Tuned Annealing for Optimization

Mir M. Atiqullah[1] and S. S. Rao[2]

[1]Aerospace and Mechanical Engineering Department
Saint Louis University
St. Louis, MO 63103
[2]Department of Mechanical Engineering
University of Miami
Coral Gables, FL 33146

Abstract. The utility and capability of simulated annealing algorithm for general-purpose engineering optimization is well established since introduced by Kirkpatrick et. al[1]. Numerous augmentations are proposed to make the algorithm effective in solving specific problems or classes of problems. Some proposed modifications were intended to enhance the performance of the algorithm in certain situations. Some specific research has been devoted to augment the convergence and related behavior of annealing algorithms by modifying its parameters, otherwise known as cooling schedule. Here we introduce an approach to tune the simulated annealing algorithm by combining algorithmic and parametric augmentations. Such tuned algorithm harnesses the benefits inherent in both types of augmentations resulting in a robust optimizer. The concept of 'reheat' in SA, is also used as another tune up strategy for the annealing algorithm. The beneficial effects of 'reheat' for escaping local optima are demonstrated by the solution of a multimodal optimization problem. Specific augmentations include handling of constraints, fast recovery from infeasible design space, immunization against premature convergence, and a simple but effective cooling schedule. Several representative optimization problems are solved to demonstrate effectiveness of tuning annealing algorithms.

Keywords

Simulated annealing, design optimization, constrained optimization, tuned annealing, cooling schedule.

1 Introduction

In pursuit of high quality solutions, design optimization algorithms not only must find near optimum end result but also demonstrate efficiency in terms of computation. As a stochastic algorithm, the simulated annealing is well known for its capability to find the globally optimal solution. Guided probabilistic moves are key to finding global optimum by simulated annealing while overcoming local optima in the design space. The basic features of the annealing algorithm can be highlighted by the following pseudocode:

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2074, pp. 669–679, 2001.

Program annealing
Set initial probability of accepting randomly generated worse design
Randomly generate a new solution by probabilistic move from current solution
If better than before accept it as next solution
If worse, accept it only with a certain probability and decrease it for next iteration..
Repeat the process by generating a new solution, until no improvement possible.
End program.

When used with limited probabilistic control, the objective function is improved over many such steps. But often its iterative and slow convergence can be prohibitive for problems with many design variables and lengthy function evaluations.

Many global optimization methods reported in the lierature[2,3] are stochastic in nature and converge asymptotically to the global optimum. When the modality of the design domain is not overly complex, these methods proved efficient. Many practical design optimization problems involve a large number of design parameters with objective functions characterized by many local minima. While the algorithm draws parallel to the physical heat treatment annealing of metals, the acceptance probability used for accepting occasional worse design draws its parallel to the temperature. The algorithm and logic used to progressively decrease this temperature is widely known as cooling schedule. Because of its discrete nature, the annealing algorithm can overcome non-smoothness and discontinuities in the design space. The convergence and global characteristics of simulated annealing (SA) are strongly affected by the structure of annealing and parameters of the cooling schedule. The analyses of several innovative cooling schedules are discussed and a parametric schedule is proposed, which is adaptive to problem at hand and the nature of the design domain and follows a 'gaussian' type decrement. Two combinatorial test problems are used as a platform for comparing effectiveness of various cooling schedules with that of the tuned algorithm.

No design should be considered complete without performing some form of optimization of the initial design. Traditional design approach used localized or specific performance based design improvement without considering globality of the process. This is mainly because of the inability of the traditional gradient based optimizers to find the global optimum.

The goal of this paper is to demonstrate the simplicity and the utility of the tuned annealing algorithm. Section two gives the algorithmic steps and the implementation details of tuned annealing algorithm. Numerical examples include a welded beam design, a 32 variable part stamping problem, a 200 variable part stamping problem, and a 25-bar space truss optimization, providing an spectrum of highly nonlinear and multimodal design problems.

2 Tune ups #1 and #2: Improve Feasibility and Handle Constraints

Simulated annealing responds to changes only in the objective function. Since almost all engineering design problems are highly constrained, SA would not be suitable for solving such problems. To take advantage of the global solution capability of SA and to alleviate this difficulty, it is customary to incorporate the constraint functions $g(x)$ to the design objective using a penalty function $P(x)$ such that:

$$P(\bar{x}) = F(\bar{x}) + r_k \sum_{j=1}^{m} G_j\left(g_j(\bar{x})\right) \tag{1}$$

where, $G_j = \max\left[0, g_j(\bar{x})\right]$ for inequality constraints and $G_j = abs\left[g_j(\bar{x})\right]$ for equality constraints. The factor r_k determines the relative degree of penalty. Two inherent problems plague the penalty function approach that usually slows the SA down and often leads the solution towards sub optimal solution. First, design problems with a large number of constraints with a large difference in the numerical values of the constraint functions pose special difficulty for the SA algorithm. Second, highly constrained problems may even pose difficulty in finding a feasible solution in the first place.

Ideally all constraints should be treated with equity and any design solution violating any constraint should be considered in comparable terms. To guide the annealing process to handle infeasible design space a constraint navigational strategy has been used successfully[2]. This approach takes the constraints into account explicitly, alleviates such scaling problems, and enhances the convergence through fast achievement of feasibility when the starting solution is infeasible (and random). Each iteration of such tuned SA algorithm can be described by the following pseudo-code:

Randomly perturb the variables: obtain a new design.
Evaluate the objective function and the constraints.
IF no constraint is violated proceed with regular annealing.
ELSEIF the number of constraints violated increases, probabilistically accept the design and go to the next iteration.
ELSEIF the number of constraints violated decreases, unconditionally accept the design and go to the next iteration.
ELSEIF the number of constraints violated remains the same
AND the amount of violation increases, probabilistically accept the design
ELSE
accept the design.
ENDIF
Proceed to the next iteration.

This tune up helps get rid of penalty function as well as scaling of constraints as often necessary in untuned algorithm.

3 Tune Up #3: Reheat

It is one of the most ambiguous parameters in annealing to determine with some certainty when to stop the random search process and delare the solution as optimal. While various cooling schedules are proposed[3,4] using statistical methods to sense closeness to optimality, numerical implementation of those are virtually impractical due to the requirement of prohibitively long Markov chains i.e. sequence of random probing of the design space. As a result algorithms have to be terminated after finite annealing process with uncertainty in the optimality of results. The situation is further complicated when real valued variables make up the design space and certain discretization is used to simulate the real space. For convergence purposes, the randomly selected steps in the random

directions must be gradually decremented in magnitude. As the SA algorithm approaches the end of a Markov chain (long sequence of random steps), the steps become too small to continue searching far from the current solution, which practically halts the SA. As a compromise between excessively long annealing process and uncertain optimal result, a reheat strategy[2] has been used. Basically multiple SA algorithm runs are executed in a sequence such that the one SA algorithm picks up a solution as its starting point, which is left off by the previous one. It is essentially a re-annealing strategy with a stopping criteria built into it. The pseudocode for the reheat is given below:

Program Reanneal
Preset annealing and reannealing parameters
Anneal for a preset number of iterations. Results in set A
Reset temperature to original (or determine for the neighborhood of the current solution A)
Reanneal for the preset number of iterations. Reusults in set B.
If B is better than A, replace A by B. Continue to Reanneal.
If B is close or worse than A, terminate reannealing. B is the current solution.
End Reanneal.

Usually the only problems that may improve successively with each reanneal are those with multimodal objectives functions. The added benefit is that by setting shorter markov chains in the annealing and using multiple reannealing, optimal results may be found without lengthy annealing run, even for the unimodal functions with just single optimal solution.

4 Tune Up #4: Simple Cooling Schedule

An annealing algorithm can be made quite robust by selecting and implementing a proper cooling schedule irrespective of the algorithmic modifications discussed earlier. Much research has been devoted to this aspect. A cooling schedule is defined by a set of parameters governing the finite time behavior of the SA algorithm. The aim in the selection of the parameters is to closely follow the asymptotic behavior of the homogeneous algorithm using inhomogeneous implementation in finite time. The cooling schedules which use information from the objective evaluation and use it to adapt their annealing strategy, adjusts themselves for the design space at hand. For virtually all engineering design problems with multimodal objectives and large number of constraints and design variables, the capability to find globally optimal solution is extremely valuable. Several well-known schedules are discussed followed by the introduction of a simple schedule. Our schedule draws power from the adaptive probing of the design space. Moreover the decrement of the *temperature*, which is the control parameter of the cooling schedule, is designed to follow certain heuristics as well as characteristics of random search. While much simpler, our schedule performed better than or as well as several published schedules tested. The structure of any cooling schedule can be described by the following three parameters:

Initial temperature: The value of the initial temperature does not affect the adaptive characteristics of the cooling schedule but is critical for enabling the SA algorithm to overcome local optima. Initially, a quasi-equilibrium state can be assumed by choosing the initial value of the temperature such that virtually all transitions have guaranteed

probability of acceptance. Many of the so-called adaptive schedules[4,5,6,7] draw their success from probing the neighborhood of the current design and determining the initial temperature. Starting with a too high temperature will unnecessarily prolong the already long SA process. As the algorithm progresses, the temperature must approach a value of zero, so that no worse solutions will be accepted. Then the algorithm virtually will not achieve any more significant improvement in the objective function. This tapering convergence is again much linked with any decreasing step length, specially if real valued design space is handled by discrete stepping of the SA.

Length of Markov chain: The number of transitions attempted at any specific temperature is called the length L_k of the Markov process at the kth step of temperature decrement. For finite time implementation, the chain length is governed by the notion of *closeness* of the current probability distribution a_{L_k,t_k} at temperature t_k to the stationary distribution q_{t_k}. The adaptive schedules have taken different approaches with different assumptions and preset conditions to determine when such *closeness* is achieved[8].

Decrement rule for temperature: The way the temperature is decremented is directly related to the notion of quasi-equilibrium of the probability distribution of configurations. It is intuitively obvious that a large decrement in the temperature t_k, at the k-th Markov chain, will necessitate a large number of transitions L_{k+1} at the (k+1)-th Markov chain before a quasi-equilibrium state is restored. Most adaptive cooling schedules follow the strategy of small decrements in temperature t_k to avoid long Markov chains L_k, albeit at the cost of increased number of temperature steps.

4.1 An Adaptive Schedule

The cooling schedules can be divided into two broad groups, static and adaptive. The schedules, which follow a predetermined course of decrement, are termed static while those using some statistical analysis of visited cost/design objective to control their decrement are known as dynamic. The static cooling schedules generally disregard the dynamic behavior of the problem at hand, often place too many restrictions on the annealing process. On the other hand, dynamic schedules, being computationally intensive, increase the computational effort many folds. To combine the beneficial characteristics of both classes, a simple hybrid schedule is proposed with two control parameters. The initial temperature t_0 should be high enough so that all configurations are equally admissible for acceptance at the initial temperature. In our approach, we include all moves for such estimations. In earlier works, the cost decreasing moves were not considered in the computation of the initial acceptance probability χ_0.

Thus, the augmented initial acceptance ratio is defined as

$$\chi_0 = \frac{\text{no. of accepted moves}}{\text{no. of proposed moves}} \approx 1.00 \approx \exp\left\{-\frac{\left(\overline{\Delta C} + 3\sigma_{\Delta C}\right)}{t_0}\right\} \tag{2}$$

which leads to the new rule for the computation of the initial temperature:

$$t_0 = \frac{\left(\overline{\Delta C} + 3s_{\Delta C}\right)}{\ln\left(1/c_0\right)} \tag{3}$$

Experiments with several design problems, with arbitrary initial starting configurations (designs), suggest that the value of t_0 is increased by 50% or more when Eqn. (3) is used compared when standard deviation is not used.

The time to arrive at a quasi-equilibrium is related to the size of neighborhood ($\Re$). We propose by saying that the chain can be terminated if either the number of acceptances or rejections reach a certain number $\Lambda|\Re|$, where Λ is a multiplication factor. That is,

$$L_k = \begin{cases} m1 + \Lambda|\Re|; \{m2 = \Lambda|\Re|, m1 < m2\} \\ m2 + \Lambda|\Re|; \{m1 = \Lambda|\Re|, m2 < m1\} \end{cases} \tag{4}$$

where m1 and m2 are the cost decreasing and cost increasing moves experienced by the algorithm.

The progression of the cooling process, and simultaneously the annealing, can be divided into three segments, viz., global optimum locating (jumping over mountains), descending the mountain (and jumping over smaller hills), and local (mostly greedy) search. A cooling strategy should reflect these segments in the correct sequence.

At the onset, any rapid decrement of the temperature should be discouraged to avoid any 'quenching' effect. In the third (last) stage of temperature decrement, the algorithm is rarely expected to jump over hills and is assumed to be already in the region of the global solution. During this stage (last third of the Markov chain), the temperature value and decrement rates should be maintained at lower values to result in a flat convergence pattern. During the middle part of annealing, the algorithm should perform most of the necessary decrements in temperature. While annealing, the value of the cost function is assumed to follow a Gaussian pattern especially at higher temperatures. The above three tier cooling can be incorporated into one Gaussian-like temperature decrement over the entire annealing process. The following formula is used to compute the annealing temperature t_k during a Markov chain k:

$$t_k = t_0 \cdot a^{-\left[\frac{k}{f \cdot k\max}\right]^b} \tag{5}$$

where a and f are the control parameters and $kmax$ is the maximum number of chains to be executed. The exponent b can be computed once a and f are selected and the final temperature t_f (some small number) is set. At the final temperature decrement step (last Markov Chain), $t_k = t_f$ and $k = k\max$. Equations (5.59) and (5.55) yield,

$$t_f = t_0 \cdot a^{-\left(\frac{1}{f}\right)^b} \tag{6}$$

An interesting feature of the decrement rule, Eqn. (5), is that it can be tailored to go through any given temperature t_k ($t_o > t_k > t_f$) during a given Markov chain k. The difficulty lies in the selection of the values for the parameters a and f. When the algorithm is in the kth

Markov chain and the parameter f equals ($k/kmax$) the corresponding temperature is given by $t_K \frac{t_0}{a}$. This indicates that the temperature attained in the kth chain is equal to the $\left(\frac{1}{a}\right)$th fraction of the initial temperature t_0. For a typical schedule, the temperature will be reduced to half of the initial temperature at about one-third the maximum number of allowed Markov chains, i.e., $a = 2$ and $f = 1/3$. This will allow 2/3 of the time to be devoted to finding the optimum in the current region.

Using a predetermined small number for the final temperature with a parametric form of the decrement rule, an upper limit is chosen for the number of iteration. As such, the algorithm is terminated if any of the following criteria are met in the order listed below:

(i) The final cost value in five consecutive Markov chains did not improve.

(ii) Similar to above. Five consecutive Markov chains did not improve the average cost $\overline{C}$ beyond a specified small fraction ε i.e.,

$$\frac{\overline{C}_{k-1} - \overline{C}_k}{\overline{C}_{k-1}} < \varepsilon \tag{7}$$

Here, the algorithm is assumed to have arrived at/very close to the optimum or have converged and, hence, it is terminated. The value of ε is set from past experience based on the cost values, scale factors, the accuracy desired and the computational effort involved.

(iii) The algorithm did not terminate in *kmax* Markov chains. At this point, the temperature reaches a value of t_f.

If proper stopping criteria are used, it is unusual to have the algorithm stopped by this method, indicating insufficient annealing for the given problem situation.

5 Numerical Examples

The effects of the tune-ups are demonstrated by the following examples.

5.1 Welded Beam Design

Welded joints can save time and money, if designed properly. In this example, the total cost of a welded beam is minimized subject to various constraints. There are four dimensions that must be determined such that the cost is minimum while satisfying the constraints, shown in Figure 1. This nonlinear optimization problem was used by many researchers and the formulation is adopted from Reklaitis[9]. The optimization problem is:

Find the 4 dimensions {W,t,l,T}

that minimize the cost $F = 1.1047t^2l + 0.04811WT(14.0 + l)$

subject to $\tau_{weld} \le \tau_d$, $\sigma_{weld} \le \sigma_d$, $t \le T$, $P_c \ge F$,

$t \ge 0.125$, and $\delta_{tip} \le 0.25$,

where, τ_d = allowable shear stress in the weld, σ_d = allowable normal stress in the beam, P_c = buckling load, δ_{tip} = end deflection, and F = 6000 lb.

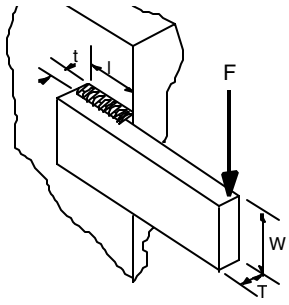

Fig. 1 : Welded beam design parameters and loading.

The derivations of the stress and deflection equations are found in [9]. The values of τ_d ,σ_d and L are assumed to be 13600 psi, 30000 psi and 14 inches respectively. The values of G and E of the material are chosen as 12E6 and 30E6 psi respectively. The last physical constraint requires that all the dimensions be non-negative, i.e. $t, l, W, T \geq 0$. This problem is solved using the regular as well as tuned SA algorithm implemented on a Sun SparcStation. The results are shown in Table 1 along with those reported by others using

Table 1: Results of Welded Beam Optimization.

Variable	Regul. SA	Tuned-SA	[10]	[11] Soln.1	[11] Soln.2
t	.1525	0.2471	0.2536	0.2918	0.2489
l	11.64	6.1451	7.141	5.2141	6.173
W	8.5576	8.2721	7.1044	7.8446	8.1789
T	0.2489	0.2495	.2536	0.2918	.2533
Constraints violated	none	none	3	1	none
Objective Value	2.9265	2.4148	2.3398	2.606	2.4331

10,11 See the reference section for the sources.

different solution methods. It can be seen that the solution by the Tuned SA is superior than the one found by the regular SA algorithm. Moreover, the present solution compared favorably with others [9] in the table except those reported in [10, 11] which violated 3 and 1 constraints respectively. Considering that the SA is inherently a discrete method, the present results demonstrate the utility of the tuned algorithm as a robust optimizer.

5.2 Two-Variable Multimodal Function Optimization

The minimization of a two dimensional function that has fifteen stationary points is considered. This function, a form of the well-known *Camelback* function, is given by Equation 8.

$$f(x, y) = 4x^2 - 2.1x^4 + \frac{x^6}{3} + xy + 4y^4 - 4y^2 \tag{8}$$

There are 15 stationary points of this function with global optimums at (0.0898, -0.7127) and (-0.0898, 0.7127), and f = -1.0316. The starting point was x= -2, y= -1, f=5.7333 (same for

implementation with and without the reheat) and the final solutions. The 'reheat tuned algorithm found one of the two global minima whereas the simple algorithm got trapped near one of the local minima on its search path. Both algorithms ran for a total of 500 iterations. The simple SA found the solution x*= -1.73, y*= -0.18, f*=2.29. While the 'reheat' tuned SA found the solution x*= -0.087, y*=0.72, f*= -1.03. The 'reheat' strategy was initiated at the midpoint (at 251^{st} iteration) where the tuned algorithm temporarily accepted worse designs and got out of the local minimum finding better solution. In computationally intensive applications such as structural design optimization, the *reheat* strategy may prove to be an acceptable compromise between a local solution requiring limited computation and the global solution that might require a prohibitively large amount of computation.

5.3 Optimal Layout of 16 Circular Parts

Flat, thin components are cut out or stamped for consumer or industrial products from sheet metal, fabric, plastic, or other sheet stock. In most cases, like automobile body components, various size parts are cut out from rolls or plates. This example deals with the relative positioning of the parts to be cutout such that they are packed as close as possible and waste is minimized. Thus the objective is minimizing the rectangular area from which the parts are cut. The parts are assumed to be circular with constraints related to inclusion and intersection/overlap detection, as shown in Figure 2. The overlap constraints state that the center distance between any pair of circular parts must be greater than or equal to the sum of the radii of the respective parts. For a pair of circular parts 1 and 2, overlap is given by: $\left(\sqrt{(x2-x1)^2+(y2-y1)^2}-(r1+r2)\right)$. Position and radii are given by (*xi,*yi) and *ri* respectively. As long as this is positive, there is no overlap. A total of $n(n+1)/2$ overlap constraints must be checked for satisfaction. Furthermore, another set of constraints specifies that the parts should be packed as close as possible to the axes but should not overlap. A set of 16 circles was randomly placed between the positions $0 \le x_i, y_i \le 25$ in., initially with 13 overlaps and an objective function value of 661.03 in^2. The optimization problem is solved using the tuned SA algorithm. In the final position of the parts generated by the SA, there were no overlap constraint violations and the pack was within 0.01 units from the axes. The objective was reduced to 584.46 in^2.

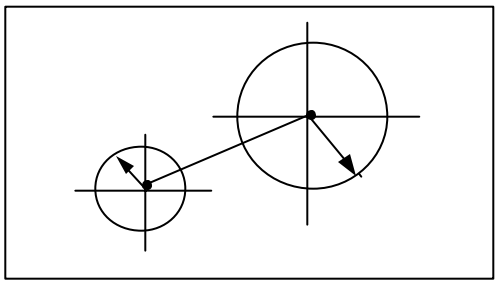

Fig. 2. Overlap calculation between 2 circles.

5.4 Optimal Layout of 100 Circular Parts

The circular part compaction is highly nonlinear and multimodal. Problems with large number of parts is becoming popular as test problems [12]. This example deals with 100 circular parts. It has more than 12 times as many design variables, the constraints increased from 136 to 5050, thus increasing the computational complexity of the problem. The numerical results are summarized in Table 2.

Table 2 : Initial and final data for the 100 part layout problem.

Items	Initial Design	Final Design
Constraints violated	44	1
Amount of violation	15899.75	.002199
Objective value	273148.9	73898.87
% waste	82.98%	37.1%

In this example, the design objective improved quickly during the early iterations but gradually tapered off in the latter iterations. This is an indication that the algorithm approached the vicinity of the optimum quickly but its characteristic slow convergence takes a relatively many more iterations before the convergence criteria are satisfied. The constraint violation of .002199 in the final design indicates existence of active constraint.

5.5 Optimal Design of a Space Truss

This example involves a 3-D space truss with 25 members grouped into 8 sets of sizes. Details of the formulation are available in many literature [7]. Dhingra and Bennage [13] used it to demonstrate an implementation of Tabu Search method. The objective is to minimize weight subject to buckling constraints under two loading conditions. Each set of truss members could assume any of the 30 allowable sizes between 0.1 to 2.6 in^2, thus allowing a total of 30^8 possible configurations. Loads varied from –10,000 lbs to 10,000 lbs. The weight was minimized from 969.01 lbs to 481.33 lbs using the tuned SA algorithm which was slightly better than that reported in [13]. The simple cooling schedule as described earlier, was compared with several prominent schedules published in the literature through solution of a 8-circle part placement problem (16 variable, each may assume any of 45 values) and this 25-bar space truss optimization problem (8 variable, 30 allowable sizes) . In the first problem, the simple algorithm performed better than all others in terms of optimum found. In solving the second problem, the SA using simple cooling schedule came out second. The authors will publish the details of the tests and their results soon.

6 Conclusion

The simulated annealing (SA), a discrete stochastic optimization method is tuned up with several augmentations. One of these augmentations enabled handling of design constraints without lumping them together with the objective function and/or scaling. The convergence of the process was also positively influenced by the approach because of superior infeasibility reduction. This is particularly critical in problems where no solution

exists that could be used as a starting point, as required by many algorithms. Furthermore, a simple cooling algorithm, which is both simpler and equally effective as other complex schedules, was implemented. The design of a welded beam and part layout optimization problems are solved using the tuned up SA, which yielded encouraging results. The 100 part layout optimization problem is taken as a representative, computationally large, optimization problem. This work showed that implementing several simple algorithmic developments could add robustness to the annealing algorithm for optimization.

References

1. Kirkpatrick, S., Gelatt Jr., C.D. and Vecchi, M.P., "Optimization by Simulated Annealing", *Science*, Vol. 220, pp. 671-680,1983.
2. Atiqullah, Mir, S. S. Rao, "Simulated Annealing and Parallel Processing: An Implementation for Constrained Global Design Optimization." *Engineering Optimization*, vol. 32, pp. 659-685, 2000.
3. Hajek, B., "Cooling Schedules for Optimal Annealing", Mathematics of Operations Research, Vol. .13, N0.4, pp. 563-571, 1988.
4. Aarts, E.H.L. and van Laarhoven, P.J.M., "Statistical Cooling : A general Approach to Combinatorial Optimization Problems," *Philips Journal of Research*, Vol. 40, pp. 193-226, 1985.
5. Aarts, E.H.L. and van Laarhoven, P.J.M.,"A New Polynomial Time Cooling Schedule," Proc. *IEEE International Conf. on Computer Aided Design*, Santa Clara, pp 206-228, 1985.
6. Huang, M.D., Romeo, F, and Sangiovanni-Vincentelli, A.L, "An Efficient General Cooling Schedule for Simulated Annealing," in *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 381-384, Santa Clara, November 1986.
7. Atiqullah, Mir M., "Global design optimization using Stochastic methods and Parallel processing," *Ph.D. Dissertation*, School of mechanical Engineering, Purdue University, West Lafayette, 1995.
8. Romeo, F. and Sangiovanni-Vincentelli, A.L., "Probabilistic Hill Climbing Algorithms : Properties and Applications," *Proceedings of the Chapel Hill Conference and VL SI*, pp. 393-417, May 1985.
9. Reklaitis, G.V., Ravindran, A., Ragsdell, K.M., " Engineering Optimization: Methods and Application," John Wiley and Sons, New York, 1983.
10. Ragsdell, K.M. and Phillips, D.T.; "Optimal Design of a Class of Welded Structures using Geometric Programming", ASME Journal of Engineering for Industry, Vol. 98, No 3, pp. 1021-1025, 1976.
11. Deb Kalyanmoy; "Optimal Design of a Class of Welded Structures via Genetic Algorithms", 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. Long Beach, CA. April 2-4, 1990.
12. S.S. Rao, and E.L. Mulkay, "Engineering Design Optimization Using Interior-Point Algorithms," AIAA Journal, Vol. 38, No. 11, November 2000.
13. W.A. Bennage, and A.K. Dhingra, "Optimization of Truss Topology Using Tabu Search," International Journal for Numerical Methods in Engineering, Vol. 38, pp. 4035-4052, 1995.