Many-objective evolutionary algorithms for designing infrastructure using voting based selection function

subtitle

Omslag/voorblad

Op de omslag komt minstens het volgende te staan:

- titel;

- (ondertitel);

- naam organisatie of bedrijf;

- auteursnaam;

- naam of namen van begeleider(s) van de HvA en vanuit het bedrijf

De informatie op de omslag wordt overzichtelijk weergegeven. De belangrijkste informatie is de titel met

eventuele ondertitel. Deze informatie neemt de meeste ruimte in beslag en komt over het algemeen

bovenaan in het midden te staan. De overige informatie plaats je rechts onderaan op de omslag.

Company: Volker Infra

Author: Ilan Beck

Begeleider uit school: Stijn Oomes

Begeleider uit bedrijf: Martinus Ruitenbeek

# Title Page

# Preface

**Table of Contents**

[​ Title Page 3](#__RefHeading___Toc333_1445170317)

[​ Preface 4](#__RefHeading___Toc335_1445170317)

[​ Abstract 7](#__RefHeading___Toc339_1445170317)

[​ 1. Introduction 8](#__RefHeading___Toc341_1445170317)

[​ 2. Problem Description 11](#__RefHeading___Toc343_1445170317)

[​ 2.1 Questions and null hypothesis 12](#__RefHeading___Toc702_3492016238)

[3. Research Method 13](#__RefHeading___Toc704_3492016238)

[3.1 Method of data generation 14](#__RefHeading___Toc706_3492016238)

[3.2 Prototype design 15](#__RefHeading___Toc478_1752712029)

[​ 3.3 Method of analysis 16](#__RefHeading___Toc480_1752712029)

[​ 4. Literature research 17](#__RefHeading___Toc708_3492016238)

[​ 4.1 Evolutionary algorithms for design 17](#__RefHeading___Toc345_1445170317)

[​ 4.2 Many Objective Optimization 19](#__RefHeading___Toc373_1445170317)

[​ 5. Conceptual Framework 21](#__RefHeading___Toc710_3492016238)

[​ 6. Prototype Genetic Algorithm 23](#__RefHeading___Toc351_1445170317)

[​ 6.1 Requirements 23](#__RefHeading___Toc712_3492016238)

[​ 6.2 Development process 24](#__RefHeading___Toc714_3492016238)

[​ Unity Prototype 24](#__RefHeading___Toc716_3492016238)

[​ Blazor prototype 25](#__RefHeading___Toc718_3492016238)

[​ 7. Practical Research 26](#__RefHeading___Toc470_1752712029)

[​ 7.1 Types of selection 26](#__RefHeading___Toc472_1752712029)

[​ 7.2 Difference for optimal results 27](#__RefHeading___Toc355_1445170317)

[​ 7.3 Difference for speed 28](#__RefHeading___Toc357_1445170317)

[​ 8. Analysis 31](#__RefHeading___Toc468_1752712029)

[​ 9. Conclusion 32](#__RefHeading___Toc349_1445170317)

[​ 10. Recommendations 33](#__RefHeading___Toc361_1445170317)

[​ Post Script 34](#__RefHeading___Toc363_1445170317)

[​ De noten 35](#__RefHeading___Toc365_1445170317)

[​ Bibliography 36](#__RefHeading___Toc367_1445170317)

[​ De bijlagen (altijd nummeren) 37](#__RefHeading___Toc369_1445170317)

[​ Traveling salesmen problem 37](#__RefHeading___Toc777_838605717)

# Abstract

# 1. Introduction

Innovation happens everywhere. Computers often play a role in this innovation. Different forms of artificial intelligence have become a part of the daily life of people and play a crucial role in industry and science. Deep Learning is a very popular form of AI but it’s definitely not the only one. Many other algorithms exist and are more suitable for some tasks. Developing and improving these systems is important for most companies.

The construction industry is slow to accept change. But while a small, local bakery might be able to keep working in the same way it has done for the last few decades this isn’t true for the construction industry. If the competition can deliver the same thing for less money customers will, ultimately, chose the competition.

It is therefore of great importance that a company looks into current and further technologies and how best to implement them.

This paper was written for Volkerwessels Infrastructuur, a company that designs and builds infrastructure like bridges, tunnels, dikes and floodgates. Volkerwessels Infrastructuur is part of Volkerwessels and is itself also composed of several companies.

When designing a piece of infrastructure there are many different aspects to consider. Big categories include, cost, structural integrity and aesthetics. These aspects are often in conflict with each other.

Teams of specialists discuss the different aspects, each team with it’s own specialism. They have to find a compromise between the different aspects. This is a form of many-objective optimization, an algorithm that does this is referred to as a many objective optimisation algorithm.

It is important to note here that there is no objective single best solution. The best solutions are called the Pareto front, solutions that can’t be improved on in regard to one objective without sacrificing at least one other objective. In practice, finding this Pareto front is impossible as the number of possibilities is far to large. The goal is finding a solution that is good enough in regard to every objective.

Evolutionary Algorithms(EAs) are a form of search methods, they can be used to find solutions to many different problems. This includes finding a solution, in the form of a design, for a design problem. EA are already used within the construction industry and Volkerwessels Infrastructuur. Dynamo, a piece of software that is part of several autodesk applications, is used a lot. Within Dynamo there is Optimo, which focusses on multi-objective optimisation.

<https://dynamobim.org/optimo/>

<https://link.springer.com/chapter/10.1007/978-3-030-69984-0_52>

<https://www.tandfonline.com/doi/abs/10.1080/10798587.2009.10643036?journalCode=tasj20>

The focus of this paper is about improving EA when designing a structure with multiple objectives. The main question is whether representing each objective as an individual “voice” improves the EA over other multi/many objective EA. Improvement can be reaching a good result quicker, getting better results (closer to the Pareto front) or making the process easier for humans to follow and influence. The last one will be hard to conclude because it is more subjective. Therefore this research will not make a definitive statement about it. However, a recommendation for subsequent research in about this can be given.

For the practical part of this research a bridge will be used as example infrastructure. It will be a very simplified bridge made of blocks. This will obvious not have any use for real infrastructure design yet. The program will be solely for research and proof of concept. If, with the limited development time available for this research, the program can design bridges similar to a young kid there is good hope it can be expanded to be of practical use in the future.

The reason a bridge is chosen is because it is easy to represent in a simple form while still offering clear challenges.

Een inleiding is een vooruitblik op de inhoud en structuur van het verslag. Je beschrijft kort de

aanleiding tot dit project. Schrijf niet in de ik/jij/wij stijl, zet je project centraal. De woorden ik/jij/wij etc zijn überhaupt verboden in je rapport (met uitzondering van voorwoord en reflectie). Schrijf ook niet over jezelf in de derde persoon (De Afstudeerder . . . ).

De inleiding bevat:

- achtergrondinformatie/onderwerp/beschrijving/context waarbinnen de opdracht is uitgevoerd;

- Waarom het rapport is geschreven, m.a.w. wat de opdracht was die aangepakt c.q. opgelost diende te

worden. Dit wordt ook wel de probleemstelling genoemd;

- welke procedure er gevolgd is om tot een oplossing te komen;

- De structuur van het rapport (leeswijzer). Dit is een vooruitblik op de inhoud en opbouw van het

afstudeerrapport.

Advies: Schrijf de inleiding pas als de hoofdtekst klaar is.

De kernhoofdstukken

Begin de hoofdstukken steeds met een inleidende tekst waarin staat wat er in het hoofdstuk besproken

of geanalyseerd wordt, waarom dat is en in welke volgorde.

Eerst schrijf je een hoofdstuk waarin je weergeeft wat de achtergrond is van je opdracht. In welke

context vindt het plaats, wat is het probleem en wat verwacht de opdrachtgever. Dit deel van het

afstudeerrapport kan na twee tot vier weken geschreven en besproken worden. Je krijgt dan mogelijk

de volgende paragrafen:

1. Beschrijving van de context van de opdracht:

o het bedrijf of de organisatie waar vanuit de opdracht geformuleerd is;

o de opdrachtomschrijving zoals die bij de aanvraag van de afstudeeropdracht goedgekeurd

is;

o analyse van de opdracht. Hier wordt de onderzoeksvraag geformuleerd, wat is het doel van

deze opdracht; wat wil men bereiken;

o probleemvraag; wat is de eigenlijke vraag;

o (definitie termen);

o (randvoorwaarden);

o deelvragen; hierin werk je de hoofdvraag uit in deelvragen.

 bijvoorbeeld de vraag of het (deel)product al bestaat en of het beter is dit aan te

schaffen kopen of dat je bijvoorbeeld bestaande software kan aanpassen of dat het

toch het helemaal ontwikkeld moet worden

Wat de andere hoofdstukken zijn, is afhankelijk van de opdracht. In ieder geval moet er een hoofdstuk

zijn over de methoden/technieken die gebruikt zijn en welke methoden/techniekenwaarom voor de

uitvoering zijn gebruikt. Zo zouden de andere hoofdstukken kunnen bestaan uit:

2. (onderzoeks)methodebeschrijving en –verantwoording;

3. onderzoek naar hoofd en deelvragen;

4. ontwerp en realisatie van het product indien van toepassing;

5. testresultaten en evaluatie als je een product gemaakt hebt of een analyse van de

resultaten/consequenties als je een onderzoeksrapport gemaakt hebt;

6. eventueel de implementatie/uitvoering.

Aandachtspunten:

- publiceer niet de complete source, Code snippets die uitwerkingen/oplossingen toelichten is

beter;

- indien van toepassing één voorbeeldtabel hier van een uitwerking, de rest in de bijlagen;

- indien van toepassing één voorbeeldscherm hier, de rest in de bijlagen;

- indien van toepassing één user story hier, de rest in de bijlagen.

Het uitgangspunt is dat het rapport zelfstandig leesbaar moet zijn zonder de bijlagen. De bijlagen

dienen slechts additionele informatie te bevatten.

# 2. Problem Description

The program has to design a block bridge that is good enough in regard to several, partly conflicting, objectives. A real bridge has many objectives, ranging into the thousands. While not all of those objectives have a direct impact on each other it is still reasonable to expect optimisation between more than 10 conflicting objectives will be necessary.

Given that some algorithms do work well for a few objective but not for more, it will be important that the prototype has a larger amount conflicting objectives. Having at least over 5 objectives would give good hope the algorithm will work with more. Each design consists of a 3D grid of ‘one’s and ‘zero’s. An ‘one’ means a block will be placed on that position, a ‘zero’ means no block will be placed. The bridge has to cross a gap between two stable surfaces. In-between the two surfaces is a lower surface, representing the river floor. The size of a single block is revered to as an unit.

The axis across the two stable surfaces is referred to as the length of the bridge and the axis parallel to the edge of the stable surfaces the width.

The following objectives will be used:

* **Cost**, the amount of blocks used, the lower the better
* **Amount of roads** run a 7 second simulation, afterwards across the width of the bridge, check for full roads. Across the length of the bridge, using 1 unit intervals, the height of the top surface is found. If the height difference between each point and the previous point is less then 1.5 units it is considered a full road. If less then 1 full road is found a partial score is given depending on how far the furthest partial road goes before a height difference of more then 1.5 units is found.
* **Flatness of the “road” surface**  using the information from the previous test the a score is given for each road. The average height difference between points on a road is subtracted from the score. If the previous test found no full road, the average height difference between points on the partial road is used.
* **Carrying capacity** run a 7 second simulation and record the end position of every block. On each valid spot, as determined by the “amount of roads” test a weight 4 times the mass of a block is dropped. After 5 seconds, compare the difference between the position each block to it’s earlier position. The lower the difference the better.
* **Stability** is measured using two different methods:
  + **Velocity** over the course of a 5 second simulation the velocity of each block is recorded 50 times, the average velocity of all blocks being the negative score. Ideally non of the blocks should move.
  + **Difference** a five second simulation is ran twice, afterwards the position of each block after the first simulation is compared to the position of each block after the second simulation, the average difference being the negative score. A lower difference means the bridge is less influenced by random fluctuations between simulations.

The experimental prototype developed for this research is using an evolutionary algorithm. This prototype has multiple fitness functions each giving a score to a chromosome. Deciding which chromosomes get selected for the next generation can be done in multiple ways. This research focusses on a selection function based on voting.

In order to compare a voting based selection function to alternatives some software is needed that can do both. This way the selection function is, as far as possible, the only thing that is different.

## **2**.1 Questions and null hypothesis

**Central question:** Can a Many-objective Evolutionary Algorithm for designing bridges be improved by using a voting based selection function?

**Sub question 1:** Which Evolutionary Algorithms are currently used for Many-objective optimization problems

**Sub question 2:** Is it more likely that better results are found when using a voting based algorithm over an algorithm that adds scores together

**Sub question 3:** Is it more likely that better results are found when using a voting bases algorithm with a small versus a large amount of designs getting voted-out

**Null hypothesis:** Using a voting based selection function does not improve the results nor the performance over other many-objective Evolutionary Algorithm selection functions.

# 3. Research Method

RIDC

Literature research

plaatje

## 3.1 Method of data generation

To get the data necessary to compare the different methods of selection a prototype will be build.

This prototype does not have to be able to design real infrastructure. Making something that can would be far beyond the scope of this paper. If the prototype can design structures on the level of a toddler playing with blocks it would be enough.  
Due to the random nature of a genetic algorithm (source) multiple tests will need to be ran for each method of selection. With enough data a statistical pattern can be found, more on this in Method of Analysis.

## 3.2 Prototype design

In order to run tests a prototype needs to be made. This prototype must be able to run a genetic algorithm to design something. At least the part of the algorithm that select which designs pass to the next generation needs to easily replaceable, without changing the rest of the prototype. That way it is possible to run tests on different methods of selection.

For the prototype the choice was made to let is design simple block bridges. Volker Infra designs, among other things, bridges. Using simple blocks means no knowledge is needed about designing real bridges, which would be far beyond the scope of this research.

In order to apply fitness scores to the bridges the prototype needs to be able to simulate the structures.

## 3.3 Method of analysis

To be able to reject the null hypothesis it needs to be shown that the more complex selection function preforms, on average, better than the simple selection function. Given that an evolutionary algorithm works using random mutations, a random starting point and possible randomized crossover one algorithm will not always perform better than the other. By looking if the difference is statistical significant it is possible to, with a high degree of accuracy, say whether or not one algorithm preforms better than the other.

If statistical significant difference can be found between the result of selection functions in favor of a complex selection function, the null hypothesis can be rejected. This means that it would technically allow for a one-tailed p value. ([GraphPad Prism 9 Statistics Guide - One-tail vs. two-tail P values](https://www.graphpad.com/guides/prism/latest/statistics/one-tail_vs__two-tail_p_values.htm) )

However a two-tailed p value is still used because seeing whether or not a difference in favor of the simple selection function exist, would still be interesting and could be useful for further research.

This research will use a significance level of 0.01

While just showing statistical significance might be enough to reject the null hypothesis it might also be interesting to look further at the data. If the complex algorithm for example preforms much better in 10% of the cases and about equal in 90% of the cases this may or may not be statistical significant but would raise an interesting question: why does it preform so good in those specific situations and is it possible to somehow raise the times this occurs.  
This would be something for a follow-up research.

# 4. Literature research

## **4.1 Evolutionary algorithms** for design

Evolutionary Algorithms (EA) are a set of search algorithms inspired by natural evolution. They are generally not considered Machine Learning(ML). EA are considered search algorithms that can be applied to a wide range of problems. EA can however be used used within Machine Learning. An example is NeuroEvolution which uses an EA to construct a neural network where an EA is used as an alternative to another algorithm like gradient decent. EA are useful if the underlying math is not known and the quality of a result has to be derived from simulation. EA are also able to deal with multi-modal functions which many other optimization algorithms can’t. Multi modal functions are...

Several types of EA exist. The most common ones being:

* Genetic algorithms (GA)
* Evolutionary strategies (ES)
* Evolutionary programming (EP)
* Genetic programming (GP)

The first 3 of them where developed independent of each other, they where later grouped together under the term of Evolutionary Algorithms.

All of them rely on the same basic principle:

1. create an initial population
2. evaluate the members of the population, determine their fitness
3. select members of the population based on their fitness, if termination condition is met: end here
4. create new individuals based on the selected population, most often this is done using mutation and crossover, go back to step 2

Mutation makes a random change to a random value in an individual. Crossover is inspired by sexual reproduction and mixes 2 individuals together.

The difference between the different forms are in the way an individual is expressed, the form of selection and reproduction.

GA and GE works normally by a combination of crossover and mutation, ES works with mutation and sometimes with crossover, EP doesn’t use crossover, only mutation.

GA represent the individuals by encoding the values, often as a string of bits. ES, EP and GP normally don’t encode the values. They operate directly on the values used in the solution. Using terminology from biology a solution is called phenotypes, when encode they are called genotypes. GA is therefore sometimes referred to as genotypic while ES, EP and GP are referred to as phenotypic.

Taking the example of a bridge, the phenotype would be numbers that directly correspond to real world values. If a metal bar is 2000 millimeters (2 meters) it’s appears as 2000 in the object. In a genotype it would, along with all other values be encoded, normally as a string of bits. This is important during mutation and crossover. If mutation consists of “flipping” a bit, 2000 millimeters can suddenly turn into 4048 or 976 millimeters. Algorithms that work with the real values can mutate relative to it’s current value as well as based on what the value represents. Adding 200 millimeters to the length of a steel reinforcement bar might be reasonable, adding 200 millimeters to the diameter of a screw is not.

https://thescipub.com/pdf/ajeassp.2009.789.795.pdf

<https://doi.org/10.1109/ICGTSPICC.2016.7955308> Evolutionary Algorithms: A Critical Review and its Future Prospects

https://www.academia.edu/15165881/Evolutionary\_computation\_from\_genetic\_algorithms\_to\_genetic\_programming

## 4.2 Many **Ob**jective Optimization

Many objective optimization is a term used for optimization with 4 or more objectives. It is distinct from multi-objective optimization which has 2 or 3 objectives. In optimization the concepts of Pareto dominance, Pareto-optimal and the Pareto front are important. Named after Vilfredo Pareto an Italian civil engineer, sociologist, economist, political scientist, and philosopher who introduced the concept (wikipedia). A solution can be said to dominate another solution if it is a better solution in regards to at least one objective and not worse in regards to any objective. If a solution dominates another solution it will always be better to take the dominating solution.

A Pareto-optimal solution is a solution that is not dominated by any other solution. The Pareto front is the collection of all Pareto optimal solutions. Ideally you want to find a solution on the Pareto front, practically for a lot of problems it’s not possible to find one, as it is not possible to explore the entire space of possible solutions.

Pareto-Dominance is used in a lot multi/many objective optimization algorithms. Examples of algorithms using Pareto Dominance are are NSGA-II and SPEA2.

A problem here is that the amount of dominated points reduces when the number of objectives increases. This is actually logical, dominance relies on one solution being equal or better at every objective. The more objectives, the higher the chance that a solution is worse at, at least one objective.

Evolutionary algorithms that rely on eliminating dominated points lose effectiveness. After all, survival of the fittest doesn’t work as well when everyone is “the fittest”.

A solution is using “fuzzy Pareto dominance” which no longer requires one solution to completely dominate the other solution.

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.709.4835&rep=rep1&type=pdf

[Comparison of multi-objective optimization methodologies for engineering applications | Elsevier Enhanced Reader](https://reader.elsevier.com/reader/sd/pii/S0898122111010406?token=523E06621BD514A7292E6690661E34D3BF75E37BBE24623F20AA16514B593D92BFDDF0036B09B581C985FF2CE03AFB60&originRegion=eu-west-1&originCreation=20220912151113)

[EMoSOA: a new evolutionary multi-objective seagull optimization algorithm for global optimization (researchgate.net)](https://www.researchgate.net/profile/Gaurav-Dhiman-6/publication/343737478_EMoSOA_A_New_Evolutionary_Multi-objective_Seagull_Optimization_Algorithm_for_Global_Optimization/links/5f59cb8ba6fdcc1164048aed/EMoSOA-A-New-Evolutionary-Multi-objective-Seagull-Optimization-Algorithm-for-Global-Optimization.pdf)

[MohammadAslDissertation.pdf](file:///C:/Users/ibeck/Downloads/MohammadAslDissertation.pdf)

[c2014022.pdf (msu.edu)](https://www.egr.msu.edu/~kdeb/papers/c2014022.pdf)

## 4.3 Many **Ob**jective Evolutionary algorithms

# **5. Conceptual Framework**

In many-objective optimization the Pareto front can be practical impossible to find. But even when it’s found it can be extremely broad. If a set of data that is on, or at least close to, the Pareto front is found, negotiations don’t end. The goal of negotiating is to ultimately choose one course of action, even if this course of action is walking away from the negotiation. If this program wants to help with negotiation the ability to actually compare solutions on, or close to, the Pareto Front is a crucial part.

When humans negotiate an important part can be presenting arguments, reasons and explanations. Not only can one change the other persons opinion but giving others information about why one wants, or doesn’t want something can help narrow down the search space.

Having a AI agent present this information in the same way humans do would be extremely complex. This might not be necessary. If agents can tell which solutions in a large set of solutions they do, and don’t want, they do give information that helps narrowing down the search space.

It might be possible to have an AI find the part that a certain agent finds important and present that to a human. Combining this with the agents specialism could give a human a clear idea about why a certain thing is preferred.  
This won’t change the opinion of an other agents, but in the context where each agent is an expert on it’s own field, changing opinion wouldn’t be desirable in the first place. It might however give the humans who, at least for now, need to make the final decision a clearer insight about the why behind the design.

## 

# 6. Prototype Genetic Algorithm

As mentioned in the chapter Research Methodology a prototype will be developed to show if an complex selection function works better than a simple one.

## 6.1 terminology

## 6.2 Requirements (updaten)

The primary goal of the prototype is to generate the data need to potentially reject the null hypothesis. The prototype needs to be able to run and collect data from an genetic algorithm with both a basic, and an advanced selection function.

The prototype needs to collect too different pieces of data: the time it took to reach an optimum and the quality of the optimum.

The time can be measured in generations, or actual time or both. The amount of generations would be based purely on the success of the algorithm, while the time might also be influenced by external factors, for example the computer installing an update in the background. However the number of generations does not take into account that one algorithm might be slower. Collection both pieces of data and comparing them is therefor the best option.

Determining when the local optimum has been reached is also something to consider. More about this in the chapter prototype design.

Measuring the quality of the optimum is problematic. The different fitness functions each have a different measure of quality. Adding it together is what one selection function does, and might bias the result unfairly in favor of that one. Ideally the results would be rated by a third-party. Using a human third party however is not possible. Even having all the individual result inspected by one person would be a long and tedious task. Even then one person would not be objective enough.

Getting a team of people to rate each result is outside the resources of this research.

Another option, given that the simple nature of the prototype, would be to calculate the optimal result, then compare the end result to that. The problem remains that the entire setup of this research is based around the fact that no solution exist that is optimal in every possible way.

Their would likely not be one optimal result, but multiple optimal results along the pareto front. If these solutions are known there is a point of reference. This does raise the question whether or not the results can be extrapolated to situations where the optimal solution can’t be calculated.

A simpler solution might be to take all results and remove all that are not a pareto equilibrium. Then look at which test has the most results remaining. After all if two designs are in all ways equal, except that one is cheaper, the cheaper one is objectively better.

## 6.3 Development process

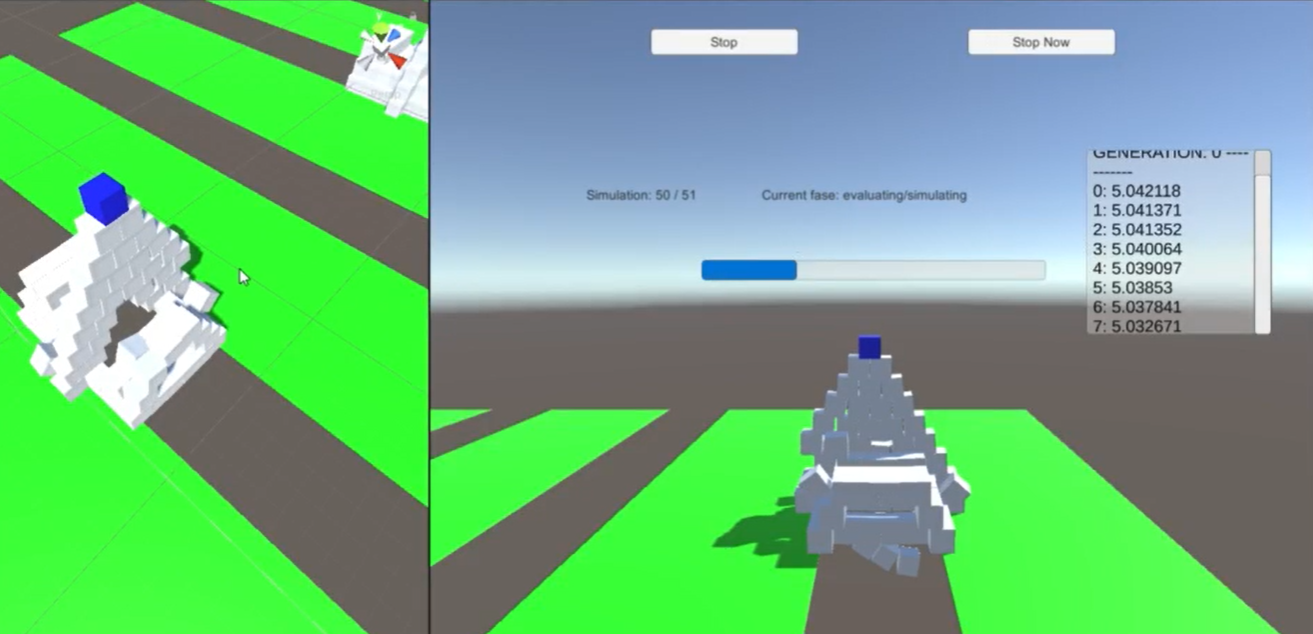
### Unity Prototype

The first prototype is made entirely in the unity game engine. This engine can easily simulate the bridges in a visual way. The code is written in the programming language C#. The part of the code that simulates the bridges uses a lot of unity features. The rest of the code can easily be separated from unity.

The chromosomes represent a 3D grid, in this specific case a 8 by 8 by 8 grid. Each value can be either 1 or 0. Representing the presence (1) or absence (0) of a block.

Three fitness functions are used.

* Cost: this simply is the amount of blocks, less blocks is better. No simulation is needed here, just counting the amount of one’s in the chromosome.
* Carrying: How much the bridge can carry before collapsing. This is tested by creating all the blocks as unity rigidbodies, allowing them to be simulated in unities physics engine. Two stationary planes form the land the bridge is build on. Empty space in the middle represents the water. A block is now dropped on the bridge after a few seconds the program looks if the block is still in the area. In this case the bridge is reset and a heavier block is dropped on it, if not the bridge is considered to have collapsed.
* Stability: The bridge it’s basic stability. The bridge is simulated in the same way as in the carrying simulation. However instead of dropping a block on it the simulation is just ran for a few seconds. During this few seconds the velocity of each induvial block is stored. The combined velocity of all blocks shows how instable the bridge is. The less velocity, the higher the bridge scores on stability.

Figure 1

This prototype is able to generate bridges made from blocks. It did take a lot of generations to do so. Additional some values where tweaked during the evolution progress. Speeding the process up, or significantly increasing the amount of computational power, will be needed for testing multiple strategies.

Unity allows the simulations to be displayed visual, showing not only the bridge, but also the tests. One can actually see a heavy block being dropped on the bridge, and whether or not the bridge collapses.

### Blazor prototype

The next step was separating the program into multiple libraries. This makes it easier to expand the code and conforms to programming standards. This makes adding new simulations, fitness functions or selection functions easier even if they use a different engine or language.  
It also means that the prototype might at some point be upgraded to fully working version of the program.

Separating the code in libraries opened the possibility to run most of the program in a different environment from unity, while still running the simulation itself in unity. The choice was made to use the Blazor framework as the main environment. Blazor is an open source framework that allows building webapps using C#, both serverside and clientside. ([Blazor | Build client web apps with C# | .NET (microsoft.com)](https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor) 15-7-2022)

This means a browser can be used as the user interface (UI). It also will allow better integration with other software used by Volker Infra that already uses Blazor.

The main code, now separated into libraries, does not require Blazor. It can be implemented by any platform or language that can call C# libraries. Some additional code will need to be written to handle the UI and call the libraries.

# 7. Practical Research

After the prototype described in the previous chapter was developed, it has been used to perform several tests. Those tests and the analysis of the results are the practical part of the research. The goal of the tests is to compare different methods of selection to each other. The first two parts of this chapter describe the different types of selection and the setup of the tests. In the third part the results are shown and explained. Finally an analysis is made which is used, along with the literature research, to draw conclusion in chapter 8. Given that this chapter uses the prototype from chapter 6, the same terminology as described in chapter 6.1 terminology is used.

## 7.1 Types of selection

Two primary methods of selection were compared against each other, **summed scores** and **voting**. Several variations of those methods have also been tried out.

Each chromosome is assigned a score by each fitness function, with six fitness functions this means six different scores. A higher score is considered better.

### Summed scores

Summed scores is used as a basic form of selection for Many-Objective Evolutionary Algorithms (MaOEA’s). Two variations have been tested

**Basic Summed Scores:** For every chromosome the score of each fitness function is added together. Each chromosome now has one score that will be used for selection, the summed score. The 30 chromosomes with the highest summed scores are selected.

**Summed Normalized Scores:** For every fitness function every score of the current generation is normalized. The lowest scores for that fitness function in that generation is 0, the highest 1. For every chromosome the normalized score of each fitness function is added together. Afterwards the process is the same as for basic summed scores. The 30 chromosomes with the highest summed normalized scores are selected.

### Voting

**Basic voting:** Consists of two steps, vote-out and vote-in. During vote-out the following happens: For every fitness function a set amount of chromosomes with the lowest scores are selected. A list is made of chromosomes that are not voted-out. A chromosome can be voted-out multiple times, if it has a low enough score for multiple fitness. Because for every fitness function chromosomes can be voted out from the full list, it does not matter which fitness function is looked at first.

During vote-in the following happens: For every fitness function a set amount of chromosomes with the highest scores is selected from the list of chromosomes that are not voted-out. Just like with vote-out a chromosome can be voted in multiple times. This means that amount of unique chromosomes that move to the next generation can vary.  
  
**Voting with larger vote-out:** The same as basic voting, only the set amount of chromosomes that are voted-out is significant larger. The amount of chromosomes that are voted-in remains the same.

**Voting with selective crossover:** While the selection itself is the same as with basic voting, information from the voting process is used during crossover. Chromosomes are not crossed over with chromosomes voted-in due to the same fitness function. The chromosomes are also not copied directly to the next generation. The idea is that this might prevent several groups of chromosomes being optimized for only one or two objectives, instead of finding a solution that compromises between the different objectives.

**Voting “remove inferior children” & selective crossover:** The same as “voting with selective crossover” except for the following: For every fitness function, before the set number of chromosomes is voted out, all chromosomes that have a score that is worse than that of both of it’s parents it gets voted-out. Afterwards the normal voting-out happens.

**Voting “remove inferior children”:** Basic crossover with “remove inferior children”.

## 7.2 Test setup

Beside the type of selection there are a lot of other decisions that had to be made. The amount of tests to be ran, the population, mutation rate as well as specific settings for the different selection functions. Time and computational power were limited which meant the amount of tests, size of the population, size of the design and the number of generations couldn’t be too large.

Three tests have been ran per type of selection. This is an amount that was doable in a reasonable time frame but still makes it unlikely that luck had a significant impact on the results.

For all tests a start population of 30 is used. For the summed scores selections the number of selected chromosomes is set to 30. For the voting based selection 5 chromosomes will be selected for each fitness function, with 6 fitness functions also resulting in 30 selected chromosomes. Some of them might be duplicates if one chromosome is selected due having a high score in multiple fitness functions. During the voting the number of voted-out chromosomes is 15 per fitness function, resulting in 90 if no overlap occurs. This is the same for all voting selections except for voting with large vote out, where for every fitness function 240 chromosomes get voted out. This means a maximum of 1440 chromosomes get voted-out.

Each test will use designs of a length of 9 units, a width of 3 units and a height of 5 units. This is small enough so that a large amount of simulations could be ran simultaneously while still offering a challenge.

A length of 9 means a significant gap has to be crossed, this is likely more challenging than a short gap.

The height of 5 means that there are 3 blocks below and 2 blocks above the surface. This is large enough to allow some form of support structures.

The reason that width is only 3 is that a larger width is unlikely to increase the design challenge, nor provide opportunities for clever designs. While this will be different in a more realistic simulation this simple block based simulation relies primarily on support in the length direction.

The tests will run for 10 generations.

As described in chapter 6 the prototype offers three different settings for mutation. For each gene in a chromosome the chance to mutate is 5 percent. The chance for each row to get swapped with another row is 8 percent. The chance for each row to get copied to another row is also 8 percent.

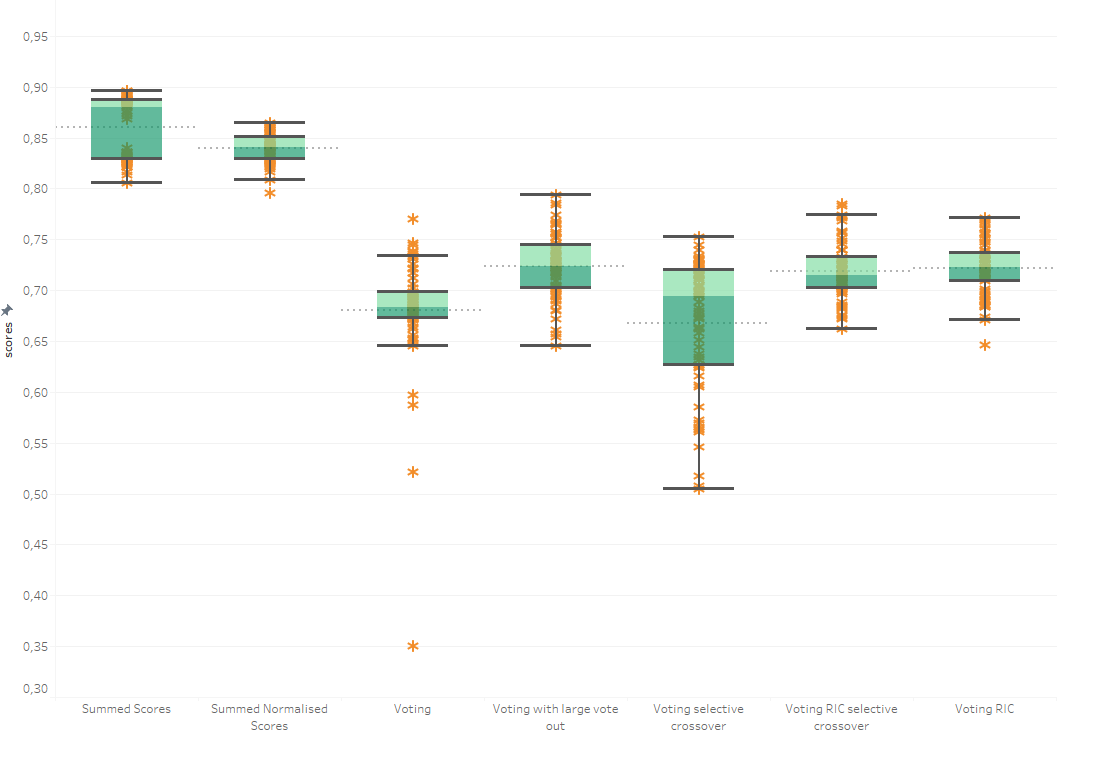
## 7.3 Results

1. The tests generated a lot of data in the form of scores and pictures of simulated bridges. The results for every chromosome consists of six different scores and two images. Given that tens of thousands chromosomes have been evaluated these results needs to be filtered and presented in a workable form in order to be able to analyze it. By only looking at the data from the winning chromosomes of the last generation of each test a massive amount of data can be filtered out. What remains is almost 600 rows of data.
2. There were some minor problems with the data. When no roads, not even partial roads were found several of the fitness functions gave faulty results. The results would either become extremely low, due to a extremely high penalty in the case of the “Weight fitness function” or become NaN (not a number) because division by zero in the case of the “flatness” fitness function. As this only affected a small number of designs and it only affected designs that should get low scores it was deemed unnecessary to fix the bugs and redo all the tests. Instead those fields were manually corrected to significant but not too high penalties (-20 for weight, -10 for flatness). Chapter 2 gives more information of the different objectives and chapter 6 further describes their implementation.
3. As described in chapter x saying which results are better is complicated. However every objective is there for a reason. A design that scores great in 1 objective but very poor in every other can be said to be inferior to a design that scores decent in every objective. The following formula will be used to give the designs an overall score.
4. For every fitness function taking all the scores of the different tests and selection functions and then normalizing them gives an idea how well the design performed for that objective, relative to all other designs. The average normalized score for each objective is then used as an measure of the overall quality of that design.  
   This is similar to the “summed normalized scores” selection function but is different in one key aspect: where the selection function normalizes the set of results of that generation, this normalizes the end results of all different tests.

### 7.3.1 Box and whisker plot

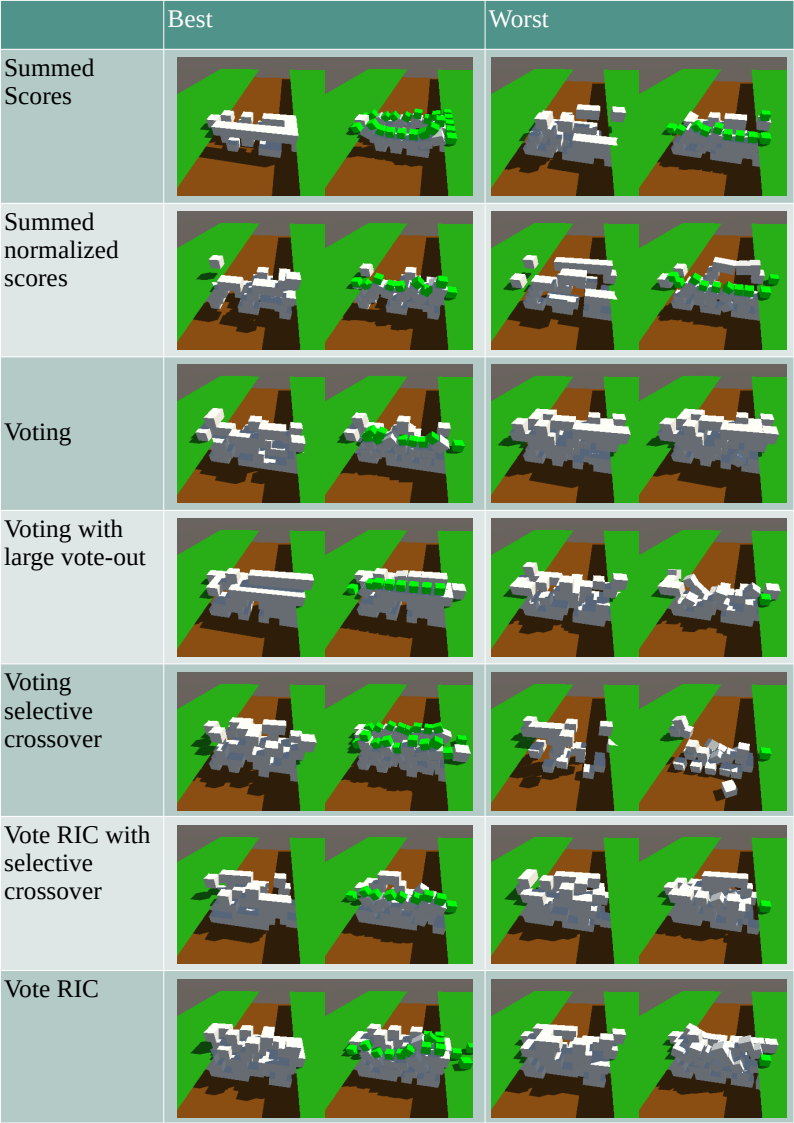
1. Below the result of each method of selection will be displayed in a box and whiskers plot to gives an idea of the relative performance. The box and whisker plot gives an idea of how data is distributed. The orange points are the designs scores of the designs. The dotted line is the average score for that selection method.

https://www.tableau.com/data-insights/reference-library/visual-analytics/charts/box-whisker

Figure 2

### 7.3.2 Table of best and worst bridges

1. As explained in chapter 6 the simulations are visually represented and pictures are taken of each design.
2. To look further into the data some pictures of the simulated bridges will be shown. For each method of selection both the picture of the design with highest score and the design with the lowest score will be shown.

Figure 3

### 7.3.3 Percentage of non-dominated designs

Finally there is the following graph which shows the percentage of designs for every method of selection that are not dominated. Dominated here refers to Pareto dominance. A design dominates another design when it is better in at least one score, and better or equal in every score. A design is considered non-dominated

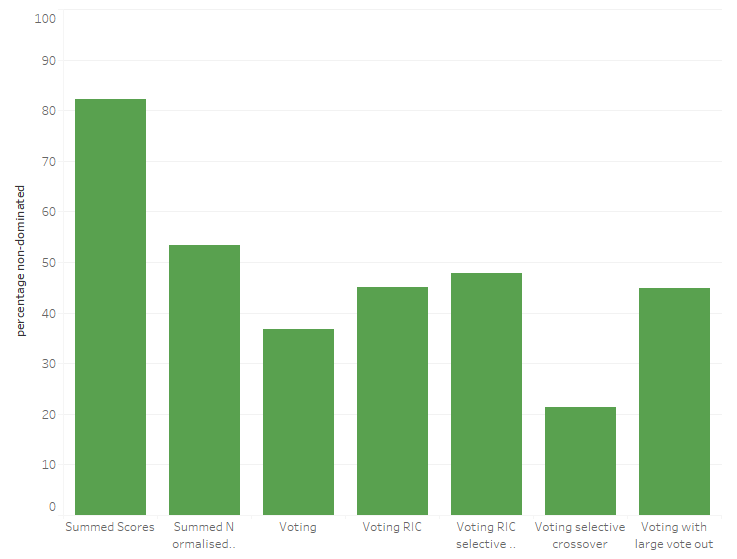


Figure 4

## 7.4 Analysis

1. Analyzing the box and whisper plot in figure 2 a clear difference in quality can be seen between the sum based and the voting based methods of selection (MoS). The average normalized scores (ANS) of the sum and normalized MoS are all rather close together even between test. For sum MoS two clusters can be found. Looking at the median and the average it is likely that the ANS of two tests overlap and form the top cluster, where ANS of the remaining test forms the bottom cluster. For normalized sum MoS the scores all fall in a small range. Summed normalized scores MoS seems to perform slightly worse then summed score MoS.
2. The ANS of the voting based MoS are distributed over a far larger range. The reason for the large distribution is likely because the voting based MoS tends to have designs that score good for a one or a few fitness functions instead of them all. As an example: where one design scores good for cost, the other scores good for stability and yet another for number of roads and flatness. While all these designs are selected as winners the designs ANS score is can be very different. This variation in itself is not a bad thing, after all finding a few good solutions can be enough. However even the best ANS of the voting based MoS only marginally surpass the worst ANS of the sum based MoS.

# 8. Conclusion

# 9. Recommendations

Plaats de aanbevelingen duidelijk apart van de conclusies. Aanbevelingen zijn niet direct afgeleid van je

onderzoeksresultaten, maar vloeien juist voort uit en zijn gebaseerd op je conclusie(s).

# Post Script

In het nawoord kun je persoonlijke evaluatieve opmerkingen kwijt, iets vertellen over de actuele

ontwikkelingen en een blik werpen op de toekomst.

# De noten

Als je van noten gebruik maakt, geef er dan uitsluitend toelichting of commentaar mee op de inhoud van

de tekst. De tekst van de noten moet zelfstandig gelezen worden. Nummer noten doorlopend. Noteer het nummer van de noot achter de betreffende zin, iets boven de regel. Plaats de noot aan het eind van

pagina, hoofdstuk of afstudeerrapport, op deze wijze: Noot 35 ...

Voetnoten plaats je:

of:

- onderaan de bladzijde;

of:

- aan het einde van ieder hoofdstuk;

of:

- aan het einde van je rapport.

In voetnoten kun je informatie kwijt die niet direct noodzakelijk voor de tekst is, zoals literatuurverwijzingen.

Zet geen dingen in je tekst tussen haakjes, maar plaats een voetnoot.

# Bibliography

# De bijlagen (altijd nummeren)

Een bijlage geeft aanvullende informatie aan de lezer die meer wil dan de basisinformatie in de

eigenlijke tekst. De informatie moet natuurlijk wel relevant zijn voor de kern van de afstudeeropdracht.

Beperk het aantal. Als bijlagen kunt je bijvoorbeeld opnemen:

- je reflectie;

- lijst met moeilijke woorden/afkortingen;

- lijst met figuren/afbeeldingen;

- het onderzoeksplan;

- (aanvullende) tabellen

- (aanvullende) berekeningen

- kopieën van vragenlijsten, formulieren, brieven, interviews (de samenvattingen daarvan staan in

de kernhoofdstukken);

- kaarten, tekeningen, schema’s.

Nummer de bijlagen en voorzie ze allemaal van een bondige en exacte titel (ook in de inhoudsopgave

### Traveling salesmen problem

This problem asks the following question: “Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?” (Yu, M. (2019). A solution of TSP based on the ant colony algorithm improved by particle swarm optimization. *Discrete & Continuous Dynamical Systems-S*, *12*(4&5), 979. )

Finding the optimal solution becomes nearly impossible as the number of cities increases. Using brute force, the time it takes to solve it increases with the factorial of the number of cities.  
Better algorithms have been found that can find an exact solution it much faster. It should be noted that there is a difference between a worst-case scenario and the practical time it takes to solve find the solution.

Finding an exact solution is not always necessary, sometimes it’s good enough to find a approximate solution. If an optimal rout exists that 81,2 km and a program finds a solution that is 82,1 km this might be a good enough solution. An evolutionary algorithm can find an approximate solution for the TSP.