INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2011; 2:279-304

THE EFFECTS OF INITIAL SAMPLING AND PENALTY FUNCTIONS IN OPTIMAL DESIGN OF TRUSSES USING METAHEURISTIC ALGORITHMS

S. Shojaee[1,*,†] and S. Hasheminasab[2]
[1]*Department of Civil Engineering, Shahid Bahonar University of Kerman, Kerman, Iran*
[2]*Civil Engineering Department, Islamic Azad University, Kerman Branch, Kerman, Iran*

ABSTRACT

Although Genetic algorithm (GA), Ant colony (AC) and Particle swarm optimization algorithm (PSO) have already been extended to various types of engineering problems, the effects of initial sampling beside constraints in the efficiency of algorithms, is still an interesting field. In this paper we show that, initial sampling with a special series of constraints play an important role in the convergence and robustness of a metaheuristic algorithm. Random initial sampling, Latin Hypercube Design, Sobol sequence, Hammersley and Halton sequences are employed for approximating initial design. Comparative studies demonstrate that well distributed initial sampling speeds up the convergence to near optimal design and reduce the required computational cost of purely random sampling methodologies. In addition different penalty functions that define the Augmented Lagrangian methods considered in this paper to improve the algorithms. Some examples presented to show these applications.

Received: 5 March 2011; Accepted: 10 October 2011

KEY WORDS: Metaheuristic algorithms; optimal design; initial sampling; constraint; trusses

1. INTRODUCTION

Over the last three decades, a wide range of powerful mathematical programming methods have been developed for solving optimization Problems. Amongst all there are some that mimicking natural phenomena such as: Genetic algorithms, Ant colony and Particle swarm optimization algorithm.

[*]Corresponding author: S. Shojaee, Department of Civil Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
[†]E-mail address: saeed.shojaee@mail.uk.ac.ir

GAs are efficient and broadly applicable global search procedures based on stochastic approach which relies on "survival of the fittest" strategy [1]. In recent years, GAs have been used in structural optimization by many researchers (Goldberg and samtani [2]; Ahn and Ramakrishna [3]; Park et al. [4]; Kaveh and Khanlari [5]; Sahab et al. [6]; Castilho et al. [7]). All these studies have demonstrated that GAs can be powerful design tools for optimization problems. Another metaheuristic algorithm is Ant Colony Optimization [8, 9].

It has been inspired by the behavior of colonies of ants when they try to get food. In fact ants during the food searching, deposit on the ground a chemical substance called pheromone; subsequent ants can smell the deposited pheromone, and tend with a higher probability to follow the paths where the pheromone concentration (and consequently smell concentration) is stronger. Note that individual ants have no explicitly knowledge of the collective task; each ant works by itself and the pheromone deposit stimulates and guides probabilistically the others ants towards the better path, producing an apparently highly organized behavior. In other words the ants do not direct their work, but are guided by it. Such type of indirect communication is called stigmergy and was for the first time introduced in 1959 by the French zoologist Pierre-Paul Grasse'.

The natural behavior of ants can be simulated and translated in computer code, called ACO, to solve complex combinatorial optimization problems. Today several versions of ACO exist because a particular implementation is problem-dependent, so ACO really means a class of algorithms. These include: Ant System (AS) [10], Elitist AS (Dorigo et al. [11]), Ant-Q (Gambardella and Dorigo [12]), Ant Colony System (Dorigo and Gambardella [13, 14]), MAX–MIN AS (Stutzle and Hoos [15]), Rank-based AS (Bullnheimer et al. [16]), ANTS (Maniezzo [17]), Hyper-cube AS (Blum et al. [18]). In this paper the Ranked-Based Ant system have been used.

The last algorithm is the Particle Swarm Intelligence. The particle swarm optimization or PSO, which is based on the social behavior reflected in flock of birds, bees, and fish that adjust their physical movements to avoid predators, and to seek the best food sources (Eberhart and Kennedy [19]). The PSO algorithm was first proposed by Kennedy and Eberhart [20]. It is based on the premise that social sharing of information among members of a species offers an evolutionary advantage.

Recently, the PSO has been proved useful on diverse engineering design applications such as logic circuit design (Coello and Luna [21]), control design (Zheng et al. [22]), and power systems design (Abido [23]) among others. Applications in structures had been done in the area of structural shape optimization (Fourie and Groenwold [24]), and in topology optimization (Venter and Sobieszczanski [25]) with promising results in such structural design applications.

In all these algorithms the initial population in first is selected randomly; however we can use some initial sampling technique to improve the speed of their convergence. Initial sampling techniques like Latin Hypercube Design, Sobol sequence, Hammersley and Halton sequences have widely been used in Metamodelig techniques such as response surface methodology, artificial neural network, kriging, and radial basis function approximations. However they have been used scarcely with metaheuristic algorithms.

Latin Hypercube Sampling (LHS) first, was developed to generate a distribution of plausible collections of parameter values from a multidimensional distribution. The sampling

method is often applied in uncertainty analysis, and was first described by McKay et al. [26]. It was further elaborated by Imam et al. [27].

Sobol sequences are an example of quasi-random low-discrepancy sequences. (Discrepancy is a quantitative measure for the deviation of the sequence from the uniform distribution). They were first introduced by I.M.Sobol in [28]. After that it has been used and discussed by researchers; Bratley and Fox [29], Niederreiter [30], Antonov and Saleev [31], Jackel [32], Press et al. [33].

Hamersley and Halton are another two useful low discrepancy sequences. They have been used in numerical (Paskov and Traub [34]; Traub [35]; Case [36]) and graphic [37-40] applications, with a significant improvement in terms of error. They were developed by Diwekar and co-workers [41-43] and Halton [44] respectively.

So we are going to apply these techniques into our initial population and proving that a well-distributed population speeds up the convergence this result is particularly important whenever the optimization task involves time-consuming function.Besides these initial sampling methods, for applying constraint we have used some penalty functions that already have been used in Augmented Lagrangian algorithms by Birgin et al. [56].

2. PROPOSED OPTIMIZATION ALGORITHMS

In order to make the paper self-explanatory, the characteristics of GA, ACO and PSO are briefly explained in the following three sections:

2.1. Genetic algorithms

GAs are an optimization strategy in which points in the design space are analogous to organisms involved in a process of natural selection. The term ‘genetic’ is used because, along with the expected design representation, GAs employ a coded representation of design attributes that is analogous to a chromosome [1]. This code is commonly a character string, with each character position being analogous to a gene, and each character assigned to a position being analogous to an allele. Organisms are generated and tested in generations, with offspring designs arising from parent designs. The creation of new designs for a new generation occurs with a process that is analogous to biological reproduction. Genetic crossover allows offspring designs to retain traits from parent designs, and infrequent mutations possibly yield radically improved designs, but almost always yield unsuitable configurations. The testing of new designs is done with a merit function, usually tailored to take the coded representation as input. In a given generation, designs with a higher merit are given a higher probability of creating offspring, and perhaps surviving themselves into the next generation.

Optimization occurs, therefore, through a process of natural selection. Designs in a given generation group in pairs (i.e., mate), with the better designs having a higher probability of pairing. These `parent' designs produce offspring by genetic crossover. In `single point' crossover, a point along the coded representations (the chromosomes) is chosen at random, and the segments of the code after the point are swapped. Infrequent, random mutations are then performed on individual alleles within the chromosomes by changing the values. These

operations yield two new codes which represent two new designs that possess traits from both parents. In this way a new generation is created. The process then iterates. After many generations, both the best design and the average quality should increase, because the merit function is more likely to allow better designs to produce offspring.

2.1.1. GA parameters

Unless otherwise specified, the GA routines utilized random initial populations, binary-coded chromosomes, single-point crossover, mutation, and an elitist stochastic universal sampling selection strategy (Baker, [61]). The probabilities of crossover (0.95) and mutation (0.01), and the population size (60) in each example were chosen according to values suggested by Grefenstette [62] and Schaffer et al. [63]. Figure 1 shows the flow chart for the GA algorithm.

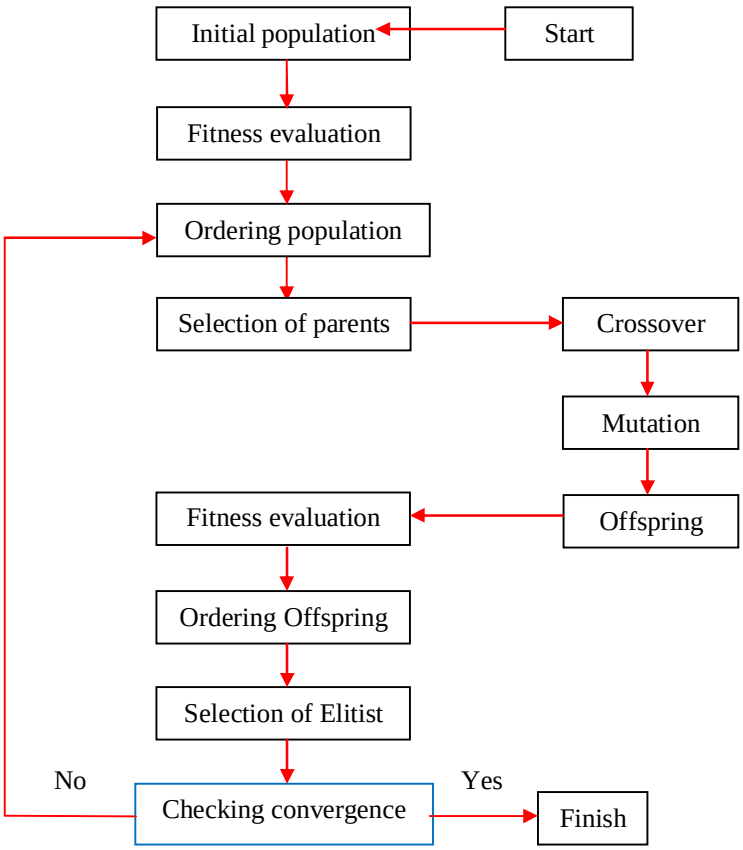

Figure 1. The flow chart for GA

2.2. Ant Colony Optimization algorithms

2.2.1. Ranked-Based Ant system

In AS_{rank} each ant deposits an amount of pheromone that decreases with its rank. Additionally, the best-so-far ant always deposits the largest amount of pheromone in each iteration.

2.2.1.1. Update of Pheromone Trails

Before updating the pheromone trails, the ants are sorted by increasing tour length and the quantity of pheromone that, an ant deposits is weighted according to the rank r of the ant. Ties can be solved randomly (in our implementation they are solved by lexicographic ordering on the ant name k). In each iteration only the (w-1) best-ranked ants and the ant (here w was taken 15% ants) that produced the best-so-far tour (this ant does not necessarily belong to the set of ants of the current algorithm iteration) are allowed to deposit pheromone. The best-so-far tour gives the strongest feedback, with weight w (i.e., its contribution $1/C^{bs}$ is multiplied by w); the r-th best ant of the current iteration contributes to pheromone updating with the value $1/C^{r}$ multiplied by a weight given by max {0, w _ r}. Thus, the AS(rank) pheromone update rule is :

$$\tau_{ij} \rightarrow \tau_{ij} + \sum_{r=1}^{w-1} (w - r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs} \quad (1)$$

where $\Delta\tau_{ij}^{r} = 1/C^{r}$ or $\Delta\tau_{ij}^{r} = 1/C^{bs}$, C^{r} is the length of tour r and C^{bs} is the best-so-far tour length. The ACO procedure is illustrated in Figure 2.

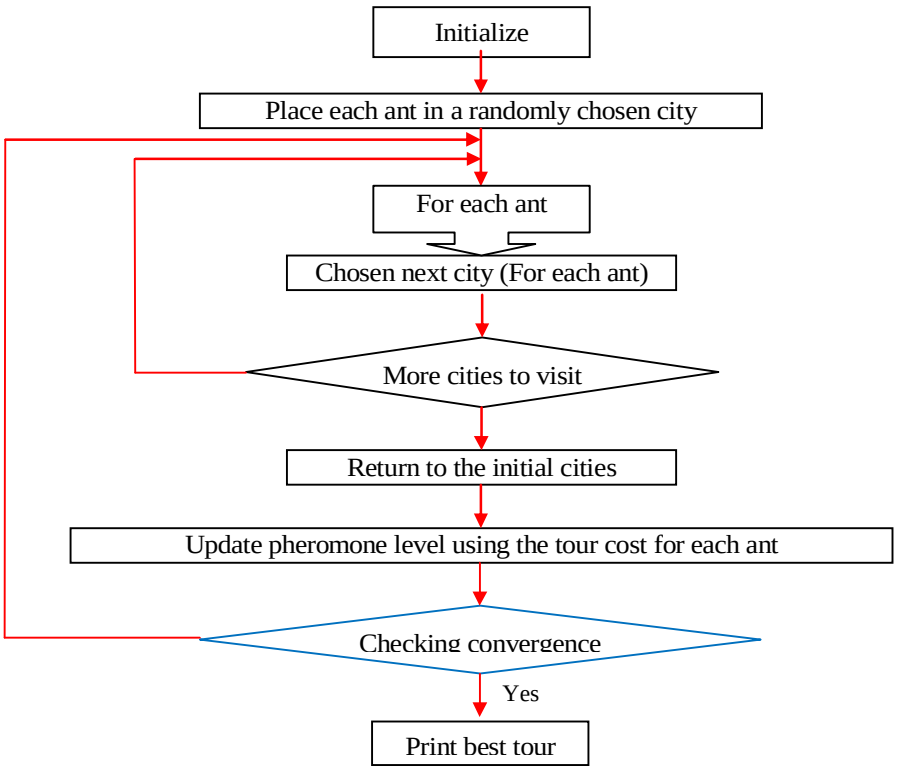

Figure 2. The flow chart for ACO

2.3. Particle swarm algorithm

In this section, we present our implementation of the PSO, which has some similarities to the implementation presented by Fourie and Groenwold [45, 46]. We mimic the social behavior of

birds. Individual birds exchange information about their position, velocity and fitness, and the behavior of the flock is then influenced to increase the probability of migration to regions of high fitness.

In flight, each bird in the flock continuously processes information about its current **position**, **velocity** and **fitness**. In addition, information regarding its position, velocity and fitness with respect to the complete flock is processed. In our optimization problem, the position of each bird is represented by the design variables **X**, while the velocity of each bird **V**, influences the incremental change in the position of each bird, and hence the design variables.

Let us consider a flock of p particles or birds. For particle **i**, Kennedy and Eberhart [20] originally proposed that the position x^i be updated as:

$$x^i_{k+1}=x^i_k+v^i_{k+1} \tag{2}$$

While the velocity v^i is updated as:

$$v^i_{k+1}=v^i_k+c_1r_1(p^i_k-x^i_k)+c_2r_2(p^g_k-x^i_k) \tag{3}$$

Here, the subscript *k* shows a (unit) pseudo-time increment. p^i_k represents the best ever position of particle **i** at time k, while p^g_k represents the global best position in the swarm at time k. r_1 and r_2 represent uniform random numbers between 0 and 1.

Kennedy and Eberhart initially proposed that the cognitive and social scaling factors c_1 and c_2 be selected such that $c_1= c_2= 2$, in order to allow a mean of 1 (when multiplied by the random numbers r_1 and r_2). The result of using these proposed values is that birds ***overfly*** the target half the time.Shi and Eberhart [49] later introduced an inertia term w, modifying the velocity equation to become:

$$v^i_{k+1}=wv^i_k+c_1r_1(p^i_k-x^i_k)+c_2r_2(p^g_k-x^i_k) \tag{4}$$

They proposed that w be selected such that $0.8<w<1.4$. In addition, they report improved convergence rates when w is decreased linearly during the optimization. Figure 3 shows the optimization procedure of the PSO algorithm.

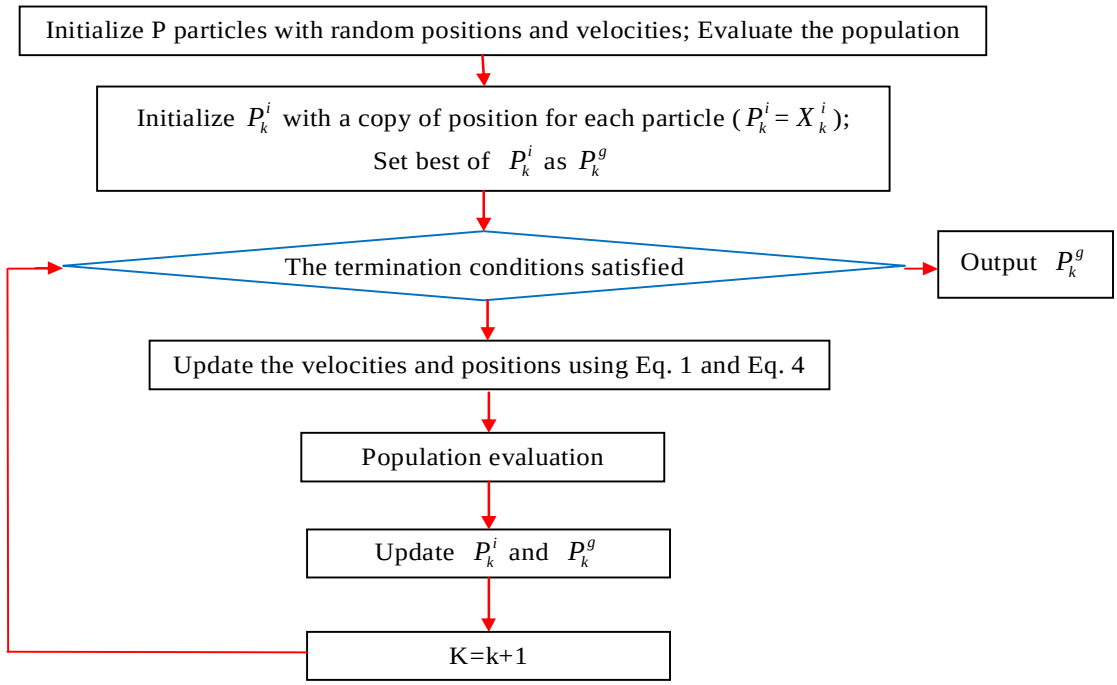

Figure 3. The flow chart for PSO

3. SAMPLING TECHNIQUES

3.1. Latin Hypercube sampling

Latin hypercube design (McKay et al. [26]) can be viewed as an N-dimensional extension of traditional Latin square design (Montgomery, [60]). On each level of every design variable only one point is placed. There are the same number of levels as runs and the levels are assigned randomly to runs. This method ensures that every variable is presented, no matter if the response is dominated by only a few ones. Another advantage is that the number of points to be analyzed can be directly defined. Detailed computer codes and manuals were published by Imam ant et al. [27].

3.2. Sobol Sequence Sampling

The algorithm for generating Sobol sequences is clearly explained in Bratley and Fox [29], Algorithm 659. To generate the *j*-th component of the points in a Sobol' sequence, we need to choose a primitive polynomial of some degree s_j over the field GF(2) (GF(2) simply means that the coefficients a in the polynomial below can either be 0 or 1).

$$P_j = x^{s_j} + a_{1,j}x^{s_j-1} + a_{2,j}x^{s_j-2} + \ldots + a_{s_j-1,j}x + 1 \quad (5)$$

Where the coefficients $a_{1,j}$, $a_{2,j}$, ..., $a_{sj-1,j}$ are either 0 or 1. A sequence of positive integers

$\{m_{1,j}, m_{2,j}, ...\}$ are defined by the recurrence relation:

$$m_{k,j} = 2a_{1,j}m_{k-1,j} \oplus 2^2 a_{2,j}m_{k-2,j} \oplus 2^{s_j-1} a_{s_j-1,j}m_{k-s_j+1,j} \oplus 2^{s_j} m_{k-s_j,j} \oplus m_{k-s_j,j} \quad (6)$$

Where $\oplus$ is the bit-by-bit exclusive-or operator. The initial values $m_{1,j}, m_{2,j}, ..., m_{sj,j}$ can be chosen freely provided that each $m_{k,j}$, $1 \le k \le s_j$, is odd and less than 2^k. The so-called **direction numbers** $\{v_{1,j}, v_{2,j}, \ldots\}$ (In some papers numbers $m_{k,j}$ also can be referred as **direction numbers**) are defined by :

$$v_{k,j} = \frac{m_{k,j}}{2^k}. \quad (7)$$

Then $x_{i,j}$, the j-th component of the i-th point in a Sobol' sequence, is given by :

$$x_{i,j} = i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus ..., \quad (8)$$

Where i_k is the *k*-th binary digit of $i = (\ldots i_3 i_2 i_1)_2$. Here the notation $(\cdot)_2$ denotes the binary representation of numbers.

3.3. Hammersley and Halton Sampling Algorithm

The algorithm that generates a set of N Hammersley points makes use of the radix-R notation of an integer. That is, a specific integer, p, in radix-R notation can be represented as:

$$p = p_m p_{m-1} \ldots p_2 p_1 p_0 \quad (9)$$

$$p = p_0 + p_1 R + p_2 R^2 + ... + p_m R^m \quad (10)$$

Where $m = [\log_R p] = [(\ln p) / (\ln R)]$**,** and the square brackets, [ ], denote the integer portion of the number inside the brackets. For example, in the familiar base-10 (i.e., radix-10) number system, the integer 756 has $p_0 = 6$, $p_1 = 5$, and $p_2 = 7$, with R=10 and m=2**.** The inverse radix number function constructs a unique number on the interval [0, 1] by reversing the order of the digits of p around the decimal point. The inverse radix number function is:

$$\Phi_R(p) = 0.p_0 p_1 p_2 \ldots p_m \quad (11)$$

$$\Phi_R(p) = p_0 R^{-1} + p_1 R^{-2} + ... + p_m R^{-m-1} \quad (12)$$

Finally, the Hammersley sequence of n-dimensional points is generated as:

$$x_n(p) = \left(\frac{p}{N}, \Phi_{R_1}(p) + \Phi_{R_2}(p) + ... + \Phi_{R_{n-1}}(p) \right) \quad (13)$$

where p = 0,1,2,…,N-1; and the values for R_1, R_2,…, R_{n-1} are the first n-1 prime numbers (2,3,5,7,11,13,17,…). This approach generates a set of N points in the n-dimensional design space.The Halton sequence is exactly the same as Hammersley except that instead of equation:

$$x_n(p) = \left(\frac{p}{N}, \Phi_{R_1}(p) + \Phi_{R_2}(p) + ... + \Phi_{R_{n-1}}(p)\right) \tag{14}$$

We have,

$$x_n(p) = \left(\Phi_{R_1}(p) + \Phi_{R_2}(p) + ... + \Phi_{R_{n-1}}(p)\right) \tag{15}$$

4. AUGMENTED LAGRANGIAN ALGORITHMS

4.1. Main algorithm

Augmented Lagrangian algorithms are very popular tools for solving nonlinear programming problems, i.e. Minimize f(x) subject to $g(x) \leq 0$, $x \in \Omega$ The set Ω is compact and convex whereas f: $R^n \rightarrow R$ and g: $R^n \rightarrow R^m$ are continuously differentiable on an open set that contain Ω. In general, $\Omega = \{x \in R^n \mid l \leq x \leq u\}$. The function f and g are, in general, noncovex. Let define $R_{++} = \{t \in R \mid t > 0\}$, N = {0, 1, 2 …} and the Augmented Lagrangian L by,

$$L(x, \rho, \mu) = f(x) + \sum_{i=1}^{m} P(g_i(x), [\rho]_i, [\mu]_i) \tag{16}$$

For all $x \in \Omega$, $\rho \in R_{++}^m$, and $\mu \in R_{++}^m$. Denote Þ, the Euclidian projection operator onto Ω; $\nabla L(x, \rho, \mu)$, the gradient vector of L with respect to x. P is the Penalty-Lagrangian function. Let $P: R \times R_{++} \times R_{++} \rightarrow R$ be such that, $P'(x, \rho, \bar{\mu}) \equiv \frac{\partial}{\partial y}(y, \rho, \bar{\mu})$ exists and is continuous for all $y \in R$, $\rho \in R_{++}$, and $\bar{\mu} \in R_{++}$. The main model algorithm is the following,

Assume that $x_0 \in \Omega$, $\tau \in (0,1)$, $m_{\max} > m_{\min} > 0$, $\gamma > 1$, $\rho_1 \in R_{++}^m$, $m_0 \in R_{++}^m$. Let $\{\varepsilon_k\}_{k \in N}$ be a sequence of positive numbers that converges to zero.

Step 1. Initialization
Set $k \leftarrow 1$.
Step 2. Solving the sub-problem
Compute $[\bar{m}_k]_i \in [m_{\min}, m_{\max}]$, i=1,…, m. Using x_{k-1} as initial approximation,

$$\text{Minimize (approximately)} \quad L(x, \rho_k, \bar{\mu}_k) \text{ subject to } x \in \Omega. \tag{17}$$

The approximate minimizer must be such that:

$$\| Þ [x_k - \nabla L(x, \rho_k , \bar{\mu}_k)] - x_k \| \leq \varepsilon_k \tag{18}$$

Step 3. Estimate multipliers, Compute:

$$[\mu_k]_i = P'(g_i(x_k) , [\rho_k]_i , [\bar{\mu}_k]_i), \ i=1,\ldots, m. \tag{19}$$

Step 4. Update penalty parameters

For all i= 1,…, m, if

$$\max \{0, g_i(x_k)\} \leq t * \max\{0, g_i(x_{k-1})\} \tag{20}$$

and

$$\left| [\mu_k]_i g_i(x_k) \right| \leq t \left| g_i(x_{k-1})[\mu_{k-1}]_i \right| \tag{21}$$

Set

$$[\rho_{k+1}]_i = [\rho_k]_i$$

Else, set

$$[\rho_{k+1}]_i = g[\rho_k]_i$$

Step 5. Begin new iteration

Set k ← k+1 and go to step 2

Remark. In practice, the parameter $[\bar{\mu}_k]_i$ will be chosen as the projection of multiplier estimate $[\mu k\text{-}1]_i$ onto the safeguarding interval $[\mu_{min}, \mu_{max}] \subset R_{++}$.

The most famous Augmented Lagrangian algorithm for minimization with inequality constraints is known as Powell-Hestenes-Rockefellar (PHR) method (Hestenes [57]; Powell [58]; Rockefellar [59]) and is given by (16) associated with the penalty function:

$$P(y, \rho, \mu) = \frac{1}{2\rho} (\max \{0, \mu+\rho y\}^2 - \mu^2). \tag{22}$$

The main drawback of PHR is that the objective function of the sub-problems is not twice continuously differentiable. This is the main motivation for the introduction of many alternative Augmented Lagrangian methods. The exponential-multiplier form of the Augmented Lagrangian mentioned above have been considered until now, perhaps, the best known alternative for overcoming this deficiency.

The different penalty functions P that define the Augmented Lagrangian methods considered in this paper depend on two functions p_i and θ_j. We list those function below, specifying at the same time the way in which they are combined.

Penalty functions:

$$P_1(y,\rho,\mu)=\begin{cases}\mu y+\frac{1}{2}\rho y^2+\rho^2 y^3 & if\ \ y\geq 0\\ \mu y+\frac{1}{2}\rho y^2 & if\ \ -\frac{\mu}{\rho}\leq y\leq 0\\ -\frac{1}{2\rho}\mu^2 & if\ \ y\leq -\frac{\mu}{\rho}\end{cases} \tag{23}$$

$$P_2(y,\rho,\mu)=\begin{cases}\mu y+\mu\rho y^2+\frac{1}{6}\rho^2 y^3 & if\ \ y\geq 0\\ -\frac{\mu y}{1-y\rho} & if\ \ y\leq 0\end{cases} \tag{24}$$

$$P_3(y,\rho,\mu)=\begin{cases}\mu y+\mu\rho y^2 & if\ \ y\geq 0\\ \frac{\mu y}{1-\rho y} & if\ \ y\leq 0\end{cases} \tag{25}$$

$$P_4(y,\rho,\mu)=\begin{cases}\mu y+\frac{1}{\rho}q(\rho y) & if\ \ \mu+\frac{1}{\rho}q'(\rho y)\geq 0\\ \min_{\tau\in R}\{\mu\tau+\frac{1}{\rho}q(\rho\tau) & otherwise\end{cases} \tag{26}$$

$$P_5(y,\rho,\mu)=\frac{\mu}{\rho}q(\rho y) \tag{27}$$

$$P_6(y,\rho,\mu)=\frac{1}{\rho}q(\rho\mu y) \tag{28}$$

$$P_7(y,\rho,\mu)=\frac{\mu^2}{\rho}q\left(\frac{\rho y}{\mu}\right) \tag{29}$$

$$P_8(y,\rho,\mu)=\frac{1}{\rho}(q(\rho y+\psi)-q(\psi))\ (with\ \psi\ such\ that\ q'(\psi)=\mu) \tag{30}$$

$$P_9(y,\rho,\mu)=q(\rho y+\psi)-q(\psi)\ (with\ \psi\ such\ that\ q'(\psi)=\mu/\rho) \tag{31}$$

q **Functions:**

$$q_1(t)=\frac{1}{2}t^2 \tag{32}$$

$$q_2(t)=\frac{3}{4}t^{4/3} \tag{33}$$

$$q_3(t)=\cosh(t)-1 \tag{34}$$

$$q_4(t)=e^t-1 \tag{35}$$

$$q_5(t)=\begin{cases}-\log(1-t) & if\ \ t\leq\frac{1}{2}\\ e^{2t-1}+\log(2)-1 & if\ \ t\geq\frac{1}{2}\end{cases} \tag{36}$$

$$q_6(t)=\begin{cases}-\log(1-t) & if\ \ t\leq\frac{1}{2}\\ 2t^2+\log(2)-\frac{1}{2} & if\ \ t\geq\frac{1}{2}\end{cases} \tag{37}$$

$$q_7(t)=\begin{cases}\frac{t}{1-t} & if\ \ t\leq\frac{1}{2}\\ e^{4t-2} & if\ \ t\geq\frac{1}{2}\end{cases} \tag{38}$$

$$q_8(t)=\begin{cases}\frac{t}{1-t} & if\ t\le\frac{1}{2}\\ 8t^2-4t+1 & if\ t\ge\frac{1}{2}\end{cases} \tag{39}$$

$$q_9(t)=\begin{cases}-\frac{1}{4}\log(-2t)-\frac{3}{8} & if\ t\le-\frac{1}{2}\\ t+\frac{1}{2}t^2 & if\ t\ge-\frac{1}{2}\end{cases} \tag{40}$$

$$q_{10}(t)=\frac{1}{16}\left(1+t+\sqrt{(1+t)^2+8}\right)^2+\log\left(\frac{1}{4}\left(1+t+\sqrt{(1+t)^2+8}\right)\right)-1 \tag{41}$$

$$q_{11}(t)=\begin{cases}\frac{1}{6}\max\{0,\ t+\frac{1}{2}\}^3-\frac{1}{24} & if\ t\le\frac{1}{2}\\ \frac{1}{2}t^2 & if\ t\ge\frac{1}{2}\end{cases} \tag{42}$$

$$q_{12}(t)=\begin{cases}e^t & if\ t\le\frac{1}{2}\\ e^{1/2}\left(\frac{1}{2}t^2+\frac{1}{2}t+\frac{5}{8}\right) & if\ t\ge\frac{1}{2}\end{cases} \tag{43}$$

$$q_{13}(t)=\begin{cases}-\log(-t)-1 & if\ t\le-\frac{1}{2}\\ 2t^2+4t+\frac{1}{2}+\log(2) & if\ t\ge-\frac{1}{2}\end{cases} \tag{44}$$

$$q_{14}(t)=\begin{cases}-\frac{1}{t} & if\ t\le-\frac{1}{2}\\ 8t^2+12t+6 & if\ t\ge-\frac{1}{2}\end{cases} \tag{45}$$

$$q_{15}(t)=\begin{cases}\frac{4}{1-t}-2 & if\ t\le-1\\ -\log(-t) & if\ -1\le t\le-\frac{1}{4}\\ 8t^2+8t+\frac{3}{2}+2\log(2) & if\ t\ge-\frac{1}{4}\end{cases} \tag{46}$$

$$q_{16}(t)=\frac{1}{2}\left(t+\sqrt{t^2+4}\right) \tag{47}$$

$$q_{17}(t)=\log(1+e^t) \tag{48}$$

$$q_{18}(t)=\begin{cases}\frac{1}{2}e^t & if\ t\le0\\ t+\frac{1}{2}e^{-t} & if\ t\ge0\end{cases} \tag{49}$$

for example :

$$P_7\ and\ \theta_9:\frac{\mu^2}{\rho}\begin{cases}-\frac{1}{4}\log\left(-2\frac{\rho y}{\mu}\right)-\frac{3}{8} & if\ \frac{\rho y}{\mu}\le-\frac{1}{2}\\ t+\frac{1}{2}\left(\frac{\rho y}{\mu}\right)^2 & if\ \frac{\rho y}{\mu}\ge-\frac{1}{2}\end{cases}$$

Table 1 summaries, the coefficients that relate the penalty functions P_i to θ_j functions (Birgin et al. [56]). The symbol "-" means that the penalty functions P_i was not associated with the θ_j functions. They, for each method have used $\mu_0\in\{10^{-6},1\}$, $\rho_1\in\{10^{-3},1,10\}$, $\tau\in\{10^{-2},0.1,0.5\}$. We have tested these types of penalty functions in

our metaheuristic algorithms, presented in Table 2 the parameters and combination that gave us the best result.

Table 1. The best (**p**,**θ**) combination with their parameters

(P_i, θ_j)	(μ_0, ρ_1)
(7 , 10)	-5 , 1
(7 , 11)	-4 , 1
(7 , 12)	-3 , 1
(7 , 13)	-5 , 1
(8 , 11)	-5 , 10

Table 2. Associations of (P_i, q_j) with their C_{ij} coefficient

	P_4	P_5	P_6	P_7	P_8	P_9
q_1	1	-	-	-	-	-
q_2	1	-	-	-	-	-
q_3	1	-	-	-	-	-
q_4	-	1	1	1	-	-
q_5	-	1	1	1	1	-
q_6	-	1	1	1	1	-
q_7	-	1	1	1	1	-
q_8	-	1	1	1	1	-
q_9	-	1	1	1	1	-
q_{10}	-	1	1	1	1	-
q_{11}	-	8	8	8	1	-
q_{12}	-	1	1	1	1	-
q_{13}	-	1/4	1/4	1/4	1	-
q_{14}	-	1/12	1/12	1/12	1	-
q_{15}	-	1/8	1/8	1/8	1	-
q_{16}	-	2	2	2	-	1
q_{17}	-	2	2	2	-	1
q_{18}	-	2	2	2	-	1

5. HOW TO APPLY

For applying these sampling techniques we have limited the number of our population in each algorithm to 60. Then in each heuristic algorithm we have used our initial sampling techniques and comparing the ratio of their convergence.

For all heuristic algorithms the stopping criteria was set to a maximum number of iteration. In continue for the next part of the work we have presented the charts that show the ratio of convergence of the heuristic algorithms that have used the Penalty-Lagrangian functions, plus the common method of applying penalty function. The term common method means the relation below:

$$\text{Penalty function} = R_p \sum_{i=1}^{n_c} \left[\max\left(\frac{g_i}{g_a} - 1, 0 \right) \right]^2$$

$$R_p = r_1 [1 + r_2 (n_{gen} - 1)] \leq 4 r_1$$

g_i and g_a are the maximum and allowable constraints respectively. R_p is the factor of penalty function, n_{gen} is the generation number, r_1 and r_2 are the constant factors.

6. NUMERICAL EXAMPLES

In this section, common truss optimization examples as benchmark problems are optimized with the proposed methods. The efficiency of proposed methods would be compared with other common methods. The algorithms specific parameters have been discussed in each algorithm themselves. The optimum weights that have been achieved for different initial sampling methods and penalty functions are presented in Tables 3 and 4. The values in these tables are the average of ten analyses run. The algorithms are coded in Matlab and trusses are analyzed using the direct stiffness method.

6.1. Twenty five-bar spatial truss

Figure 4 shows the topology of a 25-bar spatial truss structure. In this example, designs are performed for one loading case. The material density is 0.1 lb/in^3 (2767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The structural members of truss are arranged into eight groups, where all members in a group share the same material and cross-sectional properties. Table 5 defines each element group by member number (each member is defined by its start and end). This spatial truss was subjected to one loading conditions shown in Table 6. Maximum displacement limitations of ±0.35 in. (0.889 cm) were imposed on every node in every direction and the axial stress constraints vary for each group is shown in Table 7. The range of cross-sectional areas varies from 0.01 to 4.14 in^2 (from 0.06452 cm^2 to 26.71 cm^2). This truss was optimized by Camp and Bichon [48], Lee and Geem [49] and Li et al. [50] which respectively got the optimized weight of 545.33lb, 544.38lb, and 627.08lb.

Table 3. The optimum value (lb)

Initial sampling method	Ant colony			GA			PSO		
	244-bar	120-bar	25-bar	244-bar	120-bar	25-bar	244-bar	120-bar	25-bar
Uniform random	4474	16819.9	481.1	5955	17084.4	470.3	6712	17614.9	455.4
Halton	4590	16803.9	472.6	6333	17085.5	465.1	5900	17058	459.4
Hammersley	5227.3	16800	468.9	5913	17134	471.7	6675	17571.8	455.8
LHS	4999	16828	466.86	6043	17061	475.5	6577	16981	461.3
Sobol	5115	16880	481.45	5557	17140	463.3	6822	17319	454.3

6.2. A 120-bar dome truss

A 120-bar dome truss, shown in Figure 4, was first analyzed by Soh and Yang [51] to obtain the optimal sizing and configuration variables, i.e., the structural configuration optimization. In the example considered in this study similar to Lee and Geem [50] and Kelesoglu and Ülker [52], only sizing variables to minimize the structural weight is considered. In addition, the allowable tensile and compressive stresses are used according to the AISC ASD (1989) code [53], as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i^+ \geq 0 \\ \sigma_i^- & \text{for } \sigma_i^+ < 0 \end{cases} \qquad \sigma_i^- = \begin{cases} \left[\left(1-\frac{\lambda_i^2}{2C_c^2}\right)F_y\right] \Big/ \left(\frac{5}{3}+\frac{3\lambda_i}{8C_c}-\frac{\lambda_i^3}{8C_c^3}\right) & \text{for } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases}$$

Table 4. The optimum value (lb)

Penalty Functions	Ant colony			GA			PSO		
	244-bar	**120-bar**	**25-bar**	**244-bar**	**120-bar**	**25-bar**	**244-bar**	**120-bar**	**25-bar**
Common method	4753	16819	470	6102	16764	479	4875	16500	476
P1	4091	16679	501	5863	16740	485	4612	16433	478
P7_θ6	4210	16816	482	5717	16770	489	4702	16404	467
P7_θ9	4044	16798	526	5139	19027	502	4307	16428	477
P7_θ10	3900	17843	562	4851	16724	479	4841	16731	491
P7_θ11	4044	16809	515	5566	16788	484	4433	16500	483
P7_θ12	4129	16824	510	5742	16744	491	4340	16415	480
P7_θ13	4215	16640	482	5840	16765	486	4937	16507	466

P8_θ11	4189	16774	475	5485	16747	487	4716	16721	470

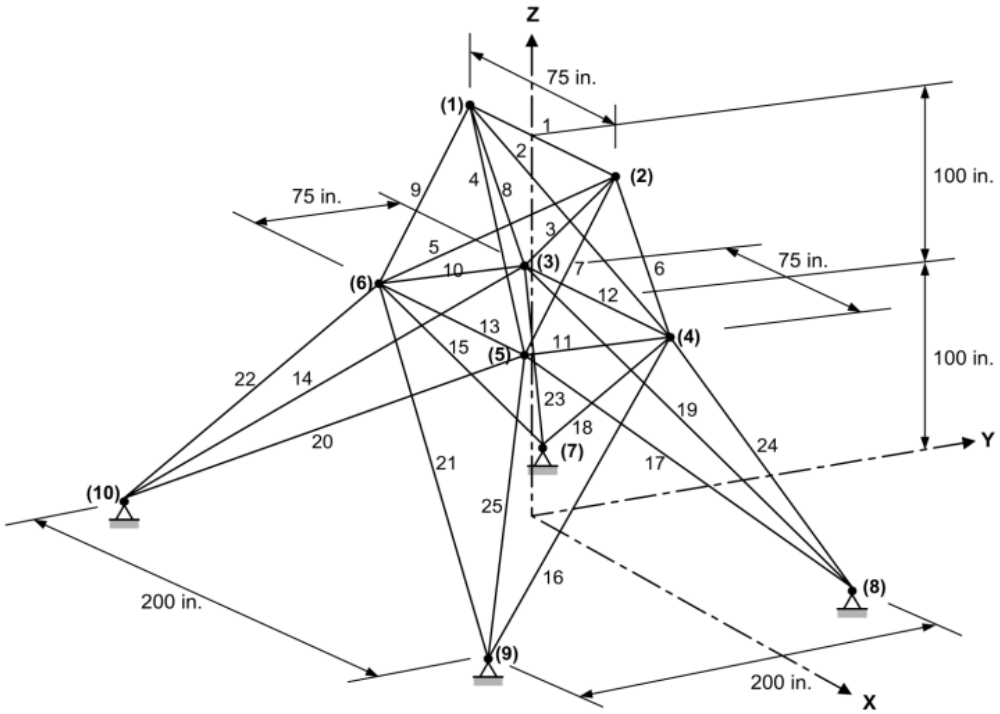

Figure 4. A 25-bar spatial truss

Table 5. Element information for the 25-bar spatial truss

Element group number							
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1:(1,2)	2:(1,4)	6:(2,4)	10:(6,3)	12:(3,4)	14:(3,10)	18:(4,7)	22:(10,6)
	3:(2,3)	7:(2,5)	11:(5,4)	13:(6,5)	15:(6,7)	19:(3,8)	23:(3,7)
	4:(1,5)	8:(1,3)			16:(4,9)	20:(5,10)	24:(4,8)
	5:(2,6)	9:(1,6)			17:(5,8)	21:(6,9)	25:(5,9)

Table 6. Loading condition for the 25-bar spatial truss

Node	**Px kips(kN)**	**Py kips(kN)**	**Pz kips(kN)**
1	1	10	-5
2	0	10	-5
3	0.5	0	0

6	0.5	0	0

Table 7. Member stress limitation for the 25-bar spatial truss

	Element group	Compressive stress limitations ksi (Mpa)	Tensile stress limitations ksi (Mpa)
1	A(1)	35.092 (241.96)	40 (275.80)
2	A(2)~A(5)	11.59 (79.913)	40 (275.80)
3	A(6)~A(9)	17.305 (119.31)	40 (275.80)
4	A(10)~A(11)	35.092 (241.96)	40 (275.80)
5	A(12)~A(13)	35.092 (241.96)	40 (275.80)
6	A(14)~A(17)	6.759 (46.603)	40 (275.80)
7	A(18)~A(21)	6.959 (47.982)	40 (275.80)
8	A(22)~A(25)	11.082 (76.41)	40 (275.80)

where σ_i^- is calculated according to the slenderness ratio, E is the modulus of elasticity, F_y is the yield stress of steel, C_c is the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions ($C_c = \sqrt{(2\pi^2 E/F_y)}$, ($\lambda_i$) is the slenderness ratio ($\lambda_i = k\, L_i/r_i$), k is the effective length factor, L_i is the member length and ri is the radius of gyration. The modulus of elasticity is 30,450 ksi (210,000 MPa) and the material density is 0.288 lb/in^3 (7971.810 kg/m3). The yield stress of steel is taken as 58.0 ksi (400 MPa). On the other hand, the radius of gyration (r_i) can be expressed in terms of cross-sectional areas, i.e., $r_i = aA_i^b$ [54]. Here, a and b are the constants depending on the types of sections adapted for the members such as pipes, angles, and tees.

In this example, pipe sections (a=0.4993 and b = 0.6777) were adapted for bars. All members of the dome are linked into seven groups, as shown in Figure 5. The dome is considered to be subjected to vertical loading at all the unsupported joints. These were taken as -13.49 kips (60 KN) at node 1, -6.744 kips (30 KN) at nodes 2 through 14, and -2.248 kips (10 KN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in2 (5cm2) and the maximum cross-sectional area is 4.18 in2 (26.96 cm2). For constraints we have stress constraints and displacement limitations of ±0.1969 in. (5 mm) imposed on all nodes in x and y-directions.

6.3. A 244-bar transformation tower

Another space truss, a 244-bar transmission tower is shown in Figure 6, is examined as the final design problem. Members of the transmission tower are initially collected into 26 groups as given by Saka [54] but in this study all members of the transmission tower are linked into 32 groups. The value of the modulus of elasticity is taken as 30450 ksi (210000 MPa) and the material density is 0.1 lb/in^3 (2767.990 kg/m^3).The allowable value of 20.30 ksi (140 MPa) is employed for tensile stresses and the formulation of buckling obeying AISC ASD (1989) code [53] is considered for compressive stresses (similar to the dome example). The displacement limitations of ±1.77 in (4.5cm) are imposed on nodes 1 and 2, and limitations of ±1.18 in. (3.0 cm) on nodes 17, 24 and 25 in x-direction. These nodes are subjected to the displacement limits of ±0.59 in. (1.5 cm) in z-direction. The load cases considered is taken from the study by ULKER [55] and are shown in Table 8. The minimum cross-sectional area of all members

is 0.775 in^2 (5.0 cm^2) and the maximum cross-sectional area is 18.2 in^2 (129.03 cm^2). To understand the coordination of shape, for node 25 we have x=500cm y=100 z=1400cm.

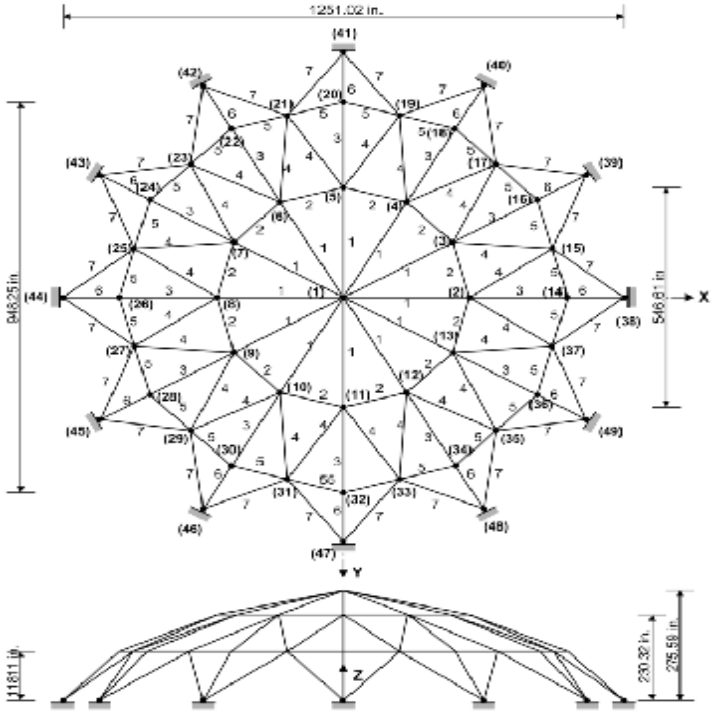

Figure 5. A 120-bar dome truss

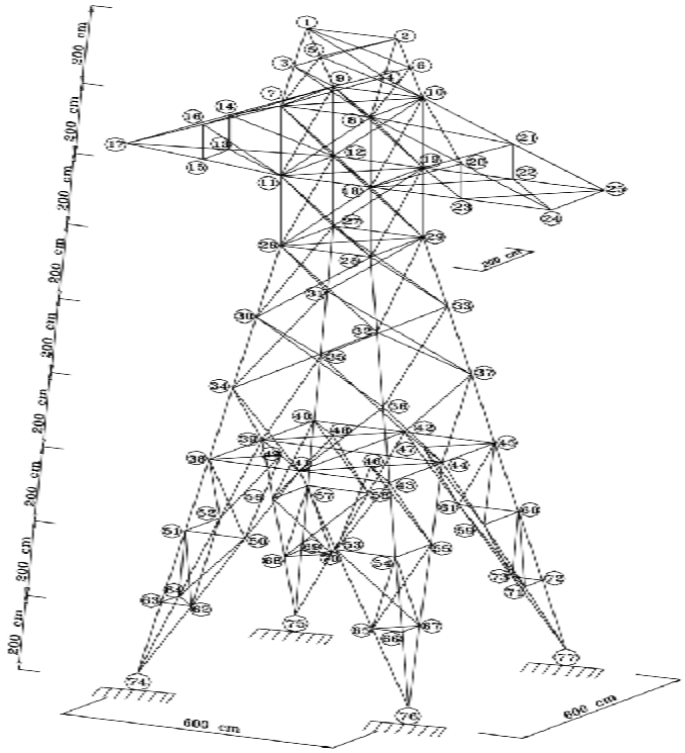

Figure 6. A 244-bar transformation tower

Table 8. The loading and displacement bounds of transmission tower

Joint Number	Loading (KN)		Displacement Limitation (mm)	
	X	Z	X	Z
1	-10	-30	45	15
2	10	-30	45	15
17	35	-90	30	15
4	175	-45	30	15
25	175	-45	30	15

7. DISCUSSION ON THE EFFICIENCY OF THE SAMPLING TECHNIQUES

Figures 7 to 9 show the ratio of convergence (the best weight against iteration) for each heuristic algorithm, we have applied our initial sampling methods for their first population. The results are the average of ten analyses run for each method, and for each algorithm the number of sample in each iteration was 60. The rest of the charts have been presented in the Appendix.

In Table 9 we have ranked the initial sampling methods for each heuristic algorithm, according to their weight gotten in the last iteration. It should be mentioned that the penalty function has been used here was the common method of section 5. The results are the average among three examples. As you see the Halton sequence has gotten the first place among three algorithms. If we consider the three heuristic algorithms altogether we can see that in the second place we have LHS and Hammersley and Sobol sequence. Besides these in the third place we have Uniform random method.

Table 9. Ordering of the initial sampling methods according to their ratio of convergence

Convergence rank	**Ant colony**	**GA**	**PSO**	**Average of 3 algorithms**
1	Halton	Halton	Halton	Halton
2	Sobol	LHS	Sobol	(LHS – Hammersley-Sobol)
3	LHS – Hammersley	Hammersley	Hammersley	Uniform random
4	Uniform random	Uniform random	LHS	--
5	--	Sobol	Uniform random	--

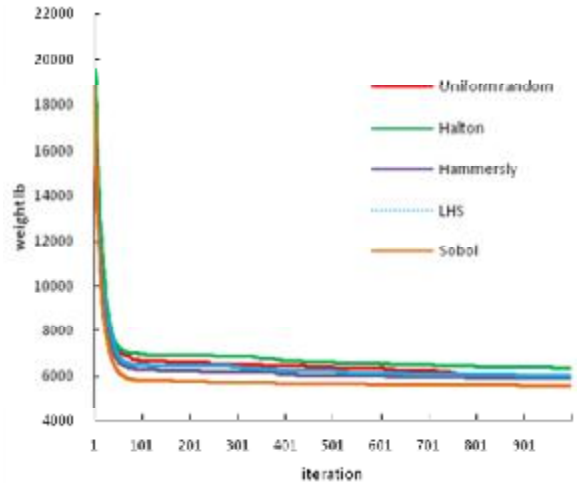

Figure 7. Convergence rate comparison between the five initial sampling methods for the 244-bar spatial truss in GA algorithm

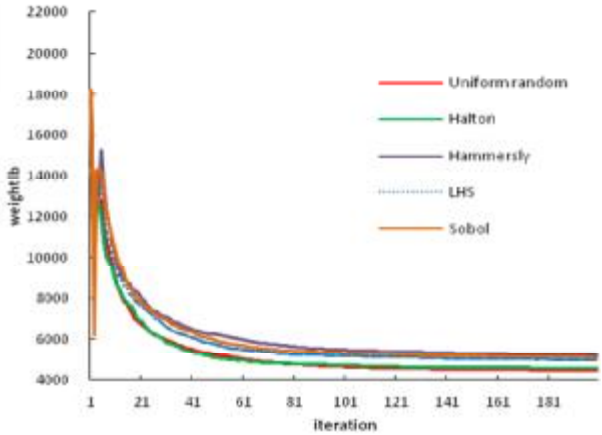

Figure 8. Convergence rate comparison between the five initial sampling methods for the 244-bar spatial truss in ACO algorithm

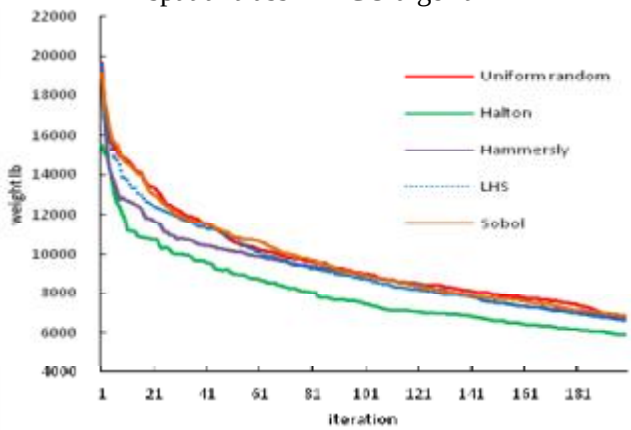

Figure 9. Convergence rate comparison between the five initial sampling methods for the 244-bar spatial truss in PSO algorithm

8. DISCUSSION ON THE EFFICIENCY OF THE PENALTY-LAGRANGIAN FUNCTIONS

In the Figures 10 to 12, we have shown the history of each penalty method, for each heuristic algorithm. The results are the average of ten analysis run for each penalty method. For generating the initial samples here we have used the RANDINT function of matlab programming, that generate a random scalar that is 0 or 1 with equal probability and also do elements uniformly distributed in the range specified. The rest of the charts have been presented in the Appendix. The stopping criterion for our algorithms was a maximum number of iteration. As you see it's different for each algorithm and each example. For example in the 244-bar transmission tower, for Ant colony and PSO the maximum number of iteration was set to 200, but in GA it was set to 1000. That is due to the fact that, the Ant colony and PSO algorithms would reach to optimum value up to 200 iteration but the GA algorithm would not reach to it up to 1000 iteration.

Here again in Table 10 we have ranked our penalty functions for three heuristic algorithms, according to the speed of their convergence. The results are average from three examples. As you see in Table 10 the common method is the last row in the column of average of three algorithms, i.e. Penalty-Lagrangian functions can speed up the convergence rate faster than the common method. To see the effect of number of variable on convergence speed, we have ranked the penalty functions for our three examples in Table 11 (with different number of variables), you see, as the problem gets more complicated, the application of the common method for penalty functions causes the algorithms converge slower to the optimum value. We have used each penalty function in each heuristic algorithm and got the best weight in the last iteration. The results here have been ranked according to taking average of three gotten weights.

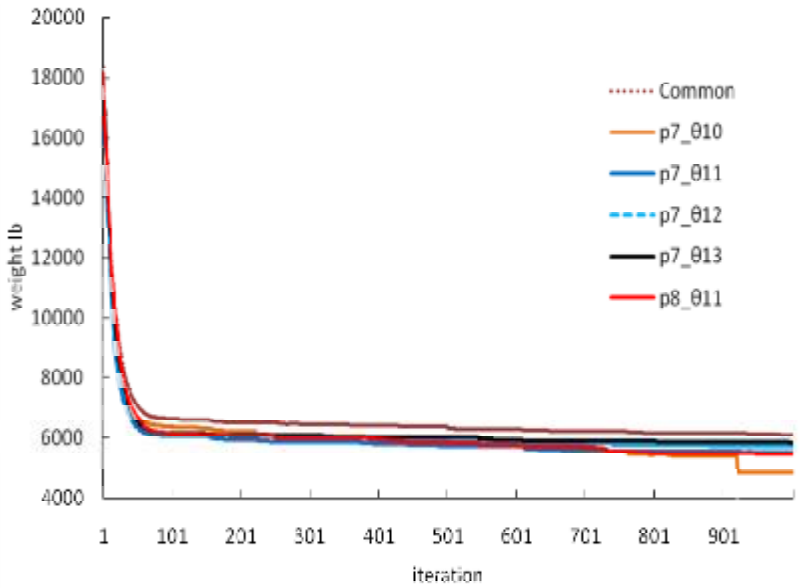

Figure 10. Convergence rate comparison between the Penalty-Lagrangian functions and common method for the 244-bar spatial truss in GA algorithm

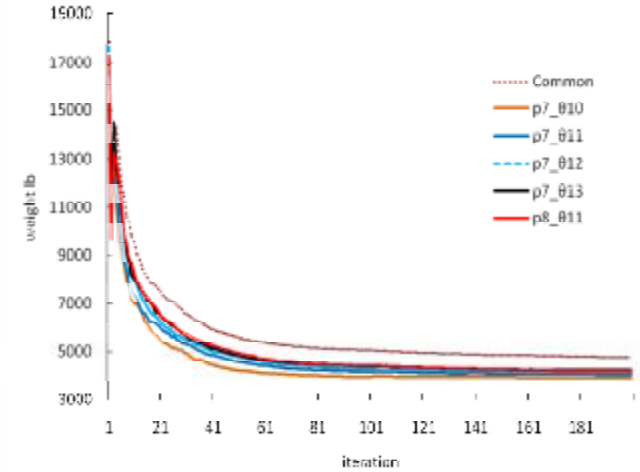

Figure 11. Convergence rate comparison between the Penalty-Lagrangian functions and common method for the 244-bar spatial truss in ACO algorithm

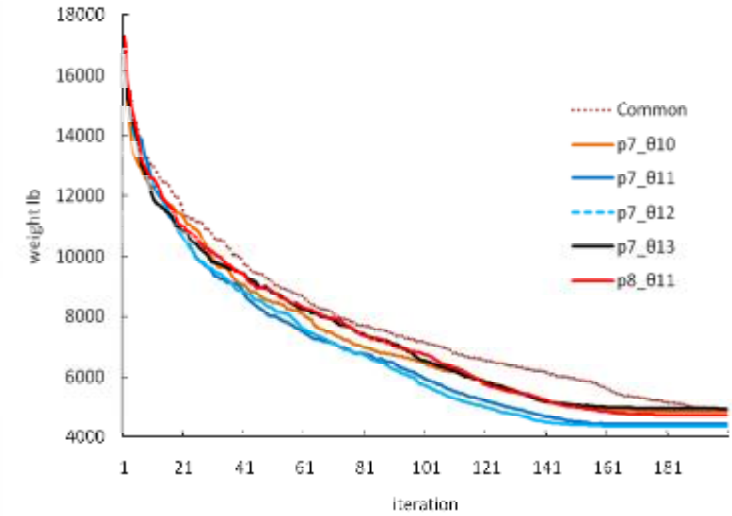

Figure 12. Convergence rate comparison between the Penalty-Lagrangian functions and common method for the 244-bar spatial truss in PSO algorithm

Table 10. Ordering of the Penalty-Lagrangian functions according to their ratio of convergence

Convergence rank	Ant colony	GA	PSO	Average of 3 algorithms
1	P7_θ13	P7_θ10	P7_θ12	P7_θ11 , P8_θ11
2	P8_θ11	P7_θ11	P7_θ11 , P7_θ13	P7_θ12
3	P7_θ11	P8_θ11	P8_θ11	P7_θ10 , P7_θ13
4	Common method , P7_θ12	Common method	Common method	Common method
5	P7_θ10	P7_θ12	P7_θ10	--

6	--	P7_θ13	--	--

Table 11. Ordering of the Penalty-Lagrangian function according to their ratio of convergence

Convergence rank	244-bar truss	120-bar truss	25-bar truss	Average of 3 examples
1	P7_θ10 , P7_θ11	P7_θ12	Common method	P7_θ11 , P8_θ11
2	P7_θ12	P7_θ13	P7_θ13	P7_θ12
3	P8_θ11	P7_θ11 , P8_θ11	P8_θ11	P7_θ10 , P7_θ13
4	P7_θ13	Common method	P7_θ11	Common method
5	Common method	P7_θ10	P7_θ10	--
6	--	--	P7_θ12	--

9. CONCLUDING REMARKS

In this paper we have used some sampling techniques such as the Latin Hypercube sampling, the Sobol, Halton, and Hammersley sequences for the selection of first initial population in the metaheuristic algorithms; GA, Ant Colony and PSO. These sampling methods have been used widely in metamodeling techniques, but rarely for our application.

What we have obtained here is that altogether the Halton sequence has gotten the first place in convergence rate among other sampling methods. After that the LHS, the Hammersley and the sobol sequence has taken the next places respectively. For the last place we had the Uniform random method. Finally in the application of Penalty-Lagrangian functions we have seen that the use of methods; p7_ θ11, p8_ θ11, p7_ θ12, p7_ θ10, p7_13, common method, has taken the ranking places for convergence rate respectively.

REFERENCES

1. Holland J.H. Adaptation in Natural and Artificial Systems, Ann Arbor: *The University of Michigan Press* 1975.
2. Goldberg D.E., and samtani M.P. Engineering Optimization via Genetic Algorithms, *Proc. Of 9th Conf. on Electronic Computation, ASCE, New York, N. Y.* 1986; 471-82.
3. Ahn CW, Ramakrishna RS. A genetic algorithm for shortest path routing problem and the sizing of populations, *IEEE Trans Evolut Comput* 2002; **6**(6).
4. Park C.H., Lee W.I., Han W.S., & Vautrin A. Weight minimization of composite laminated plates with multiple constraints, *Composites Science and Technology* 2003; **63**: 1015–26.
5. Kaveh A., Khanlari K. Collapse load factor of planar frames using modified genetic algorithm, *Common Number Math Eng* 2004; **20**: 911–25.
6. Sahab MG, Ashour AF, Toropov VV. Cost optimization of reinforced concrete flat slab buildings, *Engineering Structures* 2005; **27**: 313–22.

7. Castilho VC, El Debs MKE, Nicoletti MC. Using a modified genetic algorithm to minimize the production costs for slabs of precast prestressed concrete joists, *Engineering Applications of Artificial Intelligence* 2007; **20**: 519–30.
8. Dorigo M. Optimization learning and natural algorithms (in Italian). Ph.D. thesis, Dipartimento di,Elettronica, Politecnico di Milano, Milan 1992.
9. Bonabeau E, Dorigo M. Theraulaz G. Swarm intelligence: from natural to artificial systems, New York: *Oxford University Press* 1999.
10. Dorigo M, Maniezzo V, Colorni A. The Ant System: An autocatalytic optimizing process, *Technical report 91-016 revised, Dipartimento di Elettronica, Politecnico di Milano*, Milan 1991.
11. Dorigo M, Maniezzo V, Colorni A. Ant System: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics—Part B* 1996; **26(1)**: 29–41.
12. Gambardella LM, Dorigo M. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In A. Prieditis & S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning (1995); Palo Alto, CA, Morgan Kaufmann.* (ML-95) (pp. 252–260).
13. Dorigo M, Gambardella LM. Ant colonies for the traveling salesman problem, *BioSystems* 1997; **43(2)**: 73–81.
14. Dorigo M, Gambardella LM. Ant Colony System: A cooperative learning approach to the traveling salesman problem, *IEEE Trans Evol Comput* 1997; **1(1)**: 53–66.
15. Stutzle T, Hoos HH. Improving the Ant System: A detailed report on the MAX-MIN Ant System, *Technical report AIDA-96-12, FG Intellektik, FB Informatik, TU Darmstadt, Germany* 1996.
16. Bullnheimer B, Hartl RF, Strauss C. A new rank-based version of the Ant System: A computational study. *Cent Eur J Oper Res* 1999; **7**(1): 25–38.
17. Maniezzo V. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, *INFORMS Journal on Computing* 1999; **11**(4): 358–69.
18. Blum C, Roli A, Dorigo M. HC–ACO: The hyper-cube framework for Ant Colony Optimization. *In Proceedings of MIC'2001—Metaheuristics International Conference* 2001; **2**, 399–403.
19. Eberhart R, Kennedy J. New optimizer using particle swarm theory, *In: Sixth international symposium on micro machine and human science, Nagoya, Japan* 1995; 39–43.
20. Kennedy J, Eberhart R. Particle swarm optimization, *In: IEEE International Conference on Neural Networks, Piscataway, NJ* 1995; **4**: 1942–8.
21. Coello C, Luna E. Use of particle swarm optimization to design combinational logic circuits, In: Tyrell A, Haddow P, Torresen J, editors. *5th International conference on evolvable systems: from biology to hardware, ICES 2003. Lecture notes in computer science, Trondheim, Norway: Springer* 2003; **2606**: 398–409.
22. Zheng Y, Ma L, Zhang L, Qian J. Robust pid controller design using particle swarm optimizer, *In: IEEE International Symposium on Intelligence Control* 2003; 974–9.
23. Abido M. Optimal design of power system stabilizers using particle swarm optimization, *IEEE Trans Energy Convers* 2002; **17(3)**:406–13.
24. Fourie P, Groenwold A. The particle swarm optimization algorithm in size and shape

optimization, *Struct Multidiscip Optim* 2002; **23(4)**:259–67.
25. Venter G, Sobieszczanski-Sobieski J. Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization, *Struct Multidiscip Optim* 2004; **26**(1–2):121–31.
26. McKay MD, Beckman RJ, Conover WJ. A Comparison of Three Methods for Selecting Values of Input Variables in The Analysis of Output from a Computer Code, *Technometrics (American Statistical Association)* May 1979; **21 (2):** 239–45.
27. Imam RL, Helton JC, Campbell JE. An approach to sensitivity analysis of computer models, Part 1. Introduction, input variable selection and preliminary variable assessment, *Journal of Quality Technology* 1981; **13 (3)**:174-83.
28. Sobol IM. Distribution of points in a cube and approximate evaluation of integrals, *U.S.S.R Comput Maths MathPhys* 1967; **7**: 86-112.
29. Bratley P, Fox BL. Algorithm 659: Implementing Sobol's quasirandom sequence generator, *ACM Trans Math Softw* 1988; **14**: 88–100.
30. Niederreiter H. Low-discrepancy and low-dispersion sequences, *Journal of Number Theory* 1988; **30**:51-70.
31. Antonov IA, Saleev VM. An economic method of computing LP_τ-sequences, Zh. *Vych. Mat. Mat. Fiz.* 1979; **19**: 243–245 (in Russian); *U.S.S.R Comput. Maths. Math. Phys* 1979; **19**: 252–256 (in English).
32. Jäckel P. Monte Carlo Methods in Finance, New York: Wiley 2002.
33. Press W.H., Teukolsky S.A., Vetterling W.T., and Flannery B.P. Numerical Recipes in Fortran 77: The Art of Scientific Computing, 2nd ed, *Cambridge University Press, Cambridge, U.K* 1992.
34. Paskov SH, Traub JF. Faster valuing of financial derivatives, *J Portfolio Manag* 1995; **22**:113–20.
35. Traub J. In math we trust. What's Happening in the Mathematical Sciences 1996; **3**:101–111.
36. Case J. Wall street's dalliance with number theory. *SIAM News* December 1995; 8 – 9.
37. Heinrich S, Keller A. Quasi-monte carlo methods in computer graphics, part i: The qmc buffer. *Technical report, University of Kaiserslautern* 1994; 242/94.
38. Heinrich S, Keller A. Quasi-monte carlo methods in computer graphics, part ii: The radiance equation. *Technical report, University of Kaiserslautern* 1994; 243/94.
39. Keller A. A quasi-monte carlo algorithm for the global illumination problem in the radiosity setting. *In Proceedings of Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Springer-Verlag* June 1995; 239–51.
40. Ohbuchi R, Aono M. Quasi-monte carlo rendering with adaptive sampling. *Technical report, Tokyo Research Laboratory, IBM Japan Ltd* 1996.
41. Kalagnanam JR, Diwekar UM. An efficient sampling technique for off-line quality Control, *Technometrics* 1997; **39**(3): 308-19.
42. Diwekar UM, Kalagnanam JR. Efficient sampling technique for optimization under. uncertainty, *AIChE J* 1997; **43(2)**, 440-57.
43. Subramanyan K. and Diwekar U.M. User Manual for Fortran-Based Stochastic Sampling Code, 2006.
44. Halton J. On the efficiency of certain quasirandom sequences of points in evaluating

multidimensional integrals, *Numerische Mathematik* 1960; **2**: 84–90.
45. Fourie P, Groenwold A. Particle swarms in size and shape optimization. In: Snyman, J.; Craig, K. (eds.) Proc. *Workshop on Multidisciplinary Design Optimization (held in Pretoria)* 2000; 97–106.
46. Fourie P, Groenwold A. The particle swarm optimization in topology optimization, *In: Fourth world congress of structural and multidisciplinary optimization*, Paper no. 154, Dalian, China 2001.
47. Shi Y, Eberhart R. A modified particle swarm optimizer, *In: IEEE international conference on evolutionary computation. IEEE Press,Piscataway, NJ* 1998; 69–73.
48. Camp C, Bichon J. Design of space trusses using ant colony optimization, *J Struct Eng ASCE* 2004; **130(5)**:741–51.
49. Li LJ, Huang ZB, Liu F, Wu QH. A heuristic particle swarm optimizer for optimization of pin connected structures, *Comput Struct* 2007; **85**:340–9.
50. Lee KS, Geem ZW. A new structural optimization method based on the harmony search algorithm, *Comput Struct* 2004; **82**:781–98.
51. Soh CK, Yang J. Fuzzy controlled genetic algorithm search for shape optimization. *J Comput Civil Eng ASCE* 1996; **10(2)**:143–50.
52. Kelesoglu O, Ülker M. Fuzzy optimization geometrical nonlinear space truss design. *Turkish J Eng Environ Sci* 2005; **29**:321–9.
53. American Institute of Steel Construction (AISC). *Manual of steel construction allowable stress design. 9th ed. Chicago, IL*; 1989.
54. Saka MP. Optimum design of pin-jointed steel structures with practical applications, *J Struct Eng ASCE* 1990; **116(10)**:2599–620.
55. Ulker M, Hayalioglu MS. Optimum design of space trusses with buckling constraints by means of spreadsheets, *Turk J Engin Environ Sci*, 2001; **25**: 355-367.
56. Birgin R, Castillo A, Martinez JM. Numerical comparison of Augmented Lagrangian algorithms for nonconvex problems, 2004.
57. Hestenes M.R. Multiplier and gradiant methods, *J Optim Theory and Applics* 1969; **4**: 303-20.
58. Powell MJD. A method for nonlinear constraints in minimization problems, *in Optimization, R. Fletcher (ed.), Academic Press, New York, NY* 1969; 283-98.
59. Rockefellar RT. The multiplier method of Hestenes and Powell applied to convex programming, *Journal of Optimization Theory and Applications*, 1973; **12**: 555-562.
60. Montgomery DC. Design and Analysis of Experiments, Fourth Edition, *John Wiley & Sons, New York* 1997.
61. Baker J. Reducing bias and inefficiency in the selection algorithm, *in: Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Massachusetts Institute of Technology*, July 1987; 14-21.
62. Grefenstette J. Optimization of control parameters for genetic algorithms, *IEEE Trans Systems, Man, and Cybernetics SMC* 1986; **16** (1): 122-8.
63. Schaffer J, Caruana R, Eshelman L, Das R. A study of control parameters affecting online performance of genetic algorithms for function optimization, *in: Proceedings of the Third International Conference on Genetic Algorithms, George Mason University* June 1989; pp. 51-60.