LBL-35390
UC-405
Preprint

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Physics Division

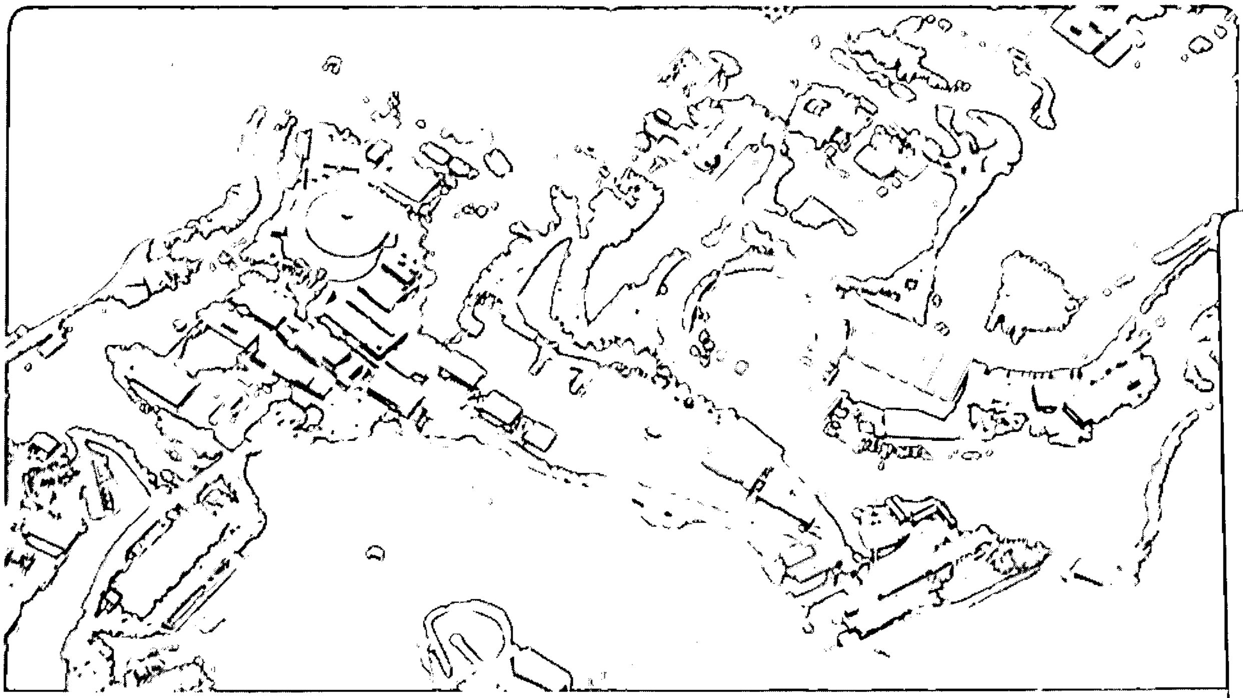
Mathematics Department

To be submitted for publication

Recognizing Knots Using Simulated Annealing

T.J. Ligocki and J.A. Sethian

March 1994



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

LBL-35390
UC-405

Recognizing Knots Using Simulated Annealing

Terry J. Ligocki* and James A. Sethian†

Mathematics Department
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

March 1994

*Supported by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.
†Supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098 and by the National Science Foundation/ARPA under grant DMS-8919074.

Recognizing Knots Using Simulated Annealing

Terry J. Ligocki* James A. Sethian†

March 1994

Abstract

This paper describes the development and implementation of an algorithm which uses simulated annealing to recognize knots by minimizing an energy function defined over all knots. A knot is represented by a piecewise linear curve and the vertices of this curve are perturbed using simulated annealing to minimize the energy. Moving one line segment through another line segment is prohibited. The resulting minimum energy configuration is defined to be the canonical form. The algorithm is then tested with two different types of energy over a collection of complex knots.

1 Introduction

1.1 Overview and Definitions

We have developed and implemented an algorithm which uses simulated annealing to recognize knots by minimizing an energy function defined over all knots. To achieve this, knots are represented by a piecewise linear curve and the simulated annealing algorithm minimizes the energy by perturbing the vertices of this curve. Perturbations that would cause one line segment to pass through another line segment are prohibited. The canonical form of the the knot is defined to be the resulting minimum energy curve. Our tests with two different types of energy and a collection of complex knots show the algorithm is effective but computationally expensive. The ability to untangle and recognize knots and filaments has a wide variety of applications, for example, in polymers, vortex filaments and statistical mechanics, see [4], [5], and also [1]. Our eventual goal is to apply the work described below to such problems.

In this paper, the term "geometric knot" refers to a non-intersecting curve in Euclidean 3-space which is the image of the unit circle under a continuous mapping. Two knots are said to be equivalent if there exists a continuous, one-parameter family of knots transforming one knot into the other knot. Given two geometric knots, in general it is not always possible to find such

*Supported by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098

†Supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098 and by the National Science Foundation/ARPA under grant DMS-8919074

a one-parameter family, thus all knots are not equivalent. For example, the circle and the trefoil knot in Figure 1 are not equivalent. This can be shown using the algebraic techniques discussed in section 1.2. Equivalent geometric knots all represent the same topological knot.

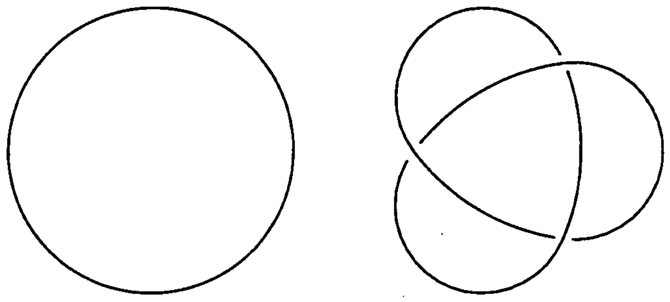


Figure 1: Knots that are not equivalent

Recognizing a knot means transforming all the geometric knots which represent a given topological knot into a single, canonical form. The choice of a canonical form of a topological knot is somewhat arbitrary. While it is desirable to have a canonical form that is easy to recognize, it is difficult to make this idea precise. In this paper, the canonical form of a topological knot is defined to be a geometric knot from this equivalence class which minimizes an energy function. If the energy function is invariant under some set of transformations of the geometric knot (e.g. rigid motions) then the canonical form is defined modulo these transformations.

1.2 Recognizing Knots

The goal of this paper is to devise algorithms for recognizing knots, such as those in Figure 2. The geometric knots in Figure 2a and Figure 2b are in fact equivalent to the geometric knots in Figure 1a and Figure 1b, respectively (i.e. they represent the same topological knot). However, if any of the crossings in Figure 2a or Figure 2b are reversed, a far more complicated topological knot can result.

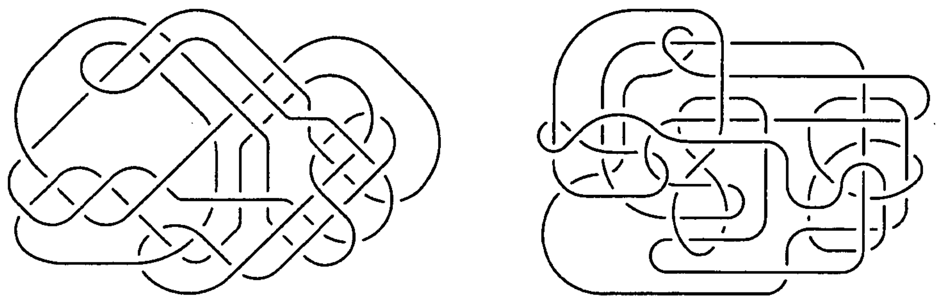


Figure 2: Tangled Knots

No theoretical technique is currently available to determine the topological knot represented by a given geometric knot. A long-standing geometric technique, formalized by Reidemeister [12], provides a set of moves that may be used to transform geometric knots; however, no algorithm currently exists for recognizing knots using these moves.

From a different perspective, algebraic approaches to finding the topological knot represented by a geometric knot include computing a polynomial of one or more variables from a drawing of the geometric knot which has under and over crossings. In this context, knots are usually thought of as a type of link; which is simply a collection of nonintersecting knots. Three such polynomials for a link, K, are the Alexander polynomial, $\Delta_K(t)$ [2], the Jones polynomial, $V_K(t)$ [9], and the HOMFLY polynomial, $H_K(l,m)$ [6]. Much of the discussion here is taken from a paper by Lickorish and Millet [10]. The Alexander and Jones polynomials can be thought of as reductions of the HOMFLY polynomial to one variable

$$\begin{aligned} \Delta_K(t) &= H_K(i, i(t^{1/2} - t^{-1/2})) \\ V_K(t) &= H_K(it^{-1}, i(t^{-1/2} - t^{1/2})) \end{aligned}$$

It should be noted that they did not arise in this way. In these formulas $i = \sqrt{-1}$, but all three polynomials have integer coefficients. All three polynomials are invariants of a link and thus are independent of how it is drawn. These polynomials can be computed from a drawing of the link such as the drawings in Figure 1 and Figure 2. First, the links need to be given an orientation (i.e. a direction along each knot needs to be chosen). Next, pick one crossing in the drawing of the link, e.g. K_+ in Figure 3, and make two new links by reversing the crossing, K_-, and removing the crossing, K_0. The polynomials for the three resulting links satisfy the formulas

$$\begin{array}{ccccccccc} & \Delta_{K_+}(t) & - & & \Delta_{K_-}(t) & + & (t^{1/2} - t^{-1/2}) & \Delta_{K_0}(t) & = 0 \\ t^{-1} & V_{K_+}(t) & - & t & V_{K_-}(t) & + & (t^{-1/2} - t^{1/2}) & V_{K_0}(t) & = 0 \\ l & H_{K_+}(l,m) & + & l^{-1} & H_{K_-}(l,m) & + & m & H_{K_0}(l,m) & = 0 \end{array}$$

These formulas uniquely define the polynomials if one assumes that the polynomial of the unknot is 1. To use these formulas it is necessary to compute the polynomial of the two new links that result. This can be done recursively until a link with a known polynomial is found (e.g. the unknot). Using the formulas for $\Delta_K(t)$, the Alexander polynomial for the trefoil knot in Figure 1 is $t^{-1} - 1 + t$ or $-t^{-1} + 1 - t$ depending on the orientation chosen. Neither is the same as the Alexander polynomial of the unknot so the unknot and the trefoil knot are not equivalent.

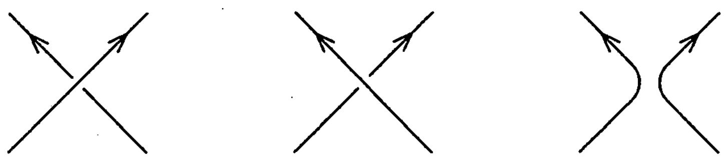
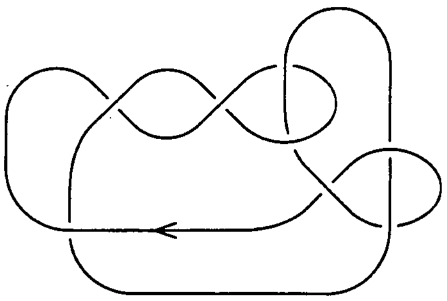

(a) K_+ (b) K_- (c) K_0

Figure 3: Modified intersections

Unfortunately these polynomials do not completely differentiate knots by equivalence class. An example is shown in Figure 4 of two non-equivalent knots that have the same HOMFLY polynomial [10], $H(l,m) = (-l^{-4} - l^{-2} + 2 + l^2) + (l^{-4} + 2l^{-2} - 2 - l^2)m^2 + (1 - l^2)m^4$. They, of course, both have the same Alexander and Jones polynomial. The numbers associated with these knots indicate the number of crossings present in the knot with a subscript which makes the entire designation unique. In the case of the Alexander polynomial, there are knots with $\Delta_K(t) = 1$ that are not equivalent to the unknot, see Figure 5. It is not known if there exists such a knot in the case of the Jones polynomial.


(a) Knot 8_8

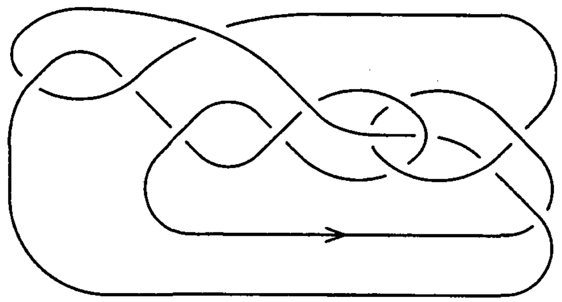

(b) Knot 13_{6714}

Figure 4: Two different knots with the same HOMFLY polynomial

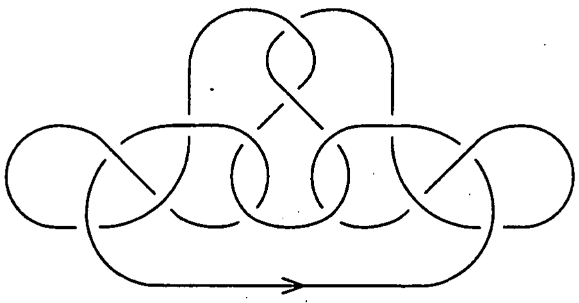


Figure 5: A knot with Alexander polynomial 1

2 Energy Minimization

In this paper, a numerical algorithm is described which categorizes knots by minimizing an energy function defined for all knots. An energy function is a real valued function over the space of all geometric knots. It is assumed that the chosen energy function has a unique global minimum for each set of equivalent geometric knots. A given geometric knot is then transformed into the equivalent geometric knot with minimum energy by using movements that cannot change the topological knot it represents.

2.1 Examples of Energy Functions

Let $\gamma(t)$ be a parametric representation of a geometric knot with parameter t. An example of a physical energy is

$$E^q(\gamma) = \int\int \frac{1}{|\gamma(u) - \gamma(v)|}\, du\, dv$$

This energy represents the electrostatic energy of a knot which has a constant amount of charge distributed evenly along the entire knot. If the energy is changed to

$$\widetilde{E}^q(\gamma) = \int\int \frac{1}{|\gamma(u) - \gamma(v)|}|\dot{\gamma}(u)||\dot{\gamma}(v)|\, du\, dv$$

then the energy represents the electrostatic energy of a knot with a constant charge density. In the case of $E^q(\gamma)$, the energy decreases as the knot grows larger and larger (i.e. the minimum is never achieved), since the unchecked electrostatic repulsion causes the knot to expand while, in the case of $\tilde{E}^q(\gamma)$, the energy decreases as the knot grows smaller and smaller and again the minimum is never achieved. To create an attainable minimum an elastic energy can be added; thus, the expansion (or contraction) of the knot under electrostatic forces is balanced by the increase in elastic energy, yielding a minimal energy which can be achieved. Also, the electrostatic energy, as stated, is not finite. This can be remedied by integrating over a tube containing the knot rather than the knot itself. As in the case of most physical energies this example is invariant under rigid motions. Note, $E^q(\gamma) + \tilde{E}^q(\gamma)$ is also an energy with a minimum energy which can be achieved.

Let $D(\gamma(u), \gamma(v))$ be the minimum distance between $\gamma(u)$ and $\gamma(v)$ along the knot. An example of a geometric energy is

$$E(\gamma) = \int\int \left(\frac{1}{|\gamma(u) - \gamma(v)|^2} - \frac{1}{D(\gamma(u), \gamma(v))^2} \right) |\dot{\gamma}(u)||\dot{\gamma}(v)| \, du \, dv$$

This energy is the result of work by O'Hara [11] which was built on by Freedman [3] where the following results are shown. This energy is invariant under any inversion with respect to a sphere (i.e. a Möbius transformation) as long as no portion of the knot is mapped to infinity. This energy function also has a C^1 minimizer for each prime topological knot.

2.2 Computational Considerations

A class of equivalent geometric knots forms an infinite dimensional space. This makes minimizing an energy function over this class computationally intractable. To discretize the problem, we choose a finite dimensional subspace consisting of the space of all piecewise linear closed curves with a fixed number of segments. If the knot has N segments, N points in Euclidean 3-space completely determine the knot by determining vertices where adjacent segments meet. The energy is minimized over this $3N$ dimensional space.

A variety of methods exist for doing global function minimization over a space of rather large dimension, such as movement by forces, gradient descent techniques, and simulated annealing. Of interest are techniques that are fairly general and which are not tied to a particular minimization problem; at the same time it is important to avoid methods that require "tuning" via a large number of parameters. While movement by forces can obtain a local energy minimum, there is no guarantee that this local minimum is a global minimum, nor does this method stop a knot from crossing itself as it moves (and thus possibly changing into an unequivalent knot). The gradient descent technique is a more general method, however it too only finds a local energy minimum and permits a knot to pass through itself.

Simulated annealing can be applied to a very general class of energy functions (even discontinuous functions). Such techniques do not inherently preserve knot type, however avoiding self-intersection may be easily enforced through an additional restriction described below. While this "fix" can be extended to movement by forces and gradient descent techniques, global minima are not necessarily found and the additional constraints degrade performance. In contrast, a reasonable implementation of simulated annealing asymptotically achieves a global minimum.

3 Simulated Annealing

Simulated annealing is a Monte Carlo technique originally used in the context of physical problems which leads to a notion of a "time" (t) which is a linear scaling of the number of iterations (n) and a "temperature" ($T(t)$) referring to a function of the time which in many cases is a monotone decreasing function. A general simulated annealing algorithm may be described as follows

1. *Choose an initial configuration and make it the current configuration.*
2. *Choose the initial temperature, $T(0) = T_0$ and a minimum temperature, T_{min}.*
3. *Calculate the energy of the current configuration, E_1.*
4. *Perturb the current configuration to produce a new configuration (which may depend on the temperature T).*
5. *Calculate the energy of the new configuration E_2, and the energy change $\Delta E = E_1 - E_2$.*
6. *With a probability equal to $\alpha(T, \Delta E)$, the acceptance function, accept the new configuration and make it the current configuration.*
7. *Increment the number of iterations n (this also increments the time t).*
8. *Calculate the temperature $T(t)$.*
9. *If the temperature is above T_{min}, return to step 3.*

Convergence of this algorithm to a global energy minimum depends on such factors as the acceptance function, the form of the energy, the minimum temperature, etc. The details of an implementation of this algorithm which minimizes an energy function over a class of equivalent geometric knots is described below. Sufficient conditions for asymptotic convergence are given in this context.

3.1 Continuous States

In this problem the set of configurations varies over a continuous space, i.e. Euclidean $3N$ space. In this case, asymptotic convergence of simulated annealing to a global minimum is guaranteed under the following conditions [7]

- The time is linearly proportional to the number of iterations (i.e. $t(n) = c_t n$).
- The current configuration is perturbed by a random amount chosen from a $3N$ dimensional normal distribution where the variance σ^2 of the distribution is linearly proportional to the temperature (i.e. $\sigma^2 = c_\sigma T$).
- The acceptance function has the form: $\alpha(T, \Delta E) = \frac{1}{1+e^{\Delta E/T}}$.
- If T_0 is the initial temperature, the temperature has the form $T(t) = \frac{T_0}{log(e+t)}$.

This is sometimes called Boltzmann simulated annealing [8]. Unfortunately, convergence requires $T \to 0$ and this happens very slowly. For practical reasons discussed below, this function was amended in the actual computations; the amended function has an additional parameter. The computations clearly show that the amended function does not guarantee convergence to a global minimum for a given choice of this additional parameter. On the other hand, in most cases the computations came close to a global minimum for some choice of this additional parameter. In all cases, the algorithm converges to a local minimum.

3.2 Energy computation

Central to the optimization problem is the computation of the energy of a knot. Given a description of the knot as a piecewise linear closed curve, this is relatively straightforward computation. The only significant issue is the efficient evaluation of the energy. In our computations, two types of energy were tried. The first energy is a discrete energy analogous to a combination of several physical energies. This is not an approximation of a finite energy integral over all geometric knots. The second energy is a discrete approximation of the geometric energy discussed in section 2.1.

3.2.1 Physical Energy

A discrete analogue of electrostatic energy can be obtained by placing point charges at the vertices of a piecewise linear knot. The charge of the point charges is kept fixed during each computation. An elastic energy can be added by viewing each segment as a spring with a fixed rest length and Hooke constant. Finally, each pair of adjacent segments can be modeled as having some energy due to "folding" that is elastic in nature and becomes infinite as the angle between the segments goes to zero. Let q_i be the charge at vertex i, $r_{i,j}$ be the Euclidean distance between vertex i and j, l_i be the rest length of segment i, and α_i be the angle between the two segments that meet at vertex i. The energies described above are then given by

1. Electrostatic energy (between vertex i and j):

$$E^q_{i,j} = \frac{q_i q_j}{r_{i,j}}$$

2. Elastic energy (for the segment between vertex i and $i+1$):

$$E^k_i = \frac{1}{2}\, k\, (r_{i,i+1} - l_i)^2$$

3. Folding energy (between segments meeting a vertex i):

$$E^f_i = f\, cot^2(\frac{1}{2}\alpha_i)$$

Thus, the total energy of a piecewise linear knot, γ, is given by

$$E(\gamma) = \sum_i \sum_{j \neq i} E^q_{i,j} + \sum_i E^k_i + \sum_i E^f_i$$

There are several things to note about this energy function. First, the energies are linear and once defined locally may be summed easily to get the total energy. Second, computing the electrostatic energy is $O(N^2)$ while computing the elastic energies is $O(N)$. Third, if only one vertex is moved, recomputing the electrostatic energy is $O(N)$ and recomputing the elastic energies is $O(1)$.

For the initial knot, an $O(N^2)$ energy computation was used. As the knot was perturbed, only one vertex was moved at a time (see section 3.3). Thus, an incremental energy computation was used. For the electrostatic energy, the energy contributed by the point charge at its current position was subtracted from the total energy (requiring $O(N)$ operations). Then the energy contributed by the point charge at its new position was added to the total energy (requiring $O(N)$ operations). For the elastic energy only two terms needed to be corrected (the segments adjacent to the vertex) and for the folding energy only three terms needed to be corrected (since the two segments that move change three angles). As a result, the new energy computation was only $O(N)$ per iteration, not $O(N^2)$.

In many cases, this difference in computation time was substantial, and differentiated between computations that were reasonable to perform and computations that were prohibitively expensive. At the end of the computation, the total energy for the final knot was computed directly using an $O(N^2)$ energy computation to check the incremental energy computation. In all cases the energy computed incremently and directly agreed to four or five decimal places (i.e. no significant incremental error seemed to be introduced).

3.2.2 Geometric Energy

One way to discretize the geometric energy integral is to sample it pointwise at the midpoints of segments of the knot. This approximation converges to the integral as the number of segments used in the approximations is increased. The details of the computation of this approximate energy are not discussed here because this approximation resulted in spurious results. These are discussed in section 4.2.

To overcome this problem, the integral can be approximated more accurately by dividing it into pieces. Assume the knot γ is parameterized so that when $i \leq u < i+1$ the point $\gamma(u)$ lies on segment i and is $u-i$ of the way from vertex i to vertex $i+1$. Then the integral can be written

$$E(\gamma) = \sum_i \int_i^{i+1} \left(\sum_j \int_j^{j+1} \left(\frac{1}{|\gamma(u)-\gamma(v)|^2} - \frac{1}{D(\gamma(u),\gamma(v))^2} \right) |\dot\gamma(u)||\dot\gamma(v)|\, du \right) dv$$

The double integral inside the summation represents the geometric energy contributed by a pair of segments. This is then summed over all possible pairs of segments. Note that the geometric energy of a segment with itself is zero so this need not be calculated. More importantly, the geometric energy of two adjacent segments which meet at an angle not equal to π is infinite. Thus, adjacent segments cannot be included in the summation. With these restrictions, the geometric energy can approximated but the evaluation of the energy requires $O(N^2)$ integrations.

An alternative formulation results when the inner sum is rewritten as a single integral

$$E(\gamma) = \sum_i \int_i^{i+1} \int_{i+2}^{i+N-1} \left(\frac{1}{|\gamma(u)-\gamma(v)|^2} - \frac{1}{D(\gamma(u),\gamma(v))^2} \right) |\dot\gamma(u)||\dot\gamma(v)|\, du\, dv$$

Note that the inner integral avoids the adjacent segments so the energy will be finite. The evaluation of the energy in this form requires only $O(N)$ integrations. This was the method used to approximate the geometric energy in our computations.

3.3 Configuration Generation

The initial knot used in the computations is user specified, and supports discrete geometric knots of any generality (any topological type and complexity). Given a geometric knot (i.e. the current configuration), the knot is perturbed by choosing a vertex at random based on a uniform probability distribution, and moving that vertex by a vector chosen at random based on a Gaussian probability distribution in three dimensions whose variance is proportional to the temperature. If this movement changes the topological type of the knot, the move is rejected and another vector is chosen. This continues until an acceptable movement is found.

A simple technique is used to determine if moving a vertex to a new position changes the topological type of the knot. If, as the vertex moves from its initial position to its new position, the two segments connected to that vertex never cross the rest of the knot, then the knot cannot have changed topological type. In order to implement this, first consider each moved segment individually. As a segment moves from its initial position to its new position, it sweeps out a triangle in three dimensions. If no other nonadjacent segment of the knot intersects this triangle then the original segment does not cross the knot as it moves. If neither adjacent segment crosses the knot when moving the vertex the knot does not change its topological type. Thus, when a vertex is moved, $N-2$ segments need to be checked to see if they intersect either one of two triangles swept out by the two segments adjacent to this vertex.

To determine if a line segment intersects a triangle we exploit several techniques commonly used in computer graphics. The guiding principle is to perform a small amount of computation at the beginning to exclude most cases, since most line segments do not intersect most triangles in the computations. Given a non-degenerate triangle, first compute the plane of the triangle in the implicit form $f(x,y,z) = ax + by + cz + d = 0$. $f(x,y,z)$ is then evaluated at the end points of the line segment. If $f(x,y,z)$ is greater than zero or less than zero at both endpoints, then both endpoints are on the same side of the plane containing the triangle and the line segment cannot intersect the triangle. If each endpoint is on a different side of the plane, pick one of the endpoints and look at the triangular cone formed by this endpoint (as the vertex of the cone) and the corners of the triangle. If the other endpoint is contained inside this cone then the line segment intersects the triangle. If one endpoint lies in the plane of the triangle, pick the other endpoint as the vertex of the triangular cone. If both endpoints are in the plane of the triangle, check to see if the line segment intersects any of the sides or vertices of the triangle. If the triangle is degenerate (i.e. the vertices are collinear), then check to see if the line segment and the triangle are coplanar. If they aren't coplanar, then there is no intersection. If they are coplanar then the above non-degenerate coplanar check is used.

3.4 Cooling

As mentioned above, the temperature is determined by a technique that amends the formula

$$T(t) = \frac{T_0}{log(e+t)}$$

As it stands, this standard form produces a temperature which changes extremely slowly once the temperature has dropped a few orders of magnitude. Beyond that time, it will not drop much lower during our computations. To avoid this, the formula is modified as follows: Pick an initial temperature and a time t_r. Set the temperature according to the above formula for $t < t_r$. When $t = t_r$ chose a "new initial" temperature T_1 to be

$$T_1 = \frac{T_0}{log(e + t_r)}$$

(i.e. the temperature at time t_r). Now, for $t_r \leq t < 2t_r$ set the temperature to

$$T(t) = \frac{T_1}{log(e + (t - t_r))}$$

Thus, in general, if $kt_r \leq t < (k+1)t_r$, let

$$T(t) = \frac{T_k}{log(e + (t - kt_r))}$$

where

$$T_k = \frac{T_{k-1}}{log(e + kt_r)}$$

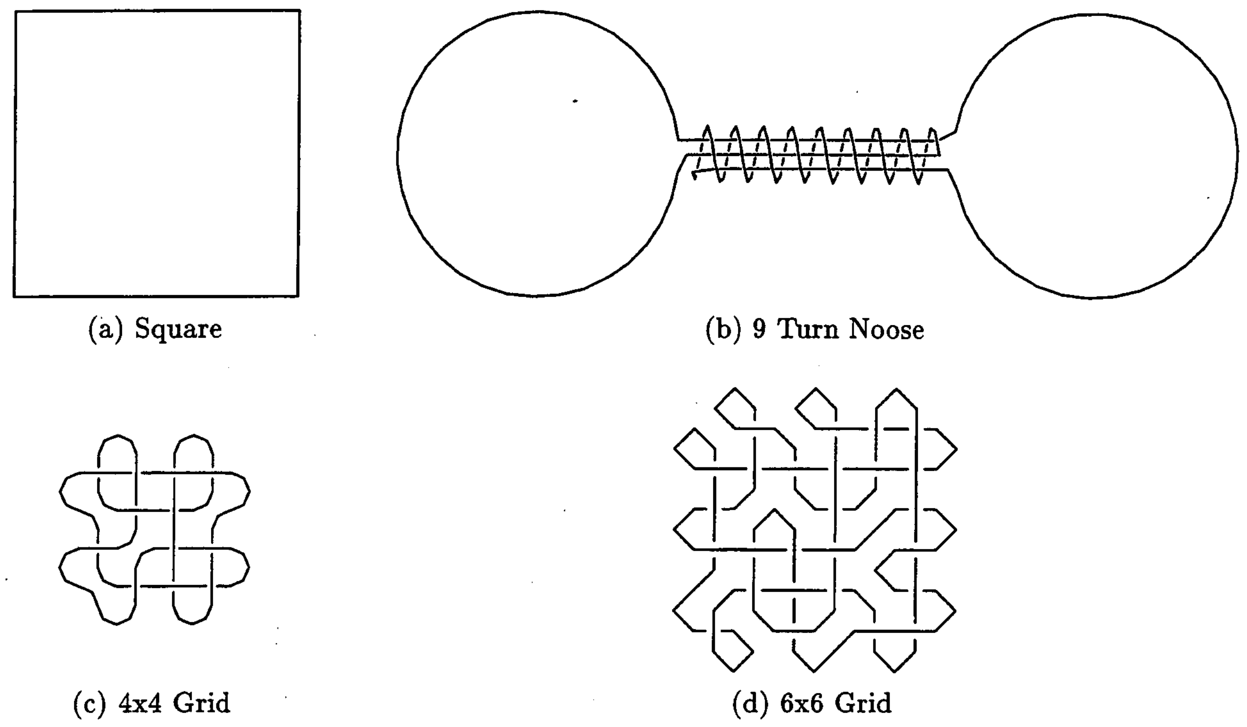
In effect, the algorithm is being reinitialized after a given amount of time. This technique introduces a new parameter t_r, which is the amount of time to wait before reinitializing the algorithm. If this parameter is too small, the evolving configuration remains far from a global minimum because the system "cools" too fast. If this parameter is too large then the algorithm never "cools" enough and the final configuration is far from a global minimum.

Finally, the algorithm is stopped when a given minimum temperature are reached. There are many other techniques for cooling and stopping a simulated annealing algorithm. The above techniques were found to be the best for these computations because they are straightforward to implement, have only a few parameters, and allow the algorithm to produce a result near a global minimum.

4 Results

4.1 Physical Energy

Most of the computations we performed used the discrete physical energy described in section 3.2.1. This energy is easy to compute and can be incrementally updated when one vertex is moved. Unfortunately, this energy does not converge as the number of segments in the knot is increased. Thus, in our computations the number of segments (and their rest length) was kept fixed so the final results would be comparable. We tested the algorithm on a variety of complex unknots, complex trefoil knots, and a few other simple toroidal knots.


(a) Square (b) 9 Turn Noose

(c) 4x4 Grid (d) 6x6 Grid

Figure 6: Initial Unknots

All the knots had 160 segments, a charge of 0.625 per vertex, and each segment had a rest length of 0.625. The spring constant k was 1 and the folding energy constant f was 10.

Three varieties of unknots were tried. The first unknot was simply a square divided into 160 segments. This is shown in Figure 6a. The second set of unknots were in the form of "nooses". A "9 turn" noose is shown in Figure 6b; there are two large loops and the rest of the knot is wrapped around the part of the curve connecting the two loops. In order to put this unknot in canonical form (i.e. a circle) one large loop needs to pass through the center section of the knot. Nooses with 3, 5, and 7 turns were also tried. The final set of unknots were generated on a grid of over and under crossings and are shown in Figure 6c-d.

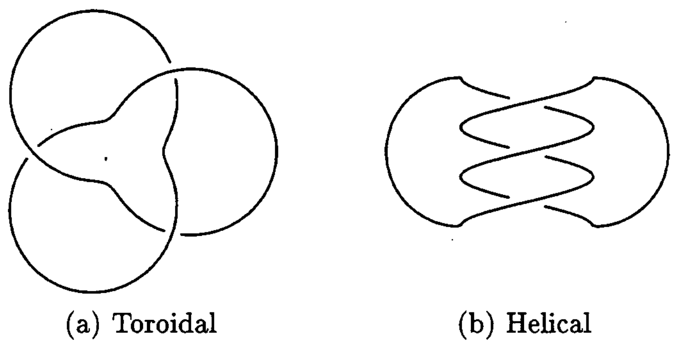

(a) Toroidal (b) Helical

Figure 7: Initial Simple Trefoil Knots

Three types of trefoil knots were tried. The first type is shown in Figure 7a and it is simply the trefoil knot realized on a torus. The second type is shown in Figure 7b and constructed from a double helix in the center and two large loops. The final type is shown in Figure 8. The example shown has each "lobe" of the trefoil knot tied in 4 "half knots". Examples were tried with 0 through 4 "half knots" per lobe.

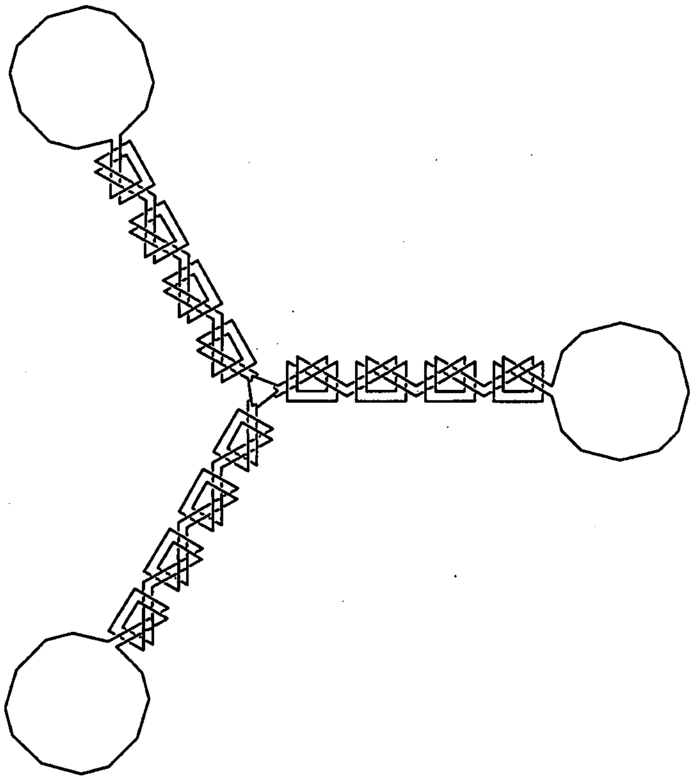


Figure 8: Initial Trefoil Knot With 4 "Half Knots" Per Lobe

Finally, several types of torus knots were tried. They are shown in Figure 9 and include two instances of a 2,5 torus knot, two instances of a 2,7 torus knot, and one 3,5 torus knot.

For the computations the restart time t_r was chosen so the cooling restarted, as described in section 3.4, when the ratio of T_k to T_{k+1} was equal to $\sqrt{10}$. This implies that $t_r = e^{\sqrt{10}} - e \approx 20.91$. The constant c_t was chosen so 16 million iterations took place during this time. The initial temperature T_0 was 100.0 and the minimum temperature T_{min} was 0.001. Thus, 160 million iterations were performed in each case to compute the minimum energy configuration. The constant c_σ was 0.5 in all cases.

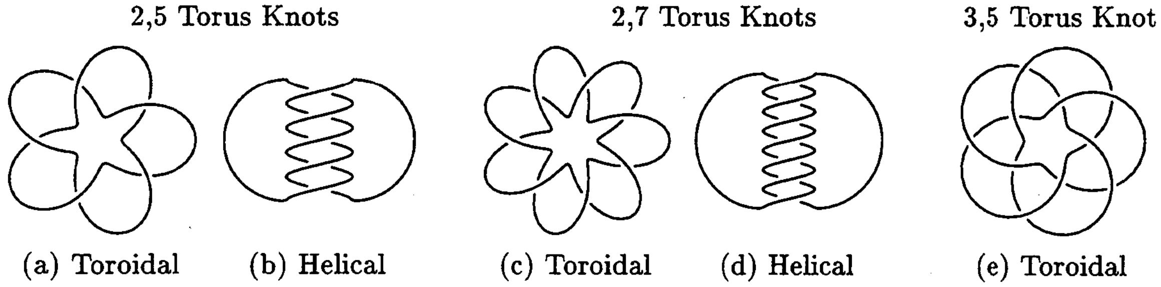


Figure 9: Miscellaneous Torus Knots

Before looking at the results of the computations it is useful to look the evolving configurations generated by the algorithm with this energy. Figure 10 shows the 9 turn noose evolving. This figure

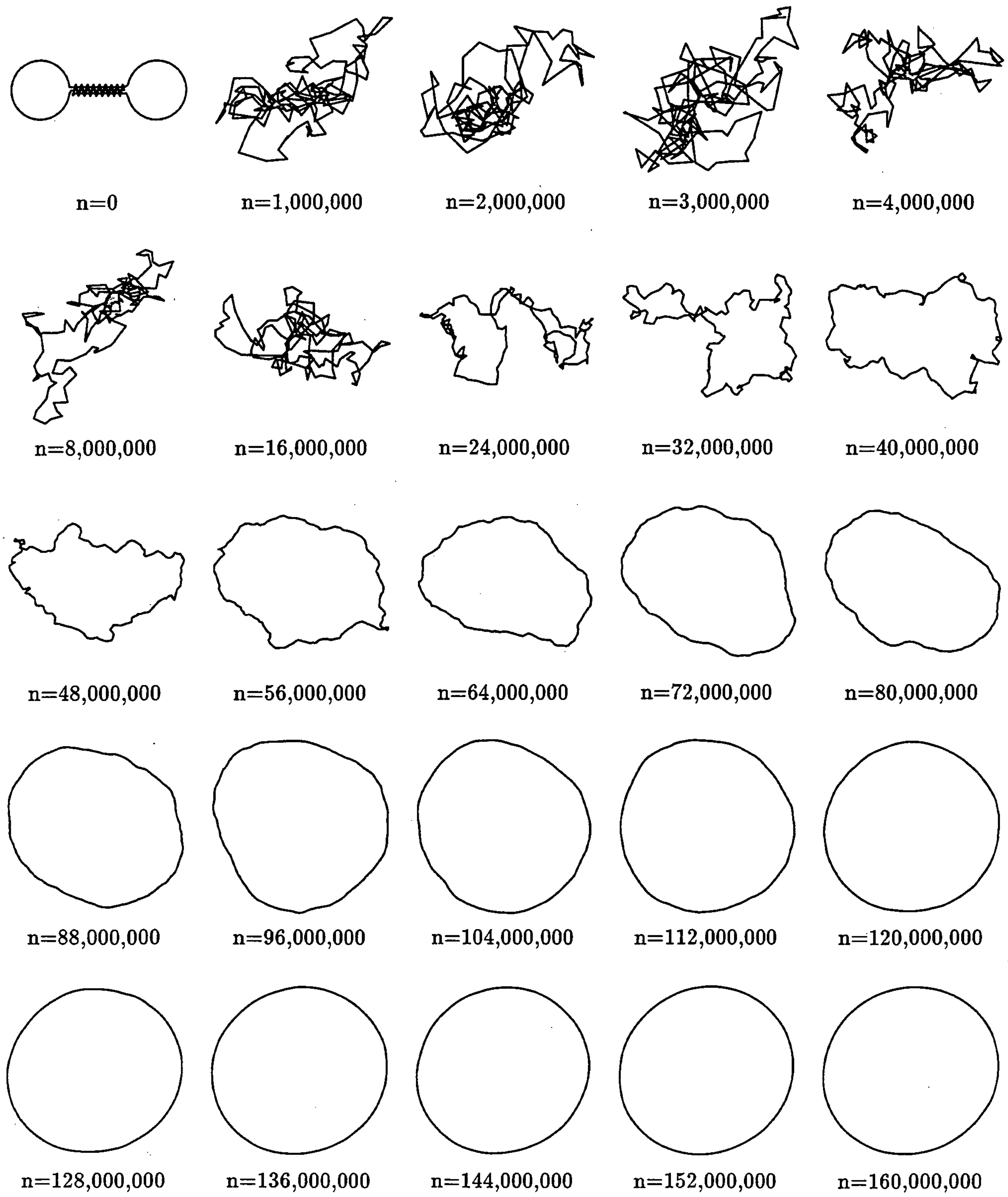
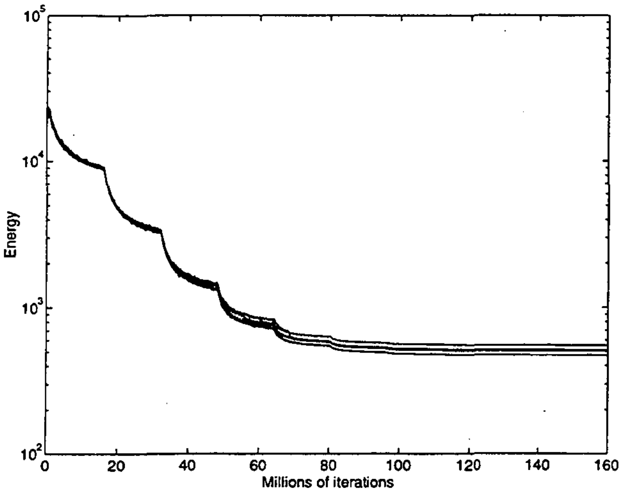
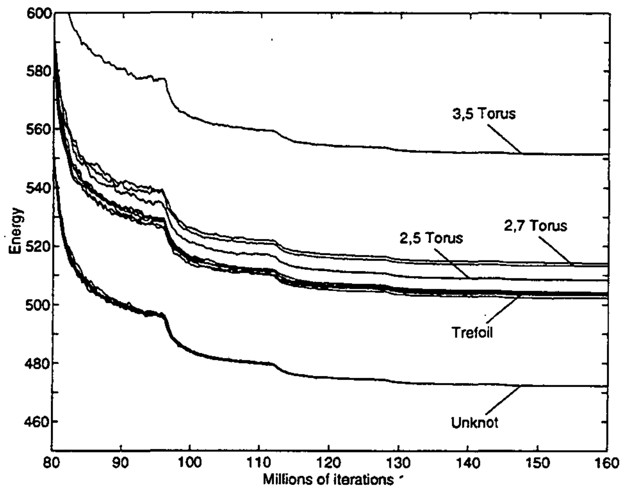


Figure 10: Noose Evolving

illustrates several things. First, at higher temperatures (i.e. near the beginning of the computation) the movements are fairly violent and the knot quickly becomes a jumble of line segments. During this time it is difficult to determine if the knot is evolving toward its canonical form. Even at iteration 16,000,000 it isn't clear what has happened. By iteration 24,000,000 the knot seems simpler and by iteration 40,000,000 all the inner loops have essentially unwound. The pictures of the next 120,000,000 iterations show how slowly the algorithm converges to final, minimum energy configuration. More will be said about this later.



(a) Overall Evolution



(b) Energy Splitting

Figure 11: Physical Energy of Evolving Knots

The overall results of the computations using the physical energy are summarized in Figure 11. The first graph shows the energy as a function of the number of iterations. The computations for all the knots mentioned above appear on this graph. Each time the cooling was restarted the energy of the knots drops rapidly and then levels off. If the cooling was not restarted the energy would not drop much more during the rest of the computation. Note that all the knots follow essentially the same energy curve until the end of the computations; here they begin to split. The second graph shows a close up of the final portion of first graph. The energies have split into five distinct levels which correspond to the five different topological knots present. Note that in some cases the minimum energy of different topological knots is not very different. Thus, some topological knots which are different might not be easily distinguished using this energy.

Some other points should be made. The convergence of these computations to a global minimum would be asymptotic if the temperature was inversely proportional to the logarithm of the number of iterations. With the restarting of the cooling no such convergence is guaranteed. During the computations several examples did not converge. One of the computations with a 5 turn noose failed to unwind all the inner loops. When computed again with the same parameters and a different set of random numbers all the inner loops unwound and the final configuration was approximately a circle. Computations with the 2,5 torus knot and the 2,7 torus knot also failed to converge during some computations. The results shown in the graphs correspond to the runs that found the lowest energies in each case.

In addition to the minimum energy, the form of the final configurations is worth noting. For the unknot the final configurations were all approximations of a circle in three dimensions. In the case of the trefoil knots the final configuration was less appealing. A typical example is shown in

Figure 12. Here the essence of the knot collapses into a small portion of space as the detail shows in an exaggerated fashion. The results are similar for the other torus knots. Since the energy is defined pointwise, the algorithm found a way to minimize the energy by interleaving the points. The 2,5 and 2,7 torus knots computations ended in configurations similar to the trefoil knot (i.e. with one large and one small loop). This explains the similarity in energy. The 3,5 torus knot computation resulted in a final configuration which had one small loop, one slightly larger loop around that loop and one large loop. This accounts for the relatively large increase in minimum energy for this knot compared with the other torus knots. Again, the essence of the knot collapsed into a very small region of space. This collapsing also seems to be responsible for some of the convergence problems.

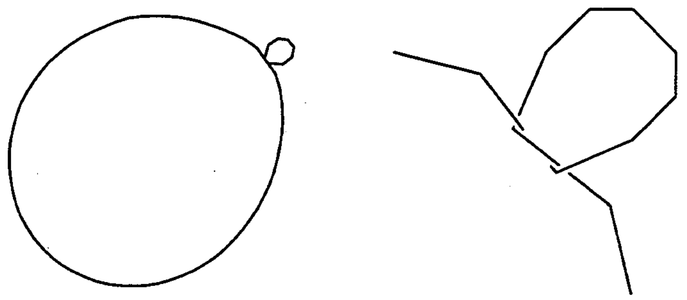


Figure 12: Minimum Energy Trefoil

In summary, the algorithm was able to put initially complex knots into a consistent form based on their topological type. The physical energy used seemed to be inadequate for several reasons. First, the minimum energies of different topological knots were often not very different and thus difficult to resolve with this algorithm. Second, the final configurations were difficult or impossible to recognize since the details of the knot structure collapsed into a very small volume.

4.2 Geometric Energy

Some computations were performed with the geometric energy described in section 3.2.2. These computations required much more time per iteration because $O(N)$ double integrals needed to be evaluated each iteration. Initially, we attempted to evaluate these integrals by point sampling the integral at the midpoints of the segments. For the trefoil knot this led to final configurations similar to those shown in the previous section. These were spurious results because the geometric energy goes to infinity as the distance between distinct portions of a knot goes to zero. The problem resulted from the point sampling of the integral at fixed values in the parameter space. The energy of these spurious final configurations was about one half the expected minimum energy [3].

To avoid these problems an adaptive technique was used in the integral evaluation. This eliminated the spurious solutions. Unfortunately, it greatly increased the time to computed each iteration. As a result, the total number of iterations feasible for a given computation was much less than in the case of the physical energy.

Due to this restriction, the knots used in these computations were not particularly complex however, when possible, several starting configurations were chosen. The knots used are shown in Figure 6a, Figure 7, and Figure 9. These include an unknot, two trefoil knots, and several other torus knots.

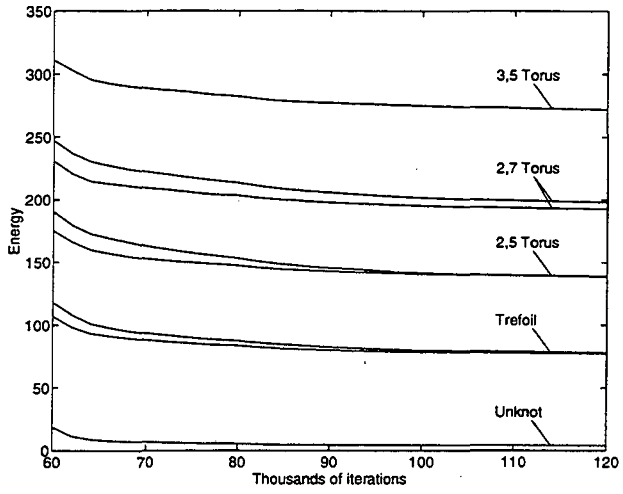
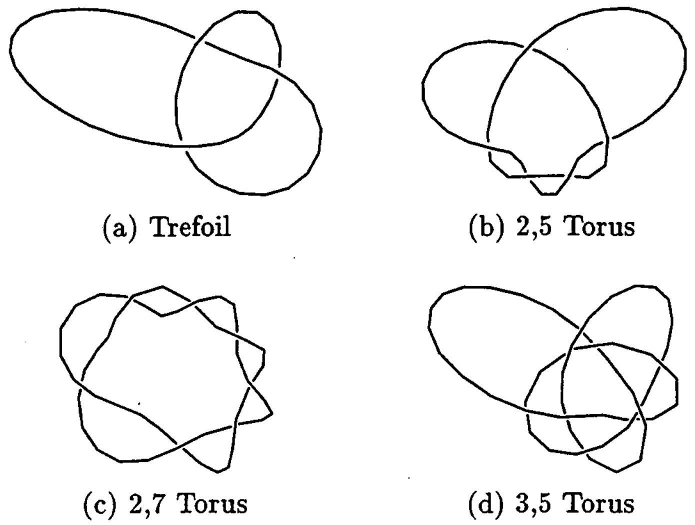


Figure 13: Geometric Energy Splitting

For all the computations the restart time t_r was chosen so the cooling restarted, as described in section 3.4, when the ratio of T_k to T_{k+1} was equal to 10. This implies that $t_r = e^{10} - e \approx 22,000$. The constant c_t was chosen so 20,000 iterations took place during this time. The initial temperature T_0 was 200.0 and the minimum temperature T_{min} was 0.0002. Thus, 120,000 iterations were performed in each case to compute the minimum energy configuration. The constant c_σ was 1.0 in all cases.

In the initial stages of the algorithm, the results look very much like the results obtained with the physical energy, but as the temperature drops the energies of the different knot types split more distinctively, see Figure 13. Note that these computations have not completely converged due to the limited number of iterations per computation. Also, it appears that a 2,9 torus knot might have a minimum energy very similar to a 3,5 torus knot.


(a) Trefoil

(b) 2,5 Torus

(c) 2,7 Torus

(d) 3,5 Torus

Figure 14: Minimum Geometric Energy Configurations

The minimum energy configurations of the geometric energy are much more recognizable. Examples for all the knots tried are shown in Figure 14 (the unknot is not included since its minimum energy configuration is simply a circle). Remember that the geometric energy is invariant under Möbius transformations of the knot. Thus some of the final configurations might look a bit unusual but they are only a Möbius transformation away from a more appealing form.

4.3 Observations

Although simulated annealing with these energies was very successful putting knots in a canonical form, several things should be noted. Picking the constant of proportionally between the temperature and variance is not always trivial. Theoretically, with logarithmic cooling, the algorithm converges to a global minimum regardless of the value of the constant but in practice it makes a very large difference if the knot is very complex. If this constant is too large, most of the perturbations are rejected because they cause the knot to cross itself or the energy increases too much for the new configuration to be accepted. If the constant is too small, the perturbations are too small and the knot doesn't reach a minimum energy configuration in a reasonable amount of time.

Also, choosing the rate of cooling is problematic. Less problematic, but still ad hoc, is the choice of the time to cool before restarting the cooling process. In general, a reasonable rule is: If the knot doesn't reach a minimum energy configuration by the end of the computation, then reduce the cooling rate and restart after a longer time. In this way more time (i.e. iterations) is spent logarithmically cooling in each temperature range; for a given problem, this is eventually long enough for the algorithm to (nearly) converge. Unfortunately, this is strongly dependent on the initial knot. The more complex the knot, the more iterations are required in each temperature range.

Finally, the convergence of the algorithm to the minimum energy configuration once the knot is near the global minimum is very slow. This problem becomes worse as the number of segments increases (i.e. the dimension of the configuration space).

5 Future Work

Given our initial success recognizing knots using simulated annealing to minimize the energy of knots, many improvements are possible. Speeding up the computation is an obvious and important improvement. This can be achieved in a variety of ways.

One way involves using a simulated annealing algorithm that converges faster. Several have been proposed in the literature. One is based on choosing perturbations from a Cauchy distribution and is outlined in [13]. This method claims convergence if $T = \frac{T_0}{1+t}$. Another method first discussed in [8] and later made available in software form claims exponential convergence (i.e. convergence if $T = T_0e^{-t}$). In both cases, the knot must be perturbed globally each iteration (i.e. all the vertices need to be perturbed simultaneously). Other approaches involve keeping the temperature fixed until the configurations reach "equilibrium" and then lower the temperature more quickly. These approaches are attractive if a suitable notion of equilibrium is available.

It is also clear that much of the computation time is spent finding the global minimum once the current configuration is close the minimum energy configuration. This time could be greatly decreased if a deterministic method for finding a local minimum (e.g. gradient descent) could be used. For this to be successful the algorithm needs to detect when the current configuration is in the global minimum basin of attraction (i.e. the global minimum is the current local minimum). It is not clear how to do this automatically in best possible way. Switching to a deterministic method at a given temperature would be a reasonable start.

Another issue is the convergence of the algorithm. Even if the correct cooling was used with this algorithm, the proof of asymptotic convergence is only valid when minimizing a function over all Euclidean $3N$ space. The algorithm is restricted to a subset of this space which corresponds to all geometric knots that can represent the topological knot corresponding to the initial geometric knot. This constraint manifests itself in the algorithm when perturbations are rejected because they would cause the knot to cross itself (see section 3.3). This problem becomes acute, both theoretically and practically, when global perturbations are used. We believe scaling the perturbations to "fit" the constraints of the space can overcome this problem. This amounts to moving the boundaries we cannot cross to an infinite distance. Thus, the algorithm will operate inside an unconstrainted space with a distorted metric. This should help make a proof of asymptotic convergence possible and it makes global perturbations reasonable in practice.

6 Conclusions

In the context of recognizing knots, simulated annealing has proved useful in finding geometric knots close to a global minimum of an energy function. Our computations have shown that given enough iterations all the knots tried end up close to a canonical form. If purely logarithmic cooling was used and the space was unconstrainted this result would be guaranteed. It is clear that physical energy used in some of these computations does not have all the properties desired. The energy minima do seem to lead to knots with a minimum of "clutter". Unfortunately, the portion of the knot that distinguishes it from other types of knots collapses into a small region and it cannot be viewed easily (if at all).

The more sophisticated geometric energy gives more desirable results. The energy minima for different topological knots are more distinct. The energy minimizing knots are much easier to recognize, but the Möbius invariance of this energy sometimes makes these final knots surprising in form.

In the case of both energies it is unlikely that the minimum energy alone of a topological knot will distinguish it from other topological knots, at least, in practice. It is also clear that faster methods of computing energy minimizing geometric knot are needed.

7 Acknowledgements

We would like to thank James Akao, Alexandre Chorin, Ricardo Cortez, Ole Hald, Vaughan Jones, Brent Milne, Sarah Patch, and Bill Thurston for helpful conversations and comments.

This work was supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098 and by the National Science Foundation/ARPA under grant DMS-8919074

References

[1] J.H. Akao. *Phase Transitions and Connectivity in Three-dimensional Vortex Equilibria.* PhD thesis, University of California at Berkeley, 1994.

[2] J.W. Alexander. Topological invariants of knots and links. *Transactions of the A.M.S.*, 30:275–306, 1928.

[3] S. Bryson, M.H. Freedman, Z-X. He, and Z. Wang. Möbius invariance of knot energy. *Bulletin of the A.M.S.*, 28(1):99–103, January 1993.

[4] A.J. Chorin. Equilibrium statistics of a vortex filament with applications. *Communications in Mathematical Physics*, 141(3):619–631, 1991.

[5] A.J. Chorin. *Vorticity and Turbulence.* Springer, 1993.

[6] P. Freyd, D. Yetter, J. Hoste, W.B.R. Lickorish, K. Millet, and A. Ocneanu. A new polynomial invariant of knots and links. *Bulletin of the A.M.S.*, 12:239–246, 1985.

[7] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[8] L. Ingber. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12(8):967–973, 1989.

[9] V.F.R. Jones. A polynomial invariant for knots via von neumann algebras. *Bulletin of the A.M.S.*, 12:103–111, 1985.

[10] W.B.R. Lickorish and K. Millet. A polynomial invariant of oriented links. *Topology*, 26(1):107–141, 1987.

[11] J. O'Hara. Energy of a knot. *Topology*, 30:241–247, 1991.

[12] K. Reidemeister. *Knotentheorie.* Springer, 1974.

[13] H. Szu and R. Hartley. Fast simulated annealing. *Physic Letters A*, 122(3,4):157–162, June 1987.