

How to Draw a Straight Line Using a GP: Benchmarking Evolutionary Design Against 19th Century Kinematic Synthesis

Hod Lipson

Computational Synthesis Laboratory,
Mechanical & Aerospace Engineering, and Computing & Information Science,
Cornell University, Ithaca NY 14850, USA

hod.lipson@cornell.edu

Abstract. This paper discusses the application of genetic programming to the synthesis of compound 2D kinematic mechanisms, and benchmarks the results against one of the classical kinematic challenges of 19th century mechanical design. Considerations for selecting a representation for mechanism design are presented, and a number of human-competitive inventions are shown.

1 Introduction

Kinematics – *the science of pure motion* – is concerned with the analysis and synthesis of mechanisms composed of connected rigid elements. It deals with the relative geometric displacements of points and links of a mechanism, without regards to forces that generate those displacements or the physical embodiment that realizes them.

The interest in kinematics has its origins in machines as old as civilization [16], but was largely invigorated in the 18th century with the invention of the steam engine and the beginnings of the industrial age [4]. Initially, designs were produced and analyzed by practitioners in an *ad-hoc* manner, but the pressure for rigorous and systematic performance led, within a few generations, to the establishment of increasingly general methods for geometric *analysis* and classification of mechanism types [17]. Many of these ideas form the basis of modern kinematic theory today. Kinematic *synthesis*, however, is still largely a challenge.

The systematic synthesis of a mechanism for a given purpose is a long-standing problem, and perhaps one of the earliest general synthesis problems to be posed. Robert Willis, a professor of natural and experimental philosophy at Cambridge, wrote in his 1841 book *The Principles of Mechanisms* [20]:

[A rational approach to synthesis is needed] to obtain, by direct and certain methods, all the forms and arrangements that are applicable to the desired purpose. At present, questions of this kind can only be solved by that species of intuition that which long familiarity with the subject usually confers upon

experienced persons, but which they are totally unable to communicate to others. When the mind of a mechanic is occupied with the contrivance of a machine, he must wait until, in the midst of his meditations, some happy combination presents itself to his mind which may answer his purpose."

Robert Willis, *The Principles of Mechanisms* [20]

Almost two centuries later, a rational method for the synthesis of mechanisms is still not clear. Despite great advances in analysis of mechanisms and classification of elementary components, founders of modern kinematic theory wrote "*While we may talk about kinematic synthesis, ... we really are talking about a hope for the future than a great reality of the present*" [6]. Analytical methods do exist for some special cases of mechanisms (such as serial articulated joints, or certain parallel mechanisms), but not for the general case. Mathematical proofs of existence show that mechanisms can be found to trace any algebraic curve, but their construction is usually impractical. The question of rational synthesis of mechanisms is today of increasing importance with the quest for design automation; when seeking computational synthesis methods, no longer can we cloak the design process with the term 'creativity'.

A number of recent works have used evolutionary computation techniques to automate mechanism design, typically in conjunction with designing a controller. The mechanisms designed were usually serial or tree-like [18,9,7,2], though some of our prior work focused on design of compound mechanisms containing multiple, entangled kinematic loops [12]. Nevertheless, the capabilities of kinematic synthesis automation and suitable representations remain largely unexplored.

The goal of this paper is twofold: To explore some representations for kinematic synthesis using genetic programming (GP), and to benchmark the performance of these algorithms against a well established kinematic design problem that has baffled some of the world's greatest inventors for nearly a century: The *straight line* problem.

2 Mechanism Representations

A kinematic mechanism can be represented as a graph, embedded in two or three dimensions. Edges of the graph represent links, and nodes of the graph represent joints. There are a number of different types of joints (e.g. prismatic or rotary in 2D, ball, prismatic, cylindrical, or screw, in 3D), and a number of different types of links (with different geometries and numbers of attachment points). Here we focus on planar mechanisms with free joints. A 2D mechanism composed of straight links and free joints is directly represented by a graph embedded in the plane.

An important concept in the description of a mechanism is its number of degrees of freedom (DoF). Some of the mechanisms' nodes may be grounded, and therefore immobile, while other nodes are free to move while being constrained by links attaching them to other nodes. The overall number of independent parameters needed to fully specify the state of the entire mechanism is its number of DoF.

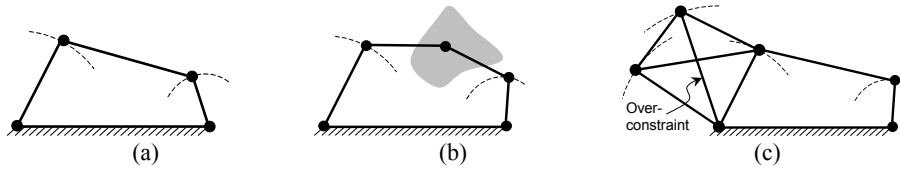


Fig. 1. Degrees of freedom of a mechanism: (a) A four-bar mechanism has 1 DoF and some of its nodes trace curves, (b) A five-bar mechanism has two DoF and some of its node can trace over an area, but (c) some structures are overlapping constraints or have degeneracies that lead to miscalculation of their number of DoF; this structure should be locked, but is free.

The number of degrees of freedom of a mechanism can be calculated directly by considering the fact that each node of the mechanism has two DoF of motion in the plane, and each link eliminates one of these DoF by providing one constraint on the distance between two nodes. The entire mechanism also has three rigid-body DoF that can be eliminated by grounding any one of the links. The total DoF of a grounded mechanism with n nodes and m links is thus $2n - m - 3$. A grounded four-bar linkage (Figure 1a) for example, has exactly one degree of freedom, and its nodes will therefore trace curves. A five-bar mechanism (Figure 1b) has two degrees of freedom, and some of its nodes will trace (fill) areas. There may, however, be mechanisms that are over-constrained in some part and under-constrained in another, leading to misleading total DoF count (Figure 1c). Other mechanisms may have geometrical singularities and degeneracies in their configurations that cause locking or unaccounted free motions. It is therefore impossible to predict the DoF that a general mechanism may have based solely on topological counting arguments.

Evolutionary algorithms progress by modifying the mechanism's graph directly, or may use an indirect encoding (genotype) from which the graph (phenotype) is constructed. Luke and Spector [13] survey a number of representations used to describe or 'grow' computational graphs, such as neural networks. Some methods use context free grammars, L-systems, and parse trees operating on nodes and edges [e.g. 5,1].

Most of the existing representations for encoding networks generate highly connected architectures that are suitable for computational networks, but which are less suitable for kinematic networks because they over-constrain the motion and create deadlocked mechanisms. Using these representations, the likelihood of generating a mechanism with exactly one degree of freedom is vanishingly small. In order to allow an evolutionary algorithm to explore the space of one-DoF mechanisms more efficiently, a more suitable representation is required. This representation must have a tree-like architecture to be used by a GP.

A second consideration in the choice of representation is that of evolvability. Many of the representations cited above result in context-sensitive and order-sensitive description of a network. For example, the structure generated by a branch in Gruau's cellular encoding depends on whether it is parsed before or after its sibling branch. If that branch is transplanted by crossover into another tree it may produce an entirely different structure. Such behavior hampers the effectiveness of recombinative operators by precluding the formation of modular components that are discovered by the search in one place and then reused elsewhere. A representation where the structure produced by a branch of the tree is minimally affected by its context may thus be more evolvable.

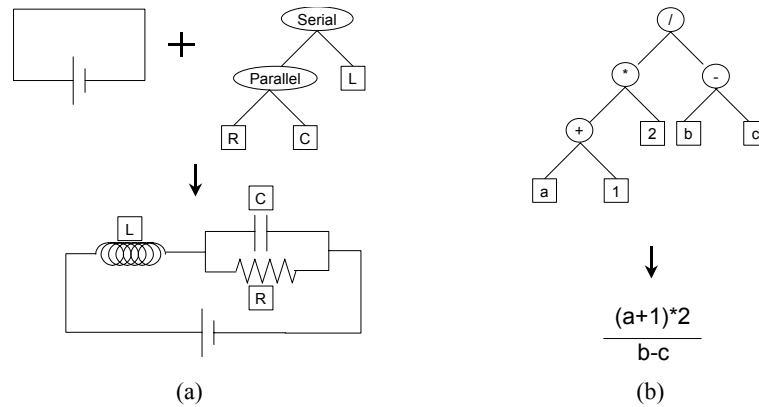


Fig. 2. Top-down and bottom-up parse-tree constructions: (a) Top-down construction of a circuit (b) bottom-up construction of a symbolic expression.

Top-down and bottom-up tree representations of mechanisms

Tree-based representations can describe a set of operations to construct a phenotype in a top-down or bottom-up manner. A top-down representation starts with an initial structure (an *embryo*) and specifies a sequence of operations that progressively modify it into its final form. Figure 2a shows a top-down tree that specifies the construction of an electric circuit, starting with an initial circuit and recursively replacing circuit segments with serial and parallel arrangements of electrical components [10]. Each node of the tree is either an operator that modified the circuit and passes segments to its child nodes, or a terminal electrical component. The specific parallel and serial operators cannot be used for construction of mechanisms as they will immediately create over- and under-constrained kinematic chains. Because of the physics of electric circuits, ordering of children under a parent does not matter. This tree is thus both order independent and context independent. In a top-down tree, parent nodes must be constructed before their children.

Figure 2b shows a bottom-up construction of a symbolic expression. Here terminal nodes represent constants or variables, and parent nodes represent mathematical operators. Because of the nature of mathematical expressions, parsing order is important, and swapping order of some child nodes would result in a mathematically different expression. The terms are unchanged, however, by the content of their siblings. This tree is thus order dependent but context independent. In a bottom-up tree, child nodes must be constructed before their parents.

Two tree-based representations for describing kinematic mechanisms are proposed here. Top-down construction of a mechanism starts with an embryonic one-DoF kinematic basis such as the four-bar mechanism shown in Figure 3a. A tree of operators then recursively modifies that mechanism by replacing single links (DoF = -1) with assemblies of links with an equivalent DoF, so that the total number of DoF remains unchanged. Two such transformations are shown in Figure 3b: The **D** and **T** operators.

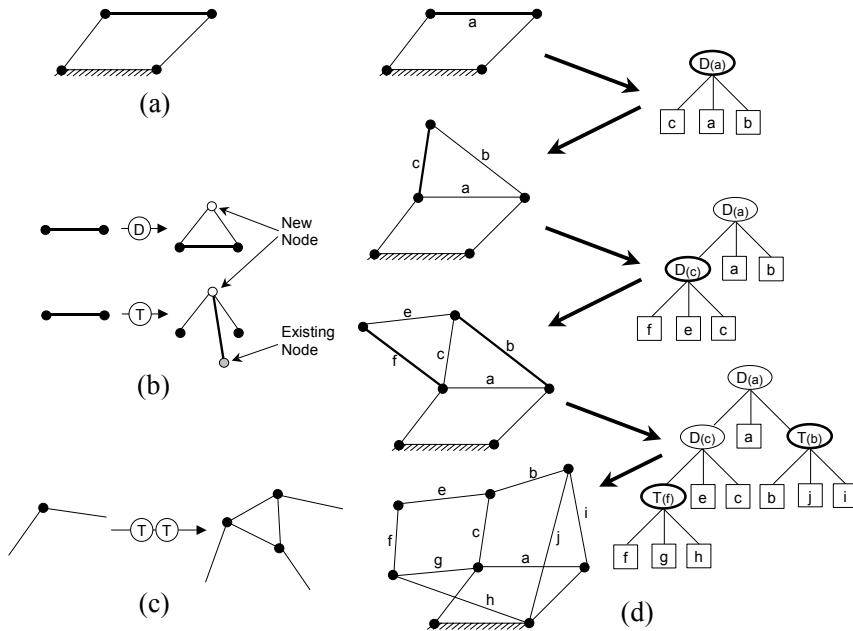


Fig. 3. Top-down construction of a one-DoF mechanism: (a) An embryonic four-bar mechanism; (b) two operators that change local topology but do not change the number of degrees of freedom; (c) operators applied in some sequence will create new mechanism, such as transform a dyad into a tryad; (d) operators can be applied in a tree to transform the embryonic mechanism into an arbitrary mechanisms while retaining the original number of DoF.

The D operator creates a new node and connects it to both the endpoints of a given link, essentially creating a rigid triangular component. The T operator replaces a given link with two links that pass through a newly created node. The new node is also connected to some other existing node. In both operators, the position of the new nodes is specified in coordinates local to the link being modified. The T operator specified the external connecting node by providing coordinates relative to link being modified; the closest available node from the parent structure is used. This form of specification helps assure the operators remain as context and order independent as possible. Figure 3c shows how a certain sequence of operators will transform a dyad into a tryad. Figure 3d shows how application of a tree of operators to the embryonic mechanism, will transform it into an arbitrary compound mechanism with exactly one DoF. Terminals of the tree are the actual links of the mechanism.

Alternatively, bottom-up construction of a one-DoF mechanism begins at the leaves of the tree with atomic building blocks and hierarchically assembles them into components. The atomic building block is a dyad as shown in Figure 4a, and has exactly one DoF when grounded. The composition operator ensures that the total number of DoF is not changed when two subcomponents are combined, and thus the total product of the tree will also be a mechanism with exactly one DoF. When combining two components each of one DoF, the resulting assembly will have five DoF (one DoF from each, plus three DoF released by ungrounding one of the compo-

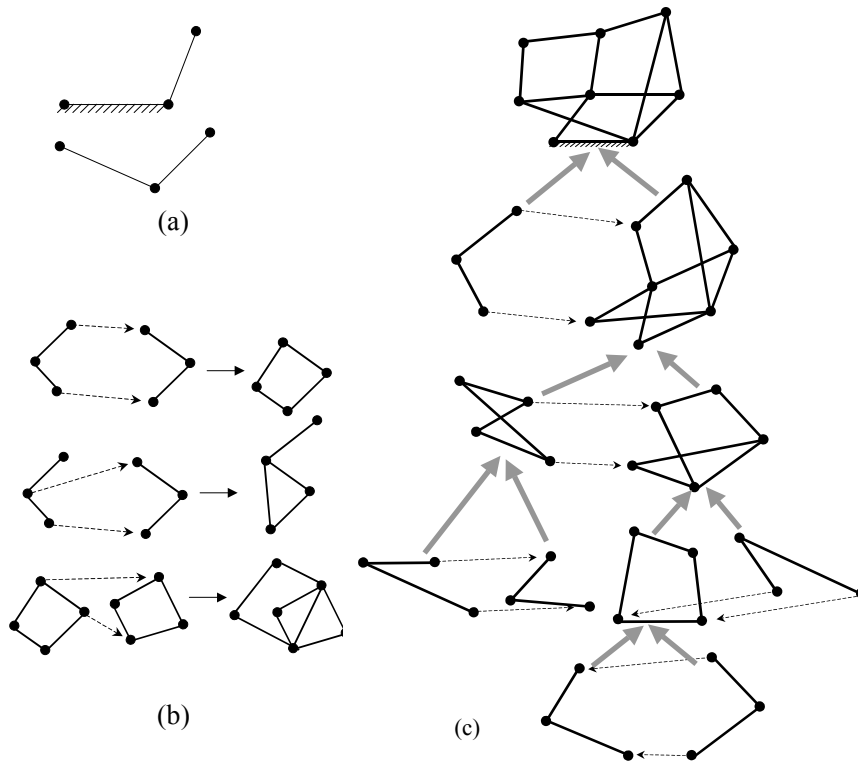


Fig. 4. Bottom up construction of a one-DoF mechanism: (a) An atomic building block of a mechanism has one DoF when grounded; (b) examples of composition of atomic and higher-level building blocks. The composition operator eliminates two vertices, thereby ensuring that the total number of DoF of the compound structure remains exactly one; (c) composition operators can be applied hierarchically in a tree to aggregate atomic building blocks into increasingly complex kinematic mechanisms, each with exactly one DoF.

ments). The total DoF is restored to one by eliminating four DoF through the merging of two point pairs. An example of this process is shown in Figure 4b. Note that points must be merged in a way that avoids overlapping constraints, such as causing two links to merge. The components may need to be scaled and oriented for the merger to work. The ground link of the entire structure is specified at the root of the tree.

3 Test Case: The Straight Line Problem

In selecting a test problem to evaluate the performance of a GP using the above representations, we sought a kinematic synthesis challenge that is on par with human inventive capacity. There are numerous ingenious kinematic mechanisms in many everyday products around us, from cars to DVD players and electric toothbrushes [14]. But nowhere was a single design challenge so clearly stated and so persistent as the *straight-line* problem.

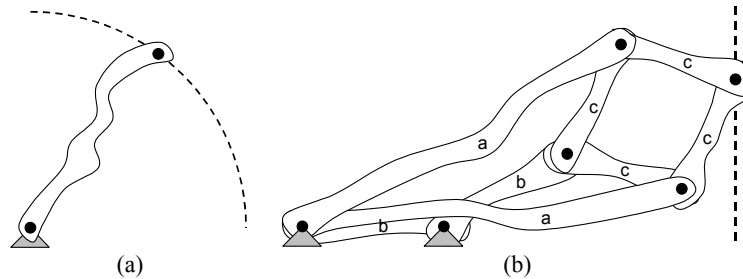


Fig. 5. Mechanisms to trace exact curves. (a) Tracing an exact circle is simple, but (b) tracing an exact straight line without reference to an existing straight line is a challenge that has occupied inventors for nearly a century. The mechanism shown is “The Peaucellier” (1876). All links are shown as crooked sticks to emphasize that the links themselves do not need to be straight; they merely constrain the distance between two nodes.

The straight-line problem seeks a kinematic mechanism that traces a straight line *without reference* to an existing straight line. It is easy to imagine a kinematic mechanism that traces an exact circle, for example, without having a circle pre-built in to it: A simple link, constrained at one endpoint and tracing at the other endpoint would create an exact circle. Figure 5a shows such a device: A compass. Tracing an exact straight line without reference to an existing straight line is, however, much more difficult. It is a challenge that has occupied inventors for nearly a century. One solution, known as “The Peaucellier” (1873), is shown in Figure 5b.

The straight-line problem was of great practical importance in the 18th and 19th centuries. The invention of the steam engine marked a new era of technological advance, but its early development was plagued with problems of reliability and machining accuracy, leading to both steam leakage around the piston heads and overwhelming friction. One of the big challenges was how to convert the reciprocating linear motion of the double-acting (push-pull) piston into a continuous rotary motion of a wheel. Though the use of a crank and a connecting rod seems trivial today, it was by no means apparent at the time, because machining accuracy was such that the piston head could not sustain side loads well and needed a straight guide to keep it from wobbling and leaking steam. Conventional designs at the time used the reciprocating motion to pump water into a reservoir and turn a waterwheel, or complex arrangements of gears and chains. James Watts’ first patent (1782) used a rack and sector (Figures 6a, 6b).

The real breakthrough in steam engine technology came with the invention of a linkage system to guide the piston in a straight line. In 1784, Watt wrote to his partner Boulton: “*I have got a glimpse of a method of causing the piston rod to move up and down perpendicularly, by only fixing it to a piece of iron upon the beam, without chains or perpendicular guides, or untowardly frictions, arch-heads, or other pieces of clumsiness... I think it a very probable thing to succeed*” [15]. The design is shown in Figure 6c. Years later, Watt told his son: “*Though I am not over anxious after fame, yet I am more proud of the parallel motion than of any other mechanical invention I have ever made*” [15].

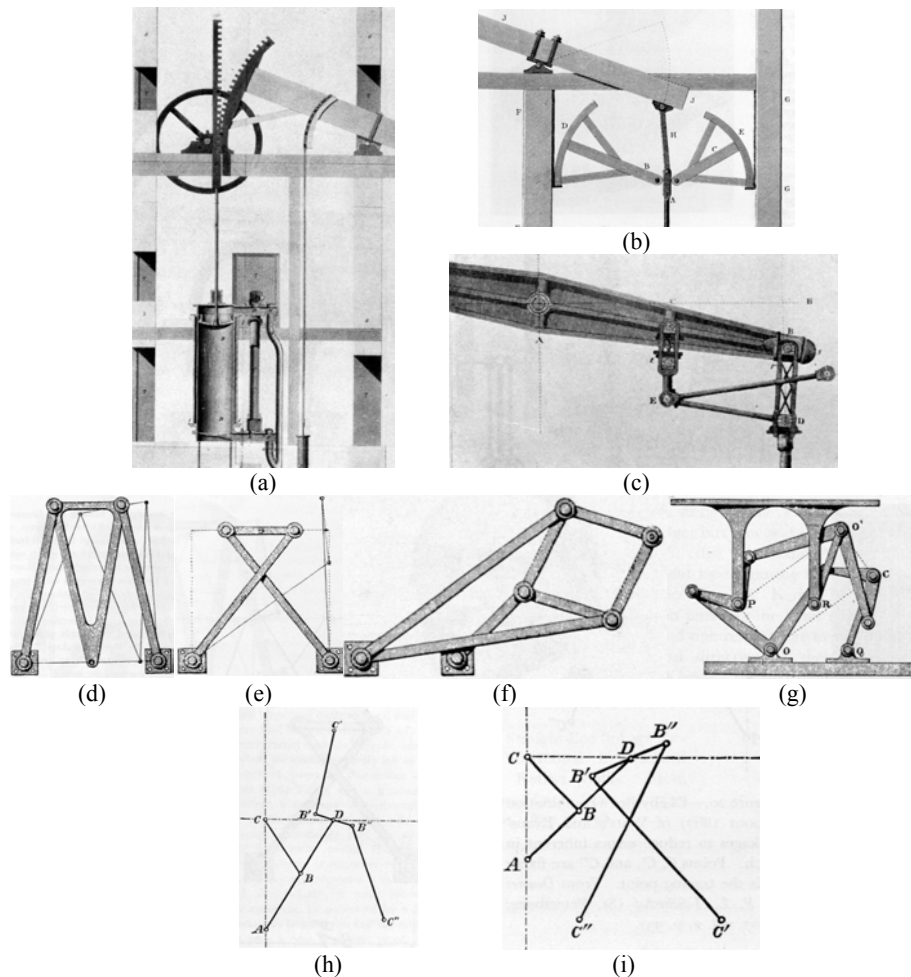


Fig. 6. Some key straight-line mechanisms: (a) Watt's original rack and sector solution, 1782 [15], (b) Watt improvement, 1784, (c) Watt's first straight-line linkage mechanism [15], (d) Robert's linkage, 1841 (e) Chebyshev's linkage, 1867 (f) Peaucellier's linkage, 1873, (g) Silverster-Kempe's linkage, 1877, (h) Chebyshev's combination, 1867 (i) Chebyshev-Evans combination, 1907. From [8].

Since the initial inception of the straight-line mechanism, many inventors engaged in improving and creating alternative designs. Figures 6d-i show a number of additional practical designs. The obsession with the straight-line mechanism continued well beyond what its practical usefulness merited, to become a mathematical puzzle in its own right. The challenge continued even after the invention of the perfect mechanism by Peaucellier in 1873 – a century after Watt's initial invention. Numerous straight-line mechanisms were proposed, as evident from the 39 different straight-line mechanisms shown in the Voigt catalog [19] of educational models. As precision manufacturing improved, the need for straight-line mechanisms diminished, and it is now lost knowledge. Ferguson provides a vivid account of that era [4].

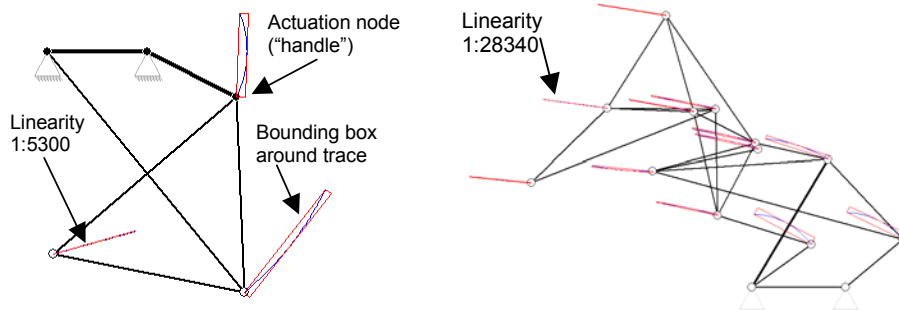


Fig. 7. Evaluation of an evolved straight-line mechanism: The mechanism is actuated at an arbitrary handle and the aspect ratios of bounding boxes of node trajectories are measured. One node of the evolved machine on the left traces a curve that is linear to 1:5300 accuracy. The machine uses the principle of Willis (1841), as seen in Figure 6d. The evolved mechanism on the right traces a curve that is linear to 1:28340 accuracy

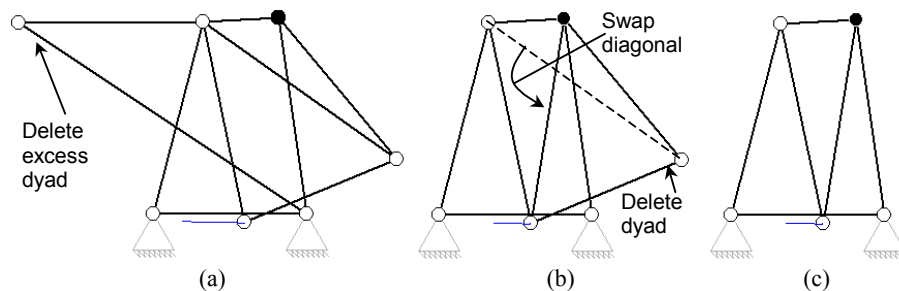


Fig. 8. Reductions of mechanisms: Complex mechanisms can be reduced to simpler mechanisms with equivalent curve traces by through iterative application of two transformations: (a) Elimination of excess dyads, and (b) swapping of diagonals within rigid subcomponents. Mechanisms (a) and (c) are thus equivalent in the curve that the lower node traces.

Simulating and evaluating straight line mechanisms

The performance of a given mechanism was evaluated using an in-house kinematic simulator [11]. This simulator approximates the rigid links with stiff elastic springs, and propagates displacements throughout the structure using a relaxational process, gradually reducing elasticity to approximate rigid behavior. The process iterates until the structure reaches equilibrium; if equilibrium is not reached, then the mechanism is deemed to be over-constrained or under-constrained in some way.

To measure the extent to which a given mechanism traces a straight line, the mechanism is actuated along its single DoF by applying some small force to one of its ungrounded nodes, selected arbitrarily. The trajectories of all nodes are recorded, and then evaluated for straightness. Straightness is computed as the aspect ratio of a tight bounding box of the trajectory. The length over width of the bounding box provides a fitness criterion that measures the maximum deviation from a straight line, as seen in Figure 7. Comparison of mechanism can be difficult, as apparently different machines

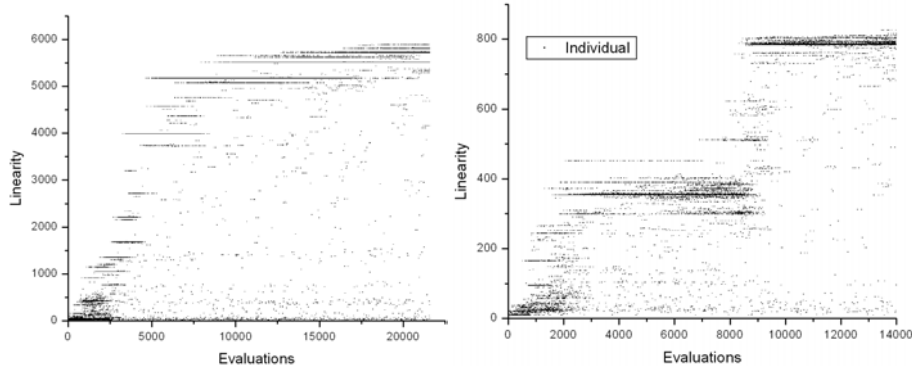


Fig. 9. Two typical runs: Each dot represents an evaluated individual.

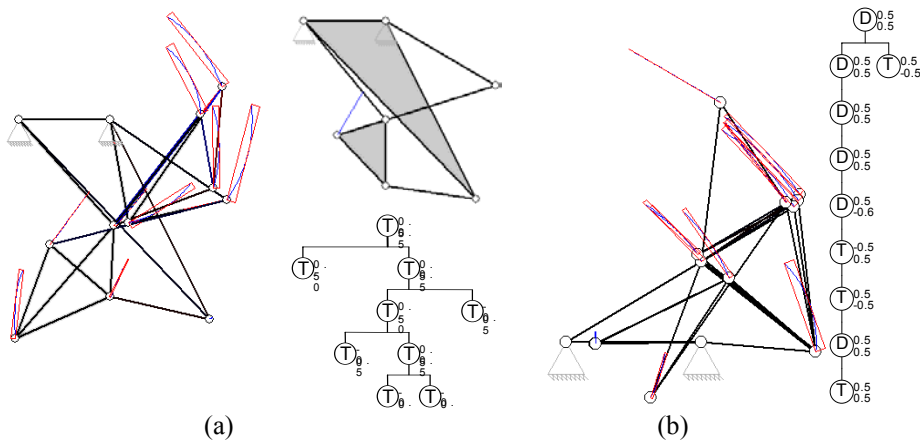


Fig. 10. Two Evolved mechanisms and their tree representations (a) Linearity 1:12819; The simplified equivalent shown top right, and (b) Linearity 1:4979.

may be functionally equivalent. Complex mechanisms can be reduced to simpler mechanisms with equivalent curve traces through iterative transformations (Fig 8).

4 Results

Straight-line mechanisms were evolved using a GP operating on trees describing mechanisms in a top-down representation (Figure 3). We used a population of 100 individuals and fitness proportional selection using stochastic-universal-sampling. The fitness was the linearity of the most linear curve traced by any of the mechanisms' vertices, when the crank node was turned 45 degrees. The search progressed in steps of discovery, as shown in two typical runs plotted in Figure 9.

A variety of mechanisms were produced, most with linearity exceeding 1000 (i.e., deviation of one millimeter over a meter) and some as high as 28000 (35 μ m over a

meter) as seen in Figure 7b. Comparing these compound mechanisms to the known classical solutions is difficult, but some clearly infringe on earlier principles such as that of Willis (1841), as seen in Figure 7a. A number of additional results and their trees are shown in Figure 10.

5 Conclusions

This paper presented the application of genetic programming to the synthesis of compound 2D kinematic mechanisms. Two tree-based representations were proposed: A top-down representations that modified an initial base mechanism, and a bottom-up representation that hierarchically composes atomic components. Both these representations allow for systematically searching the space of mechanisms with a given number of degree of freedom. Both representations are order-independent and largely context independent, which are desirable properties for evolvability.

Application of a GP to the straight-line problem yielded a number of mechanisms that are competitive with, and in some cases infringe upon, previous known inventions. It is difficult to compare the ‘inventiveness’ of the algorithm to that of James Watt: In all fairness, the genius of James Watt was in the very idea to use a linkage mechanism to guide the piston in a straight line, and to pursue this idea without knowing that a solution existed at all. It is, however, fair to compare the algorithms’ performance to that of Watt’s successors who tried to synthesize new and improved linkage mechanisms with the same functionality.

The results shown here are preliminary: Only the top-down representation was tested, and there are many aspects of the evolutionary process that could enhance these results, including, for example, selection methods, diversity maintenance, bloat prevention, and the use of automatically defined functions. Future work will further examine these issues and their application to more contemporary kinematic synthesis challenges such as the design of mechanisms for robotic locomotion.

6 Acknowledgments

This work was supported by the U.S. Department of Energy, grant DE-FG02-01ER45902.

7 References

1. Boers, E.J.W., H. Kuiper, B.L.M. Happel, and I.G. Sprinkhuizen-Kuyper. (1993). Designing Modular Artificial Neural Networks. In *Proceedings of Computing Science in The Netherlands*, H.A. Wijshoff, editor. 87–96. Amsterdam: SION, Stichting Mathematisch Centrum.
2. Bongard J., (2002) *Incremental Approaches to the Combined Evolution of a Robot’s Body and Brain*, Ph.D. Thesis, University of Zurich

3. Saylor J. Walker K., Moon F.C., Henderson D.W., Daimina D., Lipson H., Cornell University Digital Library of Kinematic Models (KMODDL), <http://kmoddl.library.cornell.edu>
4. Ferguson E.S., (1962) "Kinematics of Mechanisms from the time of Watt", Smithsonian Institution Bulletin No. 228, pp. 185-230 (available online at KMODDL [3])
5. Gruau F. (1994) Neural network synthesis using cellular encoding and the genetic algorithm. PhD thesis, Laboratoire de L'informatique du Parallisme, Ecole Normale Supérieure de Lyon, Lyon, France
6. Hartenberg R.S., Denavit J., (1954) "Systematic Mechanism Design", *Machine Design*, Vol. 26 pp. 167-175
7. Hornby G.S., Lipson H., Pollack J.B., 2003 "Generative Encodings for the Automated Design of Modular Physical Robots", IEEE Transactions on Robotics and Automation, Vol. 19 No. 4, pp 703-719
8. Kempe A. B., (1877), *How To Draw A Straight Line*, London (Available online at Cornell University Library, <http://resolver.library.cornell.edu/math/2349409> and at KMODDL [3])
9. Komosinski M., Ulatowski S., "Framstics: Towards a simulation of a nature-like world, creatures and evolution", ECAL '99, pp. 261-265, 1999
10. Koza J., (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press
11. Lipson, H. (2004) A computational relaxation method for simulating compound nonlinear mechanisms. ASME Journal of Mechanical Design, In review
12. Lipson, H., Pollack J. B., 2000, "Automatic Design and Manufacture of Artificial Lifeforms", *Nature* 406, pp. 974-978.
13. Luke, S. and L. Spector. 1996. Evolving Graphs and Networks with Edge encoding: Preliminary Report. In Late Breaking Papers at the Genetic Programming 1996 Conference (GP96). J. Koza, ed. Stanford: Stanford Bookstore. 117-124
14. Moon F.C., (2004) "Leonardo in your toothbrush" (available online at KMODDL [3] as a tutorial)
15. Muirhead J.P., (1854) The origin and progress of the Mechanical Inventions of James Watt, Vol. 1.
16. Ramelli A. (1588) "Le Diverse et Artificose Machine", Paris
17. Reuleaux F, (1876), "The Kinematics of Machinery: Outlines of a Theory of Machines", London: Macmillan (available online at KMODDL [3])
18. Sims, K. "Evolving 3d morphology and behavior by competition". In Brooks, R. and Maes, P., editors, Proceedings 4th Artificial Life Conference. MIT Press, 1994
19. Voigt Werkstatt Gustav (1907) Voigt Catalog., Kinematic Models of Professor Reuleaux, Berlin (available online at KMODDL [3])
20. Willis, R., (1841), *The Principles of Mechanisms*, London (available online at KMODDL [3])