Four-Bar Linkage Synthesis Using Non-Convex Optimization

Mémoire

Vincent Goulet

Maîtrise en informatique
Maître ès sciences (M. Sc.)

Québec, Canada

Four-Bar Linkage Synthesis Using Non-Convex Optimization

Mémoire

Vincent Goulet

Sous la direction de:

Claude-Guy Quimper, directeur de recherche

Résumé

Ce mémoire présente une méthode pour synthétiser automatiquement des mécanismes articulés à quatre barres. Un logiciel implémentant cette méthode a été développé dans le cadre d'une initiative d'Autodesk Research portant sur la conception générative. Le logiciel prend une trajectoire en entrée et calcule les paramètres d'un mécanisme articulé à quatre barres capable de reproduire la même trajectoire. Ce problème de génération de trajectoire est résolu par optimisation non-convexe. Le problème est modélisé avec des contraintes quadratiques et des variables réelles. Une contrainte redondante spéciale améliore grandement la performance de la méthode. L'expérimentation présentée montre que le logiciel est plus rapide et précis que les approches existantes.

Abstract

This thesis presents a method to automatically synthesize four-bar linkages. A software implementing the method was developed in the scope of a generative design initiative at Autodesk. The software takes a path as input and computes the parameters of a four-bar linkage able to replicate the same path. This path generation problem is solved using non-convex optimization. The problem is modeled with quadratic constraints and real variables. A special redundant constraint greatly improves the performance of the method. Experiments show that the software is faster and more precise than existing approaches.

Contents

List of Tables

List of Figures

The best way to predict the future is to create it.

Peter Drucker

Remerciements

Je voudrais d'abord remercier mon directeur de recherche M. Claude-Guy Quimper de m'avoir approché pour ce projet, ainsi que d'avoir promu mon nom auprès d'Autodesk Research. L'ouverture, l'écoute, la disponibilité et la proactivité de Claude-Guy ont été inestimables au regard de l'aboutissement de ce projet. C'est aussi grâce à Claude-Guy et Philippe Cardou que j'ai été initié à la recherche lors d'un stage qui a vraiment débroussaillé et propulsé mes habiletés informatiques. Merci d'ailleurs à Thierry Moisan, à la patience duquel je dois en bonne partie mes habiletés actuelles de programmation.

Bien sûr, mes remerciements vont tout autant à nos collaborateurs chez Autodesk Research, M. Wei Li, M. Hyunmin Cheong et M. Francesco Iorio. Merci pour votre engagement, votre implication active et votre enthousiasme au regard du projet. Merci aussi à ces personnes d'avoir tout fait en leur pouvoir pour permettre la publication de l'article connexe. Un grand merci également à toute l'équipe du Computational Science Research Group pour son accueil chaleureux lors de mon séjour à Toronto.

Merci au CRSNG dont la contribution financière a rendu ce projet possible. Merci également aux arbitres anonymes ayant critiqué le travail et permis de le rendre meilleur, jusqu'à obtenir une publication à la conférence CP 2016. Ce fut réellement un moment marquant de ce projet, mais aussi de ma vie.

Je me dois également de remercier Catherine Bilodeau, Émilie Picard-Cantin et Kim Rioux-Paradis qui ont été la cible de tous mes questionnements mathématiques au cours de mes deux années de maîtrise. Merci d'avoir reçu tous les coups avec curiosité et intérêt. Je dois également à ces personnes l'aboutissement de la preuve mathématique qui fait l'objet d'une section de ce mémoire. Un grand merci tout particulier à Émilie pour avoir pris le temps de rigoureusement valider toutes les étapes des 15 pages de la version initiale de la preuve.

Je souhaite également adresser des remerciements à tous les acteurs indirects qui m'ont supporté pendant la maîtrise et aussi pendant les étapes qui m'y ont conduit. Un merci spécial à mes parents et leur foi inébranlable en moi de chaque instant. Merci de m'avoir écouté, d'avoir toujours été disponibles, de m'avoir accompagné dans la décision d'entreprendre ce projet et de m'avoir préparé des repas dans les moments ou même m'occuper de moi-même était un luxe. Merci aussi à Sarah-Ève Goulet, Benjamin Côté, Marie-Andrée Faucher, Marilie Labonté, Jessica Marcoux et Joëlle Vaillancourt d'avoir été présents pour moi dans les bons moments et les moments difficiles.

Acknowledgements

First, I would like to thank my advisor Prof. Claude-Guy Quimper for inviting me to this project, as well as promoting my name in front of Autodesk Research. His creativity, attention, availability and proactivity have been invaluable to the success of this work. I am also indebted to Claude-Guy, as well as Prof. Philippe Cardou, who have been my very first mentors in a previous research internship, which really was my practical initiation to computer science and mechanism theory. My thanks also go to Thierry Moisan and his patience, to which I owe much of my programming skills.

Of course, my sincere thanks also go to our collaborators at Autodesk Research, Mr. Wei Li, Mr. Hyunmin Cheong and Mr. Francesco 'Frio' Iorio. Thanks for your commitment, your active involvement and your enthusiasm towards the project. Thanks to Mr. Li, Mr. Cheong, Mr. Iorio and Mr. Quimper for putting in every effort into the publication of the related accepted paper. Also thanks to the whole Computational Science Research Group for a very enjoyable stay at the Toronto office.

Special thanks to the NSERC, whose funding was absolutely critical to the very existence of this project. Thanks to the blind reviewers who have criticized this work and allowed improving it, until it was accepted as a paper at the CP 2016 conference. This was really a key moment of this project, but also of my life.

I must also thank Catherine Bilodeau, Émilie Picard-Cantin and Kim Rioux-Paradis whom I have been targeting relentlessly with every mathematical question that occurred throughout the journey. Thank you for receiving each hit with curiosity and interest. I also owe to these people the completion of the mathematical proof for the area of the four-bar linkage coupler curve feature in the thesis. Very special and large thank you to Émilie for rigorously validating every step of the original 15 pages of the proof.

I also wish to address thanks to all indirect participants whom supported me along my masters' and also during the stages that led me to it. A special thank you to my parents and their unshakable faith in me, at every instant. Thanks for listening, being available, accompanying me in deciding to embark in the project, and keeping me well-fed in times where caring about myself was a luxury. Thanks to Sarah-Ève Goulet, Benjamin Côté, Marie-Andrée Faucher, Marilie Labonté, Jessica Marcoux and Joëlle Vaillancourt for being there in good and rough times.

Introduction

Mechanical engineering is the field of engineering specialized in the study, design and use of mechanisms and machines. Mechanisms are an arrangement of solid parts that transform an original force or motion into a different but controlled force or motion. The design of mechanisms is a very broad area of research as well as an active industry. Progress in mechanical engineering is tightly connected to progress in computer science. State of the art computer-aided design (CAD/CAE) software developed by industry leaders such as Autodesk help create parts, simulate motion, predict performance and properties, etc. However, modern tools still lack practical features to perform completely autonomous mechanism synthesis. Hence, design mechanism still requires the intervention of highly trained experts. The optimality of the design often cannot be guaranteed.

Here we address the problem of synthesizing a four-bar linkage (Figure 0.1) that can trace a path specified by a user. Four-bar linkages are composed of four rigid-bodies connected in a cycle. They are very simple assemblies that can produce complex motions. Applications of four-bar linkages span across the manufacturing, agriculture, automotive and robotics industries [35]. Designing a four-bar linkage that replicates a provided curve is a time-consuming process. State of the art approaches also often lack optimality or generality. This thesis introduces an automated and fast approach to synthesize four-bar linkage using non-convex optimization. The input of the approach is the curve provided by the user, and the output is the list of parameters needed to specify the linkage. This method can serve as basis towards solving more complex mechanisms.

A mechanical design problem is a problem in which we are trying to find the parameters of a certain type of mechanism so that the desired behavior is obtained. For example, designing a gearbox means finding the number of shafts and gears required, their size and their position in order to obtained the desired gear ratios. In the case of the path synthesis problem, the desired behavior is that a certain point on the linkage traces a specified curve as the linkage is moving. The tracing point is located on a bar called the *coupler*, and therefore the curve traced by a linkage is called a *coupler curve*. Solving the problem means finding the lengths of the bars and the position of the fixed pivots of a linkage so that the coupler curve of the linkage matches the specified curve as closely as possible.

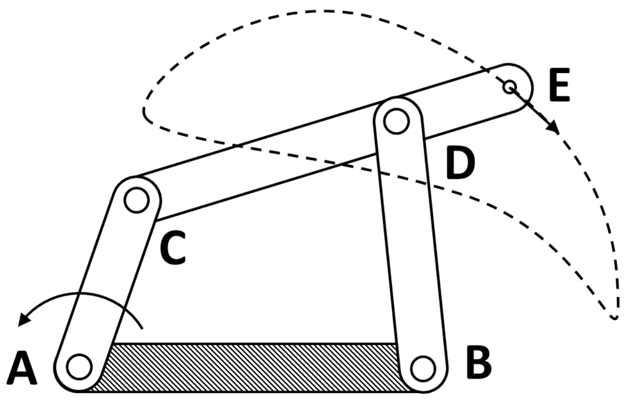


Figure 0.1: A four-bar linkage and output *coupler curve*

This thesis is structured in two parts. The first parts exposes the preliminary notions required to understand the synthesis method. Theory on non-convex optimization is provided with a description of the solving techniques it uses. Theory on mechanical linkages is provided as well as descriptions of state of the art linkage design methods. The second part focuses on the linkage synthesis method that was developed in a collaborative project with Autodesk Research. The method is first described along with a presentation of the software developed. Then, results are presented and discussed.

Part I

Preliminaries

Chapter 1

Optimization

1.1 Introduction

Optimization consists of choosing the *best* element from a given search space. Any problem which involves minimizing or maximizing a given quantity, such as cost, error, delay or profit, can be modeled as an optimization problem. Applications include optimal route planning, resource distribution or scheduling. Such problems can be modeled mathematically and an optimal solution can be computed. Quantities to be determined are *variables* which are subject to *constraints*. Finding a good model for a problem or finding how to express a constraint can be challenging tasks and may constitute a whole research project. In general, an optimization problem model has the following form:

$$\begin{aligned} &\textit{minimize } f(X) \\ &\textit{subject to } A(X) \leq 0 \end{aligned} \tag{1.1}$$

where $X = \{x_1, x_2, ..., x_n\}$ is a vector of variables

f is the objective function

$A = \{g_1(X), g_2(X), ..., g_m(X)$ is a vector of constraints

n is the number of variables

m is the number of constraints

There are problems where only the constraints matter, and there is little to no interest in optimization. These are called *constraint satisfaction problems* or CSP, and many of the techniques discussed here apply to such problems.

In this chapter, we describe how optimization problems can be modeled and solved. We discuss solving techniques relevant to the four-bar linkage path synthesis problem, and look into available software.

1.2 Mathematical Models

Mathematical models represent problems using variables and mathematical expressions to capture their main features. In this section, we describe the parts an optimization model is typically broken down into, which are parameters, variables, an objective function, and constraints. Then, we illustrate the concepts discussed by applying a modeling process to a simple example.

Parameters

A good model should be general enough to encompass a certain range of problems. Specific cases within this range are called *instances*. Instances are usually specified by a set of constants called *parameters*.

Variables

The values which can be tuned to affect the quality of the solution are the *variables* of the problem. There are different types of variables, such as Boolean, integer or continuous. They also have a *domain*, which is the set of values they can take. The domain of a variable x is noted $\mathrm{Dom}\{x\}$. Determining what quantities are variables is part of the modeling process. For a single problem, there may be several valid ways to define variables, and experiments may be necessary to determine which is best.

Objective Function

The objective function is an expression to be either minimized or maximized. A maximization problem can be easily converted to a minimization problem and vice versa by changing the sign of the objective function. It may be any arbitrarily complex function of the variables.

Constraints

Relationships in between variables and parameters are encoded by *constraints*. Constraints are computable expressions and embody the search space of a problem. The search space is the set of all combinations of variable values satisfying all constraints. When modeling a problem, one should identify all statements about the variables that *must be true*. Each statement should then be expressed as a constraint.

1.2.1 Modeling Example

We now illustrate the modeling process using the following optimization problem. Suppose the council of a small town containing N houses decides to build a radio tower. It is required that the signal from the tower reach all houses. However, the radius covered by the signal is proportional to the squared power output by the tower. The problem is to find where to build the tower so that the required power be minimal. The problem is illustrated at Fig. 1.1.

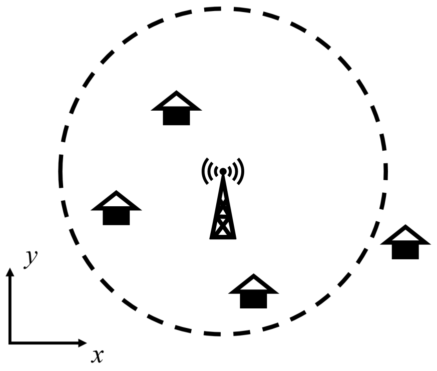


Figure 1.1: A radio tower must be positioned such so as to reach all houses with minimal power.

In the radio tower problem, the value of N as well as the coordinates x_i and y_i of all houses are parameters with fixed values.

From the problem definition, the variables are the output power P, the reach radius R, and the tower location coordinates x_T and y_T. All of these variables are continuous quantities. The nature of the variables may impose restrictions on the domains. For instance, both the power and the radius have to be positive values to make physical sense. Table 1.1 list the variables of the problem as defined so far:

Variable	Description	Domain
P	Tower output power	$[0, \infty)$
R	Reach radius	$[0, \infty)$
x_T	x-coordinate of tower	$(-\infty, \infty)$
y_T	y-coordinate of tower	$(-\infty, \infty)$

Table 1.1: Preliminary variables of the radio tower problem and domains

A part of the modeling process is to restrict the domains as much as possible without removing the optimal solution from the set of solutions. This is important because the larger the domains, the larger the search space and the more difficult it is to find and guarantee optimality. Domains restriction may arise from intrinsic properties, logical deductions or practical considerations. For instance, an intrinsic property of the radius of a circle is that it is always positive. Also, by logical deduction, we know that any solution where the tower is located outside of the smallest box enclosing the houses is not optimal. There may also be a practical upper limit P_{max} to the power output by the tower. Updated domains are listed at table 1.2.

Variable	Domain
P	$[0, P_{\text{max}}]$
R	$[0, \infty)$
x_T	$[\min(x_0, x_1, ..., x_N), \max(x_0, x_1, ..., x_N)]$
y_T	$[\min(y_0, y_1, ..., y_N), \max(y_0, y_1, ..., y_N)]$

Table 1.2: Updated variables of the radio tower problem and domains

Now for the constraints, we have to identify conditions that must be fulfilled by a valid solution. For the problem definition, we know that all houses must be within range of the tower. This constraint can be expressed as such:

$$(x_i - x_T)^2 + (y_i - y_T)^2 \leq R^2 \qquad \forall i \in \{0, 1, ..., N\} \tag{1.2}$$

From physics, we know that the output power is related to the reach of the tower as such:

$$P = aR^2 \tag{1.3}$$

where a is a coefficient obtained from radiation theory. These constraints suffice to encode the problem, because a solution that satisfies them and the domains can be interpreted to a real-world solution.

From the definition of the problem, the objective function f must minimize the input power. Because the power is already represented by a variable, we simply have :

$$f(X) = P \tag{1.4}$$

Table 1.3 summarizes the model for the radio tower problem. In the next section, we see how mathematical models can be solved using specialized software.

1.3 Solving Mathematical Problems

Solving a constraint programming model consists of setting the variables to values that optimize the objective function while respecting all constraints. This can be done with a *solver*. Solvers are software able to interpret the formulation of a model, to explore the solution space

Parameters	N Number of houses x_i, y_i House coordinates
Variables	P Tower output power R Reach radius x_T, y_T Tower coordinates
Constraints	$(x_i - x_T)^2 + (y_i - y_T)^2 \leq R^2 \quad \forall i \in \{0, 1, ..., N\}$ $P = aR^2$
Objective function	$f(X) = P$

Table 1.3: Model for the radio tower positioning problem, applicable to an arbitrary town.

using adapted strategies, to identify and to return the optimal solution. Some solvers are specialized for certain types of problems, while others cover a large set of applications. Problems can be categorized depending on the types of variables, types of constraints, and types of objective function they can process.

SAT Programs

Boolean variables and clause constraints encode SAT programs. The domain of such variables is {True, False}. The problem of finding if a SAT program has a satisfiable assignment was the first problem to be proven NP-complete [10]. SAT solvers such as GRASP [30] are specialized for this type of problem. A SAT problem has to be converted to its *conjunctive normal form* (CNF) to be processable by a SAT solver. A problem in CNF only contains negations, disjunctions and conjunctions. All logical expressions can be converted to this form. For example, take the following SAT problem :

$$\begin{aligned} x &\in \{True, False\} \\ y &\in \{True, False\} \\ \textit{subject to } &\neg(x \wedge y) \\ x &\implies y \end{aligned} \tag{1.5}$$

The same problem expressed in the CNF form is:

$$(\neg x \vee \neg y) \wedge (\neg x \vee y) \tag{1.6}$$

The two possible assignments are $\{x = False, y = False\}$ and $\{x = False, y = True\}$. To turn this into an optimization problem, one could add the objective function to maximize the number of $True$ variables. In that case, the optimal solution would be $\{x = False, y = True\}$.

Linear Programs and Integer Programs

A *linear program* is a problem of the form:

$$\begin{aligned} &\textit{minimize } c^T x \\ &\textit{subject to } Ax \leq 0 \\ &\qquad\qquad x \geq 0 \end{aligned} \tag{1.7}$$

where c is a vector of constants, x a vector containing the variables of the problem and A a matrix of constants.

An *integer program* is a linear program with the additional constraint that $x \in \mathbb{N}$. An integer program is also NP-complete since Boolean variables can be encoded as integer variables whose domain is {0, 1} and linear constraints can encode SAT clauses. Assuming $x, y, z \in \{0, 1\}$, the equivalent linear inequalities for logical AND, OR and NOT are shown at table 1.4.

Operator	Logical form	Linear form
NOT	$z = \neg x$	$z = 1 - x$
AND	$z = x \wedge y$	$z \geq x + y - 1$ $z \leq x$ $z \leq y$
OR	$z = x \vee y$	$z \leq x + y$ $z \geq x$ $z \geq y$

Table 1.4: Equivalent linear forms of basic logical operators.

To explore the search space of integer problems, solvers can use the *branch and bound* technique, which is described in detail in the next section (1.3.1). Filtering algorithms can be applied in the branch and bound process. They reduce the domain of the variables. This has the effect of pruning the search tree, hence speeding up the solving time. An integer program can be relaxed to a linear program with continuous variables. The objective value of such a relaxation provides a bound on the objective value of the non-relaxed problem.

Non-Linear Programs

Programs involving continuous variables and non-linear functions are called non-linear programs or NLPs. If a NLP can be proven convex, then methods such as gradient descent can be used to find the global optimum. However, if a problem is non-convex, there may exist several local optima, and the whole search space has to be explored. The four-bar linkage synthesis problem belongs to this category of problems, and therefore techniques for solving non-linear programs are described further in this section.

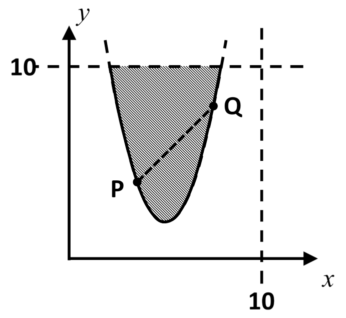
1.3.1 Non-Linear Solving Techniques

There are two main sets of solving techniques depending on whether a non-linear problem is convex or non-convex. A problem is convex if any point on a segment joining two valid solutions is also a solution. If point **P** and point **Q** are solutions of the search space, then the problem is convex if the following is also a solution:

$$\alpha\mathbf{P} + (1-\alpha)\mathbf{Q} \qquad \text{for any } \alpha \in [0,1] \tag{1.8}$$

Equation 1.8 can be graphically interpreted for simple instances, such as seen on Fig. 1.2.

$$x \in [0,10]$$
$$y \in [0,10]$$
$$\text{subject to } y \geq (x-5)^2 + 2$$



$$x \in [0,10]$$
$$y \in [0,10]$$
$$\text{subject to } y = (x-5)^2 + 2$$

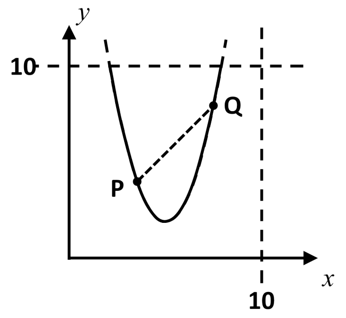


Figure 1.2: The first problem is convex because it is not possible to choose a pair of valid points and connect them by a segment passing over invalid points. In the second problem, the solution space is limited to the curve. It is non-convex because it is possible to choose a pair of valid points whose connecting segment passes over invalid points.

Convex problems are easier to handle because they contain only one minimum. Therefore, strategies like the gradient descent always converge towards the optimal solution. This is not the case for non-convex problems, in which many local minima may exist. The four-bar linkage synthesis problem is non-convex, so further techniques need to be explored to be able to solve it.

Filtering and Consistency

The search space of an optimization problem is delimited by the domains of the variables and the constraints. However, some intervals may be unachievable because constraints are violated regardless of the values assigned to the other variables. The process of detecting and removing such portions of the domains is called *filtering*. Filtering improves the *consistency* of the problem, because it removes contradictions.

There are ways of enforcing consistency that are stronger than others. For example, *domain consistency* ensures that every value of the domains is potentially part of a solution. For continuous variables, this can be a problem. Typically, continuous domains are stored as two floating point bounds. Applying domain consistency to a continuous variables might prune sub-intervals and make its domain non-continuous. Since this filtering denatures the problem, we rather apply *bounds consistency* that only prunes the smallest and the largest value from each domain. A problem is bounds consistent if for each bound of each domain, there exists for each constraint at least one feasible point in the search space. Interval arithmetic is used to contract the domains in real-valued problems.

Interval Arithmetic

When ranges of values are assigned to variables, interval arithmetic can be used to determine whether sub-intervals are consistent or not. It is an extension of regular arithmetic. The most simple operations to define are interval negation and addition. For the two real-valued intervals $[a, b]$ and $[c, d]$, these operations yield:

$$-[a, b] = [-b, -a] \tag{1.9}$$

$$[a, b] + [c, d] = [a + c, b + d] \tag{1.10}$$

From these two operations, we can define interval subtraction:

$$\begin{aligned} [a, b] - [c, d] &= [a, b] + (-[c, d]) && (1.11)\\ &= [a, b] + [-d, -c] && (1.12)\\ &= [a - d, b - c] && (1.13) \end{aligned}$$

Next, we look into interval multiplication and division. In interval multiplication, the result depends on the values of the bounds. For interval division, we have to pay attention to whether 0 is in the divisor interval. In general, these operations are defined as follows:

$$[a,b] \cdot [c,d] = [\min\{a \cdot c; a \cdot d; b \cdot c; b \cdot d\}, \max\{a \cdot c, a \cdot d, b \cdot c, b \cdot d\}] \tag{1.14}$$

$$\frac{[a,b]}{[c,d]} = \begin{cases} [a,b] \cdot [1/d, 1/c] & \text{if } 0 \notin [c,d] \\ [a,b] \cdot (-\infty, 1/c] & \text{if } c = 0 \\ [a,b] \cdot [1/d, \infty) & \text{if } d = 0 \\ (-\infty, \infty) & \text{if } 0 \in \ (c,d) \end{cases} \tag{1.15}$$

Last, let us define the square operator:

$$[a,b]^2 = \begin{cases} [a^2, b^2] & \text{if } 0 \leq a \leq b \\ [0, \max\{a^2, b^2\}] & \text{if } a \leq 0 \leq b \\ [b^2, a^2] & \text{if } a \leq b \leq 0 \end{cases} \tag{1.16}$$

This operator shows that interval arithmetic differs from scalar arithmetic. Indeed, a tighter contraction is possible with the square operator than with the multiplication operator, hence $[a,b]^2 \neq [a,b] \cdot [a,b]$.

An algorithm implementing filtering for a certain operator is called a *contractor*. Supported operators are one of the aspects to consider when choosing a solver.

We now illustrate with an example how filtering can be performed to iteratively reduce the search space. Let x, y and z be continuous variables with the following domains:

$$x \in [-2, 7] \qquad y \in [-5, 20] \qquad z \in [-4, 3] \tag{1.17}$$

These variables are also subject to the following constraints:

$$y \leq 2x + z \tag{1.18}$$

$$z \geq y^2 + 2 \tag{1.19}$$

We first use the first constraint to filter the domain of y:

$$y \leq 2[-2, 7] + [-4, 3] \tag{1.20}$$

$$y \leq [-4, 14] + [-4, 3] \tag{1.21}$$

$$y \leq [-8, 17] \tag{1.22}$$

Given the domains of x and z, we now know that constraint (1.18) is violated if y takes any value greater than 17. Therefore, by contracting the upper bound of y's domain to 17, we lose no valid solution. We now use constraint (1.19) to constrain the domain of z:

$$z \geq [-5, 17]^2 + 2 \tag{1.23}$$

$$z \geq [0, 289] + 2 = [2, 291] \tag{1.24}$$

We now have the contracted domain such that $z \in [2, 3]$. Because the domain of z was updated, we might be able to contract other domains further by going back to the first constraint. This effect is called *constraint propagation*. In some cases, constraint propagation can be iterated without limit and the domains keep shrinking. For that reason, a stopping criterion must be established so that the time required to perform filtering does not outweigh the benefits of filtering.

Linearization

Because solving linear algorithms is so efficient, a common strategy to handle non-linear problems is to build linear relaxations of the search space. The new search space is called a relaxation because it tolerates solutions that are not valid for the original problem.

Linearization algorithms implement different strategies depending on which function forms the edge of the search space. For smooth convex sections, like a parabola, tangent planes are created using the derivatives. For bilinear terms, or terms of the form $z = xy$, the well-known McCormick envelopes [31] are the tightest possible convex hull. In their article on the solver *Couenne* [4], Belotti et al. graphically present the linearization for a sample variety of functions (Figure 1.3).

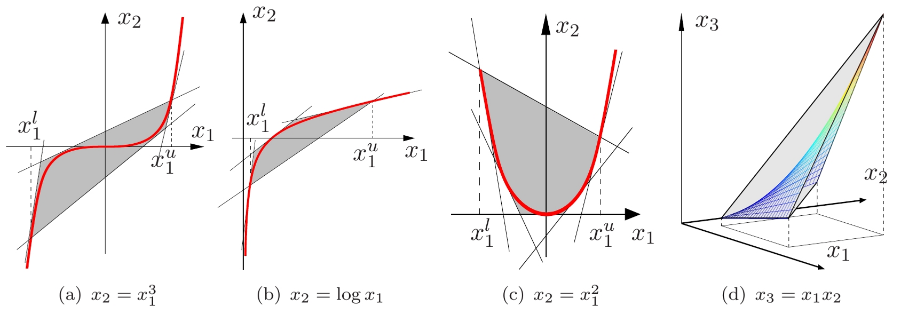


Figure 1.3: Linearization examples for basic functions (Figure 1 in [4]).

For shapes like the parabola, arbitrarily many planes can be used to approximate the curve. A compromise has to be made when choosing the number of planes (see Figure 1.4 c). More planes make solving longer, but also more precise. A strategy used in the solver Couenne

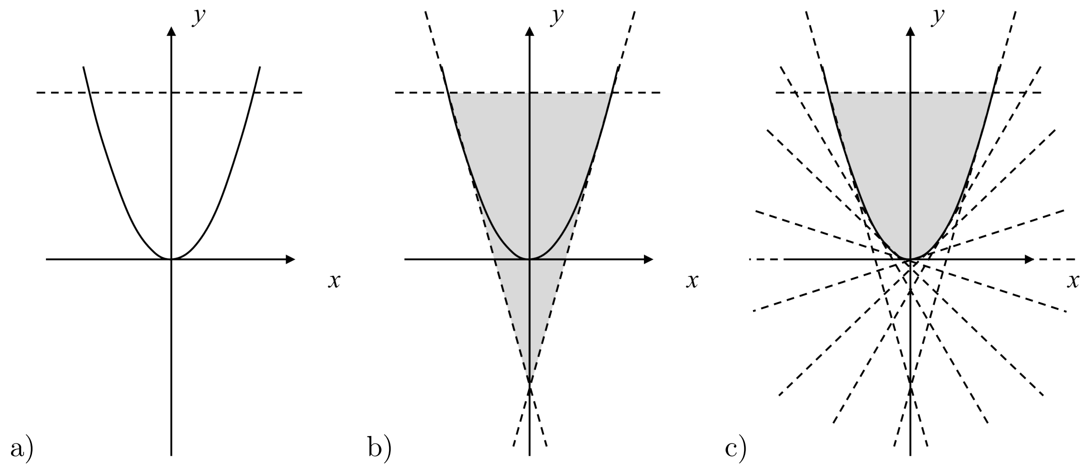


Figure 1.4: Linearization of a parabola; a) The parabola is a non-linear function; b) A scarce linearization may be exceedingly tolerant; c) A dense linearization may be an inefficient use of resources.

is to solve the linear problem, and refine the linear hull to exclude the solution found until the improvement in the quality of the solution is insufficient or null. That way, the planes generated are sure to be relevant.

Linearization helps to find bounds to solutions more quickly, but it is not sufficient alone to find a solution to a non-linear problem. It can be however be used as a heuristic to orient the search.

Branch and Bound

A major challenge of non-convex optimization is identifying a global optimum. Local optima are easily identified, but in general, the whole non-convex search space has to be checked for global optima. *Branch and bound* is a strategy to systematically search the solution space and remove parts of it until the global optimum can be identified. It builds a tree by recursively separating the original problem into sub-problems, and removes branches when they can be proven not to contain the global optimum. Here we describe a general branch and bound algorithm and provide a simple graphical example. Algorithm 1 shows the pseudocode of the algorithm.

We assume a minimization problem. Recall that a maximization problem can be converted to a minimization problem by changing the sign of the objective function. The branch and bound (Algorithm 1) starts with the original problem P_0 with a set of variables X and a specified tolerance ϵ which is a small constant representing the smallest meaningful quantity. If a variable of the problem has a domain spanning an interval smaller than ϵ, that variable is considered set. In the initialization phase, the set of non-convex problems S is created and initially contains only the main problem P_0. A variable s containing the best objective value is set to infinity. A variable *gap* containing the difference between the best objective value and the best lower bound is set to infinity as well.

The exploration phase then begins. It is a while loop. The loop keeps going as long as there are problems in S and the gap is greater than ϵ. A problem P is chosen using a heuristic and removed from S. If all variables of P are set, they represent a valid solution to the problem — though not necessarily optimal. In this case, the objective value is stored to s_{P}. The best solution between s and s_{P} is stored to s. Else, P is relaxed to RP, which is easier to minimize. If the lower bound obtained from RP is greater than s, the sub-problem is discarded because it cannot contain the global optimum. Otherwise, the problem enters the branching phase.

A variable x is chosen from X to branch on with a heuristic. Another heuristic is used to choose a branching point x_{mid} within the domain of the variable. Problem P is split into two problems P- and P+, with respectively $x \in [x_{\mathrm{min}}, x_{\mathrm{mid}})$ and $x \in [x_{\mathrm{mid}}, x_{\mathrm{max}})$ for the domain of x and identical to P otherwise. P- and P+ are added to S and the loop starts over. When the while-loop is exited, the best solution s is returned.

Algorithm 1 Branch and bound(P_0, ϵ)

```
1: S ← {P_0}
2: X ← set of variables of P_0
3: $s \leftarrow \infty$
4: $gap \leftarrow \infty$
5: while S ≠ ∅ and $gap > \epsilon$ do
6:     Choose P from S
7:     S ← S\{P}
8:     if All variables of P are set then
9:         $s_P \leftarrow$ objective value of P
10:        $s \leftarrow \min\{s, s_P\}$
11:    else
12:        RP ← Relax(P)
13:        $low \leftarrow minimize$(RP)
14:        $gap \leftarrow \min\{gap, s - low\}$
15:        if $low < s$ then
16:            Choose $x$ from variables in X
17:            Choose $x_{\mathrm{mid}}$ in Dom($x$)
18:            Create P- same as P, but Dom($x$) = $[x_{\mathrm{min}}, x_{\mathrm{mid}})$
19:            Create P+ same as P, but Dom($x$) = $[x_{\mathrm{mid}}, x_{\mathrm{max}})$
20:            S ← S ∪ {P-,P+}
21:        end if
22:    end if
23: end while
24: return $s$
```

The branch and bound algorithm can be represented as building a tree. An example of a branch and bound tree is presented on Figure 1.5.

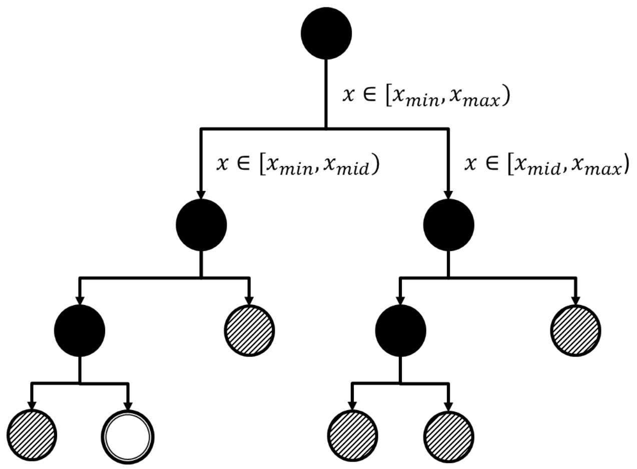

$x \in [x_{min}, x_{max})$
$x \in [x_{min}, x_{mid})$
$x \in [x_{mid}, x_{max})$

Figure 1.5: Example of a tree of sub-problems created by a branch and bound algorithm. The union of all leaves gives the original problem. The black nodes are nodes which may contain the global optimum. Hatched nodes are nodes whose lower bound are greater than the best solution. The white node contains the best solution.

1.3.2 Non-Global Optimization Techniques

1.3.3 Local Search Algorithms

Local search algorithms follow the intuition that if you are looking for a minimum, you should travel downwards. They start with some assignment of the variables, which corresponds to a point in the solution space. Then, they take a look at the surroundings of this point and move in an effort to improve the objective value of the satisfiability. They can move deterministically by calculating the gradient of the function, as is the case with the gradient descent method or the Newton-Raphson method. They can also move stochastically, as is the case with genetic algorithms. The search stops when the solution does not improve sufficiently between iterations, meaning that a local minimum has been reached. These methods are not global in the case of non-convex functions, because they have no way to guarantee that there is no lower minimum elsewhere in the solution space. To address this shortcoming, it is in some cases sufficient to launch many searches with different starting points.

Genetic Algorithms

Genetic algorithms are properly introduced here because the chapter 4 on experimentation features a comparison with such an algorithm.

Species evolution can be modeled as a naturally occurring optimization process, where fitness of populations is being maximized with respect to certain environment constraints. Genetic algorithms, also called evolutionary algorithms, are inspired from this stochastic process. They roughly comprise the following elements: genes, individuals and population, fitness, selection, reproduction, mutation and generation.

The n variables of the problem are the *genes*. The number of variables n depends on the model definition. The goal of the genetic algorithm is to find the optimal value for each gene.

Individuals are n-dimensional points in the solution space. They are potential solutions. At the beginning of the genetic algorithm, an initial *population* constituted of M individuals is generated by assigning random values for each gene of each individual. The population is therefore a set of solutions.

The *fitness* corresponds the objective value for an individual. It reflects its performance towards the constraints of the problem. The fitness metric is a function designed by the user to quantify that performance. It should be sensitive enough to rank solutions accurately. Often, penalty factors are included in the fitness metric when a solution violates a constraint that cannot be satisfied using other means.

The idea is to iteratively improve the population with respect to the *fitness* metric. The population at the beginning of an iteration is called a *generation*. To obtain the next generation, individuals are chosen according to some *selection* process involving their fitness as well a stochastic elements. Selected individuals are combined through some *reproduction* process, generating new and possibly better individuals. Generated individuals may have genes altered randomly, a process referred to as *mutation*. The new individual may be added to the population and less fit individuals may be removed so that the population globally grows fitter or closer to an optimal solution.

Genetic algorithms are very easily implemented and require no use of advanced solvers. They cannot guarantee global optimality, but its stochastic approach may be able to pull it out of local optima.

1.4 Non-linear Program Global Solver Software

Linkage synthesis problems may be modeled into one of two principal types of problems: non-linear programs (NLP) if all variables are continuous, or mixed integer non-linear program (MINLP) if integer variables are present. For instance, the parameters of a four-bar linkage are continuous quantities, such as lengths and positions. A four-bar linkage synthesis problem can therefore be modeled as a NLP. However, the geared five-bar linkage, represented at Figure 2.23 b) and discussed at section 2.7, which has an integer gear ratio, could be modeled into a MINLP. Table 1.5 lists software able to solve non-convex problem to global optimality. The 2010 survey on MINLP software by Bussieck was a good guide for most of presented software [7]. These solvers need the problem to be expressed algebraically in order to compute envelopes and estimators. Each solver has a specific set of operators it can process. Note that a much wider variety of solvers have non-linear capabilities, but they either lack variety in the constraints they can process or cannot guarantee global optimality.

Name	License	Algorithm keywords	Developed by	Reference
ANTIGONE	Commercial	Spatial branching Linearization	Princeton University	[32]
Baron	Commercial	Branch and reduce	The Optimization Firm	[44, 50]
Couenne	CPL [37]	Spatial branching Linearization	IBM Carnegie Mellon University	[4]
IBEX	LGPL [14]	Contractors	École des mines de Nantes ENPC ENSTA	[9]
LindoAPI	Commercial	Branch and bound NLP Relaxation	Lindo Systems Inc.	[28]
RealPaver	BSD [36]	Branch and prune Paving	Université de Nantes	[20]
SCIP	ZIB Academic	Branch and bound Linearization	Zuse Institute Berlin	[2]

Table 1.5: List and data of non-convex global optimization software

1.5 Conclusion

Though non-convex optimization problems are difficult, there is a lot of business and industry interest towards developing better software to address them. State of the art solvers are able to guarantee global optimality by using techniques such as branch and bound to span the entire search space of a problem. We chose *Couenne* to address the problem of four-bar linkage synthesis. The next chapter presents an overview on the theory of linkages.

Chapter 2

Linkage Synthesis

2.1 Introduction

Mechanical systems play a central role in every sphere of modern society. They are used in critical large-scale systems such moveable bridges or aircraft, as well as in everyday life items such as desktop lamps or ball pens. Since the First Industrial Revolution, new creative designs and applications keep emerging and increasing our capabilities and quality of life. Mechanical design remains an active field of research, whether to improve the performance of well-established systems or to undertake new challenges such as microelectromechanical systems or MEMS.

Even with the modern computational means, even the design of well-known mechanisms remains challenging, as the complexity of even simple assemblies grows rapidly out of hand. This chapter gives the necessary basics on mechanical systems for the four-bar linkage synthesis problem. More specifically, background mechanical notions are discussed, then a more focused study on linkages is presented, along with the state of art linkage synthesis techniques.

2.2 Mechanisms and Machines

The two terms *mechanism* and *machine* are often used interchangeably. Both refer to hardware assemblies which convert input motion into a predetermined and controlled output motion. Norton [35] proposes the following distinction. A device is called a *mechanism* if it is subject to low forces and torques, and it is called a *machine* otherwise. The mark between both terms is a gray area and is very much a matter of usage. In the scope of this work, the term *mechanism* is preferred since the forces and torques are not taken into consideration. Further analysis may be required for applications involving higher forces. Dynamic analysis constitutes an interesting avenue for future work.

Mechanisms are tools used to facilitate physical tasks. This can be achieved in a variety of ways, such as converting forces, relocating an actuation point, or converting a simple input such as rotation into a more complex motion. Mechanisms can also be combined to achieve compound capabilities. It should be noted that simple designs are generally preferred, as more parts require more maintenance, and also provide more opportunities for failure. Therefore, when investigating mechanical solutions, simpler systems should be considered first.

For analysis purposes, we assume that the building blocks of a mechanism are rigid bodies, meaning that no deformation is taken into account. We call these building blocks *links.* The geometry of the links and the way they are connected determine the behavior of the mechanism. Links have *nodes*, which are points of attachment to other links. The *order* of a link is the number of nodes it has. Links of different orders are shown at figure 2.1.

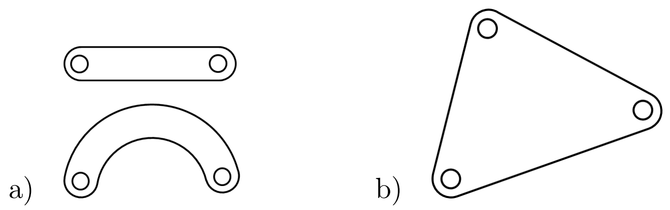


Figure 2.1: Links of different orders; a) Second order or *binary* links; b) Third order or *ternary* link.

A connection between links is called a *joints.* There are many types of joints, a few of which are presented at Figure 2.2. A high level categorization was proposed by Reuleaux [42]. He defines a *lower pair* as a connection where the contact locus is a surface, like the pivot (Figure 2.2 a) and the ball and socket joint (Figure 2.2 c). A *higher pair* is a connection where the locus of the contact is a line or a point, like the pin in slot joint (Figure 2.2 d). It should be noted that lower pairs in fact need a gap between the surfaces in order to be moveable. In practice, lower pairs have the advantage of trapping lubricant more effectively. This makes pivots especially interesting joints for their durability.

One may also want to distinguish between *force closed* joints, which need force applied to maintain the connection, like the ball and socket, and *form closed* joints which are held by their geometry, like the pivot, the prismatic joint and the pin in slot.

An important measure of mechanical mobility is the number of *degrees of freedom* or DOFs. They are the independent parameters whose value completely determine the state of an assembly. For instance, the pin joint is a 1 DOF joint, because only the angle between the links is required to determine its configuration. The pin in slot joint has 2 DOFs because the links can rotate with respect to each other and also translate in the axis parallel to the slot. In general, a free rigid body in two-dimensional space has three DOFs: two translational and one rotational (Figure 2.3 a). A free body in three-dimensional space has six DOFs: three translational and three rotational (Figure 2.3 b).

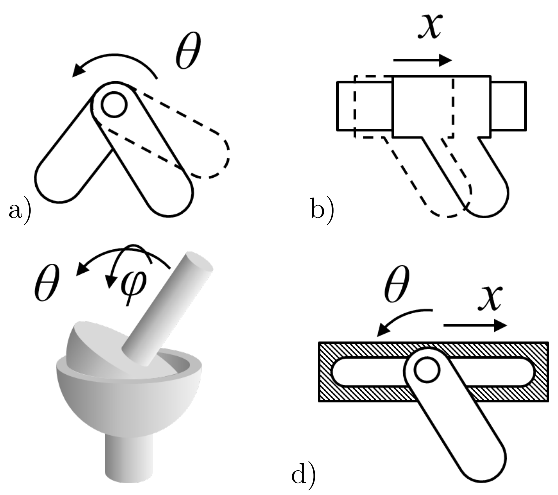


Figure 2.2: A few types of joints between links; a) The pivot is a one DOF rotational joint; b) The prismatic joint is a one DOF translational joint; c) The ball and socket joint is a 3D, force-closed, two DOFs rotational joint; d) The pin in slot is a form-closed, two DOFs rotational joint.

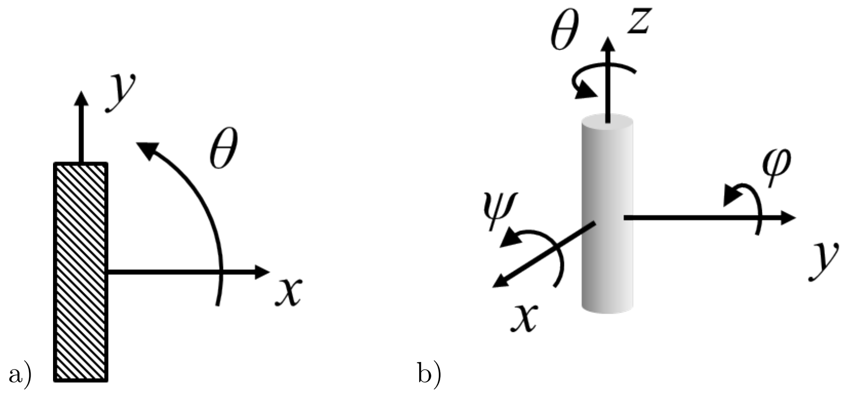
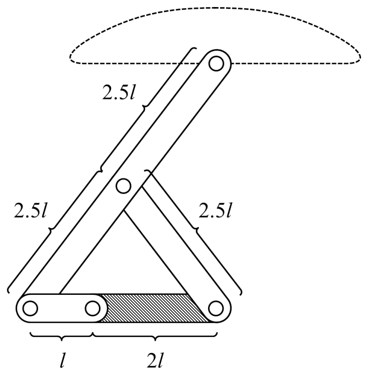


Figure 2.3: Representation of the degrees of freedom in; a) two-dimensional space; b) and three-dimensional space.

2.3 Linkages

A system of links connected by joints is called a *linkage*. This very broad definition encompasses almost any mechanism, but in practice the word *linkage* is mostly used to refer to assemblies of bars connected either by pivots or prismatic joints. Well-known linkages include the Chebyshev lambda mechanism (Figure 2.4 a), the pentagraph (Figure 2.4 b) or Theo Jansen's Strandbeest (Figure 2.5) [24, 51]. The analysis of these three linkages can be made in a two-dimensional reference frame, hence they are *planar* linkages. However, three-dimensional linkages also exist, but are much less documented because of their high complexity. A notice-able example is the *Agile Eye* [18] from Université Laval's laboratory of robotics, a spherical linkage. It should also be noted that many useful 3D assemblies can be constructed with 2D linkages in different planes.

a)


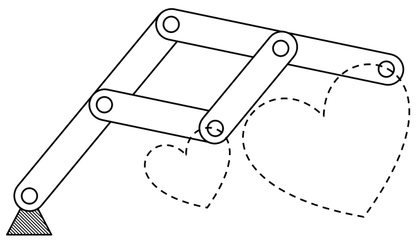
b)


Figure 2.4: Simple yet clever linkages; a) Chebyshev's lambda linkage can approximate a line at constant speed; b) The pentagraph duplicates a motion on a different scale.

a)

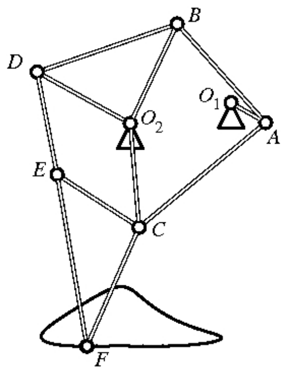
b)


Figure 2.5: a) The Strandbeest is a kinematic sculpture by Theo Jansen [24]; b) The legs [51] are eight-bar linkages which yield organic-looking walking motion.

Synthesizing even simple linkages remains a challenging field of research, as interactive speed for many useful applications has not yet been achieved. This section defines what linkages are and provides a basic analysis on their capabilities and mobility. Our discussion is focused on planar linkages. The two most common types of joints for such an assembly are the pivot and the prismatic joint.

A linkage is in a *closed chain* if its links are connected together in a loop. It is an *open chain* if it has any loose end. A linkage may be composed of several chains of any sort. Also, a link fixed with respect to the reference system is called the *frame* or *ground*.

The simplest mobile closed chain linkage is the four-bar linkage (Figure 2.7). It has been widely studied [35, 33, 17, 1, 6] in its form with four revolute joints, also called the 4R four-bar linkage. In this work, unless specified otherwise, the expression "four-bar linkage" refers to the 4R variant. Despite its simplicity, it is capable of complicated motion. Another well-studied variant has one prismatic joint and is called the *slider-crank* mechanism (Figure 2.6). This linkage is capable of converting a rotary motion to a linear motion or vice-versa. The four-bar linkage is widely present in modern machinery in both forms. It is also important because analysis of higher order linkages can be facilitated by detecting four-bar sub-components.

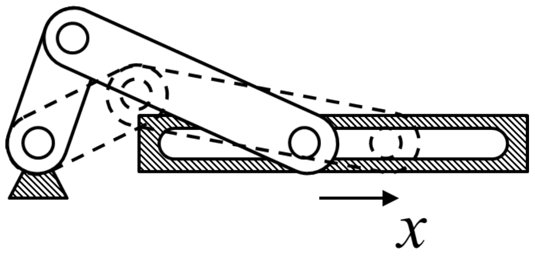


Figure 2.6: The slider-crank is an important variant of the four-bar linkage.

From Figure 2.7, we define the following notation four the four-bar linkage as used in this document:

- A point is noted $\mathbf{P}$, and its x- and y-coordinates are $\mathbf{P}_x$ and $\mathbf{P}_y$.
- A link joining points $\mathbf{P}$ and $\mathbf{Q}$ is noted $\mathbf{PQ}$
- The length of a link $\mathbf{PQ}$ is noted PQ

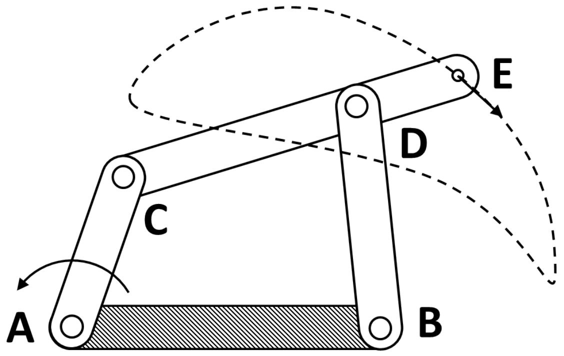


Figure 2.7: Four-bar linkage and the coupler curve traced by point $\mathbf{E}$.

Points $\mathbf{A}$ and $\mathbf{B}$ are fixed. Therefore, link $\mathbf{AB}$ is the *frame*, also called *ground* or *fixed link*. The motion of links $\mathbf{AC}$ and $\mathbf{BD}$ is limited to pure rotation. Under specific conditions discussed in the next section, either may or may not be able to make a full rotation. A link able to fully rotate with respect to the links it is connected to is called a *crank*. Otherwise, it is called a *rocker*.

Link **CD** couples link **AB** to link **BD** and is called the *coupler*. The motion of this link has the greatest complexity. Point **E** is fixed in the reference frame of link **CD**. The motion of this point traces a curve which we take as the output of the mechanism in this work. We refer to **E** as the end-effector. The curve traced is called the *coupler curve*. Examples of curves traceable with a four-bar linkage are shown at Figure 2.8.

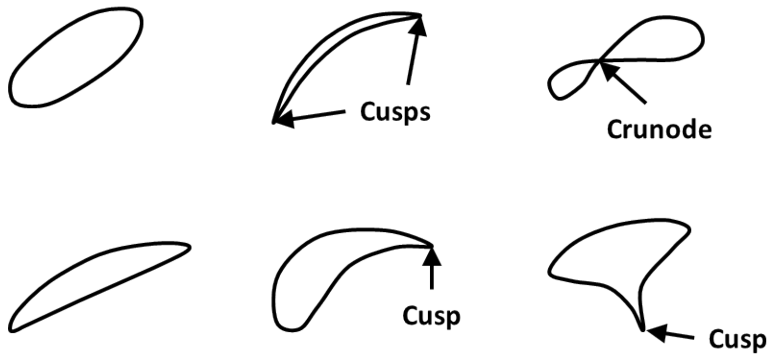


Figure 2.8: Examples of the different shapes that can be produced by four-bar linkages (inspired from Figure 3-16 part 1 and 2 of [35]). Notice that some curves may have cusps or crunodes (crossings).

In general, a four-bar linkage has 9 degrees of freedom. Three DOFs specify the position and orientation in 2D space: x, y and θ. Four DOFs specify the lengths of the bars: AB, AC, CD and BD. Two DOFs specify the position of **E** with respect to the pivots **C** and **D**. In this work, a slightly more constrained version of the four-bar linkage is considered. One DOF is removed by constraining **E** to be collinear with **C** and **D**. This allows developing an expression for the area of the coupler curve (see section 3.3).

The coupler curve of a four-bar linkage is a closed algebraic curve. A curve is said to be *algebraic* if its points are the zeros of a polynomial of two variables x and y. The *order* of the algebraic curve corresponds to the order of the polynomial, which is the highest sum of exponents of any of its terms. The coupler curve is a *sextic*, or a 6^{th} order curve. This also implies that the coupler curve of a four-bar linkage cannot pass more than six times through any single line nor any single circle [55].

The coupler curve of a four-bar linkage is *tricircular*. A p-circular curve has the algebraic property of passing p times through the complex points (1, i, 0) and (1, -i, 0) [16]. These points are known as the circular points at infinity. Complex points are points in the complex projective plane, which is the set of all points (z_1, z_2, z_3) where $z_1, z_2, z_3 \in \mathbb{C}$.

The coupler curve is also *trinodal*, meaning it can contain up to three *double points*, also called *singular points* or *nodes*. The double points are points with two tangents. They may be either of two types: *cusps* and *crunodes*. A cusp is a sharp turn at which the end-effector comes to a stop and then starts moving in a different direction. A *crunode* is a crossing in the curve reached at two separate times by the end-effector.

The general equation of the four-bar linkage coupler curve is known. The derivation is rather tedious and the results itself not so compact.

2.4 Grashof Condition

The lengths of the links relative to one another determine how the four-bar linkage behaves. Most importantly, the Grashof condition determines whether any link can perform a full rotation. The condition states that if the sum of the lengths of the smallest and longest links is smaller than or equal to the sum of the lengths of the two remaining links, then the smallest link can fully rotate relatively to both of its neighboring links. This can be expressed as follows:

$$s + l \leq p + q \tag{2.1}$$

where s and l are respectively the smallest and largest lengths, and p and q the remaining two lengths. The condition holds regardless of the order in which the links are assembled. If the Grashof condition holds but the two sides are not equal, the linkage belongs to class I. Three distinct behaviors are possible for class I four-bar linkages, depending on which link is the frame, or fixed. The cases are shown at Figure 2.9.

If the condition does not hold, it is a class II four-bar linkage. In that case, regardless of which link is the frame, the assembly is a double-rocker. Class III four-bar linkages satisfy the condition (2.1) with an equality. Linkage in class III have singular states where the linkage completely folds over itself (singularities are discussed in more detail at section 2.5.1). In such a state, it can unfold in two ways. This may be a problem when designing a linkage, as discussed in the next section. When the equality holds with bars of distinct lengths, there is one singular state. When their are two pairs of equal sides, there are two singular states. Finally, when all sides are equal, three singular states exist.

A linkage worth particular attention is the class I crank-rocker. This linkage is convenient because a continuous motor can actuate the linkage without locking. It is also the case for the double crank, but these linkages produce less interesting coupler curves. Indeed, because the two cranks tend to move together, fixed point on the coupler mostly travel circle-like paths. Double-cranks can be more inconvenient to design because all parts have to be on a different plane in order not to crash into one another at any point of the revolution.

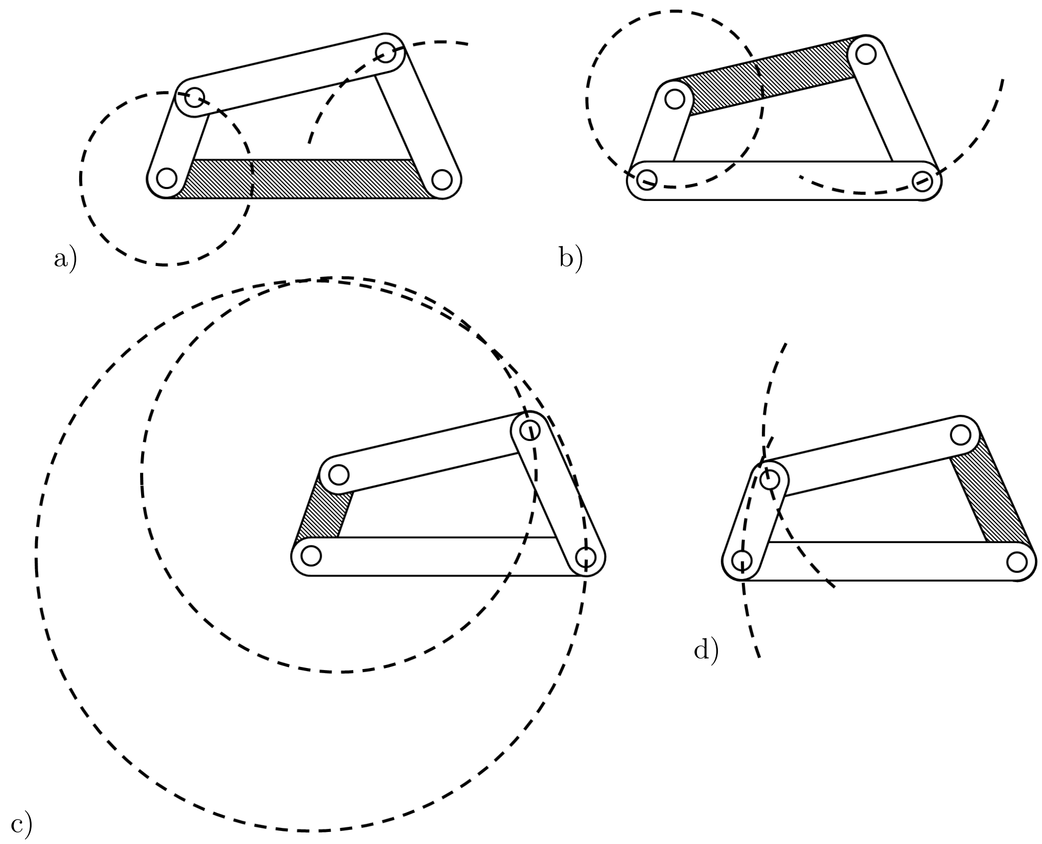


Figure 2.9: The three behaviors for class I four-bar linkages (inspired from Figure 2-15 in [35]). In a) and b), the shortest link is adjacent to the frame and the linkage is a crank-rocker. In c), the shortest link is the frame and the linkage is a double crank. In d), the shortest link is the coupler and the linkage is a double rocker.

2.5 Common Design Issues

Whenever a linkage is designed, it should be checked and tested for potential issues. Common mechanical problems inherent to linkages are singularities, order defect, branch defect and circuit defect. Also, many design methods do not take into account nearby parts the linkage may interfere with.

2.5.1 Singularities

Singularity analysis is a fundamental problem of mechanical design and has been widely studied (see [17] and references therein). For linkages, a singularity occurs when its behavior becomes indeterminate as a result of a particular alignment of the links. This happens when two connected links that are not directly controlled become collinear. Norton refers to such states as *toggle* positions [35]. Such states exist for all four-bar linkages in class III, or when the sum of the shortest and longest links is equal to the sum of the remaining two links. Figure 2.10

shows an example for a four-bar linkage. Singularities are generally undesirable because they require extra design effort to ensure full control at the singularity. Also, the range of positions about the singular state is typically associated with high acceleration in the linkage, which in practice results in high mechanical stresses in the links and joints.

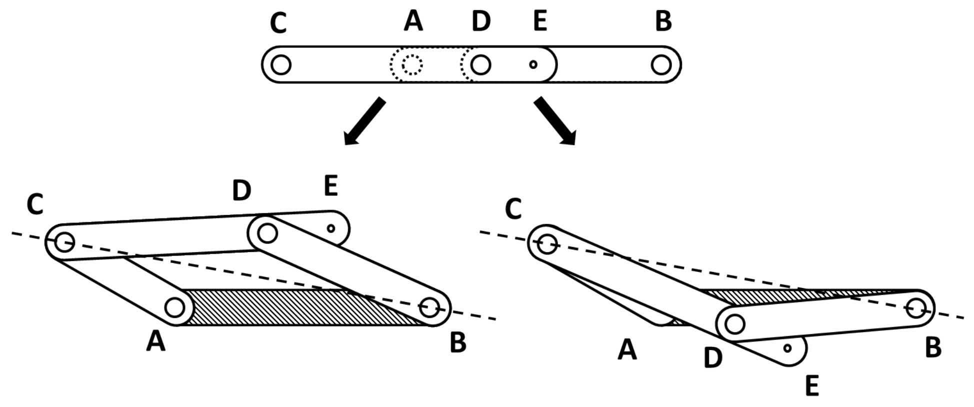


Figure 2.10: Two possible outcomes when the four-bar linkage is able to completely fold over itself. The crank is the link **AC**.

Not all alignments of links result in a singular state. If a singularity is unavoidable, an extra control feature is added to allow the linkage to follow through the singularity as desired. For instance, in piston engines, several linkages are coupled at different phase. Therefore, if one piston is in a singular state, at least one of the other pistons is not and can drive the first of singularity.

2.5.2 Order Defect

Most of the synthesis strategies begin from a set of precision points that the solution should pass through. These strategies, however, often do not constrain the behavior of the linkage between these points. Hence, the linkage may go through the points in an unwanted order. The synthesized linkage is then said to have an *order defect* [29, 3]. Whether or not the order of the points matters depends on the problem definition. It is an important factor to take into account when choosing a design method.

2.5.3 Circuit and Branch Defect

A linkage may be able to travel different curves depending on its original assembly. This is the case for the linkage presented on Figure 2.11. Such a linkage is said to have two *circuits*. A circuit is a set of all points reachable by continuous motion of the links. A linkage cannot change the circuit it is traveling on unless it is reassembled.

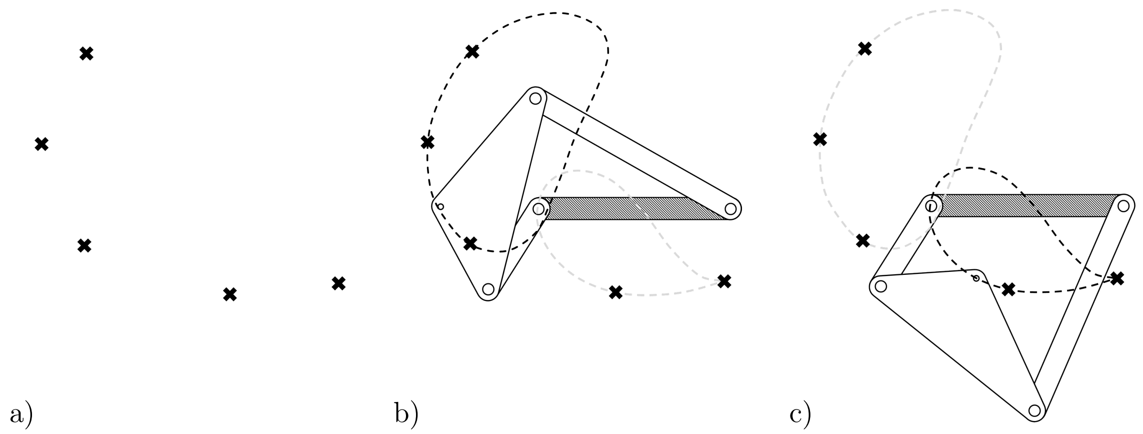


Figure 2.11: The linkage has a circuit defect because it cannot reach all precision points unless at least one pivot is detached and reattached ; a) Initial precision points; b) First circuit; c) Second circuit.

A *precision point*, in a linkage synthesis problem, is a point that must be reachable by the solution linkage. Not all linkage synthesis methods are able to discriminate circuits when fitting the precision points to a coupler curve. For instance, given the precision points presented at Figure 2.11 a), the linkage at Figure 2.11 b) and c) could be obtained. Such a solution is said to have a *circuit defect* [35, 29] because it cannot reach all precision points through continuous motion. Solutions with circuit defects are normally not valid and must be redesigned.

Simple simulation tools allow to visually detect such errors. In the case of class I and class II four-bar linkages, there are in general two circuits, depending on which side of the segment **BC** the links **CD** and **BD** are connected (Figure 2.12). Class III linkages may be seen as the limiting case where the four-bar linkage is able to toggle between the two configurations through the singular states discussed at section 2.5.1.

A circuit can be composed of one or more branches. Branches are segments of circuits separated by static states. A *static state* is a position at which torque applied on the crank in at least one direction does not accelerate the linkage. For instance, the aligned state of the linkage on Figure 2.10 is static, because no link moves if torque is applied at **A**. This occurs in class III linkages.

There is a *branching defect* in a synthesized linkage if not all precision points are reachable from the same configuration, meaning that the assembly does not go through all points when continuously actuated.

2.5.4 Other Issues

A design may be found unsuitable for a variety of reasons. Design methods often neglect a variety of contextual information. The location of pivots may be positioned at impractical

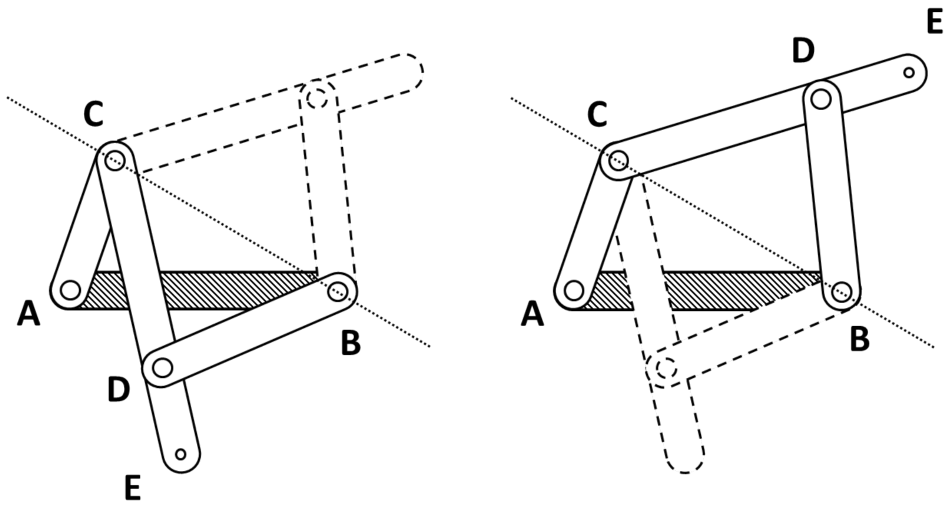


Figure 2.12: Two configurations of the same links sharing a symmetry about the line joining **B** and **C**.

locations, or the links may interfere with external components when moving. Such issues become apparent through simulation or prototyping.

In underconstrained problems, the parameters of the linkage may be adjusted to search for a better solution. For fully constrained or overconstrained cases, one can resort to the linkage *cognates* [22, 35]. The *cognates* of a linkage are linkages which have exactly the same coupler curve. Every four-bar linkage has two four-bar linkage cognates. They can be constructed geometrically as illustrated at Figure 2.13.

Also when designing the linkage in 2D, we ignore the fact that the parts must not overlap during actuation. Any pair of links that overlaps in the 2D projection of the linkage at any point in the motion must be in different planes. This includes all connected pairs, and also any two links connected to the shortest link. Overlap may turn out to be a puzzle to the designer, but it does not restrict the solution space.

No method completely guarantees obtaining a flawless design, hence the need for validation. Though numerical simulation is helpful, a prototype is often worth the cost.

2.6 Synthesis of a Four-Bar Linkage

2.6.1 Types of Mechanism Synthesis Problems

Norton [35] groups the linkage synthesis problems in three main categories: *function generation*, *path generation*, and *motion generation*.

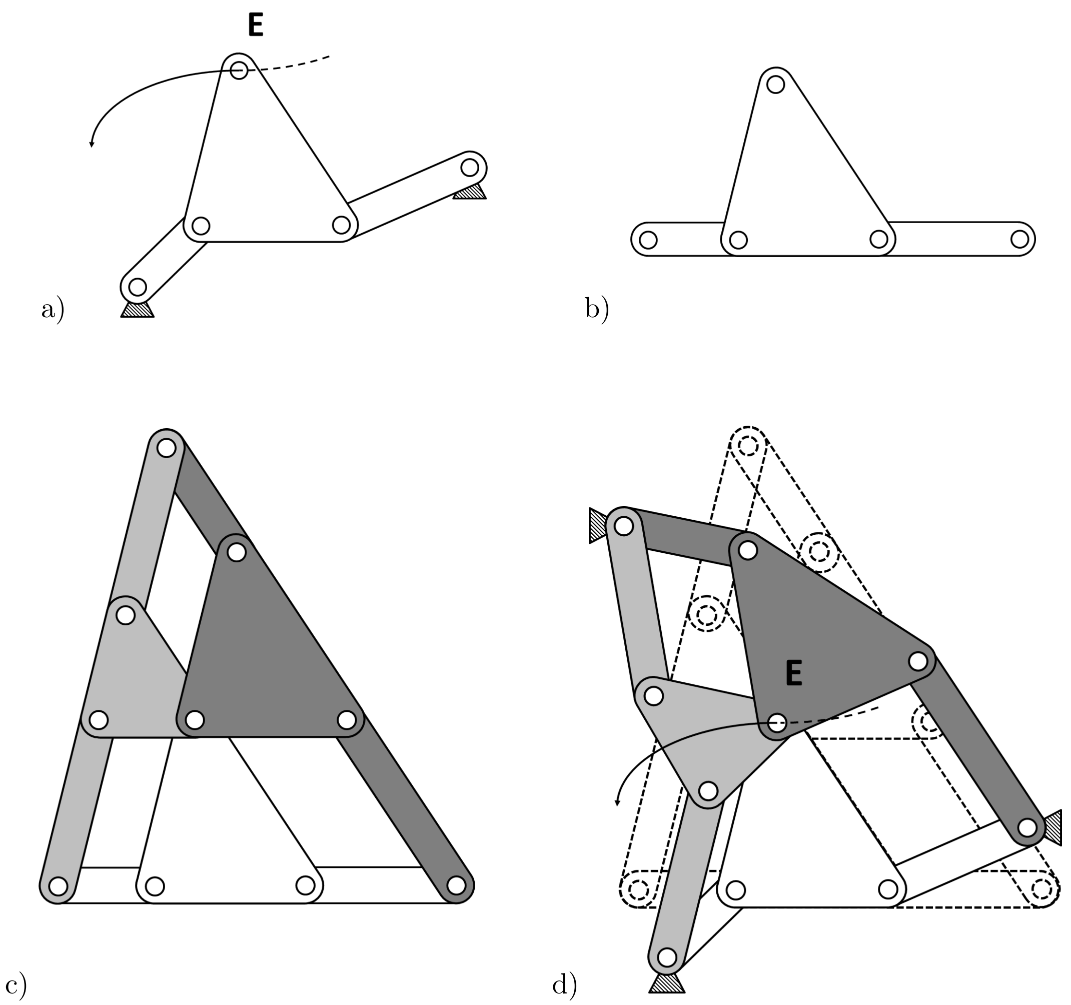


Figure 2.13: The cognates of a four-bar linkage can be constructed geometrically; a) Step 1: The original linkage; b) Step 2: The links are aligned; c) Six links are constructed along lines parallel to the linkage from b); d) The original pivots are repositioned to their original location. The third pivot moves to its location accordingly. Each cognate is represented in a different shade of gray. All three linkages produce the same coupler curve.

Function generation is achieved when the relation of the output of a mechanism to the input is some predetermined function (Figure 2.14 a). The actuation of such a mechanism is a form of calculation. With the low prices of numerical computers, this type of synthesis is losing popularity.

Path generation is achieved when a certain point of a mechanical assembly draws some predetermined curve (Figure 2.14 b). This type of problem is addressed in part II. We define two subcategories to this type of synthesis: precision point synthesis and continuous synthesis. The first aims at reaching a finite set of points, while the second is interested in recreating a whole continuous curve. The approach described in chapter 3 addresses both kinds.

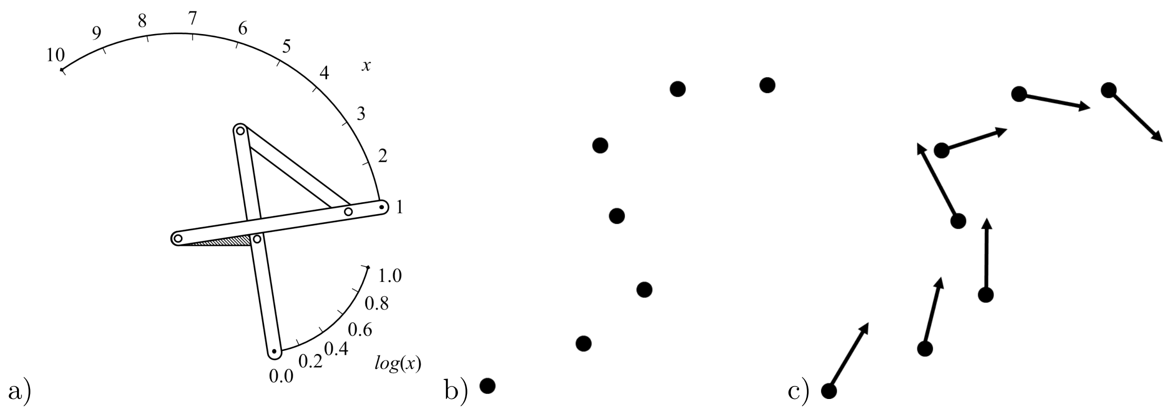


Figure 2.14: Three types of linkage synthesis problems as classified by Norton; a) Function generation problem (inspired from [38]); b) Path generation problem specified as a set of *precisions points* to be reached by the end-effector; c) Motion generation problem specified as a set of precision positions.

Motion generation is achieved when a certain line on a mechanism moves in a specified way (Figure 2.14 c). It can be interpreted as a path generation problem, where each precision point also has a prescribed orientation to it. Let us define a precision point with a specified orientation a *precision position*. A motion generation problem is defined by a set of precision positions. It is a problem more general than the path generation problem. Several simple cases can be solved geometrically or analytically.

A four-bar linkage is a great candidate mechanism for path generation as is is relatively simple, yet can achieve motion complex enough for a wide range of applications. Coupler curves can take many shapes with various interesting properties. For instance, Chebyshev's lambda mechanism (Figure 2.4 a) is famous for approximating a straight line at almost constant speed. Coupler curves with *cusps* have a brief stop in their motion. A cusp appears as a sharp angle in one point of the coupler curve. *Quick-return* mechanisms are also a common application of four-bar linkages. These mechanisms allow to have the largest portion of the cycle available for work, and swiftly move back to the starting position. This is useful for moving instruments above conveyers.

2.6.2 Overview of Existing Approaches

There are many approaches to linkage synthesis. The first that should be looked into are the graphical and analytical methods. These methods yield exact solutions and prove useful when just a few precision positions are specified. In the industry, more complex linkages are typically manually designed, using atlases like the Hrones and Nelson atlas [23] (see section 2.6.3) or CAD tools. Much energy is deployed in research to develop automated design tools. State of the art approaches using various optimization technologies are presented in this section.

For motion generation problems, the graphical method is practical for up to three positions [35]. Using numerical iterative methods, a solution satisfying up to five positions can be computed for the motion generation problem [35]. Kinzel et al. [25] propose a programming scheme able to solve 5 positions and 9 precision points.

The analytical methods yields exact solutions but requires computer assistance. Without iterative techniques, it is possible to solve up to 5 precision points. In general, any exact method is limited to 9 precision points, as there are 9 independent parameters to the four-bar linkage. However, not any set of nine points or less has a solution. Some experience about the capabilities of four-bar linkage may prove useful when choosing precision points.

Among the technologies investigated for automated approaches in related works, there are evolutionary algorithms [8, 27, 6], machine-learning algorithms [11] and algebraic algorithms [5].

2.6.3 Manual Approach

Manual approaches are the most common methods to specify, design, and evaluate linkages. Experienced engineers develop an intuition and may have access to histories of successful linkage designs. A variety of tools exist to facilitate the manual process. Here we explain how linkages can be designed with coupler curve atlases. Interactive software also exist [58, 34, 41] and serve the same purpose. They have the advantage of providing instantaneous simulation capabilities.

In 1951, Hrones and Nelson published the *Analysis of the Four-Bar Linkage* [23]. It consists of more than 700 pages, each illustrating 10 different coupler curves for a specific four-bar linkage instance. An example page is presented at Figure 2.15.

The atlas covers a wide variety of shapes. A user looking for a specific shape can then browse the collection and identify good potential starting points for design. Parameter optimization can then be performed via trial and error or using software. More recent adaptations following the same principle include the Atlas of the Four-Bar Linkage by Philip Todd et al. [53]. This atlas also has an interactive web implementation [48].

2.6.4 Graphical Approaches

The simplest, most intuitive, and also the most limited approach is the graphical synthesis of linkages. It allows the design of linkages using simply a ruler, a protractor, and a compass. This approach is practical in motion generation problems with up to three positions. The positions of the pivots may or may not be specified. It is intuitive and allows for quick identification of design issues. Norton exposes several graphical approaches for the following types of motion generation problems, taken from the table of contents of *Design of Machinery* [35]:

1. Two-Position Synthesis

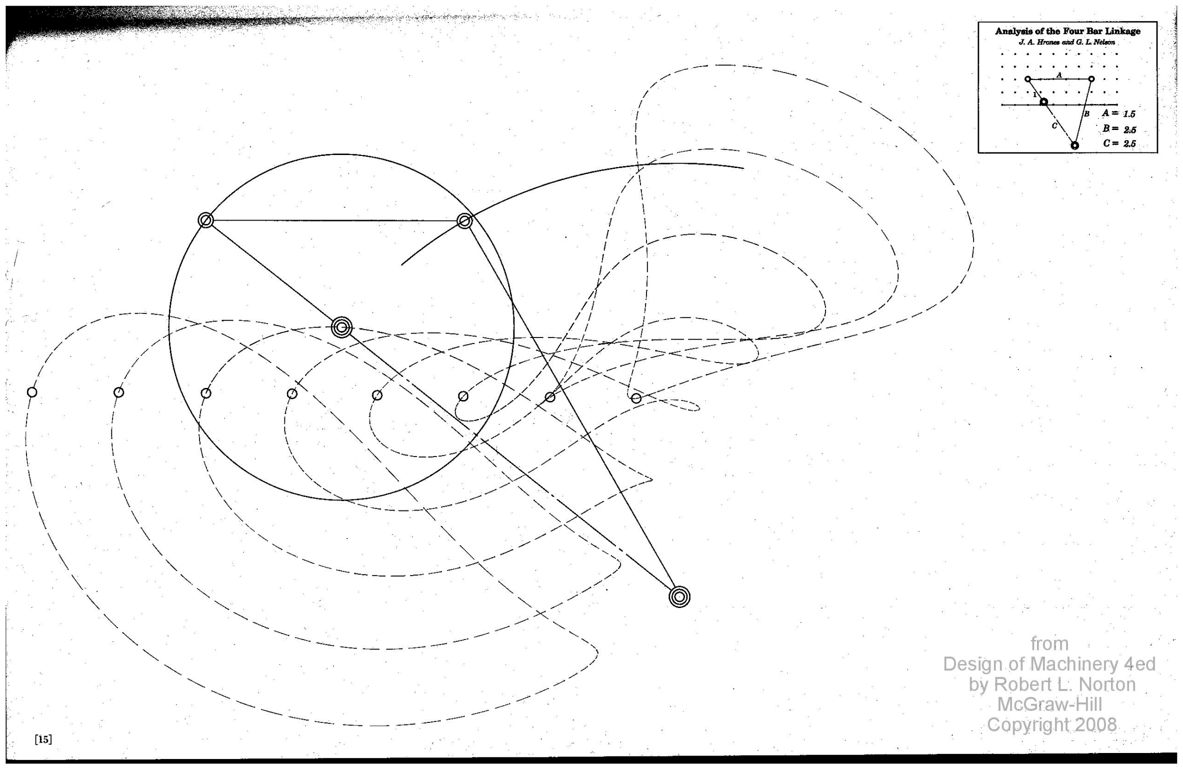


Figure 2.15: A page scanned from the Hrones and Nelson four-bar linkage atlas [23]. The top corner shows the links lengths ratios. Ten coupler curves are shown with the end-effector position relative to the coupler indicated by a hollow circle.

2. Three-Position Synthesis with Specified Moving Pivots
3. Three-Position Synthesis with Alternate Moving Pivots
4. Three-Position Synthesis with Specified Fixed Pivots

To illustrate the method, we solve a motion generation problem with three positions, and unspecified pivots. Let us represent the positions as arrows (Figure 2.16). The design satisfies the problem if an arrow drawn on the coupler can superimpose the original three arrows through continuous motion of the linkage.

We use the fact that a circle passing through three points is unique. We choose that point **B** must go through the root of the vectors, and point **C** through the head (Figure 2.17).

Points $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{C}_3$ lie on a circle whose center is fixed pivot **A**. Points $\mathbf{D}_1$, $\mathbf{D}_2$ and $\mathbf{D}_3$ lie on a circle whose center is fixed pivot **B** (Figure 2.18).

The linkage is completely defined (Figure 2.19).

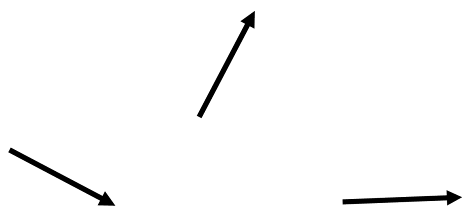


Figure 2.16: Step 1; The problem is defined by a set of vectors representing positions to match.

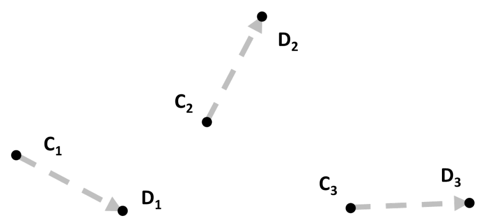


Figure 2.17: Step 2; The mobile pivots **C** and **D** are assigned respectively to the root and the head of the vectors.

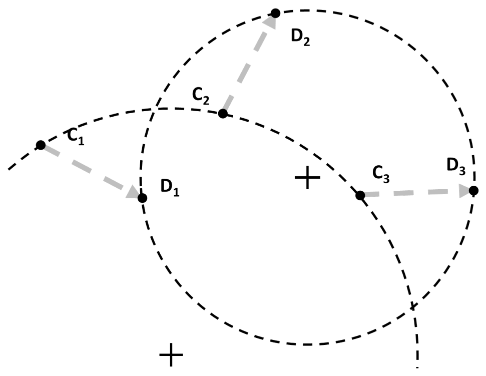


Figure 2.18: Step 3; The mobile pivots **C** and **D** are assigned respectively to the root and the head of the vectors.

The linkage shown at Figure 2.19 is unsuitable for utilization because is has a branch defect. Indeed, starting from the leftmost position, the linkage cannot reach the rightmost one without reassembly. This is not readily apparent from the drawing board, hence the importance of simulation and quick prototyping for validation.

To solve this problem, one can look for an alternate solution. Because there are fewer than five prescribed positions, the problem is underconstrained and there are therefore infinitely many linkages that satisfy the motion requirement. This means that some free choices can be made to determine the solution. In the previous process, the free choices were to assign the

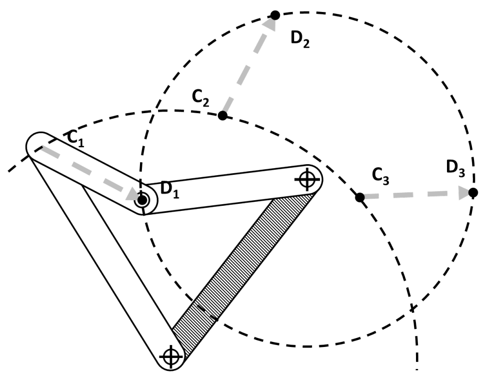


Figure 2.19: Step 4; The linkage is obtained.

B and **C** pivots to ends of the vectors. In fact, they can be assigned any two *different* points in a fixed referential to the moving vector. This variation of the graphical approach does not bring further relevant insight for this work and therefore is not discussed herein.

2.6.5 Analytical Approaches

Linkages can be represented by sets of equations using vector geometry. The links are represented as vectors starting and ending at pivots. For four-bar linkages, exact solutions exist for motion problems with up to five positions and for precision problems with up to nine points. This is because the four-bar linkage has nine DOFs. We focus on two methods for precision problems, depending on the number of points.

Five Precision Points or less

Direct analytical resolution is possible for up to five precision points [15, 45, 13]. A set of *loop equations* is obtained by forming imaginary vector loops between the precision points and the respective states of the linkage as shown at Figure 2.20.

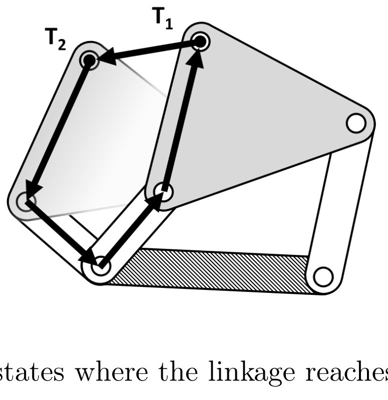


Figure 2.20: A loop between states where the linkage reaches precision points $\mathbf{T}_1$ and $\mathbf{T}_2$.

The maximum number of points for direct resolution is five. When fewer points are provided, the problem is underconstrained and some free choices need to be made. This may prove useful to control the timing of the motion for example, or specify convenient fixed pivot locations. Interactive tools such as the ones mentioned at section 2.6.3 allow quick visualization for designers.

More than Five Precision Points

To solve the synthesis problem for five to nine precision points, numerical methods are required to compute the solutions. It was only in 1992 that an exhaustive solution to the nine points problem was published [55]. The method uses polynomial continuation to find every solution satisfying an arbitrary set of nine points. The authors found that there is a maximum of 1442 non-degenerate solutions, found in the order of 100 CPU minutes. Including cognate linkages, this yields a total of 4326 solutions. In practice, many of these have complex parameter values. These solutions are discarded. Of the remaining ones, many prove impractical, as the analytical method is subject to branch defect and order defect. The unsuitable solutions can be detected and discarded, leaving a practical set to choose from. The number of solutions in the practical varies depending on the instances, and can be zero.

It is worth noting that not any set of nine points is solvable. For instance, Wampler et al. point out that no more than six points may lay on one line or one circle. This is because the equation of a four-bar linkage coupler curve is of the sixth order. Since the method is exhaustive, an unsolvable set of points yields an empty practical set.

2.6.6 Approximate Approaches

The graphical and analytical techniques previously discussed aim at solving problems where an exact solution can be found. However, there are many cases where this is not possible. Whenever more than nine parameters are fixed, the problem is overconstrained and in general no solution can be found. This does not mean that no solution fits the practical specifications of the problem. That is to say, there might exist a solution that is *good enough.* Numerical approaches can be used to find approximate solutions.

Optimization can be used to find the best solution to an overconstrained problem. It is typically a challenging task to define what a *good* solution is. An objective function is generally used to rate solutions. The best solution is the one for which the objective function is either minimal or maximal. Optimization is discussed in further detail at chapter 1. A problem that arises naturally in many applications is the synthesis of a linkage that fits an entirely specified path. We call it the *Path synthesis* problem. It can be viewed as a limiting case of the n-precision point synthesis problem, where n reaches infinity. Therefore, this problem is overconstrained, so there must be some strategy to relax the solution.

For the path synthesis problem, a good objective function should minimize the difference between the specified path and the solution path. However, measuring the similarity of curves is a challenging problem and many approaches exist.

The following approaches deal either with n-precision point problems where n may exceed 9 or the path synthesis problem.

Machine-Learning Approaches

Coros et al. [11] propose a path synthesis approach using machine learning. The resulting algorithm should allow non-experts to design linkages solely via specification of the desired output path. Specifically, the work is demonstrated via creation of animated toys. Many natural motions are cyclic and describe paths reproducible by linkages. Errors may arise because not any curve can be replicated by linkage coupler curves. Furthermore, depending on the method used, the input path may contain noise. However, approximation is sufficient here because organic motion is irregular and looks compelling to a human observer even if not sharply accurate.

Machine-learning is a good candidate for the path synthesis problem, as it is typically used when a quality metric is hard to define quantitatively. Optimizing similarity between curves is such a problem. While a human may feel comfortable telling whether curves are similar or not, choosing a quantitative metric is difficult and is an active research topic [11, 57, 26]. Common curve similarity metrics include the Hausdorff distance, the Fréchet distance or simply calculating the area between the two curves. Each have pros and cons and the best choice depends on the situation. In the case of linkage synthesis, a good similarity metric should be independent on translation, rotation and scaling. Indeed, these transformations can be easily applied to any linkage and do not change the intrinsic shape of the curve.

Coros et al. defined a path similarity metric based on six weighted scalar criteria that capture essential properties of the curve. Four of the criteria are geometric properties, namely the length of the curve, its area, its ellipticity, and its number of crossings. The remaining two criteria describe the position of the curve relative to the linkage. The authors define a center point and principal axes to both the curve and the linkage. The distance between the centers and the angle between the axes are used as criteria.

The weights of the scalar criteria are determined automatically, based on an optimization problem formulated by Xing et al. [57], because their relative importance is hard to quantify solely via reasoning. The training set is composed of a subset $\mathbf{S}$ of similar curves and a subset $\mathbf{D}$ of dissimilar curves. The optimization problem aims at setting the curves in $\mathbf{S}$ close together and the curves in $\mathbf{D}$ far apart. The subsets are built iteratively with user interaction. The user initially populates the sets with a few curves (eg. five pairs each, according to [11]). A first guess of the weights is computed by solving the optimization problem. Then, guessed pairs are presented the user, whom may label them based on their degree of similarity. Pairs labeled as *similar* are added to **S** and paired labeled as *dissimilar* are added to **D**. The number of pairs presented to the user per iteration is not specified by the authors. The updated sets are then used for the next iteration. This process is repeated until stable, reportedly after about 30 iterations.

A database of paths produced with different types of linkages – not only four-bar – is stored along with their values for the aforementioned criteria. For each type of linkage, the parameter space is explored using Poisson-disk sampling. This is so that the parameter space be represented as extensively and uniformly as possible in the database. A curve is added to the database only if is further than a certain distance to all other stored curves. This is done because similar curves may be produce even by radically different assemblies. The authors note that this method does not guarantee that all feasible subsets of the parameter space are explored.

When a user inputs a curve, the best match is found using the defined similarity metric. The linkage returned is taken as the starting point for a gradient descent parameter optimization. There is no guarantee that the solution obtained is a global optimum. A compromise has to be made on the density of the parameter space sampling so that good enough starting points are available for a wide variety of curves, while allowing to escape local optima.

This machine-learning approach for path synthesis allows interactive design of linkages by non-experts. However, the preprocessing is quite tedious, and the database memory requirement is potentially unlimited. Also, the solution space is limited to the range explored, so it not possible to guarantee global optimality. In the scope of the presented work, these limitations are not an issue, and the result is quite satisfying.

Genetic Algorithms

Because of their simple implementation and quick convergence, genetic algorithms have been implemented for linkage synthesis [8, 27, 6]. We refer the reader to section 1.3.3 for terminology and background notions.

In the referenced literature, the problem tackled is path generation using a set of n precision points. The same model is used for the four-bar linkage. The genes are the nine defining parameters of the linkage as well as n variables θ_i for the angle of the crank for each precision point. The population is composed of m individuals $\mathbf{x}_j$ with j from 1 to m. Each individual is an array of variables each representing a linkage.

The fitness metric f is the *sum of squared distances*, which is common also in other works on precision points. This metric is based on the idea that a linkage is a good solution if the end-effector **E** passes as closely as possible to every precision point. Therefore, at the position

θ_i, the distance between $\mathbf{E}$, encoded by two variables $\mathbf{E}_{xij}$ and $\mathbf{E}_{yij}$, and the corresponding precision point $\mathbf{T}_\text{i}$ should be as small as possible. The expression for this distance d_i is as follows:

$$d_i = \sqrt{(\mathbf{T}_{xi} - \mathbf{E}_{xji})^2 + (\mathbf{T}_{yi} - \mathbf{E}_{yji})^2} \tag{2.2}$$

Because it is a monotonous function, the square root can be removed as we want to minimize this quantity. By summing all the squared d_i, we obtain the metric of the sum of squared distances:

$$f(\mathbf{x}_j) = \sum_{i=1}^{m} (\mathbf{T}_{xi} - \mathbf{E}_{xji})^2 + (\mathbf{T}_{yi} - \mathbf{E}_{yji})^2 \tag{2.3}$$

Extra penalty terms may be added to account for critical constraints that are not implicitly accounted for by other means. For instance, Cabrera et al. [8] add two terms of the form $kh(\mathbf{x}_j)$ where k is a constant significantly larger than the range for f, and h is a function equal to 1 when an important constraint is not respected and equal to 0 otherwise. The first constraint is that the Grashof condition must be respected, and the second is that the angles θ_i must be increasing or decreasing.

Among the proposed genetic algorithms, differential evolution stands out as the preferred approach. It is a reproduction scheme where all individuals of a population reproduce with a made-up individual, called *disturbing vector*. The disturbing vector $\mathbf{v}$ is a combination of the best individuals and the difference between two randomly chosen individuals:

$$\begin{aligned} \mathbf{v} &= \mathbf{x}_i + a(\mathbf{x}_j - \mathbf{x}_k) && (2.4)\\ \text{where } i, j, k &\in \{1, m\} && (2.5)\\ i &= \underset{i \in \{1,m\}}{\arg\max} f(\mathbf{x}_i) && (2.6)\\ i &\neq j \neq k && (2.7) \end{aligned}$$

The value of the constant a is chosen by the user. The first term ensures that the best features of the population are present in the disturbing vector. The second term is large when the population is very dissimilar, because randomly chosen individuals have very dissimilar genes. However, as better individuals emerge and the population converges, the term reaches 0. The algorithms are reported to converge between tens to a few hundred iterations, which typically corresponds to a few seconds or more, depending on the size of the population chosen.

Good results were obtained by the means of genetic algorithms. The method is especially interesting for its capacity to deal with very noisy input with larger numbers of precision points (a few dozens). It is also a clear advantage that the implementation does not require any exotic software and can be rather simply implemented in free languages such as Python [43]. However, from our own experiments, we found that the approach is rather sensitive to local optima and often converges to non-zero fitness, even in cases where exact solutions exist (optimal fitness is 0).

Graph-Grammar Approach

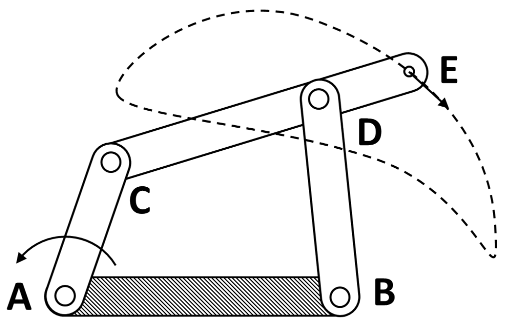
A major challenge of planar mechanism design is that the variety of possible systems is unlimited. There is no theoretical limit to the number of links that can be part of a linkage. Exploring the general solution space of all possible planar mechanisms to solve a problem is therefore a daunting task, and therefore most linkage synthesis methods limit themselves to some basic yet important types.

Graph-grammar approaches attempt to not limit the search to a finite and predetermined number of mechanism types. They generally start from trivial solutions which are incremented using a set of rules. All valid candidate assemblies are stored. The search for the type of mechanism is conducted first, then the parameters are optimized. Problems inherent to this type of search are *isomorphism* and *confluence*, occurring when the same mechanism is generated by different sequences of rules.

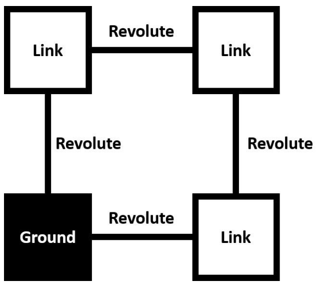
The linkages are represented as graphs, composed of nodes connected by edges. Both nodes and edges may have labels attached to them, encoding a variety of information such as kinematic and dynamic properties or indices. Figure 2.21 shows three examples of graph representations of the four-bar linkage found in the literature [40, 47, 49].

The representation proposed by Schmidt et al. [47] (example at Figure 2.21 a) is mostly interested in representing link and joints arrangements rather than solving for a specific output. They represent links by nodes, and the joints by labeled edges. Their approach can be used to generate atlases. An interesting contribution they make is to propose a linear algorithm to detect isomorphic graphs, or mechanisms that are equivalent in practice. This prevents redundant exploration.

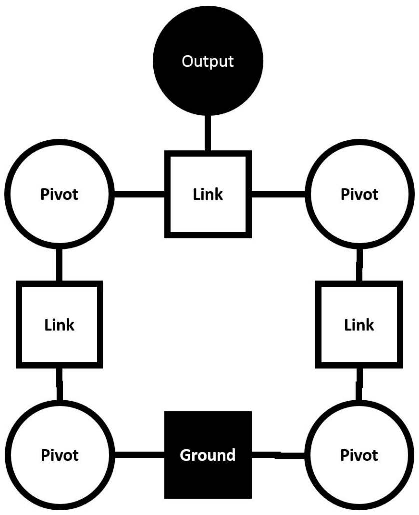
Stöckli and Shea [49] proposed a graph representation and a rule-set for generating possible solutions for a passive dynamic brachiating system. Passive dynamic mechanisms draw energy from their surroundings to move and do not require a power source. Brachiation is the action of moving by swinging from one arm to the other, like monkeys in trees. In their representation, both links and pivots are represented by nodes, linked by edges in the same sequence as the physical assembly. The graph-grammar approach serves to find possible assemblies, and then other methods are used to optimize the parameters, namely genetic algorithms.



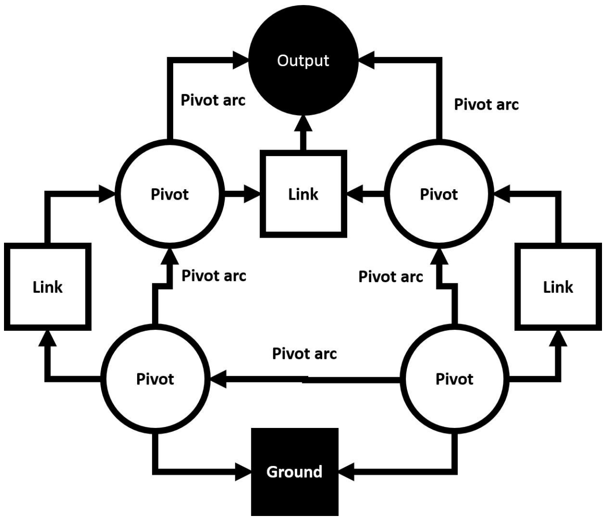
a)



b)



c)



d)

Figure 2.21: The four-bar linkage as represented in three proposed approaches; a) Reminder of the graphical representation of the four-bar linkage; b) Schmidt et al. [47]; c) Stöckli and Shea [49]; d) Radhakrishnan and Campbell [40]

Radhakrishnan and Campbell's graph representation [40] is similar to the previous one, though a little more elaborate. It includes edges joining pivots on the same link, and all edges are directed. The directed edges provide interesting properties to reduce confluence problems. Radhakrishnan provides a set of 16 rules in his thesis [39] to generate a broad range of planar linkages. Similarly to the previous work presented, valid assemblies are first generated and stored, then optimized using a third-party toolbox.

Each approach also comes with a rule-set. A rule specifies a certain transformation that increases the complexity of the assembly. A rule has prerequisites, which is a certain pattern to be detected in the graph. It then defines what transformation to apply to the prerequisite elements, whether adding nodes, changing labels or replacing nodes. For instance, in the work of Radhakrishnan and Campbell, the first rule applied is always the seeding rule. It can be applied on two nodes with label *ground* and *output*. It creates a link and a pivot to connect the output to the ground. The process is illustrated at Figure 2.22.

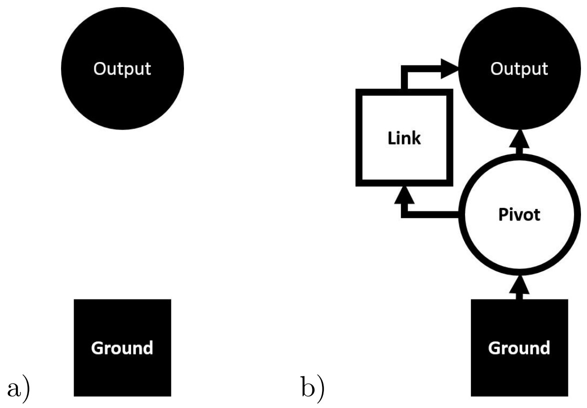


Figure 2.22: Example of a rule defined by; a) an initial pattern and; b) a transformed pattern

When searching for potential assemblies, the algorithms start with a basic type and look for all opportunities to apply a rule. A tree of different possibilities is constructed by iteratively applying this principle to the child assemblies up to a specified depth.

Graph-grammar approaches are an extension to other synthesis approaches which arbitrarily extend the types of mechanisms explored via their rule-set. They are fairly more resource-consuming because the generation step grows exponentially with the depth of the search, and the kinematic evaluation requires generic algorithms that are not as efficient as specialized analytic expressions.

Using the Analytical Coupler Curve Expression

As mentioned in sec 2.3, the analytical expression for the four-bar linkage coupler curve is known and can be exploited to synthesize linkages. It is a tricircular, trinodal and sextic curve. These terms are not explained herein as they refer to advanced geometric properties that are not used in the core work of this document. Blechschmidt and Uicker [5] devised an algorithm exploiting these geometrical properties of the the coupler curve. A planar algebraic curve can be generally expressed as follows:

$$f(x,y) = \sum_{i,j=0}^{n} A_{i,j} x^i y^j = 0 \qquad \text{for } i+j \leq n \tag{2.8}$$

where n is the *order* of the curve. A sixth order or sextic curve has 27 coefficients $A_{i,j}$, which are non-linear functions of the nine four-bar linkage parameters. Blechschmidt and Uicker describe how the tricircular and trinodal properties of the curve allow to build a linear equation system, which can be solved when the double points, singular foci and at least one ordinary point from the curve are known. We refer the reader to the original article for more detail on the nature of these points. They are not known *a priori*, but can be chosen from a set of possible options computed from the precision points.

When the coupler curve equation coefficients are obtained, the linkage parameters are obtained using an iterative technique. A first guess is found using exhaustive search, then a gradient descent converges to the final design.

Solving the linkage parameters is very time consuming. However, the quality of the curve can be evaluated before the solving is undertaken. This allows to save time in case a redesign is needed.

2.7 Other Significant Linkages

As previously noted, simpler assemblies should always be prioritized when designing a mechanical solution. Simple mechanisms, however, have limited capabilities that may not be sufficient for more complex problems. Here we briefly discuss significant n-bar linkages, with n higher than four. This is relevant for future work as the four-bar linkage is a milestone towards solving more complex linkages. Many ideas and elements of the research presented in part II presented can be generalized. This section gives further insight on the five-bar linkage, the six-bar linkage and general n-bar linkages.

The five-bar is obtained by adding one link in a four link loop. Doing so, an extra DOF is added. Therefore, an end-effector positioned on one of the couplers of a free-moving five-bar linkage can reach any point in a two DOFs workspace (Figure 2.23 a). For the design problems mentioned at section 2.6, one DOF is required. To make it into a one DOF assembly, the two links attached to the frame are typically geared (Figure 2.23 b). Curves of higher complexity can be traced by geared five-bar linkages.

It is possible to construct six-bar linkage with one DOF. The Stephenson and Watt linkages are such examples. They can be thought of as combinations of four-bar linkage. This is also the case for many higher order mechanisms. In fact, detecting four-bar sub-components in a linkage is a useful technique for automatic evaluation the kinematics of complex assemblies. Ting et al. [52] allowed to generalize the classes defined from the Grashof (see section 2.4) condition for n-bar linkages as follows:

$$\text{Class I}: \quad l_n + (l_1 + l_2 + ... + l_{n-3}) < l_{n-2} + l_{n-1} \tag{2.9}$$

$$\text{Class II}: \quad l_n + (l_1 + l_2 + ... + l_{n-3}) > l_{n-2} + l_{n-1} \tag{2.10}$$

$$\text{Class III}: \quad l_n + (l_1 + l_2 + ... + l_{n-3}) = l_{n-2} + l_{n-1} \tag{2.11}$$

where l_i is the length of the i^{th} shortest link. Therefore, l_0 is the shortest length and l_n the

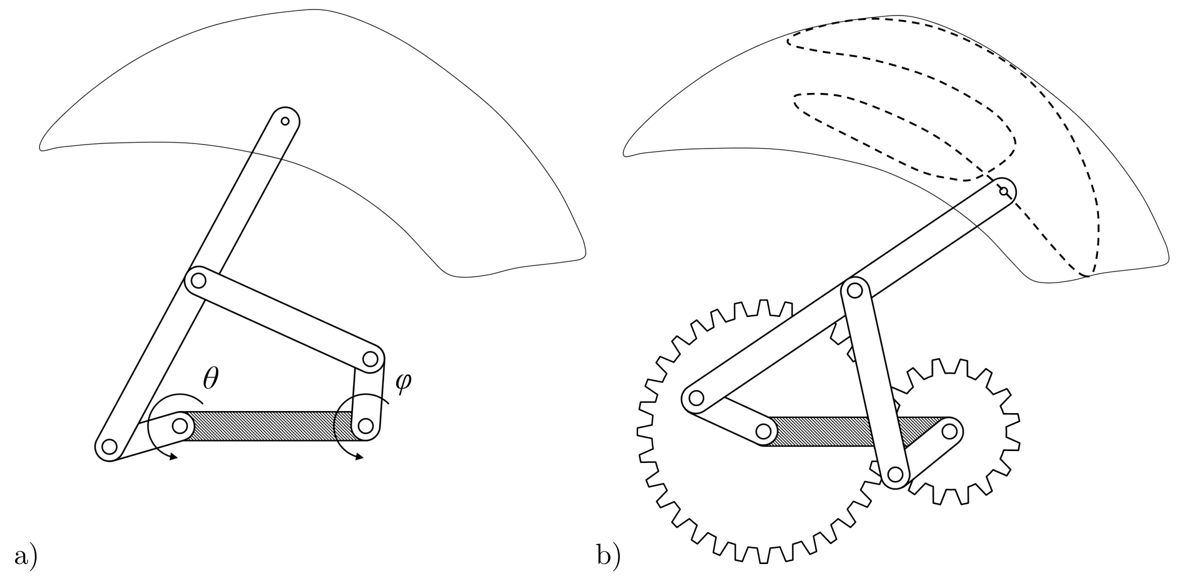


Figure 2.23: a) The five-bar linkage with two DOFs and its workspace; b) One DOF can be removed by gearing fixed pivots. The resulting curve is contained within the workspace.

largest.

2.8 Conclusion

Mechanical linkages are widespread in modern machinery. They have been widely studied and much theory is available on their subject. However, the non-linearity of the equations governing their kinematics are still challenging to solve for significant design problems such as path generation and motion generation.

The four-bar linkage is the simplest closed-loop moveable linkage. By choosing an arbitrary point on one of its links, the coupler, complex curves can be traced. We have reviewed existing path synthesis methods to find linkages whose coupler curve match a given set of points. The most popular approaches even nowadays rely on the experience of the designer, but more and more automatic approaches are brought forward. Up to a certain limit, exact approaches can be used, and for harder cases approximate approaches have to be used. Among other technologies, machine-learning, evolutionary and graph-grammar algorithms have been used. Despite the many interesting advantages of these approaches, there still seems to be room for improvement in terms of combined speed and quality of solutions generated.

The next part describes the main work of this project, where we explored non-convex optimization to develop a path synthesis tool for four-bar linkages.

Part II

Four-Bar Linkage Synthesis Using Non-Convex Optimization

Chapter 3

Non-Convex Optimization Design Method

We developed a method to effectively design four-bar linkages outputting a desired curve using non-convex optimization. The benefits of this application are that the synthesis of the continuous curve is accurate, fast, and deterministic. We modeled the mechanism using its geometric properties, keeping in mind the possible generalization to mechanisms of higher complexity, and a novel cut (also called a redundant constraint) was developed using the area of the curve. We designed a novel path sampling technique. We implemented this strategy in a simple design software. The work presented in this part was published in collaboration with Autodesk Research as a paper at CP 2016 [19].

The input of the problem is a curve specified by user. The output is an array of parameters specifying a *collinear* four-bar linkage that closely reproduces the input curve. We use the term collinear to refer to a special case of four-bar linkage where the end-effector is aligned with the pivots of the coupler — or that points **C**, **D**, and **E** are aligned (see Figure 2.7 for the notation). The problem is implemented as a non-linear program. The objective function is specified so as to minimize the distance between the input curve and the four-bar linkage coupler curve.

At section 3.1, we describe a sampling technique applied on the input curve. The section 3.2 describes the non-linear model developed as well as a distance metric used to evaluate the quality of the solution curve. Section 3.3 presents a redundant constraint on area, and a mathematical proof of the constraint follows at section 3.4. Finally, section 3.5 introduces the design software developed for experimentation.

3.1 Curve Sampling Technique

The input curve is pre-sampled into a high resolution array of coordinates. The sampling process consists of choosing n points from this array. The sampled points are used afterward as precision point in a path generation problem (see section 2.6.1 for description of the path generation problem). Here we propose a strategy to choose the n precision points $T_1, \dots, T_n$ that best represent the input curve. We call the number n of precision points the *sample number*. A compromise needs to be made when choosing the sample number. Indeed, a large sample number increase the execution time. However, it also improves the quality of the approximation, as the continuous curve is better represented. The quality metric used is described in detail further at section 3.2.3.

Some points are more important than others, like cusps and other sharp turns. We call these points of interest *features*. Figure 3.1 shows a curve with features and one without. It is also important to have some precision points between the features to depict the general behavior of the curve. The remaining precision points are spread evenly between the features. It is possible for a curve to have no feature (e.g. an ellipse). In this case, a first precision point is placed at the point of maximal curvature. The remaining precision points are spread evenly (Figure 3.1).

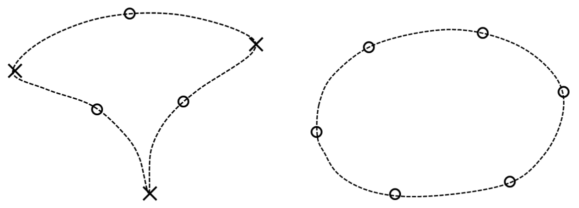


Figure 3.1: Sampling technique: features are marked by $\times$ and remaining points by $\circ$.

To identify the features, we first compute all maxima of curvature. However, we do not use the curvature as typically defined in geometry, as it approaches infinity at cusps and reaches inconveniently high values at very sharp turns. Instead, as shown on Figure 3.2, we compute the squared change in angle θ^2 between segments of the high resolution pre-sampled array of the curve. Two segments will never have a deviation of more than 180°. We square θ to amplify the variation.

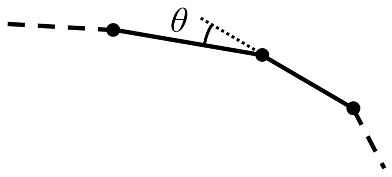


Figure 3.2: The deviation θ between consecutive segments

We compute the median absolute deviation from all squared angles θ^2. Using the first derivative of θ^2 with respect to the distance traveled on the curve, we identify the local maxima. There are usually many extrema, and filtering is needed. We keep only the extrema whose θ value is significantly greater than the overall values over the curve. Experiments have shown that filtering out data within 10 times the median absolute deviation yields satisfactory results. Algorithm 2 presents the equivalent pseudocode.

Algorithm 2 Feature filtering($\vec{x}, \vec{y}$)

```
1: $\Theta \leftarrow \{\theta_i^2 \mid \theta_i \text{ is the exterior angle at } (x_i, y_i)\}$
2: $\hat{\Theta} \leftarrow \{\theta_i^2 \in \Theta \mid \theta_{i-1}^2 < \theta_i^2 > \theta_{i+1}^2\}$
3: $m \leftarrow \text{median}(\Theta)$
4: $d \leftarrow \text{median}\{|\theta_i^2 - m| \mid \theta_i^2 \in \Theta\}$
5: return $\{(x_i, y_i) \mid \theta_i^2 \in \hat{\Theta} \wedge \theta_i^2 > m + 10d\}$
```

3.2 Model

The model ensures that the effector **E** moves as close as possible to the target curve. It minimizes the distance when the effector passes by each of the n precision points. In other words, the solver has to find a mechanism and compute n positions for this mechanism. Each position is associated with one precision point. The objective function is defined so as to minimize the distance between **E** and the precision point for each position. This section describes the variables, constraints and objective function that compose the model.

3.2.1 Variables

A collinear four-bar linkage is defined by eight parameters, which are the x- and y-coordinates of pivots **A** and **B**, the lengths of links **AC**, **BD**, and **CD**, and the distance from **C** to **E**. The variable for the length between two points such as **AC** is denoted AC. The solved linkage is interpreted directly from the values of these variables. We define two more redundant variables. The variable AB represents the distance between **A** and **B**. The variable w gives the ratio of length CE over CD.

We add to the model variables for the x- and y-coordinates of **C**, **D**, and **E** for each precision point, for a total of $6n$ variables. A single error variable e represents the maximum of all distances between effector positions $\mathbf{E}_i$ and their corresponding precision point $\mathbf{T}_i$. The positions of the linkage are unordered. Therefore the correspondence between $\mathbf{E}_i$ and $\mathbf{T}_i$ is arbitrary.

All variables for coordinates are bounded from -10 to 10. Link lengths are bounded from 0 to 10. The ratio w is bounded from 0 to 5. It is helpful for the filtering of the solver to bound the domain of e with the upper error bound e_u.

Type	Variables	Domains	Quantity
Defining parameters	A_x, A_y, B_x, B_y	$[-10, 10]$	4
	$AB,\ AC,\ BD,\ CD,\ CE$	$[0, 10]$	5
	w	$[0, 5]$	1
Position parameters	$C_{xi}, C_{yi}, D_{xi}, D_{yi}, E_{xi}, E_{yi}$	$[-10, 10]$	$6n$
Error	e	$[0, e_{\mathrm{u}}]$	1

Table 3.1: Variables for four-bar linkage model

The information on the variables is gathered in Table 3.1.

3.2.2 Constraints

Constraints are relationships between the variables. The solver must find values for the variables to satisfy all constraints. Here we explain all the constraints of our model.

The first set of constraints forces the coordinates of the pivots to be separated by distances corresponding to the lengths of the bars. For example, for the crank **AC** we have:

$$(A_x - C_{xi})^2 + (A_y - C_{yi})^2 = AC^2 \qquad \forall\, i \in [1, n] \tag{3.1}$$

We use a similar constraint to define the error e as the upper bound of the squared distance from end-effector position $\mathbf{E}_i$ to precision point $\mathbf{T}_i$.

$$(T_{xi} - E_{xi})^2 + (T_{yi} - E_{yi})^2 \leq e \qquad \forall\, i \in [1, n] \tag{3.2}$$

The following constraints ensure that the pivots **C**, **D**, and **E** are collinear. We use the fact that the components of vectors **CE** and **CD** respect the ratio w.

$$w \cdot (D_{xi} - C_{xi}) = E_{xi} - C_{xi} \qquad \forall\, i \in [1, n] \tag{3.3}$$

$$w \cdot (D_{yi} - C_{yi}) = E_{yi} - C_{yi} \qquad \forall\, i \in [1, n] \tag{3.4}$$

The lengths of the bars are not sufficient to determine the configuration of the mechanism. As shown in Figure 2.12, the same bars can be arranged into two distinct mechanisms. The two solutions share a symmetry along the segment joining **B** and **C**. For each precision point $\mathbf{T}_i$, the coordinates for $\mathbf{E}_i$ have to be on the same side of the line going through $\mathbf{BC}_i$. Since $\mathbf{T}_i$ and $\mathbf{E}_i$ are expected to lie close to one another, the coordinates of either can be used equivalently. The cross-product of vectors **BC** and **BE** changes sign depending on which side of **BC** the point **E** is. By constraining the sign of the cross-product to be the same for all positions, we constrain the configuration. We therefore add either of the next two constraints.

$$(T_{xi} - C_{xi})(B_y - C_{yi}) \geq (T_{yi} - C_{yi})(B_x - C_{xi}) \qquad \forall\, i \in [1, n] \tag{3.5}$$

$$(T_{xi} - C_{xi})(B_y - C_{yi}) < (T_{yi} - C_{yi})(B_x - C_{xi}) \qquad \forall\, i \in [1, n] \tag{3.6}$$

The two constraints are mutually exclusive, so a model may represent only one configuration at a time. To access the whole search space, we can run the two configurations in parallel. We term them the *left* and *right* configurations, according to the inequality sign.

As discussed at section 2.4, the Grashof condition [12] states that the shortest link in a four-bar linkage can fully rotate only if the combined length of the shortest and longest links is smaller than the combined length of the remaining two links. We force the crank **AC** to be the smallest link with inequalities (3.7), to be sure it can always fully rotate. However, since any of the three other links may be the longest one, we need the three inequalities from (3.8) to (3.10). This is equivalent to the Grashof condition in all cases, because the inequality where the longest link is on the same side as the crank is always the most constraining.

$$AB \geq AC \qquad BD \geq AC \qquad CD \geq AC \tag{3.7}$$

$$CD + BD \geq AC + AB + s \tag{3.8}$$

$$AB + CD \geq AC + BD + s \tag{3.9}$$

$$AB + BD \geq AC + CD + s \tag{3.10}$$

A security constant s is added to the three last constraints to avoid equality. Otherwise, the mechanism could become a class III four-bar linkage and have at least one singular state. Singularities are considered undesirable as discussed at section 2.5.1 as they require additional control and involve high mechanical stress. The security constant s can be tuned by the user to the desired tolerance. For all experiments herein, s was set to 0.1 to minimally reduce the search space while preventing singularities.

It is worth noting that the model does not require the solution found to follow the precision points in any order. Therefore, the solver could return a mechanism which goes through the precision points in an undesired order. However, this is unlikely for two reasons. First, part of the sampling is done by placing precision points between those identifying the features. This suggests a continuity between the points which the solutions tend to adopt naturally. Second, violating the order of the points generally results in a significant change in the area of the curve, which is constrained as discussed in section 3.3.

3.2.3 Objective Function

The overall goal is to minimize the distance between the two continuous curves, using a continuous metric Q defined in this section. Implementing Q in the model would require

Constraint	Quantity
$(A_x - B_x)^2 + (A_y - B_y)^2 = AB^2$	1
$(A_x - C_{xi})^2 + (A_y - C_{yi})^2 = AC^2$	n
$(B_x - D_{xi})^2 + (B_y - D_{yi})^2 = BD^2$	n
$(C_{xi} - E_{xi})^2 + (C_{yi} - E_{yi})^2 = CE^2$	n
$w \cdot (D_{xi} - C_{xi}) = E_{xi} - C_{xi}$	n
$w \cdot (D_{yi} - C_{yi}) = E_{yi} - C_{yi}$	n
$AB \geq AC$	1
$BD \geq AC$	1
$CD \geq AC$	1
$CD + BD \geq AC + AB + s$	1
$AB + CD \geq AC + BD + s$	1
$AB + BD \geq AC + CD + s$	1
$(T_{xi} - C_{xi}) \cdot (B_y - C_{yi}) \lesseqgtr (T_{yi} - C_{yi}) \cdot (B_x - C_{xi})$	n
$(T_{xi} - E_{xi})^2 + (T_{yi} - E_{yi})^2 \leq e$	n

Table 3.2: List of constraints for the four-bar linkage model

approximating the curve with a very large number of points. To avoid enlarging the model, we have opted for using only carefully selected points to approximate the curve.

Continuous Metric

The continuous metric is not evaluated by the solver, but is used at a higher level by the design application discussed at section 3.5. The objective implemented in the model is derived from the continuous metric.

Several well-established curve matching metrics exist. The Hausdorff distance [21] d is the greatest distance from any point on the curves to the closest point on the other curve. To make the metric independent of the size of the curves, we normalize it with the greatest x- or y-dimension of the curve. The normalized Hausdorff distance is herein designated as Q. In compliance with Fig. 3.3, the equation for Q is:

$$Q = \frac{d}{\max(\Delta x, \Delta y)} \tag{3.11}$$

Figure 3.4 shows matching curves yielding different Q values. A Q value of 0 is a perfect match. The user can define a threshold T under which the curves are considered a good enough match. It is worth noting that Q does not take into account the course of the curve, which might result in undesired matches, especially when a curve self-crosses. However, features of the model such as the area constraint discussed in section 3.3 make these events unlikely.

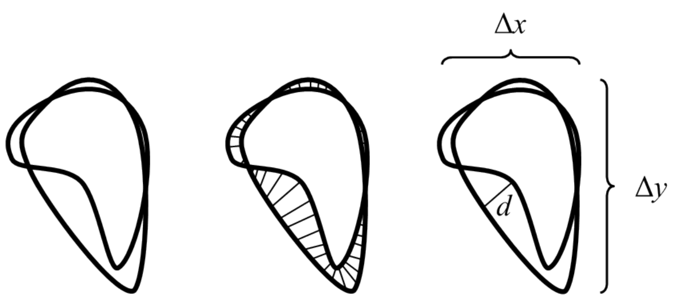


Figure 3.3: The Hausdorff distance is obtained by finding, for all points on one curve, the closest point on the other, and keeping the distance of the furthest pair. X- and y-dimensions are also shown.

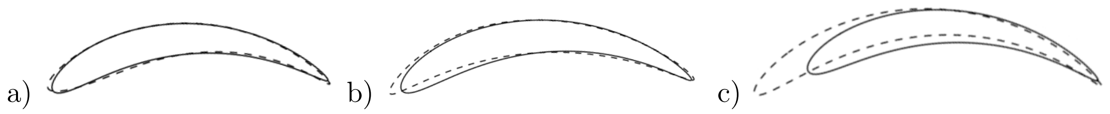


Figure 3.4: a) $Q = 1.7\%$; b) $Q = 4.4\%$; c) $Q = 20.8\%$

Discrete metric

The variable e is defined to represent how far the linkage gets to any precision point. This corresponds to the Hausdorff distance evaluated on a discrete and relatively small set of points on the curve. We therefore choose the objective function of minimizing e in the model. This can be seen as minimizing an approximate Hausdorff distance.

3.3 Constraint on Area

So far, the information contained about the curve is limited to the precision points. There is a chance that the solution found may go through the precision points, yet not produce the desired output (see Figure 3.5). If we add more precision points for a tighter fit, the model grows proportionately in size, with added variables and non-convex constraints. In general, it is desirable that the search space be as small as possible while sacrificing little precision.

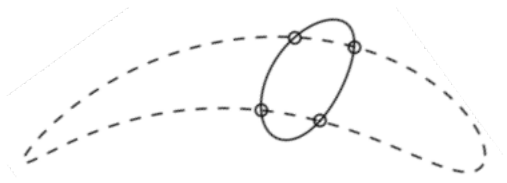
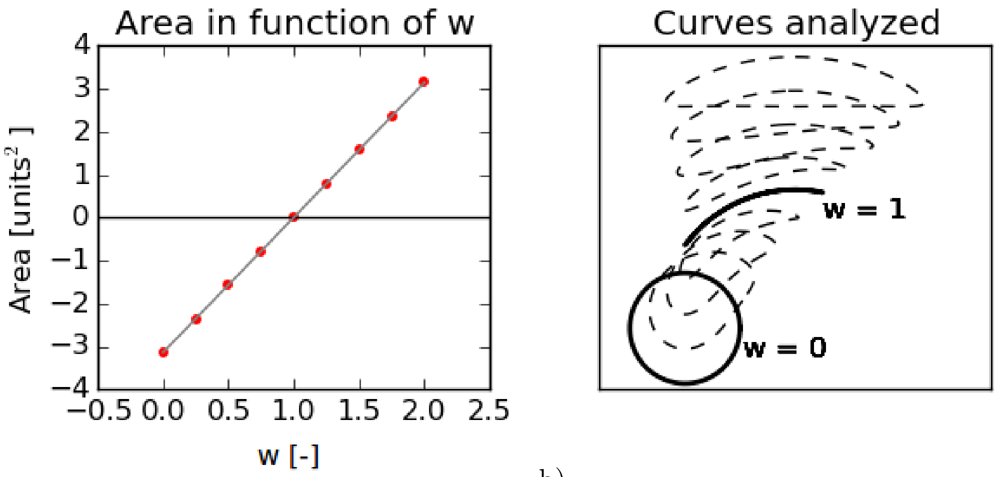


Figure 3.5: A possible solution to a curve sampled with too few precision points

A simple expression yielding the area of the coupler curve was found empirically. This section documents the empirical process that led to the finding of the expression. Section 3.4 provides a rigorous analytical demonstration.

Figure 3.6a shows that the area of the coupler curve varies linearly with the ratio w of CE over CD. Thus, an expression of the following form can be induced:

$$Area(w) = a \cdot w + b$$



a) b)

Figure 3.6: Variation of area with respect to ratio w

To determine a and b, two points are needed. First, when w is 0, the end effector **E** coincides with **C** and the coupler curve is a circle with radius **AC**. Second, when w is 1, the **E** coincides with point **D** and moves on an arc of null area (Figure 3.6 b).

$$Area(0) = -\pi \cdot AC^2 \qquad\qquad Area(1) = 0 \tag{3.12}$$

Note that the area of the circle is negative to denote that the curve is being traveled counter-clockwise. By substitution, we obtain the following expression:

$$Area = \pi \cdot AC^2 \left(1 - \frac{CE}{CD}\right) \tag{3.13}$$

A rigorous proof of this formula was thereafter written and is provided in section 3.4. The sign of the area tells us if the end effector is traveling clockwise or counterclockwise. Since this information is not known beforehand, we modify the constraint as such:

$$Area = \pi \cdot AC^2 \left|1 - \frac{CE}{CD}\right| \tag{3.14}$$

The area is a constant computed from the input curve. The curve is represented by a collection of points. The area is computed by summing all the areas under the segments joining adjacent pairs of points. This is equivalent to integration. Since we can constrain the area of the coupler curve, even smaller sample numbers yield precise solutions. Cases such as seen in Figure 3.5 are no longer possible. This allows keeping the model small.

3.4 Proof of Coupler Curve Area Formula

Let us consider the following four-bar linkage shown at Figure 3.7.

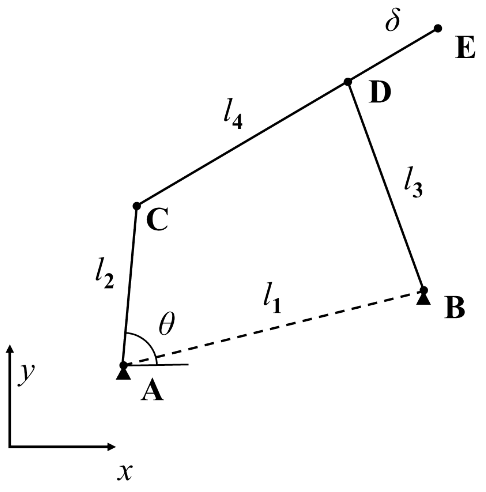


Figure 3.7: Modified notation for proof of area.

The defining parameters of the linkage are the coordinates of points **A** and **B**, and the lengths of the bars, and the parameter δ. The parameter δ is distinguished from the lengths because it can change signs. The value of δ is the position of **E** on an axis aligned with points **C** and **D** originating at point **D**. Therefore, we have the set of independent parameters T:

$$T = \{\mathbf{A}_x, \mathbf{A}_y, \mathbf{B}_x, \mathbf{B}_y, l_2, l_3, l_4, \delta\} \tag{3.15}$$

The parameter θ determines the position of the linkage. The position of points **C**, **D**, and **E** is dependent on θ. The coupler curve is defined as the locus of point **E** for $\theta \in [0, 2\pi)$.

We define a new notation herein because compactness is critical to the legibility of the proof. Table 3.3 shows the equivalence between the notation used in this chapter and the notation used throughout the rest of the document:

AC	AB	BD	CD	CE
l_1	l_1	l_3	l_4	$l_4 + \delta$

Table 3.3: Notation equivalence for lengths

Furthermore, we note the surface of the curve traveled by a point **P** by S_P. The area of the coupler curve is therefore noted as S_E.

The following three assumptions are made on the linkage:

Assumption 3.4.1. *The linkage is a class I crank-rocker.*

Assumption 3.4.2. *Link* **AC** *is a crank.*

Assumption 3.4.3. *The effector* **E** *is collinear with points* **C** *and* **D**.

The following relationships arise from assumptions 3.4.1 and 3.4.2:

$$l_2 < l_1 \qquad l_2 < l_3 \qquad l_2 < l_4 \tag{3.16}$$

Under these assumptions, as shown at equation (3.13), the signed area of the coupler curve has been found empirically to match the following expression:

$$S_E = \pi \cdot AC^2 \left(1 - \frac{CE}{CD}\right) = \pi \cdot l_2 \left(1 - \frac{l_4 + \delta}{l_4}\right) \tag{3.17}$$

$$= -\frac{{l_2}^2 \delta}{l_4} \pi \tag{3.18}$$

The sign gives information on the direction that the curve is traveled by point **E**. The area is positive if **E** travels in the same direction as **C**, and negative otherwise.

Here we prove it is in fact the exact expression. To do so, we first make simplifications. Second, we parametrize the trajectory of point **E** in function of the angle θ and T. Third, we integrate the parametric equations to obtain the expression for the area.

By definition of the integral, the area under a curve $y(x)$ over the interval $[a, b]$ is given by:

$$Area = \int_a^b y(x)\mathrm{d}x \tag{3.19}$$

For parametric equations of a closed curve, we make the following change of variables:

$$x = f(t) \qquad y = g(t) \qquad \mathrm{d}x = f'(t)\mathrm{d}t \tag{3.20}$$

$$Area = \oint g(t) \frac{\mathrm{d}f'(t)}{\mathrm{d}t}\,\mathrm{d}t \tag{3.21}$$

where the integration symbol designates that this is a *closed-line integral*, meaning that we integrate over a whole closed curve. In our case, the parametrization is $\mathbf{E}_x(\theta)$ and $\mathbf{E}_y(\theta)$. From equation (3.21), the expression to solve is:

$$S_E = \int_0^{2\pi} \mathbf{E}_y(\theta) \frac{\mathrm{d}\mathbf{E}_x(\theta)}{\mathrm{d}\theta}\, \mathrm{d}\theta \tag{3.22}$$

3.4.1 Simplifications and Definitions

The area of a coupler curve is independent on the position and the rotation of the linkage. Therefore, without loss of generality, we can place point **A** on the origin and **B** on the x-axis as such:

$$\mathbf{A}_x = \mathbf{A}_y = \mathbf{B}_y = 0 \tag{3.23}$$

By doing so, we also get:

$$\mathbf{B}_x = l_1 \tag{3.24}$$

Furthermore, we add the following definitions for convenience:

$$v = \frac{\delta}{l_4} \qquad\qquad w = \frac{l_4 + \delta}{l_4} = v + 1 \tag{3.25}$$

3.4.2 Parametrization

First, let us derive expressions for the coordinates $\mathbf{E}_x(\theta)$ and $\mathbf{E}_y(\theta)$. They define the coupler curve as θ cycles from 0 to 2π. The notation (θ) is dropped from this point on for improved legibility. Using vector geometry, we have that:

$$\overrightarrow{\mathbf{E}} = \overrightarrow{\mathbf{C}} + \overrightarrow{\mathbf{CE}} \tag{3.26}$$

The vectors $\overrightarrow{\mathbf{CE}}$ and $\overrightarrow{\mathbf{CD}}$ are related as follows:

$$\frac{\overrightarrow{\mathbf{CE}}}{(l_4 + \delta)} = \frac{\overrightarrow{\mathbf{CD}}}{l_4} \tag{3.27}$$

$$\overrightarrow{\mathbf{CE}} = w\overrightarrow{\mathbf{CD}} \tag{3.28}$$

Therefore, equation (3.26) can be expressed as:

$$\overrightarrow{\mathbf{E}} = \overrightarrow{\mathbf{C}} + w\overrightarrow{\mathbf{CD}} \tag{3.29}$$

In terms of the coordinates, this yields:

$$\mathbf{E}_x = \mathbf{C}_x + w[\mathbf{D}_x - \mathbf{C}_x] \tag{3.30}$$

$$\mathbf{E}_y = \mathbf{C}_y + w[\mathbf{D}_y - \mathbf{C}_y] \tag{3.31}$$

By applying equation (3.25), these simplify to:

$$\mathbf{E}_x = w\mathbf{D}_x - v\mathbf{C}_x \tag{3.32}$$

$$\mathbf{E}_y = w\mathbf{D}_y - v\mathbf{C}_y \tag{3.33}$$

The coordinates of point **C** are expressed as follows:

$$\mathbf{C}_x = \mathbf{A}_x + l_2 \cos\theta = l_2 \cos\theta \tag{3.34}$$

$$\mathbf{C}_y = \mathbf{A}_y + l_2 \sin\theta = l_2 \sin\theta \tag{3.35}$$

The behavior of point **D** is more complex. The distances from **D** to **C** and from **D** to **B** are respectively l_4 and l_3. There are two possible solutions for the location of **D**, which correspond to two different ways to assemble the links. The curves traveled by **E** in either case are symmetric and have the same area. Therefore, we can choose either solution without loss of generality. Figure 3.8 shows the geometrical construction used to derive the parametrization for point **D**.

The construction yields:

$$\mathbf{D}_x = \mathbf{B}_x - x_1 + x_2 \tag{3.36}$$

$$\mathbf{D}_y = \mathbf{B}_y + y_1 + y_2 \tag{3.37}$$

Using simple proportions, we find:

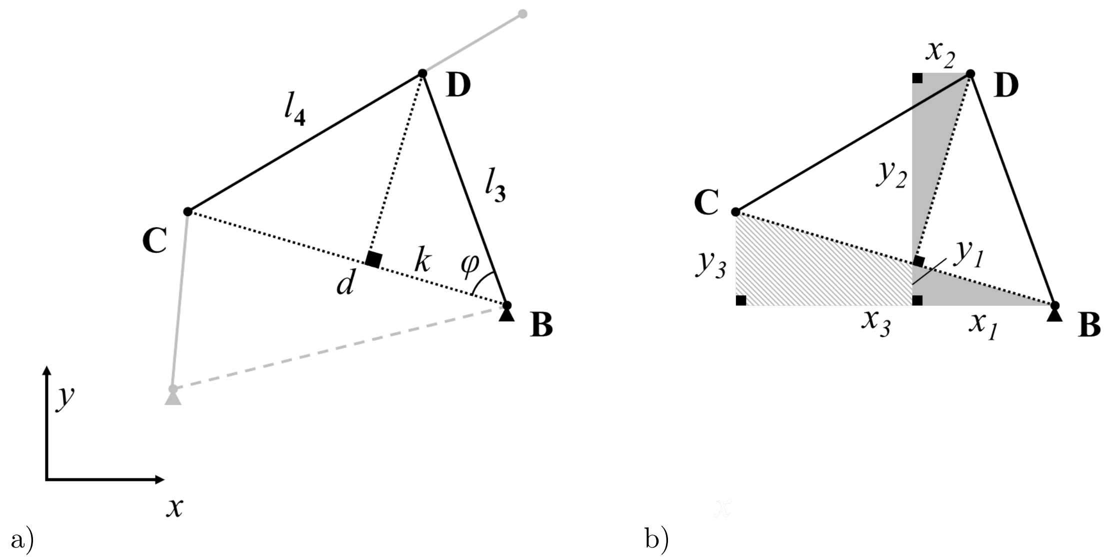


Figure 3.8: a) Distances d and k; b) Three similar right triangles aligned with the x and y axes.

$$x_1 = \frac{x_3 \cdot k}{d} = \frac{(\mathbf{B}_x - \mathbf{C}_x) \cdot k}{d} \tag{3.38}$$

$$y_1 = \frac{y_3 \cdot k}{d} = \frac{(\mathbf{C}_y - \mathbf{B}_y) \cdot k}{d} \tag{3.39}$$

$$x_2 = \frac{y_3 \cdot \sqrt{{l_3}^2 - k^2}}{d} = \frac{(\mathbf{C}_y - \mathbf{B}_y) \cdot \sqrt{{l_3}^2 - k^2}}{d} \tag{3.40}$$

$$y_2 = \frac{x_3 \cdot \sqrt{{l_3}^2 - k^2}}{d} = \frac{(\mathbf{B}_x - \mathbf{C}_x) \cdot \sqrt{{l_3}^2 - k^2}}{d} \tag{3.41}$$

The distance d is given by:

$$d = \sqrt{(\mathbf{C}_x - \mathbf{B}_x)^2 + (\mathbf{C}_y - \mathbf{B}_y)^2} \tag{3.42}$$

$$= \sqrt{(l_2 \cos\theta - l_1)^2 + (l_2 \sin\theta)^2} \tag{3.43}$$

$$= \sqrt{{l_2}^2 - 2 l_1 l_2 \cos\theta + {l_1}^2} \tag{3.44}$$

We now express k. First, we notice that:

$$\cos\phi = \frac{k}{l_3} \tag{3.45}$$

Then, using the law of cosines, we find:

$${l_4}^2 = d^2 + {l_3}^2 - 2 \cdot d \cdot l_3 \cdot \cos\phi \tag{3.46}$$

By substituting expression (3.45) into equation (3.46) and isolating k, we obtain:

$$k = \frac{d^2 + {l_3}^2 - {l_4}^2}{2 \cdot d} \tag{3.47}$$

Substituting expression (3.38) to (3.41) into equations (3.36) and (3.37), we obtain:

$$\mathbf{D}_x = \mathbf{B}_x + \frac{(\mathbf{C}_x - \mathbf{B}_x) \cdot k}{d} + \frac{(\mathbf{C}_y - \mathbf{B}_y)}{d}\sqrt{{l_3}^2 - k^2} \tag{3.48}$$

$$\mathbf{D}_y = \mathbf{B}_y + \frac{(\mathbf{C}_y - \mathbf{B}_y) \cdot k}{d} - \frac{(\mathbf{C}_x - \mathbf{B}_x)}{d}\sqrt{{l_3}^2 - k^2} \tag{3.49}$$

By applying the simplifications from section 3.4.1 and expanding k, we obtain the following expanded expressions:

$$\mathbf{D}_x = \frac{l_1}{2} + \frac{\mathbf{C}_x}{2} + \frac{\mathbf{C}_x({l_3}^2 - {l_4}^2)}{2d^2} - \frac{l_1({l_3}^2 - {l_4}^2)}{2d^2} + \frac{\mathbf{C}_y}{d}\sqrt{{l_3}^2 - k^2} \tag{3.50}$$

$$\mathbf{D}_y = \frac{\mathbf{C}_y}{2} + \frac{\mathbf{C}_y \cdot ({l_3}^2 - {l_4}^2)}{2d^2} - \frac{\mathbf{C}_x}{d}\sqrt{{l_3}^2 - k^2} + \frac{l_1}{d}\sqrt{{l_3}^2 - k^2} \tag{3.51}$$

3.4.3 Resolution

We want to solve the integral of equation (3.22), in which we substitute the parametrization from (3.32) and (3.33).

$$S_E = \int_0^{2\pi} \mathbf{E}_y \frac{\mathrm{d}\mathbf{E}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.52}$$

$$= \int_0^{2\pi} [w\mathbf{D}_y - v\mathbf{C}_y] \cdot \left[w\frac{\mathrm{d}\mathbf{D}_x}{\mathrm{d}\theta} - v\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\right]\mathrm{d}\theta \tag{3.53}$$

$$= \int_0^{2\pi} \left[w^2\mathbf{D}_y\frac{\mathrm{d}\mathbf{D}_x}{\mathrm{d}\theta} - vw\mathbf{D}_y\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta} + v^2\mathbf{C}_y\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta} - vw\mathbf{C}_y\frac{\mathrm{d}\mathbf{D}_x}{\mathrm{d}\theta}\right]\mathrm{d}\theta \tag{3.54}$$

$$= w^2\int_0^{2\pi} \mathbf{D}_y\frac{\mathrm{d}\mathbf{D}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta + v^2\int_0^{2\pi} \mathbf{C}_y\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta - vw\int_0^{2\pi} \mathbf{D}_y\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta - vw\int_0^{2\pi} \mathbf{C}_y\frac{\mathrm{d}\mathbf{D}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.55}$$

We notice that the integrals in the first two terms have the same form as equation 3.22 and correspond respectively to S_D and S_C. Hence their value is equal to the area of the curve traveled by these points. When θ increases, point $\mathbf{C}$ travels counterclockwise on a circle of radius $l_2 = 1$. As for point $\mathbf{D}$, it travels back and forth on an arc of a circle. Therefore we have:

$$S_C = -{l_2}^2\pi \tag{3.56}$$

$$S_D = 0 \tag{3.57}$$

We substitute these back into (3.55):

$$S_E = w^2 S_D + v^2 S_C - vw\int_0^{2\pi} \mathbf{D}_y \frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta - vw\int_0^{2\pi} \mathbf{C}_y \frac{\mathrm{d}\mathbf{D}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.58}$$

$$= -v^2{l_2}^2\pi - vw\left[\int_0^{2\pi} \mathbf{D}_y \frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta + \int_0^{2\pi} \mathbf{C}_y \frac{\mathrm{d}\mathbf{D}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta\right] \tag{3.59}$$

The term $\mathbf{D}_y$ is more complex to differentiate than $\mathbf{C}_y$. The following lemma states how the parametric functions can be swapped in the expression (see appendix A for proof):

$$\int_a^b f(x)\frac{\mathrm{d}g(x)}{\mathrm{d}x}dx = -\int_a^b g(x)\frac{\mathrm{d}f(x)}{\mathrm{d}x}dx \tag{3.60}$$

Using lemma (3.60), we make the following change to expression (3.59):

$$S_E = -v^2{l_2}^2\pi + vw\left[-\int_0^{2\pi} \mathbf{D}_y \frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta + \int_0^{2\pi} \mathbf{D}_x \frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta\right] \tag{3.61}$$

Let us define I_1 and I_2:

$$I_1 = \int_0^{2\pi} \mathbf{D}_y \frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.62}$$

$$I_2 = \int_0^{2\pi} \mathbf{D}_x \frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.63}$$

First, we simplify I_1. Substituting expression (3.51) into I_1 yields:

$$I_1 = \int_0^{2\pi} \left[\frac{\mathbf{C}_y}{2} + \frac{\mathbf{C}_y \cdot ({l_3}^2 - {l_4}^2)}{2d^2} - \frac{\mathbf{C}_x}{d}\sqrt{{l_3}^2 - k^2} + \frac{l_1}{d}\sqrt{{l_3}^2 - k^2}\right] \frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.64}$$

The first term is solved as follows:

$$\int_0^{2\pi} \frac{\mathbf{C}_y}{2} \frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta = \frac{1}{2}\int_0^{2\pi} \mathbf{C}_y \frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta = \frac{S_C}{2} = -\frac{{l_1}^2\pi}{2} \tag{3.65}$$

$$\tag{3.66}$$

The second term is not simplified here. The third term is solved as follows:

$$\int_0^{2\pi} \frac{\mathbf{C}_x}{d}\sqrt{{l_3}^2 - k^2}\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta = \int_0^{2\pi} \left(\frac{l_1\cos\theta}{d(\cos\theta)}\sqrt{{l_3}^2 - k(\cos\theta)^2}\right) l_2 \sin\theta\,\mathrm{d}\theta \tag{3.67}$$

where $k(\cos\theta)$ and $d(\cos\theta)$ indicate that k and d are functions of $\cos\theta$. We use the following change of variables:

$$u = \cos\theta \qquad \mathrm{d}u = -\sin\theta\mathrm{d}\theta \tag{3.68}$$

$$\int_0^{2\pi} \left(\frac{l_1\cos\theta}{d(\cos\theta)}\sqrt{{l_3}^2 - k(\cos\theta)^2}\right) l_2 \sin\theta\,\mathrm{d}\theta = -{l_1}^2\int_1^1 \frac{u}{d(u)}\sqrt{{l_3}^2 - k(u)^2}\mathrm{d}u \tag{3.69}$$

Because the bounds are equal, the integral is 0 if the integrand is continuously defined. The two following conditions need to be verified for continuous:

$$d \neq 0 \tag{3.70}$$

$${l_3}^2 - k^2 \geq 0 \tag{3.71}$$

Geometrically, d is defined as the distance between points **B** and **D**. From (3.16), we have $l_2 < l_1$. The first condition is therefore true since points **C** and **D** can never be the same distance from point **A**. Therefore, the d between points **C** and **D** is always greater than 0.

By definition, l_3 is always a positive quantity. From equation (3.45), we have:

$$k = \cos\phi \cdot l_3 \tag{3.72}$$

The domain of the cosine is $[-1, 1]$, so we know that $l_3 \geq |k|$. Both sides of this inequality are positive, so we have:

$$l_3{}^2 \geq |k|^2 \tag{3.73}$$
$$l_3{}^2 - |k|^2 \geq 0 \tag{3.74}$$
$$l_3{}^2 - k^2 \geq 0 \tag{3.75}$$

The two conditions are met. Therefore the integrand is continuous. We can safely conclude that the third term of I_1 reduces to 0.

We apply the same change of variables to the fourth term of I_1:

$$\int_0^{2\pi} \frac{l_1}{d}\sqrt{l_3{}^2 - k^2}\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta = -l_1 l_2 \int_0^{2\pi} \left(\frac{1}{d(\cos\theta)}\sqrt{l_3{}^2 - k(\cos\theta)^2}\right)\sin\theta\,\mathrm{d}\theta \tag{3.76}$$
$$= -l_1 l_2 \int_1^1 \left(\frac{1}{d(u)}\sqrt{l_3{}^2 - k(u)^2}\right)\mathrm{d}u \tag{3.77}$$
$$= 0 \tag{3.78}$$

Equation (3.64) reduces to:

$$I_1 = -\frac{l_2{}^2\pi}{2} + \int_0^{2\pi} \left[\frac{\mathbf{C}_y \cdot (l_3{}^2 - l_4{}^2)}{2d^2}\right]\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.79}$$

Next, we simplify I_2:

$$I_2 = \int_0^{2\pi} \left[\frac{l_1}{2} + \frac{\mathbf{C}_x}{2} + \frac{\mathbf{C}_x(l_3{}^2 - l_4{}^2)}{2d^2} - \frac{l_1(l_3{}^2 - l_4{}^2)}{2d^2} + \frac{\mathbf{C}_y}{d}\sqrt{l_3{}^2 - k^2}\right]\cdot\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.80}$$

The first terms yields:

$$\int_0^{2\pi} \frac{l_1}{2}\cdot\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta = \frac{l_1 l_2}{2}\int_0^{2\pi}\cos\theta\,\mathrm{d}\theta = 0 \tag{3.81}$$

The second term yields:

$$\int_0^{2\pi} \frac{\mathbf{C}_x}{2}\cdot\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta = \frac{1}{2}\int_0^{2\pi}\mathbf{C}_x\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta = -\frac{S_C}{2} = \frac{l_2{}^2\pi}{2} \tag{3.82}$$

Using the change of variable (3.68) on the fourth term yields:

$$\int_0^{2\pi} \frac{\mathbf{C}_y\sqrt{{l_3}^2-k^2}}{d}\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta = \int_0^{2\pi} \frac{l_2\sin\theta\cdot\sqrt{{l_3}^2-k(\cos\theta)^2}}{d(\cos\theta)} l_2\cos\theta\,\mathrm{d}\theta \tag{3.83}$$

$$= {l_2}^2\int_1^1 \frac{\sqrt{{l_3}^2-k(u)^2}}{d(u)} u\mathrm{d}u \tag{3.84}$$

$$= 0 \tag{3.85}$$

Hence, equation (3.80) becomes:

$$I_2 = \frac{{l_2}^2\pi}{2} + \int_0^{2\pi}\left[\frac{\mathbf{C}_x({l_3}^2-{l_4}^2)}{2d^2} - \frac{l_1({l_3}^2-{l_4}^2)}{2d^2}\right]\cdot\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.86}$$

$$= \frac{{l_2}^2\pi}{2} + \frac{{l_3}^2-{l_4}^2}{2}\int_0^{2\pi}\left[\frac{\mathbf{C}_x}{d^2}-\frac{l_1}{d^2}\right]\cdot\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta \tag{3.87}$$

Bringing equations (3.79) and (3.87) back into equation (3.61) yields:

$$S_E = -{l_2}^2v^2\pi + vw\left[{l_2}^2\pi - \frac{{l_3}^2-{l_4}^2}{2}\int_0^{2\pi}\left[\frac{\mathbf{C}_x}{d^2}-\frac{l_1}{d^2}\right]\frac{\mathrm{d}\mathbf{C}_y}{\mathrm{d}\theta}\,\mathrm{d}\theta + \int_0^{2\pi}\frac{\mathbf{C}_y({l_3}^2-{l_4}^2)}{2d^2}\frac{\mathrm{d}\mathbf{C}_x}{\mathrm{d}\theta}\,\mathrm{d}\theta\right] \tag{3.88}$$

$$= -{l_2}^2v^2\pi + {l_2}^2vw\left[\pi - \frac{{l_3}^2-{l_4}^2}{2}\int_0^{2\pi}\left[\frac{\cos^2\theta}{d^2} - \frac{\frac{l_1}{l_2}}{d^2}\cos\theta + \frac{\sin^2\theta}{d^2}\right]\mathrm{d}\theta\right] \tag{3.89}$$

$$= -{l_2}^2v^2\pi + {l_2}^2vw\left[\pi + \frac{{l_3}^2-{l_4}^2}{2}\int_0^{2\pi}\frac{\cos^2\theta - \frac{l_1}{l_2}\cos\theta + \sin^2\theta}{d^2}\,\mathrm{d}\theta\right] \tag{3.90}$$

$$= -{l_2}^2v^2\pi + {l_2}^2vw\left[\pi + \frac{{l_3}^2-{l_4}^2}{2}\int_0^{2\pi}\frac{1-\frac{l_1}{l_2}\cos\theta}{d^2}\,\mathrm{d}\theta\right] \tag{3.91}$$

Let us define the I_3 as equal to the integral inside equation (3.91). Substituting equation (3.44) in the place of d, we have:

$$I_3 = \int_0^{2\pi}\frac{1-\frac{l_1}{l_2}\cos\theta}{{l_2}^2+{l_1}^2-2l_1l_2\cos\theta}\,\mathrm{d}\theta \tag{3.92}$$

For clarity, we define $a = \frac{l_1}{l_2}$. Hence, I_3 becomes:

$$I_3 = \frac{1}{{l_2}^2}\int_0^{2\pi}\frac{1-a\cos\theta}{1+a^2-2a\cos\theta}\,\mathrm{d}\theta \tag{3.93}$$

The solution and steps to solve this integral were obtained using Wolfram-Alpha [56]. First, we use the following change of variables:

$$u = \tan\frac{\theta}{2} \qquad \mathrm{d}\theta = \frac{2\mathrm{d}u}{u^2+1} \qquad \cos\theta = \frac{1-u^2}{u^2+1} \tag{3.94}$$

Expression (3.93) becomes:

$$I_3 = \frac{1}{{l_2}^2}\int_{-\infty}^{\infty} \frac{1-a\left(\frac{1-u^2}{u^2+1}\right)}{a^2-2a\left(\frac{1-u^2}{u^2+1}\right)+1}\cdot\frac{2}{u^2+1}\mathrm{d}u \tag{3.95}$$

By rearranging and using partial fractions, we obtain:

$$I_3 = \frac{1}{{l_2}^2}\left[\int_{-\infty}^{\infty}\frac{1}{u^2+1}\mathrm{d}u + (1-a^2)\int_{-\infty}^{\infty}\frac{1}{(a+1)^2u^2+(a-1)^2}\mathrm{d}u\right] \tag{3.96}$$

We factor out the $(a-1)^2$ term form the denominator of the second term to obtain a more convenient form:

$$I_3 = \frac{1}{{l_2}^2}\left[\int_{-\infty}^{\infty}\frac{1}{u^2+1}\mathrm{d}u + \frac{1+a}{1-a}\int_{-\infty}^{\infty}\frac{1}{\left(\frac{a+1}{a-1}\right)^2u^2+1}\mathrm{d}u\right] \tag{3.97}$$

We make a change of variable in the second integrand:

$$s = \frac{a+1}{a-1}u \qquad \mathrm{d}u = \frac{a-1}{a+1}\mathrm{d}s \tag{3.98}$$

$$\begin{aligned} I_3 &= \frac{1}{{l_2}^2}\left[\int_{-\infty}^{\infty}\frac{1}{u^2+1}\mathrm{d}u + \frac{1+a}{1-a}\int_{-\infty}^{\infty}\frac{1}{s^2+1}\cdot\frac{a-1}{a+1}\mathrm{d}s\right] && (3.99)\\ &= \frac{1}{{l_2}^2}\left[\int_{-\infty}^{\infty}\frac{1}{u^2+1}\mathrm{d}u - \int_{-\infty}^{\infty}\frac{1}{s^2+1}\mathrm{d}s\right] && (3.100)\\ &= 0 && (3.101)\end{aligned}$$

Because the two integrands have the same form and the same bounds, they cancel out. We can now input this result in equation (3.91):

$$S_E = -{l_2}^2 v^2 \pi - {l_2}^2 vw \left[-\pi + \frac{{l_3}^2 - {l_4}^2}{2} \cdot 0 \right] \tag{3.102}$$
$$= {l_2}^2 [-v^2 \pi + vw\pi] \tag{3.103}$$
$$= {l_2}^2 [-v^2 \pi + v(1+v)\pi] \tag{3.104}$$
$$= {l_2}^2 [-v^2 \pi + v\pi + v^2\pi] \tag{3.105}$$
$$= -{l_2}^2 v\pi \tag{3.106}$$
$$= \frac{{l_2}^2 \delta}{l_4} \pi \quad \square \tag{3.107}$$

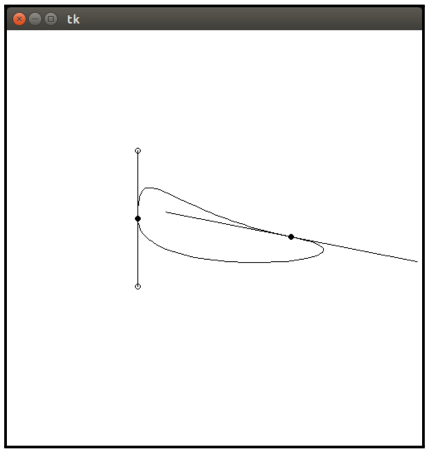
It is interesting to notice that the final expression for the area of the coupler curve is independent on the position of pivot **B** and lengths l_1 and l_3.

3.5 Linkendo: A Simple Design Software

A software application implementing the solving process was developed in Python. It is called Linkendo (Linkage Efficient Non-convex Deterministic Optimizer). It allows the user to draw a curve and returns a four-bar linkage that approximates it. The user draws a curve by positioning control points on a minimal graphic interface as shown in Figure 3.9a. The curve is then analyzed. A few samplings are done with different sample numbers. The number of samplings as well as the number of points for each sampling can be set by the user. For each sampling, two models are constructed: one with constraint (3.5) and the second with constraint (3.6). A portfolio approach, or simultaneous execution of several solver instances, is used to solve all models in parallel. When a solution is returned, its distance to the input curve is evaluated with Q as defined in section 3.2.3. If Q is below the user-defined threshold, all processes stop and the best solution is returned and displayed to the user, as shown at Figure 3.9b.

3.6 Conclusion

A quadratic model of the four-bar linkage was presented. The model introduces a new mathematical expression relating the parameters of the linkage to the area of the coupler curve. A rigorous proof of this expression was provided. The software Linkendo, implementing the exposed method, was also presented. The next section provides results on the performance of the implementation.

a) 

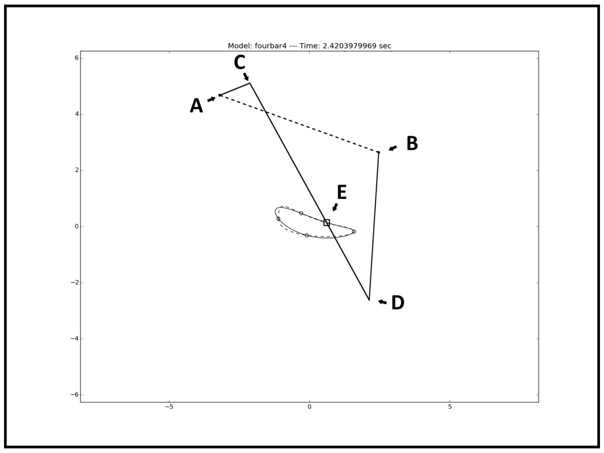
b) 

Figure 3.9: Design software screenshots; a) The user draws a curve; b) A linkage closely replicating the curve is displayed.

Chapter 4

Experimentation

The non-convex optimization presented at the previous section was tested against a state of the art genetic algorithm. Results comparing speed and precision are presented. Then, we characterize the performance of our approach. Last, we demonstrate the flexibility of the model by using it to design a robotic gripper.

4.1 Methodology

We use the software described in section 3.5 throughout the experimentation. We generated a benchmark of 100 curves. For each instance, N different samplings are made. For each sampling, we launch the two possible linkage configurations (constraints (3.5) or (3.6)). A total of $2N$ models are solved in parallel. If a solution with Q (see section 3.2.3 for the definition of the curve similarity metric Q) lower than a user-defined threshold T is found, the execution is stopped and the solution is returned. Tests conducted with the experimental timeout of 900 seconds demonstrated that 84.5 % of solutions were returned before 60 seconds, and 99.6 % were returned before 400 seconds. Thus, the timeout was set at 400 seconds. The solving flow is shown on Figure 4.1.

4.2 Benchmark

The benchmark consists of 100 coupler curves of randomly generated linkages. This is to ensure that the most possibilities are covered, rather than limiting the benchmark to the most common cases in the industry. The linkages were generated within the search space of the model. At least one valid solution is therefore guaranteed to exist. The curves are resized to fit inside a 4 by 4 units square centered at the origin. All curves measure at least 1 unit at their widest. This benchmark spans a wide range of shapes in the search space of our model, which all possess at least one solution. Some curves are presented at Figure 4.2.

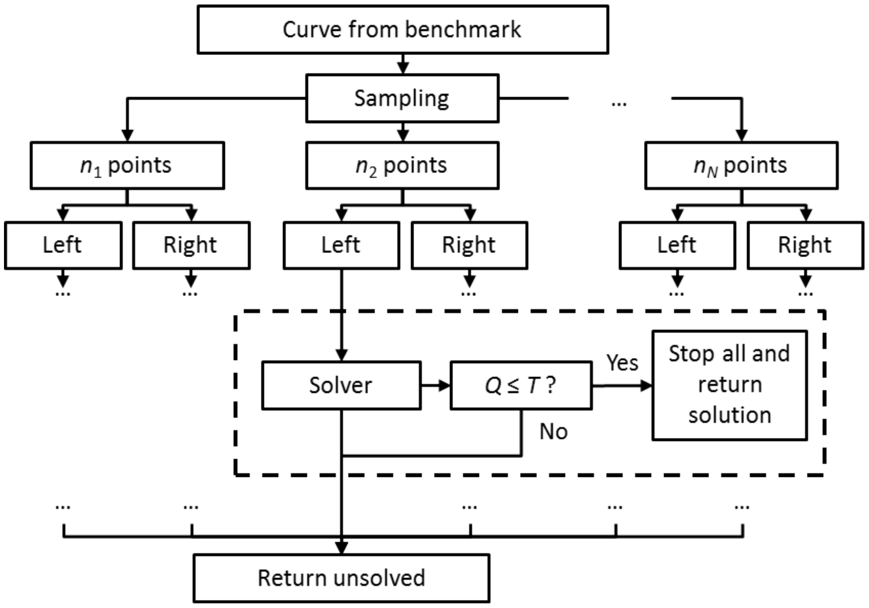


Figure 4.1: *Linkendo* software parallel solving flow

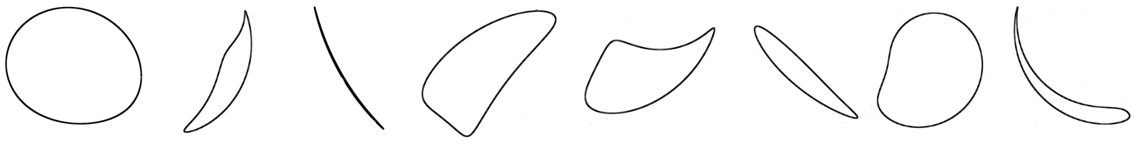


Figure 4.2: Example curves from the benchmark. A wide variety of shapes is represented. The curves have been resized to compare shape rather than dimension.

4.3 Results

4.3.1 Comparison with Genetic Algorithm

To present the performance of our non-convex optimization approach, we compare it to results obtained with the genetic algorithm proposed by Cabrera et al. [8], thereafter referred to as the GA. We implemented the GA based on Cabrera et al.'s paper.

For the comparison, we replace the *solver* block from Figure 4.1 either with the non-convex solver *Couenne* or the GA. The rest of the solving flow remains unchanged. Three samplings are done ($N = 3$) with $n_1 = 6$, $n_2 = 7$ and $n_3 = 8$. The threshold T is set at 5 %, so when a solution with lower Q value is returned, the execution stops. We set $s = 0.1$, and $e_{\mathrm{u}} = 0.01$, which was found to yield the best performance through iterative testing. Figure 4.3 shows the distribution of the solutions with respect to Q at timeout. The metric quantifies how well the input and output curves match in a continuous way.

We see that the majority of the curves were solved by Couenne with Q lower than 5 %. In

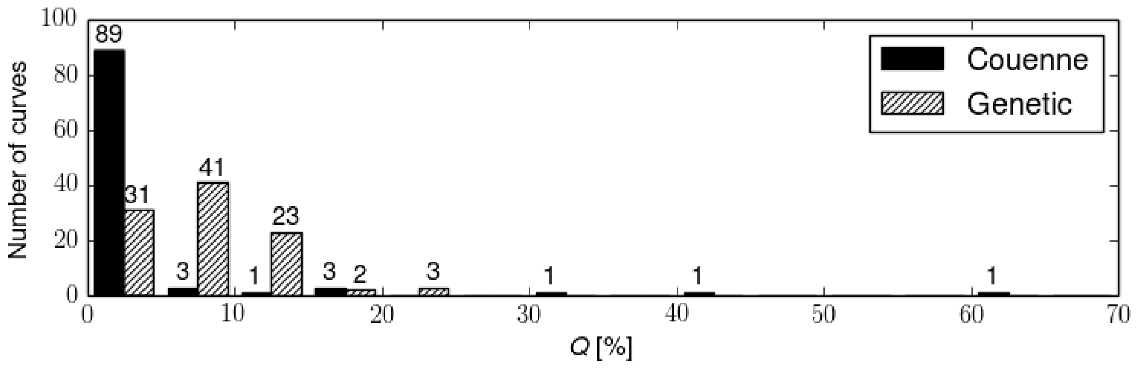


Figure 4.3: Distribution of Q values for non-convex optimization and evolutionary approaches

contrast, all curves solved by the genetic algorithm returned a Q value below 20 %, but less precise on average. Table 4.1 emphasizes that the median Q returned by Couenne and the median absolute deviation are lower than those of the genetic algorithm.

Approach	$\bar{Q}$	$\sigma^2(Q)$	$\tilde{Q}$	MAD (Q)
Couenne	3.22	71.59	1.00	1.48
Genetic	8.02	16.52	7.25	10.75

Table 4.1: Average, variance, median and median absolute deviation of Q.

For the non-convex optimization, Q is computed after Couenne has returned an optimal solution. Therefore, any solution returned by Couenne before timeout is optimal with respect to the discrete metric of the model. As for the GA, Q is computed once every few hundred generations. This constitutes an advantage for the GA because sub-optimal solutions found by Couenne must time out before evaluation. Even so, as shown in Figure 4.4, the non-convex optimization approach is faster and times out less often.

Bounds tightening allows propagation of the restricted domain of variable e. This considerably reduces the search space from the beginning. As for the GA, the final solution depends a lot on the initial random population. Though it consistently finds a reasonable approximation of the curve, it usually stalls in local minima.

4.3.2 Characterization

We show the critical impact of the area constraint and how the feature identification sampling improves the model compared to a uniform sampling.

To evaluate the impact of the area constraint, the benchmark was solved twice over three sample number sets; once with the area constraint and once without. Table 4.2 shows the

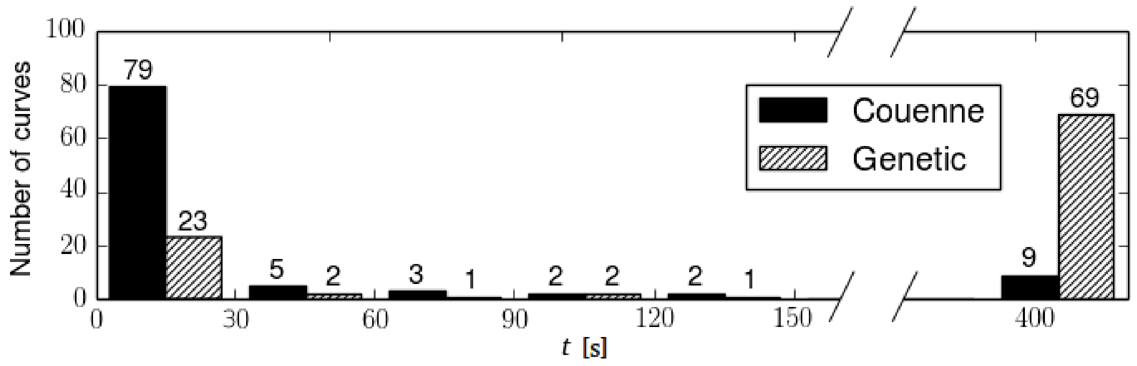


Figure 4.4: Distribution of solving times for both approaches. The curves at 400 timed out.

number of curves in the benchmark solved with Q lower than 5 % in less than τ seconds, for three values of τ. The number of curves with no solution returned is given.

Sampling	Area	$Q < 5$ %			No
		5 s	60 s	400 s	solution
{4, 5, 6}	**Yes**	**59**	**83**	**92**	**0**
$\{4,5,6\}$	No	37	58	63	0
$\{6,7,8\}$	Yes	51	81	89	1
$\{6,7,8\}$	No	50	68	78	1
$\{10,12,16\}$	Yes	30	59	69	11
$\{10,12,16\}$	No	33	57	66	14

Table 4.2: Number of curves solved under 5, 60 or 400 seconds with different samplings

Higher sample numbers yield longer times of computation without significantly improving the accuracy. In general, the area constraint improved the number of curves solved. Also, when the fewer sampling points are used, the area constraint is most efficient. Without the area constraint, the software performs best with sample numbers $\{6, 7, 8\}$. With the area constraint, lower sample numbers yield a better performance.

The feature identification sampling is compared to a uniform sampling with no analysis of the curve. The experiment was conducted with sets of sample numbers $\{4, 5, 6\}$ and $\{6, 7, 8\}$. Figure 4.5 shows how the sampling affects the distribution of Q.

For both sets of sample numbers, the feature identification brought the Q distribution closer to 0 %. This shows that without increasing the complexity of the model, choosing points strategically can help achieve greater precision.

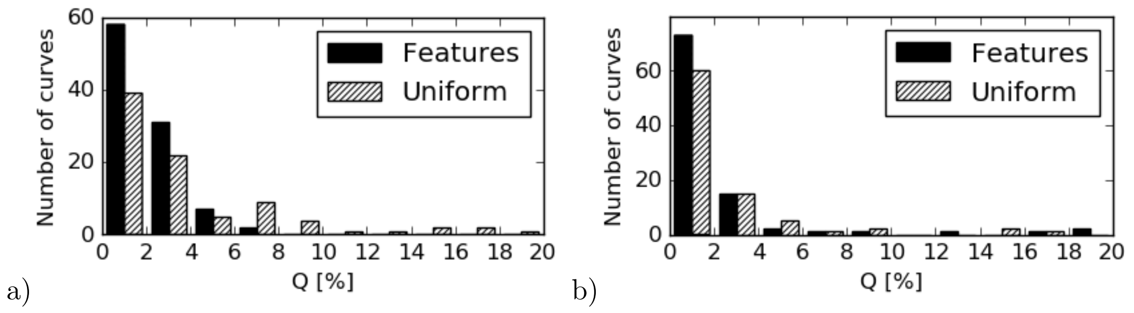


Figure 4.5: Distribution of Qs over benchmark with both sampling techniques; a) sample numbers $\{4, 5, 6\}$; b) sample numbers $\{6, 7, 8\}$

4.3.3 Design of a Gripper

A benefit of using mathematical optimization is that the model is easily customizable for specific applications. Say we wish to design a gripping mechanism made of symmetric four-bar linkages such that the tip goes through four points, with low precision p_{low} for the first three points and high precision p_{high} on the last. Furthermore, the location of the anchors is restricted. The problem is shown on Figure 4.6 (a).

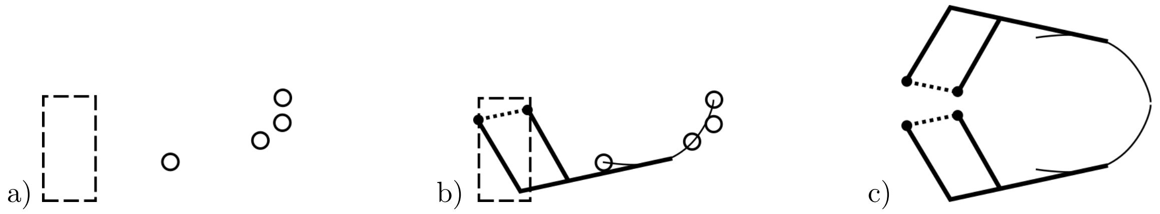


Figure 4.6: a) Precision points and anchor bounding box; b) Synthesized four-bar linkage; c) Gripper

To adapt the model, only the following modifications need to be done. We replace the following domains:

$$A_x, A_y, B_x, B_y \in [-10, 10) \quad \Rightarrow \quad \begin{cases} A_x, B_x \in [x_{\min}, x_{\max}) \\ A_y, B_y \in [y_{\min}, y_{\max}) \end{cases} \tag{4.1}$$

We set $e_{\text{u}} = p_{\text{low}}$. We add constraint $(T_{x_3} - E_{x_3})^2 + (T_{y_3} - E_{y_3})^2 \leq p_{\text{high}}$ and disable the area constraint. The resulting gripping mechanism shown at Figure 4.6b is obtained in 0.20 second, with the modified model.

4.4 Discussion

The speeds reached are suitable for interactive applications. Because we use a non-convex optimization solver, our approach is flexible and can be readily adjusted to meet specific needs or different goals. Unlike analytical approaches [46, 54], we are not limited by the number of precision points to reach. Moreover, the area constraint allows the solver to extrapolate between the precision points.

Our method aims at matching a continuous curve rather than a discrete set of precision points. However, many related works [8, 6] focus on matching precision points. Our results show capabilities in both goals. Indeed, the distance to the precision point is bounded by the error, which cannot be higher than e_{u} or to a maximum of 1 % of the size of the curve. Any solution discussed matched its precision points at least to this precision.

Our approach also presents benefits compared to machine-learning approaches. A database cannot guarantee coverage of the whole search space. With our approach, the search space is fully explorable and only limited by user-defined restrictions.

Though we focused on minimizing the error, this can be easily changed by replacing the objective function. One could minimize the sum of dimensions, the area of the coupler curve, or the difference of area between the input curve and the output curve.

4.5 Future Work

The software usability could be improved by providing tools to edit constraints. It should be noted that many aspects of the method can be parallelized. The software already uses parallel processing to solve several instances at once. Arbitrarily many instances can be solved in parallel by expanding the set of sample points. Also, the open-source solver Couenne, which creates a tree of sub-problems, could also be parallelized.

It could be possible to take advantage of the filtering capabilities of Couenne to propose hybrid approaches. For instance, by setting e_{u} to more permissive values, initial guesses for linkages could be computed and then passed on GA algorithms for optimization. This is interesting given that the outcome of genetic algorithms is highly dependent on the quality of the initial population.

Our software could extend to four-bar linkages where points $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{E}$ are not collinear. Difficulties include more symmetric configurations and the generalization of the area constraint. Joints such as sliders and complex mechanisms such as geared five and six-bar mechanisms could be modeled. 3D mechanisms could also be tackled. Our software could be combined with other design analyses such as stress analysis. Multiple linkages could be linked to a gearing software for timing control. Finally, the model could be generated as the user defines his own mechanisms by adding bars and joints.

4.6 Conclusion

Our software can quickly and accurately synthesize collinear four-bar linkages for given coupler curves. The results demonstrated the implementation performs competitively. The work presented has a variety of interesting avenues for future work, including improvement of the implementation, and also widening the scope of solvable problems.

Conclusion

This thesis provides an overview of the state of the art in both non-convex optimization and mechanical linkage synthesis. It also presents the outcome of a project conducted as a collaboration between Université Laval and Autodesk Research. In the scope of this project, a method to synthesize four-bar linkages using non-convex optimization was developed and implemented in a Python software. The project has already been published as a paper at the CP 2016 conference.

State of the art linkage design methods show limited performance or lack either optimality or generality. The proposed approach is an improvement both in generality and speed for the solving of mechanical linkages. The experiments conducted show that coupler curves can be solved accurately and fast in most cases for collinear four-bar linkages using the global non-convex solver *Couenne*. In particular, for a closed curve, a new mathematical expression using the area of the curve provides significantly better filtering capabilities. This expression, absent in the current literature, was mathematically proven. For our best model, our results show that 90 % of the curves are solved under 400 seconds, 59 % of which below 5 seconds.

The work herein discussed shows that non-convex optimization is a promising avenue for automated mechanical synthesis. The model and software implementation produced provide flexible tools usable for a variety of real-world problems. These tools can also be leveraged as a basis to extend the approach, either to different types of synthesis problems or to different types of mechanisms.

Appendix A

Proof of Lemma on the Area of Reciprocal Parametric Curves

Lemma A.0.1. *The area of a closed continuous curve parametrized as* $x_1 = f(t)$ *and* $y_1 = g(t)$ *is equal to the opposite of the area of the reciprocal curve* $x_2 = g(t)$ *and* $y_2 = f(t)$*:*

$$\oint f(t)\frac{\mathrm{d}g(t)}{\mathrm{d}t}\mathrm{d}t = -\oint g(t)\frac{\mathrm{d}f(t)}{\mathrm{d}t}\mathrm{d}t$$

Proof. Let us evaluate the sum of the areas of the reciprocal curves:

$$\oint f(t)\frac{\mathrm{d}g(t)}{\mathrm{d}t}\mathrm{d}t + \oint g(t)\frac{\mathrm{d}f(t)}{\mathrm{d}t}\mathrm{d}t = \oint \left[f(t)\frac{\mathrm{d}g(t)}{\mathrm{d}t} + g(t)\frac{\mathrm{d}f(t)}{\mathrm{d}t}\right]\mathrm{d}t$$

Using the definition for the differentiation of a product we find:

$$\begin{aligned} &= \oint \frac{\mathrm{d}\left[f(t)g(t)\right]}{\mathrm{d}t}\mathrm{d}t \\ &= f(t)g(t)\Big|_{t_{\min}}^{t_{\max}} \end{aligned}$$

By definition of the closed-line integral, the integration starts and end at the same point. Therefore we have:

$$\begin{aligned} f(t_{\min}) &= f(t_{\max}) \\ g(t_{\min}) &= g(t_{\max}) \end{aligned}$$

Under the assumption that the curve is continuous, we have:

$$\begin{aligned} f(t)g(t)\Big|_{t_{\min}}^{t_{\max}} &= f(t_{\max})g(t_{\max}) - f(t_{\min})g(t_{\min}) \\ &= f(t_{\max})g(t_{\max}) - f(t_{\max})g(t_{\max}) \qquad \text{(A.1)} \\ &= 0 \qquad \text{(A.2)} \end{aligned}$$

We rearrange the terms to complete the proof:

$$\begin{aligned} \oint f(t)\frac{\mathrm{d}g(t)}{\mathrm{d}t}\mathrm{d}t + \oint g(t)\frac{\mathrm{d}f(t)}{\mathrm{d}t}\mathrm{d}t &= 0 \\ \oint f(t)\frac{\mathrm{d}g(t)}{\mathrm{d}t}\mathrm{d}t &= -\oint g(t)\frac{\mathrm{d}f(t)}{\mathrm{d}t}\mathrm{d}t \end{aligned} \tag{A.3}$$

□

Bibliography

[1] S.K. Acharyya and M. Mandal. Performance of EAs for four-bar linkage synthesis. *Mechanism and Machine Theory*, 44(9):1784–1794, 2009.

[2] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[3] S.S. Balli and S. Chand. Defects in link mechanisms and solution rectification. *Mechanism and Machine Theory*, 37(9):851–876, 2002.

[4] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.

[5] J.L. Blechschmidt and J.J. Uicker. Linkage synthesis using algebraic curves. *Mechanisms, Transmissions, and Automation in Design*, 108(4):543–548, 1986.

[6] R.R. Bulatovic and S.R. Djordjevic. Optimal synthesis of a four-bar linkage by method of controlled deviation. *Theoretical and applied mechanics*, 31(3-4):265–280, 2004.

[7] M.R. Bussieck and S. Vigerske. MINLP solver software. *Wiley encyclopedia of operations research and management science*, 2010.

[8] J.A. Cabrera, A. Simon, and M. Prado. Optimal synthesis of mechanisms with genetic algorithms. *Mechanism and Machine theory*, 37(10):1165–1177, 2002.

[9] G. Chabert. Ibex documentation. `http://www.ibex-lib.org/doc/`. Accessed on 2016-11-14.

[10] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 3rd edition, 2009.

[11] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R.W. Sumner, W. Matusik, and B. Bickel. Computational design of mechanical characters. *Transactions on Graphics*, 32(4):83:1–83:12, 2013.

[12] E.A. Dijksman. *Motion geometry of mechanisms*. Cambridge University Press, 1976.

[13] A.G. Erdman. Three and four precision point kinematic synthesis of planar linkages. *Mechanism and Machine Theory*, 16(3):227–245, 1981.

[14] GNU lesser general public license. https://www.gnu.org/copyleft/lgpl.html. Accessed on 2016-11-27.

[15] F. Freudenstein and G.N. Sandor. Synthesis of path generating mechanisms by means of a programmed digital computer. *Engineering for Industry*, 81(1):159–168, 1959.

[16] C.G. Gibson. *Elementary geometry of algebraic curves: an undergraduate introduction.* Cambridge University Press, 1998.

[17] C. Gosselin and J. Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE transactions on robotics and automation*, 6(3):281–290, 1990.

[18] C.M. Gosselin and J-F Hamel. The agile eye: a high-performance three-degree-of-freedom camera-orienting device. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 781–786. IEEE, 1994.

[19] V. Goulet, W. Li, H. Cheong, F. Iorio, and C.-G. Quimper. Four-bar linkage synthesis using non-convex optimization. In *Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming (CP 2016)*, pages 618–635. Springer, 2016.

[20] L. Granvilliers and F. Benhamou. Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques. *Transactions on Mathematical Software*, 32(1):138–156, 2006.

[21] W. Groß. Grundzüge der mengenlehre. *Monatshefte für Mathematik*, 26(1):A34–A35, 1915.

[22] R. Hartenberg and J. Danavit. *Kinematic Synthesis of Linkages.* McGraw-Hill, 1964.

[23] J.A. Hrones and G.L. Nelson. *Analysis of the Four-bar Linkage.* Technology Press of the Massachusetts Institute of Technology and Wiley, 1951.

[24] T. Jansen, Weschler L., and L. Herzog. *Strandbeest: The Dream Machines.* Taschen, 2015.

[25] E.C. Kinzel, J.P. Schmiedeler, and G.R. Pennock. Function generation with finitely separated precision points using geometric constraint programming. *Mechanical Design*, 129(11):1185–1190, 2007.

[26] L.J. Latecki and R. Lakamper. Shape similarity measure based on correspondence of visual parts. *Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1185–1190, 2000.

[27] W.-Y. Lin. A GA–DE hybrid evolutionary algorithm for path synthesis of four-bar linkage. *Mechanism and Machine Theory*, 45(8):1096–1107, 2010.

[28] Y. Lin and L. Schrage. The global solver in the LINDO API. *Optimization Methods and Software*, 24(4-5):657–668, 2009.

[29] A.K. Mallik, A. Ghosh, and G. Dittrich. *Kinematic analysis and synthesis of mechanisms*. CRC Press, 1994.

[30] J.P. Marques-Silva and K.A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *Transactions on Computers*, 48(5):506–521, 1999.

[31] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

[32] R. Misener and C.A. Floudas. ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *Global Optimization*, 59(2-3):503–526, 2014.

[33] A.K. Natesan. Kinematic analysis and synthesis of four-bar mechanisms for straight line coupler curves. Master's thesis, Rochester Institute of Technology, 1994.

[34] R.L. Norton. Linkages. `http://www.designofmachinery.com/Linkage/`. Accessed on 2016-12-08.

[35] R.L. Norton. *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines*. WCB McGraw-Hill, 3rd edition, 1999.

[36] The Berkeley software distribution 2-clause license. `https://opensource.org/licenses/BSD-2-Clause`. Accessed on 2016-11-27.

[37] Common public license. `https://opensource.org/licenses/cpl1.0.php`. Accessed on 2016-11-15.

[38] "Pasimi". Four-bar function generator of the function log(u) for $1 < u < 10$. `https://commons.wikimedia.org/wiki/File:Func_Geen_Log(u).gif`, 2015.

[39] P. Radhakrishnan. *Automated design of planar mechanisms*. PhD thesis, University of Texas at Austin, 2014.

[40] P. Radhakrishnan and M.I. Campbell. A graph grammar based scheme for generating and evaluating planar mechanisms. In *Proceedings of Design Computing and Cognition '10*, pages 663–679. Springer, 2011.

[41] D.M. Rector. Linkages, mechanism designer and simulator. `http://blog.rectorsquid.com/linkage-mechanism-designer-and-simulator/`. Accessed on 2016-12-08.

[42] F. Reuleaux and E.S. Ferguson. *Kinematics of machinery: outlines of a theory of machines*. Courier Corporation, 2012.

[43] G. Rossum. Python reference manual. Technical report, PythonLabs, Amsterdam, The Netherlands, The Netherlands, 1995.

[44] N.V. Sahinidis. Baron: A general purpose global optimization software package. *Global Optimization*, 8(2):201–205, 1996.

[45] G.N. Sandor. *A general complex-number method for plane kinematic synthesis with applications*. Columbia University, 1959.

[46] G.N. Sandor and A.G. Erdman. *Advanced mechanism design: analysis and synthesis*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

[47] L.C. Schmidt, H. Shetty, and S.C. Chase. A graph grammar approach for structure synthesis of mechanisms. *Mechanical Design*, 122(4):371–376, 2000.

[48] Saltire Software. Atlas of linkages. `http://www.saltire.com/LinkageAtlas/`. Accessed on 2016-12-06.

[49] F.R. Stöckli and K. Shea. A simulation-driven graph grammar method for the automated synthesis of passive dynamic brachiating robots. In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2015.

[50] M. Tawarmalani and N.V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[51] Theo Jansens Strandbeest-mechanismus. `http://www.tm-aktuell.de/TM5/Viergelenkketten/Strandbeest.html`. Accessed on 2016-10-24.

[52] K.-L. Ting and Y.-W. Liu. Rotatability laws for N-bar kinematic chains and their proof. *Mechanical Design*, 113(1):32–39, 1991.

[53] P. Todd, D. Mueller, and E. Fichter. *Atlas of the Four-bar Linkage*. Saltire Software, Tigard, Oregon, 1992.

[54] J.J. Uicker, G.R. Pennock, and J.E. Shigley. *Theory of machines and mechanisms*. Oxford University Press Oxford, 4th edition, 2011.

[55] C.W. Wampler, A.P. Morgan, and A.J. Sommese. Complete solution of the nine-point path synthesis problem for four-bar linkages. *Mechanical Design*, 114(1):153–159, 1992.

[56] Wolfram|Alpha. `http://www.wolframalpha.com/input/?i=int((1-B*cos(x))%2F(1%2BB%5E2-2*B*cos(x)),dx)`. Accessed on 2016-11-30.

[57] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2003.

[58] N. Yu, A.G. Erdman, and B.P. Byers. Lincages 2000: Latest developments and case study. In *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 1421–1425. American Society of Mechanical Engineers, 2002.