2-Dimensional Outline Shape Representation for Generative Design with Evolutionary Algorithms

Paul Lapok, Alistair Lawson and Ben Paechter

Edinburgh Napier University, Edinburgh EH10 5DT, UK
{p.lapok,a.lawson,b.paechter}@napier.ac.uk

Abstract. In this paper, we investigate the ability of genetic representation methods to describe two-dimensional outline shapes, in order to use them in a generative design system. A specific area of mechanical design focuses on planar mechanisms. These are assembled of mechanical components, e.g. multiple levers, which transmit forces and torques over their contour. The shape of the contour influences the performance of the overall system. The genetic representations are based on floating-point chromosomes, where each value maps to a specific parameter of a resulting shape. In order to evaluate the performance of each representation method, a set of target shapes was defined. These consist of simple symmetric and asymmetric shapes with edges and curves, and also of more complex mechanical lever shapes, extracted from an automotive device. An evolutionary algorithm with crossover and mutation operators is used to search for the best approximation of these target shapes. The fitness function is based on two penalty values: first, calculated by comparing the area of a candidate solution with the area of a target shape; and second, based on the intersection area between a candidate solution and a target shape compared to the entire area of the target. Experiments were undertaken to investigate the capabilities of the representations in terms of search space coverage; compatibility with evolutionary operators; and the ability to produce shapes with mechanical characteristics. The results show the benefits and drawbacks of using each of selected methods of representation, and their suitability of reassembling different outline shapes.

Keywords: Evolutionary Representation, Shape Representation, Shape Optimization, Evolutionary Algorithm

1 Introduction

The performance of many engineering applications is highly dependent on functional shapes. A Generative Design System (GDS) can support the design process by providing potential solutions, and subsequently, saving designer's time. They are commonly used for aerodynamic shape optimization, e.g. to evolve airfoils [1–3], and also in topology optimization to improve the material usage of components with a focus on the inner structure. The shapes are represented in various ways, such as using constructive solid geometries [4], and voxel based representations [5]. Shape representations for evolving turbine blades were also investigated with a focus on material stress [6]. In mechanisms design [7], the shape and position of each component has a significant influence on the feasibility, function, and performance of the whole device. Some existing GDSs focus on the kinematics of massless mechanisms, such as on the problem of four-bar linkages [8]. However, these GDSs do not consider neither interactions between components' outlines, nor their relationships with the environment. Yet, in many applications, components require to be in contact with each in order to transfer forces and moments over their outline between an input and an output, e.g. in vehicle locking mechanisms [9]. Designing such devices is often a time-consuming task, which requires expertise. Usually they are created using Computer-Aided Design software, and evaluated with multi-body simulation, finite-element analysis, or by building prototypes.

This work is a step towards development of a GDS for planar mechanical systems with a focus on their dynamics, which can be evaluated using aforementioned multi-body simulation. It will not consider aerodynamic properties involving flow simulation, or material properties using topology optimization because generative systems for these areas do already exist. GDSs are not commonly used when creating dynamic mechanical systems, as there are numerous challenges which need to be addressed simultaneously to produce one. Three of the main challenges are discussed below.

First, the domain of mechanical systems is broad and involves a wide variety of applications and a large number of potential components which could be used to create a design solution. It is difficult to design a general GDS from scratch which copes with this large number of potential components. For this reason, the GDS needs to be limited to a specific set of capabilities, having just enough flexibility to generate design solutions within its' application area. Furthermore, the GDS should provide the possibility to be extended in future.

Secondly, it is difficult to describe a design problem or an aim in a general way for a mechanical system in a machine-readable format. Joskowicz worked on a description language for conceptual mechanical design and mechanical behavior [10, 11], which would allow mechanism performance categorization regardless of their design. However, this work was not linked to GDSs.

A third problem is the development of the GDS itself which is addressed in this work. GDSs use optimization algorithms which require a representation of the design problem and its prospective solutions. The latter require to define the components' shapes, which can be especially problematic. Other GDSs such as used in aerodynamic design or topology optimization have usually a specified outline shape or initial shape which is parameterized or sometimes segmented and can be manipulated by changing a limited number of parameter values. A GDS generating dynamic mechanical systems has no such initial shape or structure to apply changes to. It needs to identify a constellation of components which can work together to solve the design problem with each of them being in the right position and having the right shape. Using multiple components and describing every detail of their shapes in from of coordinates can lead to a large number of parameters which an optimization algorithm may have difficulty coping with efficiently. The number of parameters can be reduced by describing shapes approximately at an abstract level. This work presents four different representation methods and compares their performance to describe a shape. An evolutionary algorithm was developed which uses these representation methods to evolve target shapes. This method is often used as a benchmark problem to investigate shape related optimization algorithm's performance [12–15]. The representation methods were tested with 24 different target shapes, 12 being basic shapes and 12 were created based on the outline of mechanical levers taken from an automotive closure system. This is a first step towards developing a GDS for mechanical systems with a focus on the kinetic properties of the system.

2 Proposed Method

This section introduces the evolutionary algorithm and the fitness function which was used to evolve target shapes using different representation methods. These methods are explained and the experimental setup is presented in detail.

2.1 Evolutionary Algorithm

An evolutionary algorithm (EA) is a population-based algorithm inspired by the natural evolutionary processes. A population of random individuals is initially created. Each individual contains a chromosome. The chromosome used in this work consists of an array of floating-point values (genomes) in a range from 0.0 to 1.0. The chromosomes are mapped to shapes. The EA improves the quality of the fitness of the population in an iterative process. Individuals are selected from the population, copied, and mutations are applied in a systematic way which leads to new individuals called children of which subsequently the fitness is evaluated. Weaker individuals in the population are replaced with those children. One individual represents the best solution. The iterative process continues until a stop criterion is reached, in this case a set number of generations. The pseudo code for the EA used is shown in **Fig. 1**.

```
set POPULATION SIZE to 100
set NUMBER OF CHILDREN per generation to 20
set STOP CRITERION to 500000 evaluations

initialize random population of POPULATION SIZE
identify individual with best fitness in population

run generation loop
    repeat for NUMBER OF CHILDREN
        set PROBABILITY to random number between 0.0 and 1.0
        if PROBABILITY < 0.5
            select two parents from population using binary tournament selection
            create CHILD from both parents using two-point crossover
            apply simple mutation to CHILD
        else
            select one individual from population using binary tournament selection
            make CHILD by coping individual
            apply simple mutation to CHILD
        end if
    end loop

    for each CHILD
        select weaker individual using binary tournament selection
        replace weaker individual in population with CHILD
        if CHILD fitness is better than best individual
            mark CHILD as best individual
        end if
    next child

until STOP CRITERION is reached
```

Fig. 1. Pseudo Code

The algorithm holds a population of 100 individuals and produces 20 children in every generation. Children are created by copying selected individuals from the population using binary tournament selection and through gene mutation. The selection operator picks two random individuals and selects the one with the better fitness. There are two mutation operators which are applied with a probability of 50%. First of them, is a simple mutation which changes between one to four random genomes of a selected individual. The genome values are altered to random new ones, determined by a gaussian distribution based on the previous value. The second operator is a two-point crossover recombination. It takes two individuals from the population using the same selection method and swaps a chromosome segment between them. The segment is determined by two random points defining its' start and end position. This produces one child containing the segment of the first parent and up to two segments at the beginning and the end of the second parent. The children are added to the population by replacing selected individuals of the population using a tournament selection which picks in this case the weaker of two random chosen individuals.

2.2 Fitness Evaluation

The fitness of a candidate solution depends on the similarity between a candidate shape and a target shape. The fitness function is calculated using two penalties. These are based on a comparison of target and solution surface areas. As smaller the penalty value as more similar the candidate shape is to the target shape. **Fig. 2** shows target shape area, the solution shape area, and the intersection between both.

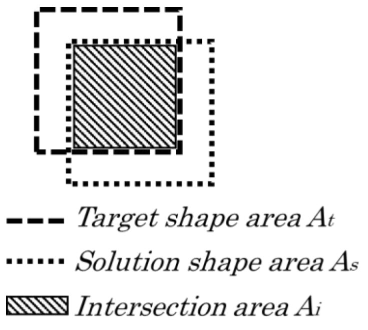


Fig. 2. Fitness Evaluation

The first penalty *Ps* results from a difference between the total area size of target and solution. Eq. 1. shows the calculation of the size penalty *Ps*. The second penalty *Pi* is given for a difference in the intersection area between target and solution and the area of the target. Eq. 2 shows the penalty *Pi* for the intersection. Eq.3 shows the total penalty *Pt*.

$$|As - At| = Ps \tag{1}$$

$$|At - Ai| = Pi \tag{2}$$

$$Ps + 2\,Pi = Pt \tag{3}$$

The intersection penalty is multiplied by two to make the penalties less comparative and to avoid creating local optima as the size penalty *Ps* and intersection penalties *Pi* may work against each other.

2.3 Genetic Representation

Selecting a shape representation is one of the most important decisions in EA based shape optimization [16]. The main purpose of the representations is to define the outline shape of mechanical components. These are subject to three requirements. Firstly, the representation should produce only non-intersecting shapes. Connecting random coordinates to make closed shapes can produce a large number of invalid shapes due to self-intersections which should be avoided as it requires additional processing to resolve them. Secondly, a shape representation should be able to cover a large search space. One representation may be able to produce shapes which another representation is not capable of which may limit the solution space. Thirdly, the representation needs to be compatible with the evolutionary mutation and crossover operators. This is important for the optimization algorithm to be able to traverse the search space efficiently.

In the following section four representation methods are explained. The first three are based on vectors and the fourth is based on rectangles. Each generates control points which are further processed by two functions. The first uses the control points to generate a spline function which makes the shape curvier by adding additional vertices.

Applying spline functions is commonly used in shape optimization [17–20]. The second function removes vertices which are too close to each other with a distance smaller than 90% of the minimum allowed shape size. This approach removes unnecessary aggregation of vertices in one location and enables the shape to contain sharp edges.

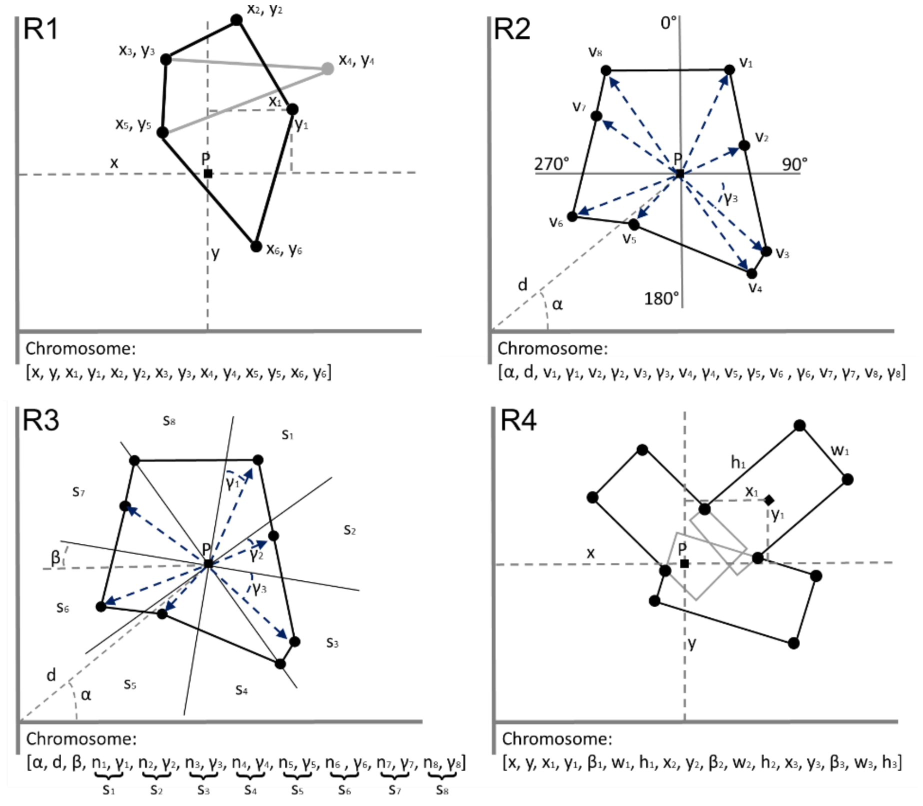

R1
x_2, y_2
x_3, y_3
x_4, y_4
x_1
y_1
x_5, y_5
x
P
y
x_6, y_6
Chromosome:
$[x, y, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, x_5, y_5, x_6, y_6]$
R2
0°
v_8
v_1
v_7
v_2
270°
P
90°
γ_3
v_6
v_5
v_3
v_4
d
α
180°
Chromosome:
$[\alpha, d, v_1, \gamma_1, v_2, \gamma_2, v_3, \gamma_3, v_4, \gamma_4, v_5, \gamma_5, v_6, \gamma_6, v_7, \gamma_7, v_8, \gamma_8]$
R3
s_8
s_1
s_7
γ_1
s_2
β
P
γ_2
γ_3
s_6
s_3
d
α
s_5
s_4
Chromosome:
$[\alpha, d, \beta, \underbrace{n_1, \gamma_1}_{s_1}, \underbrace{n_2, \gamma_2}_{s_2}, \underbrace{n_3, \gamma_3}_{s_3}, \underbrace{n_4, \gamma_4}_{s_4}, \underbrace{n_5, \gamma_5}_{s_5}, \underbrace{n_6, \gamma_6}_{s_6}, \underbrace{n_7, \gamma_7}_{s_7}, \underbrace{n_8, \gamma_8}_{s_8}]$
R4
h_1
w_1
x_1
y_1
x
P
y
Chromosome:
$[x, y, x_1, y_1, \beta_1, w_1, h_1, x_2, y_2, \beta_2, w_2, h_2, x_3, y_3, \beta_3, w_3, h_3]$

Fig. 3. Representations

Cartesian Coordinate Representation (R1). This representation is a basic mapping of chromosome float values to coordinates on a cartesian coordinate system. The coordinates are connected in the order they appear in the chromosome to form a closed shape. The representation is shown in **Fig. 3** (top, left). Each float value is translated to an x or y value between a defined min and max range. The first two genes represent the coordinate of the center of the shape. Further gene pairs represent the coordinates relative to the center. However, connecting each coordinate in the order they appear in the chromosome leads to a very high number of intersections between lines. Self-intersecting shapes cannot be evaluated so the intersections are resolved by adding a post-processing procedure. The procedure removes intersecting lines and closing the shape using the next coordinate. This reduces the number of invalid shapes significantly. The representation was chosen to investigate a direct mapping between chromosome and solution, and the influence of a post-processing procedure to resolve intersections.

Polar Vector Representation (R2). This representation maps floating-point values to vectors with a common center. The representation is shown in **Fig. 3** (top, right). The chromosome’s genomes correspond to directions and lengths of vectors on a polar coordinate system. The directions are mapped to angles between 0 and 360 degrees, and the length is mapped in a range between a defined min and max value. The first two genomes represent the center position of the shape. Further gene pairs represent the vector coordinates which are connected in clockwise direction to avoid intersections in the outline. This representation is inspired by BoxCar2D [21] where it was partly used to describe the shape of an abstraction of a car body.

Hub and Spoke Representation (R3) [22]**.** This representation uses a polar coordinate system similar to the previous described method. It is shown in **Fig. 3** (bottom, left). Floating-point values are mapped to vectors with direction and length. The center of the shape is defined by the first two genomes, and the shape is tilted by an angle specified in the third genome. Each vector has its own angle segment in which it operates. E.g. when using 6 vectors, each would have its own fixed range between 0 and 60 degrees in which it can move. The vectors are connected in clockwise direction to form a shape which makes this approach produce only valid shapes.

Rectangle-based Representation (R4). This representation uses multiple rectangles as basis shapes. It is shown in **Fig. 3** (bottom, right). The first two genes are mapped to a coordinate for the center of the shape. Then, every group of five genes is mapped to a position coordinate, tilt-angle, width and height to create a rectangle. Multiple rectangles are positioned relative to the center. The rectangles can intersect with each other and the overall outline is extracted. A similar representation which uses rectangles as basis shapes was developed by Lee and Nagao [23]. However, there are some differences. First, their representation does not extract the outline of the intersecting rectangles. Instead they force the EA to avoid intersections by giving an additional penalty for them. Second, the representation is not used with additional functions such as the spline function or a procedure to remove close vertices.

2.4 Experiments

Experiments were run 25 times on each of the 24 target shapes for 500,000 evaluations with each representation method. An Intel Core i5-2500 3.3Ghz and 4GB RAM was used for all experiments. The target shapes used are shown in **Fig. 4**. Shape p01 to p12 are lever shapes extracted from an automotive closure system and shape p13 to p24 are general basic shapes.

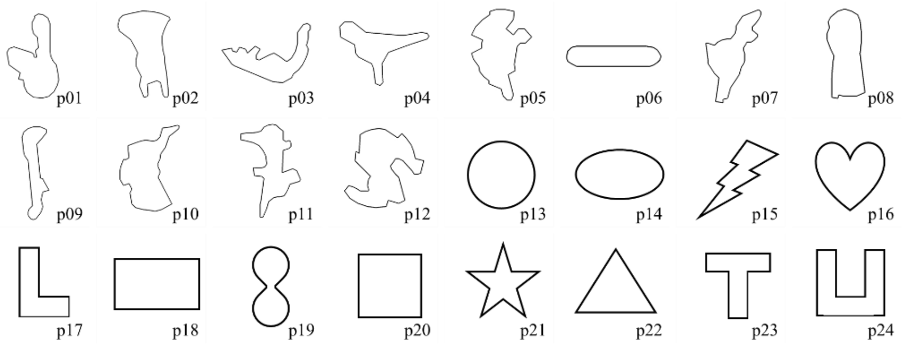


Fig. 4. Target Shapes

The algorithm was configured as follows: The population was set to 100 and the number of children generated per generation was set to 20. There was no parameter optimization conducted, these values resulted from testing and observations. The number of generations was set to 25,000 which results in a total number of 500,000 evaluations after which the experiment was stopped. The chromosome for each representation was set to a fixed length of 77 genes to represent one solution to make the comparison fair.

3 Experimental Results and Evaluation

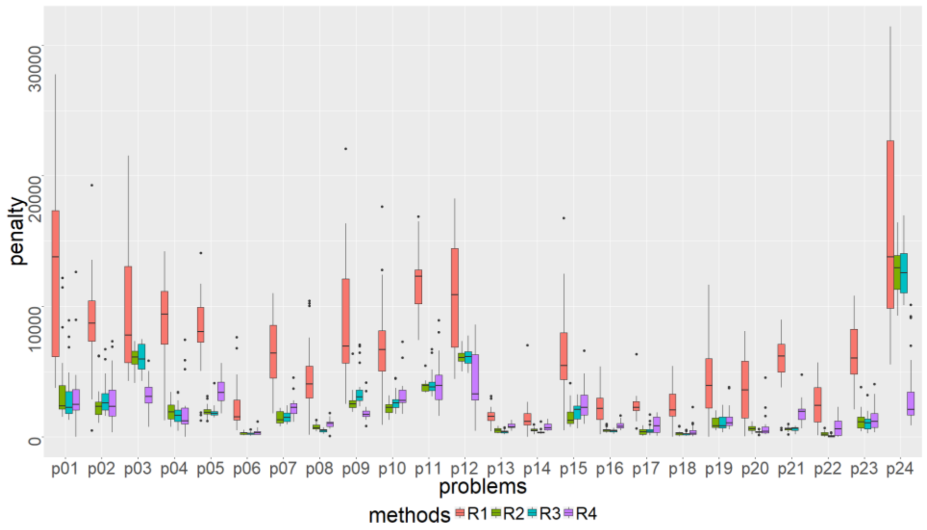


Fig. 5. Method comparison

Fig. 5 presents the comparison of the shape representation methods performance on every target shape. Each of the four tested method's mean penalty through all runs for every problem are listed next to each other.

The significance of the difference of the methods performance was evaluated using Vargha-Delaney A-measure (VDA) [24]. VDA is a statistical test which shows the difference between two non-normally distributed populations. It provides a value (A-measure) between 0 and 1 which indicates if there is a small, medium, large or no difference between populations. A value of 0.5 refers to no difference. A value under 0.44 or above 0.56 indicates a small difference. A value under 0.36 or above 0.64 states a medium difference and a value under 0.29 or above 0.71 indicates a large difference.

Table 1 (lever shapes) and **Table 2** (basic shapes) show the comparison of all methods to each other for every problem providing the A-measure.

Table 1. Comparison of methods using VDA (part 1).

comparison		problems											
method 1	method 2	p01	p02	p03	p04	p05	p06	p07	p08	p09	p10	p11	p12
R1	R2	↓ **0.922**	↓ **0.941**	↘ 0.7208	↓ **0.936**	↓ **0.912**	↓ **0.998**	↓ **0.9776**	↓ **0.955**	↓ **0.966**	↓ **0.96**	↓ **1**	↓ 0.8384
R1	R3	↓ **0.957**	↓ **0.933**	↓ 0.7328	↓ **0.947**	↓ **0.918**	↓ **0.998**	↓ **0.9776**	↓ **0.96**	↓ 0.9008	↓ **0.952**	↓ **1**	↓ 0.8352
R1	R4	↓ **0.958**	↓ **0.923**	↓ **0.981**	↓ **0.949**	↓ 0.8704	↓ **0.984**	↓ **0.9296**	↓ **0.946**	↓ **0.995**	↓ **0.923**	↓ **0.99**	↓ **0.931**
R2	R3	→ 0.6288	↗ 0.3776	→ 0.4864	→ 0.5872	→ 0.5696	→ 0.58	→ 0.4496	↓ 0.8512	↑ 0.1856	↑ 0.3024	→ 0.488	→ 0.5112
R2	R4	→ 0.5344	→ 0.4736	↓ **0.976**	→ 0.632	↑ **0.078**	→ 0.4424	↑ 0.144	↑ 0.288	↓ 0.8928	↑ 0.2176	→ 0.5264	↓ 0.7664
R3	R4	→ 0.4144	→ 0.5648	↓ **0.968**	→ 0.5432	↑ **0.066**	↗ 0.3728	↑ 0.16	↑ **0.055**	↓ **0.938**	↗ 0.3832	→ 0.5568	↓ 0.776
best performance		R2,R3,R4	R2	R4	R2,R3,R4	R2,R3	R3	R2,R3	R3	R4	R2	R2,R3,R4	R4

Table 2. Comparison of methods using VDA (part 2).

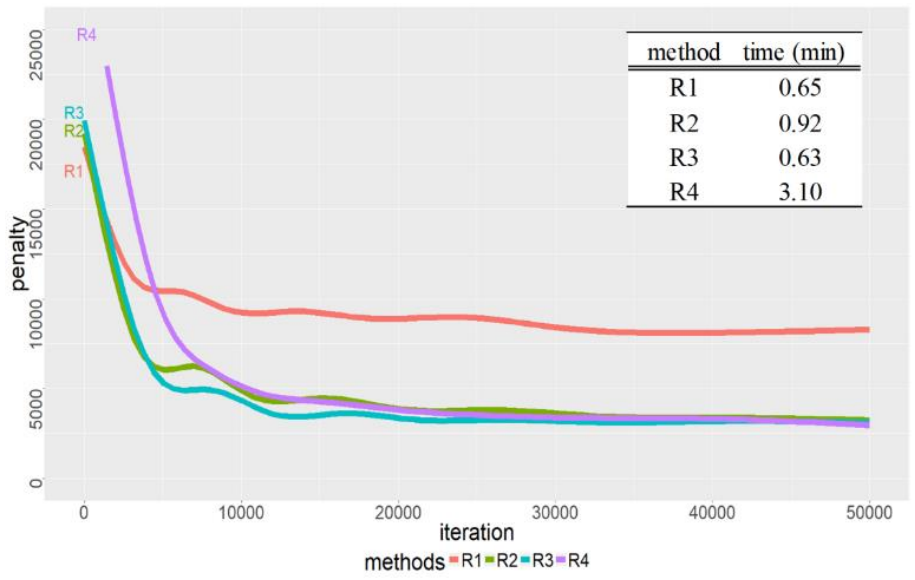
comparison		problems											
method 1	method 2	p13	p14	p15	p16	p17	p18	p19	p20	p21	p22	p23	p24
R1	R2	↓ **0.938**	↓ **0.922**	↓ **0.915**	↓ 0.8496	↓ **0.995**	↓ **0.925**	↓ **0.9248**	↓ 0.896	↓ **0.956**	↓ **0.962**	↓ **0.99**	→ 0.5488
R1	R3	↓ **0.978**	↓ **0.939**	↓ 0.8992	↓ 0.8704	↓ **0.984**	↓ **0.936**	↓ **0.912**	↓ **0.938**	↓ **0.955**	↓ **0.991**	↓ **0.997**	→ 0.5544
R1	R4	↓ 0.864	↓ 0.8016	↓ 0.856	↓ 0.8	↓ **0.93**	↓ 0.888	↓ 0.8784	↓ 0.8848	↓ **0.915**	↓ 0.8336	↓ **0.982**	↓ **0.968**
R2	R3	↓ 0.7272	↓ 0.768	↗ 0.3296	→ 0.6288	→ 0.4016	→ 0.564	→ 0.4984	↓ 0.8784	→ 0.4912	↓ 0.7968	→ 0.5656	→ 0.5024
R2	R4	↑ 0.1856	↑ 0.2488	↑ 0.2368	↑ 0.1192	↑ 0.2336	→ 0.42	↗ 0.3504	↘ 0.6608	↑ **0.038**	↑ 0.304	→ 0.4504	↓ **0.998**
R3	R4	↑ **0.042**	↑ **0.063**	↗ 0.3648	↑ **0.032**	↗ 0.3168	↗ 0.3712	↗ 0.3776	↗ 0.3344	↑ **0.035**	↑ 0.1464	→ 0.3936	↓ **0.998**
best performance		R3	R3	R2	R3	R2,R3	R3	R2,R3	R3	R2,R3	R3	R2,R3,R4	R4

The focus of the comparison was on medium and large differences between the methods. The arrows show if the first method was significantly better (arrow up), or significantly worse (arrow down), than the second method. The diagonal arrows indicate a medium difference, and the horizontal arrows indicate no difference between the compared methods. The last column shows which representation performed best on the particular problem.

Results show that R1 has the worst performance on every problem compared to the other methods. It was found that using a post-processing procedure which disables coordinates to resolve intersections consequently disables parts of the chromosome. It makes the shape optimization inefficient because mutation operators are applied to segments of the chromosome which have no influence on the solution. Furthermore, disabling coordinates also lowers the method's ability to represent complex shapes because less coordinates are left to represent it.

It was also found that R2 and R3 perform mostly equally, with R3 being the best performing solution more often. The reason for this is that both methods have a similar basis and similar range of operation. R2 has a larger flexibility in terms of shapes it can produce. R3 distributes its coordinates in specific sectors whereas R2 can concentrate all its coordinates in one sector. However, R3 has the benefit that large changes in the chromosome lead to smaller changes in the solution compared to R2. The optimization algorithm is better guided by incremental improvements which makes R3 reach a better solution quality. Both methods seem to be slightly better than R4 which is one of the best methods only 8 times, however, the difference is very small. Furthermore, R4 produces significantly better solutions on problem p03, p09, p12, and p24. The shapes in these problems have an undercut characteristic which cannot be produced by the other methods. Nevertheless, R4's performance on p05 and p21 is still good, but worse when compared to R2 and R3, because R4 is based on rectangles which makes it difficult to represent spikes and fine details.

Fig. 6 shows the solution quality increase over time. Each smoothed line shows the found solutions over each iteration of 25 experiment runs and all 24 problems for one representation method.



method	time (min)
R1	0.65
R2	0.92
R3	0.63
R4	3.10

Fig. 6. Improvement over time

Taking a penalty value of 5,000 as a baseline for the solution quality, R3 needs 5,000 iterations, R2 needs 10,000 iterations, and R4 needs around 12,000 iterations to reach the threshold. R1 never reaches the threshold. It shows that R3 is significantly faster in improving the solution quality compared to all other methods.

Fig. 6 also shows the time needed to produce 500000 iterations with each method. R1, R2, and R3 need between 0.63 and 0.92 minutes whereas R4 needs around 3.1 minutes. This is due to R4 requiring more processing for calculating the outline from the intersecting rectangles. It should be noted that the time for a single evaluation is not relevant in the context of this work. A GDS for mechanical systems uses multi-body simulation for the evaluation. It often takes a significantly longer time to evaluate a multi-body system independently to which representation is used.

4 Conclusions

This work is intended as a step towards a GDS for dynamic mechanical systems and focuses on the shape representation of mechanical components used by an evolutionary algorithm. The performance of four different shape representation methods is investigated with the aim to evolve a specified set of target shapes. These consist of mechanical lever shapes taken from an automotive closure system and a set of basic shapes. The first representation (R1) uses a direct mapping of genomes to coordinates and a post-processing procedure to resolve shape intersections. The second representation (R2) maps the chromosome to vectors which are connected in clockwise direction to avoid intersections. The third representation (R3) also maps the chromosome to vectors but allows each vector only to operate in a specified area. The fourth representations (R4) chromosome is mapped to multiple overlapping rectangles of which the overall outline is extracted. Two functions are applied to the resulting shape of the four methods. The first function applies a spline filter to the shape and the second removes vertices which are close to each other. A number of experiments were undertaken to evaluate the performance of each method to produce the target shapes. The performance was compared using the Vagha-Delaney A-measure. Results show that the direct mapping of R1 and resolving intersections in a post processing procedure leads to low quality solutions. The R1 representation is not able to evolve complex shapes. R2 and R3 perform almost equally in terms of solution quality, but with R3 performing slightly better and needing less iterations to reach a better solution quality. R4 is slower than the other representations but can produce similar results to R2 and R4 most of the time. However, R4 is the only representation capable of producing shapes with undercut characteristics to a high quality which can be considered as being more complex shapes.

Future work will investigate the number of genomes needed to generate all the target shapes with the method R3 and R4. A combined representation will be tested, able to switch between R3 and R4. It is hoped that this would benefit from R3's capabilities to find simple target shapes fast but have the ability to evolve more complex shapes by switching to R4 if needed. Furthermore, the representation R3 and R4 will be extended and embedded in a GDS to evolve dynamic mechanical systems.

References

1. Vicini, A., Quagliarella, D.: Airfoil and Wing Design Through Hybrid Optimization Strategies. AIAA J. 37, 634–641 (1999).
2. Arias-Montaño, A., Coello Coello, C.A., Mezura-Montes, E.: Evolutionary Algorithms Applied to Multi-Objective Aerodynamic Shape Optimization. Comput. Optim. Methods Algorithms Stud. Comput. Intell. Vol. 356. 211–240 (2011).
3. Olhofer, M., Jin, Y., Sendhoff, B.: Adaptive encoding for aerodynamic shape optimization using evolution strategies. Evol. Comput. 2001. Proc. 2001 Congr. 1, 576–583 (2001).
4. Pandey, A., Datta, R., Bhattacharya, B.: Topology optimization of compliant structures and mechanisms using constructive solid geometry for 2-d and 3-d applications. Soft

Comput. 21, 1157–1179 (2017).

5. Baron, P., Fisher, R., Tuson, A., Mill, F., Sherlock, A.: A voxel-based representation for evolutionary shape optimization. Ai Edam. 13, 145–156 (1999).
6. Smith, R., Warrington, S., Mill, F.: Shape Representation for Optimisation. 112–117 (1995).
7. Uicker, J.J., Pennock, G.R., Shigley, J.E.: Theory of Machines and Mechanisms. Oxford University Press (2003).
8. Ghassaei, A., Ming, J.: Evolutionary Design of Mechanical Linkages. (2015).
9. Bang, J., Thomas, J.C., Bae, G.O., Lee, D.J., Korea, S.: Optimization of a Hood Latch System. 0–49 (2008).
10. Neville, D., Joskowicz, L.: A Representation Language for Conceptual Mechanism Design. 96–108 (1992).
11. Joskowicz, L., Neville, D.: A representation language for mechanical behavior. Artif. Intell. Eng. 10, 109–116 (1996).
12. Khan, M.S., Ray, T.: A Memetic Algorithm for Efficient Solution of 2D and 3D Shape Matching Problems. 362–372 (2012).
13. Chang, W.-W., Chung, C.-J., Sendhoff, B.: Target Shape Design Optimization with Evolutionary Computation. Proc. Congr. Evol. Comput. 2003. 3, 1864–1870 (2003).
14. Nashvili, M., Olhofer, M., Sendhoff, B.: Morphing methods in evolutionary design optimization. Proc. 2005 Conf. Genet. Evol. Comput. GECCO 05. 897–904 (2005).
15. Tai, K., Wang, N.F., Yang, Y.W.: Target geometry matching problem with conflicting objectives for multiobjective topology design optimization using GA. 2008 IEEE Congr. Evol. Comput. CEC 2008. 1873–1878 (2008).
16. Lampinen, J.: Choosing a shape representation method for optimization of 2D shapes by genetic algorithms. 305–319 (1997).
17. Zhang, P., Yao, X., Jia, L., Sendhoff, B., Schnier, T.: Target shape design optimization by evolving splines. 2007 IEEE Congr. Evol. Comput. (2007).
18. Khan, M.S., Ayob, A.F.M., Isaacs, A., Ray, T.: A novel evolutionary approach for 2D shape matching based on B-spline modeling. 2011 IEEE Congr. Evol. Comput. CEC 2011. 655–661 (2011).
19. Sandgren, E., West, R.L.: Shape Optimization of Cam Profiles Using a B-Spline Representation. 111, 195–201 (1989).
20. Lampinen, J.: Cam shape optimisation by genetic algorithm. CAD Comput. Aided Des. 35, 727–737 (2003).
21. Weber, R.: BoxCar2D, http://boxcar2d.com.
22. Lapok, P., Lawson, A., Paechter, B.: Evaluation of a genetic representation for outline shapes. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17. pp. 1419–1422. ACM Press, New York, New York, USA (2017).
23. Lee, P., Nagao, T.: Hierarchical Description of Two Dimensional Shapes Using a Genetic Algorithm. Proc. 1995 IEEE Int. Conf. Evol. Comput. 2, 1–4 (1995).
24. Vargha, A., Delaney, H.D.: A Critique and Improvement of the " CL " Common Language Effect Size Statistics of McGraw and Wong Author (s): András Vargha and Harold D . Delaney Source : Journal of Educational and Behavioral Statistics , Vol . 25 , No . 2 (Summer , 2000), pp . Pub. 25, 101–132 (2017).