Parameter Calibration of Stochastic Degradation Models of Sewer and Water Pipe Networks using Genetic Algorithms

BRAM H.B. OTTENSCHOT, University of Twente, The Netherlands
LISANDRO A. JIMENEZ-ROA, University of Twente, The Netherlands
MARIËLLE STOELINGA, University of Twente, The Netherlands

Modelling the deterioration of sewer and water pipes is vital for drawing up efficient maintenance strategies such that preventive maintenance can be executed at a certain time in the pipes' life cycle when it is needed without having to perform regular, expensive CCTV inspections. One such kind of deterioration model is Homogeneous Discrete-Time Markov Chains, which are stochastic models that can predict the severity of damage by working with states symbolising the severity level, and transition for expressing the time-independent probability of moving from one state to another in discretized time steps. Calibrating these transition probabilities is however not a trivial task. That is why exploring new calibration techniques for deterioration models of sewer systems is useful. This paper focuses on the application of Genetic Algorithms (GA) in calibrating Markov Chains by using evolutionary concepts like selection, crossover and mutation. The result of this research shows promising stable results following the application of the algorithms in the sense that the calibrated models perform well if compared to real-world data, and the fact that the algorithm is stable, e.g. multiple runs converged to similar performing calibrated parameter sets.

Additional Key Words and Phrases: Multi-state degradation modelling, Markov chains, sewer systems, genetic algorithms, optimization algorithms.

1 INTRODUCTION

Sewer pipe networks are important infrastructure systems crucial for preventing overflow in the streets and the disposing of wastewater. Any breaks or failures in said systems can cause public health concerns and environmental problems and are costly to repair [14, 17]. To prevent these system failures, timely and efficient maintenance strategies are necessary.

Among the different maintenance strategies, predictive maintenance promises to achieve long-term strategic planning by harnessing predictive model capabilities. In this way, maintenance strategies can be devised such that any potential failures can be treated timely.

One such model is a Markov chain. It is a discrete-time stochastic model which can simulate degradation processes. It models degradation by having transition probabilities from one state to another. For a Markov chain model to be helpful, these transition probability values should be chosen carefully.

In this paper we will first talk about the motivation of conducting this research, then the goals and objectives we set ourselves following the motivation and then the research question that will aid in fulfilling these goals and objectives. Afterwards, a concise overview of existing literature about topics discussed in the paper is given. Then The methodology of how the research is conducted is provided followed by the results of following the methods. Conclusively the results are interpreted in the conclusion section, and later in the discussion section and we are ending with potential future research.

1.1 Motivation

Getting the optimal parameters for stochastic models is not an easy or trivial task. Tuning the values in the model to obtain an optimal set of parameters is considered a Non-deterministic Polynomial-time (NP-Hard) problem [16]. There is a need to understand and obtain these parameters in a better and more efficient way. By researching new algorithms, we can hopefully get a better insight into how algorithms can streamline the process of making and training a model to ultimately make better models. Hawari et al. [7] also discussed that a limitation to Markov chains was that the transitional probability matrix has to be filled in, with this research we aim to better understand the calibration of these values and perhaps find a new efficient method of doing so in the context of stochastic sewer system models. Using genetic algorithms to solve NP-hard solutions is no new application since it is commonly used in problems such as the travelling salesman problem [20], thus the application in this new context is promising.

1.2 Objective and goals

Counting on accurate and reliable degradation models is crucial for better management of sewer assets, for example, by yielding optimal maintenance strategies. However, calibrating these degradation models is challenging when considering different types of assumptions. So, aiming to identify better strategies to calibrate these degradation models, in this study we would like to investigate an algorithm that has not yet been used in the context of sewer pipe systems that can be used to calibrate these models. This algorithm is the Genetic Algorithm (GA), and we will investigate its usability in the context of predictive stochastic models for the degradation of sewer and water pipe systems.

1.3 Research questions

The objectives and goals of this research lead to the following general research question:

How do Genetic Algorithms perform in calibrating Markov chain models of stochastic degradation in sewer and water pipe networks based on key performance metrics?

2 RELATED WORK

2.1 Genetic Algorithms

The usage of Genetic Algorithms (GA) in complex computer problems was first proposed in a paper from Holland in 1992 [8]. He

TScIT 41, July 5, 2024, Enschede, The Netherlands

translated two important roles in evolution, namely natural selection and sexual reproduction, to computer code by simulating their components. It starts by having a population of several individuals. Each individual, or chromosome, is represented by a bit stream, such that each gene on the chromosome could have the value 0 or 1. The proceedings of the algorithm go as follows: first, the fitness value of each individual is calculated. Based on these fitness values some chromosomes are chosen, which will replicate with each other using genetic operators to make a new population. A more extensive explanation and implementation can be found in section 4.4.

GA belongs to the order of Evolutionary Algorithms (EA), an umbrella term used to describe population-based stochastic direct search algorithms that in some sense mimic natural evolution [2, 23]. EA are also referred to as population-based metaheuristics which means that they utilize multiple candidate solutions during the search process [12]. Among GA, other evolutionary algorithms are Evolutionary Programming [4, 26], Evolution Strategies [3] and Genetic Programming [13].

2.2 Degradation modelling of sewer networks

Modelling the degradation in sewer pipes has been discussed in the literature. Hawari et al. [7] discusses three main types of degradation models. The first type is physical models, in this kind researchers model the real world by recreating the properties of real-life pipes and simulating how they react to phenomena that these pipes may endure. Such models might help by mimicking corrosion in concrete pipes. These deterministic models rest on the assumption that the real-life factors are projected as well as possible. However, this is hard to fulfil since there are too many dependent factors and a shortage of data, making them often too simplistic for a deterministic model [7, 21]. The second type is models that primarily use artificial intelligence. They use neural networks or rule-based models to simulate the degradation of the pipes. The third and last type Hawari et al. talked about are models based on statistics and probability. Stochastic models like Markov Chains and Multiple linear regression. Which focuses on predicting the state of the pipes based on probability and chance.

2.3 Calibration of Markov chain models for degradation modelling of sewer networks

The calibration of the Markov chains that model the degradation of sewer pipes has been done in many different ways. One way to find the optimal parameters is by using non-linear Euclidean-based methods, Jimenez Roa et al. [10] minimized the Root Mean Weighted Square Error (Err) between the data and the model using Sequential Least Squares Programming (SLSQP). Baik et al. [1] studied degradation models of other degradation processes, and compared the different calibration methods used there. They applied the ordered probit model approach, as previously implemented for the bridge deterioration models. This technique addresses the drawbacks of nonlinear optimization-based approaches [1]. Another often-used calibration method is the Metropolis-Hastings Algorithm, a Markov Chain Monte Carlo method. and uses Bayesian logic to approximate the parameters. Among others [15, 22], Jimenez-Roa et al. [11] used the M-H algorithm, but combined it with the SLSQP, to initially find a good estimate for an optimal solution, to subsequently, further improve these with the SLSQP algorithm, avoiding premature convergence.

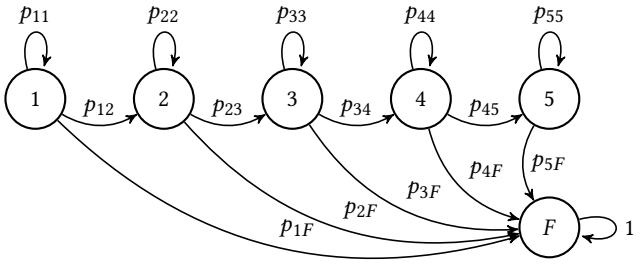


Fig. 1. Markov Chain

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & 0 & 0 & 0 & p_{1F} \\ 0 & p_{22} & p_{23} & 0 & 0 & p_{2F} \\ 0 & 0 & p_{33} & p_{34} & 0 & p_{3F} \\ 0 & 0 & 0 & p_{44} & p_{45} & p_{4F} \\ 0 & 0 & 0 & 0 & p_{55} & p_{5F} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 2. Transition Probability Matrix

3 DISCRETE-TIME MARKOV CHAINS

To test GAs, and their capabilities of calibrating stochastic models, we will use Homogeneous Discrete-Time Markov Chains (HDTMC). These models are discrete-time in the fact that transitions happen in predetermined time steps so they cannot be simulated over continuous time. Furthermore, homogeneous means that the conditional probability of any future events depends only on the present states[1]; in other words, the transition probabilities in the model do not change over time. The Markovian property can be expressed as follows

$$\begin{aligned} P(X_{n+1} = i_n + 1 | Xn = i_n, X_{n-1} = i_{n-1} \\ , ..., X_0 = i_0) \\ = P(X_{n+1} = i_{n+1} | X_n = i_n) \end{aligned} \tag{1}$$

Because the Markov chain is homogeneous, and the probabilities do not change over time, $P(X_{n+1} = i_{n+1}|X_n = i_n)$ is independent of time-steps n,

$$p_{ij} = P(X_{n+1} = i_{n+1}|X_n = i_n), \tag{2}$$

where p_{ij} is the transition probability given a system in state i a time-step n, which is transitioning to a system that will be in state j in time-step $n + 1$. The transitions are most commonly expressed as entries in a Transition Probability Matrix (TPM) $\mathbf{P}$, which is a $m \times m$ matrix where m is the number of states in the model. Since the transitional values are probabilities, all outgoing transitions from a state must add up to one. This could be expressed as the summation of each entry in a row in the TPM like in Table 3.

$$\sum_{i}^{j} P_{ij} = 1 \tag{3}$$

An HDTMC system models the probabilities of transition of a one-time step. To determine the probabilities of any time steps n we need to use the Chapman-Kolmogorov equation,

$$\mathbf{S}^{(n)} = \mathbf{S}^{(0)}\mathbf{P}^{(n)} \tag{4}$$

where $\mathbf{S}^{(0)}$ is the initial state vector which indicates the probabilities of a system being at a state at step $n = 0$, in our case we assume that at the start of their life cycle, each pipe is in pristine condition, so $\mathbf{S}^{(0)} = [1, 0, 0, 0, 0, 0]$. Equation 4 results in a probability vector giving the probability of the system being at each state at step n.

4 METHODOLOGY

To explore the possibility of applying GA to the calibration of DTMCs, we will first lay out the different factors and steps of the process, starting with the data that is used for calibrating, then moving on to how different genetic operators (chromosome encoding, mutation, crossover and selection) are implemented in GA and what the process looks like in practice for this research. And how the results will be interpreted

4.1 Data preparation

The data used to train the model comes from a case study about the sewer pipe network in the city of Breda, Netherlands. It consists of more than 25 thousand pipes built from 1950 onwards. The data is gathered by CCTV inspection where pipes are inspected from the inside. Most pipes are inspected at most once and a minority are inspected twice or more. The data observations are based on the European standard for outdoor sewerage visual NEN-EN 13508-2, where the condition of the sewer pipes is assessed on different defects on a scale from 1-5, and are then reported. The data that will be used is focused on the defect with the damage code BAF, which is surface damage. The pipes themselves are made out of concrete and transport mixed contents and wastewater [10]. The data is transformed into a list of inspections with the precise age of the pipe and the reported severity level. Since we are working with discrete-time MCs, We have to group these observations in discrete time steps. So a frequency table is created, shown in Table 1. Here the inspected pipes are grouped based on their age, and the state of the pipes is reflected by the frequency of each state. Together with a count which stands for the number of pipes in each age group.

4.2 Markov chain structure

In this paper, we will use the degradation model suggested by [11], which has 5 regular states, and one failure state F. The system is a degradation model with the assumption that there are no repairs meaning that a system cannot move to a status that represents a better condition than its current state. Each state has a transition to the next-following state, and the failure state; so the sewer pipe condition can remain the same, worsen with 1 severity level, or fail in a certain time period. The failure state is absorbing, meaning that once the system has reached state F, it will remain in state F This Markov chain structure is shown in Figure 1, with the corresponding TPM in Figure 2.

Table 1. Frequency Table of Inspection Data

Time	State						count
	1	2	3	4	5	F	
1	0.94	0.04	0.00	0.01	0.00	0.00	591
3	0.94	0.04	0.00	0.00	0.00	0.02	136
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
47	0.26	0.58	0.14	0.02	0.00	0.01	1135
49	0.40	0.48	0.09	0.02	0.01	0.01	1485
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
71	0.75	0.25	0.00	0.00	0.00	0.00	4
77	0.00	0.75	0.25	0.00	0.00	0.00	4

4.3 GA components

4.3.1 Chromosome Encoding. The first forms of GA used binary encoded chromosomes, where each gene was represented as a binary 0 or 1. So a chromosome either has the gene or not. This binary encoding scheme can also be translated to real values by taking the binary value of the bit stream. This bit-wise representation makes it so that genetic operators can be implemented in a way that is more similar to how Darwinian evolution describes it. Another way to encode the parameters is by simply taking the parameters as real values, and treating these real values as separate genes. A disadvantage of value encoding is that the genetic operators have to be slightly modified since the genes are not binary anymore [24].

We want to calibrate a Markov Chain by finding optimal values for the probabilities P_{ij} of each transition between stat i and j. The values are real numbers and probabilities, so they are bounded by interval $[0, 1]$, which means that if these values were encoded in binary we would need many bits, and it would be harder to program it. Since according to 3 the values of each row of a TPM must add up to 1, we can write all transitions P_{ii} as subtraction of all other transitions in the same row i. For example $P_{11} = 1 - P_{12} - P_{1F}$. This leaves us with 9 parameters to calibrate. Namely all non-diagonal, non-zero entries in the TPM $\mathbf{P}$. In the context of GA, this means that the chromosome will exist out of 9 genes which will be optimized.

4.3.2 Mutation. The first genetic operator that GA uses is mutation. Mutation ensures that solutions escape certain local optima by altering the genes on the chromosome. In Binary encoding, this means flipping certain bits. with a probability called the mutation rate for binary coded solution: every bit has a certain change to be changed. [24]

for real coded chromosomes a certain distribution and bounds should be applied to alter a specific gene on the chromosome. The bounds could be set, numerical values describing the maximum change. It is accompanied by a certain mutation rate, a probability which determines if a gene gets mutated. This could be a normal distribution to favour smaller changes. Or a more uniform distribution to encourage larger changes to escape certain local optima One such mutation algorithm using a uniform distribution is proposed in a paper written by Wright, he proposed a mutation involving a mutation rate and a maximum mutation size [24]. We implement

this by setting these mutation values to 0.5 below the current value and 0.5 above the current value, such that the genes can escape certain local optima quicker.

4.3.3 Crossover. Crossover, or recombination, is the operator which ensures that when two parent chromosomes are mating, the new solution will have genes from both parents, with the premise that exchanging genes from two fit parents will inherently produce even better offspring [27]. In the simplest form of crossover, certain breakpoints are chosen on the chromosomes which divide the genes in parts. These parts are exchanged between the parents, and end up creating new children's solutions. This method is called single-point crossover when one breakpoint is selected, and multi-point or K-point crossover for more breakpoints.

4.3.4 Fitness function. a fitness function evaluates all individuals from the population and gives them a fitness value. In the context of parameter calibration of Markov chain models, the fitness function assesses how well the parameters in the Markov chain model the real-life data.

the fitness function that will evaluate the individuals in this research is root mean weighted squared error (Err)[10]. This function creates a TPM from a proposed solution. And uses the Chapman-Kolmogorov equation to evaluate the probability of a pipe being in a certain state. For each time step Δt and state S_m this probability is compared to the real data from the frequency table weighted to the amount of pipes that are present in each time step. Since the differences of the fitness value between individuals differ only a fraction of the values in advanced stages of the algorithm, the fitness value gets transformed to $score$

$$score = e^{\frac{1}{Err}} \tag{5}$$

where e is raised to the power of the inverse of Err Such that a small difference in the Err value of a solution results in a big difference in the score value, thus promoting improvements of the solutions even more. The library PyGAD [6] only offered the possibility to evaluate the fitness function by rewarding higher fitness values, whereas in our case a lower Err would mean a better solution. After some test runs this method of calculating $score$ worked better than only taking the inverse $1/Err$, or negating the fitness value $-Err$.

4.3.5 Parents Selection Techniques. After each individual has been given a certain fitness value, some individuals need to be chosen to be used for mating in the next generation. To ultimately eliminate bad solutions from the population and keep better solutions apply the different selection techniques to each member of the population and their fitness value. An intuitive first approach is to assign a certain probability to be selected to each organism relative to their fitness value. This is called the roulette wheel approach where each individual gets assigned a portion of the roulette wheel based on their fitness value, and then the wheel is spun N times, based on how many parents are needed.[25].

Roulette selection might induce premature convergence of the GA to a local optimum instead of a global optimum [9], so to combat this, and give a lesser fit solution a better chance of getting selected, the rank selection method can be used. Instead of assigning probabilities based on fitness value, they are assigned based on the rank of the solutions if they are arranged based on fitness value. rank - each solution gets a rank based on its fitness value, and the chance of getting selected is proportional to its rank. The roulette is spun as often as needed to select the required number of parents.

Another way to combat premature convergence is by using Stochastic Universal Sampling (SUS) similar to the roulette wheel, the solutions are placed on the wheel based on fitness level, but instead of spinning the wheel multiple times, N evenly spaced pointers are placed on the wheel. This selection procedure also has no bias since the individuals are selected solely on their positions in the population, or in this case on the roulette wheel [18].

4.3.6 GA parameters. A GA also needs initialisation, a good population size has to be chosen, and the mutation probability must be set. Determining the right size of this population is vital for the convergence and the performance of the process. choose a population size too small, like 10, and the gene pool is simply not diverse enough, choose a population size too high, like 4000, and the computational cost will be too great, The magnitude of the search space also plays a role in choosing the population size, since a larger search space might benefit from a larger gene pool of different values. Another parameter is the mutation probability, which determines how many genes will be mutated. A mutation prevents the algorithm from converging to local optima, but setting the mutation probability to high will result in a random search. [5] Chiroma et al. [5] conducted a survey in which they researched different GA processes and noted down these 2 parameters. From this, we see that a population size of 100 is deemed a good size since it is a good tradeoff between gene diversity and computational feasibility. And a mutation rate of 0.3 is chosen to support more change during the iterations.

4.4 Calibrating a model

To implement all the chosen decisions, we opted to program in Python since previous similar projects have used the language, and thus certain resources could be re-used. Additionally, it is an easy-to-use programming language and the go-to language in machine learning because of the available low-level libraries and performance [19]. For the implementation of GA in Python a handful of libraries exist that all enable the user to implement a GA instance. Most libraries are frameworks in which the user can set up any evolutionary algorithm, like in Pyevolve, DEAP and LEAP. Compared to some libraries focusing solely on GA, such as PyGAD and EasyGA. Both categories of libraries had roughly the same functionalities in setting up GA parts of the other libraries, but the GA-focused ones were easier to set up and required less unnecessary code. From the two selected libraries, PyGAD has more genetic operations options and better follows the textbook definition of a GA. That is why we utilise the PyGAD library [6] to implement the the genetic algorithm.

The hyperparameters, as discussed in section 4.3.6, are set in the model, and the GA starts by initializing the genes of the chromosomes by randomly picking a value that is on the gene space, which in this case are probabilities, thus being in the closed interval $[0, 1]$. The fitness function gives all the individuals a fitness value, Err, as described in section 4.3.4. From these values, 40 parents are selected

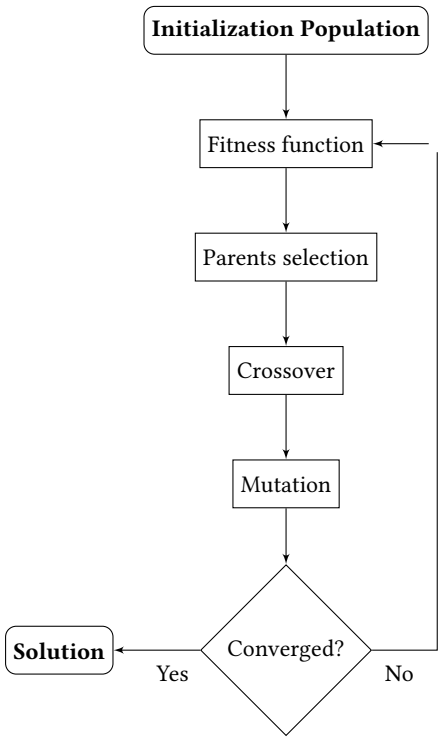


Fig. 3. Flowchart Genetic algorithm

by the SUS technique described in 4.3.5 for replication. These parents then perform a two-point cross-over and random mutation to create a new population. These newly created individuals all get a fitness value and the cycle starts over again. These steps iterate until the GA has converged, so if the fitness value of the best solution has not changed in 50 iterations or once 500 iterations have passed. The whole process is displayed in Figure 3.

4.5 Interpreting Solution

After the GA process has stopped due to convergence, or the iteration limit has been reached. The solution with the highest fitness function is picked from the population as an optimal solution. To evaluate the performance of this found solution and the calibration process, some metrics should be in place to systematically assess it. The metrics include:

- **Convergence time**. How many iterations does the algorithm need before the fitness level has stagnated, and hasn't changed for 50 iterations?
- **Accuracy**. The fitness value that the final best solution gets using the *Err* [10].
- **Stability**. How much does the final fitness value vary on different runs, using the same data?

5 RESULTS

The primary objective of this research was to research the effectiveness and performance of Genetic Algorithms in calibrating and optimizing the transitional probability matrix from Homogeneous

Table 2. Results from the Calibration Runs

Run	Score ($\times 10^{18}$)	*Err*	generations till convergence
1	4.1667	0.0233243516326512	500
2	5.3658	0.0231875564252668	284
3	7.0761	0.0230397483259682	273
4	5.7962	0.0231461487657747	500
5	4.5758	0.0232734977879196	272
6	4.8134	0.0232461142403455	500
7	5.4716	0.0231770607504200	269
8	5.4241	0.0231817478766909	234
9	6.8970	0.0230533637959656	414
10	5.2267	0.0232016893348690	500

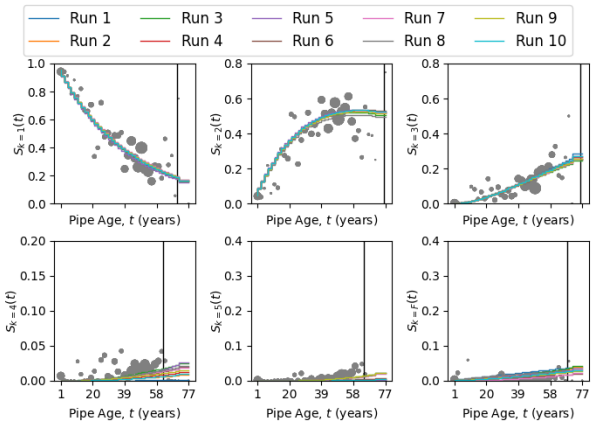


Fig. 4. Evaluation of Parameters

Discrete-Time Markov Chains such that it reflects the real-world data as best as possible. We will do this with the data described in section 4.1. To test the stability of the calibration technique we ran the process 10 times, in which each run, the GA was set up with the same data and parameters. The data recorded from these 10 runs were the *score* (Equation 5), the *Err* and the number of generations until convergence, and are shown in Table 2.

After convergence, the *Err* values of the final solutions average 0.0232 with a standard deviation of $8.44067E-05$. To better illustrate the fitness values of each solution, the proposed final solution with the highest fitness of each run has been plotted together with the real-world data points in Figure 4. In the graph, for each state, the probability of the system being at that state at a certain time step is shown. The real-world data is graphed as grey dots, following the distribution from the Frequency Table 1, where their size reflects the amount of inspection data in that time step.

To get a better insight into how GA converge over time, the fitness levels of the best solution in each generation are plotted in Figure 5. In the figure, the plotted line stops when it has converged. In Figure 5a the score is plotted against the generation, and in Figure 5b the actual Fitness value of *Err* is plotted against the generation.

In this figure we can see that most major fitness level improvements occur in generations 0 to 50; from generations 51 to 150 smaller improvements occur and from generation 151 onwards, only minor improvements happen.

6 CONCLUSION

With the newfound result, we can give an informed answer to our research question: *"How do Genetic Algorithms perform in calibrating Markov chain models of stochastic degradation in sewer and water pipe networks based on key performance metrics?"*. Overall we can say that Genetic Algorithms are a viable option in the calibration of stochastic models like HDTMCs since the resulting calibrated models, model real-world data well, and the algorithm produces stable outcomes with similar, fit solutions over multiple runs. Most calibrations have found a rough optimum as a solution between 100 and 200 generations, after which this solution gets fine-tuned with smaller improvements.

A model with an optimal calibration should follow the real-world data as well as possible. In Figure 4, most plotted staircases follow the plotted points well. The fitness scores of the runs reflect this by indicating that the distance between the predicted point (the plotted line) and the real-world data points (the grey dots) is small.

The stability of an algorithm says something about how similar the results are if the algorithm is run multiple times. A good way to measure this is by looking at the standard deviation of the results of the calibration, the fitness scores. Such a small standard deviation in the Err value shows that the GA is a relatively stable algorithm for calibrating HDTMCs. This is further substantiated in Figure 4, where all plotted runs follow roughly the same trajectory.

On the number of generations, each GA calibration run used can be said less, Because the GA method is still based on a degree of luck, some runs might reach an optimum earlier than others, hence the greatly differing convergence times. This is further founded by the fact that in general, runs with a higher iteration count do not necessarily have a better fitness value. For example, the run with the best fitness score only needed 273 generations to reach it, and the run with the (relatively) worst fitness score took all 500 generations. What can be said about the fitness vs generations plotting as shown in Figure 5b is that the majority of the fitness improvements are made early on in the calibration process, within the first 150 generations, after which most runs were near the optimum solution. Most graphs have settled around $Err \approx 0.024$ between 100 and 200 generations. And the later generations only improved their fitness marginally. This implies that a calibration process using GA with the discussed parameters does not need all 500 generations, but has sufficient time to converge to a good set of parameters in 200 generations.

7 DISCUSSION

In the current state of the GA setup, combined with the limited computational power of the technical setup, one iteration of the algorithm takes up to 6 seconds, which means that running the algorithm for 500 iterations already takes up close to an hour. We would like to have tested the algorithm for more than 10 runs, but due to time constraints, this was not achievable. The numbers from Table 2 show that some runs of the algorithm used all 500 iterations, meaning that there was still potential for them to improve further. Combing this knowledge with the fact that in some runs increases of the fitness level can happen, even after a period of less increase in the fitness level, perhaps if more iterations were allocated to the algorithm, it would perform even better.

Furthermore, it is proven that some inhomogeneous Markov chain models (or semi-Markov chains) better model the degradation of sewer pipes [11]. These structures however deemed too difficult to adapt to genetic algorithms since the rates that had to be calibrated had too large a solution space, and any attempt to apply GA would result in poor solutions which got stuck on poor local optima far away from the global optima.

8 FUTURE WORK

While researching the topic, and testing the GA algorithm, many more research directions emerged as the research went on. However unfortunately due to the time constraints that were put on this research, most of these emerging questions were left unanswered.

Another direction could be to explore how GA performs when used on different Markov chain structures, perhaps with more or less transitions. This can be extended by using different Markov chain models like inhomogeneous where the transition is dependent on time in contrast to HDTMC. Or MCs using continuous time to evaluate the transitional probabilities instead of using set discrete time steps.

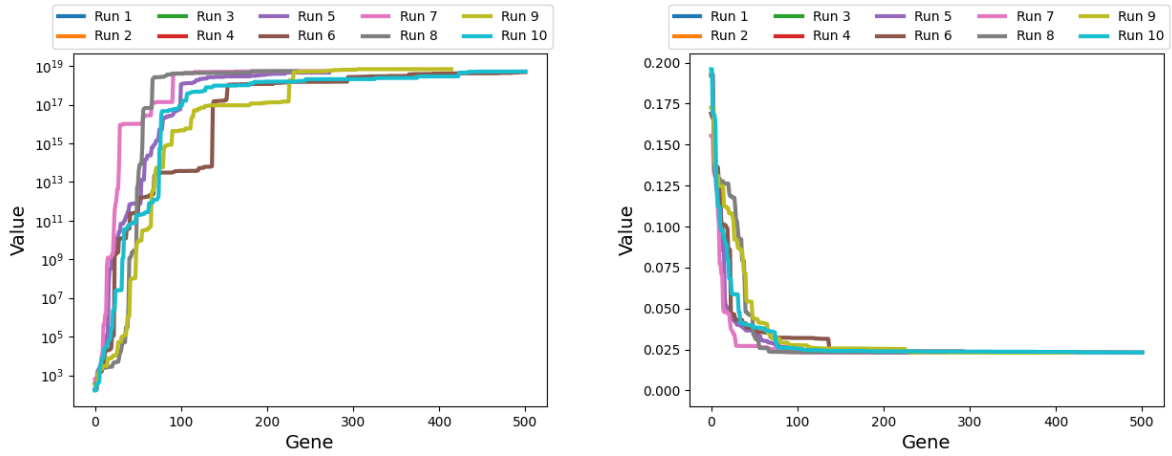
Finally, research that would be helpful is to test a multitude of used calibration methods (including, but not limited to the ones described in section 2.3) in the context of stochastic models for sewer pipe networks. And benchmark them against each other based on certain key metrics.

ACKNOWLEDGMENTS

The main author of this paper wants to thank Lisandro Jimenez-Roa and Mariëlle Stoelinga for their guidance, supervision and feedback during the period the research was conducted.

REFERENCES

[1] Hyeon-Shik Baik, Hyung Seok Jeong, and Dulcy M Abraham. 2006. Estimating transition probabilities in Markov chain-based deterioration models for management of wastewater systems. *Journal of water resources planning and management* 132, 1 (2006), 15–24.
[2] Thomas Bartz-Beielstein, Jurgen Branke, Jorn Mehnen, and Olaf Mersmann. 2014. Evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4, 3 (2014), 178–195.
[3] Hans-Georg Beyer and Hans-Paul Schwefel. 2002. Evolution strategies–a comprehensive introduction. *Natural computing* 1 (2002), 3–52.
[4] YJ Cao and QH Wu. 1997. Evolutionary programming. (1997), 443–446.
[5] Haruna Chiroma, Sameem Abdulkareem, Adamu Abubakar, Akram Zeki, Abdulsalam Gital, and Mohammed Usman. 2013. Correlation Study of Genetic Algorithm Operators: Crossover and Mutation Probabilities.
[6] Ahmed Fawzy Gad. 2021. PyGAD: An Intuitive Genetic Algorithm Python Library. *CoRR* abs/2106.06158 (2021). arXiv:2106.06158 https://arxiv.org/abs/2106.06158
[7] Alaa Hawari, Firas Alkadour, Mohamed Elmasry, and Tarek Zayed. 2020. A state of the art review on condition assessment models developed for sewer pipelines. *Engineering Applications of Artificial Intelligence* 93 (2020), 103721. https://doi.org/10.1016/j.engappai.2020.103721
[8] John H. Holland. 1992. Genetic Algorithms. *Scientific American* 267, 1 (1992), 66–73. http://www.jstor.org/stable/24939139
[9] Khalid Jebari, Mohammed Madiafi, et al. 2013. Selection methods for genetic algorithms. *International Journal of Emerging Sciences* 3, 4 (2013), 333–344.



(a) Convergence of Score

(b) Convergence of Err

Fig. 5. Fitness Convergence of Algorithm Runs

[10] Lisandro A. Jimenez-Roa, Tom Heskes, Tiedo Tinga, Hajo J.A. Molegraaf, and Marielle I.A. Stoelinga. 2022. Deterioration modeling of sewer pipes via discrete-time Markov chains: A large-scale case study in the Netherlands. In *Proceedings of the 32nd European Safety and Reliability Conference (ESREL 2022)*. 1299–1306. https://doi.org/10.3850/978-981-18-5183-4_R22-13-482-cd 32nd European Safety and Reliability Conference, ESREL 2022 : Understanding and Managing Risk and Reliability for a Sustainable Future, ESREL ; Conference date: 28-08-2022 Through 01-09-2022.

[11] Lisandro A. Jimenez-Roa, Tiedo Tinga, Tom Heskes, and Marielle Stoelinga. 2024. Comparing Homogeneous and Inhomogeneous Time Markov Chains for Modelling Degradation in Sewer Pipe Networks. In *Advances in Reliability, Safety and Security*, Vol. Part 6 - Complex Systems and Critical Infrastructures Reliability, Safety and Security Modelling and Optimization. Polish Safety and Reliability Association, 87–96.

[12] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications* 80 (2021), 8091–8126.

[13] John R Koza. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and computing* 4 (1994), 87–112.

[14] Carles Mateu Marc Ribalta, Ramon Bejar and Edgar Rubión. 2023. Machine learning solutions in sewer systems: a bibliometric analysis. *Urban Water Journal* 20, 1 (2023), 1–14. https://doi.org/10.1080/1573062X.2022.2138460

[15] Tom Micevski, George Kuczera, and Peter Coombes. 2002. Markov Model for Storm Water Pipe Deterioration. *Journal of Infrastructure Systems* 8, 2 (2002), 49–56. https://doi.org/10.1061/(ASCE)1076-0342(2002)8:2(49)

[16] Sajjad Nematzadeh, Farzad Kiani, Mahsa Torkamanian-Afshar, and Nizamettin Aydin. 2022. Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases. *Computational Biology and Chemistry* 97 (2022), 107619. https://doi.org/10.1016/j.compbiolchem.2021.107619

[17] Titilayo A. Owolabi, Saeed R. Mohandes, and Tarek Zayed. 2022. Investigating the impact of sewer overflow on the environment: A comprehensive literature review paper. *Journal of Environmental Management* 301 (January 2022), 795–825. https://doi.org/10.1016/j.jenvman.2021.113810

[18] Tania Pencheva, Krassimir Atanassov, and Anthony Shannon. 2009. Modelling of a stochastic universal sampling selection operator in genetic algorithms using generalized nets. In *Proceedings of the tenth international workshop on generalized nets, Sofia*. 1–7.

[19] Sebastian Raschka, Joshua Patterson, and Corey Nolet. 2020. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information* 11, 4 (2020). https://doi.org/10.3390/info11040193

[20] Noraini Mohd Razali, John Geraghty, et al. 2011. Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the world congress on engineering*, Vol. 2. International Association of Engineers Hong Kong, China, 1–6.

[21] Comfort Salihu, Saeed Reza Mohandes, Ahmed Farouk Kineber, M. Reza Hosseini, Faris Elghaish, and Tarek Zayed. 2023. A Deterioration Model for Sewer Pipes Using CCTV and Artificial Intelligence. *Buildings* 13, 4 (2023). https://doi.org/10.3390/buildings13040952

[22] Huu Dung Tran, BJC Perera, and AWM Ng. 2010. Markov and neural network models for prediction of structural deterioration of storm-water pipe assets. *Journal of Infrastructure Systems* 16, 2 (2010), 167–171.

[23] Pradnya A Vikhar. 2016. Evolutionary algorithms: A critical review and its future prospects. In *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)*. IEEE, 261–265.

[24] Alden H. Wright. 1991. Genetic Algorithms for Real Parameter Optimization. Foundations of Genetic Algorithms, Vol. 1. Elsevier, 205–218. https://doi.org/10.1016/B978-0-08-050684-5.50016-1

[25] Saneh Lata Yadav and Asha Sohal. 2017. Comparative study of different selection techniques in genetic algorithm. *International Journal of Engineering, Science and Mathematics* 6, 3 (2017), 174–180.

[26] Xin Yao, Yong Liu, and Guangming Lin. 1999. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation* 3, 2 (1999), 82–102.

[27] Farah Ayiesya Zainuddin, Md Fahmi Abd Samad, and Durian Tunggal. 2020. A review of crossover methods and problem representation of genetic algorithm in recent engineering applications. *International Journal of Advanced Science and Technology* 29, 6s (2020), 759–769.

A USAGE OF AI

During the preparation of this work, the authors used Grammarly in order to correct minor spelling and grammar mistakes. After using this tool/service, the authors reviewed and edited the content as needed and takes full responsibility for the content of the work.