Chain Reconfiguration

The Ins and Outs, Ups and Downs of Moving Polygons and Polygonal Linkages

Sue Whitesides*

School of Computer Science, McGill University
Montreal, Canada H3A 2A7
sue@cs.mcgill.ca

Abstract. A *polygonal linkage* or *chain* is a sequence of segments of fixed lengths, free to turn about their endpoints, which act as joints. This paper reviews some results in chain reconfiguration and highlights several open problems[1]

We consider a sequence of closed straight line segments $[A_0, A_1]$, $[A_1, A_2]$, ... $[A_{n-1}, A_n]$ of fixed lengths $l_1, l_2, \ldots l_n$, respectively, imagining that these line segments are mechanical objects such as rods, and that their endpoints are joints about which these rods are free to turn. We ask how and whether such a *chain* can be moved from one given configuration to another under various assumptions or "rules of the game". The chain may be confined to the plane throughout its motions; it may be supposed to start and finish in the plane, with motion into 3D allowed for intermediate configurations; its motions in arbitrary dimensional space may be considered. The chain may consist of an open or closed sequence of segments. The links may be allowed or forbidden to cross over or to pass through one another. All of these models are of interest to us.

When the chain consists of a closed sequence of links, we say that the chain is polygonal, or that it is a *polygon*. Consequently, it is natural to use both the language of geometry and mechanics when describing chains. The terms *node*, *vertex* and *joint* are used interchangeably, as are the terms *rod*, *link*, *edge*, and *segment*. The term "polygon" may refer either to a planar object or to a cyclic sequence of links in arbitrary dimension. In case the links are not allowed to intersect except at shared endpoints, we say that the polygon must remain *simple*, i.e., it is not allowed to intersect itself either at rest or during motion.

Polygonal chains are interesting for several reasons. First, there are aesthetic reasons. These very basic objects exhibit surprising behaviors and pose challenging, easily stated questions that arouse our natural curiosity as mathematicians, algorithm designers, and problem solvers. Second, chains can model physical objects such as robot arms and molecules. Here, a word of caution is in order. In

* Research supported by FCAR and NSERC.

[1] A preliminary version of this paper was presented to AWOCA '92, the 12th Australasian Workshop on Combinatorial Algorithms, Lembang, Indonesia, July 14-17, hosted by the Bandung Institute of Technology.

P. Eades and T. Takaoka (Eds.): ISAAC 2001, LNCS 2223, pp. 1–13, 2001.

a "real-world" context, our geometric models are often gross simplifications of complex systems. Mechanical robot arms have mass and inertia, they vibrate, their joints are not universal joints, and they don't have an arbitrary number n of links. In 3D, mechanical links cannot pass through one another, although in 2D, allowing links to pass over one another can model so-called $2\frac{1}{2}$-D situations where long links remain parallel to the plane, joined by short connections not parallel to the plane. For molecules, preferred configurations ("conformations" being the technical term in the chemistry and physics literature) depend on much more than geometry, low energy conformations being the preferred ones. The energy depends on contributions from bonds (modeled by links) which may stretch and dihedral angles (angles between the two planes determined by three consecutive bonds) which may deform. Then there are the chemical interactions between individual pairs of atoms that are not connected by a bond and that may be far from one another in the graph theoretic sense. While it seems unrealistic to suppose that results about purely geometric models are likely to find immediate and wide-spread application in other fields, it seems equally unwise to suppose that geometric studies cannot be relevant or useful. We do not survey application areas, or even potential application areas here, but mention a few pointers: for connections with molecular modelling, see for example [12,13,14,33,34]; for connections with algorithmic motion planning, see for example [24]; for connections with manufacturing, see for example [28]. In later sections, we mention some results on chains that have connections with knot theory [4] and rigidity theory [7,35].

This survey is a personal account, inevitably biased and incomplete. The intent is to focus on the developments in chain reconfiguration in the last several years, highlighting some interesting open problems. For an earlier survey, see [37].

My introduction to the subject began during 1981-82, a year which I spent visiting the Computer Science Department at Cornell University. There, John Hopcroft was working on robotics problems and suggested I read a preliminary version of Schwarz and Sharir, Piano Movers II [34]. Somewhat daunted by the length, and the fact that it was algebraic geometry, I proposed a simple problem as an alternative way to start our discussions, a problem that we later began calling the "ruler folding" problem, or the "carpenter's ruler" problem. This terminology is now used to refer to a variety of chain reconfiguration problems; the original problem (see [15,17]) was the following.

Ruler Folding:
given: a sequence of n positive integer lengths, and a positive integer k;
question: Can a sequence of links of these lengths, hinged at their endpoints, be folded so that they occupy a segment of length at most k? Here, each joint is to be completely straight, or completely folded.

This problem, and the corresponding optimization problem of finding the minimum folding length, make excellent undergraduate student exercises. While the carpenter's ruler is an easy-to-grasp object of study, to determine its properties raises in a simple setting a variety of issues in algorithm analysis and design. Clearly the answer may be determined by trying all the 2^{n-1} ways of folding the

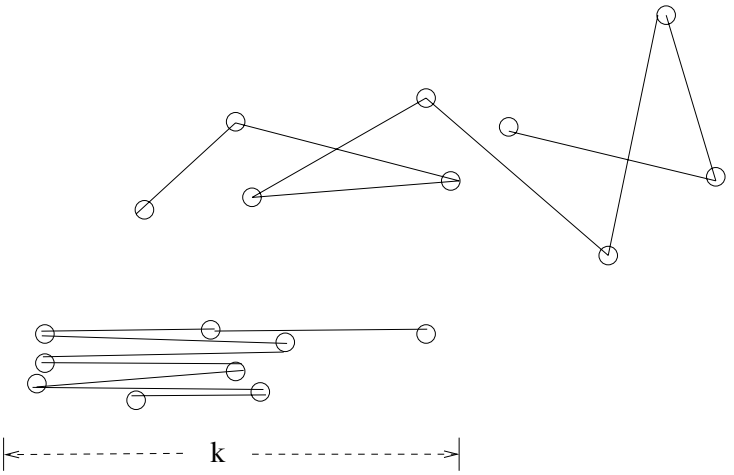

Fig. 1. Ruler Folding

ruler. However, the problem turns out to be NP-complete, by an easy reduction from the Set Partition problem. On the other hand, a simple greedy strategy gives an approximation algorithm to within a factor of 2 for the corresponding optimization version (finding the minimum folding length), and the problem can be solved in $O(n)$ time by dynamic programming when the lengths of the links are bounded above. There is more than one natural way to do this, and designing a second way having seen a first way makes an excellent exercise for students, as does analysing the running time on a Turing machine model of computation when link lengths are unbounded integers.

While the optimization version of the problem can be solved by creating a table or series of tables, some subproblems of a problem instance may not be solved optimally in *any* optimal solution of that instance. Hence the table solution is not, strictly speaking, dynamic programming as it is sometimes described [8]. Finding such examples makes a nice exercise, as does finding examples that show that, for n a power of 2, a simple divide-and-conquer approach doesn't work.

The dynamic programming solutions can be regarded as *fixed parameter tractability* results, in the sense of Downey and Fellows [9]. The idea is to confine the exponential growth in running time to a parameter of the problem that is likely to remain small for typical applications. For Ruler Folding, the dynamic programming solutions allow us to "blame" exponential running time growth on long links; if no links are very long, the solution grows linearly with the length of the input string. To sketch how this goes, let M denote the length of the longest link, let x denote the number of bits in the input data, and let n denote the number of links in the ruler. One dynamic programming method builds a series of roughly M tables, trying all possible folding lengths in the range M to $2M$. Each table has $O(n)$ rows (one for each joint) and $O(M)$ columns (one for

each possible integer coordinate for a joint in the range 0 to the folding length being tried). Computing each table entry takes $O(lgM)$ bit operations, giving an overall running time of $O(xM^2lgM)$. Note that the integer M is no greater than 2^x, since M must be given by at most x bits. Hence the running time is indeed exponential in x. Now consider all rulers that have n links but no link longer than 1000, say. For this restricted set of very reasonable instances, there is a running time upper bound that is linear in the length of the input x (and linear in the number n of links). Thus the exponential behaviour of the algorithm is the "fault" of the possibility of very long links in the general case.

The NP-completeness of Ruler Folding has some consequences for the complexity of related problems, such as determining whether or not a polygonal chain (allowed to cross or not) can be moved from one configuration to another in an environment containing polygonal obstacles. Here, one may design an instance containing a narrow gap such that the chain must folded, or nearly so, into length at most some given amount in order to fit through a narrow passage (see [15]). Furthermore, the NP-completeness of Ruler Folding can be used to show the hardness of the placement of graphs, even trees, having edges of specified lengths so that certain vertices are placed at certain points (see [39,37,38]). This type of problem is sometimes stated in terms of the realizability of distance matrices, where the entries in the matrix give the desired Euclidean distances between certain pairs of vertices in the graph.

In our first investigations of movement properties of chains, we required one end of the chain to be fixed to the plane, and we allowed links to cross over one another as this so-called "arm" moved. Determining what points of the plane can be reached by the opposite end of the chain makes a nice easy exercise for students, as does proving that any point that can be reached at all can be reached by a configuration of the arm that has at most two non-straight joints. This can be seen by thinking in terms of the polar coordinates of the point to be reached relative to an origin at the fixed point of the arm: first find a way to achieve the correct distance between the free end and the fixed end, and then rotate the arm about the origin to move the tip to the desired point. Textbooks that have included some of these problems as exercises include [21,27].

Problems for the reconfiguration of a chain (or an *arm*, as we call a chain with the location of one end fixed) get decidedly more interesting in the presence of obstacles or other constraints. Our first effort to deal with obstacles in the "workspace" of a robot "arm" was to consider the problem of moving an arm confined to a closed disk from one given configuration to another. Links were allowed to cross, and we found a polynomial time algorithm for detrmining if the desired configuration could be reached from the initial one, and, when this was possible, for designing a specific motion. Here we made a distinction between the running time for computing the motion, and the length of the description of the motion, which we gave in terms of simple motion primitives. These primitives were described, for example, in terms of rotating a link about a fixed endpoint while other joints were fixed in position or joint angles were frozen and dragged along. The basic idea was simple: find a way to move the arm to a "canonical"

position in which all joints at and beyond some joint A_i are placed on the boundary of the circle, regard the initial portion of the chain as a kind of leash whose length can be adjusted by adding or removing links, and then, by adjusting the length of the leash, rotate the remainder of the chain around the circle. A sequence of links in the tail can then, one hopes, be formed and lifted up to reach points in the disk. Kantabutra and Kosaraju [20] pushed this technique and improved the running time of our algorithm.

We studied chains and arms in other confining regions such as convex polygons. We were not able to answer the reconfiguration question even for a triangle, a problem that remains open.

Problem (chain in polygon): Given two configurations of a chain confined to a convex polygonal region, with edge crossings allowed, determine whether the chain can be moved inside the polygon from one configuration to the other.

Kantabutra then went on to explore the use of this general strategy inside other shapes, in particular, a square [18,19]. Here, the links do not rotate around the boundary so conveniently as is the case for the circle. He was able to obtain reconfiguration and reachability results for arms and chains satisfying a bound on the link lengths in terms of the length of a side of the square.

Inspired by Kantabutra's success with squares, the case of arms and chains confined inside triangles seemed interesting to try again. In 1991, Peter Eades and I tried an even simpler version of this case: an arm confined to a wedge. Here, we asked whether and how an arm could be straighted, with links allowed to cross, and were able to solve this problem when the internal wedge angle is $\Pi/2$ or greater (the proof, which remains unpublished, was given at AWOCA '92). The case of the acute wedge still remains unsolved. Perhaps the problem is NP-hard. The version of the problem in which links are not allowed to cross may also be interesting.

Problem (arm in a wedge): Given a chain with one extreme end fixed to the plane, and confined to move inside a wedge whose vertex angle is less than $\Pi/2$, design an algorithm to decide whether the arm can be straightened, and to move it to such a position when this is possible.

In March 1991, Bill Lenhart came to visit McGill, and we became fascinated with a different problem, which we came to call "turning a polygon inside-out". We decided to drop the idea of exploring the motion of chains or arms confined inside polygonal environments. Instead, we would get rid of the obstacles, but pin down both the endpoints. We quickly noticed that the motions of a chain with both endpoints fixed down correspond to the motions of a closed polygonal linkage.

When this linkage takes the form of a *simple* polygon, it has a natural orientation, as traversing the boundary in the clockwise sense visits the vertices either in increasing or decreasing order of their indices. Obviously, a triangle whose vertices are indexed cannot be moved in the plane so that its orientation changes (disallowing flips out of the plane). Hence the inside-out problem: given a simple polygon lying in the plane, when can it be moved in the plane to its

mirror image? More generally, given two configurations of a polygonal linkage in the plane, when can the linkage be moved from the one configuration to the other? We called two configurations *equivalent* if it is possible to move the linkage between the two configurations, and asked for the number of equivalence classes of polygonal linkages in the plane. The answer turned out to be pretty. The number of classes is either two or one, depending on whether the second and third largest link lengths sum to more than the length of the longest link. Thus a triangle has two equivalence classes, as its longest side has length less than the sum of the lengths of the other two sides. Furthermore, as is the case for the triangle, each configuration in one class has a mirror image in the other class. We designed algorithms for reconfiguring when possible, and eventually noticed that our reconfiguration strategy in 2D always worked in dimension 3 and above, so we obtained a little bonus: polygonal linkages have one equivalence class of configurations in dimension 3 and above.

One of our discussions about unconfined chains led us to some interesting reconfiguration problems for chains *not* allowed to intersect themselves. And what about 3D? We had a simple plan: project the 3D chain to 2D, and reconfigure the shadow in 2D to guide the reconfiguration of the 3D object. There were some difficulties with this, however. Suppose you tie a knot in your shoe lace, and attach a long knitting needle to each end of the lace. You won't be able to undo the "knot", even though it's not a knot in the mathematical sense. It's easy to imagine that a chain of links could be configured like such a knot, with very long links attached at the ends, to give a configuration of an open chain of links in 3D that cannot be straightened. Worse, we realized that even if a chain in 3D had a simple projection onto a plane, we didn't see how to straighten a simple chain in the plane. This led us to several chain straightening and polygon convexifying problems for linkages in the plane, with links not allowed to cross. My colleague Godfried Toussaint at McGill was very enthusiastic about these problems and suggested that we try to convexify *star-shaped* polygons, that is, polygons containing a non-empty "kernel" of points that can "see" all the points in the polygon.

Encouraged by Godfried's enthusiasm, we began describing these problems at every opportunity, beginning with Bill's seminar at McGill in March 1991, and later in August 1991 in our talk at the Canadian Computational Geometry Conference (CCCG '91) in Vancouver, for example, and at my AWOCA '92 sessions, where the chain straightening problem was on the handout of problems given in the problem session, and again in our talk at CCCG '92. We also described these problems in our 1993 McGill technical report. When our journal article based on turning a polygon inside-out ([25]) finally appeared in 1995, we mentioned at the very end that these problems still had not been solved.

Bill and I were not aware in 1991 that this kind of problem had been posed by topologists in the 1970's (see the discussion of the history of the problem in [7]). However, as far as we know, our 1993 McGill University technical report [25] was the first written, publicly accessible statement of these problems. In the computational geometry community, the chain-straightening problem was again

independently rediscovered by Joe Mitchell in 1992, in the context of a tube manufacturability problem. Joe was active in generating possibilities of chains that might not be straightenable in the plane, and discussed these with a number of geometers.

In the spirit of Paul Erdős, I offered a prize of one bottle of Bintang, the local beer, for AWOCA '92 participants who solved the open problems about linkages on the handout. One of the participants who took this offer seriously was Heiko Schröder, who was then at the University of Queensland, where I visited later, in December 1992. In fact, I rented Heiko's apartment, as he was on sabbatical, travelling in Europe. We started corresponding via the fax machine in his apartment about the star-shaped polygon problem. He would propose an idea and fax it to me, with pictures. I would fax back a reply. The difference in time zones made life interesting. Sometimes I would hear the fax machine grinding away at 3 a.m. Brisbane time, and leap out of bed to see what Heiko's latest idea was for winning a Bintang. Always there was some little problem. We tried hard to do a proof by induction: straighten some joint and continue on a simple polygon with fewer vertices. We tried drawing rays through every other vertex, hoping to move some pair of rays apart so that the 2-link sub-chain contained in the wedge bounded by the rays would straighten. All these efforts led to technical difficulties, such as degenerate cases to handle, or problems keeping the chain simple or the polygon star-shaped.

Eventually, Heiko proposed moving the vertices on the rays simultaneously outward along their rays at constant speed. This was quite a novel idea, since it moved an unbounded number of vertices at the same time and had a distinctly different flavor than that of chain reconfiguration algorithms whose motion primitives were localized. Surely such an expansive motion would convexify the linkage, and it had the intuitive appeal that the algorithm sort of "inflated" the polygon. We outlined a formal proof, whose details we never completed. Still, what we had seemed fairly convincing, and I promised Heiko a Bintang.

In 1992-93, Mark van Kreveld of the University of Utrecht, The Netherlands, came to McGill for a postdoc. He sportingly took a look at some chain reconfiguration problems in the plane, with links allowed to cross; in particular, he looked at problems for chains confined to polygons and/or the wedge problem. When these seemingly simple problems proved surprisingly difficult, he called on a heavy-duty weapon, Jack Snoeyink, then of the University of British Columbia. Mark and Jack proposed a new, related problem, that of folding a chain whose links all have the same length, an *equilateral chain*. If one could fold the entire chain onto a single link, one could then hope to move this link around in a containing environment to a position where it could be unfolded to give the desired configuration for the whole chain. Thus, instead of using a completely straightened configuration as an intermediate configuration between initial and desired final configurations, one would use a completely folded intermediate configuration, which seemed sensible in the context of a containing environment. Studying a polygonal environment consisting of an equilateral triangle of side length 1, we found a surprising alternation property for the foldability of equilateral chains

of link-length $l < 1$. Whether or not *every* configuration of such a chain can be folded to a single link is a property that changes *three* times as l increases from 0 to 1. For very small links with l close to 0, every configuration can be folded to a segment of length l, and for chains with l close to 1, not all configurations can be folded. Surprisingly, as l increases from 0 to 1, the foldability of equilateral n-link chains changes from always foldable, to not always foldable, to always foldable, and finally back to not always foldable. See [22] and for another example of alternation, [30].

Recalling Kantabutra's success with reconfiguring chains in squares, my student Naixan Pei and I decided to push farther the strategy of moving the chain to the boundary, and rotating it around the boundary to a position that would enable one end to reach out to touch a desired point. We were able to generalize some of Kantabutra's results to the case of *convex obtuse polygons*. These are convex polygons, not necessarily regular, such that each internal vertex angle is equal to or greater than $\Pi/2$. See [29,30,31,32].

In 1998, Anna Lubiw of the University of Waterloo and I led a small workshop on "Wrapping and Folding" (or, alternatively, "Unwrapping and Unfolding") at McGill's Bellairs Research Institute in Barbados. She and her collaborators, including Joe O'Rourke and Erik Demaine, who was her doctoral student, had obtained some interesting results having a flavor of origami. Her view was that chain reconfiguration was a kind of origami for a lower dimensional object, a line instead of a piece of paper. At this workshop, we revisited chain reconfiguration problems, this time insisting that links not be allowed to cross.

Our first result [1], produced by all the workshop participants in a real group effort, was inspired by a chain configuration that Joe Mitchell had proposed as being possibly unstraightenable in the plane. It seemed to us that this particular example could, however, be straightened, so instead, we explored the possibility that a *tree linkage* based on Joe's chain pattern could not be straightened. Here, straightening a tree means to choose some node as a root and then to move the linkage so that all root-leaf paths form essentially straight lines emanating from the root. Indeed, we found that there are tree linkages that cannot be straightened in the plane and that can exhibit exponentially many equivalence classes of configurations [1]. Here are some questions arising from this work.

Problems (tree linkages): What is the complexity of deciding whether or not a tree configuration can be straightened? Design an algorithm to do this when possible. Can every configuration of a tree linkage whose links all have length 1 be straightened in the plane?

Another idea that our 1998 Folding and Wrapping workshop explored was a line of thought suggested by Godfried Toussaint: suppose the configuration of a chain initially lies in the plane, but that to straighten it without edge crossings, we allow ourselves to lift it into 3D. Eventually, the workshop designed an algorithm that convexifies planar polygons, with intersections forbidden, by lifting one link after another to a subchain that forms a convex arch and that lies in a plane parallel to the original plane, joined to it by a connecting link at each end of the arch. We came to call this the St. Louis arch algorithm, since

the idea of storing the partial solution out of the way of the unprocessed links reminded us of the huge arch that towers over the city of St. Louis, Missouri. For this and other chain straightening and convexification results obtained by this workshop, see [2].

One of the workshop participants, Ileana Streinu, suggested an intriguing approach to the polygon convexification problem. As long as we were willing to use 3D to move an initially flat polygonal linkage to a convex shape, why not simply "flip out" the pockets? A *pocket* of a polygon is a connected component of the convex hull of the polygon with the polygon itself removed. The pockets are thus polygons whose boundaries consist of one convex hull edge $[v_i, v_j]$ together with one of the two chains of edges of the polygon between v_i and v_j. To "flip out" a pocket, one would rotate the pocket about the convex hull edge, and return it to the plane outside the original convex hull. Clearly the entire polygonal linkage would not intersect itself during or after this flipping motion, since the links in the pocket would land outside the convex hull of the remaining links. There are some dfficulties to work out. If all the pockets are flipped at the same time, they may intersect one another. Also, the polygon formed by flipping out a pocket is not in general convex, so the process must be continued; it is not even clear it terminates. We eventually found that the process *does* terminate after a finite number of flips, but that this number cannot be bounded by n. Worse yet, we eventually found that Ileana had independently rediscovered a question posed by Erdős in the 1930's and answered by several people in the mean time.

Some of the workshop participants revisited the problem of convexifying a star-shaped polygon and worked out the details and special cases [11]. The workshop also revisited the knitting needles example that Bill and I had suggested as evidence that chains cannot always be straightened in 3D; a concrete example was made and a simple proof of its nonstraightenability was given (see [2]).

The 1998 workshop served to kindle a lot of interest on reconfiguration problems for chains whose links are not allowed to cross. One of the participants, Joe O'Rourke, together with his student R. Cocan, proved that every polygonal linkage in dimension D $>$ 3 can be convexified [6]. For chains whose links are allowed to cross in dimension D $>$ 2, Bill Lenhart and I had proved this as a by-product of the our techniques for turning a polygon inside-out in the plane.

A subset of participants, together with Michael Soss, a student of Toussaint, found a convexification procedure for monotone polygons in the plane, whose links are not allowed to cross [3].

Toussaint has written a history of Erdős' pocket-flipping problem and its solutions and has given a proof combining elements of various ones of these solutions [36]. He has also led his own workshops on various aspects of motion planning for polygonal linkages, which became the topic of Michael Soss's Ph.D. thesis at McGill.

A fascinating recent development in the area is the following. One of the 1998 workshop participants, Eric Demaine, together with Robert Connelly and Günter Rote has answered the planar chain straightening problem in the affirmative: every *simple* configuration of an open chain (or a closed polygon) in the plane

can be straightened (or convexified) in the plane while keeping the linkage *simple* during the entire motion. See [7]. Their proof uses techniques from rigidity theory and linear programming, combined. They inflate the polygon by moving vertices so that the distance between non-adjacent vertices never decreases. Meanwhile, another one of the 1998 workshop participants, Ileana Streinu, has come up with an elegant, more "concrete" proof method [35], based on the notion of pseudo-triangulations. The flavour is discrete, combinatorial, and mechanical.

Problems (non-crossing planar chain straightening and polygon convexifying): With edges not allowed to cross, are there more simple ways to straighten a chain or convexify a polygon in the plane? Here, one may consider special classes of polygons, as done in [11,3].

Of course, tastes vary about what constitutes a "simple method". Note that there is a distinction to be made between the running time of an algorithm that computes the description of a motion, and number of "mechanical steps" one makes in physically carrying out a motion, or the length of the description of the motion. Then there is the question of practical implementability, both electronic and mechanical. Another consideration is whether to allow more than a constant number of joints to be active at the same time, where "active" refers to a change in the angle between two adjacent links or a change in the dihedral angle between the two planes determined three consecutive links.

Problems (3D non-crossing chain straightening and polygon convexification): In 3D, with edges not allowed to cross, when can an open or closed chain be straightened or convexified? What is the complexity of this problem? Are there interesting, nontrivial special situations for which a convexifying strategy be given? (One of the papers from the 1998 workshop [2] gives some easy examples.)

Problems (toleranced reconfiguration): What can be said about reconfiguring chains with a clearance constraint? For example, suppose that one must move a chain from one configuration to another while respecting a safety zone of some fixed radius around each link?

Finally, as mentioned earlier, chains and polygons are special cases of tree-like linkages, which themselves are special cases of graph-like linkages. For trees, even in the plane with edges allowed to cross, it is NP-hard to decide if a tree-linkage can be positioned so that its leaves are located at given points in the plane. See [37] and, for algorithms for placing trees see [38,39].

Problems (tree-like linkages): What can be said about the configurations and motions of tree-like (and more generally, graph-like) linkages?

In view of the piece-wise linear knot theoretic flavor of these problems, and their connection with rigidity theory, the subject of linkage reconfiguration offers much fertile ground to be explored.

References

1. T. Biedl, E. Demaine, M. Demaine, S. Lazard, A. Lubiw, J. O'Rourke, S. Robbins, I. Streinu, G. Toussaint and S. Whitesides. On reconfiguring tree linkages: trees can lock. Accepted in *Discrete Applied Math.*, Feb. 2001, to appear; conference abstract in Proc. of the 10th Canadian Conf. on Computational Geometry CCCG '98, McGill University, Montreal, Canada, Aug. 10-12, 1998, pp. 4-5. 8
2. T. Biedl, E. Demaine, M. Demaine, S. Lazard, A. Lubiw, J. O'Rourke, M. Overmars, S. Robbins, I. Streinu, G. Toussaint, and S. Whitesides. Locked and unlocked polygonal chains in 3D. Accepted in *Discrete and Computation Geom.*, May, 2001, to appear; conference abstract in Proc. of the 10th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA), Baltimore MD, USA, Jan. 1999, pp. 866-867. 9, 10
3. T. Biedl, E. Demaine, S. Lazard, S. Robbins, and M. Soss. Convexifying monotone polygons. Proc. of the 10th Annual International Symp. on Algorithms and Computation (ISAAC'99), Chennai, India, Dec. 16-18, 1999, Springer-Verlag Lecture Notes in Computer Science, pp. 415-424. 9, 10
4. J. Cantarella and H. Johnston. Nontrivial embeddings of polygonal intervals and unknots in 3-space. *J. of Knot Theory and its Ramifications*, vol. 7 (8), pp. 1027-1039, 1998. 2
5. A. Cauchy. Sur les polygones et les polyèdres, seconde mémoire. *Journal Ecole Polytechnique*, vol. 16 (9); pp. 26-38, 1813.
6. R. Cocan and J. O'Rourke. Polygonal chains cannot lock in 4D. Proc. 11th Canadian Conf. on Computational Geometry (CCCG), 1999. 9
7. R. Connelly, E. Demaine, G. Rote. Straightening polygonal arcs and convexifying polygonal cycles. Proc. of the 41st IEEE Symp. on Foundations of Computer Sciences (FOCS), 2000, pp. 432-442. 2, 6, 10
8. T. Cormen, C. Leiserson, and R. Rivest. Introduction to Algorithms. Undergraduate textbook, MIT Press and McGraw Hill, 1990. 3
9. R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer-Verlag, New York, 1999. 3
10. Paul Erdős. Problem no. 3763. *American Mathematical Monthly*, vol. 42, p. 627, 1935.
11. H. Everett, S. Lazard, S. Robbins, H. Schröder and S. Whitesides. Convexifying star-shaped polygons. Proc. of the 10th Canadian Conf. on Computational Geometry CCCG '98, McGill University, Montreal, Canada, Aug. 10-12, 1998, pp. 2-3. 9, 10
12. P. Finn, D. Halperin, L. Kavraki; J-C. Latombe; R. Motwani; C. Shelton, and S. Venkatasubramanian. Geometric manipulation of flexible ligands. *Applied Computational Geometry*, Springer-Verlag, pp. 67-78, 1996. 2
13. Aviezri Frankel. Complexity of protein folding. *Bulletin of Mathematical Biology*, vol. 55 (6), pp. 1199-1210, 1993. 2
14. Maxim Frank-Kamenetskii. Unravelling DNA. Addison-Wesley, 1997. 2
15. J. Hopcroft, D. Joseph, and S. Whitesides. On the movement of robotic arms in 2-dimensional bounded regions. *SIAM J. on Computing* vol. 14, May 1985, pp. 315-333. 2, 4
16. J. Hopcroft, D. Joseph, and S. Whitesides. Movement problems for 2-dimensional linkages. *SIAM J. on Computing* vol. 13, Aug. 1984, pp. 610-629.
17. J. Hopcroft, D. Joseph and S. Whitesides. On the movement of robot arms in two-dimensional bounded regions. Proc. of the IEEE 23rd Annual Symp. on the Foundations of Computer Science (FOCS), Chicago IL, USA, Nov. 3-5, 1982, pp. 280-289. 2

18. V. Kantabutra. Motions of a short-linked robot arm in a square. *Discrete Comput. Geom.* vol. 7, 1992, pp. 69-76. 5
19. Vitit Kantabutra. Reaching a point with an unanchored robot arm in a square. *Int. J. of Computational Geometry and Applications* vol. 7 (6), pp. 539-550, 1997. 5
20. V. Kantabutra and R. Kosaraju. New algorithms for multilink robot arms. *J. Comput. System Sci.*, vol. 32, pp. 136-153, 1986. 5
21. Dexter Kozen. The Design and Analysis of Algorithms. Graduate textbook, Springer-Verlag, 1992. 4
22. M. van Kreveld, J. Snoeyink and S. Whitesides. Folding rulers inside triangles. *Discrete and Computational Geometry* vol. 15, 1996, pp. 265-285; conference abstract in Proc. of the 5th Candadian Conf. on Computational Geometry, Queen's U., Kingston, Canada, Aug. 5-10, 1993, pp. 1-6. 8
23. J. Kutcher. Coordinated Motion Planning of Planar Linkages. Ph.D. thesis, John Hopkins U., 1992.
24. Jean-Claude Latombe. Robot Motion Planning. Kluwer Academic Publishers, Boston, 1991. 2
25. W. Lenhart and S. Whitesides. Reconfiguring closed polygonal chains in Euclidean d-space. *Discrete and Computational Geometry*, vol. 13, 1995, pp. 123-140; conference abstracts in Proc. of the 3rd Canadian Conf. on Computational Geometry, Vancouver, Canada, Aug. 6-10, 1991, pp. 66-69 ("Turning a Polygon Inside-out"), and in Proc. of the 4th Canadian Conf. on Computational Geometry, St. John's, Newfoundland, Canada, Aug. 10-14, 1992, pp. 198-203 ("Reconfiguring with Linetracking Motions"); see also Reconfiguring Simple Polygons, technical report, McGill University, School of Computer Science SOCS-93.3, 1993. 6
26. A. Lubiw and J. O'Rourke. When can a polygon fold to a polytope? Technical Report 048, Dept. of Computer Science, Smith College, June 1996.
27. Joseph O'Rourke. Chapter 8.6, Computational Geometry in C. Cambridge University Press, 1998. 4
28. J. O'Rourke. Folding and unfolding in computational geometry. Proc. Japan Conf. Discrete Comput. Geom., Dec. 1998, LNCS vol. 1763, pp. 258-266, 1999. 2
29. Naixun Pei. On the Reconfiguration and Reachability of Chains. Ph.D. thesis, School of Computer Science, McGill U., 1996. 8
30. N. Pei and S. Whitesides. On folding rulers in regular polygons. Proc. of the 9th Canadian Conf. on Computational Geometry CCCG '97, Queen's University, Kingston, Ontario, Canada, Aug. 11-14, 1997, pp. 11-16. 8
31. N. Pei and S. Whitesides. On the reachable regions of chains. Proc. of the 8th Canadian Conf. on Computational Geometry CCCG '96, Carleton University, Ottawa, Ontario, Canada, Aug. 12-15, 1996, pp. 161-166. 8
32. S. Whitesides and N. Pei. On the reconfiguration of chains. Computing and Combinatorics, Proc. of the 2nd Annual International Conf., COCOON '96, Hong Kong, June 17-19, 1996, J-Y Cai and C-K Wong, *eds.*, Springer-Verlag Lecture Notes in Computer Science LNCS vol. 1090, pp. 381-390. 8
33. Micha Sharir. Algorithmic motion planning. J. E. Goodman and J. O'Rourke, eds., Handbook of Discrete and Computational Geometry, chapter 40, pp. 733-754, CRC Press, Boca Raton FL, 1997. 2
34. J. Schwartz and M. Sharir. On the "piano mover's" problem, II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Math.* vol. 4, pp. 298-351, 1983. 2

35. Ileana Streinu A combinatorial approach to planar non-colliding robot arm mtion planning. Proc. of the 41st IEEE Symp. on Foundations of Computer Sciences (FOCS), 2000, pp. 443-453. 2, 10
36. Godfried Toussaint. The Erdős-Nagy theorem and its ramifications. Proc. 11th Canadian Conf. on Computational Geometry, Vancouver, Aug. 1999. 9
37. Sue Whitesides. Algorithmic issues in the geometry of planar linkage movement. *Australian Computer Journal*, vol. 24 (2), pp. 42-50, 1992. 2, 4, 10
38. S. Whitesides and R. Zhao. Algorithmic and complexity results for drawing Euclidean trees. Advanced Visual Interfaces, Proc. of the International Workshop AVI '92, Rome, Italy, May 25-29, 1992, T. Catarci, M. F. Costabile, and S. Levialdi, *eds.*. World Scientific Series in Computer Science vol. 36, 1992, pp. 395-410. 4, 10
39. Rongyao Zhao. Placements of Euclidean Trees. Ph. D. thesis, School of Computer Science, McGill U., 1990. 4, 10