Motion Planning for Knot Untangling

Andrew M. Ladd and Lydia E. Kavraki

Dept. of Computer Science, Rice University, Houston, TX 77005, USA

Abstract. When given a very tangled but unknotted circular piece of string it is usually quite easy to move it around and tug on parts of it until it untangles. However solving this problem by computer, either exactly or heuristically, is challenging. Various approaches have been tried, employing ideas from algebra, geometry, topology and optimization. This paper investigates the application of motion planning techniques to the untangling of mathematical knots. Such an approach brings together robotics and knotting at the intersection of these fields: rational manipulation of a physical model. In the past, simulated annealing and other energy minimization methods have been used to find knot untangling paths for physical models. Using a probabilistic planner, we have untangled some standard benchmarks described by over four hundred variables much more quickly than has been achieved with minimization. We also show how to produce candidates with minimal number of segments for a given knot. We discuss novel motion planning techniques that were used in our algorithm and some possible applications of our untangling planner in computational topology and in the study of DNA rings.

1 Introduction

A knot is a simple, closed piecewise linear curve and is called an unknot if there exists an ambient isotopy[1] transforming it to a triangle. Otherwise it is a non-trivial knot [29]. Two key problems in computational knot theory are testing whether a given knot is the unknot and testing whether two knots are equivalent [6]. The explicit computation of isotopies between geometric objects is the central topic of study in motion planning. A difficult example exists in very tangled embeddings of simpler knots. In Figure 1, we illustrate the untangling of a very complicated unknot. The computation of untangling paths is the task addressed in this paper. There are two purposes for this work. Firstly, we want to explore how motion planning techniques can be applied to knot untangling and determine if it has advantages over other methods. Secondly, by studying an easily posed high dimensional planning problem involving manipulation of a reparametrizable deformable object and motion in a non-manifold space we hope to learn new techniques for motion planning and robotics.

Some of the current challenges at the frontiers of modern motion planning applications stem from high dimensional or even non-parametric configuration spaces, complex but 'soft' constraints on allowable motions imposed by

[1] Piecewise linear homeomorphism of the space around the knot.

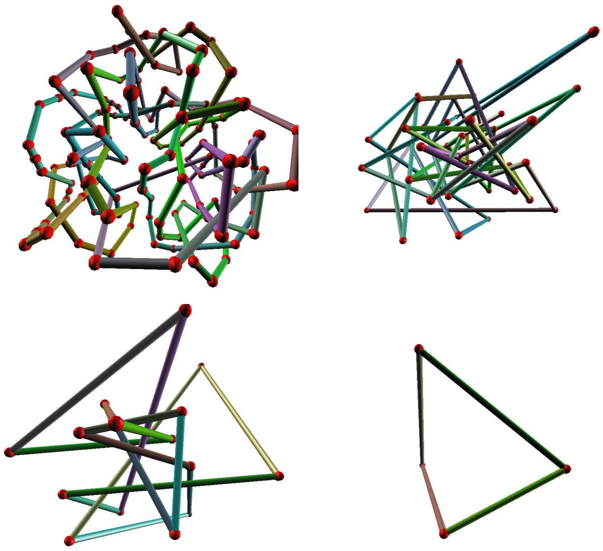

Fig. 1. Knot untangling of a tangled 133 segment unknot.

the energetics of an underlying physical system and the possibility of innovative uses of planners for discovery. Recent problems of interest are daunting but may have a great deal of structure that can be exploited to sidestep the inherent complexity of high dimensional planning in constrained spaces. Motivating examples can be found in recent work on planning schemes for modular and reconfigurable robots [8,41,42], planning for deformable robots [28,5], abstractions of random planning analysis to non-parametric, non-manifold spaces [27], approximate dimensionality reduction of configuration space [40], determining protein folding pathways [38], planning in the energy landscapes of molecules [4], and computing stationary distributions of Markov chains defined over roadmaps for analysis of Monte Carlo simulations of protein folding phenomena [3].

Knots are artificial geometric objects. The space of all knots does not have a finite basis and a given embedding of a knot might need hundreds of parameters to describe. Many configurations of a given knot are nearly degenerate and must be avoided to maintain the topology of the knot as it is manipulated. The number of connected components in the space grows roughly exponentially with the number of pieces in the knot. A given knot configuration is reparametrizable: a fact which can be used or ignored in a planner. The use of reparametrization is a critical difference from traditional planning tasks. In particular, the space of allowable configurations of a knot is not a manifold. Other non-manifold configuration spaces can be found for true deformable robots and for reconfigurable robots.

Problem Statement In this paper, we take the approach of using motion planning to find a sequence of motions and transformations that reduces the number of segments in a knot to the minimal number possible for that knot type. During these motions and transformations the topological type of the knot is preserved. We call this the knot untangling problem. When cast as a robotics problem, untangling is about manipulation of a reparametrizable, deformable object possibly requiring many parameters to describe important configurations and a complex structure which lends itself to the definition of energy functions.

Importance In computational topology, untangling can be used for unknot detection and more generally for the computation of the minimum number of segments to draw a given knot in 3d. This work hopes to gain insight into other robotics problems such as deformable robots, reconfigurable robots, protein and molecule robots, and, in general, planning with many linkages. Finally, it is possible that this work can provide insight into designing computational tools for reasoning about DNA rings as discussed in Section 2.

Contribution This paper describes the design and implementation of a novel motion planner for knot untangling. It makes some use of energy minimization and formulates a sequence of local goals that, when taken together, achieve a global untangling. It is a randomized tree-based planner. We demonstrate the effectiveness of this planner by applying it to some standard benchmarks used in other implementations as well as applications to some random examples. Our experimental results are significantly better than comparable implementations in the literature. We also discuss how the techniques employed might be applied to other high dimensional motion planning problems. Finally, we discuss the context of these results in both physical knotting and motion planning. Particular aspects of interest are the combined use of potential field methods and tree-based planners, the formulation of local goals, reparametrization for dimensionality reduction and the application of planning to computational topology.

Organization In Section 2 we discuss background material in computation with knots and discuss some connections to biology and robotics. In Section 3 we briefly discuss the geometry of knots as it pertains to our work. In Section 4 we describe the knot untangling planner. In Section 5 we detail some experiments and their results. In Section 6, we provide some explanation for the design choices in the planner we developed. In Section 7 we discuss the techniques we used in the context of motion planning. Finally, Section 8 we discuss future work and applications.

2 Background

We will now describe two classic problems in computational knot theory [6].

Problem 1 (UNKNOT). Given a knot K, determine if K is the same knot type as the unknot.

Problem 2 (KNOT EQUIVALENCE). Given two knots K_1 and K_2, determine if K_1 is the same type as K_2.

One possible proof that a given knot is the unknot is to exhibit an explicit isotopy transforming it to the triangle, in other words, to find a path transforming the given knot into a triangle. In the remainder of the background section, we will discuss in more detail some of the previous work on the unknotting and untangling problems. First we describe current work on exact algorithms, invariant calculation and complexity. Then we will describe some approaches to untangling which employ energy methods. Thirdly, we briefly discuss some interesting analogies with robotics. Finally, we will describe some of the connections that knotting has to biology.

2.1 Computation with Knots

Classification of knots is often attempted by computing knot invariants [29,2,16]. Invariants are typically computable constants or polynomials that are invariant over all embeddings of a knot type. If two knots differ over some particular invariant then they are different knots, however the converse does not necessarily hold.

There are many interesting invariant quantities defined for knots. The crossing number is the minimum number of crossings over all regular drawings of the knot on the plane. The unknotting number of a knot is the minimum number of the crossing flips required to transform the knot to the unknot. The stick number is the minimum number of segments required to draw a given knot in 3-space. The genus is the minimum genus over all surfaces that the knot can be drawn on. For a knot K, we write $c(K)$ for crossing number, $u(K)$ for unknotting number, $s(K)$ for stick number and $g(K)$ for the genus. An unknot K has $c(K) = u(K) = g(K) = 0$ and $s(K) = 3$. The simplest non-trivial knot, the trefoil, has $c(K) = 3$, $u(K) = g(K) = 1$ and $s(K) = 6$.

The UNKNOT problem is known to be in **NP** and KNOT EQUIVALENCE is in **PSPACE** [15]. More recently the MAXIMUM GENUS problem, $g(K) \leq k$, has been shown to be **NP**-complete [1]. Another recent constructive approach to unknot recognition works by enumeration of unknot diagrams and is polynomial in some cases [7]. Hass notes in [1] that if MINIMUM GENUS, $g(K) \geq k$, is in **NP** then UNKNOT is in both **NP** and $\mathbf{co-NP}$.

There are several polynomial invariants that are computed on knots. Each has various distinguishing powers. The Alexander-Conway invariant can be computed in **PTIME** but has limited distinguishing power [29]. Computing the Jones, Kauffmann and HOMFLY polynomials are **#P**-hard [20]. These polynomials can be computed in practice for small knots although is not known if there are multiple knots with the same polynomial as the unknot. Jenkins gives an efficient implementation of a HOMFLY calculator, although the size of knots it can handle is limited by bit arithmetic in the code. If c is the number of crossings in a link, then the HOMFLY polynomial can be computed in time $O(n!2^n c^3)$ and in space $O(n!c^2)$, where n is bounded above by $\sqrt{c}+1$ [21]. By using combinations of invariants [2], attempts to tabulate all distinct knots have been made. A recent list consists of all prime knots with at most sixteen crossings and consists of 1701936 distinct knot types. [16]. b

A key tool used in polynomial invariant calculation is a regular plane projection of a knot or knot diagram. These graph-like objects can be manipulated by three simple moves, called the Reidermeister moves, to obtain all knot diagrams for that knot type [29]. Hass and Lagarias derive an upper bound on the number of Reidermeister moves to transform a tangled unknot into a triangle. If k is the number of crossings in a given unknot diagram, it takes at most $O(2^{ck})$ where $c = 10^{11}$ [14] to transform this diagram to an unknot. Some intuition for the difficulty here comes from the fact that for some diagrams a non-minimizing Reidermeister move must be made, that is to say a crossing must be added before others can be removed.

2.2 Energy Functions

Knot energy was originally defined as way of arriving at "ideal" knot configurations. A good survey of this area can be found in Scharein's thesis [34].

There are several different varieties of energy functions in the literature. Many of the early equations were for the case where the function was a smooth curve and hence the energy was infinite for piecewise linear knots. For further details on energy for smooth curves refer to [9]. Due to the computational nature of our work, we restrict ourselves to energy defined over piecewise linear knots and give only two important examples from the literature. Other examples of energy functions can be found in [30,11,13,34] and include geometric energy, Möbius energy, spring energy, electrostatic energy and others.

Minimum Distance Energy Minimum distance (MD) energy is an approximation of distance integral over all pairs of points. The approximation is for the case of a piecewise linear knot and uses the minimum distance between pairs of segments (d_{MD}). Given a knot K with segments $s_1, ..., s_n$, we define MD energy of the knot as follows

$$E_{MD}(K) = \sum_{s_i, s_j \text{ disjoint}} \frac{|s_i||s_j|}{d_{MD}(s_i, s_j)^2}.$$

This energy function is a sum that can be computed in $O(n^2)$ time. We describe a calculation of d_{MD} in Appendix A. The energy is bounded from below by $2\pi \cdot c(K)$ [37]. To our knowledge, no convergence guarantees have been proved for a gradient descent energy minimization and experimental evidence suggests the presence of local minima, large plateaus and nearly non-unimodal behaviour [43,13].

A key problem with MD energy is that there is a tendency for tangled portions of the knot to become small. This is a result of the length normalizer term in the equation. In particular, the global minimum for non-trivial knots do not conform well to intuitive notions of "ideal" configurations. It has been suggested that this causes convergence problems during energy minimization [30].

Symmetric Energy Another approach is to view the knot as a radiating tube of small radius. Symmetric energy measures the amount of self-illumination of the knot. This energy can be calculated for a knot K parametrized by $x(t)$, $y(t)$,

$$E_S(K) = \int\int \frac{|dx \times r||dy \times r|}{|x-y|^2},$$

where $dx = \dot{x}(t)dt$ and $r = (x-y)/|x-y|$. This energy function does not shrink tangled portions of the knot and has been used to make beautiful drawings of knots with the software KnotPlot [34].

2.3 Energy Minimization

Given an energy function defined over the space of knots, there are various techniques to find its global minimum. The objective is to optimize the rate of convergence to the global minimum or at least to a good local minimum.

Random Perturbation and Annealing A simple technique for energy minimization is the random perturbation method. Many variations are possible and have been attempted. This technique has been used by Simon [37] and Scharein [34]. An important extension to perturbation techniques is to use simulated annealing to choose the magnitude of the random perturbation. The annealing schedule proposed by Ligocki and Sethian generates new configurations with a standard deviation proportional to an inverse logarithmic function of time [30]. Unfortunately, there seems to be a very poor rate of convergence to the global minimum for several reasons: rejection of self-intersecting perturbations, minima effects in the energy function and the high-dimensionality of the space necessitates slow cooling.

An important criticism of perturbation and annealing methods is that much of the time is spent checking for self-intersection. It is difficult to avoid making $\frac{n^2-3n}{2}$ intersection checks when the vertices are moved and usually many perturbations are required to escape a local minimum or difficult regions. Experimental evidence suggests that there are many such regions.

Gradient Descent Another standard energy minimization approach is to use gradient descent which will quickly find a local minimum. Few self-intersection checks are needed assuming an energy function that goes to infinity as the knot approaches a singular embedding. To escape a local minimum, heuristic perturbation methods can be used after convergence occurs. This technique has been successfully implemented in Wu's MING knot evolver [43]. This evolver uses MD energy and was used to find a 22 edge unknot which was a MD energy local minimum as well as other difficulties with MD energy.

Stochastic Energy Functions An interesting approach to random perturbation can be obtained by perturbing the energy function at random rather than the object. Grzeszczuk, Huang and Kauffman [12,13] use spring energy, electrostatic energy, randomly positioned point charges and reparametrization of the knot to create a random family of energy functions. The magnitude of the perturbation is determined by an annealing schedule. Kauffman and al. report their method to be roughly twice as fast as that of Wu on a difficult benchmark.

2.4 Analogies with Robotics

We would like to point out the relation between efforts in knot energy minimization and potential field efforts in motion planning. Many of the same difficulties arise. In motion planning, potential field methods have largely been replaced by global probabilistic planners [22,19,18,17,23,24,26,25,28,5]. It is noteworthy to observe that Scharein, in his thesis, remarks that with some human interaction, e.g., dragging vertices around, minimizations of some very tangled knots could be achieved considerably more quickly [34].

2.5 Knots in Nature

The definition of energy functions on knots has had several purposes: to guide the simplification of tangled embeddings, to find aesthetic, symmetric drawings, to search for ideal energy minimal configurations and to explore connections with the energetics of knotting phenomena in DNA.

This final purpose is motivated by the abundance of long ring-shaped molecules in living organisms. The discovery of knotted DNA led to the application of knot theory to the study of long rings. Since then a number of DNA knots have characterized in detail, and their knot-types have provided insights into the mechanisms of the reactions that produced them.

It is known that the probability that a random knot on a square lattice is the unknot goes to 1 exponentially with the length of the knot [39,32]. Further work in this direction attempts to asymptotic knotting and linking of simple curves moving at random in some space under various models. This general body of work provided for theoretical evidence that knotting phenomena occur in long DNA rings. This has been validated experimentally, for example Shaw and Wang have explicitly measured knotting probability in DNA molecules as a function of concentration [35,36].

Recent studies have shown the formation of knots and links in DNA can play an important role in cell replication. Knotted and linked conformations are more spatially compact. However, during replication DNA must untangle. There is some insight into the mechanism whereby this occurs and it may be that knot energetics in DNA rings play an important role in the life cycle of a cell [33,10].

3 Geometry of Knots

In the remainder of the paper, we will describe the design and implementation of our planner. In this section, we discuss the geometry of knots, introduce notation and describe the calculations used to avoid self-intersection.

A given knot K with n pieces is defined by a sequence of vertices $p_1, ..., p_n$. For convenience, we sometimes say p_{n+1} to mean p_1 and p_0 to mean p_n. Formally the knot consists of the subset of $\mathbf{R}^3$ defined by the union of the segments $s_i = \overline{p_i p_{i+1}}$, for $1 \leq i \leq n$. We say two segments belonging to a knot are disjoint when they do not share an endpoint.

The clearance of a knot K with vertices $p_1, ..., p_n$ is the function

$$clr(K) = \min_{s_i,\ s_j \text{ disjoint}} \{d_{MD}(s_i, s_j)\}.$$

d_{MD} is the minimum distance between a pair of line segments in $\mathbf{R}^3$. This expression measures the distance between disjoint segments. When $clr(K) = 0$, the knot K is singular. When $clr(K) > \epsilon$ for some $\epsilon > 0$, K is ϵ-non-singular. The self-intersection equations we use are not numerically stable for small values of $clr(K)$ when evaluated with floating point precision. Also, it is assumed that all knots we consider are non-singular.

Given two knots of n pieces, K_a with vertices $p_1, ..., p_n$ and K_b with vertices $q_1, ..., q_n$, we can define a linear interpolation between K_a and K_b parametrized by time t, $K(t)$, which is the knot defined by vertices $r_i(t) = p_i + t \cdot (q_i - p_i)$. This interpolation preserves the topology of the knots in question when the knots $K(t)$ are non-singular for all $t \in [0, 1]$. In particular, existence of such an interpolation between K_a and K_b is a sufficient condition for concluding topological equivalence of K_a and K_b as it implies an explicit ambient isotopy. Detecting the existence of zeroes for $clr(K(t))$ for $t \in [0, 1]$ is the self-intersection detection problem for knot manipulation.

Adding and removing vertices can be accomplished via a so-called elementary or triangle move, illustrated in Figure 2. A vertex p_i can be removed from knot K with vertices $p_1, ..., p_n$ to obtain knot K' with vertices $p_1, ..., p_{i-1}, p_{i+1}, ..., p_n$ if the triangle $\Delta p_{i-1} p_i p_{i+1}$ is unpunctured by all segments disjoint from s_{i-1} and s_i. Alternately, when three vertices are collinear,

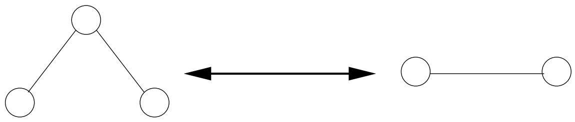

Fig. 2. An elementary move.

the middle vertex can be removed. This move and its inverse are the elementary moves and any two equivalent knots can be related by a sequence of elementary moves. The best known upper bound on the number of moves required to transform a given unknot by elementary moves to a triangle is $O(2^{ct})$, where c is 10^7 and t is the number of tetrahedrons in a space triangulation where the knot can be drawn on the 1-skeleton [14]. The authors know of no non-trivial lower bounds in the literature.

Geometry of Line Segments

The geometry of piecewise linear knots is defined in terms of the geometry of line segments. In this subsection, we briefly describe how we calculate the zeroes for the clearance function.

Consider a pair of non-parallel infinite lines in 3-space $l_a(s) = p + s \cdot \boldsymbol{u}$ and $l_b(s) = q + s \cdot \boldsymbol{v}$. The lines will intersect when they are coplanar, that is to say when they both lie on a plane with normal $\boldsymbol{n} = \boldsymbol{u} \times \boldsymbol{v}$. This occurs when

$$\boldsymbol{n} \cdot (q - p) = 0. \qquad (1)$$

When both lines are moving along known polynomial trajectories, i.e., $p, q, \boldsymbol{u}, \boldsymbol{v}$ are functions of t, a time variable, equation (1) is a polynomial in t, the zeroes of which occur at the intersection points of l_a and l_b.

The same holds for segments. The only possible intersection points occur at the zeroes of equation (1). At each zero, it can be verified if the segments are indeed intersecting by the minimum distance between them. The equations we use for this calculation are detailed in Appendix A.

Suppose a pair of segments $\overline{p_1(t)p_2(t)}$ and $\overline{p_3(t)p_4(t)}$ are moving along polynomial trajectories where the degree of the ith trajectory is δ_i. The degree of the intersection polynomial from equation (1) is found by making the following simple observation. When

$$deg(\boldsymbol{n}) = \max\{\delta_1, \delta_2\} + \max\{\delta_3, \delta_4\}$$

and if $\boldsymbol{w}$ is a vector between any pair of points on the opposite segments

$$deg(\boldsymbol{w}) \geq \max\{\min\{\delta_1, \delta_2\} + \min\{\delta_3, \delta_4\}\}$$

implies that the degree of the intersection polynomial is the minimum over all choices for $\boldsymbol{w}$ of $\deg(\boldsymbol{n}) + \deg(\boldsymbol{w})$. It is important to know the degree of the resulting polynomial when solving for the zeroes of d_{MD}, a naïve calculation can result in numerical failure.

4 Untangling Planner

The untangling planner works to reduce the number of vertices in a given knot. In this section, we will describe the design and implementation of our untangling planner. Essentially it is a tree-based planner similar to an RRT [23,24,26,25] or expansive space planner [19,17,18]. Rather than planning a path towards a particular configuration such as the unknot, the planner is used to find a configuration wherein a single vertex can be removed via an elementary move. When no progress is being made, the energy of the knot is optimized instead. If no progress optimizing energy is made, a random move is made instead. Other important design decisions that were made involved the sampling, energy and weighting functions used in the planner and the choice to avoid configurations where the clearance is very low and that are prone to numerical instability in the self-intersection detection.

Configuration Space Let $\mathcal{K}_n$ be the set of all non-singular knots K with n vertices and let $\mathcal{K} = \bigcup_{n \geq 3} \mathcal{K}_n$. This is the configuration space for the planner.

Let K be a knot with n vertices $p_1, ..., p_n$ and let K' be a knot with vertices $q_1, ..., q_n$. We say that $K \sim K'$ if the linear interpolation between them has no intermediate configuration that is singular. In particular, this proves that K and K' are the same topological knot.

Given a knot K with vertices $p_1, ..., p_n$ and $\epsilon > 0$ such that $clr(K) \geq \epsilon$, we define two local planners.

Linear Interpolation Planner The first planner moves a single, given vertex as close as possible towards a target point. Given $1 \leq i \leq n$ and a point $q \in \mathbf{R}^3$, we define a parametric knot $K(t)$ with vertices $p_1, ..., p_{i-1}, p_i + t \cdot (q - p_i), p_{i+1}, ..., p_n$. We find all the zeroes of equation (1) applied to all pairs of disjoint segments in $K(t)$ where one segment contains the ith vertex. We note that (1) will be a linear equation according to the analysis in Section 3. Suppose the zeroes occur at $0 < t_1 < \cdots < t_m < 1$. Beginning at the first zero, they are processed in order. The jth zero, t_j, will occur between some segment s with fixed endpoints and a segment $s'(t)$ which has p_i as endpoint. We calculate $d_{MD}(s, s'(t_j))$ and verify that it is larger than ϵ. If not, we find the smallest integer $r > 0$ such that $clr(K(\frac{t_j}{2^r})) > \epsilon$ and set $t^* = \frac{t_j}{2^r}$. If after processing all the zeroes, t^* is undefined then we calculate the smallest integer $r \geq 0$ such that $clr(K(2^{-r})) > \epsilon$ and set $t^* = 2^{-r}$. This allows us to find a knot $K' = K(t^*)$ where $K \sim K'$ and $clr(K') > \epsilon$. We summarize this construction as $K' = lip_i(K, q)$, lip is a mnemonic for linear interpolation planner. Since we are only moving a single vertex, this calculation can occur in linear time.

Elementary Move Planner The second planner removes a vertex by an elementary move. Given $1 \leq i \leq n$, we take the midpoint q of p_{i-1} and p_{i+1} and compute t^* as with the previous section. If $t^* = 1$, then the move is successful and we can safely remove the ith vertex with an elementary move to obtain K' with vertices $p_1, ..., p_{i-1}, p_{i+1}, ..., p_n$. We summarize this by the binary function $elm_i(K)$ which takes the value 1 if $t^* = 1$ and the value 0 otherwise.

Algorithm 1 Untangle a knot K_0

1: Generate a permutation of $1, ..., n$ called $j_1, ..., j_n$.
2: **for** i ranges from 1 to n **do**
3:    Check to see if $elm_{j_i}(K_0) = 1$.
4:    If so return K_0 with the j_ith vertex removed.
5: **end for**
6: Create a weighted tree T with root K_0 having weight 1.
7: **while** some number of iterations N **do**
8:    Choose a knot K at random from T using the weight distribution.
9:    Choose a vertex p_i at random from K.
10:    Check to see if $elm_i(K) = 1$, if so return the path from K_0 to K in T and K' which is K with p_i removed.
11:    Choose a point q distributed uniformly in the spanning sphere of K.
12:    Compute $K' = lip_i(K, q)$ and the associated t^*.
13:    Divide the weight of K in T by 2.
14:    Add K' to T with parent K and weight t^*.
15:    Check to if $elm_i(K') = 1$, if so return the path from K_0 to K' and K'' which is K' with the ith vertex removed.
16: **end while**
17: Computes the energies of all nodes of K.
18: If there is a node with energy less than K_0 return it and the path to it from K_0 in T.
19: Choose a knot K at random from T using the weight distribution.
20: Return the path from K_0 to K in T and K.

Vertex Freeing Algorithm 1 is called with some knot K_0, the input knot, and, N, which is the number of iterations to run the main loop before aborting. N should be chosen to balance the amount of energy optimization and the aggressiveness of the planner in trying to free a vertex. Implicitly there is a minimum tolerated knot clearance as a parameter as well.

The algorithm is split into three sections. Lines 1-5 are the first section, lines 6-16 the second and lines 17-20 are the third section. Let n be the number of vertices in K_0. The first section exhaustively searches for a vertex that can be removed by an elementary move. The vertices are considered in a random order to eliminate bias. The first section runs in time $O(n^2)$: each call to elm costs $O(n)$ geometric comparisons and n calls are made.

The second section is a probabilistic tree expanding planner which searches for a move that frees a vertex. This section is more complex and requires some simple data structures for a correct implementation. The tree is stored as a directed graph. The distribution of weights and associated probabilities for the tree nodes can be stored in binary heap-like structure which supports $O(\log N)$ weight insertion, weight modification and choose operation. Therefore lines 8, 13 and 14 cost $O(\log N)$. Lines 10, 11, 12 and 15 each require $O(n)$ operations as they require a single traversal of the knot. The remaining lines in section two can be implemented in constant time. Thus the section can be implemented in time $O(Nn + N \log N)$. The space usage for this section is dominated by the storage for the tree nodes and can be seen as $O(nN)$. By keeping only incremental updates and check-pointing periodically the space can be reduced to closer to $O(N)$ without sacrificing much of the runtime.

Finally, the third section requires energy calculations over all nodes in the tree. We are assuming that the time to calculate the energy function is dominated by an operation on pairs of the disjoint edges. The energy of K_0 can be calculated in $O(n^2)$. Since each other node is constructed by an incremental change to one vertex, the energy can be calculated from the previous node in time $O(n)$ with appropriate book-keeping in the tree. Thus line 17 can be done in time $O(n^2 + nN)$ and this cost dominates section three.

Global Scheme The untangling algorithm is called repeatedly some number of times until the knot is no longer becoming simpler. Depending on the application in mind, it may be enough to simplify a certain amount or it might be better to search aggressively for a minimal embedding.

Choosing an Energy Function The energy function that we used was based on MD energy but we dropped the length normalizer to obtain

$$E(K) = \sum_{s_i, s_j \text{ disjoint}} \frac{1}{d_{MD}(s_i, s_j)^2}.$$

This odd choice of energy function seems experimentally to be a good one despite the obvious problem with it. Some explanation of why this seems to be is offered in Section 7.

5 Experiments and Results

This section will describe experiments using our untangler. We will show that our planner is significantly faster than the energy minimization approaches in the literature. The software packages we are comparing against were the simulated annealer of Ligocki-Sethian [30], KnotPlot by Scharein [34], MING knot evolver by Y-Q. Wu [43] and stochastic energy optimizer by Grzeszczuk, Huang and Kauffman [12,13]. Each of these packages either report some results in a related paper or are publicly distributed.

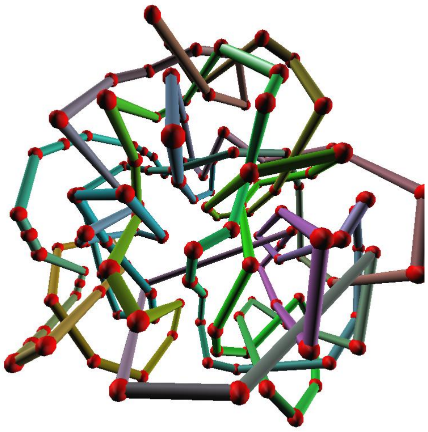

Fig. 3. The Ochiai unknot.

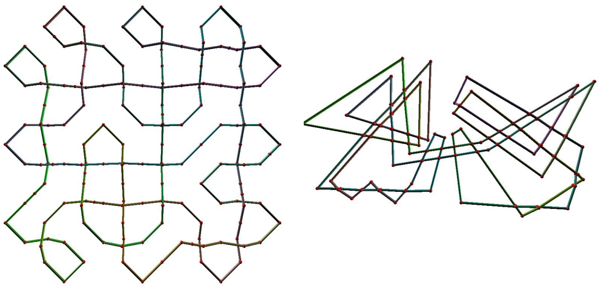

Fig. 4. Ligocki-Sethian example 2 and Twisted Freedman unknot.

Ligocki-Sethian unknot The second Ligocki-Sethian unknot shown in Figure 4 is a nearly flat weave with 160 vertices that was used as a test case for a simulated annealer. It was reported that it took more than 16000000 iterations to obtain an embedding that was recognizably a circle[30]. Each iteration involved $O(n^2)$ collision checks for each configuration generated. Our untangler solves this problem nearly instantaneously.

Twisted Freedman unknot The Twisted Freedman unknot is another nearly flat weave with 122 vertices that was used to show that MING evolver was faster and more precise than perturbation methods [43]. It is shown in Figure 4. We ran MING for more than 12 hours on a 300MhZ SGI O2 without the knot untangling to a circular embedding. Our untangler ran on a single cpu of a dual AMD 1900MP and spent an average of approximately 8 minutes for the runs that completed in less than 25 minutes. About a third of the runs ran for more than 25 minutes and were terminated.

Ochiai unknot The Ochiai is an unknot with 139 vertices that has foiled many energy minimizers. It was originally given as an example of a general construction in a paper by M. Ochiai in 1990. Since then it has often been used to test computational knotting programs. It is reported in [13] that MING running on an SGI O2 untangled it in 108 hours and stochastic minimization took 48 hours. KnotPlot was not able to reduce it [34]. Our untangler averaged about 10 minutes on a single cpu of a dual AMD 1900MP and had roughly one sixth of its runs terminated after 35 minutes.

Random unknots We generated several random unknots in hopes of obtaining more interesting examples to test on. We generated unknots by beginning with a circle of vertices and then making a large number of random moves. Even by running the tangler for many hours, we were not able to generate any configurations that were not solved instantaneously by the untangler.

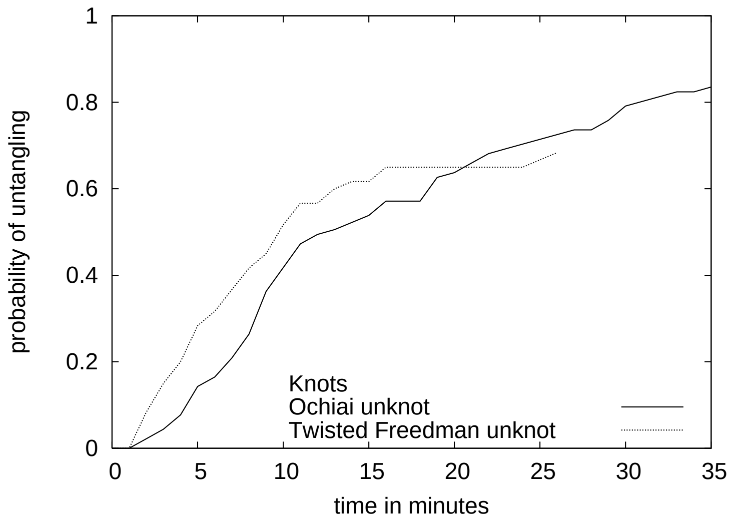

Fig. 5. Probability of finding solution versus time in minutes

Summary of unknot experiments We summarize the results of our untangling experiments in Figure 5. Once we factor in the difference in processor speeds, it is clear that our method is orders of magnitude faster than the energy minimizations in the literature. A second advantage is the probabilistic nature of the algorithm allows to run many in parallel to obtain better times. In the case of the Twisted Freedman and Ochiai this would mean solution times closer to two or three minutes.

Stick number of non-trivial knots The results we have reported up to now have been for unknots. Another application of the untangler is to minimize

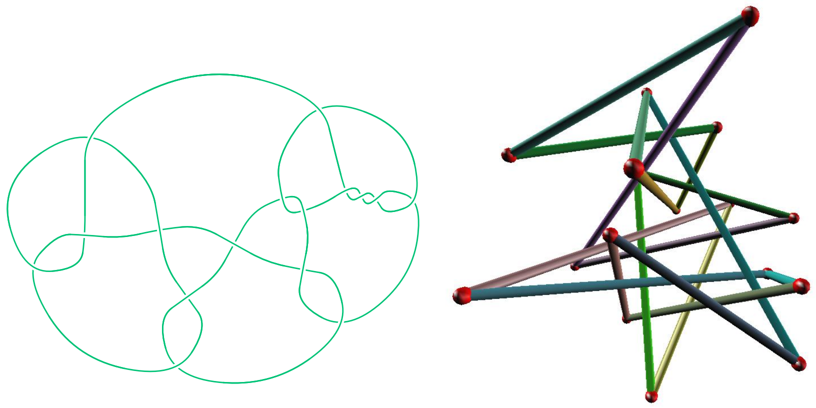

Fig. 6. The knot diagram for 16_{148} and 3d drawing with 17 sticks.

the number of segments used to draw a given knot. As an example we took the non-alternating knot 16_{148} from the database of knots stored in the software Knotscape by Hoste and Thistlethwaite and found a 3d presentation of it with 17 sticks after a few minutes of computation. The initial embedding had 50 segments.

6 Analysis of Heuristics

The algorithm we gave engendered several design choices. In this section, we explain the motivations for these choices.

Use of trees Knot space has many properties that make it hostile to PRM planners that build a roadmap [22]. It is very difficult to sample uniformly from a single connected component. Determining whether two knots are in the same component is an instance of UNKNOT or KNOT EQUIVALENCE and is challenging for examples with many pieces. Even finding two random knots K, K' such that $K \sim K'$ for 30 segment knots requires the generation of several million random knots. It is also difficult to find fast algorithmic solutions to knot matching that takes into account the symmetries of knots. For these reasons, we elected to use a tree expanding planner in the spirit of other such planners [19,18,17,23,24,26,25] to avoid the difficulties with global sampling.

Tree expansion In tree-based probabilistic planning, it has been observed that it is undesirable for the tree to be concentrated in a small area of the space. The planner will be much faster if it covers the space more quickly and other tree-based planners have employed a number of heuristics for this reason

[19,18,17,23,24,26,25] When moving all vertices simulataneously along some linear path, the knot is constrained by the most constrained vertex. When moving a single vertex at random n times, we observed that the final position of the knot was typically much further from the start position. Since these two schemes are roughly the same amount of work computationally, moving one vertex at a time has a clear advantage. A further benefit is that the intersection equations have lower degree.

Freeing a vertex Finding a path that takes a given tangled unknot to a triangle requires finding a very long path. Worse, when trying to simplify a knot of unknown type, the final configuration may not be known. One possiblity for finding a path to untangled configuration is to construct a very large search tree. This is impractical as it is very space intensive. In addition, it is also very slow if backtrack after performing an elementary move is possible, even for completely trivial input such as n vertices arranged in a planar polygon. This trouble occurs because there can be at most $n!$ ways to reduce the knot to a triangle. By disallowing backtrack, essentially throwing away the tree but for a path ending in an elementary move, we speed the planner greatly at the cost of having some probability of putting the knot in a bad configuration. We believe that such bad configurations are responsible for the occasional long run times observed in the experiments section.

Energy hints versus random After running the loop in Algorithm 1 N times the next configuration to consider is chosen. In our implementation we used energy to make this choice. By eliminating lines 17 and 18 from the final section of Algorithm 1, thus always executing the random guess on line 19, we obtained very poor results. We ran 50 trials on the Ochiai unknot for nearly an hour with almost no progress. We observed that the knot tended to stay in high energy configurations when moving at random. This is consistent with our observation that the planner seems to hampered by high energy knot configurations.

Choice of energy function The energy function we used is non-standard and was chosen for two reasons: it is easy to compute and it does not shrink tangled portions of the knot like MD energy does. The length normalizer in MD energy has a tendency to shrink certain tangled sections of the knot [30,13]. This is very undesirable and is one of the sources of convergence problems for MD energy. The problem of the knot becoming very large is dealt with by sampling in a sphere. We ran 50 trials on the Ochiai unknot with MD energy instead of unnormalized MD energy and very little progress was made. We found it surprising that the difference between the two functions was so striking because they superficially seem similar. When visualizing intermediate configurations during the untangling we observed that MD energy tended to shrink the tangled portions of the knot. The unnormalized MD energy tended to seperate some large triangles or "ears" from the knot. The planner would have ample space to pass other parts of the knot through these ears. That particular move seemed to be critical to making progress in the untangling.

7 Discussion

In the section we will discuss some our implmented techniques as they relate to motion planning and detail some future work.

Use of local goals Our planner decomposes the problem of finding an untangling into local goals. These goals consisted of finding a position where an elementary move can be made. This has two advantages: the local goal is much easier to plan for and the final configuration does not need to be known. This idea appears in robotics literature in task planning and in hierarchical planning. In our case, formulating the planning problem in terms of sequence of local goals was essential to obtain a usable planner. When it is possible to find such formulation, we suggest that it will be often a major improvement to the speed of the planner.

Use of energy There are several planning applications with natural energy functions such as protein folding, molecular docking and realistic deformable robots. Also, there has been some investigation in using energy when planning for such systems [38,4]. Our planner is a potential field planner which uses tree expanding planning to escape minima. In particular, we have demonstrated that both planning and energy hints are necessary for the untangler's efficiency. Furthermore, we observed that for the knot untangling planner, low energy configurations were typically better for the planner. We propose that the reason for this is that high energy configurations are more constrained and it is more difficult for the tree to escape from this strangely shaped region of configuration spaace. Energy functions with similar properties, either natural or artificially designed, might exist for other planning instances. This topic merits further investigation and awareness of hybrid energy and planning algorithms may lead to improvements in a variety of planners.

8 Future Work and Applications

Probabilistic completeness Although we have not shown probabilistic completeness for our planner, we give a conjecture which is both necessary and sufficient for our planner to be probabilistically complete.

Conjecture 1. Let K be a piecewise linear knot with n pieces. Suppose $s(K) < n$. Then there is a sequence of knots with n pieces $K = K_0, K_1, ..., K_m$ with $K_i \sim K_{i+1}$ and K_m has three collinear vertices.

The remainder of the argument follows the scheme described in [27].

Applications to Computational Topology In this paper, we provide a planner for knot untangling. The planner can be used for probabilistic unknot detection. A more general application of our planner can be found in stick number calculation of a knot. Our approach might give a fast method for finding good candidates for minimal stick knots. Another application is to calculate unknotting number of a knot. Given some knot, the task is to find a path containing as few crossings as possible that transforms that knot to the unknot. We propose a two-tiered planner combining our untangler with higher level planner that plans across knot types using a crossing operator.

Applications to DNA knotting Determining low energy paths for DNA knotting and unknotting is of interest in determining how certain processes occur in cell biology [10]. To achieve this we would need to use more complex energy functions that better reflect the energetics of DNA rings. This could mean combining simulation in the spirit of our work on knot tying for simulated rope [31] and planning. It may perhaps entail all paths analysis such that done with Stochastic Roadmap Simulation [3].

Acknowledgements

Work on this paper by Andrew Ladd and Lydia Kavraki has been supported in part by NSF 9702288, NSF 0114796, a Whitaker Biomedical Engineering Grant and a Sloan Fellowship to Lydia Kavraki. Andrew is also supported by FCAR. We would like to thank Nate Dean for discussions and insights. We also thank Rob Scharein for providing us with the Ochiai and Ligocki-Sethian unkots. Finally, we thank Jeff Phillips for our fruitful collaboration in our earlier work and for discussions about this work.

References

1. I. Agol, J. Hass, and W. Thurston. The computational complexity of knot genus and spanning area. (preprint.).
2. C. N. Anerizis. *The Mystery of Knots - Computer Programming for Knot Tabulation*. Series on Knots and Everything. World Scientifical Publishing Co. Pte. Ltd., 1999.
3. M. Apaydin, D. Brutlag, C. Guestrin, D. Hsu, and J. Latombe. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. In *International Conference on Computational Molecular Biology (RECOMB)*, April 2002.
4. M. Apaydin, A. Singh, D. Brutlag, and J. Latombe. Capturing molecular energy landscapes with probabilistic conformal roadmaps. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2001.
5. O. Bayazit, J.-M. Lien, and N. Amato. Probabilistic roadmap motion planning for deformable objects. In *IEEE International Conference on Robotics and Automation*, 2002.

6. M. Bern, D. Eppstein, and al. Emerging challenges in computational topology, 1999.
7. J. S. Birman, P. Boldi, M. Rampichini, and S. Vigna. Towards an implementation of the b-h algorithm for recognizing the unknot. In *KNOTS-2000*, 2000.
8. Z. Butler, K. Kotay, D. Rus, and K. Tomita. Cellular automata for decentralized control of self-reconfigurable robots. In *IEEE International Conference on Robotics and Automation*, 2001.
9. X. Dai and Y. Diao. The minimum of knot energy functions. *Journal of Knot Theory and its Ramifications*, 9(6):713–724, 2000.
10. R. Deibler, S. Rahmati, and E. Zechiedrich. Topoisomerase iv, alone, unknots dna in escherichia coli. *Genes and Development*, 15:748–761, 2001.
11. Y. Diao, C. Ernst, and J. Rensburg. In search of a good polygonal knot energy. *Journal of Knot Theory and its Ramifications*, 6(5):633–657, 1997.
12. R. Grzeszczuk, M. Huang, and L. Kauffman. Untangling knots by stochastic energy optimization. *IEEE Visualization*, pages 279–286, 1996.
13. R. Grzeszczuk, M. Huang, and L. Kauffman. Physically-based stochastic simplification of mathematical knots. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):262–278, 1997.
14. J. Hass and J. Lagarias. The number of reidemeister moves needed for unknotting. (preprint.).
15. J. Hass, J. C. Lagarias, and N. Pippenger. The computational complexity of knot and link problems. In *IEEE Symposium on Foundations of Computer Science*, pages 172–181, 1997.
16. J. Hoste and M. Thistlethwaite. The first 1,701,936 knots. *Math. Intelligencer*, 20(4):33–48, 1998.
17. D. Hsu. *Randomized Single-Query Motion Planning In Expansive Spaces*. PhD thesis, Department of Computer Science, Stanford University, 2000.
18. D. Hsu, R. Kindel, J. Latombe, and S. Rock. Control-based randomized motion planning for dynamic environments. In *Algorithmic and Computational Robotics: New Directions: The Fourth International Workshop on the Algorithmic Foundations of Robotics*, pages 247–264, 2001.
19. D. Hsu, J. Latombe, and R. Motwani. Path planning in expansive spaces. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 2719–2726, 1997.
20. F. Jaeger, D. L. Vertigan, and D. Welsh. On the computational complexity of the jones and tutte polynomials. In *Math. Proc. Camb. Phil. Soc.*, 108, pages 35–53, 1990.
21. R. Jenkins. A dynamic approach to calculating the homfly polynomial for directed knots and links. Master's thesis, Carnegie Mellon University, 1989.
22. L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Transaction on Robotics and Automation*, 12(4):566–580, June 1996.
23. J. J. Kuffner and S. M. LaValle. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 473–479, 1999.
24. J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2000.
25. J. J. Kuffner and S. M. LaValle. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

26. J. J. Kuffner and S. M. LaValle. Rapidly exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions: The Fourth International Workshop on the Algorithmic Foundations of Robotics*, pages 293–308, 2001.
27. A. Ladd and L. Kavraki. A measure theoretic analysis of prm. In *IEEE International Conference on Robotics and Automation*, May 2002.
28. F. Lamiraux and L. Kavraki. Planning paths for elastic objects. *International Journal of Robotics Research*, 20(3), 2001.
29. W. Lickorish. *An Introduction to Knot Theory.* Springer, 1997.
30. T. Ligocki and J. A. Sethian. Recognizing knots using simulated annealing. *Journal of Knot Theory and Its Ramifications*, 3(4):477–495, 1994.
31. J. Phillips, A. Ladd, and L. Kavraki. Simulated knot tying. In *IEEE International Conference on Robotics and Automation*, May 2002.
32. N. Pippenger. Knots in random walks. *Discrete Applied Mathematics*, 25:273–278, 1989.
33. L. Postow, B. Peter, and N. Cozzarelli. Knot what we thought before: The twisted story of replication. *BioEssays*, 21:805–808, 1999.
34. R. Scharein. *Interactive Topological Drawing.* PhD thesis, University of British Columbia, 1988.
35. S. Y. Shaw and J. C. Wang. Knotting of a dna chain during ring closure. *Science*, 260(23):533–536, April 1993.
36. S. Y. Shaw and J. C. Wang. Dna knot formation in aqueous solutions. *Journal of Knot Theory and Its Ramifications*, 3(3):287–298, 1994.
37. J. Simon. Energy functions for polygonal knots. *J. Knot Theory and its Ramif.*, 3:299–320, 1994.
38. G. Song and N. Amato. Using motion planning to study protein folding pathways. In *International Conference on Computational Molecular Biology (RECOMB)*, pages 287–296, April 2001.
39. D. Sumners and S. Whittington. Knots in self-avoiding walks. *Journal Physics A: Mathematicals General*, 21:1689–1694, 1988.
40. M. Teodoro, G. Phillips, and L. Kavraki. A dimensionality reduction approach to modeling protein flexibility. In *International Conference on Computational Molecular Biology (RECOMB)*, April 2002.
41. S. Vassilvitskii, J. Suh, and M. Yim. A complete, local and parallel reconfiguration algorithm for cube style modular robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2002.
42. J. Walter, B. Tsai, and N. Amato. Choosing good paths for fast distributed reconfiguration of hexagonal metamorphic robots. In *IEEE International Conference on Robotics and Automation*, 2002.
43. Y.-Q. Wu. Ming user manual. www.math.uiowa.edu/ ∼wu/ming/ming.pdf, 1996.

Appendix A: Calculating d_{MD}

A line segment is drawn between two points p and q. Equivalently, it is described by a point p and vector $\boldsymbol{v}$ where $\boldsymbol{v} = q - p$. The minimum distance $d_{MD}(\overline{p_1 p_2}, \overline{p_3 p_4})$ between a pair of non-parallel line segments can be calculated by taking expressions over dot products of vectors between the points

$$d_{MD}(\overline{p_1 p_2}, \overline{p_3 p_4}) = d(p_1 + cl(\mu_a)\boldsymbol{u}, p_3 + cl(\mu_b)\boldsymbol{v}$$

where $\boldsymbol{u} = p_2 - p_1$, $\boldsymbol{v} = p_4 - p_3$,

$$\mu_a = \frac{d_{1343} d_{4321} - d_{1321} d_{4343}}{d_{2121} d_{4343} - d_{4321}^2},$$

$$\mu_b = \frac{d_{1343} + \mu_a d_{4321}}{d_{4343}},$$

$$cl(x) = \begin{cases} x < 0 & 0 \\ x > 1 & 1 \\ \text{otherwise} & x \end{cases},$$

and

$$d_{ijkl} = (p_i - p_j) \cdot (p_k - p_l).$$