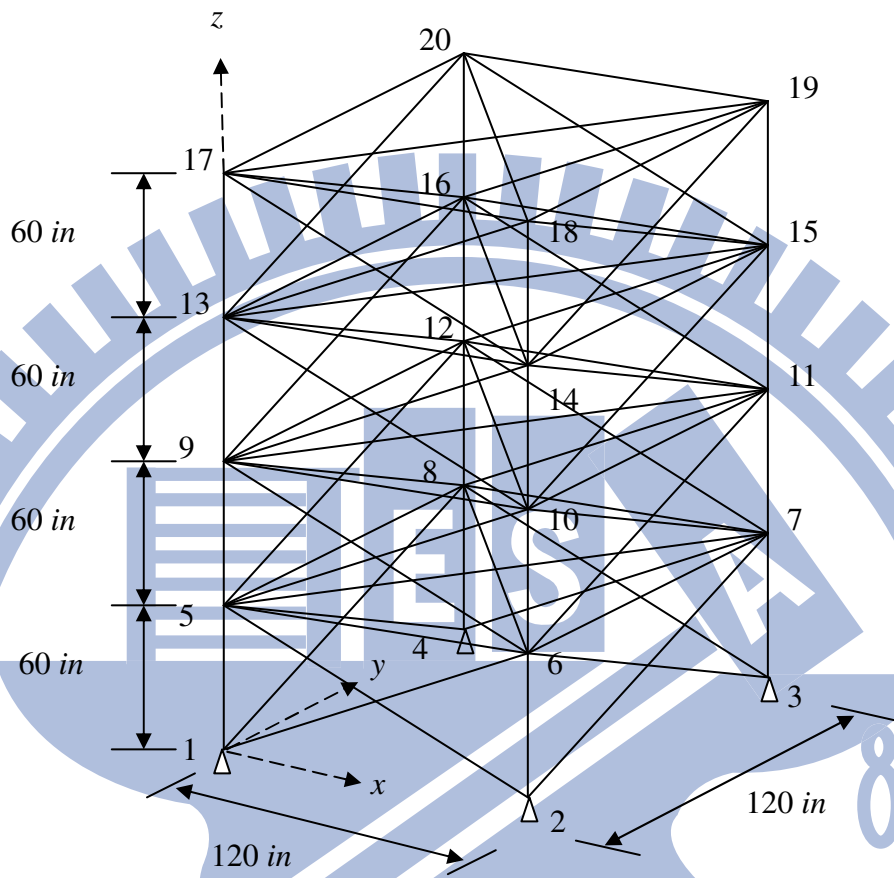


Figure 4-8 A 72-bar space truss structure

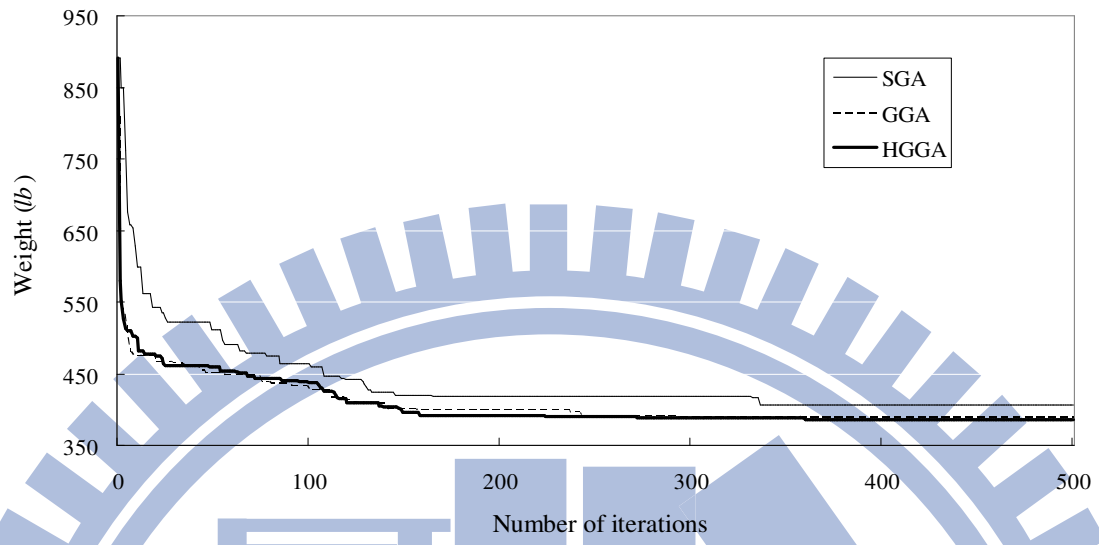


Figure 4-9 The comparison of convergence rates for the 72-bar space truss structure

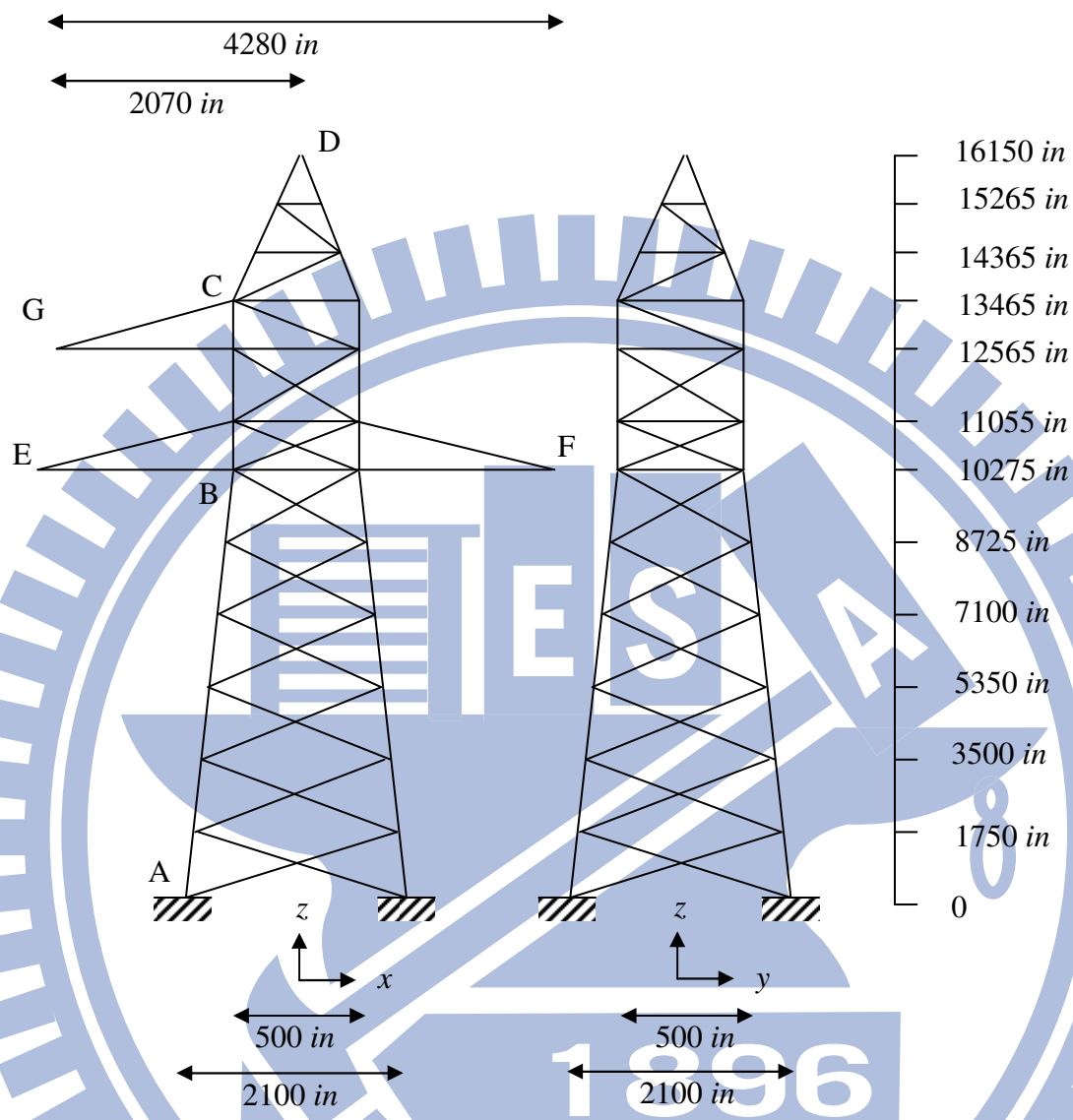


Figure 4-10 A 160-bar truss structure

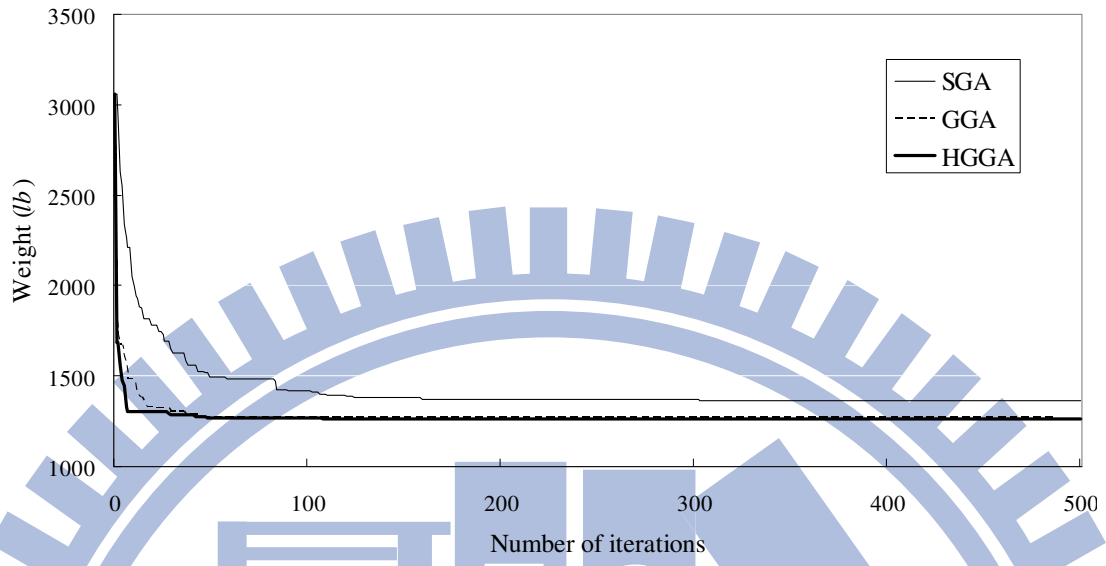


Figure 4-11 The comparison of convergence rates for the 160-bar space truss structure

