Four-Bar Linkage Synthesis Using Non-Convex Optimization

Vincent Goulet[1], Wei Li[2], Hyunmin Cheong[2], Francesco Iorio[2], and Claude-Guy Quimper[1]

[1] Université Laval
[2] Autodesk Research

Abstract. We show how four-bar linkages can be designed using non-convex optimization techniques. Our generative design software takes as input a curve that needs to be reproduced by a four-bar linkage and outputs the best assembly that approximates this curve. We model the problem using quadratic constraints and show how redundant constraints help to solve the problem. We also provide an algorithm that samples the curve based on its characteristics. Experiments show that our software is faster and more precise than existing systems. The current work is part of a larger generative design initiative at Autodesk Research.

1 Introduction

A mechanism is an arrangement of machine parts that generates a specified motion. The synthesis of a mechanism is the process of determining the position, the orientation, and other parametric properties of parts according to constraints governing their alignment or motion. The design of mechanisms has greatly benefited from the advent of computer techniques. Computer-aided design and engineering software (CAD/CAE) have been widely used in the documentation, analysis, and optimization of designs. Though, still nowadays, the existing technologies and tools lack a function for automated mechanism synthesis. Creating mechanisms that meet specified motion and geometric requirements demands highly trained expert designers. As the available computing power keeps growing, so does the interest in the development of generative design tools.

In this paper, we address the problem of generating a four-bar linkage that outputs a prescribed curve (Figure 1). Four-bar linkages are simple yet practically important mechanisms that can generate complex motion. Since the First Industrial Revolution, they have been widely applied in mechanical systems, including manufacturing, agriculture, robotics, and automotive industry [16]. However, it is a laborious process to manually design a four-bar linkage based on a target curve. The current state of the art of four-bar linkage design is a time consuming process, and the results often lack optimality or generality. Hence, this paper introduces an automated and efficient four-bar linkage synthesis approach, which is also an important milestone towards synthesizing more complex mechanisms.

We first establish the current state of the art regarding mechanical assembly and path synthesis. We then introduce the terminology and notation used for the four-bar linkage and the global non-convex optimization strategy. The contributions presented are

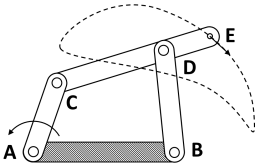


Fig. 1. A four-bar linkage and output *coupler curve*

the feature identification sampling technique, the four-bar linkage quadratic model, the special constraint on the area of the curve, and the design software developed. Results emphasizing the speed and quality of our method follow along with a discussion.

2 Related Work

2.1 Mechanical Assembly

A long standing challenge of mechanical design is the automation of design synthesis tasks [21]. Existing mechanism synthesis methods include systematic search [22, 17], machine-learning based approaches [8], stochastic search [26] and graph-based approach [18, 20]. However, to the best of our knowledge, existing generative design approaches still lack the generality or performance to be practical.

2.2 Path Synthesis of Four-Bar Linkage

Without a computer-aided approach, a human designer typically uses prior knowledge and/or atlases of coupler curves to identify a candidate linkage to produce the desired curve [16]. Then the chosen linkage is modified until it is satisfactory [25].

Analytical approaches formulate the four-bar linkage constraints, solve the problem and return exact solutions [19, 24]. They require a set of points or positions input by the user, which can be challenging to provide. In many cases, there is no mechanism that can produce exactly the desired path. In fact, although the mechanism found goes through the specified points, it may not go through the desired curve (see Fig. 8 for an example). Also, analytical approaches are limited to solving problems with five or less target points, (see [14] and the references therein).

Alternatively, numerical methods are used to synthesize approximate mechanisms with acceptable tolerance between the input path and the coupler curve. Genetic algorithms have been widely applied to the four-bar mechanism synthesis problem [7, 2]. Genetic algorithms and other stochastic search methods [6] have the same limitation – there is no assurance that they will find a global optimum. Also, because the objective function of four-bar mechanisms is highly constrained, the typical evolutionary algorithms have to choose a very large number of initial population so that a considerable amount of them can play in the next iteration. This technique unnecessarily increases CPU time and reserves a large amount of memory during the computing iterations. The lack of consistency also makes it challenging for performance evaluation.

Machine-learning approaches [8, 25] store a large number of coupler curves in a database. Automated procedures for fitting coupler curves are used to locate potential linkage solutions from the database. Neural network [12] and sequential quadratic programming [8] can be used to match coupler curves. However, such an approach requires building a large linkage database. Another limitation is that the quality of the generated motion directly depends on the mechanisms in the database and sampling techniques.

3 Preliminaries

3.1 Mechanical Linkage

A mechanical linkage is a set of rigid bodies, called *links*, connected by joints. Though many types of joints exist, we herein only consider the *revolute* joint or *pivot*, which allows for one degree of freedom rotation. This paper focuses on the two-dimensional four-bar linkage (Figure 1), made of four links in a closed loop. The joints **A**, **B**, **C** and **D** are pivots. The positions of **A** and **B** are fixed. The motion of point **E** is the output of the mechanism, therefore it is called the *end effector*. The link **AB**, called the *frame*, cannot move. The link **AC**, called the *crank*, drives the motion of the linkage. The link **BD** is driven back and forth about **B** and is called the *rocker*. The link **CDE**, called the *coupler*, couples the rocker to the crank. The path traced by **E** over a full rotation of the crank is called the *coupler curve*. A four-bar linkage is *collinear* if point **E** is aligned with **C** and **D**. A linkage is *Grashof* if one link is able to fully rotate. We assume this condition is met and the crank **AC** can fully rotate.

3.2 Non-Convex Global Optimization

Mathematical optimization aims at finding good solutions to a problem according to some user-defined criteria. *Global* optimization is a family of techniques that guarantee that the solutions returned are absolutely optimal. These techniques often consist of relaxing the problem to a form efficiently solvable to optimality. This relaxed solution provides a bound for *branch and bound search.*

To be computable, the *global optimization problem* is modelled mathematically. The model consists of a set of variables, a set of constraints that need to be satisfied, and an objective function. Each variable has a *domain*, a set of all values it can be assigned.

The *solver* takes as input the model and finds suitable values for all variables, such that constraints are satisfied and the objective is optimized. Solvers are available for a wide range of applications and can be categorized by the types of variables they can handle, whether Boolean, integer, or real. Solvers can also be categorized by the types of functions they can handle, whether logic, linear, convex, or non-convex.

Modelling a four-bar linkage requires real variables and non-convex constraints. The global optimization solver *Couenne* [5] is specialized in both regards, and is the solver used for all experimentation presented. It was chosen over related candidates of comparable performance such as Baron [23] and AlphaBB [4] because it is open source. The constraint solver IBEX [1] was also considered but did not show sufficient performance. Other considered continuous solvers include RealPaver [10], SCIP [3] and LindoAPI [15].

Non-convex problems are difficult to solve even for the best available software. Couenne combines many techniques from constraint programming and other optimization subfields. It uses constraint propagation and interval arithmetic to achieve bounds tightening on each variable [5], therefore reducing the search space. It relaxes the non-linear constraints into linear envelopes. It uses branch and bound to create more tightly bounded subproblems. By adding redundant constraints, this envelope is further tightened. Whether redundant constraints make the solving faster depends on their number and complexity. Testing is required for validation. Couenne feeds the linear problem to CPLEX [13] to compute the solution to the relaxation.

4 Contribution

We developed a strategy 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. The time frame of this project spanned six months. The first two months were used to survey the available technology. The last four months were used to develop the model and strategy. We modelled the mechanism using its geometric properties, keeping in mind the possible generalization to mechanisms of higher complexity, and a novel cut (or redundant constraint) was developed using the area of the curve. We also designed a novel point sampling technique. We implemented this strategy in a simple design software.

4.1 Fitness Metric

We aim at designing a four-bar linkage which replicates as tightly as possible a continuous curve. To make this problem tractable for the constraint solver, we strategically sample the curve using the technique describe in Sec. 4.2. The model described in Sec. 4.3 minimizes a single variable e which represents the maximum distance from the curve to a sample point. We sample the curve with as little points as possible to keep the search space small. Note that the solver could return a solution with zero error, meaning the solution curve reaches all sample points, and yet not match the input curve, as shown in Fig. 8. In general, it is necessary to evaluate how well the continuous input curve matches the output curve after it is returned, regardless of the objective value.

Several well-established curve matching metrics exist. The Hausdorff distance [11] d is the greatest distance from any point on the curves to the closest point on the other curve. To render 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. 2, the equation for Q is:

$$Q = \frac{d}{\max\left(\Delta x, \Delta y\right)}$$

Figure 3 shows matching curves for 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 match. It is worth noting that Q does not consider 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 Sec. 4.4 make these events unlikely.

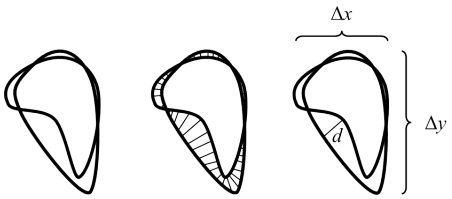


Fig. 2. The Hausdorff distance is obtained by finding, for all points on one curve, the closest point on the other, and keeping the distance of the furthest pair. X- and y-dimensions are also shown.

Fig. 3. Q = (a) 1.7 %; (b) 4.4 %; (c) 20.8 %

4.2 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. Here we propose a strategy to choose the n sample points $T_1, \ldots, T_n$ that best represent the input curve. We call the number n of sample points the *sample number*. A compromise needs to be made when choosing the sample number. Indeed, large sample numbers increase the execution time. However, they also improve the Q value as they better represent the continuous curve.

Some points are more important than others, like cusps or sharp turns. We call these points of interest *features*. Figure 4 shows a curve with features and one without. It is also important to have some sample points between the features to depict the general behaviour. The remaining sampling points are spread evenly between the features. It is possible for a curve to have no feature (e.g. an ellipse). In this case, the samples are distributed uniformly with one placed at the point of maximal curvature (Fig. 4).

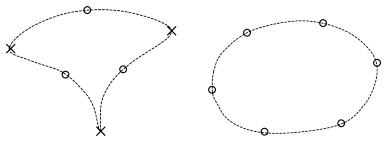


Fig. 4. Sampling technique: features are marked by $\times$ and remaining points by $\circ$.

To identify the features, we find all maxima of curvature. However, we do not compute the actual curvature, because it approaches infinity at cusps and reaches inconveniently high values at very sharp turns. Instead, as shown on Fig. 5, we compute the squared change in angle θ^2 between segments of the high resolution pre-sampled array. We square θ to amplify the variation.

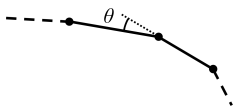


Fig. 5. The deviation θ between consecutive segments

We compute the median absolute deviation from all squared angles θ^2. Using the first derivative of θ^2 with respect to the distance travelled 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. Experience has shown that filtering out data within 10 times the median absolute deviation yields satisfactory results. Algorithm 1 presents the equivalent pseudocode.

Algorithm 1 Feature filtering($\boldsymbol{x}, \boldsymbol{y}$)

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

4.3 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 to each of the n sample points. In other words, the solver has to find a mechanism and compute n positions for this mechanism. Each position brings the effector close to its corresponding target point. This section describes the variables, constraints and objective function that compose the model.

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 nevertheless 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 target 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 sample point $\mathbf{T}_i$.

Without loss of generality, 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 4. It is helpful for the algorithm's filtering to bound the domain of e with the upper error bound e_{u}.

The information on the variables is gathered in Tab. 1.

Table 1. Variables for four-bar linkage model

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	$C_x, C_y, D_x, D_y, E_x, E_y$	$[-10, 10]$	$6n$
Error	e	$[0, e_{\mathrm{u}}]$	1

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 force the coordinates to be separated by distances corresponding to the lengths of the bars. For example, for the crank **AC** we have:

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

We use a similar constraint to define the error e as the upper bound of the squared distance from points $\mathbf{E}_i$ to points $\mathbf{T}_i$.

$$(T_{x_i} - E_{x_i})^2 + (T_{y_i} - E_{y_i})^2 \leq e \qquad \forall\, i \in [1, n]$$

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

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

The lengths of the bars are not sufficient to determine the configuration of the mechanism. As shown in Fig. 6, 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 target point $\mathbf{T}_i$, the coordinates for $\mathbf{E}_i$ have to be on the same side of the segment $\mathbf{BC}_i$. Since $\mathbf{T}_i$ and $\mathbf{E}_i$ must lie close to one another, constraining either one is equivalent. 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.

$$(T_{x_i} - C_{x_i})(B_y - C_{y_i}) \geq (T_{y_i} - C_{y_i})(B_x - C_{x_i}) \qquad \forall\, i \in [1, n] \qquad (1)$$
$$(T_{x_i} - C_{x_i})(B_y - C_{y_i}) \leq (T_{y_i} - C_{y_i})(B_x - C_{x_i}) \qquad \forall\, i \in [1, n] \qquad (2)$$

The two constraints are mutually exclusive, so a model may represent 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.

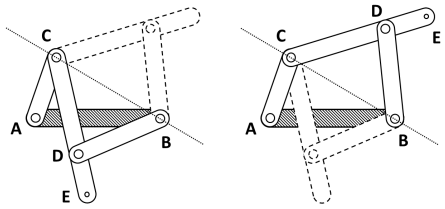


Fig. 6. The two possible configurations of the same links

The Grashof condition [9] 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. The following constraints enforce this:

$$AB \geq AC \qquad BD \geq AC \qquad CD \geq AC$$
$$CD + BD \geq AC + AB + s$$
$$AB + CD \geq AC + BD + s$$
$$AB + BD \geq AC + CD + s$$

A security constant s is added to the three last constraints to avoid equality. Otherwise, the mechanism could fold over itself completely. In this state, its behaviour is indeterminate, as it can unfold in two ways, as shown in Fig. 7. It is the singularity where the mechanism can switch between the two configurations of Fig. 6. Singularities are generally undesirable as they require additional control and involve high mechanical stress. The security constant s can be tuned to the desired tolerance. For all experiments herein s was set to 0.1, to minimally reduce the search space while preventing singularities.

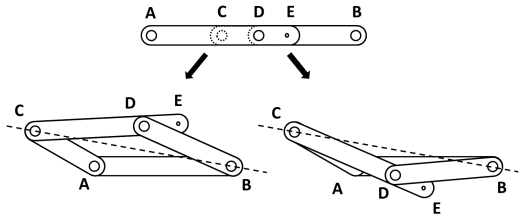


Fig. 7. Singular behaviour when CD + BD = AC + AB

It is worth noting that the model does not require the solution found to follow the sample points in any order. Therefore, in theory, the solver could return a mechanism which goes through the sample points in an undesired order. However, this is unlikely for two reasons. First, part of the sampling is done by filling gaps between the features. This creates a continuity between the points which the solutions tend to follow naturally. Second of all, violating the order of the points generally results in a significant change in the curve area, which is constrained as discussed in Sec. 4.4.

Table 2. List of constraints for four-bar linkage model

Constraint	Quantity
$(A_x - B_x)^2 + (A_y - B_y)^2 = AB^2$	1
$(A_x - C_{x_i})^2 + (A_y - C_{y_i})^2 = AC^2$	n
$(B_x - D_{x_i})^2 + (B_y - D_{y_i})^2 = BD^2$	n
$(C_{x_i} - E_{x_i})^2 + (C_{y_i} - E_{y_i})^2 = CE^2$	n
$w \cdot (D_{x_i} - C_{x_i}) = E_{x_i} - C_{x_i}$	n
$w \cdot (D_{y_i} - C_{y_i}) = E_{y_i} - C_{y_i}$	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
$(T_{x_i} - C_{x_i}) \cdot (B_y - C_{y_i}) \leqslant$ $(T_{y_i} - C_{y_i}) \cdot (B_x - C_{x_i})$	n
$(T_{x_i} - E_{x_i})^2 + (T_{y_i} - E_{y_i})^2 \leq e$	n

Objective Function The goal is to minimize the distance between the two continuous curves, using the continuous metric Q. Implementing Q in the model would require 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.

The variable e is defined in an analogous way to Q, but for a low resolution approximation of the curve. We therefore choose the objective function of minimizing e, which is the maximum distance between the output and input curves at the sample points. Therefore, even though ultimately we want to minimize the continuous metric Q, we had to define a discrete metric for the solver to use.

4.4 Constraint on Area

So far, the information contained about the curve is limited to the sample points. There is a chance that the solution found may go through the sample points, yet not produce the desired output (see Fig. 8). If we add more points for a tighter fit, the model will grow proportionately, 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.

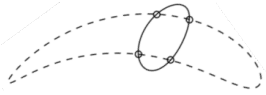
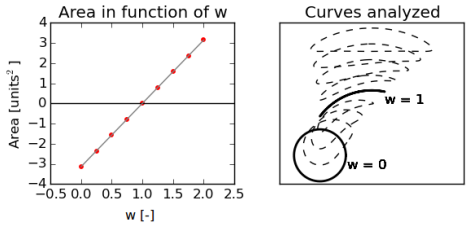


Fig. 8. Possible solution to a curve with few sample points

A relationship found empirically allowed including the area of the curve in the model. Figure 9 (a) 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)

Fig. 9. 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 (Fig. 9 (b)).

$$Area(0) = \pi \cdot AC^2 \qquad Area(1) = 0$$

By substitution, we obtain the following expression:

$$Area = \pi \cdot AC^2 \, (CE/CD - 1)$$

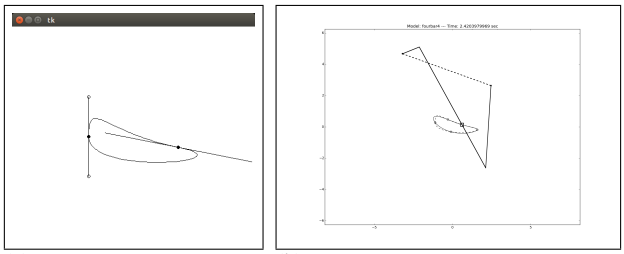
The sign of the area tells us if the end effector is travelling clockwise or counterclockwise. Since this information is not known beforehand, we modify the constraint as such:

$$Area = \pi \cdot AC^2 \, |CE/CD - 1| \tag{3}$$

The area is a constant computed from the input curve. Since we can constrain the area of the coupler curve, even smaller sample numbers yield precise solutions. Cases such as seen in Fig. 8 are no longer possible. This allows keeping the model small.

4.5 Simple Design Software

A software application implementing the solving process was developed in Python. 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 Fig. 10 (a). The curve is then analyzed. A few samplings are done with different sample numbers. For each sample number, two models are constructed: one with constraint (1) and the second with constraint (2). A portfolio approach is used and all models are launched in parallel. When a solution is returned, its distance to the input curve is evaluated with Q. 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 Fig. 10 (b).



(a) (b)

Fig. 10. Design software screenshots; (a) User draws a curve; (b) Matching linkage is displayed.

5 Experimentation

We first show a precision and speed comparison with a genetic algorithm. Then, we characterize the performance of our approach. Last, we demonstrate the flexibility of the model by using it to design a robotic gripper.

We use the software described in section 4.5 throughout the experimentation.We generated a benchmark of 100 random curves. For each instance, N different samplings are made. For each sampling, we launch the two possible linkage configurations (constraints (1) or (2)). A total of $2N$ models are solved in parallel. If a solution with Q lower than 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 Fig. 11.

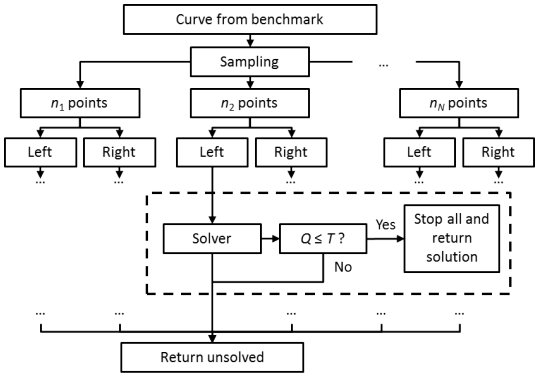


Fig. 11. How a curve is solved

5.1 Benchmark

The benchmark consists of 100 coupler curves of randomly generated linkages. The linkages were generated within the search space of the model. The curves are resized to fit inside a 4 by 4 units square centred 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 Fig. 12.

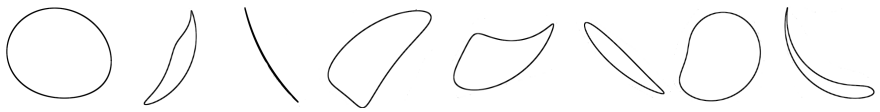


Fig. 12. Example curves from the benchmark

5.2 Results

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 [7], thereafter referred to as the GA.

For the comparison, we replace the *solver* block from Fig. 11 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 13 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.

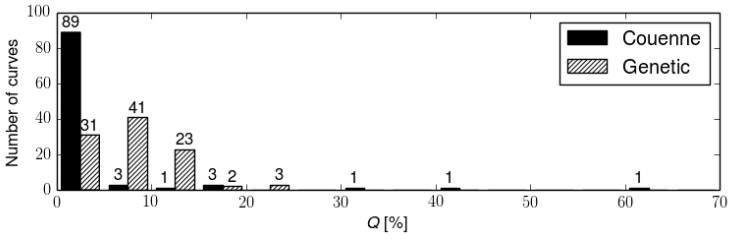


Fig. 13. Distribution of Q values for non-convex optimization and evolutionary approaches

We see that the majority of the curves were solved by Couenne with Q lower than 5 %. In contrast, all curves solved by the genetic algorithm returned a Q value below 20 %, but less precise on average. Table 3 emphasizes that the median Q returned by Couenne and the median absolute deviation are lower than those of the genetic algorithm.

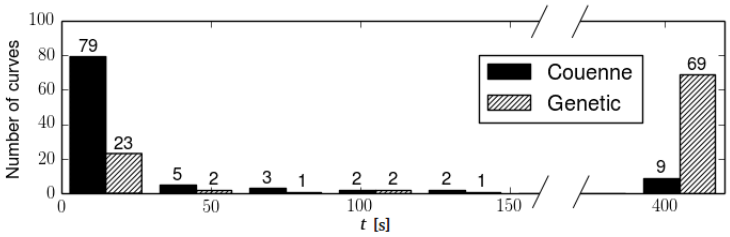


Fig. 14. Distribution of solving times for both approaches. The curves at 400 timed out.

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 Fig. 14, 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.

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 shows the 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.

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 ef-

Table 3. Average, variance, median and median absolute deviation of Q.

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. Number of curves solved under 5, 60 or 400 seconds with different samplings

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

ficient. 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 15 shows how the sampling affects the distribution of Q.

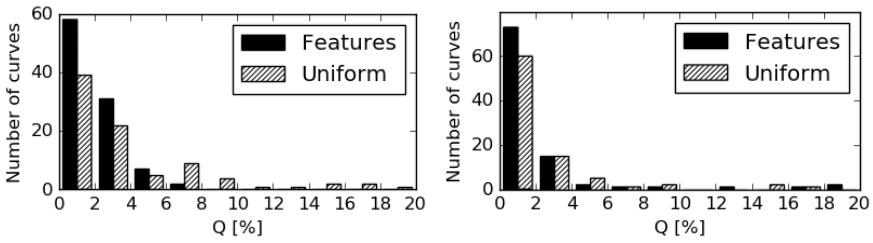


Fig. 15. Distribution of Qs over benchmark with both sampling techniques; (a) sample numbers $\{4, 5, 6\}$; (b) sample numbers $\{6, 7, 8\}$

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.

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 at Fig. 16 (a).

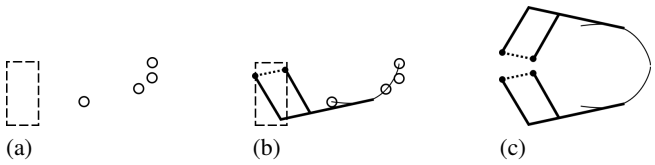


Fig. 16. (a) Target points and anchor bounding box; (b) Synthesized four-bar linkage; (c) Gripper

To adapt the model, only the following modifications need to be done. We replace $A_x, A_y, B_x, B_y \in \{-10, 10\}$ by $A_x, B_x \in \{x_{\min}, x_{\max}\}$; $A_y, B_y \in \{y_{\min}, y_{\max}\}$. 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 Fig. 16 (b) was obtained in 0.20 seconds, with the modified model.

5.3 Discussion

Our software can quickly and accurately synthesize collinear four-bar linkages for given coupler curves. Indeed, 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 [19, 24], we are not limited by the number of target points to reach. Moreover, the area constraint allows the solver to extrapolate between the target points.

Our method aims at matching a continuous curve rather than a discrete set of target points. However, many related works [7, 6] focus on matching target points. Our results show capabilities in both goals. Indeed, the distance to the target 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 target 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.

5.4 Future Work

The software usability could be improved by providing tools to edit constraints.

Our software could extend to four-bar linkages where points **C**, **D** and **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 modelled. 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.

6 Conclusion

The current state of the art of four-bar linkage synthesis is limited in speed, memory consumption, lack of optimality or lack of generality. Our paper contributes an improved method to the general and fast solving of mechanical linkages. We showed how to accurately solve complete coupler curves in a short time for collinear four-bar linkages using non-convex optimization. For closed curves, a novel constraint can leverage the area of the curve for increased accuracy and performance. For our best model, 90 % of the curves could be solved under 400 seconds, 59 % of which below 5 seconds.

Our approach was implemented in simple software where a user can enter a curve and visualize the solution found. This milestone paves the way for modelling mechanisms of increased complexity such as the general four-bar linkage or five-bar linkages. It also provides a very flexible basis for solving four-bar linkages with various constraints or different objectives.

References

1. Ibex library online documentation. http://www.ibex-lib.org/doc/. Accessed: 2016-06-28.
2. SK Acharyya and M Mandal. Performance of eas for four-bar linkage synthesis. *Mechanism and Machine Theory*, 44(9):1784–1794, 2009.
3. Tobias Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. http://mpc.zib.de/index.php/MPC/article/view/4.
4. I.P. Androulakis, C.D. Maranas, and C.A. FLOUDAS. alphabb: A global optimization method for general constrained nonconvex problems, 1995.
5. 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.
6. Radovan R. Bulatovic and Stevan 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. J. A. Cabrera, A. Simon, and M. Prado. Optimal synthesis of mechanisms with genetic algorithms. *Mechanism and Machine theory*, 37(10):1165–1177, 2002.
8. Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Trans. Graph.*, 32(4):83:1–83:12, July 2013.
9. Evert Albertus Dijksman. *Motion geometry of mechanisms*. CUP Archive, 1976.
10. Laurent Granvilliers and Frédéric Benhamou. Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software*, 32(1):138–156, 2006.
11. Wilh Groß. Grundzüge der mengenlehre. *Monatshefte für Mathematik*, 26(1):A34–A35, 1915.
12. D. A. Hoeltzel and Wei-Hua Chieng. Pattern matching synthesis as an automated approach to mechanism design. *Journal of Mechanical Design*, 112(2):190–199, 1990.
13. IBM. IBM ILOG CPLEX Optimization Studio: High-performance software for mathematical programming and optimization, 2016. See http://www.ilog.com/products/cplex/.
14. Edward C. Kinzel, James P. Schmiedeler, and Gordon R. Pennock. Function generation with finitely separated precision points using geometric constraint programming. *Journal of Mechanical Design*, 129(11):1185–1190, 2007.
15. Youdong Lin and Linus Schrage. The global solver in the lindo api. *Optimization Methods Software*, 24(4-5):657–668, August 2009.
16. Robert L. Norton. *Design of Machinery: An Introduction to the Synthesis and Analysi of Mechanisms and Machines*. WCB McGraw-Hill, 1999.
17. Barry O'sullivan and James Bowen. A constraint-based approach to supporting conceptual design. In *Artificial Intelligence in Design'98*, pages 291–308. Springer, 1998.
18. Pradeep Radhakrishnan and Matthew I. Campbell. A graph grammar based scheme for generating and evaluating planar mechanisms. In *Design Computing and Cognition'10*, pages 663–679. Springer, 2011.
19. George N. Sandor and Arthur G. Erdman. *Advanced mechanism design: analysis and synthesis*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.
20. Fritz R Stöckli and Kristina Shea. A simulation-driven graph grammar method for the automated synthesis of passive dynamic brachiating robots. In *ASME 2015 IDETC/CIE Conferences*. American Society of Mechanical Engineers, 2015.
21. Devika Subramanian. Conceptual design and artificial intelligence. In *Proceedings of IJCAI'93*, pages 800–809, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.
22. Devika Subramanian and Cheuk-San Wang. Kinematic synthesis with configuration spaces. *Research in Engineering Design*, 7(3):193–213, 1995.

23. M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.
24. John Joseph Uicker, Gordon R. Pennock, and Joseph Edward Shigley. *Theory of machines and mechanisms*. Oxford University Press Oxford, 2011.
25. V. Unruh and P. Krishnaswami. A computer-aided design technique for semi-automated infinite point coupler curve synthesis of four-bar linkages. *Journal of Mechanical Design*, 117(1):143–149, 1995.
26. Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. Motion-guided mechanical toy modeling. *ACM Trans. Graph.*, 31(6):127:1–127:10, November 2012.