Simulated Annealing Applications

Bradley J. Buckham (93-07482), Casey Lambert (99-05232)
Mechanical Engineering Department, University of Victoria

18 November 1999

Abstract The analogy between the global optimization method of simulated annealing and the physical process of annealing metals is briefly reviewed. The parameters and components, which must be defined for any optimization problem using simulated annealing, are identified, and a review of literature regarding application of annealing to scheduling, image correction, IC circuit design, path generation, and mechanism synthesis is presented.

1. Introduction

Simulated Annealing was first introduced in 1983 as an intriguing technique for optimizing functions of many variables [1]. Simulated annealing is a heuristic strategy that provides a means for optimization of *NP* complete problems: those for which an exponentially increasing number of steps is required to generate the/an exact answer. Although such a heuristic (logical) approach can't guarantee to produce the exact optimum, an acceptable optimum can be found in a reasonable time, while keeping the computational expense dependent on low powers of *N*, the dimension of the problem.

Simulated annealing is based on an analogy to the cooling of heated metals. In any heated metal sample the probability of some cluster of atoms at a position, r_i, exhibiting a specific energy state, $E(r_i)$, at some temperature *T*, is defined by the Boltzmann probability factor:

$$P(E(r_i)) = e^{-\left[\frac{E(r_i)}{k_B T}\right]}$$

where k_B is Boltzmann's constant. As a metal is slowly cooled, atoms will fluctuate between relatively higher and lower energy levels and allowed to equilibrate at each temperature *T*. The material will approach a *ground state,* a highly ordered form in which there is very little probability for the existence of a high energy state throughout the material.

If the Energy function of this physical system is replaced by an objective function, $f(\mathbf{X})$, that is dependent on a vector of design variables, $\mathbf{X}$, then the slow progression towards an ordered ground state is representative of a progression to a global optimum. To achieve this, a control parameter *T*, analogous to a temperature, and a constant *C*, analogous to Boltzmann's constant, must be specified for the optimization problem. In standard iterative improvement methods, a series of trial points is generated until an improvement in the objective function is noted in which case the trial point is accepted. However, this process only allows for downhill movements to be made over the domain.

In order to generate the annealing behaviour, a secondary criterion is added to the process. If a trial point generates a large value of the objective function then the probability of accepting this trial point is determined using the Boltzmann probability distribution:

$$P[\text{accept } \mathbf{X}_t] = e^{-\left[\frac{f(\mathbf{X}_t) - f(\mathbf{X}_0)}{CT}\right]}$$

, where $\mathbf{X}_0$ is the initial starting point. This probability is compared against a randomly generated number over the range [0..1]. If $P[\text{accept } \mathbf{X}_t] \geq random[0..1]$ then the trial point is accepted. This dependence on random numbers makes simulated annealing a stochastic method. Various implementations will use various methods of random number generation, an example is the Lehmer generator [2]. Repeating this iterative improvement many times at each value of the control parameter *T*, the methodical thermal rearrangement of atoms within a metal at a temperature *T* is simulated [1]. This iterative improvement forms the *inner loop* of the method. The variation of the control parameter *T* is contained in an *outer loop*. The initial value of the control parameter is suitably high and a methodology for decrementing *T* is applied. Figure 1 provides a flowchart representation of the annealing algorithm.

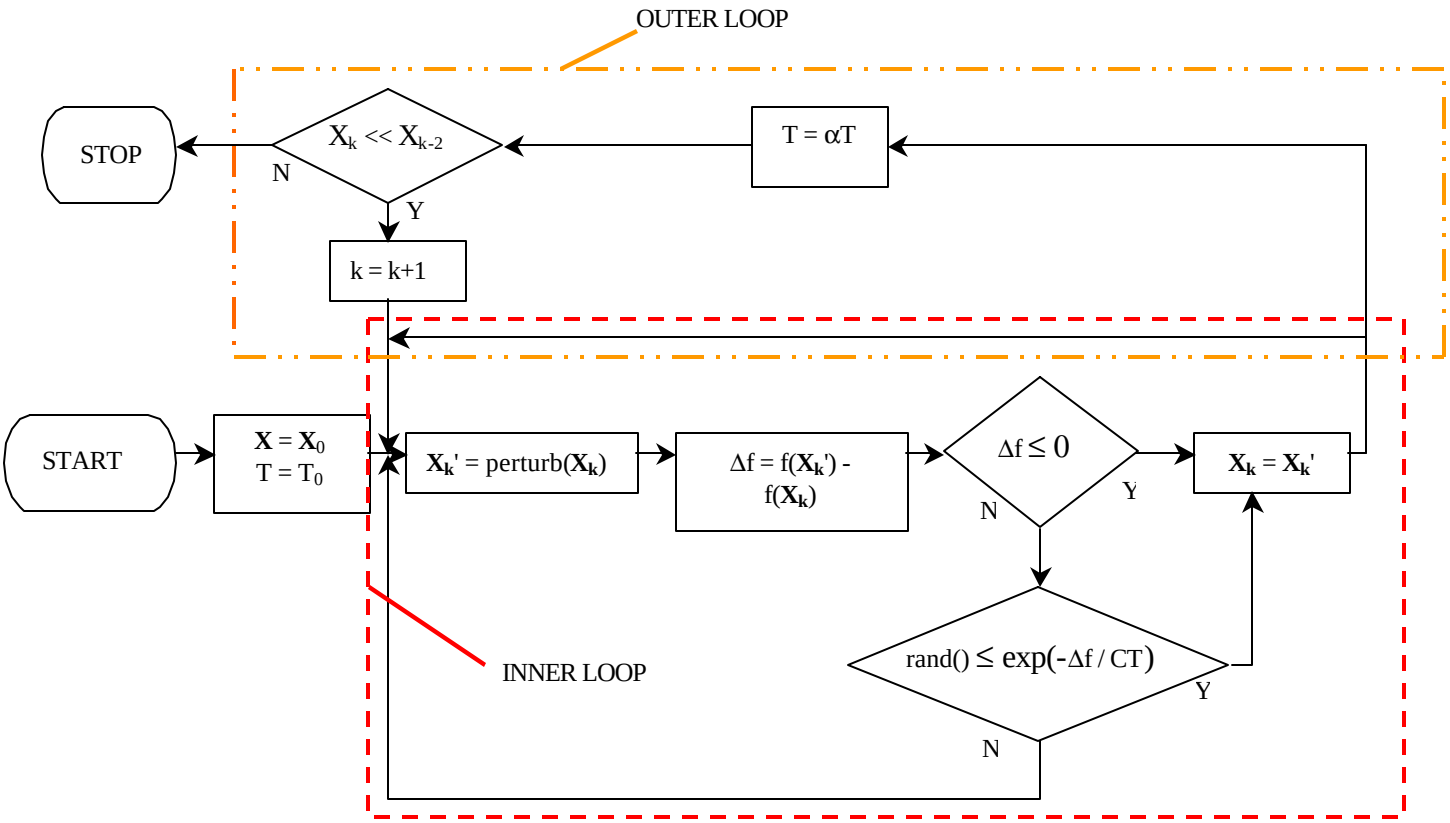


Figure 1. A flowchart representation of the annealing process.

Application of the annealing strategy to any engineering optimization problem requires definition of four major components [1] [2] [3] [4].

1. ***Problem Configuration***: a definition of the suitable domain over which the optimum can be sought. This knowledge is often expressed in the form of constraint equations.
2. ***Neighborhood Configuration***: a method of iteratively perturbing the design vector to create new trial points.
3. ***Objective function***: a scalar equation that weighs all of the design variables to provide a measure of the goodness for each trial point.
4. ***Cooling / Annealing Schedule***: a methodology for specifying the maximum number of inner loop iterations and the manner in which the control parameter will be decremented in each iteration of the outer loop.

Often the most difficult step in the annealing process is the development of an appropriate cooling schedule. To ensure the success of the optimization, the temperature (control parameter) must be controlled so that it is large enough to move off a local minimum, but small enough not to move off a global minimum. Due to the wide variety and complicated nature of most combinational optimization problems, the suitable cooling schedule will be unique for each problem. Ideally, the temperature should be lowered slow enough to ensure that a good minimum is achieved, but also quick enough so that computational time is minimized.

Some practical combinatorial optimization problems which have been addressed using simulated annealing include scheduling, image correction, mechanism synthesis, the physical design of integrated circuitry, and path generation for robotic obstacle avoidance problems. In this paper we will investigate the nuances of simulated annealing by presenting its application to each of these problems.

2. Applications of Simulated Annealing

2.1. Simulated Annealing in a Java Environment (Travelling Salesman Example)

To simplify the complexity associated with applying simulated annealing to certain problems it is suggested in [5] to implement the algorithm in a Java environment. To demonstrate the merits of this approach, the 'Travelling Salesman' problem was solved in a Java environment [5]. The goal of the travelling salesman problem is to minimize the distance traveled by connecting a certain amount of destinations (cities). When simulated annealing was first applied to this type of problem [1] it was able to locate optimal solutions for up to 6000 sites. Previously, the largest problem of that kind for which a proved solution was found contained only 318 sites. The convergence of the annealing algorithm to the global minimum depends on the starting temperature and the cooling rate. If the starting temperature is too low, it is possible that the program would lose it's ability to move out of a global minimum. Obtaining an appropriate cooling schedule is difficult and remains a black art. By displaying the process within an interactive Java environment it is possible to get a feel for the effectiveness of a particular cooling rate.

Figure 2 shows a Java user interface that uses simulated annealing to solve this type of problem. The white area is the travelling region and destinations are specified by 'clicking' on the screen. 'Clicking' in the appropriate bar sets the cooling rate and the initial starting temperature. Both the temperature and the cooling rate can be adjusted at

any point during the optimization procedure. The advantages of using a Java environment over code written in C are:

- much simplified user interface,
- in-situ viewing and tuning of the annealing process
- multi-threaded programming,
- ability to graph and visualize the in-situ annealing data.

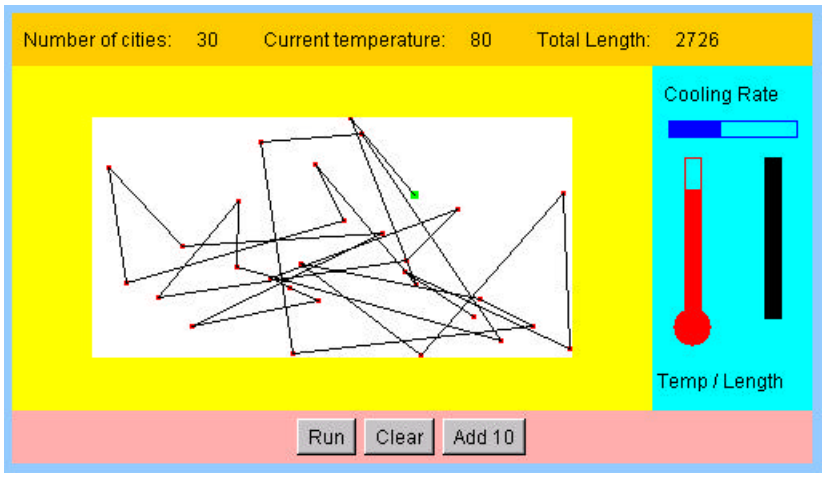


Figure 2. Java Environment for Travelling Salesman Problem

The program was ran with the configuration in Figure 2. 30 sites were randomly chosen and after approximately 3 minutes the program converged when the temperature dropped to zero. This can be seen in Figure 3. It is unknown at this time if the results in Figure 3 are represent the global minimum. However, more confidence in the quality of the solution can be obtained by adjusting the temperature and cooling rate and repeating the process.

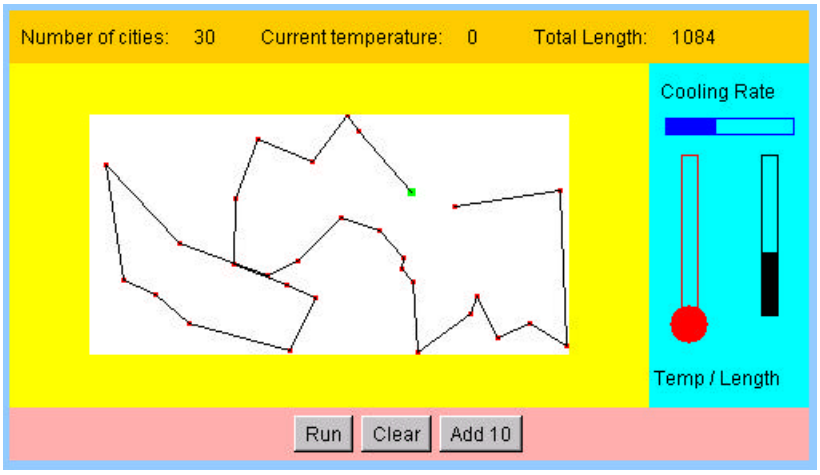


Figure 3. Results for 30 Site Travelling Salesman Problem

2.3. Image Reconstruction

Simulated annealing can be applied to image generation for high resolution computer graphics. In [6] a simulated annealing technique was used for Positron Emission Tomography (PET) image reconstruction from noisy projection data. PET is a method to display metabolic activity in a slice through a patient's body. To address the problem image degredation resulting from noise picked up from the scanner, statistical reconstruction techniques have received much attention recently. The advantage of using the simulated annealing algorithm comes from its ability to avoid getting trapped in local minima.

The basic method consists of iteratively reconstructing an image that best fits the measured data, P^m. At each iteration step, the pseudo data, P^p, that corresponds to the present state of the image will be calculated. The measured data and the pseudo data will be compared and if the difference between the measured data and the pseudo data is minimized, the reconstructed image will converge towards the original image. This process is represented graphically in Figure 4.

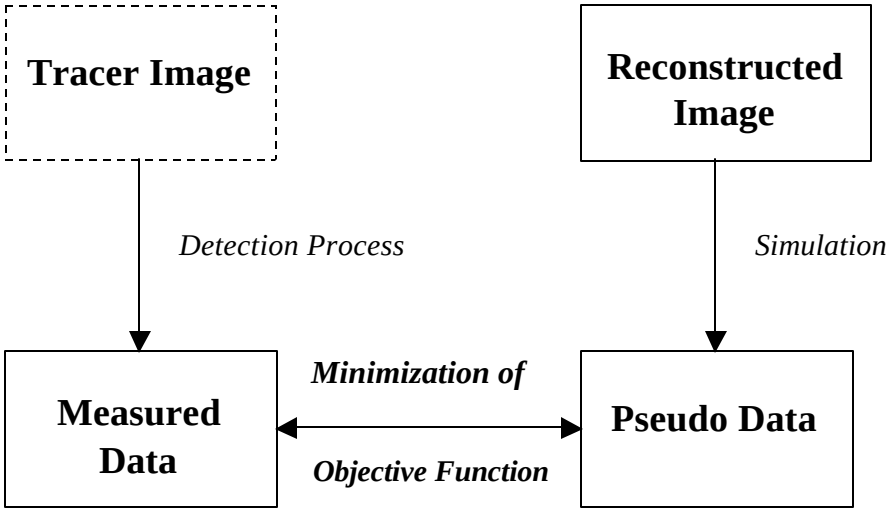


Figure 4. Image Reconstruction Scheme For Simulated Annealing.

The objective or cost function is simply the difference between the two data sets squared:

$$E = \sum_{i,j} (P_{ij}^m - P_{ij}^p)^2$$

where *i* and *j* are the references for individual pixels. At each iteration step the intensity of one randomly chosen pixel is altered. The change in the objection function is calculated and a decision is made as whether to accept the change or not. The reconstructed image eventually converges to the original image.

To determine a cooling schedule, an initial temperature and the rate at which the temperature is cooled must be specified. The starting temperature was chosen so that about 80% of the positive transitions are accepted. Trial and error can be used to determine the appropriate starting temperature, however, the authors in [6] developed a

criterion which calculates this starting temperature performing one iteration. To determine the cooling rate, a trade-off must be made between the temperature decrement between each specific temperature and the number of iterations performed at that temperature. Trial and error is a suitable way to determine the appropriate cooling parameters. The iterative process will be stopped once the objective function reaches a constant equilibrium value for successive stages.

In [6], this process was shown to provide good results for an image area of 64 X 64 pixels. The original image and the reconstructed images can be seen in Figure 5. (A) is the original image and (B) is the image reconstructed using simulated annealing. The blurred image (C) was obtained using the maximum likelihood method, which does not globally optimize the image.

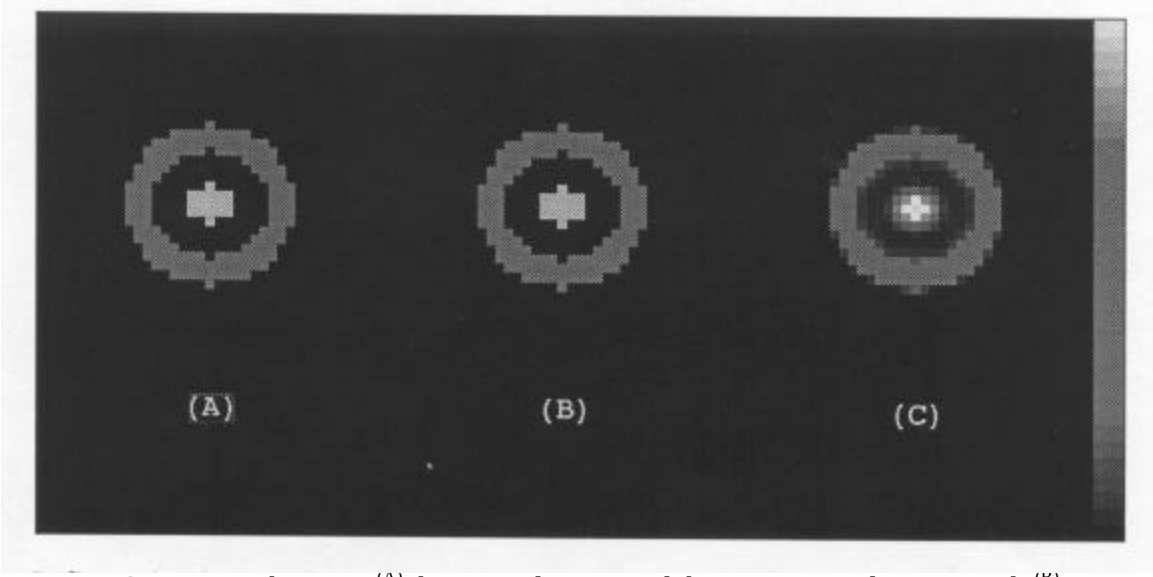


Figure 5. Comparison between (A) the original image and the reconstructed images with (B) simulated annealing and (C) maximum likelihood.

2.2. IC Circuit Design

A common application of the simulated annealing algorithm is the physical design of IC circuitry [1] [3]. There are three stages to this process.

1. Partition the individual gates of a logic diagram between chips.
2. Physically place the circuits which make up each gate within each chip.
3. Route the wires between terminal locations.

The first of these steps provides a good example of the application of the annealing algorithm and will be focussed on in this section. The constraints for this problem are to minimize the number of signals that must cross between chips, and to ensure that there will be room on each chip by attempting to evenly distribute the number of circuits.

These conflicting constraints result in the optimization problem being *frustrated*. This means that no trial point will satisfy both of these constraints simultaneously. The annealing of such a problem is directly analogous to the physical annealing of a family of magnetic alloys called *spin glasses*. These alloys contain competing ferromagnetic and anti-ferromagnetic qualities which result in a large degree of disorder existing even after

a slow annealing to a low energy ground state. Similarly, for a frustrated optimization problem, it has been found that a large number of degenerate ground states will exist, with nearly equal optimality [1]. Since simulated annealing is a stochastic approach, it is highly likely that it will find one of these many ground states.

To partition N circuits between two chips, one first establishes a connectivity matrix $\mathbf{A}_{N\times N}$, and the partition vector $\mathbf{i}_{1\times N}$. The connectivity matrix is an upper triangular matrix in which the integer entry, a_{ij}, is the number of signals being passed between circuits *i* and *j*. Each $\mathbf{i}_i$ entry indicates which chip the i^{th} circuit is located on, through a value of +1 or -1. To formulate the problem configuration, the constraints are enforced using a penalty approach. The number of signals that cross the chip boundaries is computed using:

$$\sum_{i=1}^{N} \frac{a_{ij}}{4}(m_i - m_j)^2 \ , \ i > j$$

To determine the square of the difference between the number of circuits on the two chips:

$$\left(\sum_{i=1}^{N} m_i \right)^2$$

Combining these two factors, one obtains the objective function:

$$f(\mathbf{i}) = \sum_{i=1}^{N} \left(l - \frac{a_{ij}}{2} \right) m_i m_j$$

where l is a weighting factor that describes the relative importance of even circuit distribution and signal crossings. The neighbourhood configuration is defined by randomly selecting one circuit and switching its location between chips. In [1], an example of this type of problem is the partitioning of the logic design of the IBM 370 microprocessor. This involves the division of 5000 primitive logic gates and 200 input signals. Using the criteria just outlined the output of the optimization produces the number of total pins, or terminals, required for the partition. Figure 6 presents the results of this example, including the probability distribution for acceptable circuit partitions at several temperatures achieved in the cooling schedule. It should also be noted that the dark arrow represents the optimum point as produced by local optimization methods.

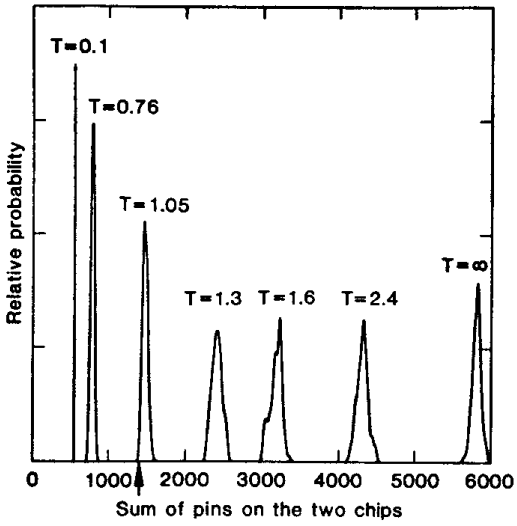


Figure 6. The total number of pins required for partitions accepted at various stages of the cooling cycle. Note that the total number of pins must equal 200 plus double the number of signal crossings. Also note that at the lowest temperature there is virtually one state (optimum point) that is acceptable.

2.3. Path Generation

In [2] a direct form of robotic path planning is presented which uses simulated annealing to form an optimal planar tool path. This direct path planning relies on a distance determination function as the basis of the objective function. In this work, the goal was to use a fixed number of control points to specify a B-spline curve that would meet both the global geometry requirement of obstacle avoidance and the local intrinsic requirement of a smooth continuous curvature. B-splines were chosen because of the relatively intuitive way in which control points affect the shape of a B-spline curve. Also, since any B-spline curve is guaranteed to fit within the polygon defined by the control points, by the convex hull property, the use of B-splines to define an obstacle free path allows for easy quick determination of whether interference exists.

A B-spline curve is defined by the parameter *u*. The planar co-ordinates of any point on the curve are defined by a summation of weighted Bernstein basis functions, *B*.

$$\mathbf{p}(u) = \sum_{i=1}^{n} \mathbf{p}_i B_{i,k}$$

where $\mathbf{p}_i$ is the vector of co-ordinates for the i^{th} control point. It should be noted that the parametric derivatives of the B-spline curve can be easily computed from the above definition. It is also evident that in order to determine an optimum B-spline curve, the only degrees of freedom (design variables) which exist are the co-ordinates of the free control points. An example path-planning problem is presented in Figure 7. Note that only the two interior control points in this case are free since the start and end points are to be interpolated.

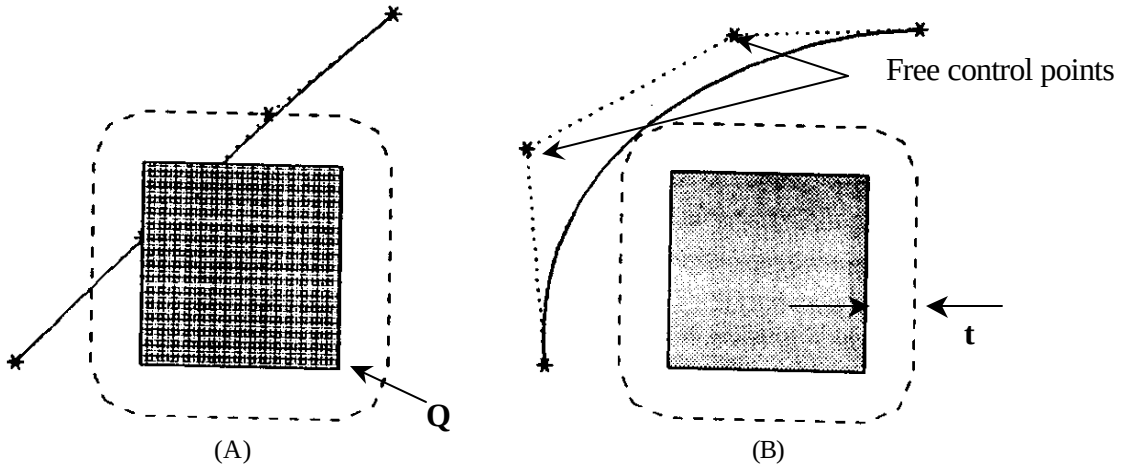


Figure 7. (A) A start and end point are presented for the path. An obstacle defined by the convex polygon Q exists with an imaginary boundary of thickness *t*. (B) The optimization must vary the two interior, free control points to produce the optimal obstacle free path.

Four constraints restrict the planar domain over which a suitable optimum can be found, and thus define the problem configuration. These four *costs* (*C*) which contribute to the objective function are:

1. Obstacles, defined by convex polygons (***Q***) and surrounded by borders of thickness *t* must be avoided within the plane (C_O).
2. The length of the path must be minimized (C_L). In the best case the length of the final path would equal the straight line distance between the two set end points.
3. Keep the paramteric and spatial relationship constant along the path (C_P).
4. Try to keep the curvatures as small as possible throughout the B-spline (C_K).

The first two factors reflect on the global constraints while the last two are the local intrinsic requirements of the final curve. To evaluate the above constraints *N* parametric intervals over the curve are created which act as evaluation points:

$$u_0 = 0$$

$$u_i = u_{i-1} + \frac{1}{N+1},\ i = 1,2,...,N-1$$

Each of these constraints is implemented in an exterior penalty form, creating the objective function:

$$f(\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_n) = C_O + C_L + C_P + C_K$$

where,

$$C_O = \sum_{i=0}^{N-1} C_i \qquad C_i = \begin{cases} K_1 \left| d(\mathbf{p}(u_i), \mathbf{Q}) \right| & \leq 0 \\ K_2 \left(d(\mathbf{p}(u_i), \mathbf{Q}) \right) & 0 \leq d(\mathbf{p}(u_i), \mathbf{Q}) \leq \mathrm{t} \\ 0 & t \leq d(\mathbf{p}(u_i), \mathbf{Q}) \end{cases}$$

$$C_L = K_3 \left(L_c - \left| \mathbf{p}_3 - \mathbf{p}_0 \right| \right),\ L_c = \sum_{i=0}^{N-2} \left| \mathbf{p}(u_{i+1}) - \mathbf{p}(u_i) \right|$$

$$C_P = K_4 \sum_{i=0}^{N-2} \left(\left| \mathbf{p}(u_{i+1}) - \mathbf{p}(u_i) \right| - \frac{L_c}{N-1} \right)$$

$$C_K = K_5 \sum_{i=0}^{N-1} \frac{1}{r_i}$$

and $d(\mathbf{p}(u_i), \mathbf{Q})$ represents the distance calculation between the point $\mathbf{p}(u_i)$ and the surface $\mathbf{Q}$.

The concept of an epsilon ball is used to define the neighbourhood configuration: the allowable perturbations that could be made to the location of each control point for each iteration. This approach uses four random variables to generate changes in each control point. Two variables = ±1 are used to determine the sense of the change in each co-ordinate direction and two random numbers in the range [0..ε] are applied to determine the shift in each direction. The limit on this range is set by the input parameter $e_{\max}$ and the progress of the annealing:

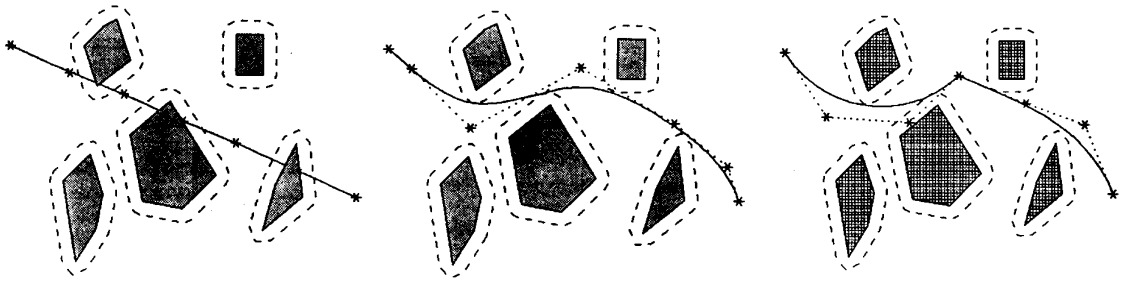
$$e = e_{\max} \left(\frac{\log(T - T_f)}{\log(T_0 - T_f)} \right)$$

Using this limitation on the allowable perturbation size, large steps at low temperatures are avoided. This helps increase the efficiency of the routine near the optimum point. In order to speed up operation at higher temperatures a hybrid approach to the variation of the control parameter can be adopted. Such an approach allows for variation of the *T* value within the inner loop according to:

$$T_j = \left(\frac{f_j}{f_{last}} \right) T_{last} \qquad (1)$$

where j indicates the j^{th} iteration of the inner loop. Using this additional variation of *T*, the need to achieve equilibrium at higher temperatures is eliminated. This can be safely done since the acceptance of trial points at high temperatures is very high.

An example result for generation of a third order spline using seven control points about multiple obstacles is given in Figure 8. Note how the use of B-splines allows for the specification of discontinuities through the *knot vector* $\mathbf{x}$.



(A) (B) (C)

Figure 8. B-spline synthesis through multiple obstacles. (A) The initial configuration. (B) Optimized path for a smooth B-spline. (C) Optimized path with a user-specified discontinuity.

Two significant observations are presented in [2]. Firstly, finer path details emerged at lower temperatures while gross features were evident very early at higher temperatures. This is an example of how the algorithms behaves in a divide and conquer manner [1]. This is also evident in the application of simulated annealing to the design of IC circuitry. Secondly, it was noticed that better initial starting configurations did not result in accelerated convergence. This is because the progress of the annealing algorithm is set by the cooling schedule. Thus, making better initial guesses or picking better trial points can only indirectly increase the efficiency of the routine through implementation of measures such as (1).

2.4. Planar Mechanism Synthesis

A general problem in the planar mechanism design is to create a linkage which propels a tracer point through a desired trajectory [4] [7]. Using classical analytical or graphical methods this is accomplished by specifying certain points through which must lie on the tracer's path. Obviously, more points will be specified when there are more fine-scale features of the path that the designer wants to capture. Unfortunately, as more and more points are prescribed (N>4), the ability of the classical approaches to maintain accuracy between the control points breaks down. In [4] and [7] an application of simulated annealing is used to overcome this problem, and design a crank rocker mechanism to pass a tracer through N prescribed positions.

The objective function that is minimized in this example is the squared error between set analogous locations on the specified curve and the curve traced by the crank rocker mechanism of Figure 9. For example, in the ideal case a rotation of the crank through an input angle q_{2j} would produce a point $\mathbf{p}_j$ on the specified curve and a point $\hat{\mathbf{p}}_j$, on the path formed by the tracer.

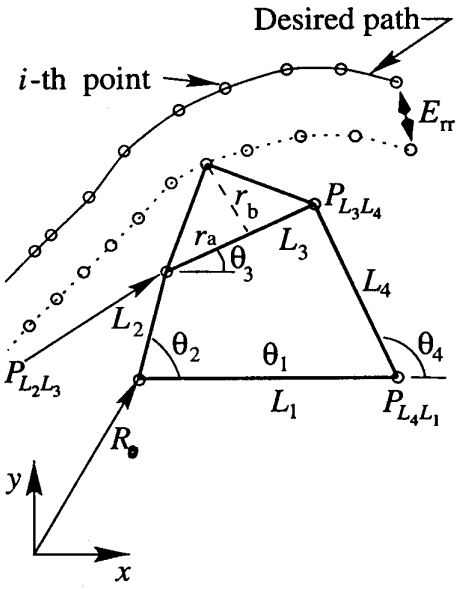


Figure 9. The closed-loop definition of a general, planar, crank rocker mechanism.

Summing the error between the locations of all such pairings yields the objective function:

$$f(\mathbf{p},\hat{\mathbf{p}}) = \sum_{j=1}^{N} \left(\mathbf{p}_j - \hat{\mathbf{p}}_j\right)^2 \qquad (2)$$

By transforming the problem into an optimization problem, it has been noted that the designer gains more control over the overall *goodness* of the overall path while relinquishing control over the interpolation of the original N control points [4].

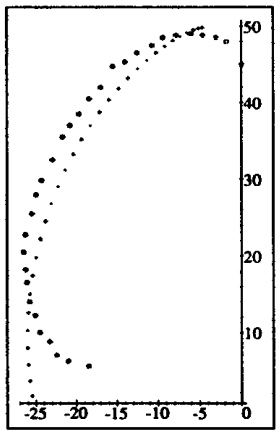
Following the structure of the flowchart of Figure 1, each iteration of the inner loop perturbs the design variables which design the structure of the mechanism, $\mathbf{X} = \{R_x, R_y, L_1, L_2, L_3, L_4, r_a, r_b, q_1\}$ using the concept of the epsilon ball discussed in Section 2.3. For each trial point, the set of q_{2j} values is sequenced, the $\hat{\mathbf{p}}_j$ values evaluated, and the objective function is evaluated.

In [4] a hybridized annealing schedule is also used. In this instance the number of inner loop iterations that is allowed is a dynamic parameter governed by:

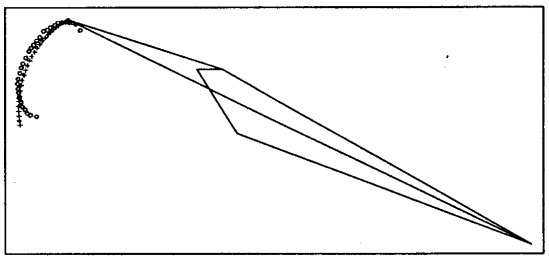
$$N_{in} = N_{dof}\left[2 + 8\left(1 - \frac{\log(T - T_f)}{\log(T_0 - T_f)}\right)\right]$$

Adding this degree of freedom within the annealing algorithm gradually increases the number of accepted trial points within the inner loop. This is done as a result of the observation that at higher temperature equilibrium is attained faster than at the lower final temperatures. Therefore, it isn't desirable to waste function evaluations on needless calculations performed in the first few outer loop cycles.

An example of the output of this application is given in Figure 10, which shows the tracer path for a synthesized crank rocker following a curve specified by 32 points.



(A)



(B)

Figure 10. Desired and Synthesized paths for a N=32 planar path generation problem. [(A)] Close-up of the desired and optimized path. [(B)] Crank-rocker mechanism.

In [7] two observations have been made which help to improve the efficiency of the mechanism synthesis. Firstly, at low temperatures the optimization is *frozen*. This means that in the latter portion of the optimization the routine is not accepting any higher energy states, and is thus acting as a local optimization based on iterative improvement. Thus to improve efficiency, once the annealing algorithm begins to reject a large percentage of the higher energy states the optimization switches to a more efficient local search method such as Powell's Conjugate Direction Search.

It is also presented in [7] that some trial points may not be accepted, even at high temperatures, by the structural objective function but yet will actually be good approximations to the actual optimum design vector. This is because there are two components to the structure of the traced path: a shape and a position, including orientation and location within the plane. By not evaluating the *goodness* of these components individually, an optimization will miss suitable trial points, such as the mechanism configuration producing the trial path shown in Figure 11.

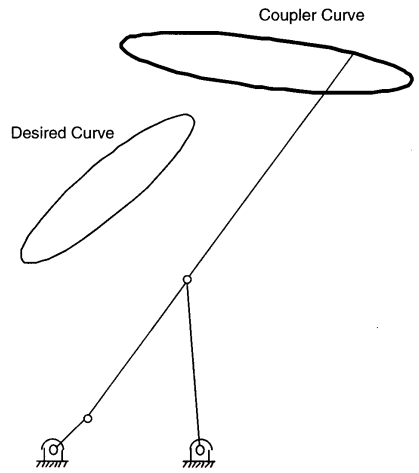


Figure 11. A desired curve and possible solution point that is missed because of the choice of objective function.

To correct this problem the design vector is partitioned into two vectors which describe the shape and position of the tracer path respectively:

$$\mathbf{X}_S = \left\{ L_1/L_2, L_3/L_2, L_4/L_2, r_a/L_2, r_b/L_2 \right\}$$
$$\mathbf{X}_P = \left\{ R_x, R_y, L_2, q_1 \right\}$$

Thus the optimization is broken into two steps. Firstly, $\mathbf{X}_S$ is optimized using a curve definition that contains only shape information. This is accomplished using a Fourier series description of the closed curve. The shape of the tracer curve shown in Figure 12 is given by:

$$f(t) = m_0 + \sum_{k=1}^{\infty} A_k \cos(kt - a_k)$$

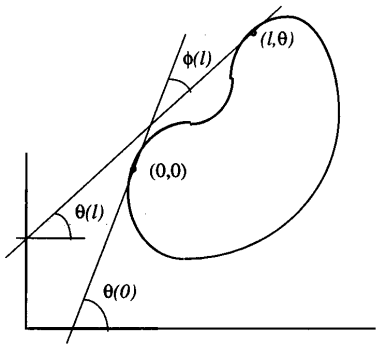


Figure 12. By mapping the change of the parameter $q(l)$ around the closed path, and expressing it relative to the value at some starting point, $q(0)$, a continuous parametric definition of any closed curve is created.

Thus, for any trial point, a polygonal approximation of the tracer path formed by the progression through the N q_2 values is formed and defined by a Fourier series description. Optimization of the shape then occurs using the objective function:

$$f(f,\hat{f}) = \sum_{j=1}^{N} \left(f_j - \hat{f}_j \right)^2$$

Secondly, to position the crank rocker, two pairs of analogous points are selected on the optimized and the desired curves. The distance between these points is minimized using an structural objective, similar in form to (2).

4. Conclusions

Simulated annealing is a heuristic global optimization routine based on an iterative improvement strategy. This method allows for trial points to be accepted which result in a temporary decrease in the *goodness* of the solution. This acceptance is governed by a Boltzmann probability distribution which was originally used to simulate the movement of atoms between energy states in a heated metal. Since the annealing process can avoid being trapped in local optimums, it provides a better alternative for the solution of combinatorial optimization problems. In order to apply the simulated annealing algorithm four components of the problem have to be defined. Firstly, constraints must be recognized and the domain of the problem established. Frequently this is accomplished by including constraints with an external penalty approach. Secondly, a method of perturbing the design variables must be specified. This is generally done in a selectively random manner which reflects the stochastic nature of simulated annealing. Thirdly, an objective function must be defined. Lastly, a cooling schedule is defined which determines the probability of accepting higher cost states throughout an optimization. The ability of simulated annealing to determine a global optimum, even in the occurrence of frustration, makes it a useful tool for image reconstruction, IC circuitry design, mechanism synthesis, path generation, and scheduling.

5. References

[1] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., *Optimization by Simulated Annealing*, Science, May 1983, Vol. 220, no. 4598, pg. 671 - 680.

[2] Malhotra, A., Oliver, J.H., Tu, W., *Synthesis of Spatially and Intrinsically Constrained Curves Using Simulated Annealing*, ASME Journal of Mechanical Design, March 1996, Vol. 118, pg. 53 - 61.

[3] Rutenbar, R.A., *Simulated Annealing Algorithms: An Overview*, IEEE Circuits and Devices, January 1989, pg. 19 - 26.

[4] Martinez-Alfaro, H., Valdez, H., Ortega, J., *Linkage Synthesis of a four bar Mechanism for N Precision Points Using Simulated Annealing*, Proceedings of the 1998 ASME Design Engineering Technical Conference, Sept. 13 - 16 1998. 7 pg.

[5] Niu, X., *Basic Adaptive Simulated Annealing in a Java Environment*, University of Berkeley, 1997, 7 pg.

[6] Sundermann, E., Lemahieu, I., *PET Image Reconstruction Using Simulated Annealing*, Universtiy of Ghent, Bergium, 1996, 9pg.

[7] Ullah, I., Kota, S., *Globally-Optimal Synthesis of Mechanisms for Path Generation Using Simulated Annealing and Powell's Method*, Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering, August 18 - 22 1996, 8 pg.