Evolving Planar Mechanisms for the Conceptual Stage of Mechanical Design

Paul Lapok

A thesis submitted in partial fulfilment of the requirements of Edinburgh Napier University, for the award of Doctor of Philosophy

October 2020

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institutes of learning.

The thesis is the result of the student's own independent work.

Abstract

The design of planar mechanical systems is a challenging and time-consuming task for engineers. Evolutionary computing has been shown to be a successful tool applied to design automation in various mechanical engineering fields. This thesis aims to determine how evolutionary computing can be applied in the early conceptual stage to support planar mechanical design. Building on existing work, it investigates suitable evolutionary representations and defines and evaluates a framework for evolving planar mechanical systems in a physics environment. It focuses on the design of representations in a multi-step approach, capable of producing mechanical shapes and mechanisms consisting of several components and linkages, adapting to their environment and able to traverse different landscapes.

Based on a review of the literature on evolutionary computing in design, shape representations, generative design tools, and evolving mechanisms, a generative system was developed, allowing a series of empirical studies to be conducted. Analysis of the results demonstrated the importance of breaking down the representation design into multiple stages. It showed that the implemented representations, combined with the generative tool and evolutionary operators, are capable of evolving solutions for problems with different complexity. The results indicate the representation's large impact on the solution quality, and therefore, careful design is necessary. This work provides insight into design decisions and compatibility with evolutionary computing techniques, offering a promising outlook for using this method to support the conceptual stage of mechanical design. In future, this work has the potential to be developed into an industry tool for assisting engineers in the early stage of planar mechanism design.

Acknowledgements

I want to thank my director of studies Alistair Lawson for introducing me to the possibility of doing a PhD under his supervision and for endless helpful discussions and support throughout it.

I would also like to thank my second supervisor Ben Paechter, for contributing his knowledge, his insightful comments and support in numerous meetings.

-

Also special thanks to my friends,
Emilia Sobolewska, for her valuable criticism, inspiration and companionship;
Sandra Dulisch, for her enormous patience and encouragement;
and finally, Christoph Zmarzlik, for endless nights of fruitful conversations.
Thank you all for always being there for me. Also you, Bex.

-

I would like to dedicate this work to my family, to my beloved parents
Eugen and Bozena Lapok, and my sister Susanne Baumgarten.
Thank you for your love, understanding and support through my whole life.
Furthermore, to my grandparents, especially my grandfather Hubert Łapok†.
You always satisfied my curiosity in knowledge throughout my entire childhood.

Table of Contents

List of Publications

The following publications resulted from this work:

Lapok, P., Lawson, A., & Paechter, B. (2017). Evaluation of a genetic representation for outline shapes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17* (pp. 1419–1422). New York, New York, USA: ACM Press. https://doi.org/10.1145/3067695.3082501

Lapok, P., Lawson, A., & Paechter, B. (2019). 2-Dimensional Outline Shape Representation for Generative Design with Evolutionary Algorithms. In *EngOpt 2018 Proceedings of the 6th International Conference on Engineering Optimization* (pp. 926–937). Springer International Publishing. https://doi.org/10.1007/978-3-319-97773-7

Lapok, P., Lawson, A., & Paechter, B. (2019b). Evolving planar mechanisms for the conceptual stage of mechanical design. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '19* (pp. 383–384). New York, New York, USA: ACM Press. https://doi.org/10.1145/3319619.3322006

List of Figures

List of Tables

1 Introduction

This work focuses on the development of a generative design method for the early phases of planar mechanism design using an evolutionary computing approach. A mechanism is a system of components working together to transfer a given input motion into the desired output motion. Experienced engineering designers require a good understanding of mechanics and mechanical problems to construct them (Pahl, Beitz, Feldhusen, & Grote, 2007). Most commercially produced mechanisms are planar (Myszka, 2012). Meaning, that all the relative motions of the components are in one plane or parallel planes (x, y). In contrast, spatial mechanisms operate in 3-dimensions (x, y, z) (Y. Zhang, 2003).

A typical design process of planar mechanisms includes several stages. It starts with the *conceptual stage* that focuses on producing ideas followed by simple drawings, to solve a problem. Promising solutions are taken further into the *preliminary design stage*, where the overall system configuration is defined, and more accurate evaluations can be made (Ertas & Jones, 1996). The aim is to produce a set of prototypes which can be analysed and selected by engineers for further development. Well-performing concepts are handed over to the *detailed design stage*, where more comprehensive drawings and models of the solution are carried out and tested. A generative design method turns the computer into a design generator capable of producing, visualising, and analysing prototypes to increase the efficiency of this process (Shea, Aish, & Gourtovaia, 2005).

This work investigates a generative design approach concerned specifically with planar mechanisms. Generative design tools include a variety of methods which will be discussed. One of these is evolutionary computing, which provides a range of problem-solving techniques based on the principles of biological evolution, such as natural selection and genetic inheritance (Eiben & Smith, 2015b). Evolutionary techniques have been implemented in a variety of generative design applications, such as aerodynamic shape optimisation (Arias-Montaño, Coello Coello, & Mezura-Montes, 2011; Olhofer, Jin, & Sendhoff, 2001; Vicini & Quagliarella, 1999), or topology optimisation used to improve the material usage of components with a focus on their inner structure (Baron, Fisher, Tuson, Mill, & Sherlock, 1999; Pandey, Datta, & Bhattacharya, 2017).

However, in this work, one faces new challenges, as opposed to other problems which have already been tackled within the generative design domain. The dynamic nature of a mechanical system, including collisions of the outline shapes, is rarely considered and is

lacking an adequate framework to address it. The outline shapes of mechanical components are challenging to parameterise as they can be complex, and the relationship between the position and shape of different components defines how these interact with each other and the environment which results in the performance of the mechanism.

In previous work relating to planar mechanisms (Chen & Chou, 2016; Ghassaei & Ming, 2015; Tsuge, Plecnik, & McCarthy, 2016), the focus was on kinematics and curve tracing to generate mechanisms, but did not consider mass; friction; component shapes; and collisions between components and the environment. Considering these may lead to a new type of generative design tools, able to evolve more detailed mechanisms or even fully-functional mechanical devices.

This work proposes a method for computationally evolving planar mechanisms by breaking down the problem into four stages. The first stage concentrates on evolutionary representations of free shapes. It is important to identify a suitable shape representation which covers the problem domain and works well in combination with an evolutionary algorithm. The second stage concentrates on a physics-based evaluation of potential solutions. For that purpose, the focus was on the design, implementation and subsequently testing of a simulator. The third stage concerns the definition and evaluation of a framework that specifies the assembly of planar mechanisms. The fourth stage builds on the findings of the previous stages and expands the design framework and focuses on evolving potential design solutions for linkages to validate it.

1.1 Motivation and Problem Statement

Mechanisms are an important part of our daily life and are often hidden in many devices such as cars, planes, robots, manufacturing production lines, and more. The design of these mechanisms requires much time and resources. The manufacturing industries are seeking to find ways to make their processes more efficient. For example, the implementation of the Internet of Things and Services, also known under the term “Smart Factory”, received a large focus to lower costs and increase productivity in manufacturing(Wang, Wan, Li, & Zhang, 2016). 75% of manufacturing costs are typically already committed at the end of the conceptual design stage (the initial stage of the design process) (Ullman, 2009). It means that the decisions to optimise the processes at a later stage only influence 25% of the manufacturing costs. More attention to the early design stages is required to address this. For instance, the automation of design can become an important step towards reaching industrial efficiency goals in the future by enabling the creation of outputs at a faster pace. Design automation could shorten the

development time and provide more resources to focus on lowering future manufacturing costs. According to the literature, there are not enough tailored tools to support the early design stages (Zboinska, 2015), and those utilised are more suitable for later stages (Colombo, Mosca, & Sartori, 2007; Robertson & Radcliffe, 2009; Zboinska, 2015). The creative thought process of designers may be influenced negatively by using these tools, which may result in not noticing better suited and more efficient design solutions (Robertson & Radcliffe, 2009).

In summary, using generative tools at early design stages has a high potential to save development time and manufacturing costs; however, there is a lack of support tools tailored for the early design stages. There is no framework for planar mechanisms available that considers mass; friction; component shapes; collisions between components and environment; and to conduct experiments to identify a method to evolve mechanisms for the conceptual or preliminary design stage. A framework such as this might lead to more advanced types of automated design tools.

1.2 Research Questions and Contribution

This work presents an approach for evolving conceptual planar mechanism design prototypes for problems defined by engineers using the computer. It intends to extend the previous work in evolving mechanical designs by considering additional parameters. In this work, a potential design solution consists of multiple interacting components with the freedom to evolve the placement and shape of components. The performance of a solution can be determined from the interactions between these components as a system and with the environment.

The following research questions were defined:

RQ1: Which evolutionary representation can be used to efficiently represent and evolve the shape of planar mechanical components?

RQ2: Which evolutionary representation and evolutionary operators can be efficiently used to represent and evolve mechanical components in a physics environment?

RQ3: To what extent is the evolutionary representation and evolutionary operators able to evolve mechanisms consisting of multiple components with the aim of traversing different landscapes?

RQ4: To what extent are the evolutionary representation and evolutionary operators able to evolve four-bar mechanisms with the aim of traversing different landscapes?

The main contribution of this work is the development, and investigation of a relevant framework, including bespoke software provides a way to specify design problems, including planar mechanical systems. Solutions are created using an approach which has not been done before in this specific domain of planar mechanism conceptual design.

The relationships between Research Questions and thesis contribution are as follows:

RQ 1:

1. A comparison of different genetic representations for shapes including an in-depth investigation of the best performing representation
2. A method to compare, and evaluate different shape representations used in an evolutionary computing context
3. A software tool to run experiments, visualise, and record the process of evolving shapes for mechanical components
4. A set of problems and an analysis of the experimental results

RQ 2:

5. A comparison of different genetic representations for the shape of components used in a physics environment
6. Implementation of bespoke software to run experiments; including visualising, simulating and evolving design solutions
7. A set of problems and an analysis of the experimental results

RQ 3:

8. A detailed description of the problem, including its variables and parameters
9. A scripting language to define design problems
10. A set of problems and an analysis of the experimental results, focusing on evolving planar mechanisms
11. The validation of the framework through the evolution of mechanisms consisting of multiple components

RQ 4:

12. The validation of the framework through the evolution of four-bar mechanisms
13. A set of problems and an analysis of the experimental results, focusing on evolving planar mechanisms

1.3 Thesis Overview

This work consists of seven chapters. Following the introduction in Chapter 1, the second chapter provides a literature review, including a set of relevant definitions. It focuses on shape representations, simulators, generative design tools, and evolving mechanisms. Chapter 3 addresses RQ1; it provides an investigation into different shape representations suitable for planar mechanism design and usable in an evolutionary computing context. Chapter 4 is concerned with RQ2; it presents a simulator capable of resolving physical scenarios, constituting an important part of the design evaluation process. Chapter 5 addresses RQ3, by providing a relevant framework, including the computable model for defining the real-world problem, as well as a simulation tool allowing evaluation of the performance of the mechanism in its' ability to traverse a 2-dimensional landscape. Chapter 6 relates to RQ4; it is based on the previous findings and focuses on a generative system with the capability to evolve four-bar mechanism designs. Finally, Chapter 7 provides the conclusion and future work.

1.4 Summary

This chapter gave an introduction to the early stages of planar mechanism design. It emphasised the complexity of the design process and has highlights that producing well-performing mechanisms takes expertise, time, and resources. This work investigates the possibility of forming solutions computationally by creating a generative design system for planar mechanism design. Evolutionary computing techniques are employed to address the problem. Partial automation of the conceptual design stage can have a large impact on industry design processes, output quality, and costs. However, there is no framework available for evolving planar mechanisms using evolutionary computing.

This thesis addresses the problem in four stages. In the first stage, evolutionary representations were evaluated to find a representation capable of reproducing mechanical shapes. In the second stage, a physics simulator will be implemented and tested with multiple representations. The third stage focused on the specification of a framework for evolving planar mechanisms, evaluated by evolving mechanisms consisting of several components. The fourth stage included validation of the framework through the evolution of four-bar mechanisms.

The research questions and contributions were defined, focusing on the representation of mechanical components and mechanisms in the context of evolutionary computing. The work emphasises the design and implementation of a tool capable of evolving target shapes to compare different shape representations; the implementation of a simulator

including a scripting language to define design problems for experiments; a method to define and analyse experiments around mechanism design; and a validation of the framework. The next chapter provides further background and relevant literature.

2 Literature Review

2.1 Introduction

This chapter provides the background of the research, followed by a literature review and critical evaluation regarding shape representations, generative design tools, and evolving mechanisms.

The background information includes an introduction to early design stages; planar mechanisms; evolutionary computing; and evolutionary representations. Section 2.3 discusses shape representations covering different engineering design fields, as well as methods of representation evaluation. Section 2.4 investigates a method for representation evaluation. Section 2.5 provides insight into generative design tools and discusses their benefits and drawbacks. Section 2.6 concerns evolving mechanisms; it summarises the work done in the field and emphasises the differences between them.

2.2 Background

This section focuses on early design stages, planar mechanisms, evolutionary computing, and evolutionary representations.

2.2.1 Early Design Stages

Engineering design is a broad field of which one area is the design of mechanical systems. The engineering design process consists of phases that differ in the fidelity of a potential design solution at the end of each phase. However, no clear boundaries can be drawn between the phases because solutions are evaluated and re-designed in an iterative manner (Pham & Yang, 1993).

The process starts with the conceptual design stage, which works on an abstract level. Traditionally, during the conceptual stage, a relatively small team of engineers develop ideas and make design sketches. Conceptual design requires creative work utilising novel components, or a combination of known components in a novel way. There is no fixed methodology to follow for conceptual design, and there could be many ways which lead to well-performing conceptual design solutions (Renner & Ekárt, 2003a). However, conceptual design plays a central role in ensuring design quality and innovation (Colombo et al., 2007).

The next phase is the preliminary design stage, where the overall system configuration is defined. Designs at this stage are more accurate and provide a basis for first evaluations (Ertas & Jones, 1996). However, in the conceptual and preliminary design phase, accuracy is not as important as the ability to search for a variety of different designs simultaneously (Cvetkovi & Parmee, 1999). Promising designs are then taken further into the detailed design stage, where technical drawings are made and used to produce the required product. At this point, the design is set and is subject to only minor optimisation efforts (Pahl et al., 2007).

2.2.2 Planar Mechanisms

Planar mechanism design is a specific field in mechanical engineering, undergoing the previously mentioned design stages. A mechanical system typically consists of mechanisms assembled from moving components such as driving components, levers, gears, chains, springs, and others. Most commercially produced mechanisms are planar (Myszka, 2012). This means that all relative motions of the components are in one plane or parallel planes (Y. Zhang, 2003). Components transform input forces and movements to achieve specified forces and movements at the output (Uicker, Pennock, & Shigley, 2003). The challenge of the mechanism design process is to shape components and to assemble them into a system which moves in such a way as to meet the output requirements in response to the given input specifications. The capabilities of the driving component, with the occurring forces in the system, need to be taken into account to make the mechanism fulfil the desired task.

Once a concept which meets the relevant requirements is established (e.g. addressing the design problem with a specific planar mechanism), preliminary drawings are produced and evaluated to identify if those satisfy the requirements. Promising concepts are handed over to the detailed design stage, including technical sketches which are necessary to build prototypes for physical testing. The entire process is iterative and ends with a mechanism which fulfils the required task (Pahl et al., 2007).

Planar mechanisms can be assemblies of individual mechanical components but also assemblies of interconnected components such as Four-bar mechanisms.

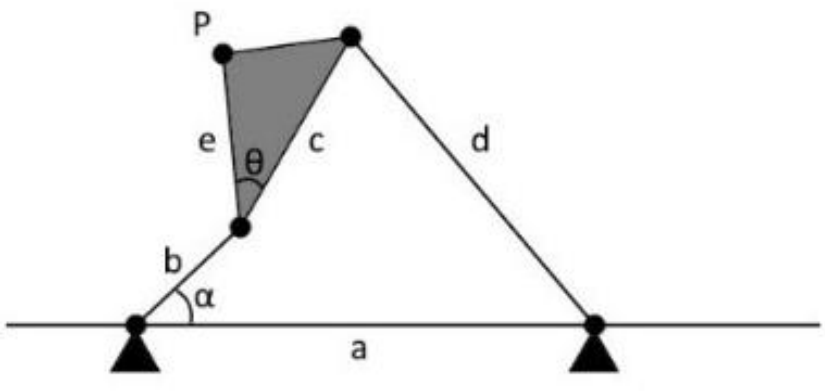


Figure 1: Four-bar Linkage

A four-bar mechanism consists of four parts, such as shown in Figure 1. These are three links and one frame. The *bars b – d* plus *frame a* are connected with four rotation joints to each other. The *frame a* is not movable in space. It keeps two rotation joints on a constant distance from each other. *Bar b* is the driving component and introduces a rotary motion into the system. It connects the frame to *bar c*. *Bar c* is connected to *bar d*, linked to the frame. The tracing point *P* moves relative to *bar c*. It draws a coupler curve in space when the mechanism gets into motion. The same mechanism can produce different coupler curves if *P* has a different position.

2.2.3 Evolutionary Computing and other Approaches in Mechanism Design

Real-world design problems include a large number of design parameters which can be addressed with a variety of approaches. Classical methods, such as gradient methods are often not suitable (Renner & Ekárt, 2003a). For those methods, the optimisation problem would need to be defined by a function to describe the search direction towards the greatest increase, as the design problems may have many local maxima. However, in some specific cases, numerical methods were used (Mariappan & Krishnamurty, 1996), which indeed utilise a gradient method for optimal synthesis of mechanisms. Others applied case-based reasoning (Bose, Gini, & Riley, 1997), a method to store and retrieve design artefacts of functional features to create four-bar mechanisms, was used with the objective to follow defined planar coupler curves. The same problem was tackled with neural networks (Hoeltzel & Chieng, 1990), utilising a system called pattern matching synthesis. A neural network was trained with patterns obtained from parametrically generated coupler curves and retrieved these which best matched the desired curve. Furthermore, path synthesis was also used to generate planar four-bar mechanisms with genetic algorithms (Cabrera, Simon, & Prado, 2002; Roston & Sturges, 1996). However, these methods focused mainly on the kinematic behaviour of linkages without considering interactions with other components via the outline shape.

Metaheuristics, such as Ant Colony Optimization, Evolutionary Computation, Simulated Annealing, Tabu Search and others, are algorithmic frameworks designed to solve complex problems (Bianchi, Dorigo, Gambardella, & Gutjahr, 2009). They are often applied to the class of Stochastic Combinatorial Optimization Problems. Engineers are usually interested in finding the global maximum and in avoiding getting trapped in a local maximum which suggests using stochastic optimisation techniques, such as evolutionary algorithms, as being more suitable, and providing promising toolsets for the automated design of physical systems (Eiben & Smith, 2015a). This is especially the case when considering interactions between shaped components rather than linkages that match specified curves because one function cannot describe the problem. Evolutionary algorithms are inspired by the biological evolutionary process using operations such as reproduction, mutation, recombination, and selection able to traverse a large search space (Renner & Ekárt, 2003a). They work in an iterative manner to identify the best suitable solution for a problem similar to the conventional engineering design process (Pham & Yang, 1993). Renner and Ekárt discussed six categories of applications of mechanical engineering which applied genetic algorithms most; these are conceptual design; shape optimisation; data fitting; reverse engineering; mechanism design; and robot path design. They are especially appropriate for solving complex optimisation problems (Renner & Ekárt, 2003a), which are discussed later in section 2.3.

2.2.4 Evolutionary Representation

In evolutionary computing, an evolutionary representation is the encoding process of transition from genotype to phenotype (meaning from parameter space to solution space). The genotype also called the chromosome, includes genes which are the parameters, while the phenotype is the solution defined by said parameters going through the encoding process. The process can be divided into direct and indirect encoding (Eggenberger-Hotz, 2004).

The concept of direct encoding refers to the relationship between parameter and phenotype attributes. Each parameter of the genotype represents a value of the phenotype directly. Geometric design optimisation, e.g. of lens geometries (Eggenberger-Hotz, 2004); generative CAD design (Krish, 2011); and nozzle geometry optimisation (Genge & Roosen, 2000) applied this type of encoding. For instance, Eggenberger-Hotz, who evolved a lens geometry, utilised an evolutionary strategy employing a direct encoding which performed well for geometrical optimisation. However, the author was not able to get precise solutions for problems with more than 40 parameters (Eggenberger-Hotz, 2004). Using a direct mapping to describe complex shapes requires a large number of

parameters to define all of the details, leading to a significant increase in the search space dimension and processing time to identify well-performing solutions.

On the contrary, indirect encoding is a process which reduces the number of genes needed to represent a phenotype solution. Multiple genes act in combination to evolve phenotypic traits with no direct reference to geometric properties. Often rules are used to describe a growth process. Each rule may influence several phenotype features. Using a lower number of genes and values for each gene reduces the search space and allows applying the evolutionary process to more complex problems. (Bentley & Kumar, 1999)

Indirect representations are often used in Grammatical Evolution (Ryan, Collins, & Neill, 1998), a form of grammar-based Genetic Programming (GP). GP uses grammar guided algorithms which are usually based on decision trees. However, linear representations are used in the wider fields of evolutionary computing and better studied which is an advantage over tree-based representations, as it provides access to a larger background of theory and practice (McKay, Hoai, Whigham, Shan, & O'neill, 2010).

In many cases, indirect encoding outperforms direct encoding due to the reduction of the search space; thus, it reduces the time to find a solution (Bentley & Kumar, 1999; Hotz, 2004). A low number of genes is important as the performance of evolutionary algorithms decreases with an increased number of genes (Eggenberger-Hotz, 2004). Indirect encoding procedures are difficult to design and may cause problems, such as bloat, the growth of unnecessary large trees when using tree-based representations; pleiotropy, which occurs when one gene influences two or more unrelated phenotypic traits; and disruption of child solutions if care is not taken (Bentley & Kumar, 1999). Often multiple encodings are required because the design space is too large to be covered by a single encoding (Krish, 2011).

2.3 Shape Representation

This work requires an evaluation of different types of shape representations to identify their suitability to evolve mechanical systems. Mechanical systems consist of components with various shapes and placements, working together, and contributing towards meeting the system's design task. A representation must describe a solution in a form which is suitable for manipulation by an evolutionary algorithm. It enables a computer to create and optimise the shape of the design. Enduring that similar designs are always close to each other in the design-space (Bentley & Wakefield, 1997).

Representing and optimising shapes is an important element in design where the shape defines the performance of the product. An inefficient optimisation algorithm requires an evaluation of numerous shapes before its convergence, however, a poorly designed shape representation limits the evolution of various shapes, which are both not desirable and require serious attention (Khan & Ray, 2012). A specific research area is focusing on shape matching to address the shape optimisation problem. In shape matching or target shape design optimisation, algorithms and representations are used to evolve predefined shapes as a benchmark problem which allows evaluating the performance of different techniques (Tai, Wang, & Yang, 2008). It shows the importance and difficulty of identifying a suitable shape representation.

This section provides a review of representations used in various engineering fields. In particular, on applications which focus on structural optimisation; on aerodynamic optimisation; and on mechanical optimisation, as these are often utilising evolutionary algorithms to generate solutions. It investigates the benefits and drawbacks of the used representations in these specific areas.

Within the structural optimisation domain, a process called topology optimisation (Bendsøe & Kikuchi, 1988) is used to evolve optimal designs, initially in structural mechanics (Deb & Goel, 2001), then more recently in other engineering domains such as thermal optimisation (Alexandersen, Sigmund, & Aage, 2016); or wave optimisation (Takahashi, Nakamoto, Matsumoto, Isakari, & Kitabayashi, 2018). Topology optimisation focusses mainly on the inner structure, meaning the material distribution of mechanical components. A mesh of pixels or voxels can be used to segment the design space (or initial component). The optimisation process adds or removes material and evaluates the design's performance.

In all areas of topology optimisation, the material, and its distribution is of prime importance. The distribution stipulates the transfer of temperatures, the emission of electromagnetic or acoustic waves, or transmission of forces through the inner material's structure. Often different types of finite element analysis are used to evaluate a design's performance. However, topology optimisation usually focuses on single components and not on complete mechanical systems. In the case of optimising a structure's material stress, a mechanical system, e.g. a mechanism designed of interconnected levers, is broken down into individual components, each analysed individually. Linkages and connection points between levers are not changed, and the transmitting forces and torque between levers are used as input parameters to specify the problem for each component to be able to reduce the material usage. In this way, the kinematic behaviour of the whole

mechanical system stays unchanged. Topology optimisation's main purpose is to generate a new component's structure and not its functionality.

In the field of aerodynamic optimisation, the focus has been placed on the outline of a component, rather than the structure (Arias-Montaño et al., 2011). Designs are evaluated using flow simulation based on Computational Fluid Dynamics. Aerodynamic shape optimisation often uses a direct parameterisation as an encoding method which means that the genotype encodes parameter values directly related to phenotype attributes. E.g. a basic aerofoil shape is parameterised, and limits for each parameter are specified. A search method adjusts the parameters until identifying the optimum design which fulfils the requirements.

In another example, namely the area of mechanical optimisation concerns different types of mechanisms with a focus on mechanical behaviour. Some conventional mechanisms consist of rigid components, and the important part is their outline, rather than their inner structure. The outline defines the behaviour of the mechanism resulting from the interaction between components. A subdomain in mechanical optimisation is compliant mechanisms (Pandey et al., 2017). These are flexible mechanisms which transfer input forces and movement from input to output, through elastic body deformation. They usually consist of one single part where elastic sections act as joints which enable a constrained motion of individual rigid sections. In this case, the focus is on the shape and the structure, as the latter defines the freedom of movement and the former interacts with other individual sections of the mechanism that determines its behaviour.

Some researchers use pixel-grid representations for compliant mechanisms (Sharma, Deb, & Kishore, 2008), others use solid constructive geometries to represent a shape by placing and constraining several nodes within the design domain, using Delaunay Triangulation to generate a skeleton (Pandey et al., 2017). Afterwards, widths are added over the skeleton to produce complex structures. While utilising this method, the outline shape is usually less complex because there are a low number of nodes describing the outline.

Artificial life and robotics are two other areas of interest. Each focuses on evolving morphologies and the control of virtual creatures or robots that can be interpreted as mechanisms as well. For instance, in his work, Sims concentrated upon evolving virtual creatures (Sims, 1994). The focus was on evolving biological behaviour, and biological morphologies built of interconnected blocks rather than detailed shapes. The representation consists of a directed graph where each graph contains the development

instructions for growing a creature with the ability to reuse instructions to create similar recursive components within it. The approach provided a way to create complex structures using a low number of parameters, rather than a direct parameterisation approach. Another example comes from the area of soft robots (Cheney, MacCurdy, Clune, & Lipson, 2013), where the authors successfully evolved walking robots made of soft materials without using a controller. The behaviour resulted from the placement of contracting and expanding materials within the robot using a cellular representation. The representation used a neural network which defined how to assemble the robot. The authors employed a virtual physics environment to study their method. They measured the walking distance in a similar way to the previous work in artificial life.

Reviewing the different representations in these domains makes it possible to identify three different categories of representations:

- Cellular-based
- Direct parameter-based
- Indirect parameter-based

Cellular representations such as pixels or voxels are often used in the engineering design domain when the material distribution is of importance. Pixels represent 2-dimensional shapes, while voxels represent 3-dimensional shapes. Cellular representations focus on inner material distribution and inner structures. The design space is a pixel grid where every pixel represents either material or void. The chromosome, usually a bit or integer string, encodes the states of the pixels.

Direct parameter-based shape representations describe design or shape by direct parameterisation. This category of representations requires an initial starting point, such as a design or shape to be parameterised, such as the profile or an aerofoil with, e.g. width and height, and radii within certain limits. Each change of parameter value changes the design and its performance.

Indirect parameter-based shape representations employ a parameterisation approach as well, usually applied when the problem domain has no initial design to be parameterised. The representation may encode, e.g. building blocks, where a set of shapes is defined, and the chromosome includes information about their placement and orientation. When a chromosome includes a set of coordinates to describe a design, the process of resolving them into a valid solution makes the parameterisation indirect, as even one change in the genotype may lead to more than one change of feature in the phenotype.

2.4 Representation Evaluation

The conventional way to evaluate search space coverage and the effectiveness of evolutionary algorithms and operators is to conduct experiments and compare the performance of different representations. However, evolutionary algorithms require a large number of evaluations because, as previously explained, they refine potential solutions iteratively.

In some cases, evaluations may require computationally expensive simulations, which result in a long runtime. Without conducting experiments, it is not always possible to tell if a representation covers the search space of the domain sufficiently, especially because the optimum solution is often unknown. A way to solve these issues is to investigate the representation outside of the application area, for instance, by evaluating representations in their ability to evolve target shapes instead of conducting experiments in the final application domain. Target shapes have often been used as a benchmark problem to investigate the performance of shape-related optimisation algorithms and representations (Chang et al., 2003; Khan & Ray, 2012; Nashvili et al., 2005; Tai et al., 2008). This approach provides a way to identify the search space coverage without computationally expensive simulation by employing a simpler comparison method lowering the runtime of the evolutionary algorithm and concentrates solely on comparing a candidate shape with a target shape, e.g. taken from the problem domain. The method provides a way to evaluate to what extent a representation can recreate the optimum solution and how effective the genetic operators are in navigating through the search space. It is possible to identify when the algorithm gets trapped in a local optimum. The fast evaluation enables the running of experiments quickly and testing representations with a much higher number of evaluations compared to experiments conducted in the target domain. Furthermore, the visual feedback and comparison provide an insight into the representation's functionality.

2.5 Generative Design Tools

Various researchers (Colombo et al., 2007; Robertson & Radcliffe, 2009; Zboinska, 2015) studied the work of designers and found that they were often using the same tools for conceptual design, as for the detailed design. This concerned Computer-Aided Design (CAD) tools, which are made for the detailed design stage, and are being used to visualise and communicate design ideas as well. The authors emphasise that conceptual and detailed design consists of very different activities, sometimes with conflicting requirements. Krish claimed that CAD is rarely used during the conceptual design stage

which is in contrast to the opinion and observations of other authors, such as (Zboinska, 2015), (Colombo et al., 2007), and (Robertson & Radcliffe, 2009). However, Krish (2011) agreed that CAD software, in its current form, is more useful at a later stage of design (Krish, 2011). According to (Colombo et al., 2007), CAD tools should support the entire design process.

Robertson and Radcliffe (2009) investigated the negative influence of CAD tools on the creativity of engineers in the conceptual design stage. They produced several findings illustrating this problem.

- First, they found that communicating the CAD model might give an illusion of completeness to the design team, which tends to discourage creative thoughts in a group.
- Second, the functionalities of a CAD tool may drive the shape of the outcome solution.
- Third, the time pressure forces designers to generate solutions in the easiest way possible, which drives the design decisions away from what best meets the design criteria, to what is easier to design with the available tools.
- Fourth, the higher the proficiency of the CAD designers is, the more it leads to complex designs. The design philosophy moves away from simplicity and sufficiency to excellence and perfection, which may cause a waste of resources at this stage.
- Fifth, by comparison of two groups, they found that more ideas were generated by the group which did not use advanced 3D CAD tools.
- And sixth, when the design concept became more detailed, there was a strong disincentive to make major changes to the design even if the changes would solve numerous problems or make improvements such as decrease the project risk.

These findings show that there is a need for design tools tailored specifically for early design stages. Automated design tools are needed which provide potential solutions to engineering teams which may eliminate biases appearing while using conventional design tools.

These types of applications, called generative design tools, used in engineering, are often based on evolutionary computing techniques. They are employed to evolve specific mechanical components rather than systems of components, such as flywheels (Eby, Averill, Punch, & Goodman, 1999), rotor shafts (Byung Gun Choi & Bo Suk Yang, 2000), aerodynamic structures (Arias-Montaño et al., 2011; Gaier, Asteroth, & Mouret,

2018), trusses (J. Liu & Ma, 2017), lenses (Li, Zigoneanu, Popa, & Cummer, 2012), and many more.

Some generative tools in engineering design focus on dynamic systems of components, such as linkages (Chen & Chou, 2016; Y. Liu & McCarthy, 2017; Tsuge et al., 2016), cams (J. Lampinen, 2003; Mundo, Liu, & Yan, 2006), wing folding mechanisms (Jitsukawa, Adachi, Abe, Yamakawa, & Umezu, 2017), gears and gear drives (Padmanabhan, Chandrasekaran, Ganesan, Patan, & Navakanth, 2017), or other areas of mechanism synthesis (Cabrera et al., 2002; Kyung & Sacks, 2006). Usually, key geometries of these components or systems are parameterised, mutated for a reasonable time, and evaluated until finding a well-performing design. These applications simplify the real-world problem by focusing on the kinematic behaviour of mechanisms which supports the design process. However, these tools are not looking at collisions between shaped components, their masses, and are not considering friction. Including these physical attributes adds a new layer of complexity which moves generative design one step closer towards physical mechanisms and will be covered in this work.

2.6 Evolving Mechanisms

The manufacturing industry provides toolboxes of drives, gears, joints, and other machine elements to build a variety of robots such as those recently presented by the company Boston Dynamics. The bio-inspired quadruped robots, such as Boston Dynamics machines (Raibert, 2008), were created by human designers, however, in future, they could be automatically generated by a machine, as evolutionary computing is starting to make a transition towards automated creation of physical artefacts (Eiben & Smith, 2015a).

Robots are mechanical systems constructed with connected mechanisms and evolving such mechanisms is a step towards reaching the goal of design automation of complex machines. Recent work focused on walking robots, such as on bipedal robots (Ambrose, Ma, Hubicki, & Ames, 2017; Ames et al., 2017; Lawati & Yousef, 2016), quadruped robots (Digumarti, Gehring, Coros, Hwangbo, & Siegwart, 2014; Ruan, Wu, Zhou, & Yao, 2015; Vishal & Manivannan, 2016), and hexapod robots (Belter & Walas, 2014; Cully & Mouret, 2016; Roennau, Heppner, Nowicki, & Dillmann, 2014). Others focused on modular robots (Kamimura et al., 2005), and snake-like robots (Kohl, Kelasidi, Mohammadi, Maggiore, & Pettersen, 2016; Reyes & Ma, 2014). Most of these work in robotics used evolutionary computing techniques to evolve the machine controllers but not to evolve the physical design of the robots. The physical design is usually predefined,

and the focus is on evolving the robot's control pattern. Evolving robot designs is still not well researched as there is little literature regarding generative mechanical design focusing explicitly on evolving robots' mechanics. However, the relationship between design topology and control patterns influences the performance of a mechanical system. In an ideal scenario, these should evolve together.

A mechanism is a system of interconnected components which produces complex behaviour when movement is introduced. They are versatile and can be assembled with connected and not connected levers, gears, chains, springs, joints, and more. The operating principle which determines the mechanical behaviour is the transfer of forces and moments through contact or linkage. At early design stages, complex mechanisms are often abstracted at a 2-dimensional level by putting their kinematics in the foreground. Notably, most commercially produced mechanisms are planar (Myszka, 2012). The objective is to design a system which meets the desired behaviour, or at least a behaviour which is sufficient (Renner & Ekárt, 2003a). Designing mechanisms, such as linkages required to perform desired motions, is a highly unintuitive process. It often involves rigorous experimentation in a high dimensional parameter space usually intending to fit designer specified curves (Ghassaei & Ming, 2015; Tsuge et al., 2016). However, linkage design does not consider shapes of components, collisions between them, and their physical attributes, such as mass or gravity. Their inclusion would further complicate the design process and thus would require generative design tools.

A class of important mechanisms are four-bar mechanisms. Their utilisation ranges from simple devices, such as windscreen-wiping or door-closing mechanisms, to complicated ones, such as rock crushers, sewing machines, round balers, and suspension systems of automobiles (Renner & Ekárt, 2003a). Four-bar mechanisms have been evolved using a genetic algorithm (Roston & Sturges, 1996), and further through employing a case-based reasoning approach (Bose et al., 1997). Ghassaei and Ming focused on evolving four-bar linkages for two scenarios, curve fitting, and task fulfilment, in this case, walking (Ghassaei & Ming, 2015). They proposed a novel software system that allows users to visualise and interact with the various optimisation parameters. The authors considered gravity and collisions between the mechanism and the environment when evolving walking behaviour. However, the mechanism consists of bars without specifically evolving shapes to interact with the environment or other components of the mechanism. The authors state that the search space of the problem is very large.

Within the engineering domain, there are also more complex mechanisms and synthesis of mechanisms. Collision-free adjustable six-bar linkages were synthesised using a twin-

space crowding genetic algorithm (Chen & Chou, 2016). Six-bar linkages were used to evolve lower limbs (Tsuge et al., 2016) and also to evolve manufacturing mechanisms (Chen & Chou, 2016). Mechanisms were evolved to draw algebraic curves (Y. Liu & McCarthy, 2017), also wing fold mechanisms (Jitsukawa et al., 2017), and even eight-bar mechanisms (Parrish, McCarthy, & Eppstein, 2015).

Another category of mechanisms to consider are cam mechanisms. A cam is a rotating or sliding piece in a mechanical linkage used, especially in transforming rotary motion into linear motion (Uicker et al., 2003). In contrast to evolving linkages, cams transfer forces through contact and collision with other components. Cam shapes have been optimised using a genetic algorithm (J. Lampinen, 2003); they have also been generated for precise path generation (Mundo et al., 2006). However, as previously, this work considers only the kinematic properties to evolve an assembly which follows a specified path. The cams were modelled to be always in contact with a follower, which simplifies the problem by avoiding the necessity to resolve collisions. Furthermore, it does not consider friction between components. The focus is on rotating cams and not on shaped components moving through the design space.

Research has also been proposed regarding the evaluation of the behaviour of mechanical systems. Jaskowicz suggested a behaviour language for mechanical systems, for comparing different systems, such as gears, or systems with a different type or number of components, based on their resulting behaviour, which is mainly described by the output motion (Joskowicz, 1999). Being able to specify objectives and evaluate mechanical systems is a crucial part of creating generative design systems for mechanism design.

2.7 Summary

First, this chapter introduced the research background, explaining the conceptual design stage, planar mechanisms, evolutionary computing and evolutionary representations. This was followed by the critical evaluation of the literature in the area of shape representations, generative design tools and evolving mechanisms.

The generative design tools were reviewed; it was shown that there is a shortage of applications targeting the early conceptual design stages. Furthermore, human designers tend to be biased by the available applications. It showed that tools suggesting a broader range of solutions might be beneficial for engineers to reduce their bias and workload.

Evolutionary computing provides a promising toolset to evolve mechanisms. Tackling this problem with a generative design tool based on evolutionary computing requires an

evolutionary representation capable of creating mechanical shapes. Different types of shape representations were reviewed concerning principles such as cellular, direct, and indirect representations. Such representations specifically focusing on mechanical components, are not available and require to be designed and evaluated. It was found that an indirect encoding is most suitable for shape representations in planar mechanism design when applied it in an evolutionary computing context. It provides a way to define complex shapes with a low number of genes. In section 2.4, a method to evaluate the ability of shape representations to create target shapes was found. It is a computationally inexpensive process to identify and evaluate representations' abilities to be applied within an evolutionary algorithm to create shapes for a specific problem domain. These findings led to RQ1.

RQ1: *Which evolutionary representation can be used to efficiently represent and evolve the shape of planar mechanical components?*

The area of engineering optimisation was reviewed with an emphasis placed on structural, aerodynamic, and mechanical optimisation, presenting relevant solutions and examples of representations employed in these fields. Every area has its own unique way of describing the problem domain and evaluating potential solutions. The literature review showed that generative tools in mechanisms design mostly focused on evolving the kinematics of mechanisms or the control patterns, without including attributes such as mass and friction, or collisions between components. Considering these would allow generating mechanisms which are closer to physical systems. For that purpose, a representation is needed and a simulation environment and a design objective to evaluate it. No suitable simulation environment could be identified in the area of planar mechanism design which is compatible with evolutionary computing techniques. However, the design objective of measuring the walking distance which was used in artificial life and robotics appears to be suitable for mechanism design. It is wide-ranging and would provide a foundation for experiments to evaluate the representation in combination with an evolutionary algorithm. These findings led to RQ2.

RQ2: *Which evolutionary representation and evolutionary operators can be efficiently used to represent and evolve mechanical components in a physics environment?*

Section 2.6 investigated the achievements and difficulties in mechanism design. Mechanism design mostly focused on linkages, rather than the shape of components, or interaction between multiple not linked components. However, planar mechanisms may consist of multiple not linked components. The available literature does not provide a

framework to describe these. It is required to employ an evolutionary representation and investigate its ability to evolve solutions within an evolutionary computing context. These led to RQ3 and RQ4, which focus on mechanisms consisting of multiple components and linkages.

RQ3: *To what extent is the evolutionary representation and evolutionary operators able to evolve mechanisms consisting of multiple components with the aim of traversing different landscapes?*

RQ4: *To what extent are the evolutionary representation and evolutionary operators able to evolve four-bar mechanisms with the aim of traversing different landscapes?*

The next chapter focuses on the evaluation of several shape representations to identify a suitable representation capable of reproducing shapes for mechanical components.

3 Evolutionary Shape Representations for Mechanical Design

3.1 Introduction

This chapter presents four different genetic representations for describing two-dimensional outline shapes and an investigation of their suitability to be used in a generative design system. The evaluation focuses on their ability to evolve a set of defined target shapes. These shapes consist of simple symmetric and asymmetric shapes with edges and curves, as well as more complex mechanical component shapes derived from the problem domain, namely from an automotive device. The representations are used to approximate target shapes using an evolutionary algorithm with crossover and mutation operators.

As explained in Chapter 2, planar mechanisms consist of mechanical components which transfer movement and forces via their outline shape. In other words, the function and performance of a mechanism rely on the shapes of its components and their interactions. As shapes play a significant role in mechanism design, it is important to find representations that work well within an evolutionary computing context.

A shape representation method should cover a reasonably sized search space of the problem domain; produce only valid solutions, and allow an evolutionary algorithm to navigate through the search space. The research will investigate the following criteria:

- Search space dimension
- Search space coverage
- Search space validity
- Search space navigation

The following section provides the background to this chapter.

3.2 Background

The performance of many engineering applications is highly dependent on functional shapes. A generative design system uses optimisation algorithms which require a representation of the design problem and its prospective solutions. The latter requires a definition of the components' shapes, which can be especially problematic within an evolutionary computing context. As shown in the literature review section 2.3,

mechanical engineering design is using many different shape representations, such as cellular representations, used in topology optimisation; direct parameter-based representations, used in aerodynamic shape optimisation; and indirect parameter-based representations, used in robot design. However, these representations often do not describe the outline of a shape. Instead, the outline is a fixed constraint defined in the problem domain. A common approach is to parameterise certain features of a basic shape which was done, for instance, in design optimisation, e.g. of aerofoils. Changing the parameter values within defined boundaries adjusts the shape until reaching the optimum. However, such basic shapes might not exist for many areas of mechanism design where components are customised to create mechanisms that meet specific path and force characteristics. The shape of a mechanical component is entirely a result of its function.

It is tempting to think that the best approach would be to encode the coordinates of such free shape directly into a chromosome, as it can result in any shapes. Although, this approach would require a large number of parameters as a certain degree of complexity is needed, which in turn, would increase the dimension of the search space (Chang et al., 2003). Furthermore, most solutions in the search space would be invalid due to intersecting outlines. A component with an invalid shape cannot be evaluated and does not return any fitness value. The evolutionary algorithm is not able to navigate through the search space without a fitness value. Therefore, there are several things to consider when designing a shape representation. This research emphasises search space dimension; coverage; validity; and navigation.

The size of the search space is determined by the number of genes in the chromosome and the value range of each gene, which means, the total number of possible value combinations. A low number of possible combinations results in a small search space, as opposed to a large number of potential combinations, resulting in a large search space. The coverage of the search space is determined by the representation's capabilities and limitations to reassemble shapes of the problem domain. The evolved shapes need to be valid, which means they should not contain intersections, as only valid solutions return a fitness value. Evolutionary operators need to be able to navigate through the search space, able to reach all areas of the search space, in a way that similar size changes on the genotype lead to equal size changes on the phenotype. The following section explains the principles in more depth.

Bespoke software is developed and utilised to conduct a series of relevant experiments. It was employed for target shape matching and investigating algorithms, operators, and representations.

3.2.1 Search Space Dimension

The evolutionary computing context requires a consideration of the dimension of the search space, determined by the number of possible combinations between the number of genes used and the number of possible gene values. A larger search space has a serious impact on the performance of evolutionary computation (Chang et al., 2003). A small as possible search space is prefered; hence, the number of genes, and also the number of gene values should be kept as low as possible. An indirect parameter-based approach can be used to reduce the number of genes. In this case, one gene describes multiple features of a solution.

3.2.2 Search Space Coverage

The search space dimension defines the number of potential solutions and solution types that the representation can produce. Designing a representation requires investigating whether the search space covers the problem domain, meaning, the representation's capability of producing solutions of certain types. The investigation of the coverage is an important step, as it is often not visible if the problem domain is sufficiently covered.

The following two simplified thought experiments should illustrate the problem: Imagining the search space of the problem domain being a canvas which would allow all possible shapes to be drawn on. A representation consists of a chromosome with a limited number of parameters, and a routine to translate these into a drawing. It results in the number of potential drawings being limited. However, the canvas allows drawing an infinite number of shapes, depending on resolution and complexity. It means one shape representation alone is never able to represent all of the possible shapes; thus can often not cover the whole problem domain. A representation describes a subset or specific type of solutions, and its limitations need to be determined.

A more practical example comparing two representations to each other: The problem is to design the shape of a wheel for a car, (imagining not knowing the shape of a wheel). One representation consists of two descriptive parameters, each describing a side length of a rectangle — another representation consisting of just one parameter describing the radius of a circle. In this case, just one of the representations is capable of sufficiently covering the search space of the problem domain and producing accurate solutions. However, both representations will return a solution. The representation using one parameter will evolve a wheel and achieve high performance. The rectangle representation will evolve a square-shaped wheel which will also achieve some performance, yet, not the optimum, even though it uses a larger number of parameters to

describe a solution. The problem is that the optimum is unknown, that means one may assume a square-shaped wheel is a good idea when using the wrong representation.

Moreover, extending this thought experiment to more complex problems with complex representations; it gets more difficult to identify if a representation covers the problem sufficiently. A systematic investigation of the coverage can lead to designing better performing representations. An approach may be to test representations with a benchmark problem for which the optimum is known.

3.2.3 Search Space Validity

A representation should always produce valid solutions and avoid infeasible shapes when mapping from genotype to phenotype. Figure 2 shows a valid shape on the left-hand side and an invalid shape due to a crossover of lines on the right-hand side.

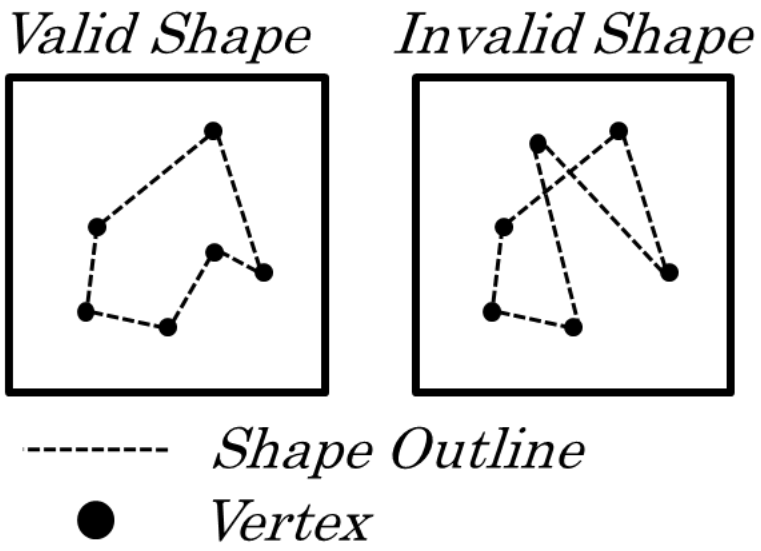


Figure 2: Shape Validity

There are theoretically more invalid shapes which cannot be evaluated than valid ones if taking the simple approach of simply connecting a string of coordinates into a shape. The existence of invalid solutions in the search space indicates that it is unnecessarily large and contains invalid areas. It is not possible to evaluate invalid solutions, and without providing the evolutionary algorithm with a fitness value, the navigation through the search space is impossible or at least ineffective. However, if an unnecessarily large search space is acceptable, the approach to avoid invalid solutions would be either to filter them or to repair them, which requires additional computational time and resources. For this reason, genotypes should produce solutions that can be evaluated by the generative system.

3.2.4 Compatibility with Evolutionary Operators

An evolutionary algorithm should be able to efficiently navigate the search space using evolutionary operators such as mutation and recombination. Furthermore, the operators should not introduce too large disruption and should be able to pass features from parent solutions to child solutions. Regarding mutation, a small change in the genotype should lead to a small change in the phenotype. Recombination operators should keep some characteristics of the parents, such as shown in Figure 3.

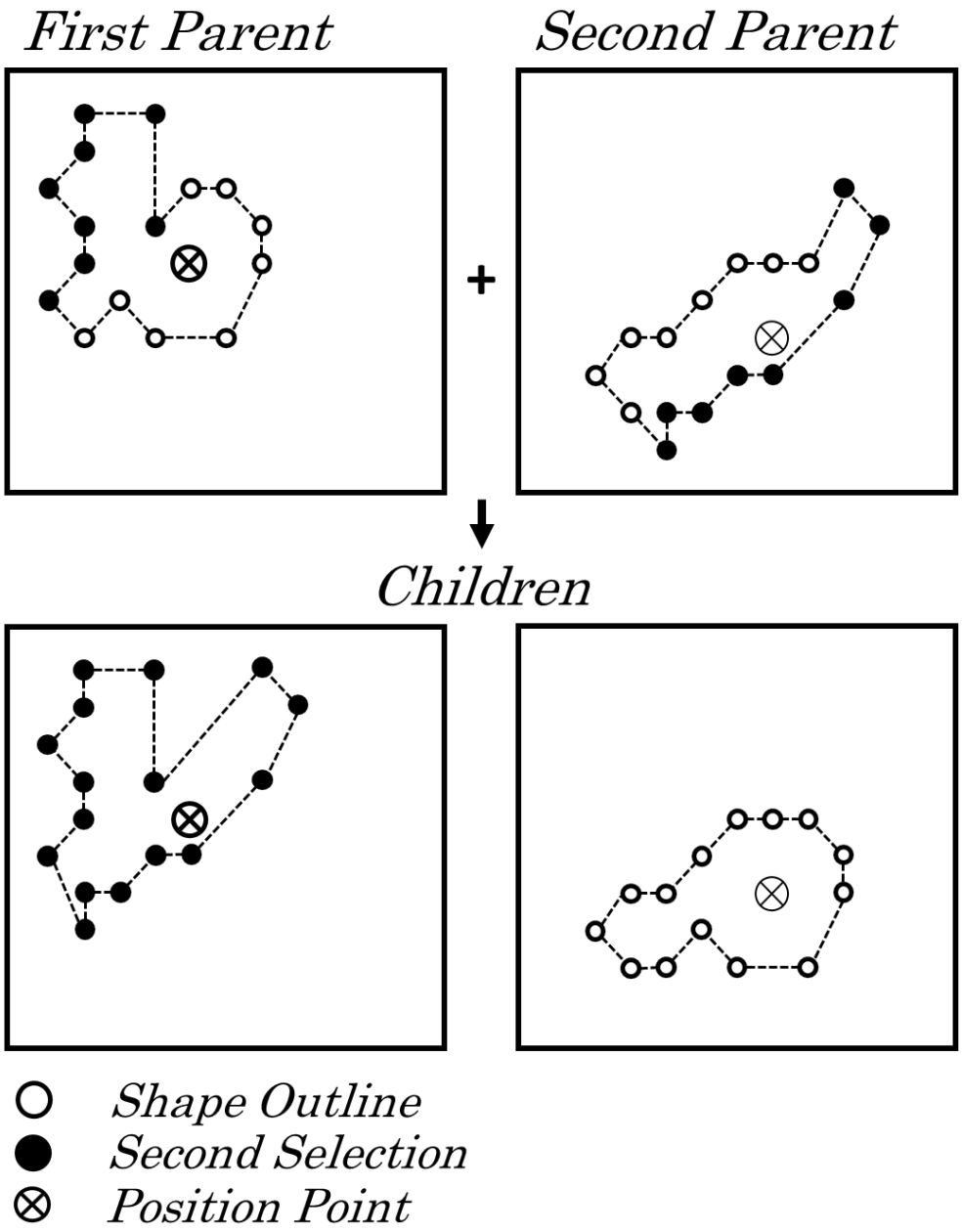


Figure 3: Recombination Operation

A recombination operation produces a swap of features between two-parent phenotypes which results in child solutions. If these operators are not working properly together with the representation, the algorithm will not be able to evolve the solution.

3.2.5 Applications for Experiments

Two software applications were implemented, providing the ability to conduct experiments. Figure 4 shows the first application. It is used to define the target shapes.

Figure 4: Target Shape Definition

The application has a graphical user interface and enables loading images, e.g. of levers, into the view. The image is used to extract the outline of the lever and obtain its coordinates used to define a target shape for the next application by copying them into a text file. The second application for target shape matching provides the capability to conduct experiments and is shown in Figure 5.

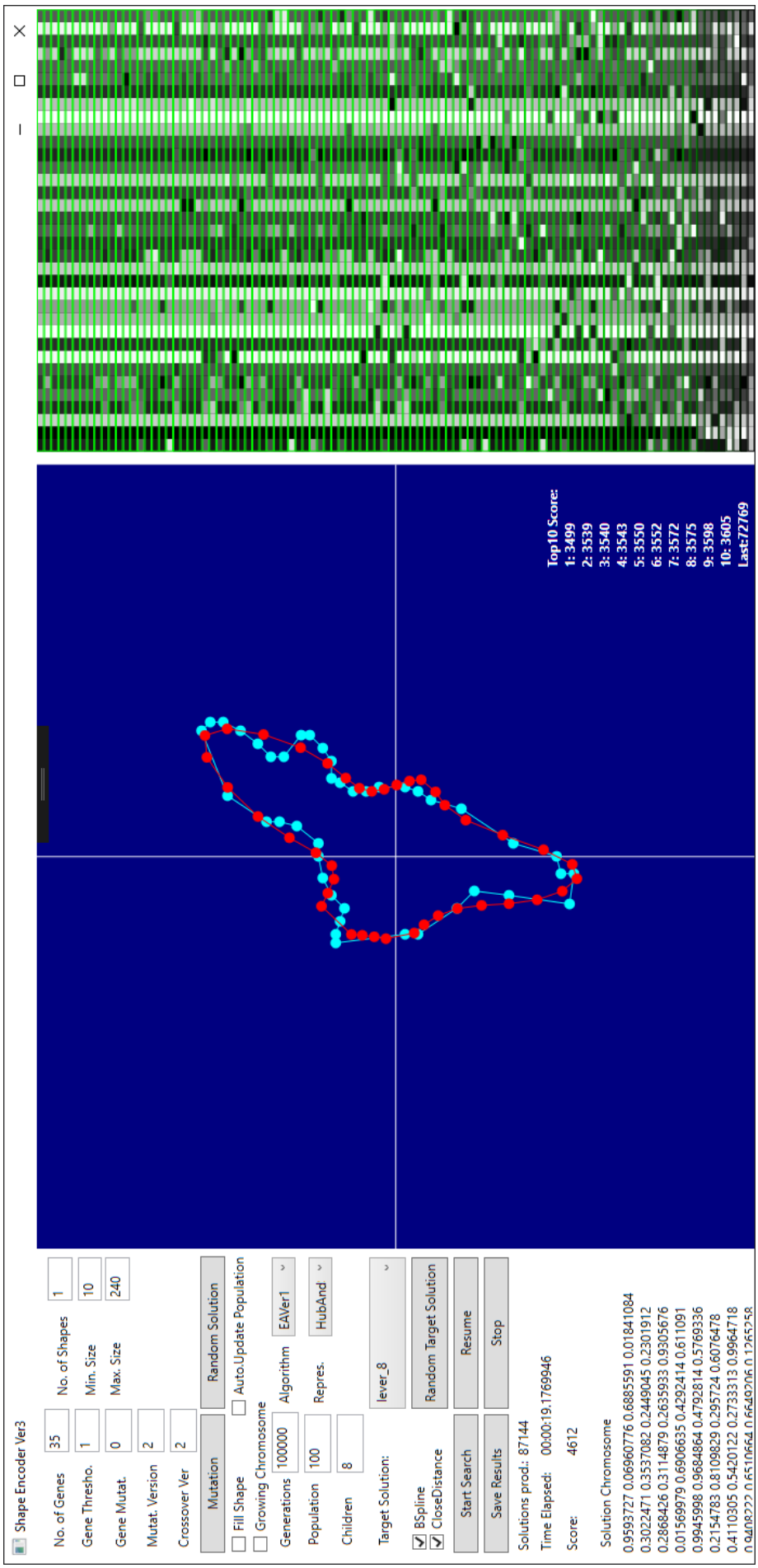


Figure 5: Target Shape Matching Application

This software also provides a graphical user interface and embeds the evolutionary algorithm. It uses the text file with the target shape coordinates and loads it into the view (middle section). Algorithm related parameters can set on the left-hand side. The interface offers a way of setting the number of genes (length of the chromosome) to be used for a solution and the number of individual shapes to be evolved. It provides a way to choose different genetic operators which were implemented for testing and to set different thresholds related to the operators and the solution. Examples may include minimum and maximum size.

Different evolutionary algorithms and representations can be chosen, including the maximum number of generations, population size, and the number of children per generation. Filters can be activated or deactivated, as detailed later in this section 3.3.2. After starting the algorithm with the specific configuration, it updates the view each time finding a better performing solution. Moreover, it provides information such as the chromosome values of the best-performing individual and the fitness of the ten best individuals. It also shows the run time and the number of produced solutions. Additionally, buttons are offered to pause the search and save the results.

The right-hand side of the interface provides a visualisation of the population of the evolutionary algorithm. Every row stands for one individuum of the population with every column representing a gene. The algorithm uses a real value chromosome, and each gene is colour coded in greyscale from white to black with similar values having a similar colour.

The software is the basis for experiments to test algorithms, evolutionary operators, and representations. The user interface enables spotting problems with the algorithm and to identify a well-performing setup.

3.3 Method

Rather than testing each representation in the final application domain, they were tested for their capability to evolve target shapes using an evolutionary algorithm. This method has often been employed as a benchmark problem to investigate shape-related optimisation algorithm's performance (Chang et al., 2003; Khan & Ray, 2012; Nashvili et al., 2005; P. Zhang, Yao, Jia, Sendhoff, & Schnier, 2007). It supports the development of representations by giving an insight into their underlying principles and the evolutionary process. Testing the representations directly in the final application domain does not allow this level of depth and would require a significantly higher computational

processing effort that is caused by the necessary physics simulation. It would make it difficult to understand certain behaviour of the evolutionary computation regarding evolving the shape, due to lack of visibility of prospective solutions.

For this research, four representations were developed and tested with 24 different target shapes, 12 being free invented shapes and 12 based on the outline of mechanical levers taken from an automotive closure system. The next section presents the target shapes. Subsequently, the description of the representations is shown, followed by the evolutionary algorithm and the fitness evaluation method. Last shows a presentation of the experiments to compare the representations' performance.

3.3.1 Target Shapes

Evolving target shapes allows developing a representation systematically; it provides a way to justify design decisions; allows to compare different representations to each other; supports the choice of a representation for a specific problem, and it is a useful tool to identify a representation's drawbacks. The representations are evaluated based on their ability to evolve different shapes within a mechanical engineering context, with characteristics such as corners; curves; symmetries; a-symmetries; and other problem domain-specific features. A set of target shapes was defined for evaluation of the representations abilities to cover the shape characteristics.

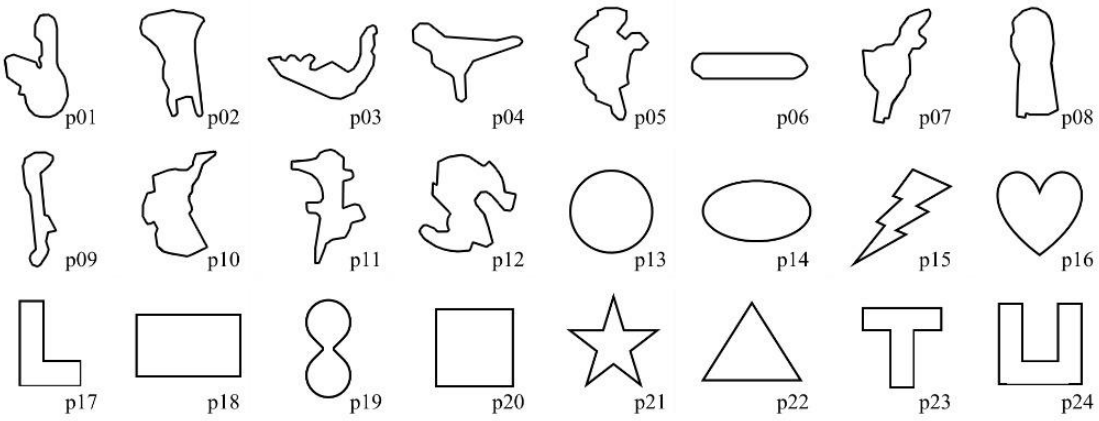


Figure 6: Target Shapes

The illustrations in Figure 6 were used to test the capability of the representations to reproduce these shapes and to identify to what extent the produced solutions are valid. This approach has given an insight into the compatibility of the representation with the evolutionary operators.

3.3.2 Evolutionary Representation

Selecting a shape representation is one of the most important decisions in evolutionary computing based shape optimization (Jouni Lampinen, 1997). In this work, the

representation's purpose is to define the outline shape of mechanical components subject to four requirements.

Firstly, a shape representation should have a small number of variables representing a solution to be used within an evolutionary computation efficiently. Secondly, it should cover a large search space of the problem domain. It is necessary to remember that one representation may be able to produce solutions which another representation is not capable of producing. Thirdly, the representation should return only valid, non-intersecting shapes. Connecting random coordinates to generate a closed shape results in a search space with a large number of invalid solutions should be avoided. It increases the size of the search space; makes navigation difficult due to not evaluable solutions; or requires additional processing to resolve intersections. Lastly, the representation needs to be compatible with the evolutionary mutation and recombination operators to enable the evolutionary algorithm to navigate through the search space efficiently.

Four representations were developed each based on interconnected vertices produced by a spline function which makes the solution shape curvier. The spline function uses control points encoded in a chromosome, which is a common approach in shape optimisation (Khan, Ayob, Isaacs, & Ray, 2011; J. Lampinen, 2003; Sandgren & West, 1989; P. Zhang et al., 2007). The algorithm for the spline function is shown in Figure 7 and Figure 8.

```
// Convert control points to spline
public static List<Point> BSpline(List<Point> coordinates)
{
   List<Point> shapeCoordinates = new List<Point>();
   if (coordinates.Count >= 4)
   {
      // Loop through all control points and close circle
      for (int i = 0; i < coordinates.Count; i++)
      {
         int a = i;
         int b = i + 1;
         int c = i + 2;
         int d = i + 3;

         if (d >= coordinates.Count) d = d - coordinates.Count;
         if (c >= coordinates.Count) c = c - coordinates.Count;
         if (b >= coordinates.Count) b = b - coordinates.Count;

         List<Point> spline = bSplineAlgorithm(coordinates[a],
         coordinates[b], coordinates[c], coordinates[d], 3);
         shapeCoordinates.AddRange(spline);
      }
   }
   else shapeCoordinates = coordinates;

   return shapeCoordinates;
}
```

Figure 7: C# Code Spline Function A

```
// Convert contrlpoints to spline
private static List<Point> bSplineAlgorithm(Point p1, Point p2,
Point p3, Point p4, int divisions)
{
   List<Point> spline = new List<Point>();
   double[] a = new double[5];
   double[] b = new double[5];
   a[0] = (-p1.X + 3 * p2.X - 3 * p3.X + p4.X) / 6.0;
   a[1] = (3 * p1.X - 6 * p2.X + 3 * p3.X) / 6.0;
   a[2] = (-3 * p1.X + 3 * p3.X) / 6.0;
   a[3] = (p1.X + 4 * p2.X + p3.X) / 6.0;
   b[0] = (-p1.Y + 3 * p2.Y - 3 * p3.Y + p4.Y) / 6.0;
   b[1] = (3 * p1.Y - 6 * p2.Y + 3 * p3.Y) / 6.0;
   b[2] = (-3 * p1.Y + 3 * p3.Y) / 6.0;
   b[3] = (p1.Y + 4 * p2.Y + p3.Y) / 6.0;
   Point startPoint = new Point();
   startPoint.X = a[3];
   startPoint.Y = b[3];
   spline.Add(startPoint);

   int i;
   for (i = 1; i <= divisions - 1; i++)
   {
      float t = System.Convert.ToSingle(i) /
      System.Convert.ToSingle(divisions);

      Point sPoint = new Point();
      sPoint.X = (a[2] + t * (a[1] + t * a[0])) * t + a[3];
      sPoint.Y = (b[2] + t * (b[1] + t * b[0])) * t + b[3];
      spline.Add(sPoint);
   }
   return spline;
}
```

Figure 8: C# Code Spline Function B

A potential shape has a centre point and the vertices reassemble a closed outline shape around the centre. The centre point defines the location of the shape on a 2-dimensional plane. The representations differ in the way of placing the control points. A function for removing vertices that are too close to each other was added, with a distance smaller than 90% of the minimum boundary parameter. It removes unnecessary aggregation of vertices in one location and to enable the shape to afford sharp edges. The chromosome used for all representations consists of an array of real values (genes) in a range from 0.0 to 1.0, with seven digits of precision, interpreted into coordinate values between a minimum and maximum boundary parameter of 10 and 240 pixels. A parameter defines the length of the chromosome used within the representation that allows increasing or decreasing the detail of the solution. Using a longer chromosome creates more control points. The following sections explain the representations' differences.

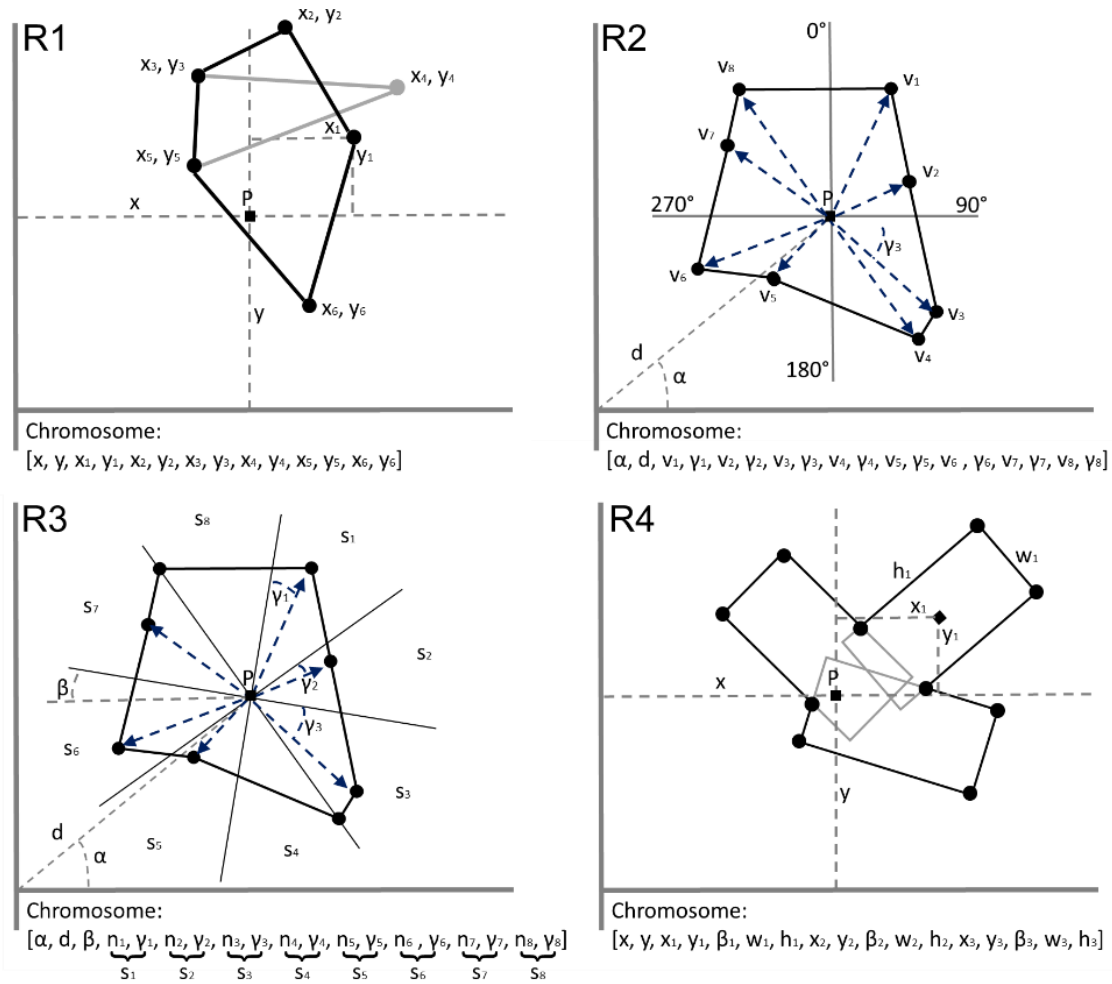


Figure 9: Representations

Representation R1 - Cartesian Coordinate-based. This representation shows a direct mapping of the chromosome's real values to coordinates of control points on a cartesian coordinate system. A minimum and maximum defined shape size constrain the solution.

A closed shape is formed by connecting the control points in an order they appear in the chromosome. Figure 9 (top, left) shows the representation.

The first two genes represent the coordinate of the centre point of the shape. Additional gene pairs represent the coordinates relative to the centre point. However, connecting each coordinate in an order in which they appear in the chromosome leads to a very high number of intersections between lines. Self-intersecting shapes are not evaluable, so a post-processing step was added to resolve the intersections. Intersecting lines are removed by not using the related control point and closing the shape with the next one. If this leads to new intersections, this control point is also not included, and the shape is closed using the control point after. This process continues until no further intersections appear. The post-processing step avoids producing invalid phenotypes.

The representation was chosen to investigate the simplest mapping method between chromosome and solution and the influence of using a post-processing procedure to resolve intersections.

Representation R2 - Polar Vector-based. This representation maps real values to vectors with a common centre. The representation is shown in Figure 9 (top, right). The chromosome's genes correspond to directions and lengths of vectors on a polar coordinate system. Genes related to directions correlate to angles between 0 and 360 degrees, and those related to lengths correlate to a range between a defined minimum and maximum shape size value. The first two genes define the centre position of the shape. Further gene pairs represent the vector coordinates which are sorted by angle and connected in a clockwise direction to avoid intersections in the outline. The representation was inspired by BoxCar2D (Weber, 2015) where the shape of an abstraction of a car body was described similarly.

Representation R3 - Hub and Spoke-based (Lapok, Lawson, & Paechter, 2017)**.** This representation is also based on a polar coordinate system, similar to the previous method. It is shown in Figure 9 (bottom, left). As previously, real values correlate to vectors with direction and length. The first two genes define the centre of the shape. However, there are some differences. One is that an additional gene defines the tilt angle of the polar coordinate system.

Furthermore, each vector has its angle segment in which it operates. E.g. when using six vectors, each vector has its fixed range between 0 and 60 degrees in which it can operate. Invalid shapes are avoided by connecting the vectors in a clockwise direction.

Representation R4 - Rectangle-based (Lapok, Lawson, & Paechter, 2019)**.** This representation uses multiple rectangles as basic shapes. It is shown in Figure 9 (bottom, right). The first two genes define the centre of the shape. Subsequently, every group of five genes translates to a position coordinate, tilt-angle, width and height of a rectangle. Multiple rectangles are positioned relative to the centre. The rectangles may overlap with each other, so the overall outline is extracted. The edges of the outline are the control points for the spline function.

Lee and Nagao developed a similar representation that uses rectangles as basis shapes (Lee & Nagao, 1995). However, there are some differences. First, their representation does not extract the outline of the intersecting rectangles. Instead, they encourage the evolutionary algorithm to avoid overlapping by giving an additional penalty for it. Second, the representation is not used with additional functions such as the spline function or a procedure to remove close vertices.

R4 can produce multiple not connected shapes on one plane, which is interesting from the perspective of mechanical design. In this way, a component consists of multiple not connected shapes which are moving as one component on one or also on multiple planes. Shapes on the same plane can interact with each other. In theory, this provides a way to model a 3-dimensional system. Section 5.2.1 explains this concept in more depth.

3.3.3 Evolutionary Algorithm

The algorithm initially creates a population of random individuals. Each contains a chromosome. The chromosome used in this work consists of an array of real values (genes) in a range from 0.0 to 1.0. The chromosomes are mapped to shapes using the different representations. The algorithm improves the quality of the population's fitness iteratively. Individuals are selected from the population, copied, and mutated in a systematic manner that leads to new individuals (children) of which the fitness is evaluated. Children replace weaker individuals of the population. One individual represents the best solution. The iterative process continues until reaching a stop criterion; in this case, a set number of generations. Figure 10 shows the pseudo-code for the evolutionary algorithm.

```
set POPULATION SIZE
set NUMBER OF CHILDREN per generation
set STOP CRITERION to a number of evaluations
set RECOMBINATION PROPABILITY between 0.0 and 1.0

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 < RECOMBINATION PROPABILITY
            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 copying 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
```

Figure 10: Pseudo Code Evolutionary Algorithm

The algorithm was set up to create a population of 100 individuals and to produce 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 better fitness. There are two evolutionary operators, applied with a probability of 50%. First of them is a mutation with a variable mutation rate. Variable means that the rate changes in each iteration randomly and applies changes to between one to four random selected genes of a selected individual; altering the gene values to random new ones which are determined by a Gaussian Distribution based on the previous value. The Box-Muller transform equation was used (Muller, 1958). The pseudo-code is shown in Figure 11. It returns a new gene value which has a higher probability to be closer to the previous one based on a normal distribution.

```
set SIGMA = 0.2
set MEAN = OLD_GENE_VALUE

# Create two random floating-point numbers U1 and U2 that are greater than or
equal to 0.0, and less than 1.0. However, the first number cannot be 0.

set U1 = 0
while(U1 ==0)
    U1 = Round(1.0 – Random_Double)
    U2 = Round(1.0 – Random_Double)

# Calculate new gene value based on Gaussian distribution (Box-Muller transform)

set Z0 = Round(Sqrt(-2.0 * Log(U1)) * Sin(2.0 * Pi * U2))
set NEW_GENE_VALUE = Z0 * SIGMA + MEAN

*Random_Double return a number between 0.00 and 1.00
*Round is rounding to two decimal places
```

Figure 11: Pseudo Code Box-Muller transform equation

The second operator is a two-point crossover recombination followed by a mutation using the same principle as explained in the previous section. The recombination takes two individuals from the population using the same selection method and exchanges a chromosome segment between them; determining the segment by two random points defining the start and end position. The child contains the segment of the first parent and up to two segments, at the beginning and the end, of the second parent. Children are being added to the population in each generation by replacing selected individuals of the population using a tournament selection. In this case, it picks the weaker of two randomly chosen individuals. This procedure repeats until the maximum number of generations is reached, or the user terminates the process. The algorithm was carefully designed in an iterative manner supported by the visual interface. The interface allowed to evaluate the performance visually and guide the development process of the algorithm and operators.

3.3.4 Fitness Evaluation

The candidate solution's fitness relates to the similarity between a candidate shape and a target shape. In literature, often the symmetrised Hausdorff distance or Euclidean distance is used to calculate the fitness (Chang et al., 2003; Khan & Ray, 2012; Nashvili et al., 2005; P. Zhang et al., 2007). However, this research uses a different approach, not directly dependent on the comparison of coordinates between the solution and target shape. Instead, the fitness function uses the sum of two penalties, based on a comparison of target and solution surface areas. The penalty value decreases when the candidate shape is more similar compared to the target shape. A total penalty value of zero means that the solution and the target shape are identical in form and position. Figure 12 shows the target shape area, the solution shape area, and the intersection between both.

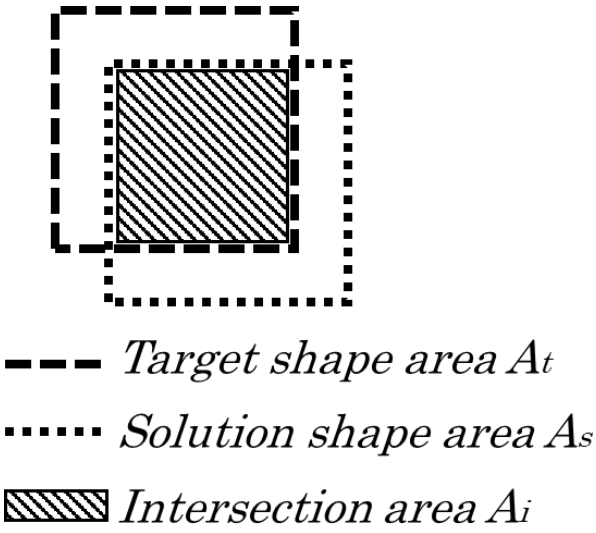


Figure 12: Fitness Evaluation

The following equations show the calculation of the penalties:

$$|At - As| = Ps \quad (1)$$

$$|At - Ai| = Pi \quad (2)$$

$$Ps + 2\,Pi = Pt \quad (3)$$

The first penalty *Ps* results from the difference between the total area size of the target and the area of the solution, shown in Eq.1. *Pi* is the second penalty which results from the difference between the total size of the target and the intersection area with the solution, shown in Eq.2. Eq.3 shows the total penalty *Pt* in which *Pi* has a double weight to avoid a direct competition of the penalties as the size penalty *Ps* and intersection penalties *Pi* may work against each other. The double weight is important to avoid a similar penalty value change when an applied mutation increases the size of the area and at the same time, changes the intersection area. In this case, the penalty values would eliminate each other, and the total penalty would not change. The algorithm would not be able to navigate to a better solution. By doubling the intersection penalty, it receives greater attention, and the algorithm avoids getting trapped in a local optimum.

The areas were extracted using the open-source Clipper C# library (angusj, 2010). The areas for the shapes were calculated using the following C# method shown in Figure 13.

```
/// <summary>
/// Calculate the area of the shape
/// </summary>
/// <param name="shapes">List of shapes where ech shape is a
/// list of coordiantes</param>
/// <returns>Area of all shapes</returns>
private float AreaOfShape(List<List<IntPoint>> shapes)
{
    double result = 0f;
    if (shapes.Count == 0)
        return (float)result;

    foreach (List<IntPoint> shape in shapes)
    {
        shape.Add(shape[0]);
        float area = Math.Abs(shape.Take(shape.Count - 1)
            .Select((p, i) => (shape[i + 1].X - p.X)
            * (shape[i + 1].Y + p.Y)).Sum() / 2);
        result += area;
    }

    return (float)result;
}
```

Figure 13: C# code to calculate the area of a shape

3.3.5 Experiments

Experiments were run 25 times on each of the 24 target shapes for 500,000 evaluations with each representation method. Experiments were conducted using an Intel Core i5-2500 3.3Ghz with 4GB RAM. The target shapes used are shown in Figure 6. Shape *p01* to *p12* are lever shapes extracted from an automotive closure system and shape *p13* to *p24* are general basic shapes.

The algorithm configuration looked as follows: The population was set to 100, and the number of children generated per generation to 20. These values resulted from testing and observations. The number of generations was 25,000, which results in a total number of 500,000 evaluations after which the experiment stopped. The chromosome length for each representation was set to 77 genes to represent one solution to make the comparison fair. In general, a higher number of genes leads to a higher representation quality for all representations as the representations can generate a higher number of control coordinates. Representations which use more genes to create a coordinate may have a disadvantage, e.g. R1 uses two genes for one x-y coordinate, whereas R4 uses 5 genes to place four x-y coordinates (rectangle).

3.4 Results and Evaluation

The results include a comparison of the four representations' performances in producing each target shape. Figure 14 shows an example of evolving *p04* with the representations *R1 – R4*.

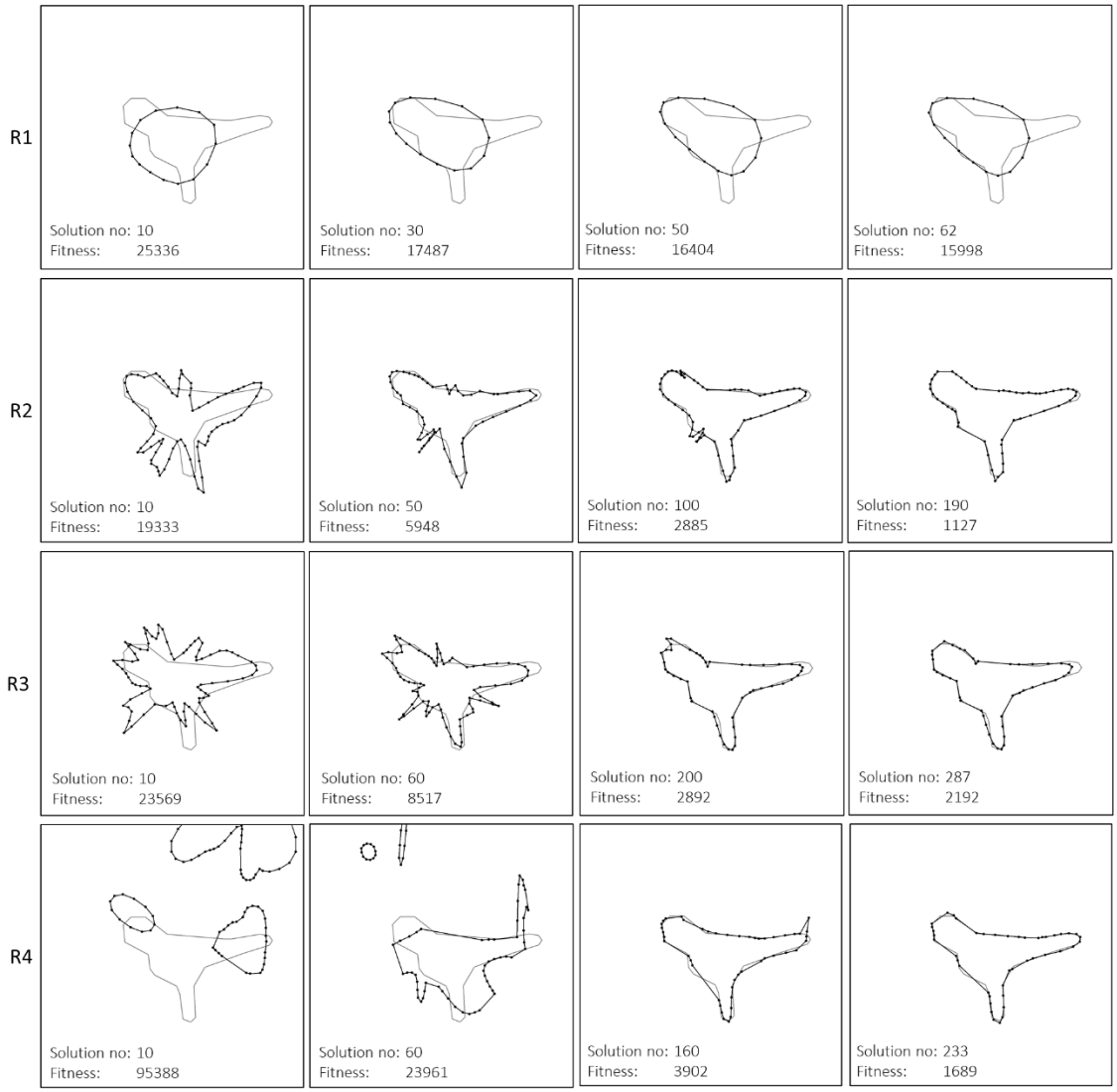


Figure 14: Evolved solutions with R1 - R4

Each row shows a representation and each column a different stage at the evolutionary process. A smaller fitness value means that the evolved shape is more similar to the target shape. The last column shows the final shape after 500,000 evaluations. The solution number increased each time a better solution was found. A higher solution number indicates that the shape evolved in more incremental steps. It is noticeable that in this example, *R1* was not able to represent the target shape efficiently. The other representations performed well. *R2* and *R3* seem to evolve similarly, however, *R3* seems to improve in smaller steps. *R4* starts the evolutionary process with multiple shapes scattered over the canvas but can approximate the target shape in a similar way such as *R2* and *R3*.

The Mann-Whitney U-Test was used for statistical analysis since normality of the distributions cannot be assumed. A *p*-value of $p \leq 0.05$ indicates high confidence that distributions significantly differ. The *p*-value refers to the median distribution of best-performing solutions at the end of each run.

Table 1: Comparison of Methods p-values

comparison		problems (p-values)							
method 1	method 2	p01	p02	p03	p04	p05	p06	p07	p08
R1	R2	3.18E-07	9.02E-08	7.42E-03	1.24E-07	5.85E-07	1.50E-09	6.97E-09	3.39E-08
R1	R3	3.03E-08	1.53E-07	4.76E-03	5.86E-08	4.10E-07	1.50E-09	6.97E-09	2.43E-08
R1	R4	2.71E-08	2.87E-07	5.52E-09	5.25E-08	7.06E-06	4.37E-09	1.89E-07	6.53E-08
R2	R3	1.18E-01	1.38E-01	8.69E-01	2.90E-01	3.99E-01	3.32E-01	5.41E-01	2.05E-05
R2	R4	6.77E-01	7.49E-01	7.82E-09	1.09E-01	3.18E-07	4.85E-01	1.58E-05	1.01E-02
R3	R4	2.99E-01	4.32E-01	1.38E-08	6.00E-01	1.38E-07	1.23E-01	3.74E-05	6.89E-08

comparison		problems (p-values)							
method 1	method 2	p09	p10	p11	p12	p13	p14	p15	p16
R1	R2	1.55E-08	2.43E-08	1.33E-09	4.07E-05	1.12E-07	3.18E-07	4.78E-07	2.24E-05
R1	R3	1.17E-06	4.22E-08	1.33E-09	4.81E-05	6.97E-09	1.00E-07	1.29E-06	7.06E-06
R1	R4	1.91E-09	2.87E-07	2.73E-09	1.70E-07	1.01E-05	2.55E-04	1.58E-05	2.75E-04
R2	R3	1.37E-04	1.66E-02	8.84E-01	8.92E-01	5.87E-03	1.15E-03	3.88E-02	1.18E-01
R2	R4	1.90E-06	6.16E-04	7.49E-01	1.24E-03	1.37E-04	2.32E-03	1.41E-03	3.88E-06
R3	R4	1.12E-07	1.57E-01	4.91E-01	8.17E-04	2.71E-08	1.18E-07	1.01E-01	1.38E-08

comparison		problems (p-values)							
method 1	method 2	p17	p18	p19	p20	p21	p22	p23	p24
R1	R2	1.91E-09	2.58E-07	2.58E-07	1.57E-06	3.21E-08	2.17E-08	2.73E-09	5.54E-01
R1	R3	4.37E-09	1.24E-07	5.85E-07	1.12E-07	3.39E-08	2.57E-09	1.70E-09	5.09E-01
R1	R4	1.89E-07	2.54E-06	4.46E-06	3.07E-06	4.78E-07	5.22E-05	4.92E-09	1.38E-08
R2	R3	2.33E-01	4.38E-01	9.85E-01	4.46E-06	9.15E-01	3.19E-04	4.26E-01	9.77E-01
R2	R4	1.24E-03	3.32E-01	6.97E-02	5.12E-02	2.17E-08	1.75E-02	5.48E-01	1.50E-09
R3	R4	2.63E-02	1.18E-01	1.38E-01	4.46E-02	1.74E-08	1.80E-05	1.97E-01	1.50E-09

Table 1 shows the p-values of comparing two representations to each other for each problem. All comparisons where no significance could be determined are highlighted in grey. It can be seen that the performance of *R1* differs significantly. The other representations show significant differences in performance for some problems, however, for seven cases, no significant differences could be detected.

Figure 15 shows the comparison of the representations *R1 – R4* next to each other for every target shape, with the mean penalty of all 25 runs and the confidence interval. The supplementing tables for the boxplots can be found in Appendix 1.

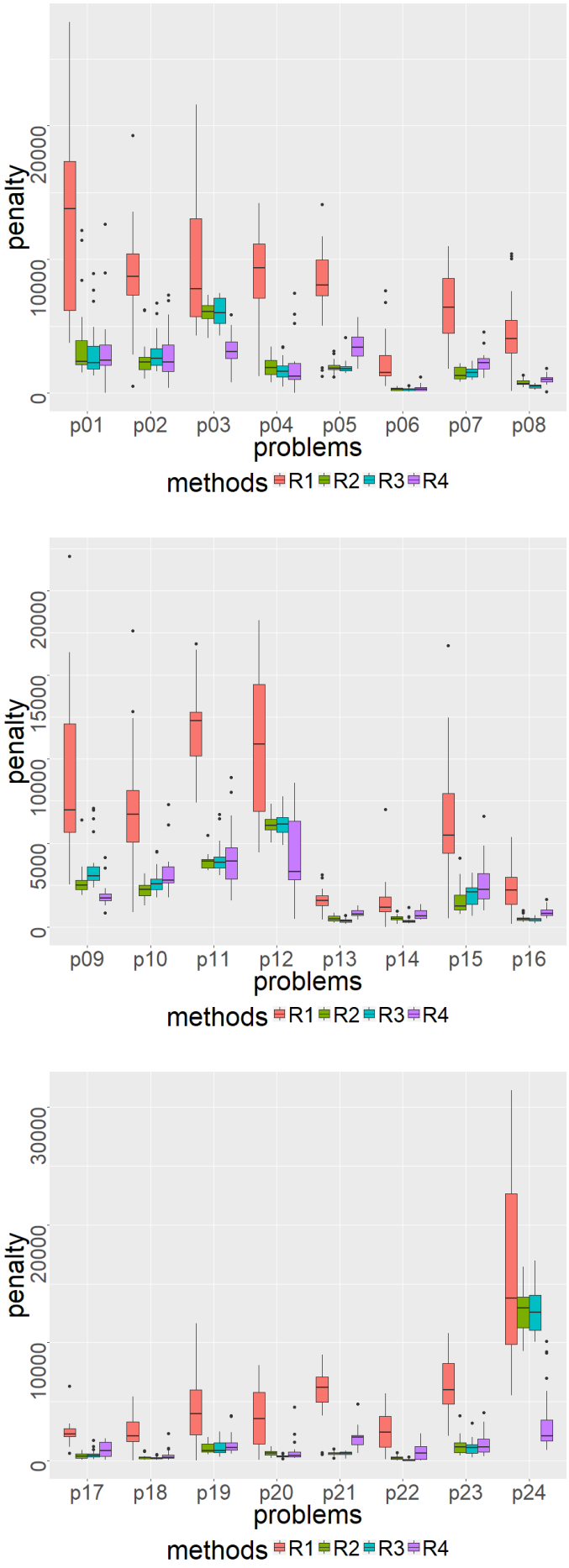


Figure 15: Method Comparison

Vargha-Delaney A-measure (VDA) (Vargha & Delaney, 2017) is used to identify which Representation outperforms another. VDA is a statistical test to evaluate differences (effect size) 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 2 provides the A-measure for the comparison of all representations on every problem. The comparison focuses on medium and large differences between the representations. The arrows show if the first method was better (arrow up), or 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 row shows which representation performed best on the particular problem.

Table 2. Comparison of Methods using VDA.

comparison		problems (A-measure)							
method 1	method 2	p01	p02	p03	p04	p05	p06	p07	p08
R1	R2	↓ **0.922**	↓ **0.941**	↘ 0.7208	↓ **0.936**	↓ **0.912**	↓ **0.998**	↓ **0.978**	↓ **0.955**
R1	R3	↓ **0.957**	↓ **0.933**	↓ 0.7328	↓ **0.947**	↓ **0.918**	↓ **0.998**	↓ **0.978**	↓ **0.96**
R1	R4	↓ **0.958**	↓ **0.923**	↓ **0.981**	↓ **0.949**	↓ 0.8704	↓ **0.984**	↓ **0.93**	↓ **0.946**
R2	R3	→ 0.6288	↗ 0.3776	→ 0.4864	→ 0.5872	→ 0.5696	→ 0.58	→ 0.4496	↓ 0.8512
R2	R4	→ 0.5344	→ 0.4736	↓ **0.976**	→ 0.632	↑ **0.078**	→ 0.4424	↑ 0.144	↑ 0.288
R3	R4	→ 0.4144	→ 0.5648	↓ **0.968**	→ 0.5432	↑ **0.066**	↗ 0.3728	↑ 0.16	↑ **0.055**
best performance		R2,R3,R4	R2	R4	R2,R3,R4	R2,R3	R3	R2,R3	R3

comparison		problems (A-measure)							
method 1	method 2	p09	p10	p11	p12	p13	p14	p15	p16
R1	R2	↓ **0.966**	↓ **0.96**	↓ **1**	↓ 0.8384	↓ **0.938**	↓ **0.922**	↓ **0.915**	↓ 0.8496
R1	R3	↓ 0.9008	↓ **0.952**	↓ **1**	↓ 0.8352	↓ **0.978**	↓ **0.939**	↓ 0.8992	↓ 0.8704
R1	R4	↓ **0.995**	↓ **0.923**	↓ **0.99**	↓ **0.931**	↓ 0.864	↓ 0.8016	↓ 0.856	↓ 0.8
R2	R3	↑ 0.1856	↑ 0.3024	→ 0.488	→ 0.5112	↓ 0.7272	↓ 0.768	↗ 0.3296	→ 0.6288
R2	R4	↓ 0.8928	↑ 0.2176	→ 0.5264	↓ 0.7664	↑ 0.1856	↑ 0.2488	↑ 0.2368	↑ 0.1192
R3	R4	↓ **0.938**	↑ 0.3832	→ 0.5568	↓ 0.776	↑ **0.042**	↑ **0.063**	↗ 0.3648	↑ **0.032**
best performance		R4	R2	R2,R3,R4	R4	R3	R3	R2	R3

comparison		problems (A-measure)							
method 1	method 2	p17	p18	p19	p20	p21	p22	p23	p24
R1	R2	↓ **0.995**	↓ **0.925**	↓ **0.925**	↓ 0.896	↓ **0.956**	↓ **0.962**	↓ **0.99**	→ 0.5488
R1	R3	↓ **0.984**	↓ **0.936**	↓ **0.912**	↓ **0.938**	↓ **0.955**	↓ **0.991**	↓ **0.997**	→ 0.5544
R1	R4	↓ **0.93**	↓ 0.888	↓ 0.8784	↓ 0.8848	↓ **0.915**	↓ 0.8336	↓ **0.982**	↓ **0.968**
R2	R3	→ 0.4016	→ 0.564	→ 0.4984	↓ 0.8784	→ 0.4912	↓ 0.7968	→ 0.5656	→ 0.5024
R2	R4	↑ 0.2336	→ 0.42	↗ 0.3504	↘ 0.6608	↑ **0.038**	↑ 0.304	→ 0.4504	↓ **0.998**
R3	R4	↗ 0.3168	↗ 0.3712	↗ 0.3776	↗ 0.3344	↑ **0.035**	↑ 0.1464	→ 0.3936	↓ **0.998**
best performance		R2,R3	R3	R2,R3	R3	R2,R3	R3	R2,R3,R4	R4

Results show that *R1* has the worst performance on every problem compared to the other representations. The findings show that using a post-processing procedure which disables coordinates to resolve intersections consequently disables parts of the chromosome. It makes the shape optimisation inefficient due to applying mutations to segments of the chromosome that do not influence the solution. Furthermore, disabling coordinates also lowers the method's ability to represent complex shapes as fewer coordinates are left to represent it.

Another finding is that *R2* and *R3* perform mostly equally, with *R3* being the best performing solution more often. Both methods are having a similar basis and similar range of operation, which may cause the outcome. *R2* has 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 of large changes in the chromosome, leading to smaller changes in the solution compared to *R2*. The optimisation algorithm is better guided by incremental improvements, making *R3* obtain a better solution quality. Both methods seem to be slightly better than *R4*, which eight times performed better than other methods; however, the difference is very small.

Furthermore, *R4* produces better solutions to problem *p03*, *p09*, *p12*, and *p24*. The shapes in these cases have an undercut characteristic which cannot be produced by the other methods. Nevertheless, *R4's* performance on *p05* and *p21* is still good, although worse when compared to *R2* and *R3*, due to *R4's* rectangle base that makes it difficult to represent spikes and fine details.

Figure 16 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.

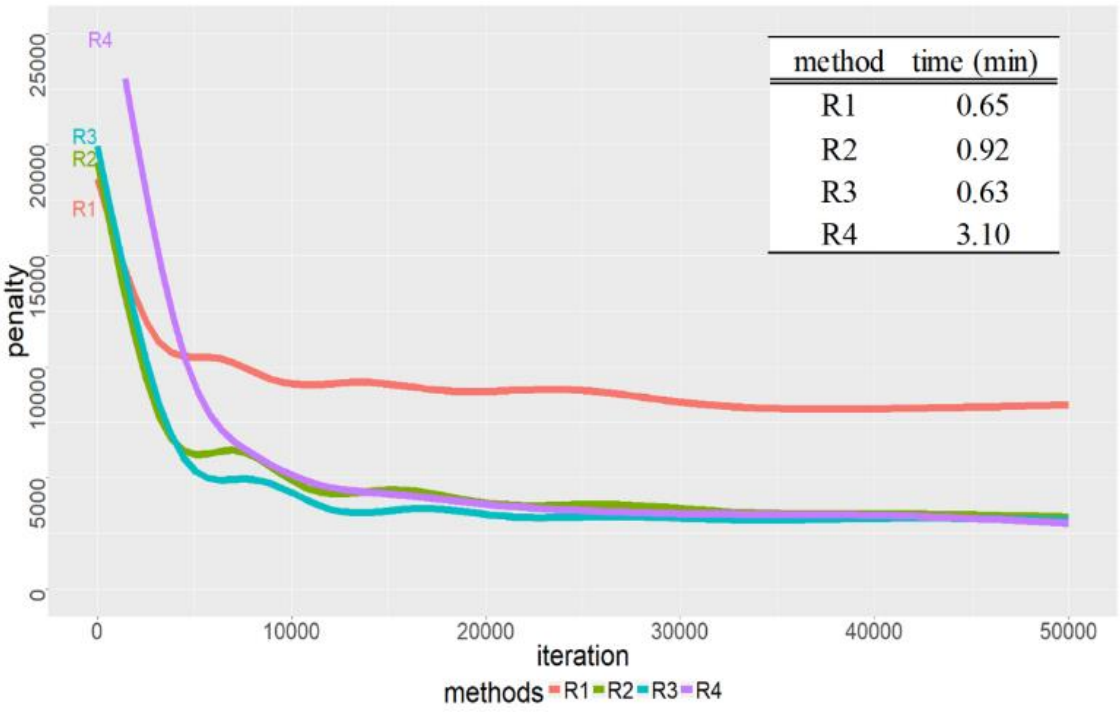


Figure 16: Improvement Over Time

If 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 faster in improving the solution quality compared to all other methods. Figure 16 also shows the time needed to perform 500,000 iterations with each method. *R1*, *R2*, and *R3* need between 0.63 and 0.92 minutes, whereas *R4* needs around 3.1 minutes. *R4* is requiring more processing for calculating the outline of the intersecting rectangles. However, the time for one iteration was far below one millisecond with all representations. Taking into account that the representation's purpose at a later stage is to generate mechanical systems, where the evaluation of a solution requires a physics simulation which is taking a longer time than the calculations of the mapping procedure from genotype to phenotype.

3.5 Summary

This chapter focused on the evaluation of shape representations for mechanical components used in combination with an evolutionary algorithm. It is an initial step towards a generative design system for mechanism design. Representations were developed iteratively by using the method of evolving target shapes, instead of direct evaluation in a physics simulator. It allowed investigating the representation's capabilities and limitations in-depth, employing a helpful tool to identify flaws, improve, and compare representations performance. The requirements for a shape representation were defined, and the method used for development, testing, and evaluation of shape

representations was presented. Furthermore, explaining that the potential shape representation needs to fulfil criteria to be well-performing, including keeping the number of descriptive parameters as low as possible; covering the entire relevant search space; avoid producing invalid solutions, and be compatible with genetic operators.

Four representations were developed and evaluated. Their performance was investigated 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*) used direct mapping of genes to coordinates and a post-processing procedure to resolve shape intersections. The second representation (*R2*) mapped the chromosome to vectors which were connected in a clockwise direction to avoid intersections. The third representation (*R3*) mapped the chromosome to vectors as well; however, it allowed each vector to operate in a specified area. In the fourth representation (*R4*), the chromosome was mapped to multiple overlapping rectangles of which the overall outline was extracted. Two functions were applied to the resulting shape of the four methods. The first function applied a spline function to the shape, introducing curves and the second removed vertices too close to each other, avoiding aggregation of vertices.

Several experiments were undertaken to evaluate the performance of each method to produce the target shapes. The performance was compared and statistically evaluated 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 was not able to evolve complex shapes. *R2* and *R3* performed almost equally in terms of solution quality, with *R3* performing slightly better and needing fewer iterations to reach a superior result. *R4* was slower than the other representations; however, it could produce similar results to *R2* and *R3* in many cases. *R4* was the only representation capable of producing shapes with undercut features, an example is shown in Figure 17, to a high quality which can be considered as being more complex shapes.

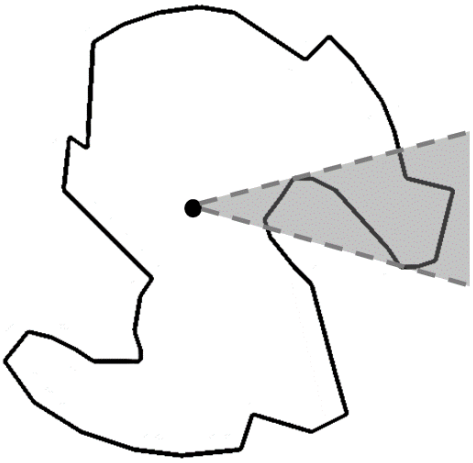


Figure 17: Undercut feature

Findings showed that *R4* covers a large search space using only 77 descriptive parameters. All representations were designed to produce exclusively valid solutions, and the evolutionary algorithm was able to navigate the search space with all representations. Therefore, *R4* was chosen to be extended and used for evolving mechanisms. In the following chapter, the representation is developed further and embed it in a physics simulator evolving shaped components.

4 Evaluation Method for Evolutionary Design using a Physics Simulator

4.1 Introduction

In the previous chapter, different shape representations were evaluated in their ability to create mechanical component shapes guided by an evolutionary algorithm. After showing that the representations are capable of evolving mechanical shapes, the next step is to embed them into a physics environment, which will allow investigating their ability to create shapes capable of adapting to surrounding obstacles and satisfying functional objectives.

The previously employed fitness function focused on target shapes. This chapter presents a method to evaluate the performance of potential solutions within a physics scenario. For that purpose, a simulator and visualisation tool was developed, allowing to specify design aims, design problems, and visualise the movement of components. The simulator resolves the movement of physics components according to a scenario.

Box2D, a two-dimensional game physics engine (Catto, n.d.), was chosen as the backbone for the simulator. Box2D can resolve movements and collisions between rigid bodies in a virtual world, including forces, torque, friction, restitution, mass, and gravity. It provides a way to define a virtual world of rigid bodies and is capable of defining parameters around material properties and masses. The physics engine returns position and orientation of all components in the virtual world on a frame by frame basis. It simulates seconds-long scenarios, within a few milliseconds, depending on its complexity, in contrast to conventional simulations used in mechanical design, as they tend to focus on precision, rather than speed. Conventional methods often compute details such as elastic deformation which is not necessary at early design stages. Although the implemented simulator is less accurate than others employed within the industry, it is accurate enough to be used for resolving the motion in the less detailed, early conceptual design phase, when focusing on shape and placement of components. It is well suited to evaluate a large set of potential solutions at a very fast pace.

For this research, the simulator is embedded in a generative system, using the evolutionary algorithm of the previous chapter to evolve the shape of a component. The approach allows producing candidate solutions, and evaluate them, as well as adapting a component to its environment with the objective to traverse a set of landscapes. Different

scenarios can be defined with the same design objective, enabling a performance evaluation of the evolutionary algorithm.

The simulation environment and its capabilities are detailed below, together with a validation of the simulator and the generative abilities of the method.

4.2 Background

This section gives insight into the simulation capabilities required for planar mechanical systems. It discusses different simulation approaches, explains the physics parameters, and provides the details of the generative application.

4.2.1 Requirements for Physics Simulator

A physics simulator employed for this research is concerned with qualities of mechanical systems; in particular, it allows resolving movements and collisions between components of these systems. Other approaches which use simulation in similar context focus on kinematic properties, as explained in the literature review. Kinematic simulation resolves motion; however, it omits mass, friction, gravity or collisions between components.

Often larger mechanical systems are broken down into subsystems such as cam and follower mechanisms, to reduce the problem complexity (J. Lampinen, 2003; Mundo et al., 2006; Ruan et al., 2015). The optimisation within kinematic simulators is limited to subsystems with components that do not lose contact with each other, which makes the calculation of the behaviour easier. In real-life mechanisms, some sections of components are only occasionally in contact with other components or the environment.

There is currently no known simulator available tailored for planar mechanism prototypes that can be used in combination with an evolutionary algorithm to conduct experiments. Evolutionary computation requires hundreds or thousands of evaluations to be performed, which necessitates a physics simulator to be fast in resolving the kinematic behaviour of candidate solutions. The simulator should be able to compute multiple components simultaneously without the requirement of breaking systems down into smaller subsystems.

The simulator should also have the ability to resolve scenarios in which components are not constantly in contact. The length of the simulation should be specifiable. The output format should contain the locomotion, components movement, of the scenario for each simulation frame, as each of them embeds the position and orientation of the involved components at a specific time. The selected frames can be used to replay a visualisation of the scenario to review solutions; this, in turn, may be used to evaluate the performance

of a solution. Furthermore, the simulator needs to include parameters, such mass; friction; and restitution, to be able to reassemble a real-world mechanism, as well as drive implementation and gravity which introduce input forces and movement to the scenario.

4.2.2 Physics Parameters

Relevant physics parameters considered in this research include gravity, as well as material parameters such as mass, friction, and restitution that influence a mechanical system's kinematic behaviour.

Every component in the system has a mass, influencing the inertia, and the force needed to move it. Mass can be specified for each component directly by definition, or results of the specification of the material density. In the latter case, the mass increases with the growth of the surface area of the component.

Friction specifies the amount of resistance force between the contact surfaces of two components when sliding against each other, while restitution represents the energy loss in a collision between two components. The latter is a material attribute which simulates the bouncing behaviour of colliding components.

Gravity influences all dynamic components. It is a constant force and influences the overall behaviour of a mechanical system specified by one parameter. Components on a 2-dimensional layer accelerate in the direction of the defined gravitational direction. However, in cases when looking at the mechanical system from a birds-eye perspective, the gravity can be turned off by setting it to zero.

Also, all listed parameters can be part of the optimisation. In this work, though, the aim is to optimise the shape and configuration rather than the material choice, which is why these parameters are constant values in the problem description.

Throughout experiments, the material density was set to 1.0 grams per cubic centimetre, the restitution coefficient to 0.6, and the friction coefficient to 0.5 which should represent a hard plastic.

4.2.3 Application for Experiments

Research to the date shown that there is no known suitable application with the functionality required to conduct relevant experiments. For that reason, a generative tool was designed and implemented, uniting the evolutionary algorithm and physics engine, and providing a graphical user interface to conduct experiments and visualise them. Figure 18 shows the application.

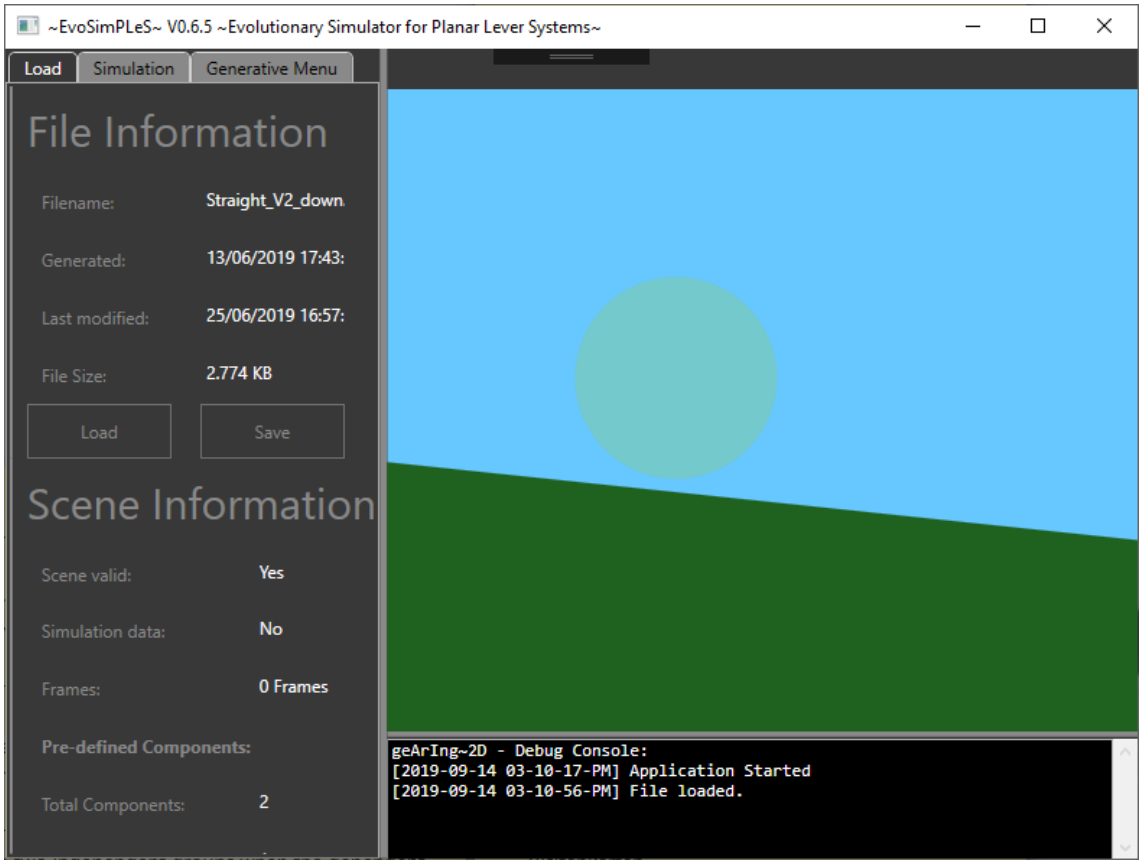


Figure 18: Generative Tool - Load Menu

The interface consists of three sections. The user controls are on the left-hand side, the simulation view on the top-right, and a feedback console on the bottom-right.

The user controls provide three tabs, namely load; simulation; and generative menu. The load tab has two buttons, one for importing files containing a design scene, and the second for saving the session, e.g. with a design solution. After loading a file, the view shows attributes of the file and the problem, e.g. several components within the scene. The file contains the framework to define mechanism design, explained in detail in Chapter 5.

The simulation tab opens a menu connected to the physics simulator, shown in Figure 19.

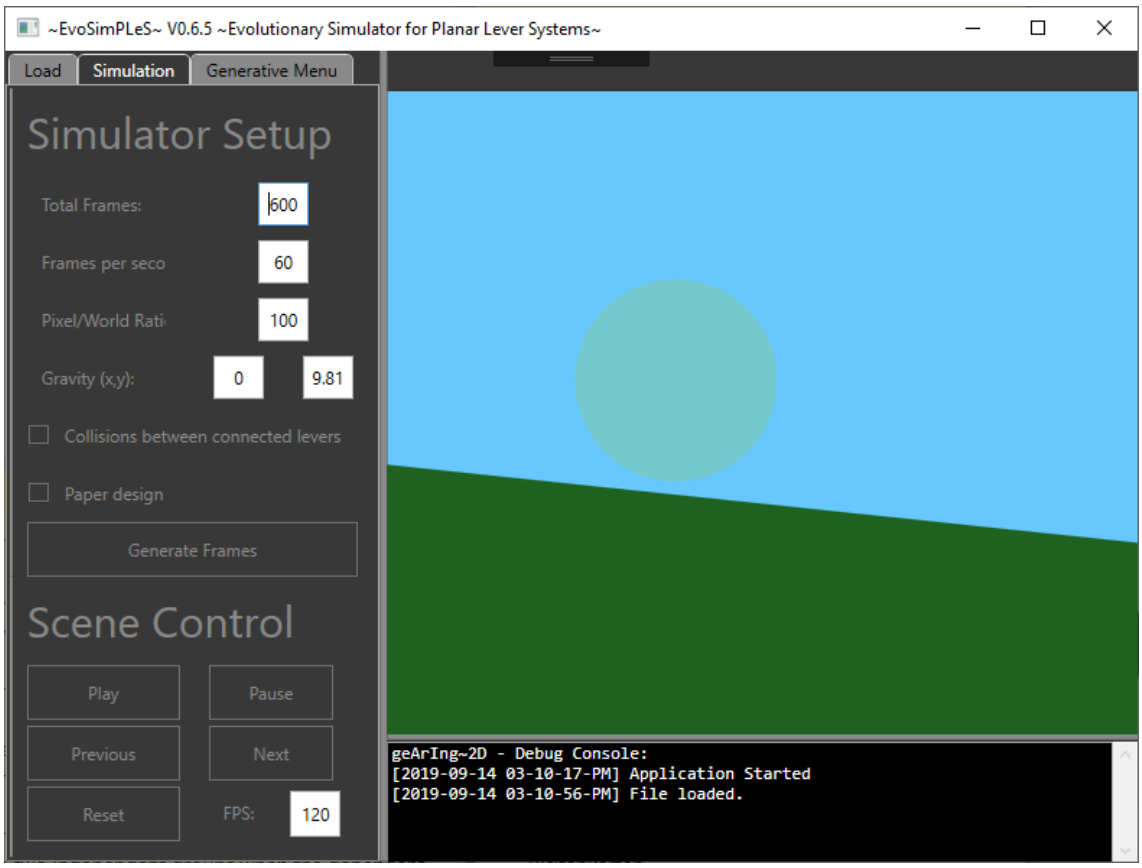


Figure 19: Generative Tool - Simulation Menu

The simulation menu enables setting the length of the simulation according to the number of frames and the frame rate. Dividing the total frames by the frames per second results in the total simulation time in seconds. The gravity in meter per square second is settable in x and y-direction.

A checkbox provides the option specifying whether connected components should collide with each other or not; another changes the colour setting of the simulation. The "Generate Frames" button starts the simulation of the current scene, which can be subsequently visualised, paused, or skipped frame by frame using the related buttons. Furthermore, a button allows resetting the simulation to the first frame. The visualisation speed is adjustable be defining it in frames per second.

Figure 20 shows the generative menu.

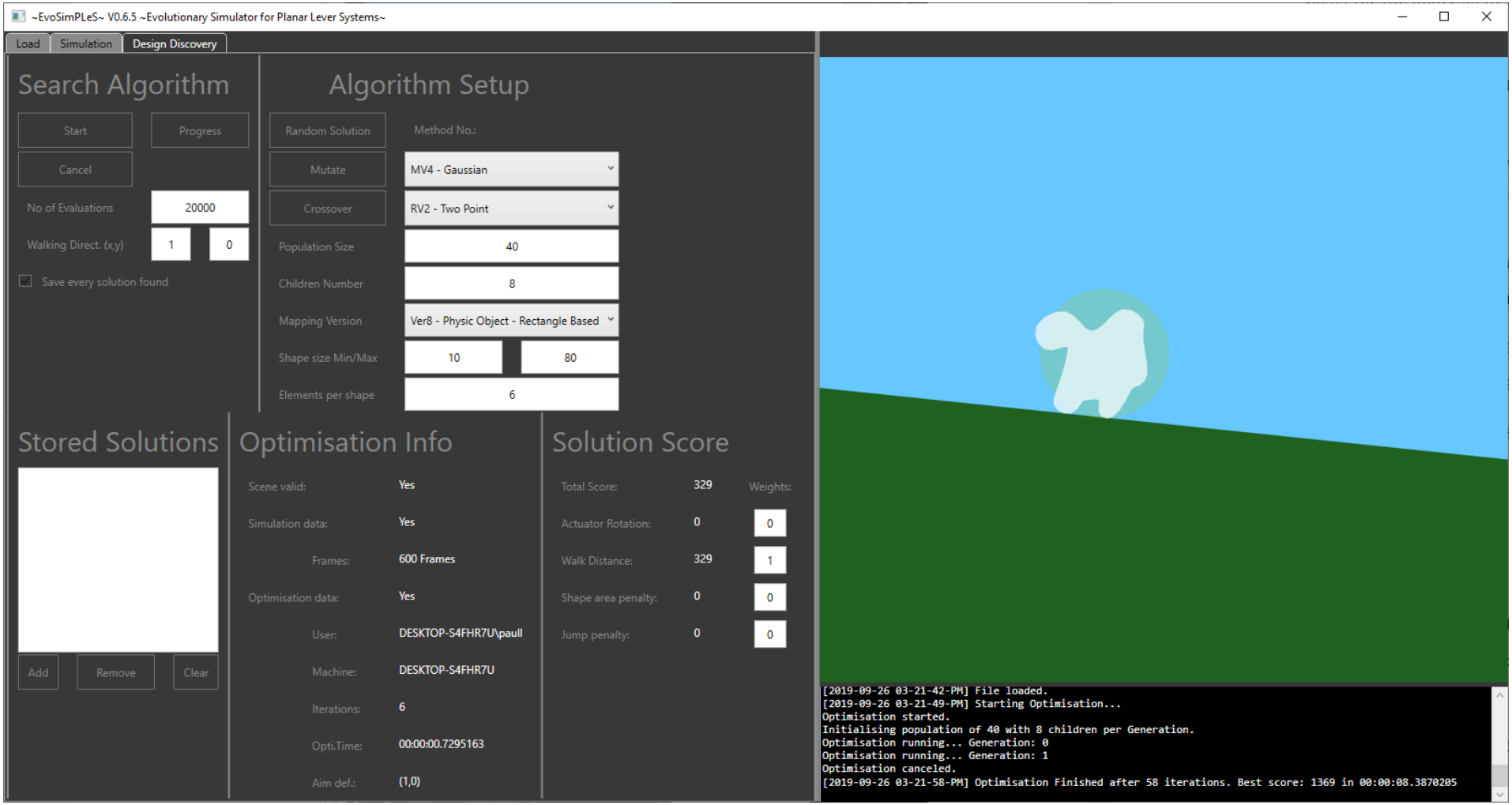


Figure 20: Generative Tool - Generative Menu

The menu is connected to the generative system and contains the control elements for configuration. The evolutionary algorithm can be started and stopped, and the current progress can be manually requested. The number of evaluations defines the stop criterion. The design objective is to let a solution traverse a physics landscape, explained in detail throughout this chapter. The walking objective can be set in x and y-direction, in which the solution should move.

A checkbox can be enabled to produce output data. It creates a folder on the desktop and a solution file each time the algorithm finds a result with better fitness. The data is the basis for the analysis of the performance of the algorithm.

The “Algorithm Setup” section contains the configuration of the algorithm. Different implemented evolutionary operators, such as mutation and recombination, can be selected from a drop-down menu. The population size and number of children per generation can be defined. Furthermore, a drop-down menu provides a way of choosing a representation. This part of the menu allows testing representations of similar or different mechanisms, e.g. to produce single or multiple components, or linkages, further explained throughout this and the following chapters. Representation related thresholds can be defined, such as minimum and maximum size of a solution and number of shape elements utilised.

Additionally, the generative menu has a section with a list box containing all found solutions. These can be selected and visualised.

An “Optimisation Info” section provides feedback on the generative process. A “Solution Score” section includes the option to set a weight for several fitness values given to different properties of the solution, such as the rotation of the actuator, walking distance, shape area, and a jump penalty. In this work, the focus is mainly on walking distance fitness.

The simulation view on the right-hand side shows the physics environment as previously explained. The user can scroll through the scene and zoom in and out. The console at the bottom gives feedback on the simulation, the generative process, and occurring errors.

The simulator was implemented in C# using the WPF framework. The code architecture separates the view from the logic. The software can be extended to accommodate new algorithms, operators, representations, and design objectives.

4.3 Method

A simulator was implemented, providing a visualisation of the locomotion and behaviour of a physics scenario. It delivers the data for the fitness function of the evolutionary algorithm, which evaluates the change of the configuration of the scenario throughout a specified timespan.

The evolutionary algorithm used here is similar to the one in the previous chapter, except applied in a new context including the physics simulator as well as extended forms of representations. The evolutionary algorithm's capability is investigated to evolve shape components able to traverse physics environments. A set of scenarios was designed containing descendant landscapes with different profiles (which will be introduced in Figure 26) to test the evolutionary algorithm's performance.

For the experiments, a shape component is placed on the top of the descendant landscape. From there, gravity pulls it to the ground and makes it roll down. The profile of the landscape is designed to hinder the component's ability to roll. The objective function evaluates how far and quick the component moves down the landscape, which allows the evolutionary algorithm to evolve solutions capable of overcoming this obstacle. In result, it changes the shape of the component and tries to evolve the best suitable shape for the landscape.

This chapter focuses on testing the following:

- Firstly, the function of the simulator using unit and acceptance tests.
- Secondly, the suitability of the simulator to be employed in an evolutionary computing application, by evolving components whose fitness is dependent on their shape and interaction with a physics scenario, to fulfil the design goal.

4.3.1 Functionality Testing

Throughout the development, the simulator was tested using unit tests. Their purpose is to validate whether every unit of the software performs as intended. A unit is the smallest testable part of the software, usually called a method. A method takes input variables and provides an output. Unit tests are other test methods implemented for each method of the software. They include all input scenarios and feed the software with defined values, and compares the output to the desired output. If the output is different from the desired output, then the test fails. The generative system's code, including simulator, is covered by unit tests where possible. The unit-tests are embedded in the code.

Furthermore, acceptance tests are used to validate the correct implementation of the simulator. Different scenarios are defined to examine the simulator visually, e.g. whether the placement of the components is correct; the computation of collisions is reasonable, and the parameters, such as mass, friction, restitution, and gravity, are correctly applied. The acceptance tests, including all tested scenarios, can be found in Appendix 2.

4.3.2 Evolutionary Representation

The previous chapter provided an evaluation of four representations based on different principles. The rectangle-based representation explained in section 3.3.2 (Representation R4 - Rectangle-based) performed well and was taken further to be used for evolving component shapes within a physics environment, investigating its ability to evolve solutions for different landscapes which fulfil a design goal.

However, throughout initial experiments, it was found that using the representation without any changes produced many scattered shapes, which led to extending the representation and creating two further versions of it. These included minor adjustments. Their performance was compared in experiments.

The first representation *R* is similar to the one in section 3.3.2, served as a baseline in this chapter. *R** is the second representation with a modified distance constraint that defines the displacement of rectangles from the centre point of the component. The rectangles have a higher probability of overlapping. *R*** is the third representation, in this case, based on *R**, broaden with an additional gene per rectangle which enables or disables it. *R*** has a higher probability to solve a problem with a simpler shape assembled with fewer rectangles than the other versions. The representation should lower the possibility of getting trapped in a local optimum when some undesired rectangles hinder the shape's ability to be evolved to a better solution.

Figure 21 shows the encoding for *R*.

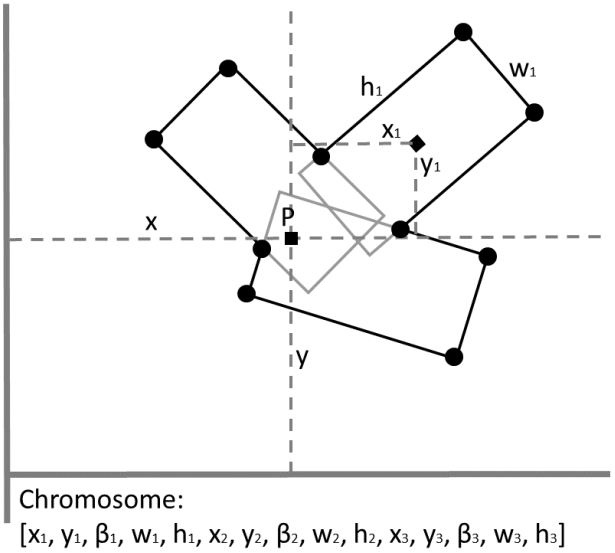


Figure 21: Encoding

As previously explained, *R* employs multiple rectangles to assemble the shape of a component. The component has a centre point. A representation constraint defines the maximum distance of positioning a rectangle away from the centre. A group of five genes describe a rectangle which each gene is describing its x and y offset position, tilt-angle, width, and height. Overlapping rectangles construct an overall outline. The edges of the outline fulfil the role of control points for a spline function. The representation produces multiple shapes if rectangles do not overlap, which still behave as one component and stay the same distance appart when moving.

*R** has the same encoding as *R;* however, the rectangles displacement from the centre point is limited to a smaller maximum distance. Figure 22 shows an example of the maximum distance in which placing a rectangle away from the centre point is possible.

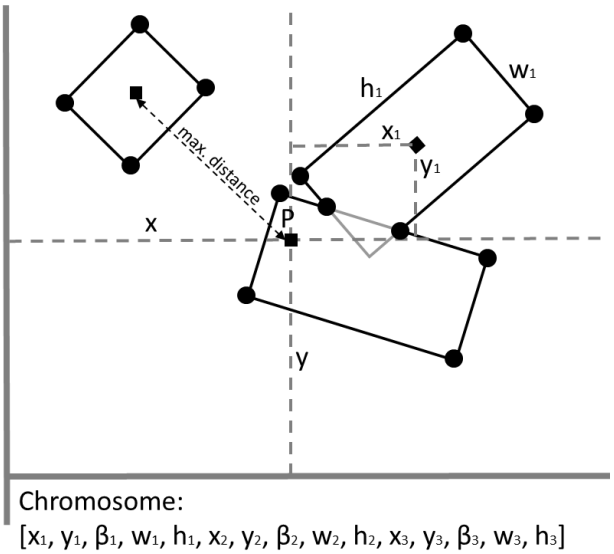


Figure 22: Maximum Distance from the Origin

R has a maximum distance of 80 pixel which is twice the distance of *R**.

Representation *R*** has an encoding modification, including additional genes such as shown in Figure 23.

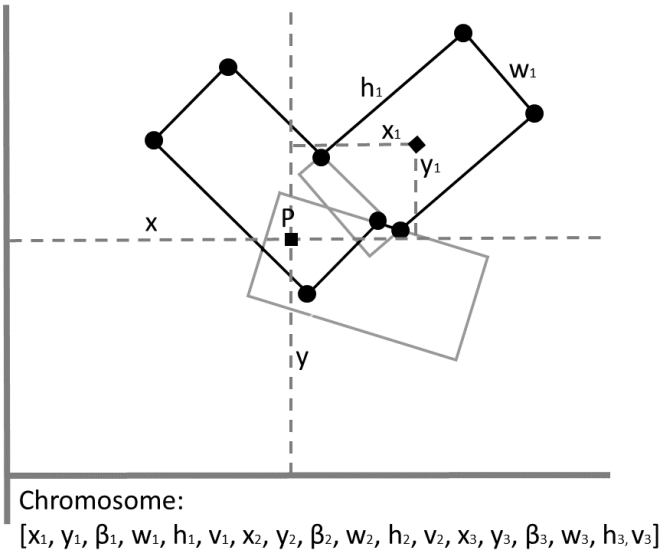


Figure 23: Turning-off Rectangles

For *R***, one additional gene (*v*) per rectangle enables the algorithm to blend out individual rectangles. It works in a binary way. Due to using real value genes, the algorithm enables a rectangle if the number is mutated to an even value and disables it when the number is odd. The representation has an advantage in producing simpler shapes, e.g. it can produce ellipsis and circles by disabling rectangles. The other representations would need to position additional rectangles inside of one rectangle to generate an ellipsis or circle which requires specific gene configurations on the genotype.

4.3.3 Evolutionary Algorithm

For the experiments, the evolutionary algorithm explained in Chapter 3 is used with minor changes, although with a different configuration and including different representations. A population size of 40 individuals and eight children per generation was defined, a similar ratio as the one used previously. The smaller population size was found to perform better throughout initial testing. All other algorithm properties were identical.

For this chapter, two different mutation operators and two different recombination operators were implemented and compared. The first mutation operator *M1* changes chromosome values to new ones, with a deviation based on a Gaussian Distribution. This means that new values are more likely to be closer to the old ones. The second mutation operator *M2* changes values randomly. The first recombination operator is a one-point

crossover operator *R1* with a random crossover point. Figure 24 shows an example of a one-point crossover operation.

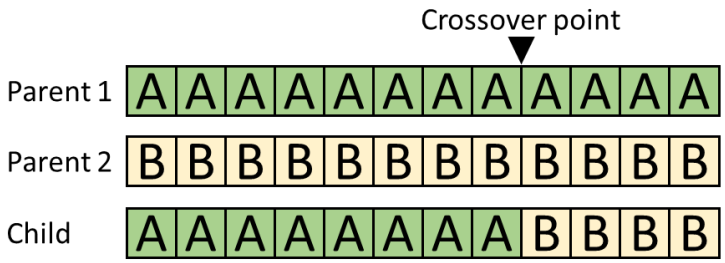


Figure 24: One-point Crossover R1

The crossover *R1* takes one part of the first parent's chromosome and the second part of the second parent's chromosome to create a child chromosome. The second recombination operator, *R2*, is a two-point crossover operator, which uses a random start and random end crossover point, shown in Figure 25.

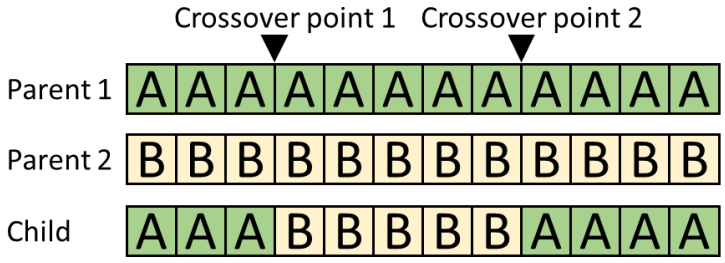


Figure 25: Two-point Crossover R2

The crossover operator *R2* is similar to the one used in Chapter 3. It exchanges a section between the start point and an endpoint of parent one's chromosome, with parent two's chromosome, to create a child chromosome. Experiments were conducted to evaluate the operators. Section 4.4 shows the results.

The gene sequence is essential to preserve solution features. E.g. a uniform crossover which exchanges multiple values instead of a sequence between parents does not conserve the solution's features and would introduce too large disruption.

4.3.4 Fitness Evaluation

The fitness of a potential solution is determined through measuring the travel distance of the shape component on a descending landscape. The objective is to maximise the position of the evolved component in the x-direction at the end of a specified timespan. The shape component is pulled by gravity and rolls down the landscape while encountering obstacles that require the shape to adapt accordingly to the landscape, to not get stuck on its path down.

4.3.5 Experiments

The experiments tested the evolutionary algorithm's ability to adapt a shape component in a physics environment to a provided landscape. A candidate shape is mapped into a physics component and placed at the top of a descending surface path. The component is pulled down by gravity with the objective to roll down the landscape as far and fast as possible within a specified timespan. The evolutionary algorithm evolves the shape to the topology of the landscape required to fulfil the objective. The idea is that if the evolutionary system is capable of adapting a shape to a ground surface, then it is also capable of adapting a shape within a mechanical context to its surrounding components in the environment, which is important for evolving mechanical components, described in the next chapter. Five different landscapes with different topology and difficulty were designed to test the algorithm, shown in Figure 26.

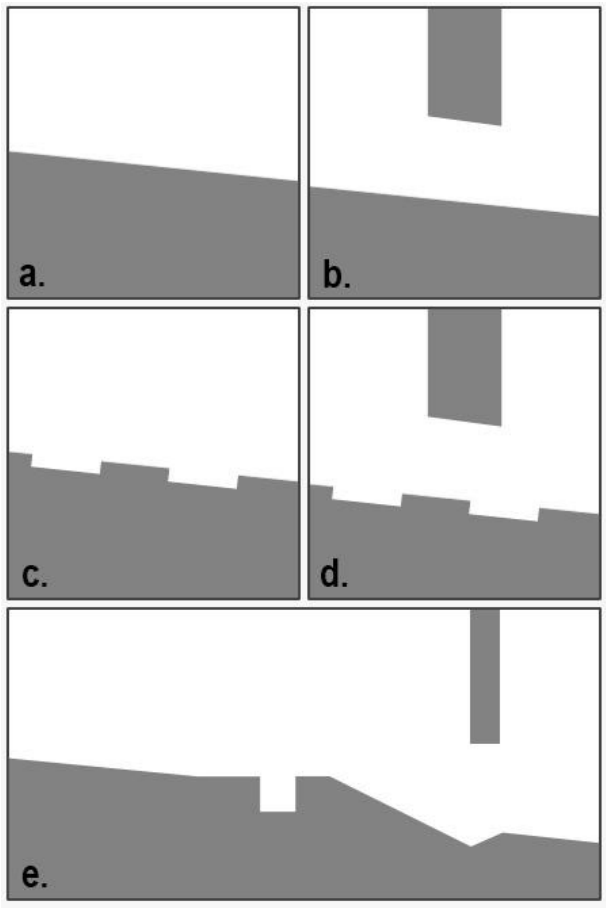


Figure 26: Simulation Landscapes

Landscape a is a simple straight descending path; *landscape b* is a straight path as well, except it includes wall obstacles which are repeating over the length of the path which adds complexity by limiting the height of the descending path. *Landscape c* is a more complex digital shaped path; *landscape d* is a digital shaped path as well, except with a repeating height limit. *Landscape e* is an irregular path with the highest complexity due

to introducing multiple obstacles, such as a hole, a height limit, and a small rising section. The dimensions of the landscape relate to an evolved shaped as follows: The *landscapes a-d* show a section of an estimated 400 x 400 pixels and *landscape e* 400 x 800 pixels. The maximum size of an evolved component can reach is around 240 x 240 pixel.

Each of the three representations was tested using these landscapes with two different mutations and recombination preferences, and compared to random sampling taken as a baseline. Random sampling means that in each iteration new random chromosome values were assigned. The experiments were run 24 times on each descending landscape and stopped after 20,000 evaluations. It is a lower number of iterations than used in the previous chapter, as the evaluation is computationally more expensive. This leads to 30 to 40 hours runtime for one experiment with 24 repeats on one landscape, using an Intel Core i5-2500 3.3Ghz with 4GB RAM.

4.4 Results and Evaluation

Experiments were run to investigate the generative system's capabilities to evolve component shapes that efficiently traverse a provided descending path, pulled by gravity.

- Firstly, the simulator was validated.
- Secondly, the mutation and recombination operators were evaluated to identify their effect on evolving solutions using one version of the rectangle representation (*R**) on one problem instance (*landscape a*).
- Thirdly, the evolutionary algorithm was evaluated in its ability to evolve solutions for one problem instance (*landscape a*) using the three versions of the rectangle representation by comparing it to random sampling.
- Fourthly, two different mutation (*M1* and *M2*) and recombination-settings (*R1* and *R2*) were evaluated on four different problem instances (*landscape a-d*) using the three versions of the rectangle representation.
- Then, the generative system's performance was evaluated using an environment (*landscape e*) with enhanced complexity, including irregularities utilising the three versions of the rectangle representation with the two different mutation and recombination settings.

The Mann-Whitney U-Test was used for statistical analysis since normality of the distributions cannot be assumed. A *p*-value of $p \leq 0.05$ indicates high confidence that distributions significantly differ. The *p*-value refers to the median distribution of best-performing solutions at the end of each run.

The key findings can be summarised as follows:

Evaluation of evolutionary operators:

- *M1,* which uses a Gaussian Distribution, performs significantly better than *M2*
- *R1* and *R2* show no significant difference in performance

Evaluation of a generative system's ability to evolve solutions:

- *R* performs on average similar to random sampling, however with a wider confidence interval
- *R** and *R*** perform better than random sampling
- *R** performing better than *R***

Evaluation of evolutionary settings (including environment (*landscape e*) with enhanced complexity):

Table 3 summarises the results for each landscape. It shows the best performing evolutionary setting for each representation and environment. In some cases, both settings perform equally. The best performing representation for each environment is presented as well.

Table 3. Evaluation of evolutionary settings.

Representation	Landscape				
	a	b	c	d	e
R	S2	S2	S1/S2	S1	S1/S2
R*	S1	S1	S1/S2	S1/S2	S1/S2
R**	S1/S2	S1/S2	S1/S2	S1/S2	S1
Best performance	R	R**	R, R*	R	R*

- R performs better when bigger changes are applied (*S2*), however, it has a higher potential to get trapped in local optima
- R* is stable throughout all problem instances and evolutionary settings
- R** appears to perform well with simple problems and is biased towards producing simpler shapes

The findings are discussed in detail in the following sections.

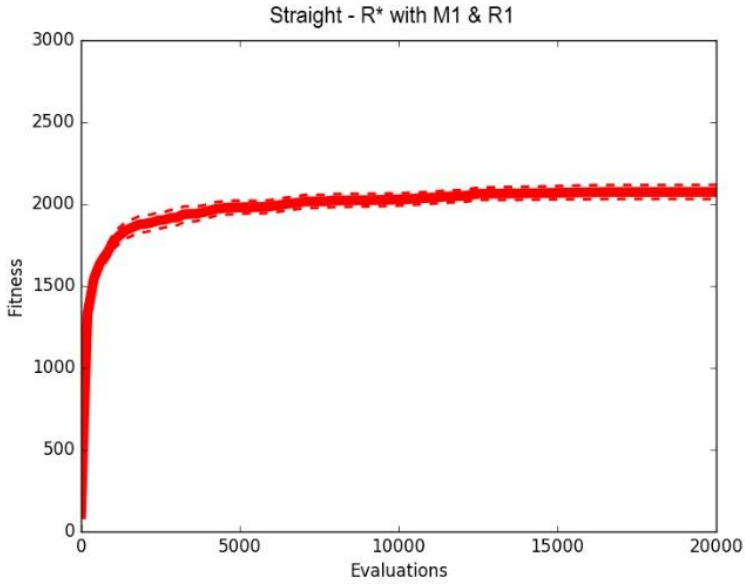
4.4.1 Simulator Validation

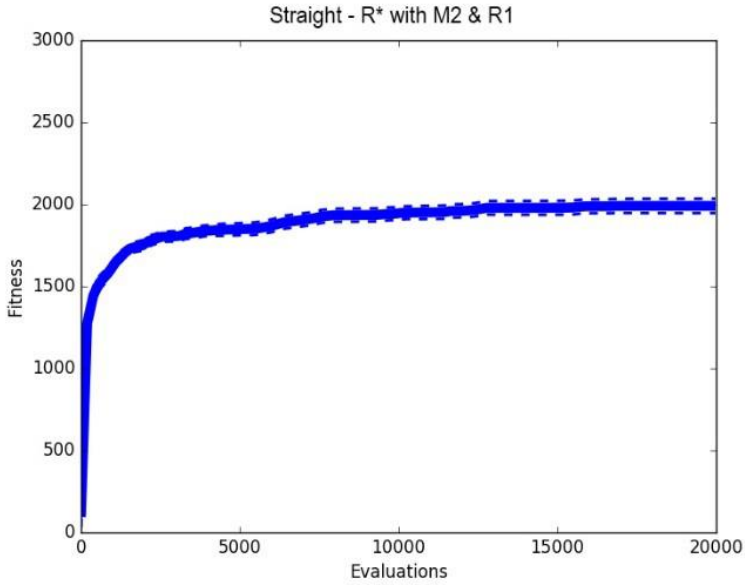
The simulator passed all unit-tests related to the code implementation. Acceptance tests were conducted, including collision tests of shape components such as circles, polygons, and mixed shape types on one and multiple layers (further explained in section 5.2.1). Tests of the parameter setting, namely gravity, density, friction, and restitution were

conducted. The revolution joint was tested as a single joint and within a linkage. Also, the actuator function was investigated. The simulator passed all tests. It performed well when choosing a frame rate of over 60 frames per second (fps). Lower frame rates produced simulation errors in some cases due to clipping, namely penetrating components, especially when faster movements were involved. Choosing higher frame rates improved the simulation quality; however, it resulted in longer simulation time.

4.4.2 Evaluation of Evolutionary Operators

In this section, the different mutation and recombination operators are evaluated to identify their effect on evolving solutions using representation *R** on *landscape a*. A perfect circle reaches a score value of 3350 in this landscape. Figure 27 shows three sets of results with different evolutionary operator settings.





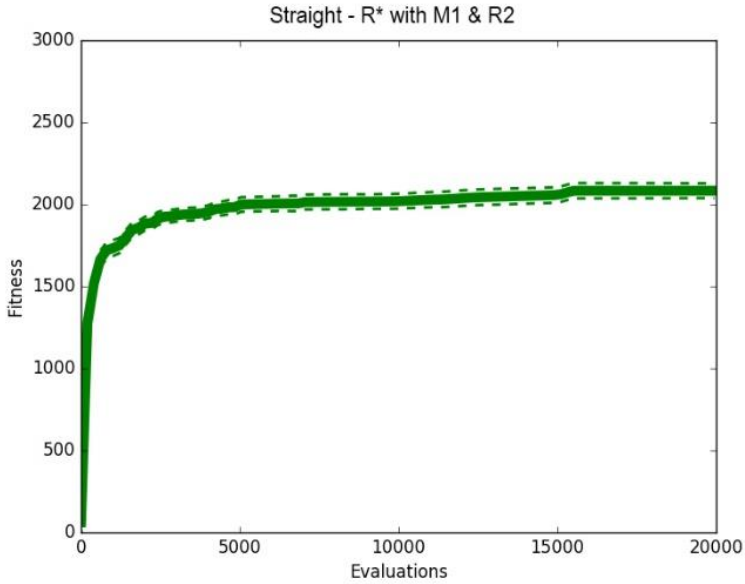


Figure 27: Comparison of Evolutionary Operators

The plots show the fitness of the best solution after every evaluation. For the first plot (**red**) in the figure above, the evolutionary algorithm uses operators *M1* and *R1*, which were explained previously in section 4.3.3. For the second plot (**blue**), the operators *M2* and *R1* are used; and the third plot (**green**) shows the performance achieved with operators *M1* and *R2*. The plots allow comparing the performance of the individual operators.

Analysis of the first and second plot shows a difference in the performance of the mutation operator. It demonstrates that *M1*, which uses a Gaussian Distribution, performs better than *M2*, which assigns random values to a gene. The difference between both distributions is significant ($p = 0.00319$). In plots one and three, different recombination operators are used; however, there is no significant difference between the distributions ($p = 0.959$), which shows that the recombination operators perform very similarly.

4.4.3 Evaluation of Generative System's Ability to Evolve Solutions

The evolutionary algorithm was evaluated in its ability to evolve solutions for *landscape a*, recording the solution's fitness increase over 20,000 evaluations. The three versions of the rectangle representation were compared to random sampling to investigate whether the evolution is occurring. Furthermore, the representations were compared to each other. Figure 28 shows the performance of the random sampling (**red**) compared to the performance of the representation *R*, *R**, and *R*** (**blue**) including the p-value indicating the confidence in the difference of the populations.

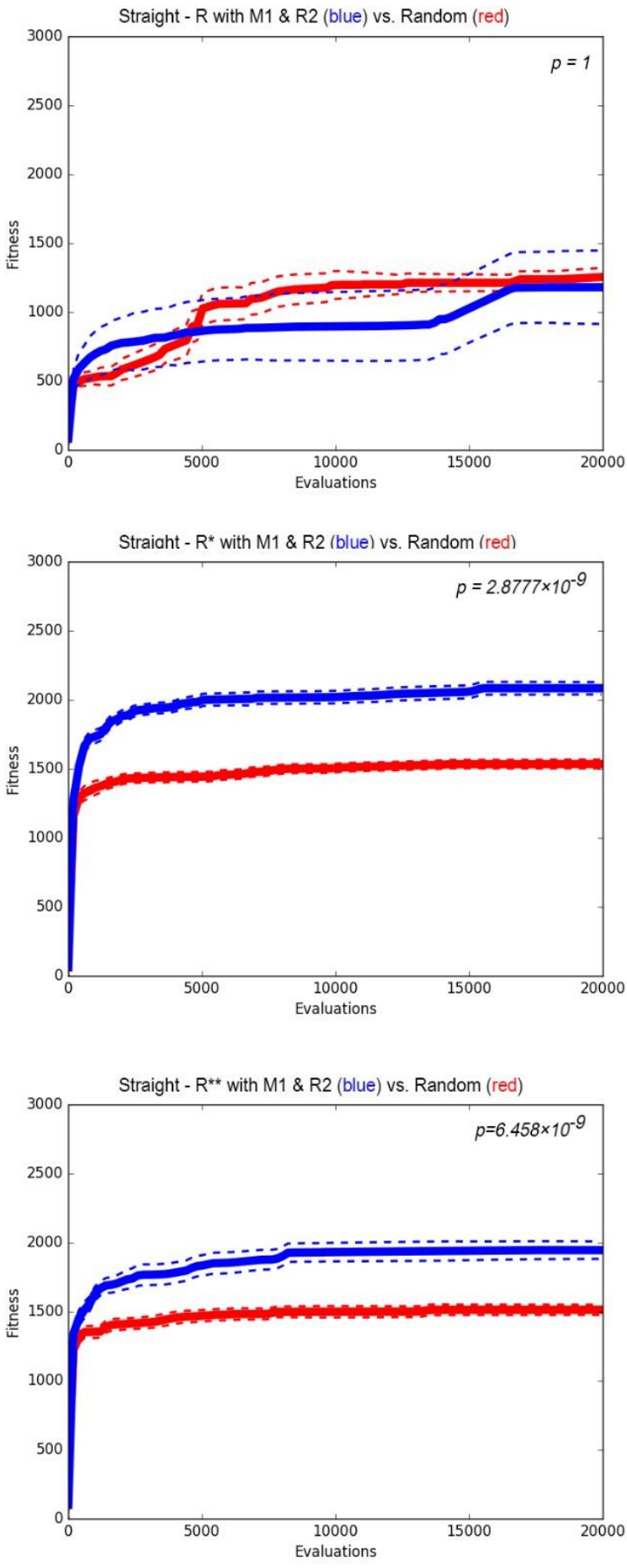


Figure 28: Comparison of Generative System to Random Sampling

The results show that *R* performs on average similar to random sampling. Random sampling reaches a fitness value of around 1250 whereas *R* reaches 1150. However, *R* has a much wider confidence interval than random sampling and may occasionally find better-performing solutions. The wider confidence interval may be caused by a too large disruption introduced when generating new solutions. *R* allows placing rectangles in a larger area, which may lead to too large changes in the phenotype.

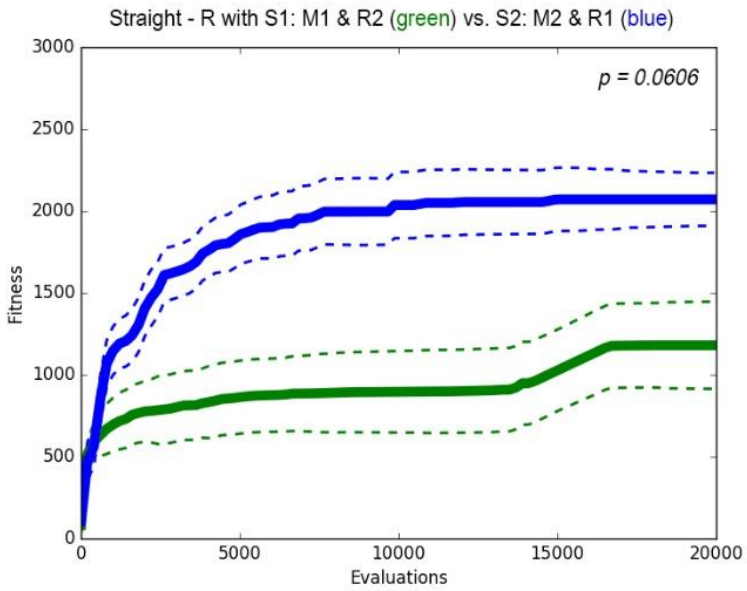
When looking at the results produced by random sampling, compared to representations *R** and *R***, the solutions reach a fitness value of around 1500 with a very similar trend. The evolutionary algorithm produces better results with fitness values of around 2000. Both distributions significantly differ from each other, with *R** performing better than R** ($p = 8.366 \times 10^{-4}$) on *landscape a*. The difference between *R** and *R*** could be attributed to *R*** using a larger chromosome length, which increases the search space and computational effort to traverse it.

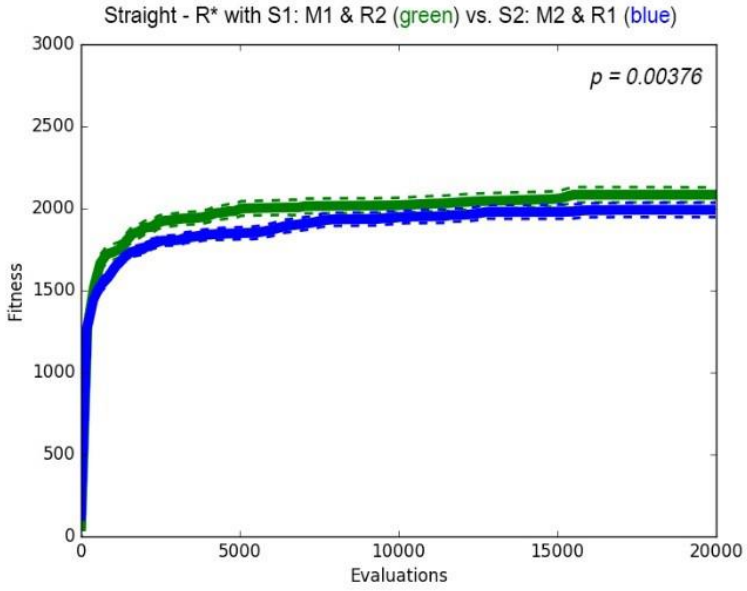
4.4.4 Evaluation of Evolutionary Settings

In this section, the three representations are evaluated on their ability to produce solutions for *landscapes a – d* with two different evolutionary operator settings. The focus is not on the mutation rate, as a variable rate was used that can mutate up to 25% of the chromosome. Instead, the research focused on the actual gene value changes, as using a real value chromosome. The aim is to evaluate the performance of each representation to produce solutions for different landscapes using two different configuration settings and to identify which representation – setting setup performs well.

In the first setting *S1(**green**)*, the mutation operator *M1* based on Gaussian Distribution was employed, applying a smaller gene value change. The operator is used together with the recombination operator *R2*, a two-point crossover that exchanges a sequence of one parent with another parent to create a child solution. In the second setting, *S2 (**blue**)* employs the mutation operator *M2* which changes the gene value randomly, together with recombination operator *R1*, a one-point crossover which exchanges one part of one parent with another parent to create a child. The difference between *S1* and *S2* is that *S1* applies smaller changes compared to *S2* while producing a child. *S2* applies larger changes to the chromosome. The experiment concentrated upon determining how the representations cope with these different evolutionary settings. The performance of the representations and settings was compared for each problem instance individually.

Figure 29 shows the results using the three representations in combination with *S1* and *S2* on *landscape a*.





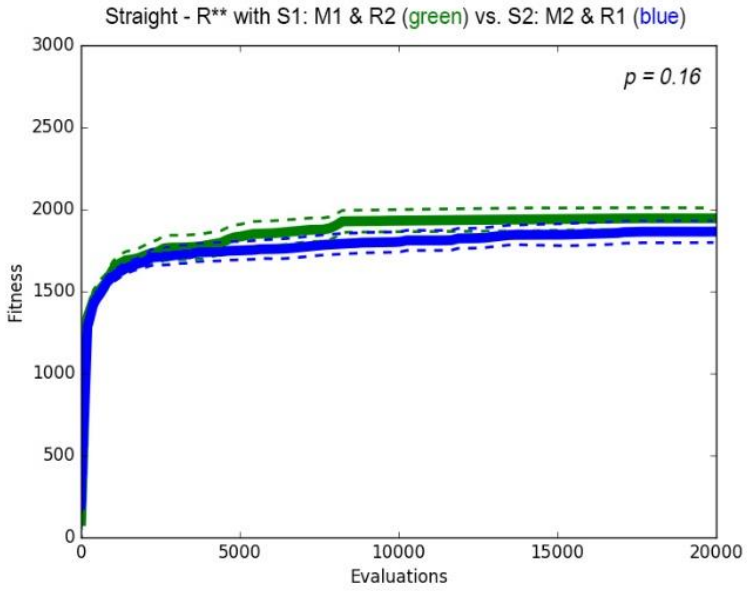


Figure 29: S1 and S2 for R, R* and R** on Landscape a

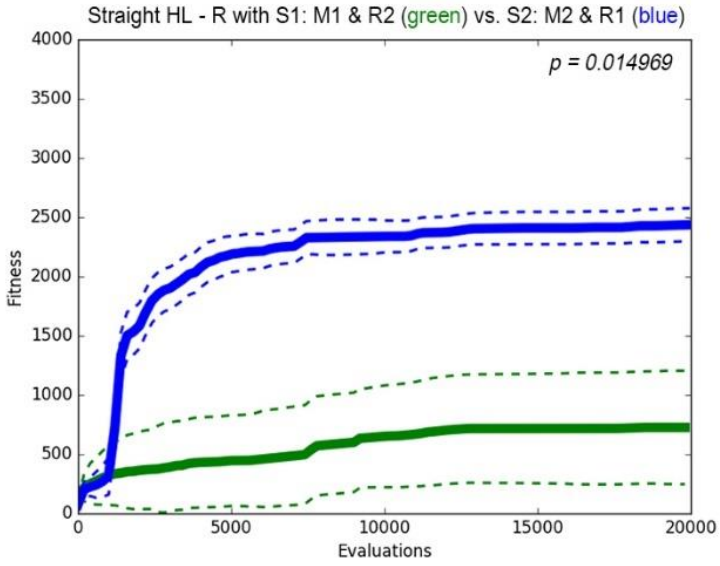
The first plot shows the representation *R* with the two evolutionary operator settings. *R* does not perform well with setting *S1* compared to *S2*. The populations do not show a significant difference ($p = 0.0606$). The confidence interval in both plots is wider compared to the other results. It is an indicator that evolution gets trapped in a local optimum. A larger change using *S2* helps the algorithm to escape it. *R* does not work well with smaller mutations based on Gaussian Distribution and two-point crossover.

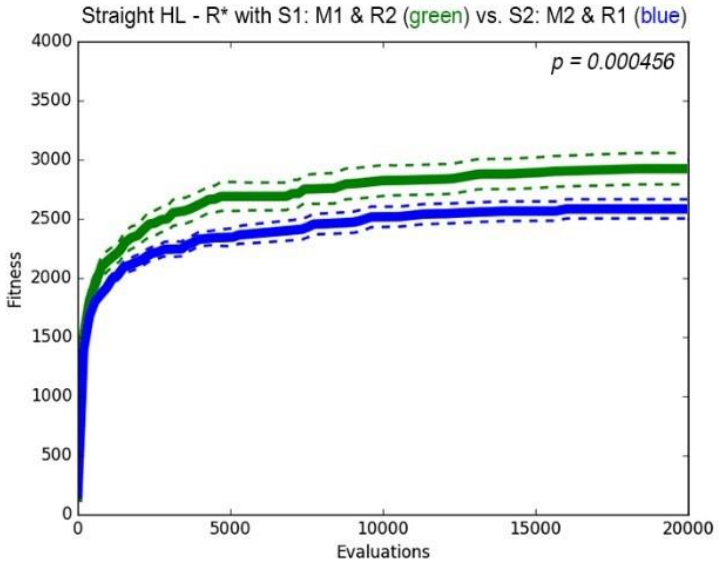
The following two plots for *R** and *R*** show a narrow confidence interval. Looking at *R**, applying smaller changes with setting *S1* increases the performance of the solutions in contrast to larger changes using *S2*. Both populations are significantly different ($p = 0.00376$).

For *R***, the performance comparison between settings *S1* and *S2* shows no difference ($p = 0.16$). There is no difference in applying smaller and larger mutations.

R performs better than *R** ($p = 0.035$) and *R*** ($p = 0.0088$) when applying larger changes using *S2*.

Figure 30 shows the results using the three representations in combination with *S1* and *S2* on *landscape b.*





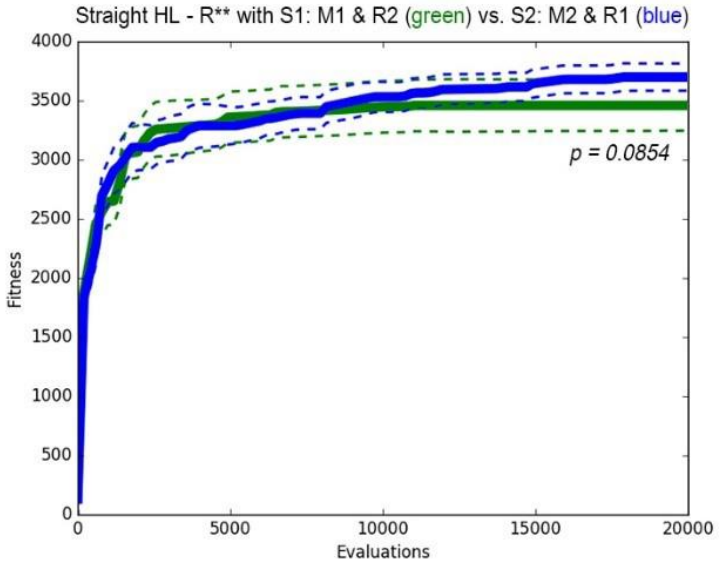
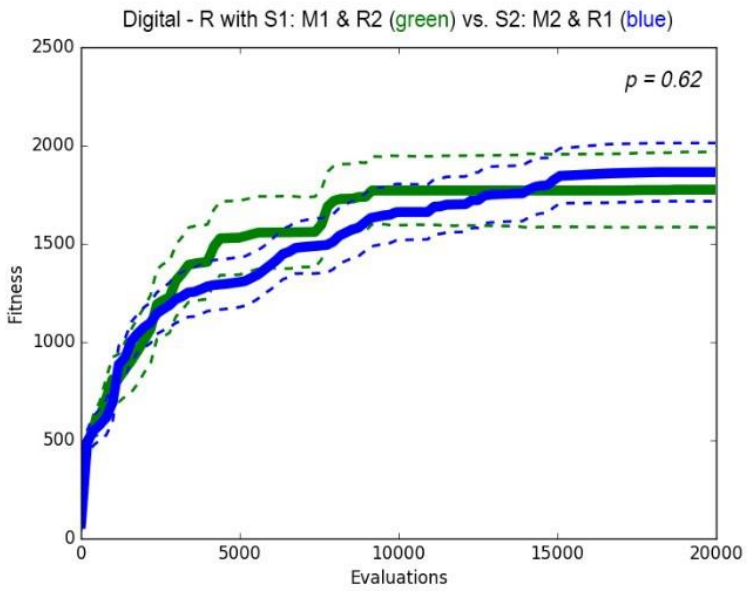
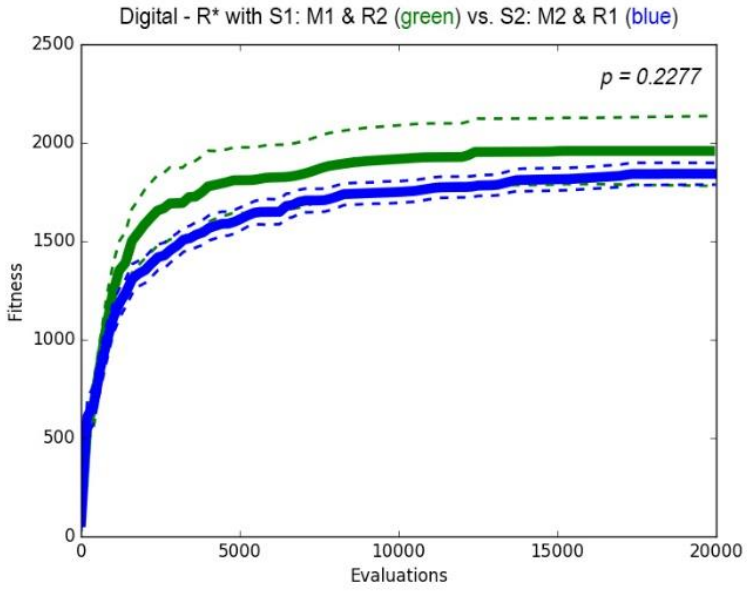


Figure 30: S1 and S2 for R, R* and R** on Landscape b

According to the results above, *R* does not cope well when applying small changes (*S1*) in *landscape b* ($p = 0.014969$). The evolution gets trapped in a local optimum quickly and has a wider confidence interval. *R** performs better with setting *S1* compared to *S2* ($p = 0.000456$). *R*** performs similarly with both settings with no difference ($p = 0.0854$). However, it appears that *R*** is constantly increasing its performance. *R*** with setting *S2* reaches the highest performance value with a difference compared to *R* ($p = 8.41 \times 10^{-9}$) and *R** ($p = 2.85 \times 10^{-9}$).

Figure 31 shows the results using the three representations in combination with *S1* and *S2* on *landscape c.*





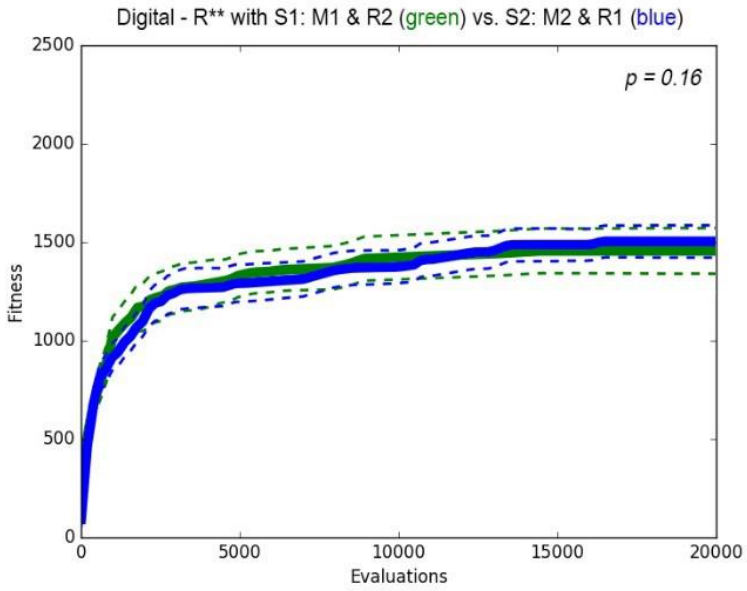
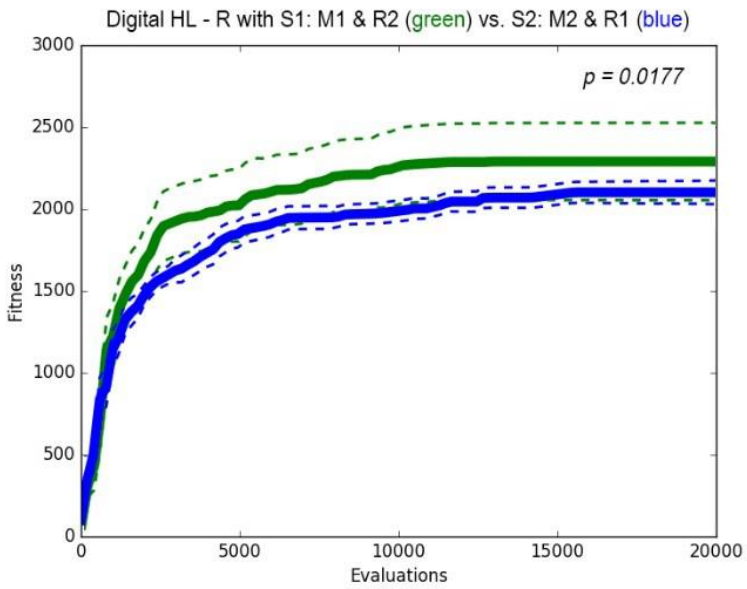
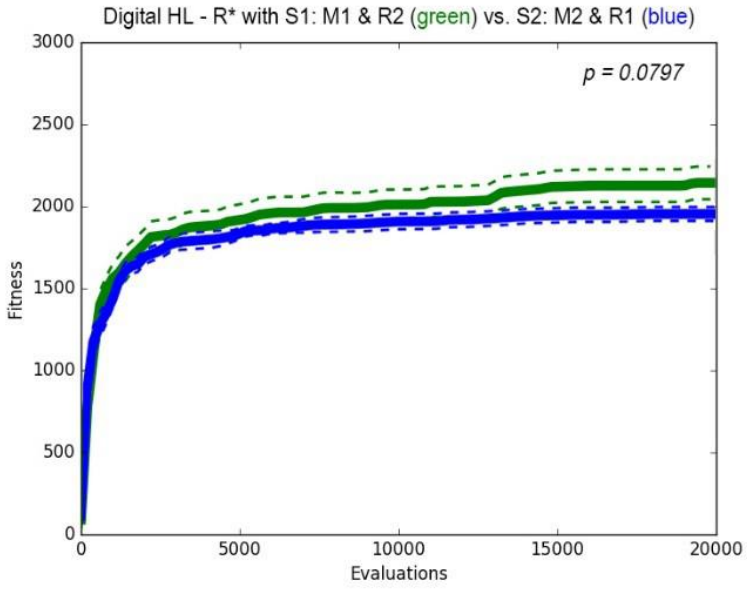


Figure 31: S1 and S2 for R, R* and R** on Landscape c

The settings *S1* and *S2* perform similarly on *R* with no difference ($p = 0.62$) and a wider confidence interval. There is also no difference comparing *S1* and *S2* in *R** ($p = 0.2277$); for *R*** ($p = 0.16$). However, *R* and *R** perform similarly well ($p = 0.82$) while using *S2*, whereas, *R*** performs worse compared to *R* ($p = 0.0006$) with a significant difference. *R*** seems to perform in general worse on *landscape c*.

Figure 32 shows the results using the three representations in combination with *S1* and *S2* on *landscape d.*





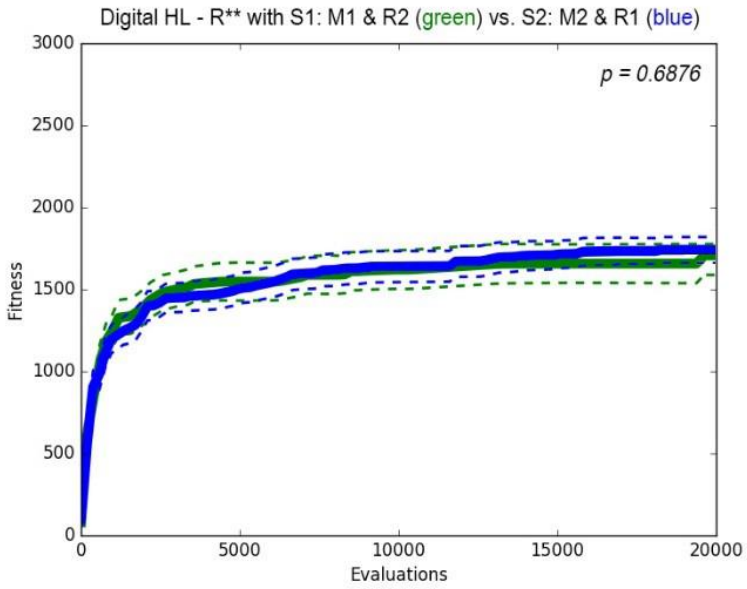


Figure 32: S1 and S2 for R, R* and R** on Landscape d

This figure shows that *R* performs better applying smaller changes with *S1* compared to *S2* ($p = 0.0177$). However, *S1* produces a wider confidence interval which indicates that the algorithm gets trapped in local optimum more often. *R** shows no significant difference comparing *S1* and *S2* ($p = 0.0797$). *R*** demonstrates no significant difference between *S1* and *S2* ($p = 0.6876$) either. While analysing *S2*, *R* is the best performing representation compared to *R** ($p = 0.0016$) and *R*** ($p = 1.4 \times 10^{-6}$).

The performance of the algorithm was investigated with different representations and two different evolutionary settings *S1* and *S2*. *S1* applied smaller changes to the chromosome than *S2*.

Findings show that *R* performs worse when smaller changes are applied, and it has a higher potential to get trapped in local optima compared to the other representations. With it's broader rectangle placement constraint, it has a larger bias towards evolving shape fragments that disturb the movement of the shape component, such as shown in Figure 33

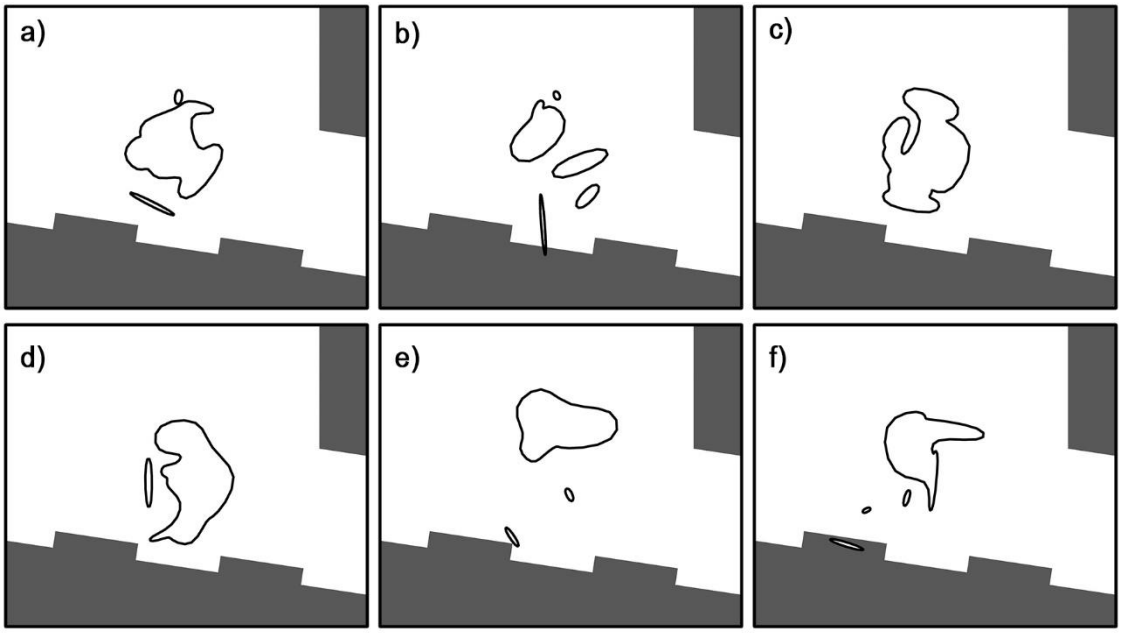


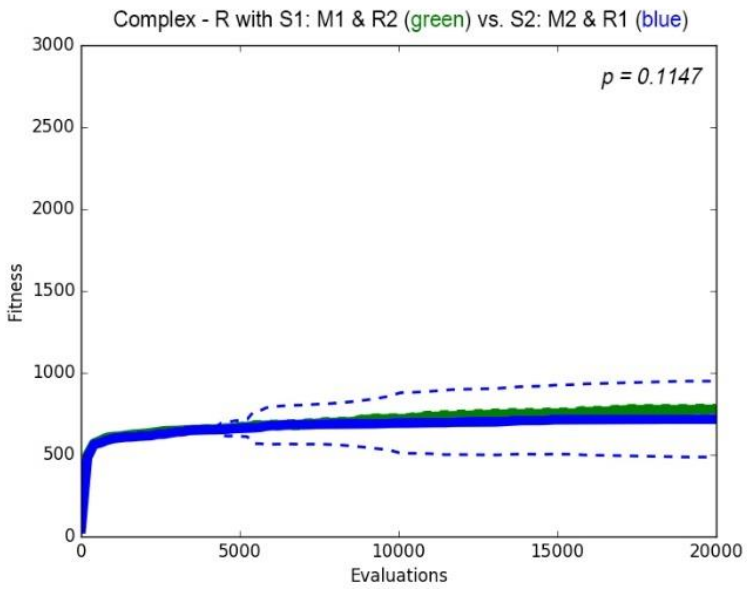
Figure 33: Fragments

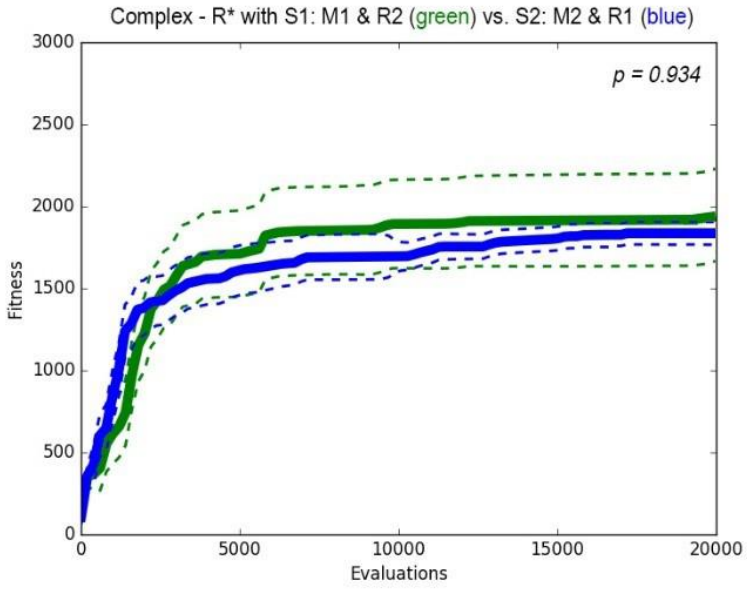
The performance of *R** is stable throughout all problem instances and evolutionary settings. *R*** appears to perform well with simple problems such as *landscape a*, and worse compared to the other representations on more complex landscapes. The reason may be founded in the attributes of *R***, as it can produce simpler shapes by removing rectangles which gives it a higher bias towards evolving round shapes compared to the other representations. These are always using multiple rectangles and are confronted with shape fragments that disturb the performance.

4.4.5 Evolving Solutions for Environments with Enhanced Complexity

This section is dedicated to investigating the performance of the representations on a problem instance with enhanced complexity by applying irregularities in the landscape.

Landscape e is irregular; it contains a hole, a height limit, and is unevenly descending. Figure 34 shows the results. The performance of the three representations is investigated on this landscape with the evolutionary settings *S1*(**green**) and *S2* (**blue**).





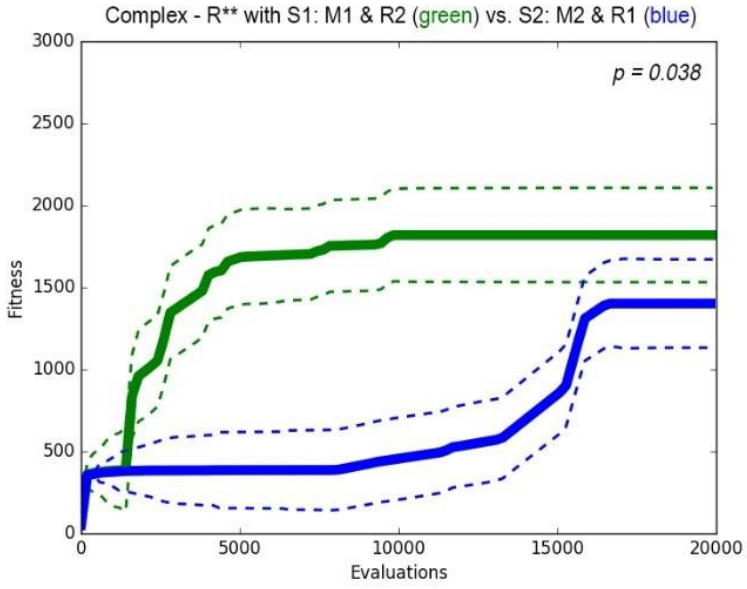


Figure 34: S1 and S2 for R, R* and R** on Landscape e

Overall, the plot shows that *R* does not perform well on *landscape d*. It gets trapped in a local optimum when using *S1* and *S2* ($p = 0.1147$). The wide confidence interval in the second plot shows that the algorithm appears to be able to escape from it when applying larger changes using *S2*. *R** performs equally well while using *S1* or *S2* ($p = 0.934$). For *R***, using smaller changes with *S1* leads to better performance compared to *S2* ($p = 0.038$). *R* does not perform well on the more complex landscape. When analysing the setting *S2*, *R** is the best performing representation compared to *R* ($p = 1.9 \times 10^{-6}$) and *R*** ($p = 0.000777$).

As previously suggested, the reason may be that *R* can place rectangles in a wider area around the centre, which may produce shape artefacts that disturb the movement of the shape component. *R*** is biased towards simpler shapes because of its ability to remove rectangles from the shape. It leads to evolving round shapes first, which move quicker at the beginning and get stopped by obstacles very quickly. The solution is not able to escape the local optima anymore. *R** performs well with both evolutionary settings and can evolve well-performing solutions when small or larger changes are applied.

4.5 Summary

An evolutionary algorithm was used to evolve the shape of a component and to adapt it to different landscapes in a physics environment. The capability to adapt shapes to its environment is crucial for an evolutionary system for evolving more complex mechanical systems, e.g. ones involving multiple components. The physics environment was implemented and functionally verified with unit and acceptance tests.

An evaluation was conducted of applying an evolutionary algorithm to evolve a shape component which can traverse a descending landscape pulled by gravity. The rectangle shape representation was taken from Chapter 3. It did not appear to function as expected, which led to the design of two modified versions of the representation. The first modification was a change of the maximum distance at which rectangles can be placed relative from the centre of the shape component. The second modification enabled the algorithm to remove rectangles when creating a shape.

- Firstly, the simulator was validated.
- Secondly, two different mutation operators and two different recombination operators were evaluated by comparison.
- Thirdly, the generative systems ability to evolve solutions by comparison to random sampling was evaluated.

- Fourthly, an investigation was conducted into three different representations, with two different evolutionary settings, using four problem instances with different complexity. The evolutionary setting S1 applied smaller changes to the chromosome than setting S2.
- Finally, the representations and evolutionary settings were applied to a more complex irregular problem instance.

The findings show that the simulator performed well in the evolutionary computing context. The representation R* with the modified placement constraint performed well through all problem instances and with both evolutionary settings. The initial representation R did not cope well with simpler problem instances as it evolved shape fragments that disturb the movement of the shape component. R** did perform well in simpler problems and worse in more complex ones. The reason for this may be that the representation is biased towards producing simpler (round) shapes as it is capable of removing rectangles. Simpler shapes were performing well on, e.g. a descending path without obstacles, and then, when obstacles were introduced, the representation was not able to evolve further and got stuck in its initially well-performing design.

The following chapter will extend the work by moving from evolving shape components into defining a framework for evolving conceptual designs of planar mechanisms.

5 The Conception of a Framework for Evolving Designs of Planar Mechanisms

5.1 Introduction

The following chapter is dedicated to a framework specifying boundaries, constraints, and limitations for evolving planar mechanical designs. The framework allows the definition of scenarios, using the simulator discussed in Chapter 4.

As previously discussed, an evolutionary representation, able to evolve single components to meet a design goal, was evaluated. In this chapter, it is taken forward, with a focus on evolving mechanisms. In this scenario, the algorithm evolves multiple shape components in a dynamic environment rather than a static landscape. Joints are used to attach the shape components to the bearing plate while introducing a rotatory movement and torque. The aim is to evolve a mechanism capable of moving as far as possible through a set of defined landscapes, within a given time. The solution's behaviour and its' performance are the effect of input movement, as well as shapes of components interacting with each other, and with the landscape.

This solution employs the same fitness evaluation as previously used, with the difference to evolve actuator driven mechanisms, rather than a single shape component. Furthermore, these mechanisms consist of multiple components acting as levers; revolution joints; mounted on a bearing plate.

The new set of different landscapes, even instead of descending, is provided. The forward movement results from the driving components rather than from gravity, which is acting on the complete system and pushing the mechanism onto the landscape. A scripting language is introduced that enables specifying design problems and solutions, and storing them in a file. The evolutionary algorithm is evaluated through a set of problem instances, designed specifically for experiments which include evolving the placement and shape of multiple components mounted on a bearing plate simultaneously.

The following sections will provide extensive background regarding the employed model, the method to evolve design solutions, and the results of the experiments.

5.2 Background

In order to gather the requirements, and develop a computational representation (i.e. a model) for planar mechanisms, it is important to understand the way these works. A large number of mechanical systems were summarised and classified by Artobolevsky (Artobolevsky, 1975). These are mostly planar and can be divided into gear, and lever systems. Gears can be represented in a simple way, e.g. by pairs of circles. A gear system's behaviour can be obtained by calculating the transmission ratio, which does not require simulation or resolving collisions and motion. Lever-type components transmit forces and movements in a similar way to gears; the main difference is that these can have an infinite variation in placement and shape, which makes it more complex to resolve the locomotion and transmission ratio between them. For instance, two levers may not always be in contact with each other while interacting on different sections of their outline. Obtaining the locomotion and behaviour requires a dynamic simulation.

Planar mechanical systems interact through collisions of interconnected components on a single axis plane. These can be either linked together in the form of lever chains or as individual components positioned in a relative distance from each other, occasionally getting into contact. Both types transfer motion and torque through the outline shape and the linkage. Each component has a mass, produces friction between itself and other components when in contact, and has restitution, which influences the behaviour of the overall mechanical system as well. Components can be mounted on a bearing plate, which keeps them in relative distance to each other. A mechanical system can be imagined as a clockwork mechanism where multiple components need to be positioned, constraint, and shaped in a specific way, to perform the desired design task. Assembling a mechanism made out of random lever combinations often does not generate movement as it is likely that they hinder each other. In addition to levers, a mechanical system consists of joints, which constrain the motion of a lever.

Mechanism design is a broad area that can include a variety of parts. This work focuses on a limited set of them and their virtual representations. It excludes components such as specific joints, e.g. translational joints; springs; and dumpers, as the implementation of those would be beyond the scope of this work. However, the simulator allows such extensions to be implemented in future.

The proposed model is a 2-dimensional representation of planar mechanisms at a lower level of detail, which decreases the number of parameters needed to describe it (Pahl et al., 2007). The model focuses on the shapes of components and their interactions between

each other. The attention is placed upon early-stage design prototypes and does not consider a more detailed evaluation of their elastic behaviour.

5.2.1 2-Dimensional Environment

The 2-dimensional environment is the space in which virtual physics objects can be placed. It consists of multiple layers. Each layer may contain the shape and position of components and their joints. Collisions between components can only take place if their shapes are on the same layer. Figure 35 shows a multi-layer environment.

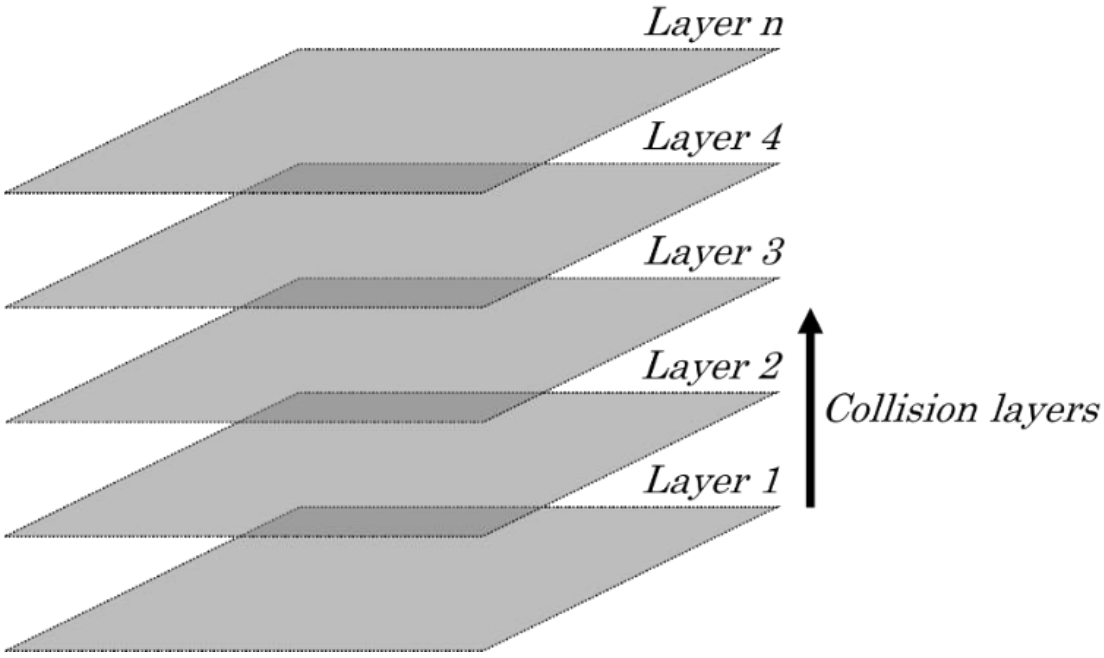


Figure 35: 2-Dimensional Multilayer Virtual Environment

A mechanism may consist of multiple components, and each component may be made of multiple shapes. It may also have shapes on different layers which enables the representation of 3-dimensional components in 2-dimensions as long as the collisions take place in one axis plane on the same layer. As an example, Figure 36 shows the representation of a 3-dimensional component with an undercut, as one component consisting of three shapes, placed on three layers.

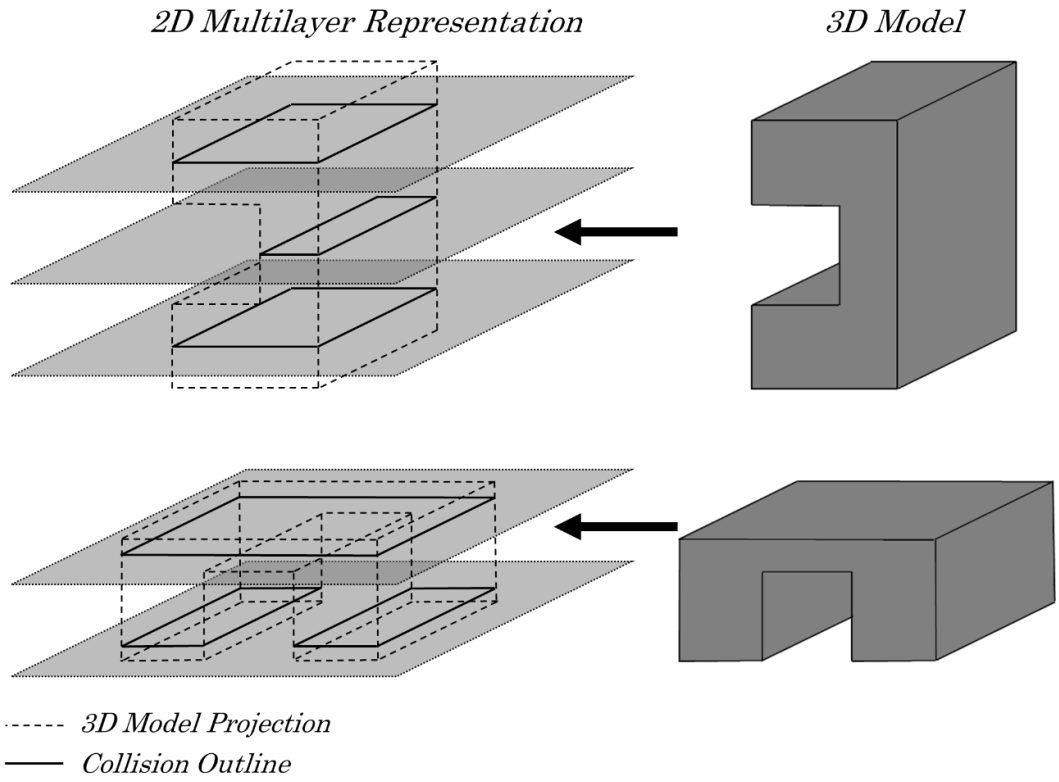


Figure 36. Representation of a 3D Model in the 2D Environment

The 3-Dimensional component is broken down into three rectangle shapes, each placed on a different layer. However, all move simultaneously and behave as one. Each rectangle can collide with other shapes on the same layer with an influence on the other shapes on different layers.

5.2.2 Lever Representation

One component of mechanisms is the lever. It interacts with other components via its outline shape, which results in the motion of the system. Figure 37 shows a real lever component taken from an automotive closure system of a car lock.

Figure 37: Real Lever Component

A real lever component has many characteristics, such as areas acting as shock absorbers to reduce noise, sections of material reduction, and positioning points used for manufacturing purpose. These areas are not relevant for obtaining the locomotion and are not considered in the model.

Instead, this research focusses solely on the virtual representation in the initial prototype design state. Once finding a well-performing mechanism, its components can be taken further to the production stage that would require choosing the right material according to the appearing forces, as well as adding cavities for material reduction and dumpers. Figure 38 shows the simplified virtual representation of the real lever.

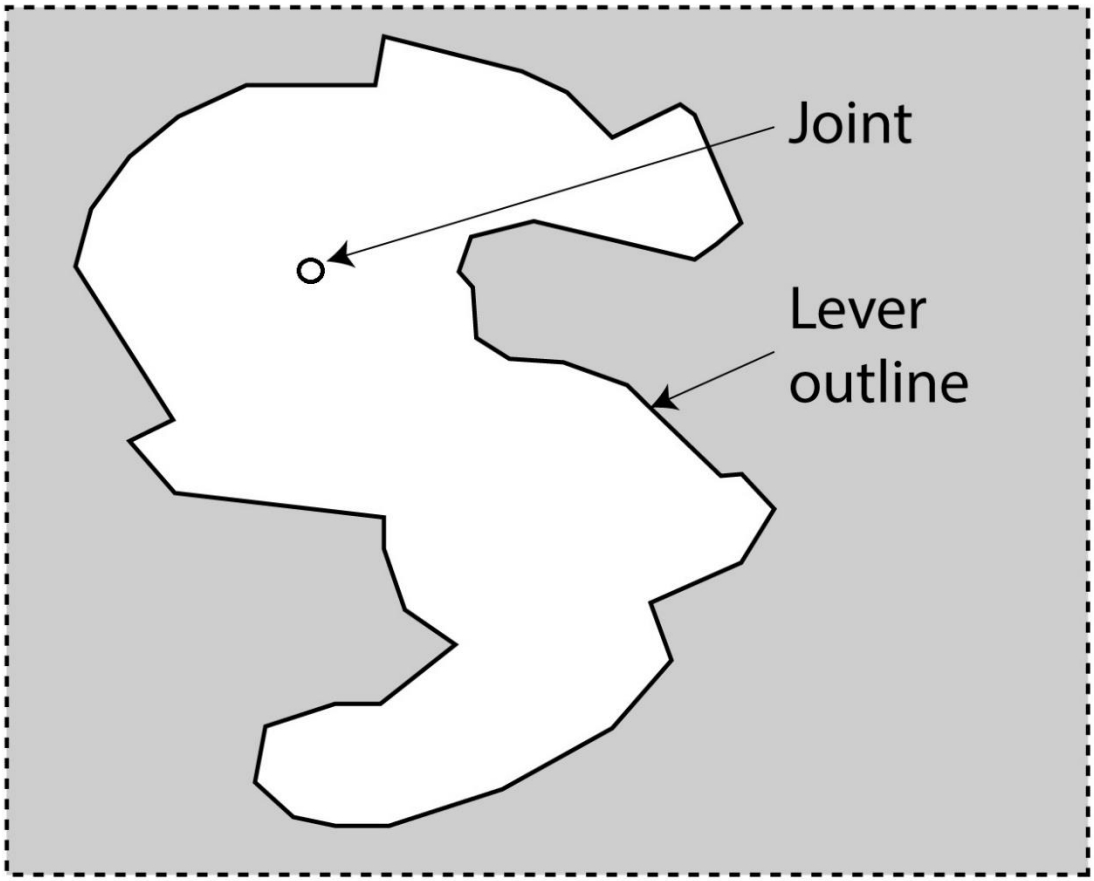


Figure 38: Lever Representation

Each lever has an outline, a joint position or, in case of a lever chain, multiple joins connecting it to other components. The virtual representation does not take internal cavities into account, as they are not relevant in the context of obtaining locomotion. It is either a single interconnected structure or when thinking 3-dimensional, it may also consist of multiple not connected shapes, connected in another layer, such as previously explained in section 5.2.1. Levers may be symmetrical, although often do not show symmetries. Some sections of the outline will collide with others, or the environment;

other sections will not. Figure 39 shows a component placed on a virtual plain with one layer.

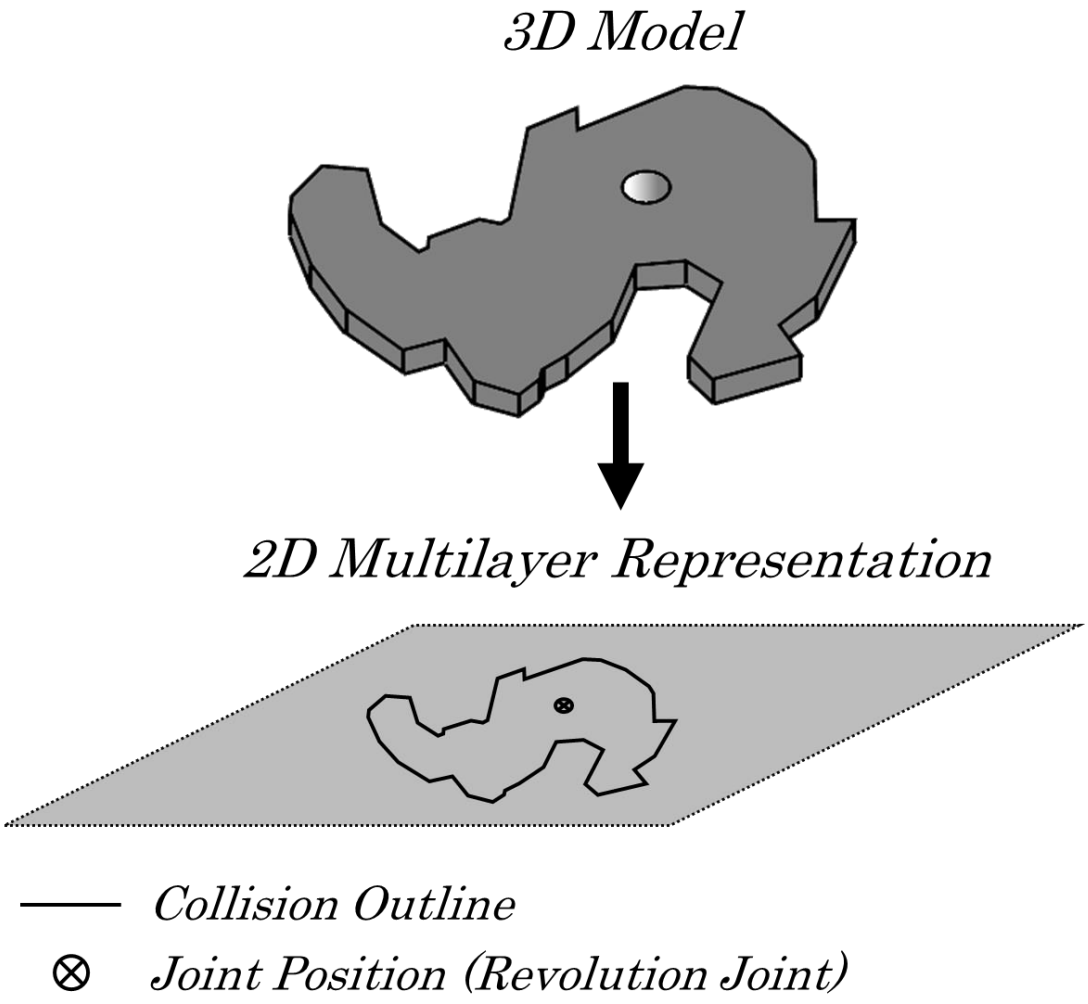


Figure 39. Representation of 3-dimensional Lever

5.2.3 Joints

Joints are functional components and have a position but no shape. They constrain the freedom of movement of lever components which influences their kinematic behaviour. Joints connect lever components to the bearing plate, or each other, which constrains the component's freedom of movement. This research considers revolution joint which constraints a component only to conduct a rotatory movement around the joint.

5.2.4 Actuators

An actuator introduces input forces and movement to the system. It acts upon a component which is attached to the environment or a bearing plate with a joint. Attributes are assigned to the joint, such as the rotation speed in revolutions per minute, and a maximum torque that can be applied. Figure 40 shows an example of the rotation and torque specification.

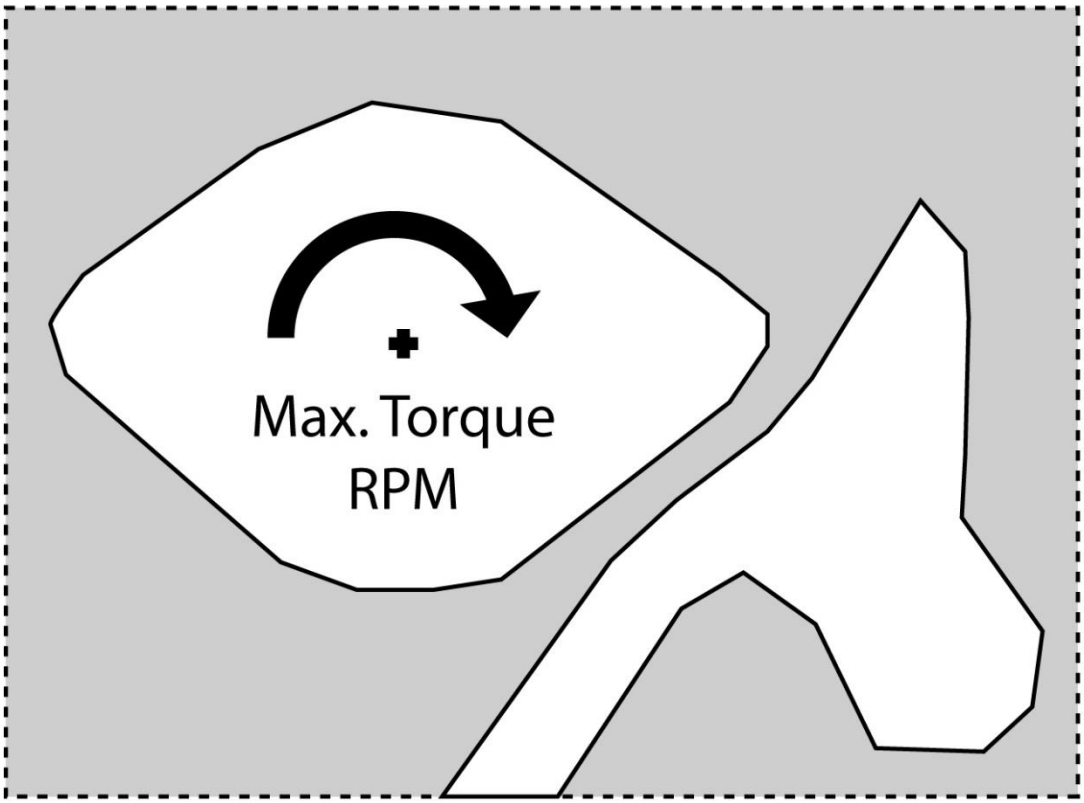


Figure 40: Rotation Torque and RPM

In this figure, a lever component is attached to the environment with a joint. The joint has a revolution per minute and maximum torque specification which sets it into motion. It represents the input characteristics of, e.g. an electric motor.

5.2.5 Mechanism Representation

The mechanism is virtually represented in an abstract way which lowers the level of detail and focuses on the essential parts which are necessary to decrease the number of parameters describing it (Pahl et al., 2007). In this way, the search space can be scaled down, which lowers the processing time to evolve solutions. Figure 41 shows a photograph of a locking mechanism taken from an automotive closure system and the abstraction into a virtual model.

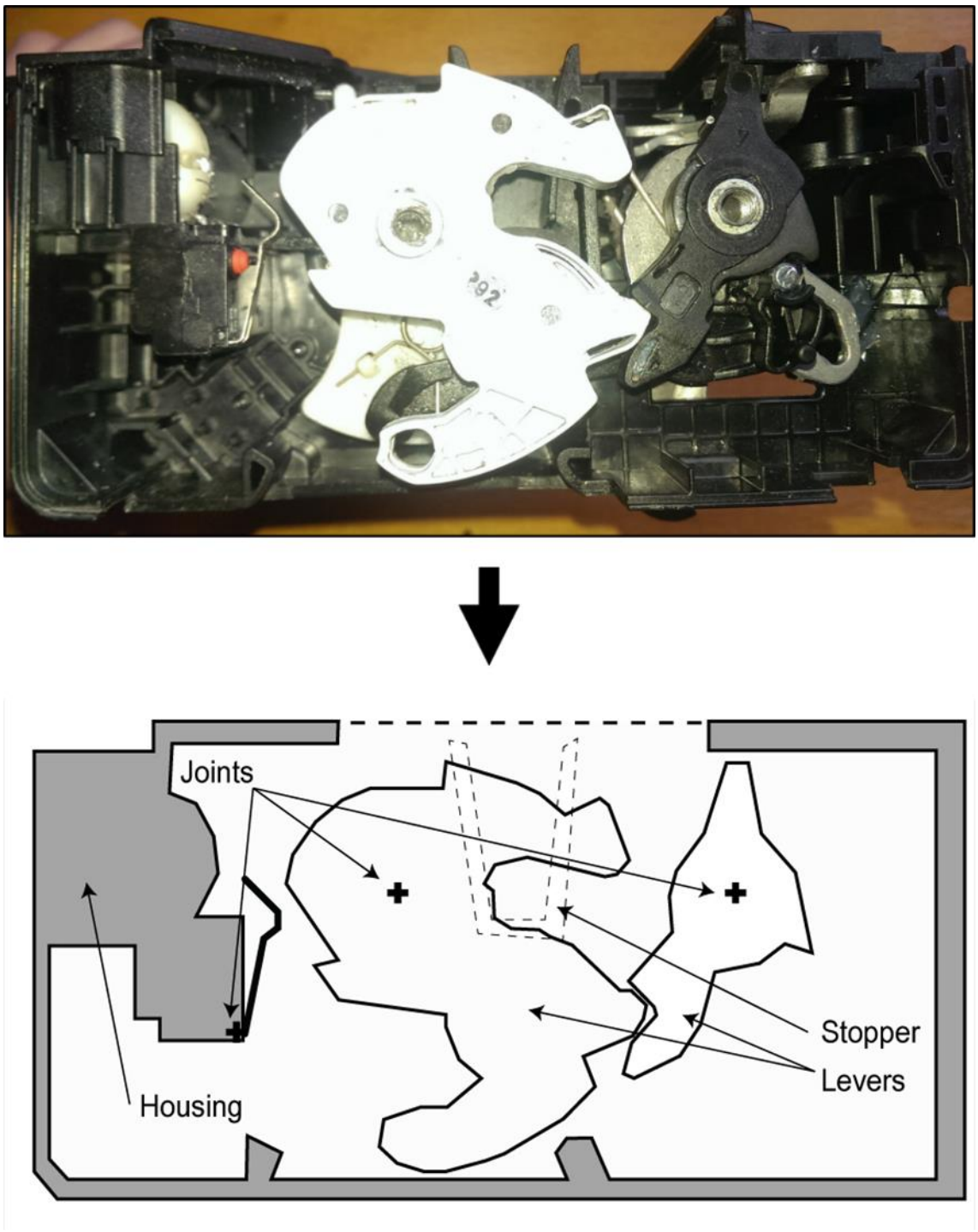


Figure 41: 2-Dimensional Mechanical System

In the virtual representation, the components are rigid bodies with no elasticity. The 3-dimensional system is simplified to 2-dimensions on two layers. The first layer contains the housing outline and two lever components, which are constraint by two revolution joints. The housing is static in this case and does not move, whereas the lever components are dynamic and able to move within their boundaries. Both can collide with each other. The second layer contains a static wall element in the background of the lever components on a different layer. It does not collide with the other components.

5.2.6 Mechanism Types

Two types of planar mechanisms were implemented. The first type of mechanism is a set of individual components connected to the environment or a bearing plate using joints, although with no connections between each other, as shown in Figure 42.

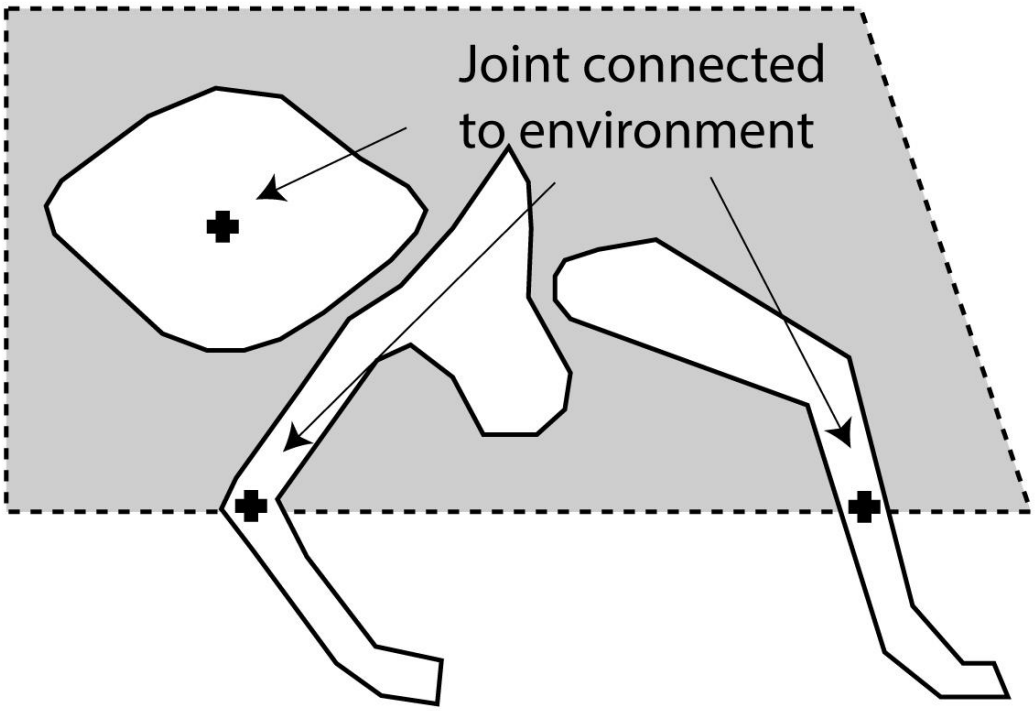


Figure 42: Individual Components

The components have a relative distance to each other. If a driving component introduces motion, it transfers them to others, over their outlines, through collision.

The second type is the lever chain or linkage. These consists of a set of lever components, interconnected with each other using joints. At least one component in the linkage is attached to the environment or a bearing plate. Components, which are directly connected do not collide with each other; however, they can collide with other components in the environment.

Figure 43 shows an example of a lever chain with two components connected to the environment, and another one connecting the two components with two joints to each other. If one component moves, it moves the other components due to their connection.

Both types of mechanisms can be combined, as shown in Figure 44. The figure shows a lever chain on the left-hand side consisting of three components that can collide, with a single lever component on the right-hand side. Both are connected to the environment using joints.

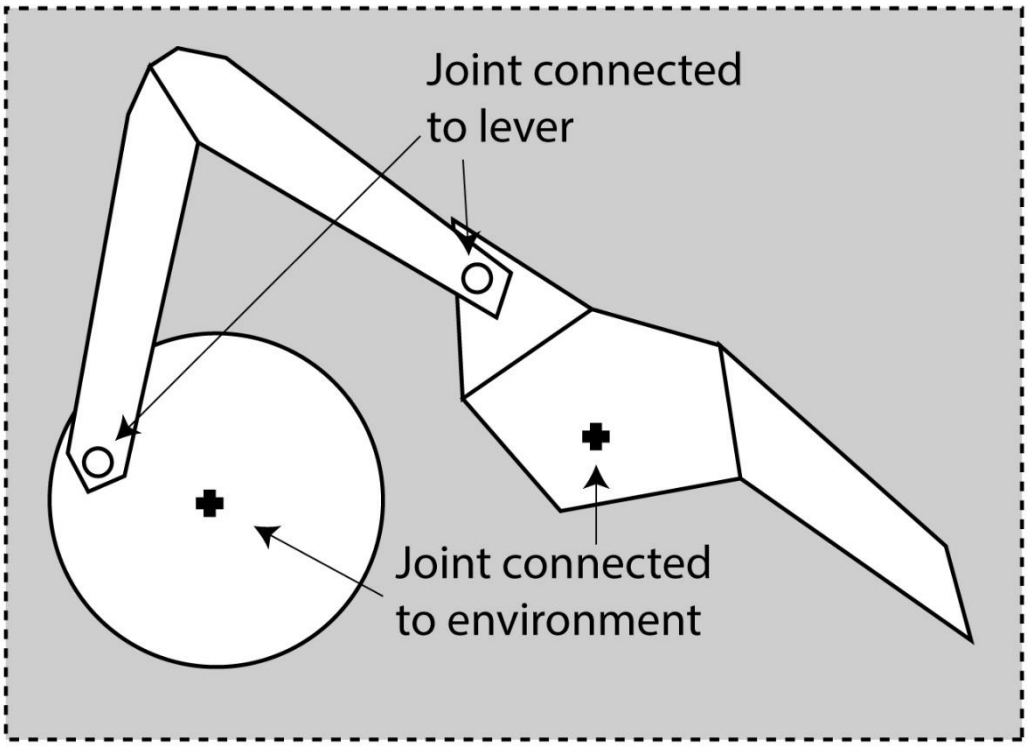


Figure 43: Linkage

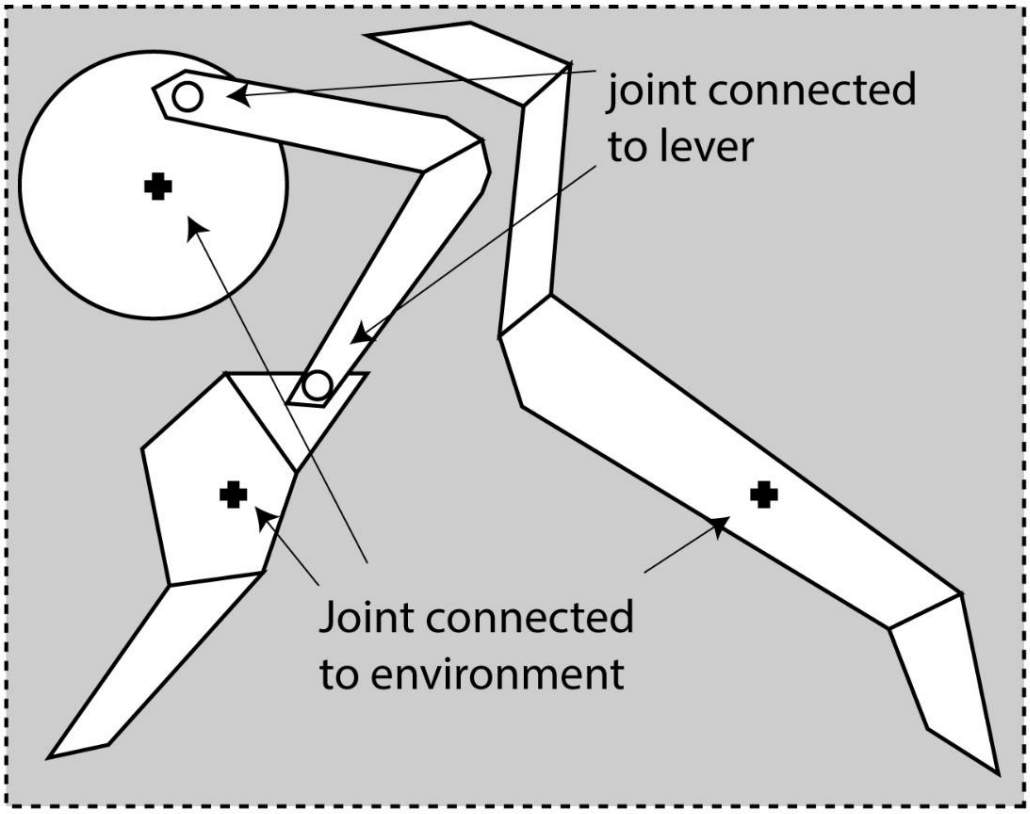


Figure 44: Linkage with Collision

5.2.7 Bearing Plate

Mechanical components can be attached to a bearing plate. The bearing plate holds the mechanism together and specifies the relative distance between joints. It is a dynamic component as it can move freely in a virtual environment.

5.2.8 Problem Scope

The problem scope includes a variety of components such as lever components, obstacles, and bearing plate; joints, and actuators. These are explained in Section 5.2. Other components, such as walls and stoppers, fulfil the role of obstacles in the 2-dimensional environment. These can be static or dynamic components. Static obstacles such as ground path, and walls, cannot move. However, they collide with other dynamic components on the same layer. Dynamic obstacles are components which can move freely. Static and dynamic obstacles are elements placed in the environment.

A bearing plate is a dynamic element in the environment. It represents a surface for attaching lever components with joints, or dynamic obstacles, such as walls and stoppers. A bearing plate may correspond to mechanism-housing which guarantees that all attached lever components with joints and walls have a constant relative position to each other. Together they assemble the mechanism. The problem scope provides a configuration of physics parameters, such as gravity for the environment, or mass, friction, and restitution can be set for every component individually.

A design scenario needs to be defined beforehand by, e.g. an engineer, in a computer-readable format, e.g. in a file. A scripting language was developed to enable the definition. An interpreter translates the file into a physics scenario, which can be processed by the physics simulator. The resulting simulation, a sequence of frames, is written into an output file for further analysis. Appendix 3 provides an example of a problem definition file.

The file is based on the eXtensible Markup Language (XML) using the Scalable Vector Graphics (SVG) standard with customised tags, compatible with being opened in a browser. Figure 45 shows the structure of the file.

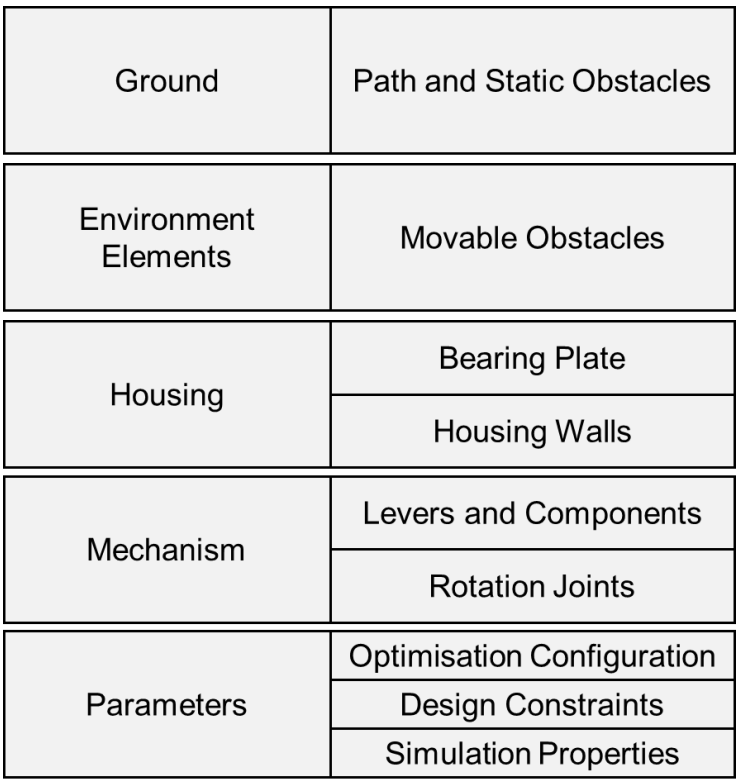


Figure 45: File Structure

The file consists of five sections. Four of them are describing the geometric and material properties of the environment and its components, and one section contains parameters for the evolutionary algorithm, design constraints, and the simulation properties.

The first section describes the ground. It specifies the static shape of the landscape, path and obstacles. The second one defines the environment elements. Movable obstacles can be included in the environment. The third section specifies housing, which is movable in space. It can contain walls or mechanical stops. The fourth is the mechanism, the solution, which consists of lever components, their shapes, and joints. The last section contains the parameters for the optimisation algorithm; the geometric constraints of the mechanism, such as minimum and maximum size; and parameters for the physics simulator.

5.2.9 Solution Hierarchy

The solution can be a single component or multiple components connected with joints to a bearing plate. Figure 46 shows the hierarchy of the solution.

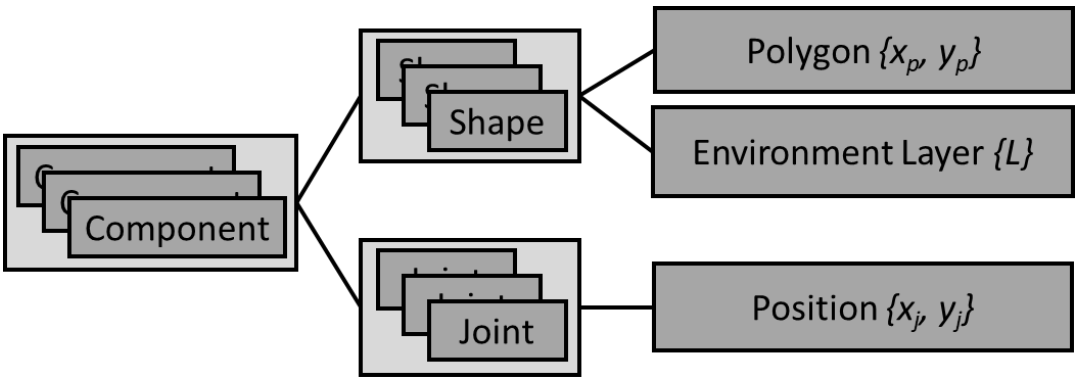


Figure 46. Planar Mechanism Model

The bearing plate can move freely on a layer. It contains the components. Each component may consist of multiple shapes and can have multiple joints. Polygons describe the shapes with an array of coordinates placed on one layer. Shapes can be placed on multiple layers simultaneously, as previously explained. Shapes on the same layer can collide with each other, whereas these on different layers do not collide. Each component may have one or more joints assigned with a position coordinate and a specification which components it links together.

5.2.10 Design Objectives

Usually, the focus in mechanical design is to move a lever from one to another position by working against forces and moments such as shown in Figure 47.

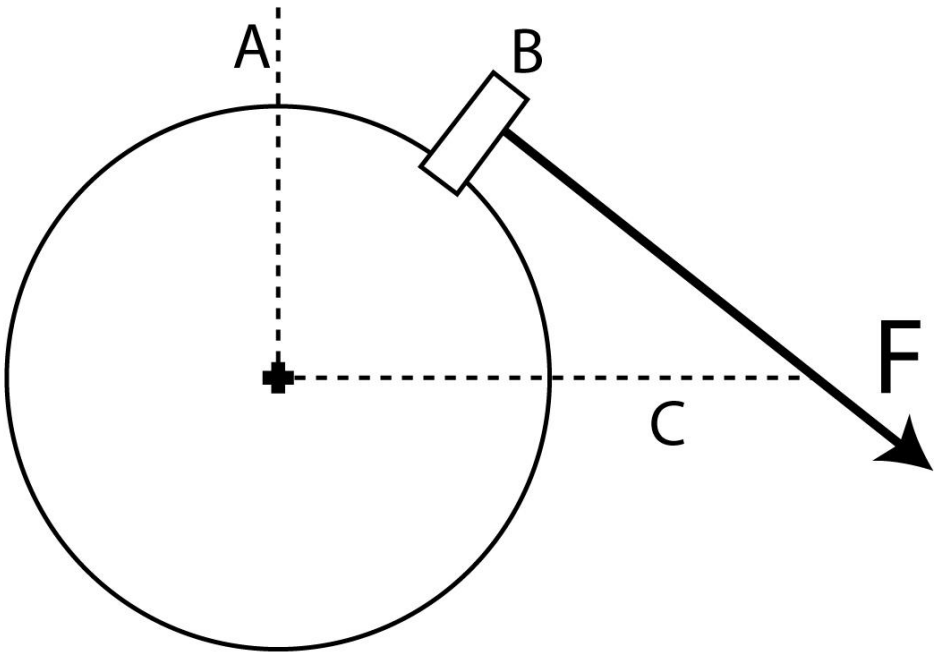


Figure 47: Force and Movement Objective

The figure shows three positions. The component's initial position is *A* and the desired position *C*. The component is pushed anti-clockwise in the direction of *A* by an incoming

force, e.g. initiated by other components. A minimum force *F* is needed to pull it to position *C*. If *F* is too small, the component would not move and stay in position *A* or not reach *C* completely. *F* needs to be high enough to rotate the lever into the desired position *C*. As larger the applied force is, as faster the component would move to the desired position *C*.

Another design objective may be to follow a specified path or rotation. Figure 48 shows an example of a path specification.

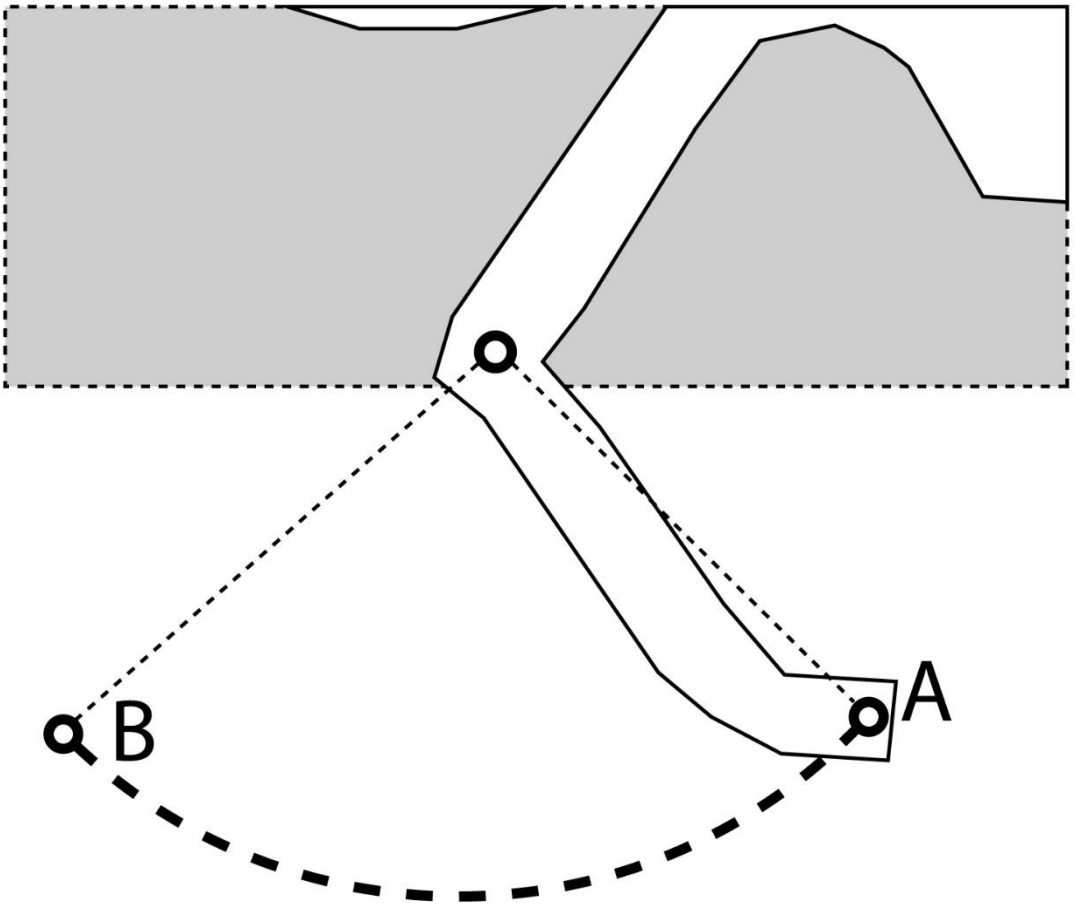


Figure 48: Path-based Objective

A is the start position of the lever and *B* the desired end position. The thick dashed line is the desired path. An additional objective may be the desired time to complete the movement. A component with a specific shape needs to be attached with a joint in the exact position to allow a driving component to push it in the right way to enable it to follow the path from *A* to *B*. The evaluation process, in this case, would be a comparison of the desired position and the measured position, and the time needed to reach it. Path-based objectives are evaluated through comparing of the undertaken path and desired path. Techniques such as Procrustes Distance Calculation are used to compare paths, especially if desiring more complex paths (Dryden & Mardia, 2016).

Procrustes Distance Calculation is a mathematical comparison of the similarity of different paths usable for fitness evaluation. Another technique is Dynamic Time Warping. It can be used to compare the similarity of time or distance-dependent requirements, such as force changing over time or distance. (Ratanamahatana & Keogh, 2004)

As explained previously, the design of planar mechanisms can have a variety of different objectives related to specific motion and forces at an output which is highly dependent on the specific design task. In real-world design scenario, often an output component needs to move in a specific way, apply a specific force, or should not extend a specific torque. The variation of design objectives allow defining many real-world design scenarios; however, these are often not directly comparable and may be very particular or focus on one mechanical part. This research does not focus on these because evaluating a generative system requires more flexibility. In this work, the overall behaviour of the mechanism is evaluated to obtain the overall performance instead of focusing on the behaviour of a single component. This approach makes it easier to evaluate the generative system instead of focusing on individual design cases.

The emphasis on a high-level behavioural objective enables comparing the performance of different scenarios, whilst keeping the fitness function exchangeable to any design problem. It removes the focus from specified target forces and movements, which can often be only approximated or not solved at all depending on the problem. Instead, it gives attention to the global aim of moving a mechanism forward, which always provides a solution.

The fitness evaluation of a potential solution is based on the output of the simulator, therefore on the configuration of the complete scenario throughout a defined timespan. This type of output enables making kinematic analysis and can be used to implement other objectives, such as producing specific desired component motion, making it extendable to other design objectives in future.

The area of Artificial Evolutionary Life Forms focuses on the evolution of high-level behaviours, such as swimming, walking, jumping, or following (Bentley, 1999). In this work, the design objective was defined similarly to moving a complete mechanism forward as well; it takes place through environments with different terrain and obstacles. Input parameters for the driving components such as revolution per minute and maximum torque, and an initial design, such as bearing plate to place a solution design, can be predefined. Similar distance-based evaluations were done in soft-robotics (Cheney et al.,

2013), evolving simple car shapes ("BoxCar2D," 2015), and Genetic Algorithm walkers (Matsunaga, 2015). Evolutionary Artificial Life-Forms also used behaviour based fitness evaluation (Bentley, 1999). The approach utilised in this work considers the behaviour derived from the shapes of components with a constant input patter. In other research, the focus was often placed on evolving the input pattern on a predefined design or simpler shape manipulation using building blocks. This approach provides the freedom to focus on the evaluation of the generative system, rather than on the definition of objectives for mechanical design. At this stage, it provides the basis to create a set of design scenarios and to investigate the performance of the evolutionary algorithm in solving them.

5.3 Method

This section investigates the capabilities of the evolutionary algorithm and the framework in evolving design solutions. The previously evaluated shape representation from Chapter 4 is extended to be used to evolve multiple components mounted on a bearing plate. Several landscapes were designed for experiments to evolve mechanisms capable of traversing these landscapes within a fixed time.

5.3.1 Evolutionary Representation

The representation is capable of placing lever components on a bearing plate with two fixed joint positions. Both components are set up as actuators with a speed and torque specification which is also optimised by the algorithm. These have an upper and lower limit, which were found suitable through initial testing. The speed can vary between 15rpm to 60rpm, and torque varies in a range from 10 Nm to 80 Nm. The evolutionary representation is based on a rectangle shape explained and evaluated in previous chapters. It was extended by multiplying the number of genes, enabling the representation of two components, and adding additional genes to evolve the speed and torque for each component. Figure 49 shows an initial solution created by the representation in a landscape with stairs.

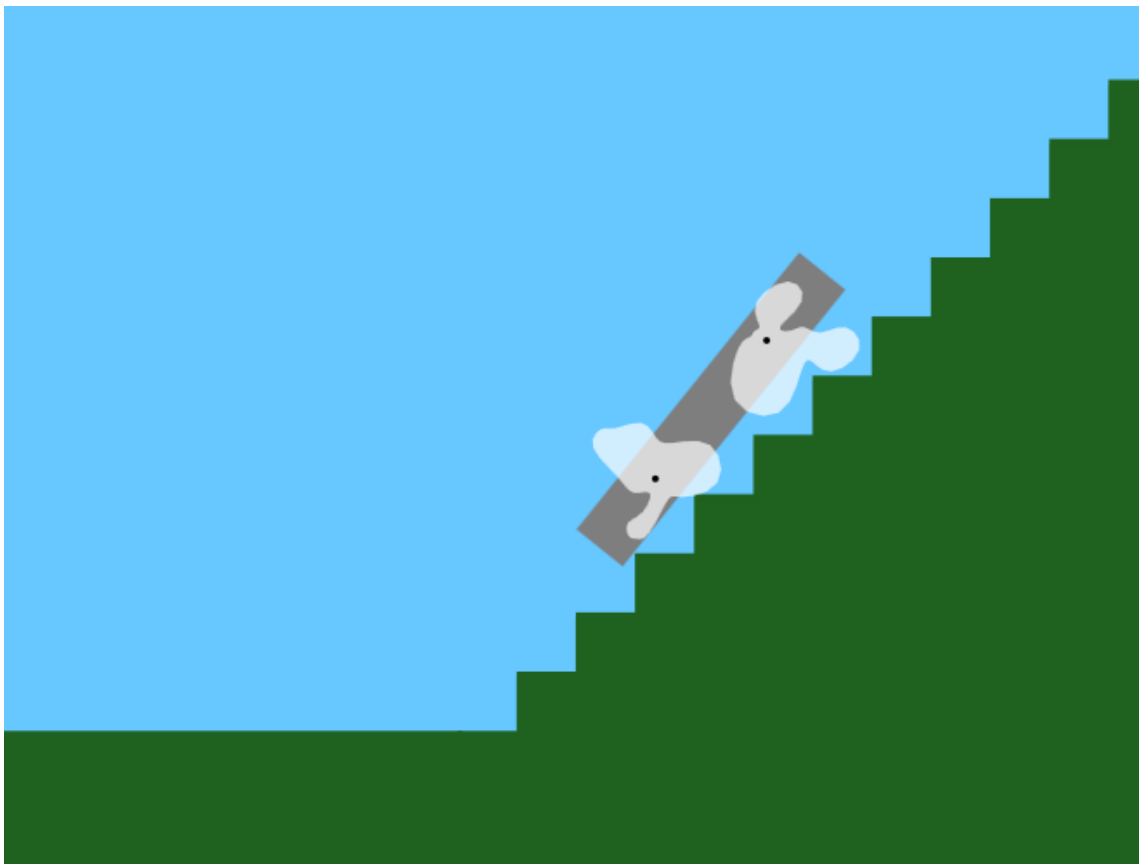


Figure 49: Mechanism with Two Levers Climbing Stairs

The scenario shows a mechanism which is climbing stairs. It consists of two lever components which are attached with joints to a bearing plate.

5.3.2 Evolutionary Algorithm

The evolutionary algorithm is similar to the one used in the previous chapter with the following configuration: The population uses 40 individuals and produces ten children in every generation. These parameters were chosen after the initial testing. Previously, findings showed that the rectangle representation *R** worked well with evolutionary setting *S1*, so it was used further during the experiments.

5.3.3 Fitness Evaluation

The objective is to evolve a moving mechanism which is capable of traversing different landscapes. The evolutionary algorithm evolves a mechanism with two lever components mounted on a bearing plate to traverse different landscapes. The objective is to maximise the mechanism's position in the x-direction at the end of a specified timespan. The distance is measured from the middle point of the bearing plate in the first frame to the middle point of the bearing plate in the last frame. In contrast to the previous chapter, the extended representation can produce a mechanism.

5.3.4 Experiments

A bearing plate is positioned on a landscape with gravity applied in the negative y-direction, set to 9.81 m/s^2. All components were given the same material parameters for density (1.0), friction (0.5) and restitution (0.6). Five different environments were designed to investigate the generative system's abilities to produce solutions for these environments, shown in Figure 50.

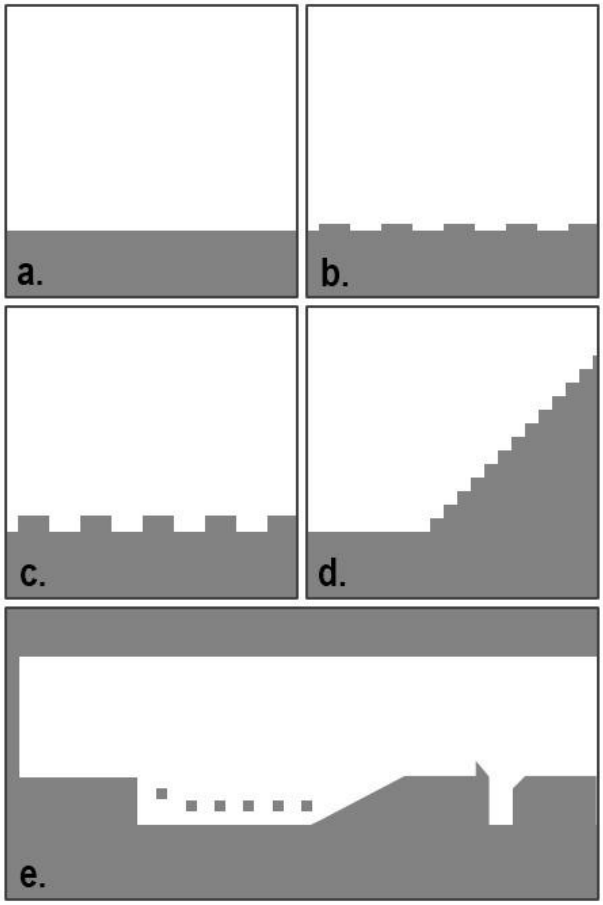


Figure 50: Environments

The figure shows a straight *landscape a*; a digital shaped *landscape b*; a second digital shaped landscape with different scale *c*; as well as a landscape containing stairs *d*. Furthermore, a complex *landscape e* containing different obstacles, such as uneven terrain, walls and holes.

Experiments were run 24 times on each of the environments at 60 frames per second for 600 frames which equates to 10 seconds of simulation on each landscape. The complex environment was simulated for 1,800 frames which equates to 30 seconds of simulation - as the environment is changing over a longer path. This configuration was found to be an appropriate balance between outcome and simulation time. Each experiment stopped after 20,000 evaluations.

5.4 Results and Evaluation

Experiments were run to investigate the generative system's capabilities to evolve a mechanical system consisting of multiple components attached to a bearing plate with revolution joints. The objective was to efficiently traverse a set of provided landscapes driven by joint actuators.

- Firstly, the evolutionary algorithm is evaluated in its ability to evolve solutions for five problem instance (landscapes) by comparison to random sampling.
- Secondly, solutions evolved for each landscape are investigated.
- Thirdly, the simulator limitations are discussed which emerged throughout the experiments.

The Mann-Whitney U-Test was used for statistical analysis since normality of the distributions cannot be assumed. A *p*-value of $p \leq 0.05$ indicates high confidence that distributions significantly differ. The *p*-value refers to the median distribution of best perfroming solutions at the end of each run.

5.4.1 Evaluation of the Generative System's Ability to Evolve Solutions

In this section, the evolutionary algorithm ability to evolve solutions for five different landscapes is evaluated, recording the fitness increase over 20,000 evaluations. The evolutionary algorithm is compared to random sampling to investigate if evolution is happening.

Figure 51 shows the performance of random sampling (**red**) and the performance of the evolutionary algorithm (**blue**) for five landscapes. The p-value shows if distributions are significantly different for each comparison.

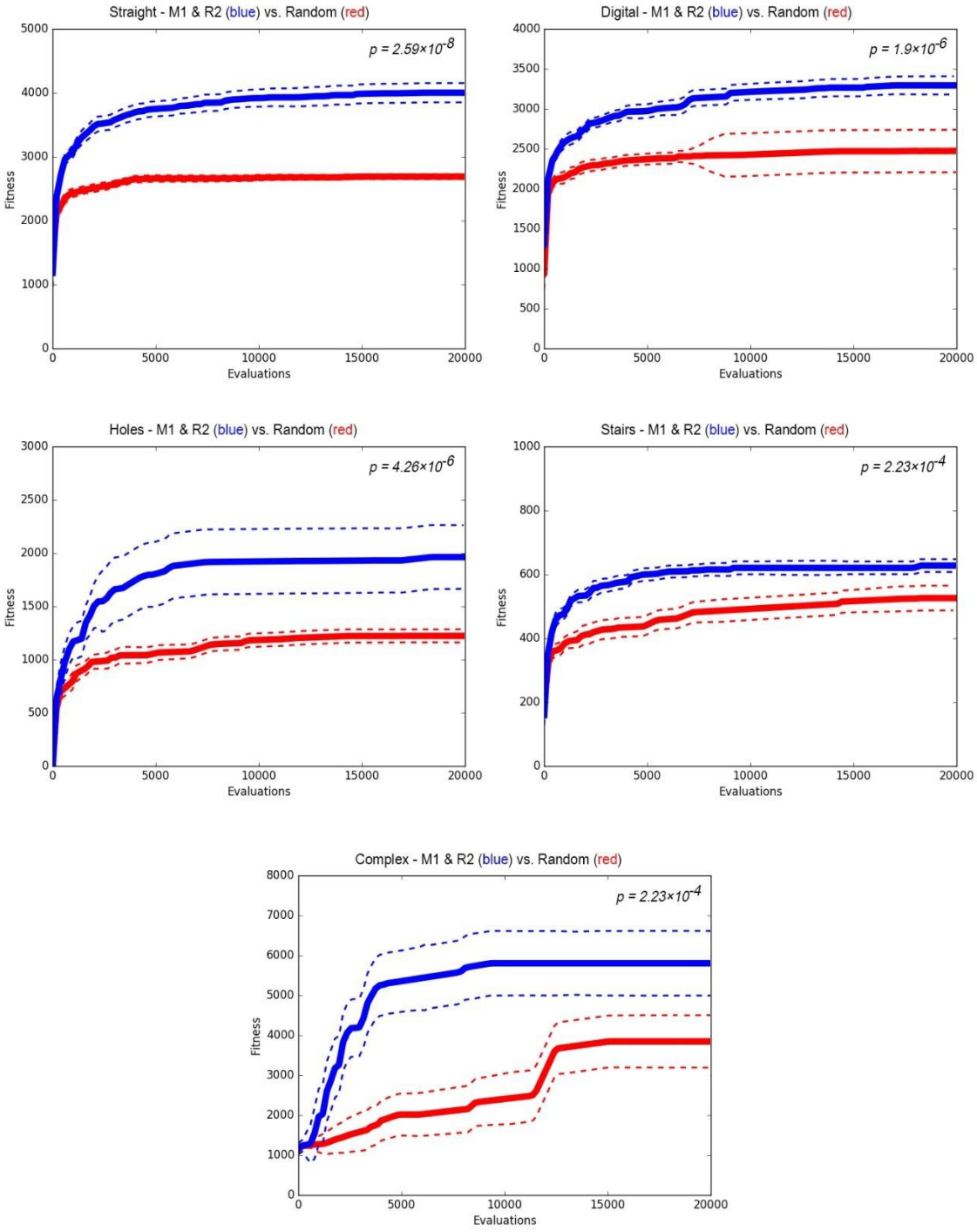


Figure 51: Evaluation of Evolvability

The results show that the evolutionary algorithm outperforms random sampling. All distributions significantly differ from each other. It indicates that the algorithm can evolve solutions and overcome local optima which can be observed in the complex landscape. Random sampling produces solutions which get stuck in obstacles in the beginning and is just occasionally able to overcome them.

5.4.2 Evolved Solutions in Different Landscapes

Figure 52 shows the five landscapes, including one evolved mechanism for each of them. The mechanisms consist of a bearing plate with two attached driven levers that may apply different speed and torque.

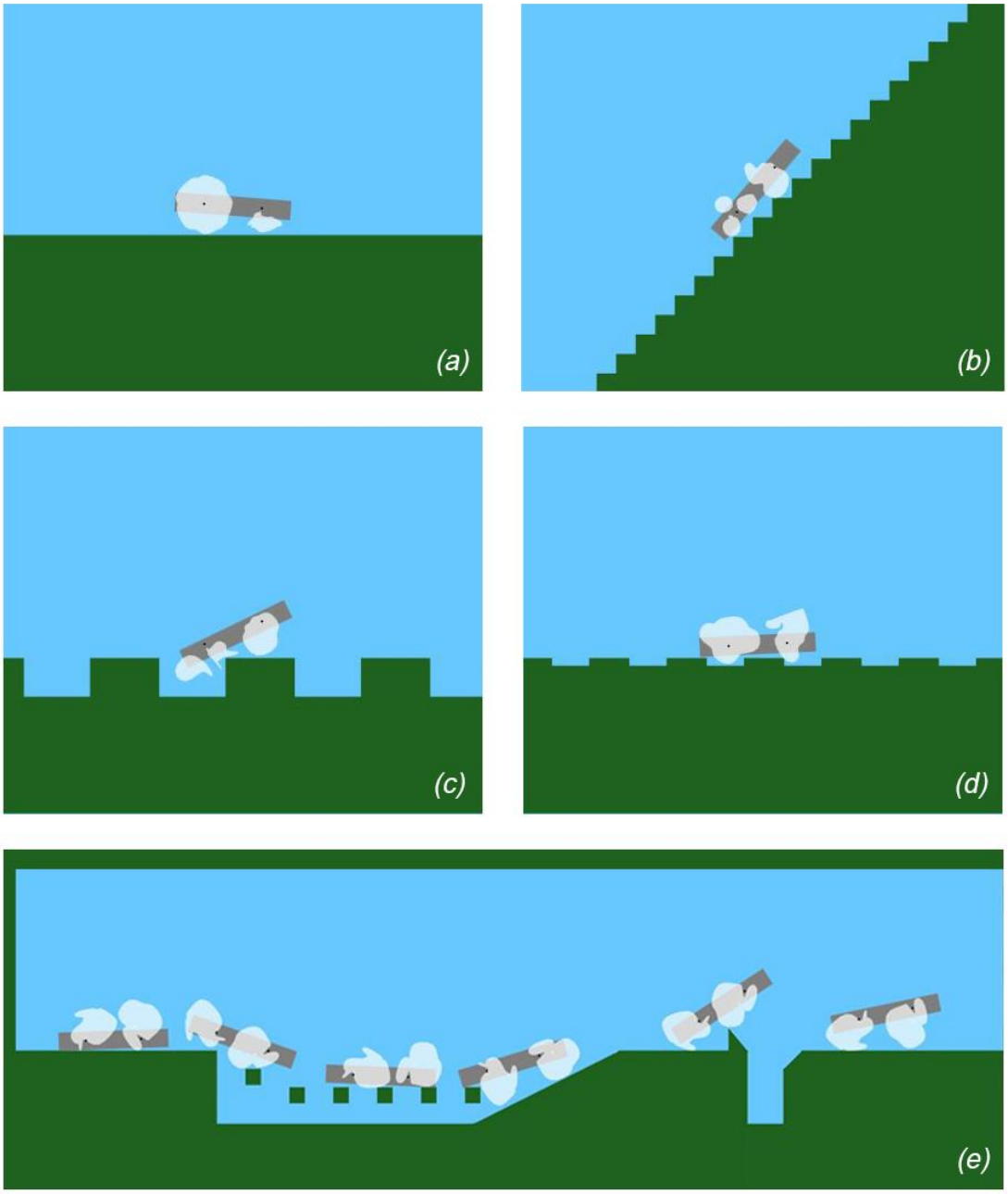


Figure 52: Mechanism Solutions in Different Landscapes

Plot *a* shows a solution evolved in the straight landscape. It consists of an approximated wheel type lever on the rear, with a joint in its middle point, and an asymmetrically shaped lever in the front. The front lever's outer shape is rounded and might contact with the ground surface. The rear wheel drives the assembly forward, whereas the front lever moves the bearing plate's front part periodically up, whereby the front part of the bearing plate loses contact with the ground for a short time. The rear lever is not perfectly round,

which would be expected from an optimal solution; while, it is capable of moving the mechanism forward, and the middle point is centred. The uneven characteristic may produce more friction between ground and lever, which would lead to less slipping. The front lever does not block the rear lever, and also does not hinder the mechanisms in its forward movement. The bearing plate is not steady throughout the forward movement; however, it was not the objective. The periodical uplift of the front lever and contact loss with the ground surface may reduce friction, which benefits the rear lever in pushing the mechanism forward. This may be the reason why the front shape just partially evolved round characteristics.

Plot *b* shows a mechanism evolved in a landscape containing stairs. It has a lever consisting of circle shapes, arranged in approximately 120 degrees angle between each other around its middle point, which is close to the joint. In each lever rotation, one circle lands on top of one stair and pushes the mechanism up. At the same time, the front lever, with a decentred asymmetric shape, acts as a mechanical stop by pushing itself against the front side of a stair, each time the rear lever reaches a new step. Furthermore, each time the front lever pushes against the front of a stair, a segment of shape lands on the top of it, and while rotating, lifts the front of the bearing plate. In this case, the front of the bearing plate does not get caught in the stairs.

Plot *c* shows a mechanism evolved in a landscape with periodic holes. The rear lever consists of two shapes, which are decentred from their rotation point. The shapes have segments with hook characteristics. One hook is pushing the mechanism forward by catching the edge of the hole, which makes the rear part of the mechanism slip into the hole. The other hook lifts it out of a hole again. The front lever has round characteristics and is also out of the centre. It fulfils a forward pull by using the inner hole walls. At the same time, it guides the bearing plate front and avoids getting blocked in a hole.

Plot *d* shows a mechanism evolved in a landscape with a tooth-shaped surface. The rear lever's middle-point is positioned close to the joint. It has round characteristics with notches. The notches avoid the contact with the tooth edges, only the round parts of the shaping role over the straight tooth bottom and the top. The front lever is asymmetrical and has segments which hook into a tooth and pull the mechanism forward. Other segments are straight and slide over the top of a tooth and lift the front of the bearing plate up to avoid getting blocked in it.

Plot *e* shows an evolved mechanism in different positions while traversing the complex landscape. Both levers are asymmetrical and out of the centre. It is difficult to assign

specific functions to the shapes' segments because the interactions with the ground are versatile. The mechanism is able to pass all obstacles by having the right timing, shape, and weight distribution, which is not the case for randomly created solutions. Most of them tend to get stuck in obstacles.

In all solutions, both levers do not interfere with each other or slide against each other without getting stuck. The mass of the levers needs to be considered. Sometimes parts of the lever may seem unnecessary; however, because of their mass these may be important to balance the mechanism. Yet, this may lead to unreasonable looking shapes.

The discussed solutions were selected due to their interesting shapes and interactions while still being describable or explainable. The variety of solutions is wide if considering all conducted experiments. Shape segments contribute in different ways to a more or less steady forward movement. The shapes are very complex and difficult to describe. Sometimes the same shape sections have more than one function, especially when evolved in the complex environment *e*. The landscapes *a-d* have a repeating ground path, so shape segments seem to fulfil specific repeating functions.

Random sampled solutions tend to look more complex; however, they get stuck on the path in obstacles, or the lever motion is not synchronised, which leads to levers blocking each other.

5.4.3 Simulator Limitations

The simulator in combination with the generative system encountered several issues which needed to be addressed, namely:

- Intersections with the ground surface.
- Overlapping shape components.

Initially, the bearing plate was positioned closer to the ground surface. The generative system produced lever components which occasionally overlapped with it, such as shown in Figure 53 *a*. The physics engine usually resolved the overlap, however, in some cases, the overlap was too large, and one or multiple components got stuck in the ground surface which led to unreasonable behaviours, such as an unstable simulation, jumping of the component, sometimes even catapulting the mechanism out of the scene. This problem was addressed by placing the mechanism higher and dropping it on the ground surface, which resolved the problem, shown in Figure 53 *b*.

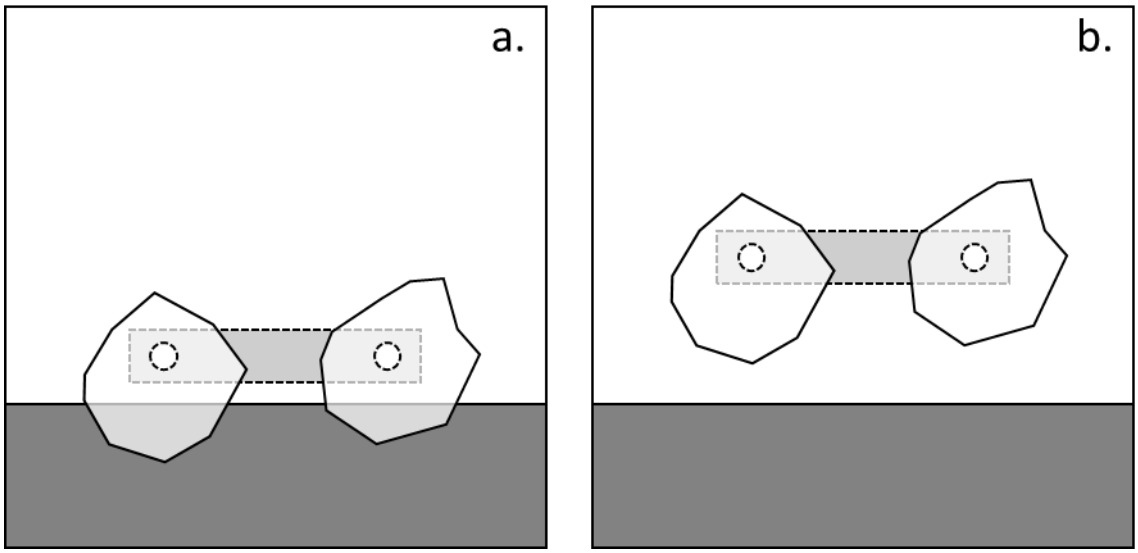


Figure 53: Overlapping with Ground

In some cases, the evolutionary algorithm created initial solutions which may overlap with other components in the first frame, such as shown in Figure 54. Again, the physics engine was able to resolve most of them and rearrange the components. However, components were occasionally jammed together in a way that the physics engine was not able to resolve the overlap. An example is shown in Figure 54 *b*. Either it was too large, or there was no mechanism configuration available without overlapping components.

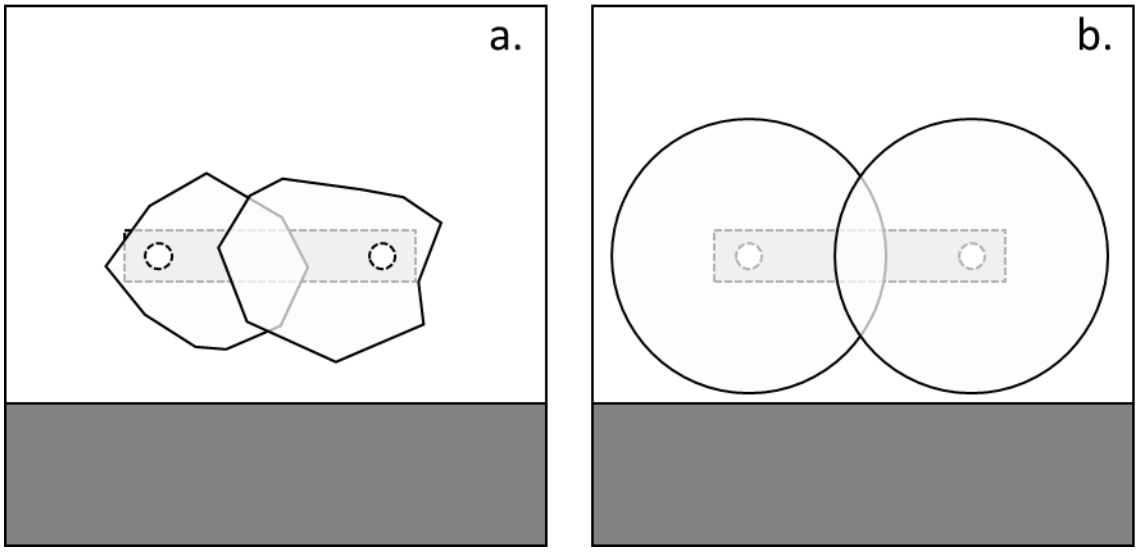


Figure 54: Overlapping with Components

These solutions tend to vibrate or move in an uncontrolled manner, or even jump, which may produce a false positive fitness value and inhibit evolution which is shown in Figure 55. A filter was implemented to solve this problem. Before evaluating the fitness of a solution, it goes through a filter which recognises false behaviour. The filter measures the rotation of the driving components and compares it with the mechanism's distance travelled. A solution is tagged as invalid if the driving component did not rotate; however,

the mechanism still moved forward at the same time. In this case, the solution was discarded by returning a fitness value of zero.

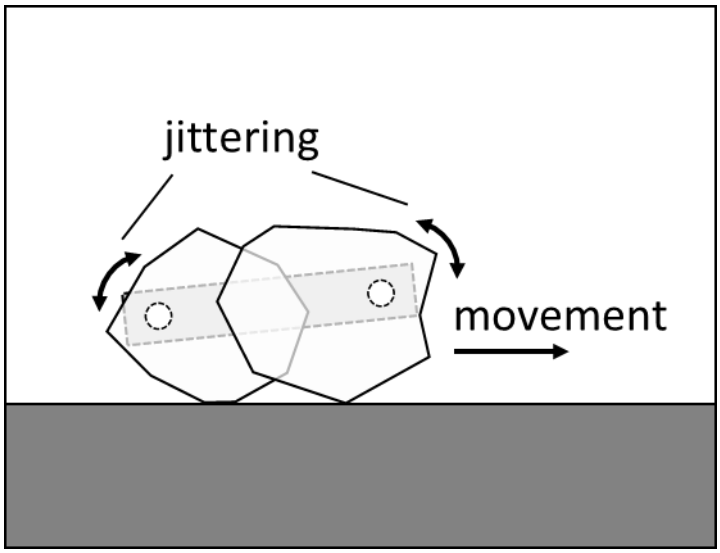


Figure 55: False Movement

These limitations need to be considered when simulating physics scenes with the current implementation of the simulator or addressed by improving the code.

5.5 Summary

This chapter described the framework for planar mechanism design to evolve mechanisms using an evolutionary computing approach. The framework enables the definition of components attached to a bearing plate with rotation joints. Furthermore, drive specifications can be set to introduce forces and movement into the system.

An evolutionary algorithm was used to evolve a mechanism consisting of two lever components with the objective of traversing different landscapes in a physics environment. The rectangle shape representation used in the previous chapters was extended to evolve mechanisms. Experiments were conducted to evaluate the evolutionary algorithm's capability to evolve solutions by comparing it to random sampling. Solutions were shown, and the limitations of the simulator investigated.

Two problems were encountered when using the simulator in combination with the generative system. Firstly, overlapping of initial solutions with the ground surface, and secondly, overlapping of lever components with each other. In some cases, the physics engine was not able to resolve the overlap, which led to undesired behaviours, such as vibration, jumping, and unstable simulation. The first problem was addressed by positioning the bearing plate further away from the ground and dropping it on the surface. The second problem was resolved by introducing a filter which discarded solutions with

undesired behaviour by assigning a fitness value of zero. The filter investigated each solution before evaluating its fitness. It measured the rotation of the driving components and compared them to the forward movement of the mechanism. When a forward movement was detected without rotation of the driving component, the solution was flagged as invalid.

The generative system was used to evolve mechanisms for five landscapes with different complexity. The results were compared to a random sampling. It was found that the performance of the evolutionary algorithm and random sampling significantly differ in favour of the evolutionary method. This indicates that the algorithm is capable of evolving solutions which overcome obstacles in the landscape, whereas random sampling gets stuck in obstacles.

The following chapter will extend the work by focusing on mechanisms with linkages.

6 Evolving Four-Bar Mechanisms

6.1 Introduction

In this chapter, the proposed framework is employed to evolve four-bar linkages, which involves the locomotion of the mechanisms, collisions and interactions in different environments. The focus is on evolving linkages together with an attached shape component capable of traversing a set of provided landscapes using an evolutionary algorithm.

Previous research in automated design often focused on mechanical linkages, such as four-bar mechanisms, intending to generate mechanisms which follow a specified path in space as closely as possible (Bose et al., 1997; Cabrera et al., 2002; Renner & Ekárt, 2003b; Roston & Sturges, 1996). More complex systems of this category were studied as well, for instance, six-bar linkages (Tsuge et al., 2016), or even assemblies of mechanisms (Ghassaei & Ming, 2015). Linkages were investigated from the perspective of their build, namely bars, and did not consider the shapes of components or collisions between them. Furthermore, properties such as torque; gravity; friction; or mass, are not considered.

The background section introduces the four-bar linkages and design objectives. It is followed by the method section, which explains the representation used and the experimental setup. The results discuss the influence of the attached shape component; the performance of the algorithm compared to a random sampling method; the influence of evolutionary operators; and the performance of the algorithm on problems with enhanced complexity.

6.2 Background

This section provides the definition of four-bar mechanisms and their use in generative design, according to existing literature. This is followed by a discussion regarding design objectives.

6.2.1 Four-Bar Mechanism

The area of four-bar linkage design was chosen as it focuses on multiple connected components and can create interesting locomotion. However, four-bar linkages are usually studied from the perspective of being assembled on a non-movable frame and designed to follow a specified trajectory, a coupler curve, in space as close as possible, with a tracer point. There is no consideration of collisions between components or with

other environmental obstacles. Four-bar linkages were explained in chapter 2.2.2. The current research considered more parameters, compared to classic four-bar mechanism problem.

In this work, the four-bar linkage design was extended to a rigid body model placed in a physics environment with gravity acting toward a ground surface on the components, shown in Figure 56. It allows the assembling of mechanisms with the ability to collide with their environment and to introduce additional parameters such as mass, friction, and restitution.

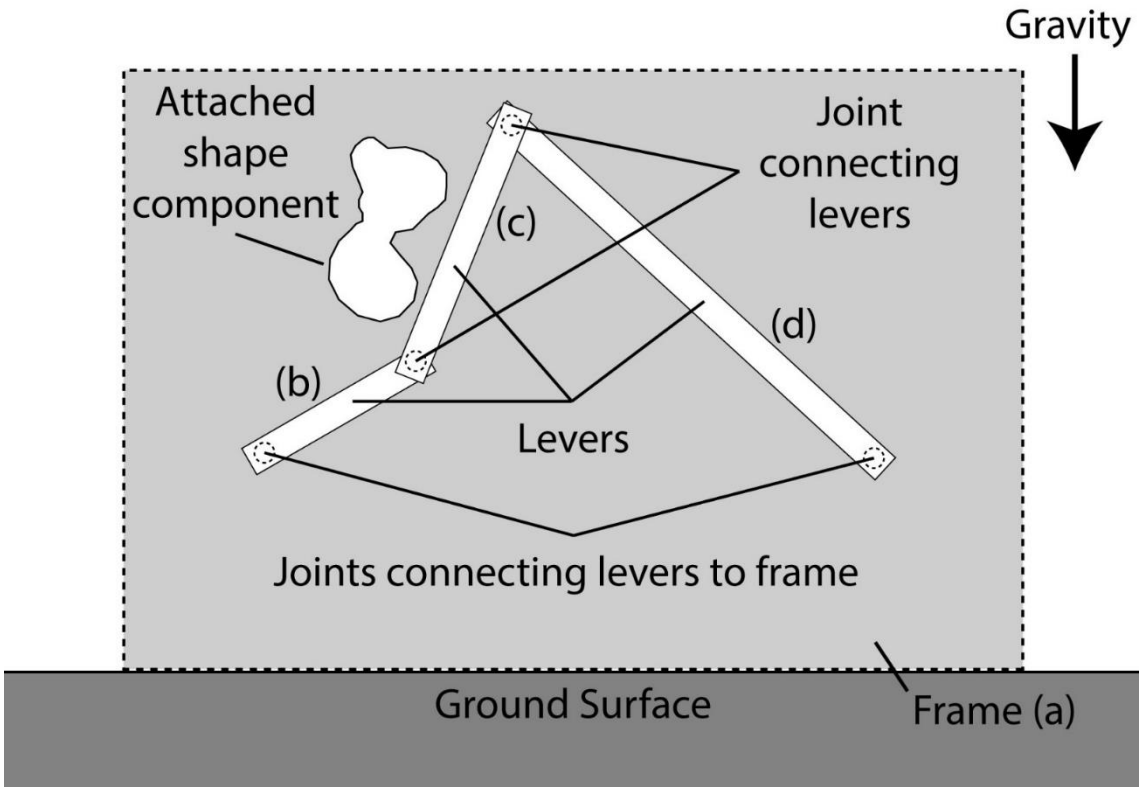


Figure 56: Four-bar Mechanism with Attached Shape Component

The mechanism is attached to a frame which is movable in space. It also has a lever component attached to the middle bar with an undefined shape that is moving with it. *Bar b* fulfils the role of a driving component with rotation speed and torque range, which allows the assembly to move. The Gravity makes it fall towards the ground. The positions of all rotation joints, length of the *bars b, c* and *d*, the shape and position of the attached lever component, and also the rotation speed and maximum torque applied to *bar b* are evolved using an evolutionary algorithm. During the movement, the components' shapes collide with the ground surface in different angles and motion patterns. The patterns result in a variety of mechanism behaviours, e.g. a forward or backward movement.

Figure 57 shows a movement pattern of a four-bar mechanism with two attached shape components.

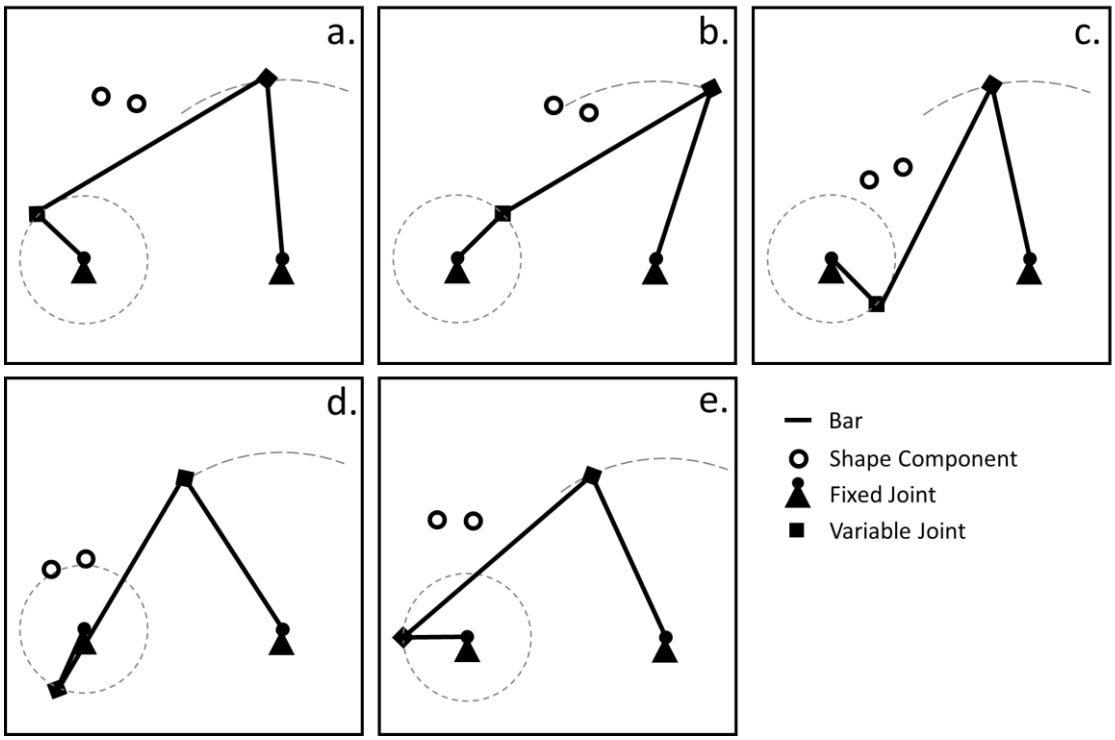


Figure 57: Four-Bar Mechanism Movement

The first bar makes a complete rotation around the joint while the last bar just makes a short movement to the right and left. The attached shape components stay in relative position to the middle bar.

6.2.2 Design Objective

There can be different objectives for four-bar mechanism design. The typical problem might be, e.g. to follow a trajectory or to produce a movement or torque characteristic at an output. Various approaches exist for defining design objectives and measuring the performance of potential design, e.g. as explained previously, in compliant mechanism design, the authors compare the generated path of the mechanism with a user-defined path (Sharma et al., 2008). The design problem is to transfer an input force and motion to an output force and motion, which is also applicable in planar mechanism design.

However, this research focusses on evolving general behaviour in environments, including different obstacles. The aim is to let the mechanism traverse a landscape as quickly as possible in a specified time. It provides benefits compared to objectives that involve following different trajectories, namely a set of different design problems can be defined by changing the obstacles in the environment, whilst being still able to compare the results to each other using a fixed simulation timespan.

In this scenario, the trajectory that makes a mechanism perform a forward movement is unknown, as it is dependent on the shape and configuration of the mechanism's components. Many trajectory-shape combinations can perform well in solving a problem. The evaluation of the distance travelled gives the generative system freedom in evolving more diverse solutions. Furthermore, it allows studying the evolutionary method in an environment, because the adaptation of components to the environment can be inspected by observation. In contrast, focusing on the forces and trajectories of interacting components may be more difficult, as forces are not visible. Lastly, the emphasis on the travel distance makes it easier to understand the problem.

6.3 Method

An evolutionary representation for the planar mechanism was developed in order to define the mapping procedure between genotype and phenotype. The representation defines the search space of the algorithm. There were three requirements defined for the evolutionary representation.

- Firstly, the representation needed to allow coverage of a large part of the search space. In the context of mechanism design, this means that the representation needs to be able to produce a large variety of solutions with distinctive characteristics.
- Secondly, the evolutionary representation needed to be compatible with evolutionary operators, such as mutation and recombination. For mutation, it is necessary, that changes in the genotype result in equally sized changes in the phenotype (Bentley, 1999). For recombination operators, it is necessary that the parent phenotypes are passing some characteristics of each of them to the child phenotype – otherwise, the recombination operator is simply providing mutation.
- The third requirement focused on the genotype to phenotype mapping. To be able to evaluate the candidate's fitness, invalid solutions should be either avoided, recognised as invalid and eliminated or resolved.

As a use-case, the focus was on a four-bar linkage with the attached shape component. Figure 58 shows an example of the mechanism in a virtual environment with a height limit. Four-bar mechanisms were chosen because they produce a large variety of trajectories depending on their bar configuration. In combination with a shaped component, which follows the trajectory, it can develop interesting and useful interactions with the ground surface or other obstacles.

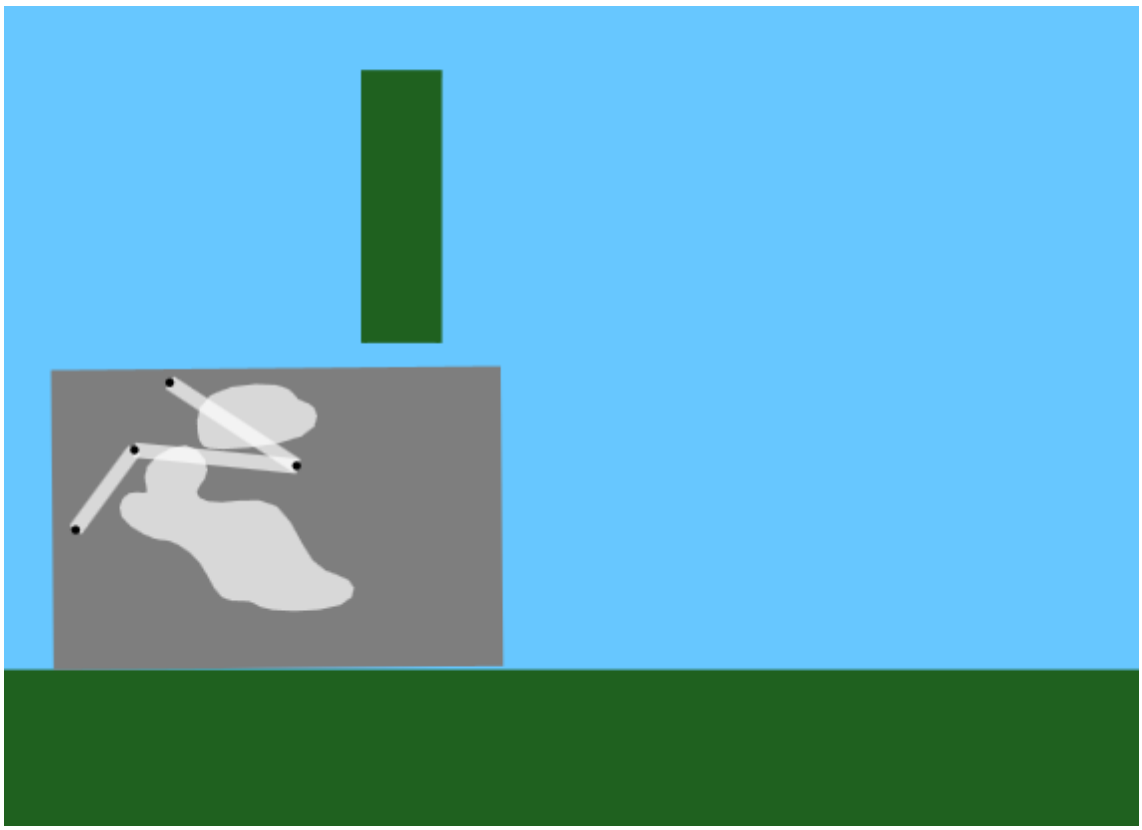


Figure 58: Mechanism in Virtual Environment

An evolutionary representation was developed to create four-bar mechanism solutions based on the findings gathered and described in the previous chapters. An evolutionary algorithm was used to evolve solutions with the capability to traverse a set of different landscapes.

6.3.1 Evolutionary Representation

The evolutionary representation for the mechanism consists of three parts, shown in Figure 59.

2 genes	8 genes	50 genes
Speed and torque	Position of joints	Shape description

Figure 59: Chromosome Representation

The chromosome is an array of real values between 0 and 1 rounded to seven digits of precision. The first part of the chromosome defines the speed and torque of *bar b*, the driving component, which rotates around the first joint. The second part defined the position of the four joints that subsequently defines the length of the connecting *bars b*, *c*, and *d*. The third part is the shape of the attached lever component using the

representation employed in previous chapters, with the centre point placed in the middle of *bar c*.

The first two chromosome values define the speed and torque of the driving component within a specified range of 15 to 45 rpm and 10 to 70 Nm. The next eight values are mapped into the joint position coordinates within the *frame a*. The remaining values define the shape and position of the attached lever component within the maximum and minimum boundaries set to 10 and 80 size units (pixels) in the problem file. A chromosome containing 60 parameters in total describes the mechanism.

6.3.2 Evolutionary Algorithm

The evolutionary algorithm is similar to the one used in the previous chapters with the following configuration: The population size was 40, and 4 children were produced in each generation. These parameters performed well through initial experimentation. Similar to chapter 5, the rectangle representation *R** was used with evolutionary setting *S1*.

6.3.3 Fitness Evaluation

The objective is to evolve a moving mechanism which is capable of traversing different landscapes. The algorithm evolves a four-bar linkage with a shape component attached to one bar. It is mounted on a bearing plate with the aim to traverse different landscapes and maximise the mechanism's position in the x-direction at the end of a specified timespan. The distance is measured from the middle point of the bearing plate in the first frame to the middle point of the bearing plate in the last frame. The extended representation can produce four-bar linkages, as opposed to the representation described in previous chapters. Furthermore, a new set of landscapes is employed.

6.3.4 Experiments

The mechanism was evolved within an environment, including a ground surface. Gravity was applied in the negative y-direction and set to 9.81 m/s^2. All components were given the same material parameters for density (1.0), friction (0.5) and restitution (0.6). Seven different environments were defined to investigate the generative system's abilities to produce solutions, as shown in Figure 60.

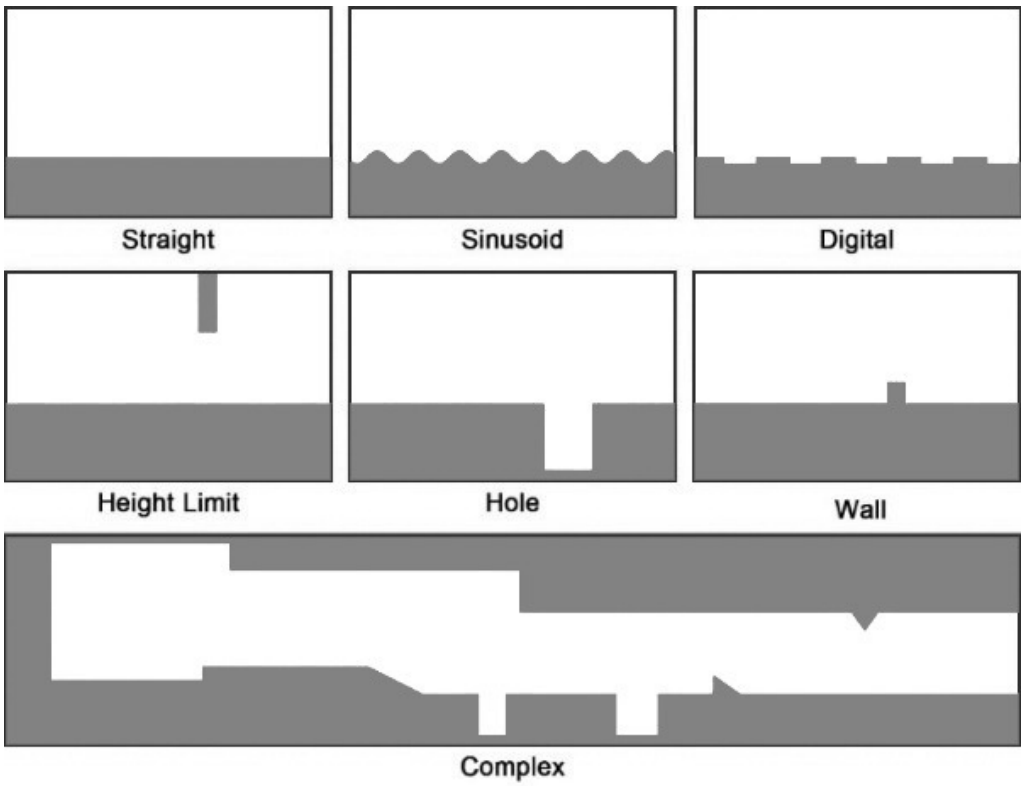


Figure 60: Environments

The figure shows a straight, a sinusoid shaped, and a digital shaped ground surface. Furthermore, three landscapes containing a height limitation; a hole; and a wall. The last landscape is complex; it includes many combined characteristics of the other environments.

Experiments were run 24 times on each of the environments for 1,200 frames, which equates to 10 seconds of simulation using a frame rate of 120 per second.

The complex environment was simulated for 3,600 frames which equates to 30 seconds of simulation - as the environment is changing over a longer path. The configuration was found to be an appropriate balance between outcome and simulation time. The driving component was allowed to have a rotation speed between 15 rpm to 60 rpm, and torque in a range from 10 Nm to 80 Nm. The algorithm stopped after 20,000 evaluations. The fitness of a candidate solution was evaluated by measuring the distance travelled by the mechanism through the environment at the end of the simulation period.

6.4 Results and Evaluation

Experiments were run to investigate the generative system's capabilities to evolve design solutions which efficiently traverse an environment.

- Firstly, the influence on the quality of the solution of evolving the attached shape component was analysed, by comparing the results of evolved four-bar linkages with an attached shape, to evolved linkages without attached a shape.
- Secondly, the generative system was compared to random sampling.
- Thirdly, the mutation and recombination operators were evaluated to identify their effect on a solution.
- Fourthly, the generative system's performance was evaluated on an environment with enhanced complexity.
- Finally, the simulator limitations which emerged throughout the experiments were discussed.

The Mann-Whitney U-Test was used for statistical analysis. A *p*-value of $p \leq 0.05$ indicates high confidence that distributions between two populations significantly differ. The *p*-value refers to the median distribution of best-performing solutions at the end of each run.

6.4.1 Performance Validation whilst using the Attached Shape

The impact of evolving an additional shape component attached to the four-bar linkage compared to evolving just the linkage alone was investigated. Figure 61 shows the results of six different environments.

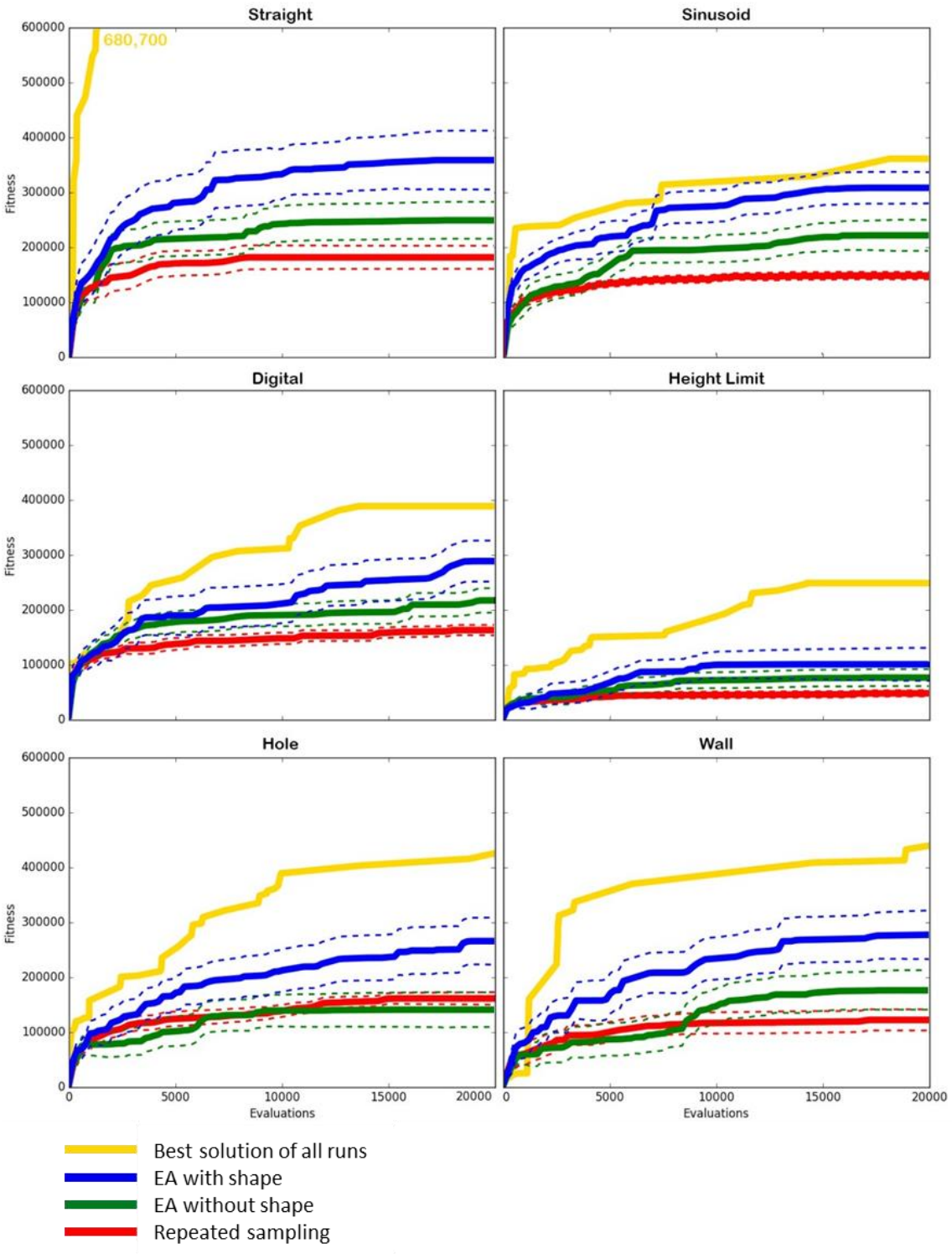


Figure 61: Median Fitness

All plots show the median fitness of 24 runs with ±95% bootstrapped confidence intervals: Evolutionary algorithm with shape (**blue**) vs without shape (**green**) vs random sampling with shape (**red**). They also present the best solution of all runs (**gold**). The mechanism's distance travelled determines its fitness.

The results show that evolving an additional shape component (**blue**), which is attached to the four-bar linkage, leads to better results compared to evolving just the four-bar

linkage (**green**). The mechanism with attached shape moves faster and further through the environments because the attached shape components provide an additional advantage. The results for the height limit environment are the only ones showing no significant difference. Figure 62 shows several well-performing solutions evolved in each environment.

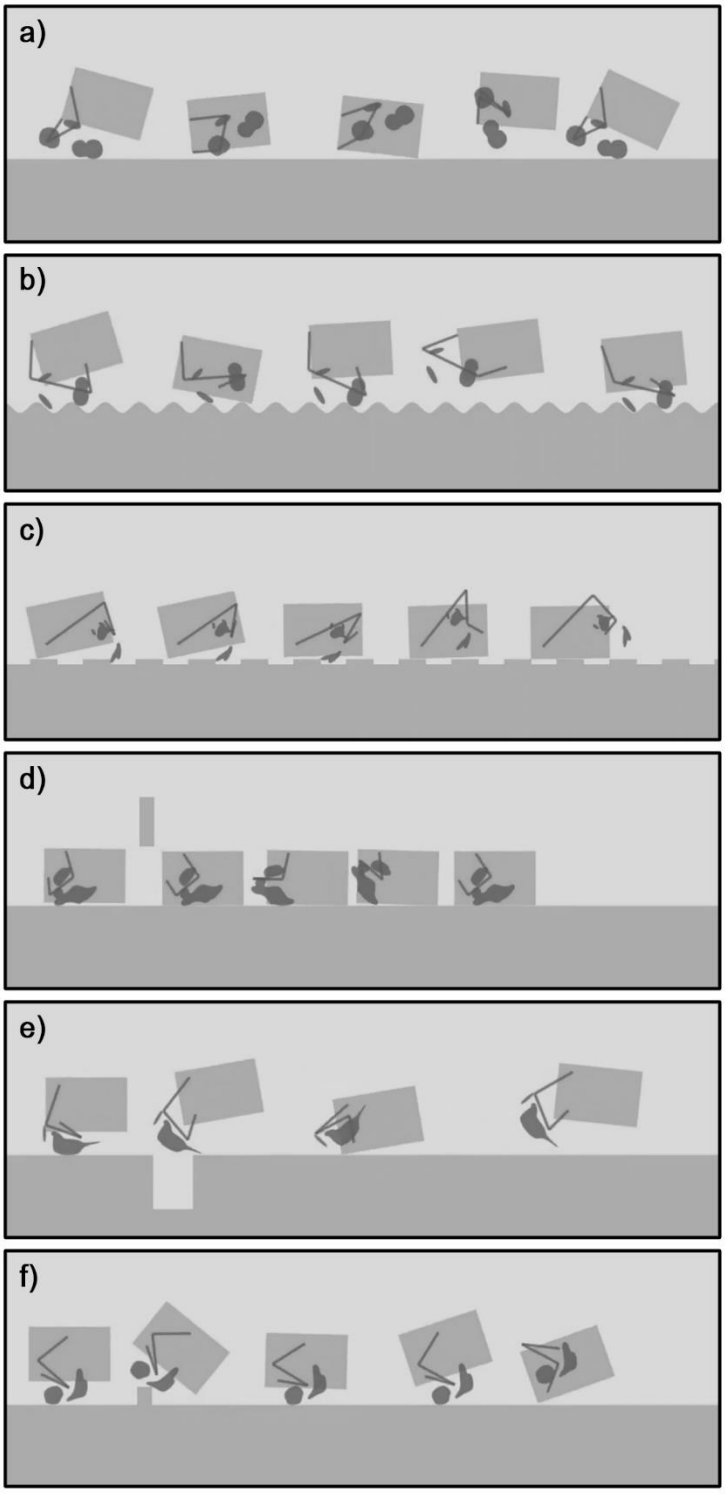


Figure 62: Evolved Solutions

Each image contains a few frames of a mechanism at different stages moving through a landscape. The mechanism in *a* evolved a movement pattern similar to a foot touching the ground rolling from the heel over the toe which makes the mechanism leap. A similar movement pattern can be seen in *d* and *e*. In *d*, due to the hight constraint, the movement pattern is slightly different and keeps the mechanism closer to the ground. In *e*, the pattern and leaping are similar to *a*, however, it evolved a shape with a spike helping it to overcome the hole. In *c*, the mechanism evolved a hook shape and a patter which digs the hook between the niches in the landscape to pull it forward. The mechanism shown in *f* evolved a movement pattern and shape which lifts the front part of the mechanism and leaps it forward to overcome the wall.

6.4.2 Performance Validation using Random Sampling

The evolutionary algorithm was compared to random sampling to validate its ability to evolve solutions. In Figure 61, comparing random sampling (**red**) with the evolutionary algorithm (**blue**) shows that the latter significantly outperforms random sampling. It finds more solutions and has a sharper increase in their quality. Random sampling ends in a flat line and is unable to improve further, whereas the evolutionary algorithm continues to find better-performing solutions. The results for the limited height environment show no significant difference.

6.4.3 Investigation of Mutation and Recombination Operators

The evolutionary algorithm's genetic operators were investigated to evaluate their contribution to finding better performing solutions. Firstly, the algorithm was tested by using the mutation operator without recombination. Figure 63 shows the evolution of a mechanism by applying only the mutation, starting with *a* and ending with *l*.

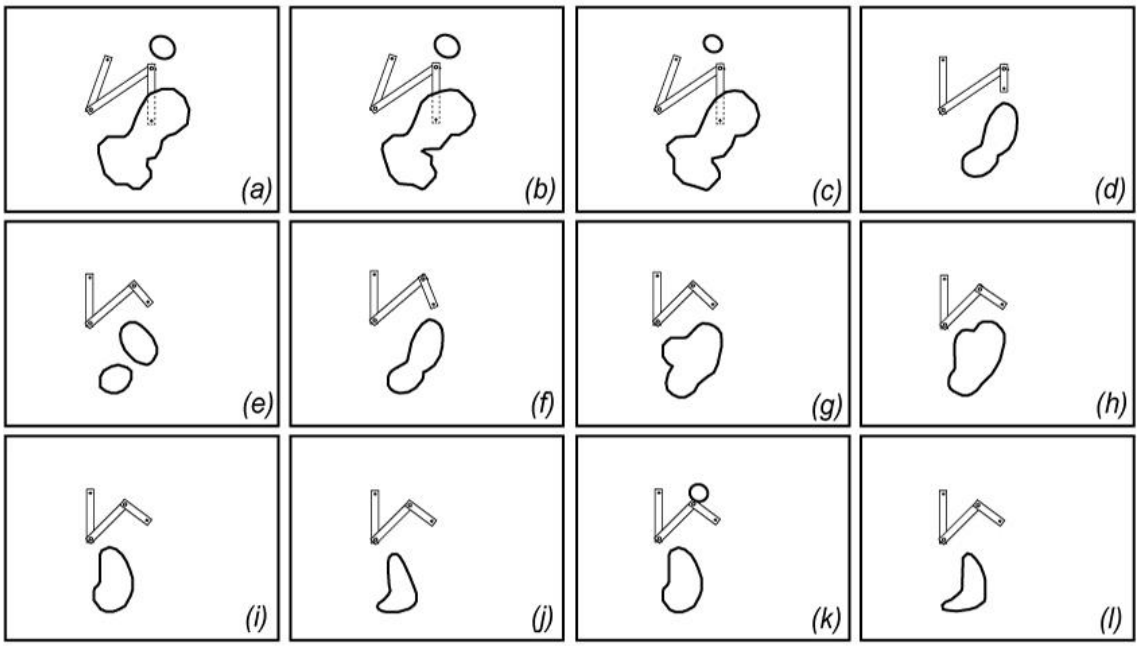


Figure 63: Mutation Operator

Firstly, the figure shows that the mutation operator applies small changes to the bar length and position of the joint, also to the position and shape of the attached component. The component can separate into multiple shapes, all moving relative to the middle bar. Visually, the operator does not produce too a large disruption, which is important for the mutation operator to improve a solution efficiently.

Secondly, the recombination operator was investigated by evolving a solution using recombination and further, one Gaussian based mutation operation. The mutation was applied to avoid premature convergence, which was found to appear when using recombination only. Figure 64 shows two parents, and as an example, four potential resulting children.

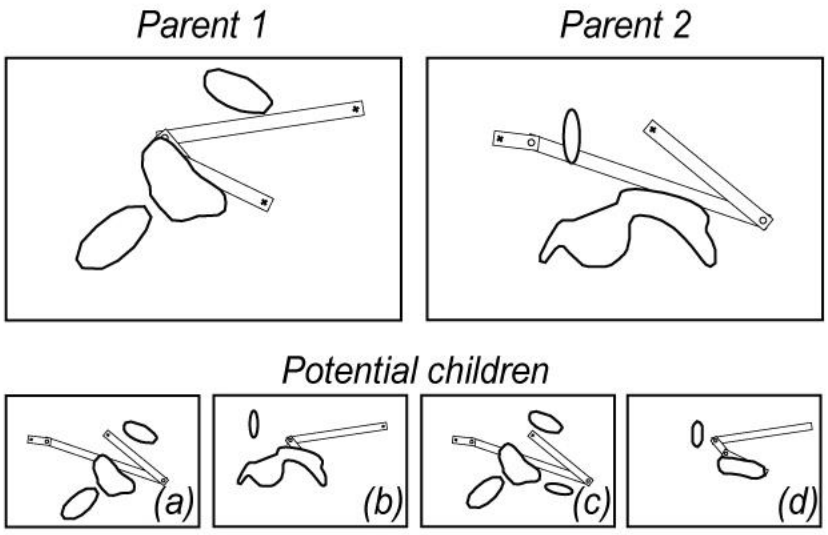


Figure 64: Recombination Operator

The figure shows that all of the potential children contain some features of the parents. E.g. *child a* has the linkage of *parent 2* and the shape components of *parent 1*. *Child b* has the shape components of *parent 2* and partly the linkage of *parent 1*. *Child c* is similar to *child a* but includes some new shape fragments. *Child d* has the linkage of *parent 1* and some shape characteristics of *parent 2*. However, it seems that the recombination operator introduces a disruption which may be too large. It does not take account of the grouping of genes sequences that describe single rectangles. The rectangle information is divided and partly transferred to the new generation, which leads to large changes and new shape fragments.

Figure 65 shows a comparison between using the recombination operator with one mutation operation, and using the mutation operator only, for the hole environment to determine the contribution of the recombination operation.

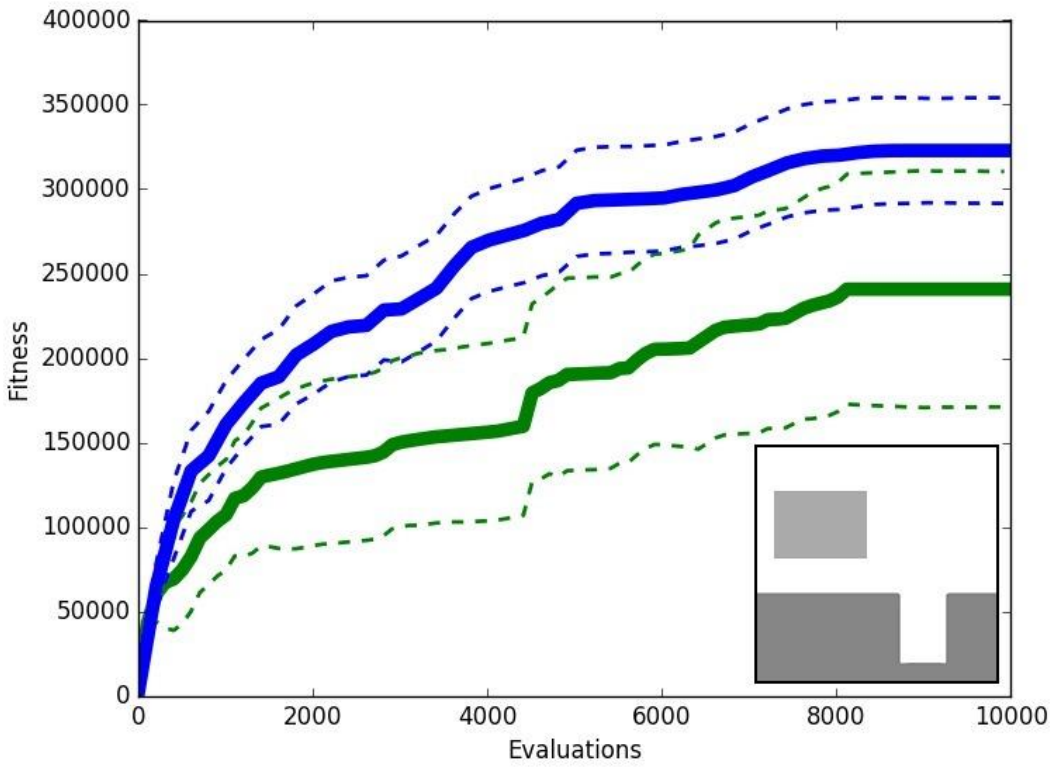


Figure 65: Recombination with Mutation (blue) vs Mutation-only (green)

The results show that using the recombination with the mutation operator leads to significantly better results when compared to using the mutation operator only. This means that the recombination operator helps to escape from local optima and navigate into different regions of the search space.

6.4.4 Performance on Problems with Enhanced Complexity

The complexity of the problem was increased by providing a more challenging path, including multiple obstacles. Figure 66 shows a solution evolved in a complex environment which traverses the landscape at different positions.

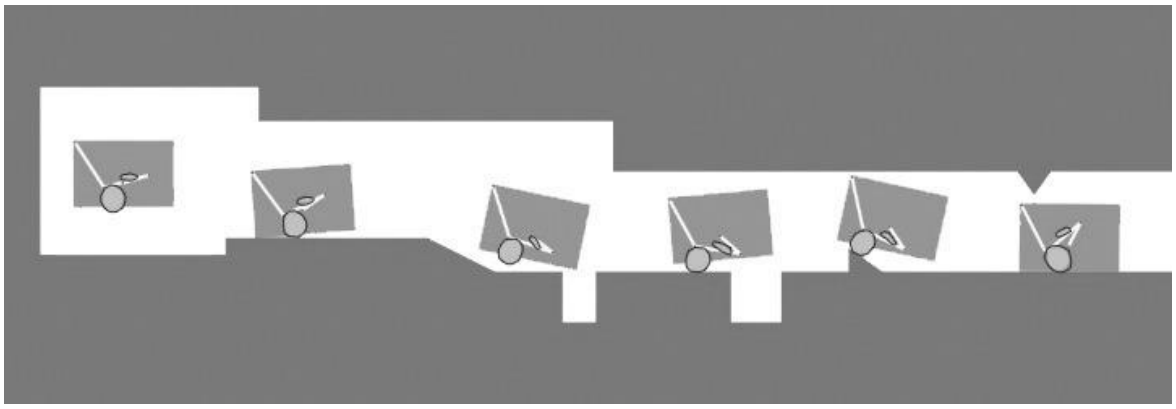


Figure 66: Moving through a Complex Environment

The evolutionary algorithm was able to evolve solutions, which succeeded to pass all the obstacles. Figure 67 shows the median fitness of 24 runs with ±95% bootstrapped confidence intervals (**blue**) and the best performing solution (**green**). The environment is added on the y-axis, showing the mean position of all mechanism and the position of the best mechanism in the environment, as well as their fitness.

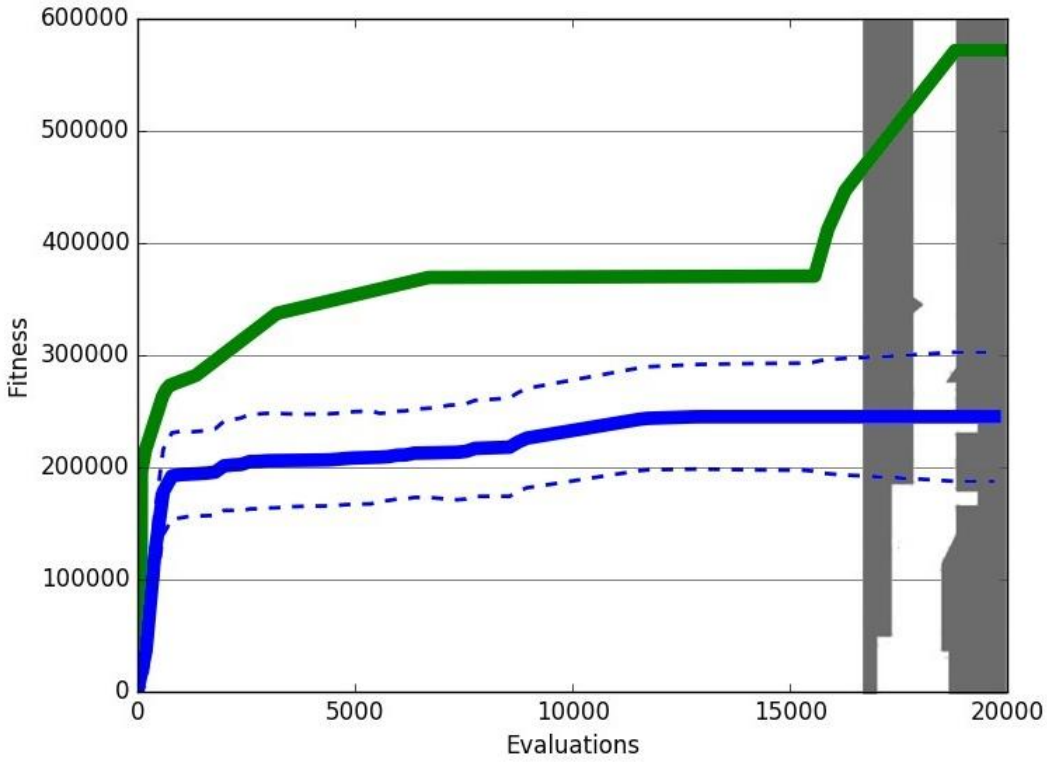


Figure 67: Evolution in a Complex Environment

The figure shows that the evolutionary algorithm can also evolve mechanisms in a complex environment. Comparing the graph to the environment path shows that most of the solutions get stuck at some obstacles for some time. However, the evolutionary algorithm manages to find solutions which bypass these in the long-run, allowing some to travel through the complete environment and even to further improve their fitness after passing the obstacles.

6.4.5 Simulator Limitations

The simulator was capable of resolving the physics scenarios provided by the generative system. Nevertheless, because of the nature of the physics engine, it was found that simulation errors may still occur due to overlapping shapes, jittering and clipping, which are all potentially exploitable by the evolutionary algorithm. These errors were addressed with the previously discussed filter, which detects if a mechanism is moving forward, e.g. due to jittering, without rotation of the driving component. Those unrealistic solutions were filtered and rated with a fitness value of zero.

Through experimentation, it was found that using a simulation frame rate of 120 frames per second; as well as increasing the position and velocity iterations, which are Box2D internal settings, from the default of 8 velocity and 3 position iterations to 16 and 6 iterations per frame, reduced simulation errors. These would be otherwise exploited by the evolutionary algorithm when simulating linkages

The position and iteration count controls how many times the constraint solver sweeps over all the contacts and joints in the virtual environment. Increasing the iteration always yields a better simulation (“Box2D: Overview,” n.d.).

6.5 Summary

This chapter presented a method which enables the study of evolutionary algorithms for evolving planar mechanisms. As a use case, the focus was on the ability to traverse different environments by evolving freely movable four-bar linkages with an additional attached shape component.

The generative system, including simulator, was validated by showing its capability to produce and evaluate design solutions. The results indicate that evolving an additional shape attached to the mechanism led to better solution quality. Furthermore, a comparison of the evolutionary algorithm to random sampling showed that the algorithm evolves solutions, rather than randomly selecting them.

The contributions of the genetic operators were investigated, and it was found that both operators work well and that the recombination operator supports finding better performing solutions. The operator enables the algorithm to escape from local optima and navigate to different regions of the search space. However, it was found through visual analysis that the recombination operator introduces, in some cases, a large disruption due to breaking valuable gene sequences. This may slow down the evolution, which can be addressed in future by considering the chromosome encoding, e.g. not breaking the groups of rectangles within it, when using recombination. Moreover, the algorithm performs well when increasing the problem complexity using a landscape with multiple obstacles and enhanced complexity.

Overall, the results show that the system is capable of increasing the fitness of candidate solutions and producing interesting mechanisms, which perform the desired behaviour. The shape and configuration adapted to the environment and its obstacles, able to reach different areas of the search space.

7 Conclusion and Future Directions

This chapter provides a summary of the work conducted, followed by a list of contributions and a discussion. Lastly, it explains the potential future work.

7.1 Summary of Work Conducted

The research was divided into five chapters. In Chapter 2, a literature review was conducted, exploring conceptual design, planar mechanisms, evolutionary computing and representations, and furthermore, the area of shape representations, generative design tools, and evolving mechanisms. The literature showed that there is a shortage of research targeting the early conceptual design stages, additionally, that tools suggesting a broader range of solutions might be beneficial for engineers to reduce their bias and workload.

It was found that there is no specific area concentrating upon the generative design of planar mechanisms, which considers the shape and interaction of components, including attributes such as mass, friction, and restitution. The work in mechanical optimisation to date focused on the movement of components and did not investigate linked mechanical systems, including collisions in a dynamic environment. Considering upon these allows a closer approximation of real-world mechanisms and contribution towards innovative generative design system.

The field of engineering optimisation was reviewed; every area has its unique way to describe its problem domain. However, these often focused on the control patterns and behaviour of mechanisms.

Looking at problems solved with evolutionary computing showed that an indirect encoding is most suitable for shape representations. This work provides a way to define complex shapes with a low number of genes. A method to evaluate shape representations able to create target shapes was identified, which is a computationally inexpensive process to develop and evaluate representations' ability to be applied within evolutionary algorithms to create shapes for a specific problem domain.

In Chapter 3, the focus was placed on the design of a shape representation capable of generating mechanical shapes, guided by an evolutionary algorithm using the target shape matching technique instead of working directly in a physics environment. Several representations were developed and evaluated, which gave insight into the underlying mechanisms of the evolutionary process and to design and improve the representations in

a fast and systematic way, capable of producing relevant shapes. The results showed that the implemented rectangle-based representation worked best for reproducing mechanical shapes in an evolutionary computing context.

In Chapter 4, a simulator was employed, capable of simulating a virtual world based on a physics engine, which was necessary to create and compute design scenarios considering interaction and collisions, including attributes such as mass, friction, and restitution. Usually, these types of scenarios are simulated with a computer using multi-body dynamics solvers, which produce a close approximation of reality. However, they are computationally expensive and are often the bottleneck in an evolutionary design system, as those systems require a large number of evaluations to be conducted.

A simulator, based on the game physics engine Box2D, was developed to address this issue. It is capable of resolving collisions and of performing at a fast pace, recording every simulation frame in an output file, providing visual feedback, and was used as the basis for the performance evaluation of potential design solutions. It was validated using unit and acceptance tests and tested together with the generative system.

Experiments were conducted with the focus on developing an evolutionary representation for single shaped components. Three representations were designed, based on the initial well-performing representation for shape matching. Their performances were compared, according to their ability to produce physics components capable of traversing a virtual landscape, with two different evolutionary settings. The evolutionary settings applied different size changes to the genotype. It was found that one of the three tested representations performed well with both evolutionary settings and evolved solutions within the physics environment that fulfilled the design aim.

In Chapter 5, a framework was proposed based on real-world mechanisms focusing on lever mechanisms and linkages. A scripting language was designed that allows the specification of various design problems. An interpreter translates a problem file into a physics scenario for the simulator, which computes the locomotion. The results are used for performance evaluation of a potential design. This approach made the real-world problem understandable for the computer and enabled applying evolutionary computing techniques to evolving design solutions. The ability to evolve mechanisms consisting of multiple components, attached to a bearing plate with joints, was evaluated. Through comparison to random sampling, the results showed that the algorithm is capable of evolving mechanical design solutions, successful in traversing several landscapes with different complexity. Furthermore, the limitations of the simulator were discussed.

In Chapter 6, the simulator and framework were employed to evolve mechanical linkages. They were used to evolve four-bar mechanisms with the same design aim as previously, although, traversing different landscapes. In previous research, the area of four-bar mechanisms was mostly focusing on evolving the kinematic behaviour of mechanisms to follow a specific path without considering further attributes such as mass, friction, restitution, and gravity. The contribution of this research was to consider those parameters, and further, including collisions between components as well. The results have shown that the framework and generative system performed well in finding design solutions even for complex problems.

The overall outcomes of this work showed that evolutionary algorithms can be successfully applied to evolve mechanisms and that the chosen approach performed well in experiments. The developed representation performs in a satisfactory manner with different evolutionary operators and is capable of producing well-performing results irrespective to the size of applied mutations. It showed that it could produce mechanical shapes, also within a physics environment. Furthermore, it could evolve mechanisms consisting of multiple components, including linkages which fulfilled the design objective.

The implemented simulator was validated and tested throughout the experiments and performed successfully. Its limitations were investigated, and issues were resolved. However, further work is necessary to turn this research into a usable application for the industry to support designers in the preliminary stage of mechanisms design. Especially, the design objective definition needs to be developed further as this work used a simplified objective of traversing different landscapes. It was appropriate for conducting experiments and analysing comparable results. However, an industrial application would need a more practical way to specify design aims.

This work provides an entry point for evolutionary computing researchers and a stepping stone towards a generative design system for planar mechanism design, capable of providing engineers with prototypes for specific design tasks. It contributes towards an understanding of generative design systems, focused on industrial applications.

7.2 Summary of Contribution

The following research questions were addressed:

RQ1 (Chapter 3): Which evolutionary representation can be used to efficiently represent and evolve the shape of planar mechanical components?

The first question was addressed by implementing an experimental tool and method to evolve target shapes. Different shape representations were developed and compared in order to evaluate their capability to produce defined target shapes of the problem domain. Target shapes were taken from an automotive closure system and generated with the experimental tool. This approach provided insight into the working principles of the representation and evolutionary algorithm, and also into the representation's search space coverage, which helped to design and refine the representation. A rectangle-based representation was developed and evaluated, which performed well compared to other tested approaches.

RQ2 (Chapter 4): Which evolutionary representation and evolutionary operators can be efficiently used to represent and evolve mechanical components in a physics environment?

The second question was addressed by implementing a simulator capable of computing the locomotion of mechanical components and linking it to a generative design system capable of evolving design solutions. The initially designed, best-performing shape representation was embedded in the application using physics simulation. It was compared to two similar representations with minor changes, with the purpose of improving the performance. Different evolutionary settings were tested. The experiments investigated the capability to evolve the shape of a component that adapts to its environment, in order to traverse several landscapes with different complexity. Furthermore, the simulator was validated with unit and acceptance tests. One outstanding rectangle-based representation performed well with different evolutionary operators, applying larger and smaller mutations.

RQ3 (Chapter 5): To what extent are the evolutionary representation and evolutionary operators able to evolve mechanisms consisting of multiple components with the aim of traversing different landscapes?

The third question was answered by providing a description of the problem domain and creating a framework for planar mechanism design. A scripting language was developed that provides a way to define mechanical problems. The simulator and generative system were tested and evaluated with a set of landscapes, which were defined using the scripting language. An evolutionary algorithm was employed to evolve mechanisms with multiple components, joints, and actuators attached to a bearing plate. The framework was validated through experiments. Mechanisms were evolved for different problems, and the

experiments showed that the algorithm performed evolution by outperforming random sampling.

RQ4 (Chapter 6): To what extent are the evolutionary representation and evolutionary operators able to evolve four-bar mechanisms that are capable of efficiently traversing a landscape?

The last question was addressed by applying the framework and simulator in the domain of four-bar linkages. The evolutionary operators were analysed in depth. Furthermore, the evolutionary algorithm was investigated, and the framework additionally validated by showing its capability to evolve a variation of solutions for a set of problems with different complexity. Furthermore, it was found that both evolutionary operators, mutation and recombination, contributed towards finding better solutions.

This research contributes to knowledge by providing a method, including framework, evolutionary representation, and evaluation using a simulator, to evolve planar mechanisms with evolutionary computing techniques. The generative system was tested and validated. In future, the findings may lead to innovative generative design applications. They can facilitate further research and initiate new applications in design automation in order to increase the efficiency of the early mechanism design stage in the industry context.

7.3 Discussion

In this section, the research is discussed. It explains challenges encountered during this research, summarises the limitations of the simulator and the generative design system, and presents other implementations which were tested but not included in this work. Furthermore, it discusses the limitations of experiments and provides ideas for other potential approaches which could have been employed to address the topic.

7.3.1 Challenging Issues

This research encountered several challenges and issues which needed to be addressed. The first was the choice of the programming language to implement the necessary software. Different languages were tested, such as Python, Java, C++, and JavaScript, which all provided implementations of Box2D. However, these had some limitations, mostly a time needed to implement software, for instance, creating user interfaces or debugging code. Subsequently, C# was identified as an appropriate choice. The implementations progressed quicker, due to already existing C# skills. These still needed to be widened to produce an extendable piece of software with a modular architecture.

C# provided a large toolset that allowed designing the user interface and generative system, necessary for conducting the experiments.

Retrospectively the architecture could have been implemented on a server-side basis. Running the software on a server would have allowed conducting experiments on a computer cluster that may have shortened the time of development, and could allow a web-based interface to be published. However, the work had different priorities, and it was found not relevant. With some adjustments, the current architecture could allow extending the software to run on server-side in future.

The experimental results had a size of over 100GB, which could not be analysed by hand. Tools and scripts were needed instead. For that reason, several scripts were developed using Python, R, and Octave; each took additional time to objective learn.

A further challenging task was to find a general design in mechanical design. It was found difficult to produce experiments using a specific design case. Instead, general cases were selected as they provide enough freedom for an evolutionary algorithm to evolve a variety of solutions. It took time to realise that measuring the travel distance was a suitable objective general enough to be used for validation, especially because it was commonly used in other fields but not used in mechanical design.

Another challenge was the extensive experimental runtime. Although a single experimental run did not take long, many iterations of experiments resulted in over 5,000 hours of computational runtime. Hardware was needed to shorten the time. A setup of five computers was configured to solve that issue. These gave feedback on the experimental state and progress via Email, which increased productivity.

7.3.2 Simulator

In general, the simulator, based on the physics engine Box2D, performed well and was a suitable choice. The software was implemented in a way that allows exchanging Box2D with other physics engines with a low programming effort, which may be interesting for future research. It also provides a way to exchange it with a different type of simulation, e.g. particle simulation to optimise designs for aerodynamics. In general, it was found that the simulator resolved movements and collisions accurately. However, it needs testing and fine-tuning whilst applied to different kinds of problems such as evolving multi-component mechanisms, linkages, or problems beyond this research, to reduce simulation errors. As shown, different frame rate settings were used for each problem type. Setting the frame rate requires performing several simulations and investigation of

the results for visible errors. It is not possible to recommend a specific setting as it is highly dependent on the design problem and employed representation.

The simulator works in combination with an evolutionary algorithm which performs a large number of evaluations in a short time. It means that it eventually exploits implementation errors and instabilities of the physics engine, such as jittering and jumping. A filter was implemented to address these. However, it does not fix the implementations itself. It is a workaround that spots errors and acts accordingly upon them. Otherwise, changes in the physics engine code may require additional effort.

7.3.3 Generative System

The generative system was based on an evolutionary algorithm. The algorithm included the basic principles of computational evolution and was able to evolve solutions. The implemented algorithm can be considered as state of the art and was tested with different configurations. However, the focus of this research was not on a comparison of different algorithms, but rather, on the representation, as it has a larger impact on the performance of the algorithm, instead of changing the algorithm's routine or investigating its parameters, such as mutation or recombination rate.

The analysis of specific industrial design problems and solving them with the generative system was out of the scope of this work. The system was designed specifically with the travel distance objective in mind. A specific industrial design case would have required additional implementations.

7.3.4 Experiments

The software was designed iteratively and went through numerous iterations, in which implementation errors were corrected, representations developed, and evolutionary configurations tested. In total, there were 17 different representations developed, four different mutation operators, and four different recombination operators. Many of these did not perform well and were not used in experiments.

Furthermore, several scripts in Python, R, and Octave were developed that allowed analysing results and plotting figures.

Experiments were conducted on multiple computers for a total computation time of over 5,000 hours which equals to nearly 210 days of computation on a single computer. A setup of five computers was used, which lowered the time to produce results.

7.3.5 Locomotion-based fitness function

This work shows how to use an Evolutionary Computing approach to evolve mechanical systems. A design objective of traversing landscapes was chosen to produce comparable results throughout this work. It was found general enough to give the algorithm design freedom. However, for industrial application objectives need to be defined in a very specific way which requires in-depth research of methods to define design problems in different engineering domains in future. Instead of measuring the travel distance, another approach to define a design objective could be to define a path instead. The fitness function would evaluate how close the centre of the machine follows the path.

This work showed that design solutions can be evolved using a locomotion-based fitness function. This can be further developed into a mechanical design application for evolving components during the early design stage, e.g. as a CAD support tool. One would need to exchange the objective of traversing a landscape with e.g. rotating a lever component in a certain way. It would allow evolving mechanical components able to rotate another component within a system without human intervention.

7.3.6 Different Approaches

The literature review showed that there are many approaches to implement generative design systems. In this work, evolutionary computing was used, which has shown to perform well. Evolutionary computing is usually a blind approach; often, it does not include knowledge about the design problem or past problems. However, there are other methods such as machine learning, and neural networks, which have the ability to identify patterns and correlate them to the fitness of the design, which may be an advantage. Evolutionary computing was selected due to the priority being given to the representation, as there was none available and the working principles are easier to observe than other approaches. However, the representation may also be usable in the variety of fields, and other researchers and practitioners may try to apply different methods using the finding from this work.

Another interesting approach would be to focus on evolving design and behaviour at the same time, such as in artificial life. Using the proposed representation would allow evolving far more complex designs compared to those used before, which focused solely on the behaviour.

The implemented software can also be extended by adding another physics simulator such as particle simulation to evolve, e.g. for aerodynamics or compliant mechanisms.

7.3.7 Potential Improvements

The way the genotype maps to the phenotype space can be further improved. Currently, genes can take values between zero and one with seven digits of precision. However, the coordinates in the phenotype space are many orders of magnitude less (240 pixels). The disproportion was not spotted until all experiments were conducted. As all experiments used the same configuration, it should not have any influence on the results as such. A lower precision in the genotype representation may increase the speed of finding better solutions.

7.4 Future Work

Mechanism design has a large number of different application areas. These range from industrial applications, such as production systems, to design conveyor systems, through sorting machines, to automotive design, e.g. for wiper mechanisms, or mechanical redundant devices such as closure systems. The proposed framework can be further developed with a focus on other design problems and types of mechanisms. Different mechanical components can be implemented, such as translation joints, springs, and mechanical stops. It could lead to mechanisms with more complex behaviour and including additional automatisation, such as spring design and optimisation.

Also, the actuator control pattern could be evolved to produce more complex input movement. It would open the door to evolving actuator behaviour, and the shape and configuration of components simultaneously, which is closer to biological evolution.

Another focus could be on the design interface for engineers, to allow them to specify a general design problem graphically. In this work, it is done by using input files containing the bespoke scripting language. In future, the generative system can be a part of a design tool. It may support the conceptual design process for mechanism design, able to propose design solutions to engineers for general, or even specific problems related to planar mechanisms. It could be run as an addition to traditional CAD. Furthermore, the ability to extend the generative system to 3D mechanism design should be investigated which would make it even more suitable to be used in a CAD scenario.

The software is designed in a modular way. It allows to exchange the evolutionary algorithm with other search heuristics, to add other representations, and also the physics engine can be exchanged with another, e.g. 3D physics engine.

The representation and evolutionary algorithm could be used in a different context such as evolving aerodynamic shapes, such as turbine blades, using particle physics simulation.

This work could be used to implement a design system for a specific engineering domain and test it with engineering designers. Solutions evolved by the algorithm could be compared to human designs to investigate if innovative solutions could be identified or at least solutions of which no human thought of.

7.5 Conclusion

Generative design aims to produce a set of solutions, which can be analysed and selected by engineers for further development, in order to increase the output of the preliminary design stage. This work addressed the area of generative design and presented a method of evolving planar mechanisms for the preliminary stage of mechanical design, using evolutionary computing techniques, which has not been done before in this specific domain. The main contributions of this work are:

- A software tool to run experiments, visualise, and record the process of evolving shapes for mechanical components
- Implementation of bespoke software to run experiments; including visualising, simulating and evolving design solutions
- Development and evaluation of a number of evolutionary representations for shapes and mechanisms
- The validation of a framework through the evolution of mechanisms
- Empirical data, and analysis of the experimental results focusing on evolving planar mechanisms

In future, this work has the potential to be developed into an industry tool for assisting engineers in the early stage of planar mechanism design.

Bibliography

Alexandersen, J., Sigmund, O., & Aage, N. (2016). Large scale three-dimensional topology optimisation of heat sinks cooled by natural convection. *International Journal of Heat and Mass Transfer*, *100*, 876–891. https://doi.org/10.1016/j.ijheatmasstransfer.2016.05.013

Ambrose, E., Ma, W., Hubicki, C., & Ames, A. D. (2017). Toward Benchmarking Locomotion Economy Across Design Configurations on the Modular Robot : AMBER-3M. *IEEE Conference on Control Technology and Applications*, 1270–1276.

Ames, A. D., Tabuada, P., Jones, A., Ma, W. L., Rungger, M., Schürmann, B., … Grizzle, J. W. (2017). First steps toward formal controller synthesis for bipedal robots with experimental implementation. *Nonlinear Analysis: Hybrid Systems*, *25*, 155–173. https://doi.org/10.1016/j.nahs.2017.01.002

angusj. (2010). Clipper - Polygon and line clipping and offsetting library (C++, C#, Delphi). Retrieved June 15, 2019, from https://sourceforge.net/projects/polyclipping/

Arias-Montaño, A., Coello Coello, C. A., & Mezura-Montes, E. (2011). Evolutionary Algorithms Applied to Multi-Objective Aerodynamic Shape Optimization. *Computational Optimization, Methods and Algorithms: Studies in Computational Intelligence, Volume 356*, 211–240. https://doi.org/10.1007/978-3-642-20859-1_10

Artobolevsky, I. I. (1975). Mechanisms in Modern Engineering Design. Moscow: Mir Publishers.

Baron, P., Fisher, R., Tuson, A., Mill, F., & Sherlock, A. (1999). A voxel-based representation for evolutionary shape optimization. *Ai Edam*, *13*(03), 145–156. https://doi.org/10.1017/S0890060499133031

Belter, D., & Walas, K. (2014). A Compact Walking Robot – Flexible Research and Development Platform. *Advances in Intelligent Systems and Computing*, *267*, 343–352. https://doi.org/10.1007/978-3-319-05353-0

Bendsøe, M. P., & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and*

Engineering, *71*(2), 197–224. https://doi.org/10.1016/0045-7825(88)90086-2

Bentley, P. (1999). *Evolutionary Design by Computers*. Morgan Kaufman Publishers.

Bentley, P., & Kumar, S. (1999). Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *GECCO'99 Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1* (pp. 35–43). Orlando, Florida: Morgan Kaufmann Publishers Inc. Retrieved from http://discovery.ucl.ac.uk/171657/

Bentley, P., & Wakefield, J. (1997). Conceptual evolutionary design by a genetic algorithm. *Engineering Design and Automation*, 1–13. Retrieved from http://www.cs.ucl.ac.uk/staff/ucacpjb/BEWAJ2.pdf

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, *8*(2), 239–287. https://doi.org/10.1007/s11047-008-9098-4

Bose, A., Gini, M., & Riley, D. (1997). A case-based approach to planar linkage design. *Artificial Intelligence in Engineering*, *11*(2), 107–119.

Box2D: Overview. (n.d.). Retrieved May 10, 2020, from https://box2d.org/documentation/index.html

BoxCar2D. (2015). Retrieved February 26, 2016, from http://boxcar2d.com/),

Byung Gun Choi, & Bo Suk Yang. (2000). Optimum Shape Design of Rotor Shafts Using Genetic Algorithm. *Journal of Vibration and Control*, *6*(2), 207–222. https://doi.org/10.1177/107754630000600203

Cabrera, J. a., Simon, a., & Prado, M. (2002). Optimal synthesis of mechanisms with genetic algorithms. *Mechanism and Machine Theory*, *37*(10), 1165–1177. https://doi.org/10.1016/S0094-114X(02)00051-4

Catto, E. (n.d.). Box2D - A 2D Physics Engine for Games. Retrieved March 31, 2018, from http://box2d.org/

Chang, W.-W., Chung, C.-J., & Sendhoff, B. (2003). Target Shape Design Optimization with Evolutionary Computation. *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, *3*, 1864–1870.

Chen, C. H., & Chou, J. H. (2016). Evolutionary design of adjustable six-linkage bar manufacturing mechanisms using niche genetic algorithms. *IEEE Access*, *4*, 4809–

4822. https://doi.org/10.1109/ACCESS.2016.2597869

Cheney, N., MacCurdy, R., Clune, J., & Lipson, H. (2013). Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding. *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation - GECCO '13*, 167. https://doi.org/10.1145/2463372.2463404

Colombo, G., Mosca, A., & Sartori, F. (2007). Towards the design of intelligent CAD systems: An ontological approach. *Advanced Engineering Informatics*, *21*, 153–168. https://doi.org/10.1016/j.aei.2006.11.003

Cully, A., & Mouret, J.-B. (2016). Evolving a Behavioral Repertoire for a Walking Robot. *Evolutionary Computation*, *24*(1), 59–88. https://doi.org/10.1162/EVCO_a_00143

Cvetkovi, D., & Parmee, I. C. (1999). Genetic Algorithms Based Systems For Conceptual Engineering Design. In *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED 99*. Munich.

Deb, K., & Goel, T. (2001). A hybrid multi-objective evolutionary approach to engineering shape design. *Lecture Notes in Computer Science*, *1993*, 385–399.

Digumarti, K. M., Gehring, C., Coros, S., Hwangbo, J., & Siegwart, R. (2014). Concurrent Optimization of Mechanical Design and Locomotion Control of a Legged Robot. *17th International Conference on Climbing and Walking Robots*, 315–323. https://doi.org/10.1142/9789814623353_0037

Dryden, I. L., & Mardia, K. V. (2016). *Statistical Shape Analysis : with applications in R* (2nd.). Wiley.

Eby, D., Averill, R. C., Punch, W. F., & Goodman, E. D. (1999). Optimal design of flywheels using an injection island genetic algorithm. *Ai Edam*, *13*(05), 327–340. https://doi.org/10.1017/S0890060499135066

Eggenberger-Hotz, P. (2004). Comparing Direct and Developmental Encoding Schemes in Artificial Evolution a Case Study in Evolving Lens Shapes. *Proceedings of the 2004 Congress on Evolutionary Computation*, 752–757. https://doi.org/10.1109/CEC.2004.1330934

Eiben, A. E., & Smith, J. (2015a). From evolutionary computation to the evolution of things. *Nature*, *521*(7553), 476–482. https://doi.org/10.1038/nature14544

Eiben, A. E., & Smith, J. E. (2015b). *Introduction to Evolutionary Computing* (2nd ed.).

Berlin Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-662-44874-8

Ertas, A., & Jones, J. (1996). *The Engineering Design Process* (2nd ed.). New York: John Wiley & Sons, Inc.

Gaier, A., Asteroth, A., & Mouret, J. (2018). Data-Efficient Design Exploration through Surrogate-Assisted Illumination. *Evolutionary Computation*, *26*(3), 381–410. https://doi.org/10.1162/evco_a_00231

Genge, O., & Roosen, P. (2000). Simultaneous experimental optimization of conflicting injection nozzle properties by an evolution pareto set determination. In *ILASS-Europe* (Vol. 1, pp. 1–6). Darmstadt.

Ghassaei, A., & Ming, J. (2015). *Evolutionary Design of Mechanical Linkages*. Retrieved from https://canvas.harvard.edu/files/4287562/download

Hoeltzel, D. a, & Chieng, W.-H. (1990). Pattern matching synthesis as an automated approach to mechanism design. *Journal of Mechanical Design*, *112*(2), 190–199.

Hotz, P. E. (2004). Comparing direct and developmental encoding schemes in artificial evolution: A case study in evolving lens shapes. In *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004* (Vol. 1, pp. 752–757). https://doi.org/10.1109/CEC.2004.1330934

Jitsukawa, T., Adachi, H., Abe, T., Yamakawa, H., & Umezu, S. (2017). Bio-inspired wing-folding mechanism of micro air vehicle (MAV). *Artificial Life and Robotics*, *22*(2), 203–208. https://doi.org/10.1007/s10015-016-0339-9

Joskowicz, L. (1999). Computer-Aided Mechanical Design Using Configuration Spaces. *Science*, 14–21. https://doi.org/10.1109/5992.805133

Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., & Kokaji, S. (2005). Automatic locomotion design and experiments for a modular robotic system. *IEEE/ASME Transactions on Mechatronics*, *10*(3), 314–325. https://doi.org/10.1109/TMECH.2005.848299

Khan, M. S., Ayob, A. F. M., Isaacs, A., & Ray, T. (2011). A novel evolutionary approach for 2D shape matching based on B-spline modeling. *2011 IEEE Congress of Evolutionary Computation, CEC 2011*, 655–661. https://doi.org/10.1109/CEC.2011.5949681

Khan, M. S., & Ray, T. (2012). A Memetic Algorithm for Efficient Solution of 2D and

3D Shape Matching Problems (pp. 362–372). https://doi.org/10.1007/978-3-642-35101-3_31

Kohl, A. M., Kelasidi, E., Mohammadi, A., Maggiore, M., & Pettersen, K. Y. (2016). Planar maneuvering control of underwater snake robots using virtual holonomic constraints. *Bioinspiration and Biomimetics*, *11*(6), 884–899. https://doi.org/10.1088/1748-3190/11/6/065005

Krish, S. (2011). A practical generative design method. *CAD Computer Aided Design*, *43*(1), 88–100. https://doi.org/10.1016/j.cad.2010.09.009

Kyung, M. H., & Sacks, E. (2006). Robust parameter synthesis for planar higher pair mechanical systems. *CAD Computer Aided Design*, *38*(5), 518–530. https://doi.org/10.1016/j.cad.2006.01.004

Lampinen, J. (2003). Cam shape optimisation by genetic algorithm. *CAD Computer Aided Design*, *35*(8 SPEC.), 727–737. https://doi.org/10.1016/S0010-4485(03)00004-6

Lampinen, Jouni. (1997). Choosing a shape representation method for optimization of 2D shapes by genetic algorithms. In *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications (3NWGA)* (pp. 305–319). Helsinki, Finland.

Lapok, P., Lawson, A., & Paechter, B. (2017). Evaluation of a genetic representation for outline shapes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17* (pp. 1419–1422). New York, New York, USA: ACM Press. https://doi.org/10.1145/3067695.3082501

Lapok, P., Lawson, A., & Paechter, B. (2019). 2-Dimensional Outline Shape Representation for Generative Design with Evolutionary Algorithms. In *EngOpt 2018 Proceedings of the 6th International Conference on Engineering Optimization* (pp. 926–937). Springer International Publishing. https://doi.org/10.1007/978-3-319-97773-7

Lawati, M. Al, & Yousef, H. (2016). Stability analysis and trajectory design of a 2-D.O.F. bipedal walker. In *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)* (pp. 1–5). IEEE. https://doi.org/10.1109/CCECE.2016.7726617

Lee, P., & Nagao, T. (1995). Hierarchical Description of Two Dimensional Shapes Using

a Genetic Algorithm. *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, *2*, 1–4. https://doi.org/10.1109/ICEC.1995.487458

Li, D., Zigoneanu, L., Popa, B.-I., & Cummer, S. a. (2012). Design of an acoustic metamaterial lens using genetic algorithms. *The Journal of the Acoustical Society of America*, *132*(November 2011), 2823. https://doi.org/10.1121/1.4744942

Liu, J., & Ma, Y. (2017). Truss-like structure design with local geometry control. *Computer-Aided Design and Applications*, *14*(3), 324–330. https://doi.org/10.1080/16864360.2016.1240453

Liu, Y., & McCarthy, J. M. (2017). Synthesis of a linkage to draw a plane algebraic curve. *Mechanism and Machine Theory*, *111*, 10–20. https://doi.org/10.1016/j.mechmachtheory.2016.12.005

Mariappan, J., & Krishnamurty, S. (1996). A generalized exact gradient method for mechanism synthesis. *Mechanism and Machine Theory*, *31*(4), 413–421. https://doi.org/10.1016/0094-114X(95)00077-C

Matsunaga, R. (2015). Genetic Algorithm Walkers. Retrieved February 26, 2016, from http://rednuht.org/genetic_walkers/

McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., & O'neill, M. (2010). Grammar-based Genetic programming: A survey. *Genetic Programming and Evolvable Machines*, *11*(3–4), 365–396. https://doi.org/10.1007/s10710-010-9109-y

Muller, M. E. (1958). An Inverse Method for The Generation of Random Normal Deviates on Large-Scale Computers. *Mathematical Tables and Other Aids to Computation*, *12*(63), 167. https://doi.org/10.2307/2002017

Mundo, D., Liu, J. Y., & Yan, H. S. (2006). Optimal Synthesis of Cam-Linkage Mechanisms for Precise Path Generation. *Journal of Mechanical Design*, *128*(6), 1253. https://doi.org/10.1115/1.2337317

Myszka, D. H. (2012). *Machines and mechanisms: applied kinematic analysis*. Pearson Higher Ed USA.

Nashvili, M., Olhofer, M., & Sendhoff, B. (2005). Morphing methods in evolutionary design optimization. *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation GECCO 05*, 897–904. https://doi.org/10.1145/1068009.1068159

Olhofer, M., Jin, Y., & Sendhoff, B. (2001). Adaptive encoding for aerodynamic shape optimization using evolution strategies. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress On*, *1*, 576–583. https://doi.org/10.1109/CEC.2001.934443

Padmanabhan, S., Chandrasekaran, M., Ganesan, S., Patan, M. N. K., & Navakanth, P. (2017). Optimal Solution for an Engineering Applications Using Modified Artificial Immune System. In *IOP Conf. Series: Materials Science and Engineering* (Vol. 183). IOP Publishing. https://doi.org/10.1088/1757-899X/183/1/012025

Pahl, G., Beitz, W., Feldhusen, J., & Grote, K. H. (2007). *Engineering Design: A Systematic Approach*. Springer London. Retrieved from https://books.google.de/books?id=57aWTCE3gE0C

Pandey, A., Datta, R., & Bhattacharya, B. (2017). Topology optimization of compliant structures and mechanisms using constructive solid geometry for 2-d and 3-d applications. *Soft Computing*, *21*(5), 1157–1179. https://doi.org/10.1007/s00500-015-1845-8

Parrish, B., McCarthy, J., & Eppstein, D. (2015). Automated Generation of Linkage Loop Equations for Planar One Degree-of-Freedom Linkages, Demonstrated up to 8-Bar. In *Proceedings of the ASME Design Engineering Technical Conference* (Vol. 7). https://doi.org/10.1016/j.cognition.2008.05.007

Pham, D. T., & Yang, Y. (1993). A genetic algorithm based preliminary design system. *ARCHIVE: Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering 1989-1996 (Vols 203-210)*, *207*(24), 127–133. https://doi.org/10.1243/PIME_PROC_1993_207_170_02

Raibert, M. (2008). *BigDog, the rough-terrain quadruped robot*. *IFAC Proceedings Volumes (IFAC-PapersOnline)* (Vol. 17). IFAC. https://doi.org/10.3182/20080706-5-KR-1001.4278

Ratanamahatana, C., & Keogh, E. (2004). Everything you know about dynamic time warping is wrong. *Third Workshop on Mining Temporal and Sequential Data*, 22–25. https://doi.org/10.1097/01.CCM.0000279204.24648.44

Renner, G., & Ekárt, A. (2003a). Genetic algorithms in computer aided design. *Computer-Aided Design*, *35*(8), 709–726. https://doi.org/10.1016/S0010-4485(03)00003-4

Renner, G., & Ekárt, A. (2003b). Genetic algorithms in computer aided design. *Computer-Aided Design*, *35*(8), 709–726. https://doi.org/10.1016/S0010-4485(03)00003-4

Reyes, F., & Ma, S. (2014). On planar grasping with snake robots: Form-closure with enveloping grasps. *2014 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2014*, 556–561. https://doi.org/10.1109/ROBIO.2014.7090389

Robertson, B. F., & Radcliffe, D. F. (2009). Impact of CAD tools on creative problem solving in engineering design. *Computer-Aided Design*, *41*(3), 136–146. https://doi.org/10.1016/j.cad.2008.06.007

Roennau, A., Heppner, G., Nowicki, M., & Dillmann, R. (2014). LAURON V : A Versatile Six - Legged Walking Robot with Advanced Maneuverability. https://doi.org/10.1109/AIM.2014.6878051

Roston, G. P., & Sturges, R. H. (1996). Genetic algorithm synthesis of four-bar mechanisms. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, *10*(05), 371. https://doi.org/10.1017/S0890060400001700

Ruan, Q., Wu, J. X., Zhou, S. H., & Yao, Y. A. (2015). Fluctuation Compensation of a Multi-legged Walking Platform Using Cam Mechanism. In *Proceedings of the 14th IFToMM World Congress* (pp. 294–298). https://doi.org/10.6567/IFToMM.14TH.WC.OS13.104

Ryan, C., Collins, J., & Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In F. T. C. Banzhaf W., Poli R., Schoenauer M. (Ed.), *Genetic Programming. EuroGP 1998. Lecture Notes in Computer Science, vol 1391* (pp. 83–96). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0055930

Sandgren, E., & West, R. L. (1989). Shape Optimization of Cam Profiles Using a B-Spline Representation. *Journal of Mechanisms, Transmissions, and Automation in Design*, *111*(2), 195–201. https://doi.org/10.1115/1.3258983

Sharma, D., Deb, K., & Kishore, N. N. (2008). An Improved Initial Population Strategy for Compliant Mechanism Designs Using Evolutionary Optimization. In *Volume 1: 34th Design Automation Conference, Parts A and B* (pp. 1175–1184). ASMEDC. https://doi.org/10.1115/DETC2008-49187

Shea, K., Aish, R., & Gourtovaia, M. (2005). Towards integrated performance-driven

generative design tools. *Automation in Construction*, *14*(2), 253–264. https://doi.org/10.1016/j.autcon.2004.07.002

Sims, K. (1994). Evolving virtual creatures. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '94*, 15–22. https://doi.org/10.1145/192161.192167

Tai, K., Wang, N. F., & Yang, Y. W. (2008). Target geometry matching problem with conflicting objectives for multiobjective topology design optimization using GA. *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 1873–1878. https://doi.org/10.1109/CEC.2008.4631044

Takahashi, T., Nakamoto, K., Matsumoto, T., Isakari, H., & Kitabayashi, T. (2018). A multi-objective topology optimisation for 2D electro-magnetic wave problems with the level set method and BEM. *European Journal of Computational Mechanics*, *25*(1–2), 165–193. https://doi.org/10.1080/17797179.2016.1181042

Tsuge, B., Plecnik, M., & McCarthy, J. (2016). Homotopy directed optimization to design a six-bar linkage for a lower limb with a natural ankle trajectory. *Journal of Mechanisms and Robotics*, *8*. https://doi.org/10.1016/j.cognition.2008.05.007

Uicker, J. J., Pennock, G. R., & Shigley, J. E. (2003). *Theory of Machines and Mechanisms*. Oxford University Press. Retrieved from https://books.google.co.in/books?id=KE6HMtnXE_0C

Ullman, D. G. (2009). *The mechanical design process* (4th ed.). New York: McGraw-Hill Higher Education.

Vargha, A., & Delaney, H. D. (2017). 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*(2), 101–132.

Vicini, A., & Quagliarella, D. (1999). Airfoil and Wing Design Through Hybrid Optimization Strategies. *AIAA Journal*, *37*(5), 634–641. https://doi.org/10.2514/2.764

Vishal, D., & Manivannan, P. V. (2016). Multi-body dynamics simulation and gait pattern analysis of a bio-inspired quadruped robot for unstructured terrains using adaptive stroke length. *Artificial Life and Robotics*, *21*(4), 493–499. https://doi.org/10.1007/s10015-016-0304-7

Wang, S., Wan, J., Li, D., & Zhang, C. (2016). Implementing Smart Factory of Industrie 4.0: An Outlook. *International Journal of Distributed Sensor Networks*, *2016*. https://doi.org/10.1155/2016/3159805

Weber, R. (2015). BoxCar2D. Retrieved March 31, 2018, from http://boxcar2d.com

Zboinska, M. a. (2015). Computer-Aided Design Hybrid CAD / E platform supporting exploratory architectural design. *Computer-Aided Design*, *59*, 64–84. https://doi.org/10.1016/j.cad.2014.08.029

Zhang, P., Yao, X., Jia, L., Sendhoff, B., & Schnier, T. (2007). Target shape design optimization by evolving splines. *2007 IEEE Congress on Evolutionary Computation*.

Zhang, Y. (2003). Introduction to Mechanisms. Retrieved June 6, 2017, from https://www.cs.cmu.edu/~rapidproto/mechanisms/tablecontents.html#top

Appendices

Appendix 1 Supplementing Tables for Chapter 3

Table 4: Shape Experiment Results part 1

Problem	Method	ymin	lower	middle	upper	ymax	outliers
p01	R1	3753	6163	13779	17332	27761	
	R2	1509	2129	2371	3930	5663	12193, 11449, 8415
	R3	1311	1780	2259	3481	4926	7690, 6880, 8926
	R4	0	2072	2463	3608	4730	8992, 12647
p02	R1	2871	7348	8692	10442	13576	489, 19294
	R2	1051	1734	2324	2666	3473	6215, 6175
	R3	1612	2051	2602	3313	4831	5958, 6709
	R4	360	1606	2310	3585	5846	7350, 6895, 6929
p03	R1	4277	5701	7789	13065	21558	
	R2	4123	5573	6095	6562	7326	
	R3	4272	5195	5972	7122	7478	
	R4	768	2585	3094	3832	5053	5838
p04	R1	1263	7105	9378	11138	14209	
	R2	761	1389	1889	2426	3438	
	R3	463	1196	1622	2046	2794	3398, 3440
	R4	0	1004	1237	2224	2365	5886, 5205, 7448
p05	R1	5035	7267	8076	9943	11720	1239, 1696, 1869, 14111
	R2	1267	1754	1861	2091	2525	1201, 3135, 2957
	R3	1490	1665	1804	1963	2399	4137
	R4	1806	2768	3398	4215	5656	
p06	R1	495	1300	1518	2815	4778	7642, 6795
	R2	115	205	286	356	524	
	R3	78	173	257	318	411	559
	R4	142	195	292	436	726	1200
p07	R1	1787	4493	6419	8566	10986	
	R2	818	1071	1299	1940	2224	
	R3	968	1194	1503	1813	2402	
	R4	1126	1800	2241	2566	2793	3718, 4546
p08	R1	134	2990	4053	5423	7600	10060, 10440, 10242
	R2	418	627	694	900	1265	1313
	R3	239	373	496	576	716	
	R4	610	814	1024	1155	1576	1836, 90

Table 5: Shape Experiment Results part 2

Problem	Method	ymin	lower	middle	upper	ymax	outliers
p09	R1	2514	5644	6962	12112	16361	22065
	R2	1920	2244	2504	2803	3585	6368
	R3	2345	2776	3043	3580	3816	7068, 6447, 5682, 6950
	R4	1292	1563	1710	1982	2305	4150, 3510, 840
p10	R1	899	5050	6693	8121	12421	17631, 12812
	R2	1291	1862	2253	2489	3186	
	R3	1762	2254	2581	2860	3748	4463, 4514
	R4	1747	2639	2796	3578	3887	6085, 7308
p11	R1	7390	10196	12286	12796	16491	16867
	R2	3385	3506	3924	4039	4314	5454
	R3	3081	3537	3834	4192	5148	6712, 6438
	R4	1592	2850	3929	4739	6618	8921, 8033
p12	R1	4439	6884	10877	14428	18244	
	R2	5016	5797	6059	6408	7324	
	R3	4878	5646	6133	6541	7771	
	R4	470	2837	3286	6316	8592	
p13	R1	429	1272	1570	1868	2290	3134, 2938
	R2	275	352	475	646	841	
	R3	183	300	384	454	617	689, 704
	R4	442	709	771	1003	1284	
p14	R1	3	933	1185	1804	2670	7011
	R2	195	446	500	619	775	957
	R3	228	294	333	391	440	1166, 616, 602
	R4	438	527	676	988	1376	
p15	R1	499	4402	5458	7963	12474	16752
	R2	772	1039	1252	1903	3155	4118
	R3	652	1357	2092	2347	3235	
	R4	976	1695	2240	3189	4838	6614
p16	R1	182	1349	2191	2963	5367	
	R2	284	422	495	564	704	919, 849, 1006, 835
	R3	227	376	462	521	696	
	R4	512	686	815	1030	1466	1642

Table 6: Shape Experiment Results part 3

Problem	Method	ymin	lower	middle	upper	ymax	outliers
p17	R1	1164	2015	2234	2719	3142	6320, 644
	R2	60	204	372	582	884	
	R3	125	327	413	576	888	956, 1716, 1211
	R4	67	343	848	1544	1859	
p18	R1	16	1595	2072	3287	5434	
	R2	44	160	254	336	380	820, 799
	R3	76	161	213	268	374	460, 545
	R4	50	206	249	475	878	1066, 974, 2278
p19	R1	0	2203	3949	5988	11641	
	R2	495	711	829	1433	2003	
	R3	337	678	829	1522	2442	
	R4	574	862	1084	1510	2407	3807, 3747
p20	R1	27	1433	3573	5798	8099	
	R2	201	490	623	806	1174	
	R3	184	324	367	420	501	595, 166, 618
	R4	192	306	433	748	958	4526, 1584, 2247
p21	R1	3794	4955	6188	7103	8963	670, 538
	R2	357	533	618	703	911	203, 1001, 188
	R3	179	495	637	717	841	
	R4	647	1335	1959	2152	3011	4778
p22	R1	112	1149	2419	3771	5706	
	R2	6	119	201	338	517	674, 671
	R3	0	9	54	108	223	336, 313
	R4	1	114	627	1210	2276	
p23	R1	2092	4798	6028	8253	10818	
	R2	416	700	1154	1536	2280	3824
	R3	262	619	1077	1375	1986	3206
	R4	347	730	1173	1810	3282	4055
p24	R1	5542	9854	13786	22673	31440	
	R2	9270	11304	12950	13904	16429	
	R3	10064	11056	12574	14071	16961	
	R4	863	1678	2088	3422	5886	7002, 9250, 10115, 9113

Appendix 2 Simulator Acceptance Tests

This section summarises the acceptance tests used to validate the physics part of the simulator implementation.

Appendix 2.1 Collision Tests on One Layer

This section describes different collision tests on one layer which were used to validate the simulator.

The tests validate the collision between a dynamic body consisting of different shape types, such as polygon shapes and circle shapes with environment objects. Each scene contains a bearing plate with walls on which a dynamic object is dropped. The bearing plate has a static position and has static walls. The gravity is set to 9.81 in the negative y-direction, and the frame rate is set to 60fps. The material parameters for friction are set to 0.2, for restitution 0.6, and density 1.0. The setup is shown in Figure 68.

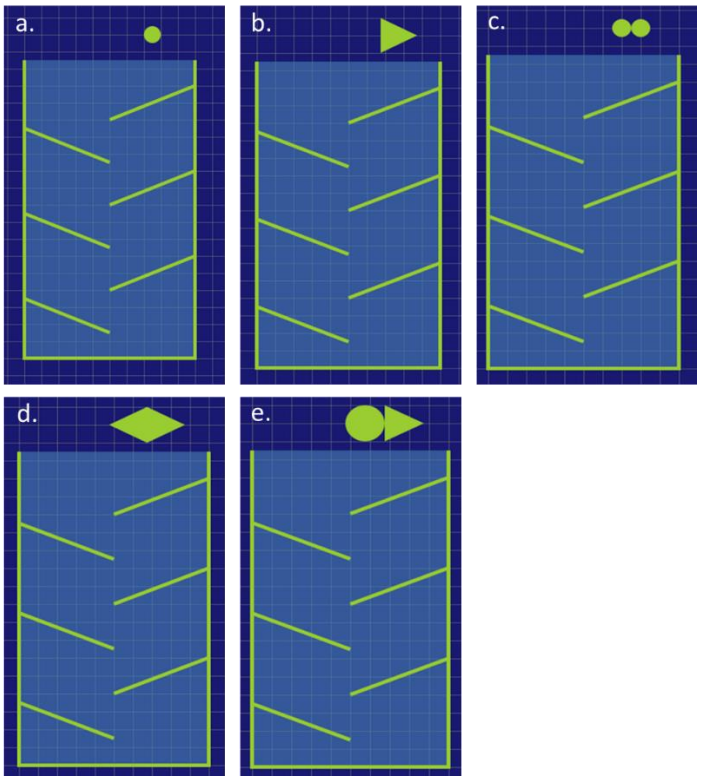


Figure 68: Collision

Test *a* validates the circle shape type; *b* the polygon shape type; *c* two circle shapes in one scene which should act as one component; *d* two polygon shapes which should act as

one component; and *e* which combines the polygon shape type with a circle shape, both should act as one component.

Expected

It is expected that the dynamic body falls, without colliding with the bearing plate. However, it needs to collide with the walls which are placed on the same layer. The shape component should bounce off and either roll or slide down the contour of the walls.

Outcome

In all cases, the shaped component falls, collides, and behaves as expected.

Appendix 2.2 Collision Test with Different Dynamic Components

The test validates the collision between a dynamic component made of multiple shapes and a bearing plate with walls. The plate has a static position and contains static walls which work as obstacles. Both shapes should collide with the outer wall. The shapes of the dynamic component and the walls get different layers assigned. Both components should collide with the outer wall of the bearing plate. Besides, the circle shape should collide with the wall 2, 4, and 6. The polygon shape should collide with wall 1, 3, and 5. The gravity is set to 9.81 in the negative y-direction, and the frame rate is set to 60fps. The material parameters for friction are set to 0.2, for restitution 0.6, and density 1.0. The setup is shown in Figure 69.

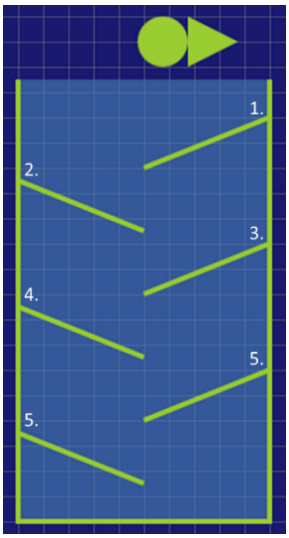


Figure 69: Collision Multi-layer

Expected

It is expected that the shape component falls, without colliding with the bearing plate. Then the polygon shape collides with wall 1, and the circle shape should go right through

it, still connected to the polygon. The component should slide with both shapes towards wall 2, and the circle shape should collide with it. The polygon shape should not collide and fall through. Both components should then slide further down.

Outcome

The shapes fall, collides, and behaves as expected.

Appendix 2.3 Collision Test with Different Static Components

The test is similar to the previous test, however, in this case, the layers of the static components are changed, and the dynamic component is placed on one layer. The setup is similar to Figure 69.

Expected

The dynamic component should only collide with wall 1, 3, 4, and 6.

Outcome

The body falls, collides, and behaves as expected.

Appendix 2.4 Parameter Tests

In this section, the tests are focusing on the gravity, density, friction, and restitution parameter.

Appendix 2.4.1 Gravity

The gravity parameter is tested by assigning a low gravity value of 2.0 in one scene, and then a high gravity value of 18.0 in another. A similar scenario is used as represented in Figure 69. The gravity is set in the negative y-direction, and the frame rate is set to 60fps. The material parameters for friction are set to 0.2, for restitution 0.6, and density 1.0.

Expected

It is expected that the body in the first scene falls slow, and the one in the second faster.

Outcome

Everything behaves as expected when changing the gravity in the simulator.

Appendix 2.4.2 Density

The density parameter is tested by assigning a low-density value of 0.1 in one scene, and then a high-density value of 2.0 in another. A similar scenario is used as represented in

Figure 69. The gravity is set to 9.81 in the negative y-direction, and the frame rate is set to 60fps. The material parameters for friction are set to 0.2 and for restitution 0.6.

Expected

It is expected that the body in scene one behaves lighter compared to the one in scene two.

Outcome

Everything behaves as expected.

Appendix 2.4.3 Friction

The friction parameter is tested with a polygon shape as it has more surface in contact with the wall. The restitution is set to 0 for all bodies to reduce the bouncing behaviour. Each wall segment from 1 to 6 has a different friction value assigned, increasing from 0.0 to 1.0. Each step increases the value by 0.2. The dynamic component has friction set to 0.1. The gravity is set to 9.81 in the negative y-direction, and the frame rate is set to 60fps.

Expected

It is expected that the body slides down the walls and slows down on each step because of the increase of the friction between the dynamic component and the wall.

Outcome

Friction increases and component slows down as expected.

Appendix 2.4.4 Restitution

The restitution parameter is tested with a circle shape as the bouncing behaviour should be better visible. The restitution of the dynamic component was set to 0. The walls 1 to 6, have an increasing restitution parameter from 0.0 to 1.0. Each step increases the value by 0.2. All bodies have the friction set to 0.2. The gravity is set to 9.81 in the negative y-direction, and the frame rate is set to 60fps.

Expected

It is expected that the body starts to bounce more on every step-down.

Outcome

The bouncing behaviour increases on the way down, which is as expected.

Appendix 2.5 Single Joint Test

This section focuses on testing the revolution joint. It is positioned in the centre of a polygon shape and connects the polygon shape with a bearing plate. The setup is shown in Figure 70. The gravity is set to 9.81 in the negative y-direction, and the frame rate is set to 60fps.

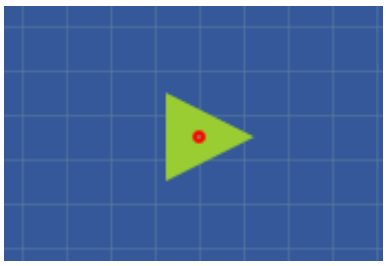


Figure 70: Revolution Joint

Expected

It is expected that the joint is placed in the correct position and act as the centre point of the rotation. The component should start to seesaw around the revolution joint.

Outcome

The joint is placed correctly, and the polygon shape rotates around the revolution joint.

Appendix 2.6 Linkage Test

A chain of three components is used to test the correct implementation of the simulator. The bodies are connected with revolution joints. The first body is connected to the bearing plate. The second body is connected to the first, and the third is connected to the second body. The gravity is set to 9.81 in the negative y-direction, and the frame rate is set to 60fps. The setup is shown in Figure 71.

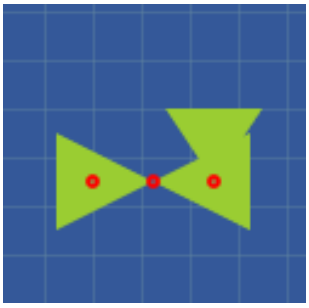


Figure 71: Revolution Joint Chain

Expected

It is expected that the components behave like a chain. Connected components should not collide with each other.

Outcome

The components behave as expected. Connected components are not colliding.

Appendix 2.7 Actuator Tests

This section focuses on testing the actuator functionality of the simulator. The actuator is tested by using the scenario seen in Figure 70 and setting the *isMotor* variable to true. The speed is set to 60rpm.

Expected

It is expected that the body rotates with a speed of one rotation per second.

Outcome

The body rotates with one rotation per second as expected.

Appendix 3 Problem-file Format

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" baseProfile="full"
width="1000px" height="500px" viewBox="0 0 1000 500">

    <style>@import url(style.css);</style>
    <desc fileID="0">Example Problem</desc>


    <!-- Grid Background -->
    <g id="background_grid">
        <defs>
            <pattern id="grid20" width="20" height="20" patternUnits="userSpaceOnUse">
                <path d="M 100 0 L 0 0 0 100" />
            </pattern>
        </defs>
        <rect id="bgcolor" height="500" width="1000" y="0"></rect>
        <rect fill="url(#grid20)" height="500" width="1000" y="0"></rect>
    </g>

    <!-- Environment Configuration -->

    <!-- Ground -->
    <g id="ground">
        <g id="1" predefined="true" transform="translate(0, 300) rotate(0) scale(1)" type="static">
            <polyline type="polygon" points="-1000,200,-1000,0,10000,0,10000,200" friction="0.5"
            restitution="0.6" density="1.0" plane="0"/>
        </g>
    </g>

    <!-- Environment element -->
    <g id="env_element"></g>

    <!-- Housing -->
    <g id="housing">
        <g id="2" predefined="true" transform="translate(250, 170) rotate(0) scale(1)" type="dynamic">
            <!-- Housing Background -->
            <polyline type="polygon" isSolutionSpace="true" points="-150,25, -150,
            -25, 150,-25, 150,25" friction="0.2" restitution="0.6" density="1.0" plane="1"/>
        </g>
    </g>

    <!-- Drive -->
    <g id="mech_config">
        <g id="levers">
            <!-- potential solution -->
        </g>

        <!-- transform needs to be similar to transform of bodyB -->
        <g id="joints">
        </g>
    </g>

    <!-- Parameter definition -->
    <parameter>
        <optimisationcfg populationSize="40" childrenNumber="10" solutionsToProduce="20000"
        trackDirectionX="1" trackDirectionY="0" mutationMethod="4" crossoverMethod="2"
        mappingVer="15">
            <weight name="actuatorrotation" value="0" />
            <weight name="walkingdistance" value="1" />
            <weight name="areapenalty" value="0" />
            <weight name="jumppenalty" value="0" />
        </optimisationcfg>

        <constraints noOfPlanes="1" shapeSizeMin="10" shapeSizeMax="80" nodesPerShape="6"
        shapesPerLever="1" allowedShapeTypes="0" noOfLevers="1" noOfJointsPerLever="1"
        allowedJointTypes="0" />

        <simulation gravity_x="0" gravity_y="9.81" simulation_frames="600" collideConnected="False"
        frameRate="60" pixelWorldRatio="100" />
    </parameter>

    <!-- Results -->
    <results user="empty" machine="empty" date="empty" time="empty" processingTime="empty"
    solutionsProduced="0" totalscore="0" chromosome=""></results>

    <!-- Simulation Data -->
    <errors></errors>

    <!-- Simulation Data -->
    <simulation_frames valid="false"></simulation_frames>

</svg>
```