Universiteit Leiden

Master Computer Science

Network-driven analysis of evolving populations in genetic algorithms

Name:	Virgil Woerdings
Student ID:	s1620495
Date:	22/12/2022
Specialisation:	Bioinformatics
1st supervisor:	Dr. A.V. Kononova
2nd supervisor:	Dr. F.W. Takes

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Genetic algorithms (GAs) are a popular optimisation strategy. Despite a lack of theoretical understanding of their behaviour, interactions, and dynamics, GAs are widely applied to real-world problems. Network-driven analysis of GAs and other population-based algorithms can be used to study population dynamics. We present a network representation, specific to real-valued GAs. We show that we can study GA dynamics through the analysis of topological features of these network representations. We analyse how changes in topological features relate to changes in GA parameters. Moreover, we present both a generative and a randomisation-based null model to quantify the significance of the patterns we observe. We find that an increase in parent selection pressure increases connectivity and decreases average path lengths in our representations. Unexpectedly, this also decreases fitness assortativity. Our results show that network-driven analysis can greatly improve our understanding of the selection of GA parameters. Finally, we propose several points of interest for future research.

Contents

1 Introduction **1**

2 Background and Related Work **2**
2.1 Genetic algorithms . . . 2
2.2 Tuning and Configuring Evolutionary Algorithms . . . 3
2.3 Network Science . . . 4
2.4 Random Networks as Null Models . . . 4
2.5 Applying Networks Analysis to Population-based Algorithms . . . 5

3 GA Data **6**
3.1 Objective Functions . . . 6
3.2 GA parameters . . . 7
3.2.1 Mutation . . . 7
3.2.2 Crossover . . . 7
3.2.3 Parent Selection . . . 7
3.2.4 Correction . . . 7
3.2.5 Population Size . . . 8
3.2.6 Additional GA settings . . . 8
3.3 GA Variant Names . . . 8

4 Methods **10**
4.1 Constructing Networks from Genetic Algorithms . . . 10
4.2 Network Topological Features . . . 12
4.2.1 Assortativity . . . 12
4.2.2 Centralisation . . . 12
4.2.3 Laplacian Eigenvalues . . . 13
4.2.4 Leaf-leaf Distance . . . 13
4.2.5 Treeness . . . 14
4.2.6 Shortest Path Length . . . 15
4.2.7 Subtree Size . . . 15
4.3 Random Null Models . . . 15
4.4 Visualisation of Topological Time Series . . . 16

5 Results **17**
5.1 Random Function f_0 . . . 17
5.1.1 Effect of Network Size . . . 17
5.1.2 Effect of Population Size . . . 18
5.1.3 Effect of Parent Selection . . . 21
5.1.4 Effect of Mutation . . . 22
5.1.5 Effect of Crossover . . . 22
5.1.6 Effect of Correction . . . 23
5.2 Shuffled Networks on Function f_0 . . . 25
5.3 Sphere Function f_1 . . . 27

6 Discussion **31**

7 Conclusion **32**

References **33**

Appendices **37**

A GA Performance Plots **37**
A.1 GA Performance on f_0 . . . 37
A.2 GA Performance on f_1 . . . 42

B Network Pictures **47**

C Plots for Random Function f_0 **54**
C.1 Effect of Population Size 54
C.2 Effect of Parent selection 57
C.3 Effect of Mutation 60
C.4 Effect of Crossover 63
C.5 Effect of Correction (Population Size 5) 66
C.6 Effect of Correction (Population Size 20) 69
C.7 Effect of Correction (Population Size 100) 72

D Shuffled networks **75**

E Plots for Sphere Function f_1 **78**

F Optimum Coordinates of f_1 **81**

1 Introduction

Population-based metaheuristic algorithms such as evolutionary algorithms and swarm-based algorithms are used in different fields to solve complex optimisation problems. These algorithms are often applied when finding an approximate optimum is favoured over an exact solution, usually due to the computational complexity or computational time required to find exact solutions. However, population-based algorithms sometimes exhibit unexpected and undesirable behaviour, like premature convergence, lack of convergence, or a bias towards part of the search space [1, 2]. Because the causes of such behaviour are not well understood, it can be hard to select an appropriate algorithm or parameters in real-world applications. Recently, researchers have tried modelling interactions within population-based algorithms as networks in an attempt to gain a deeper understanding of the dynamics in and behaviour of simulated populations [3, 4, 5, 6].

In these network-based approaches, the active population of an algorithm is represented as a network. Exchanges of information that occur within these populations are expressed as connections in the resulting network representation. Consequently, the population's interactions can be analyzed and quantified by calculating the topological features of the network representations. However, compared to other population-based algorithms, GAs use a unique set of operators to create new solutions. While previous methods of representation are able to express population interactions and can be used to improve algorithm performance, they may not be suitable to express how genetic information is inherited in GA populations.

In this thesis, we describe an approach to representing and analysing real-valued genetic algorithms (GAs) as networks. We show that algorithmic behaviour is reflected in these networks by quantifying and analysing topological features. We speculate that network-orientated approaches will greatly improve our understanding of the consequences of algorithm and parameter selection.

The rest of this thesis is structured as follows:

- Section 2 summarises relevant background information and the state of the art in genetic algorithms, network science, and the combination of the two fields.
- Section 3 describes the GA data that are analysed in this thesis.
- Section 4 details the methods used to create and analyse GA network representations.
- Section 5 presents and discusses the results of the GA network analysis.
- Section 6 discusses the strengths and weaknesses of the suggested approach and makes recommendations for future research.
- Section 7 summarises the main arguments presented in this thesis.

2 Background and Related Work

Before we explain and analyse our network representations of genetic algorithms, we review relevant concepts, background information, and previous research in this section. Firstly, we explain the theory behind genetic algorithms. Secondly, we summarise contemporary methods used to analyse and improve GA performance. Thirdly, we explain how network theory can reveal the mechanisms behind complex systems. Fourthly, we discuss how random networks are used as null models to establish a baseline for network analysis. Finally, we discuss state-of-the-art methods that combine methods of network science with genetic algorithms and other population-based algorithms to study algorithm dynamics.

2.1 Genetic algorithms

Genetic algorithms (GAs) belong to a class of optimisation algorithms, inspired by biological processes. GAs apply evolutionary operators like parent selection, mutation, crossover, and fitness selection to a population of chromosomal representations of potential solutions to find new and better solutions [7]. Depending on the properties of the problem, GAs can be applied either to discrete problems where solutions are encoded using strings of bits or integers as chromosomes, or to continuous problems with real-valued chromosomes.

More specifically, GAs belong to the class of evolutionary algorithms (EAs). These algorithms are used in a heuristic setting where instead of searching for an exact solution of an optimisation problem, one is looking for a solution that is good enough but not necessarily the global optimum. Typically, such a setting is used for objective functions without an analytical expression, known as black-box problems, or for functions that are to complex to solve exactly. EAs belong to an even bigger class of population-based algorithms in which multiple candidate solutions are maintained and updated, usually using some interaction between the solutions.

Figure 1 shows a general schematic representation of a discrete genetic algorithm. Each chromosome in an initial population is assigned a fitness based on the objective function. Every generation, crossover, and mutation operators are applied to a selection of a population of chromosomes in an attempt to produce more fit individuals. If successful, more fit individuals replace less fit individuals in the population. Because GAs are often applied to optimise functions that are complex or expensive to evaluate, the algorithm can either terminate after a maximum number of fitness evaluations or when an individual of sufficiently good fitness has been generated. In this thesis, we only discuss continuous GAs, which are described in more detail in Section 3.

Algorithm 1 shows a summary of a general genetic algorithm. It starts with the algorithm's initialisation (Line 2). An initial population is generated, typically randomly. The algorithm then enters a loop where, each iteration, it typically selects two parents (Line 4-5) and creates a new

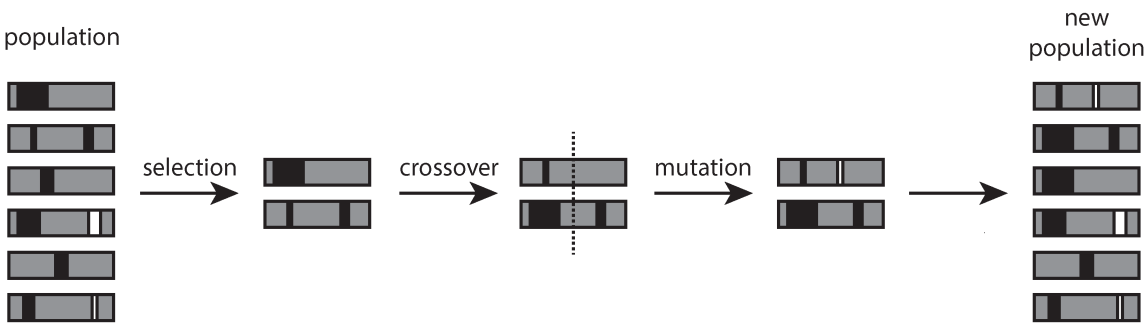


Figure 1: Schematic representation of a discrete genetic algorithm. Every iteration, the algorithm performs parent selection, crossover, and mutation to create new individuals which replace bad solutions in the population if their fitness is good enough. This schematic represents discrete GAs in general. For a more complete description of the continuous GAs analysed in this thesis, we refer to Section 3.2. Figure adapted from [8].

Algorithm 1 General genetic algorithm to minimise an objective function f.

```
1: function GENETIC_ALGORITHM(f)
2:     population ← initialise()
3:     while number_evaluations < max_evaluations do
4:         p1 ← select_parent(population)
5:         p2 ← select_parent(population)
6:         child ← crossover(p1, p2)
7:         child ← mutate(child)
8:         if is_invalid(child) then
9:             child ← correct(child)
10:        end if
11:        fitness ← f.evaluate(child)
12:        number_evaluations ← number_evaluations + 1
13:        if fitness < lowest_fitness then
14:            population.find_worst() ← child
15:        end if
16:    end while
17:    return population.find_best()
18: end function
```

solution using crossover (Line 6) and mutation (Line 7) operators. If an invalid solution is created, it is corrected (Line 9). Then, the fitness of the solution is evaluated (Line 11). In the final step, fitness selection is performed (Line 14). If the new individual is better than the worst fitness in the current population, the worst individual is replaced by the new individual, however other survivor selections schemes are also possible. The algorithm terminates after a maximum number of fitness evaluations (Line 3).

GA parameters have different functions. For example, crossover mixes solutions and enables the algorithm to converge in a subspace, mutation increases the diversity and facilitates escaping from local optima, and a large population size prevents one good solution from dominating reproduction and causing premature convergence [9, 10]. However, in actual applications, changes in operators and parameters can have a more complex impact on GA behaviour. To select appropriate settings, it is important to understand how each choice impacts the GA's behaviour.

Genetic algorithms have been successfully applied in many complicated real-world applications, such as scheduling problems, forecasting, and image processing [7]. Although they are more specific, real-valued genetic algorithms can also be applied in a wide range of problem-solving tasks. For example, real-valued GAs have been successfully applied to optimise sensor placement in structural health monitoring [11], predicting higher heating values in several materials [12], and in designing electromagnetic devices [13].

GAs can be very efficient in finding a sufficiently good solution to a complicated problem [7]. They can explore non-linear domains without requiring information about the objective function's characteristics like the frequency of discontinuities and local optima, or its gradient. However, the performance of a genetic algorithm on a specific problem is highly dependent on the formulation of the fitness function, nature of the problem, and choice of parameters like population size, evaluation budget, rate of mutation, rate of crossover or selection criteria [9, 14]. Appropriate parameter selection ensures that the algorithm converges efficiently to a good solution while thoroughly exploring the search domain.

2.2 Tuning and Configuring Evolutionary Algorithms

There have been many studies focused on improving the performance of evolutionary algorithms. The approaches studied can generally be classified into parameter tuning and parameter controlling approaches [15]. In parameter tuning approaches, the best parameters for a given application are determined in advance and kept constant during the run of the algorithms. Examples of tuning approaches are applying a different model or a second EA to optimise parameters, or sampling different sets of parameters. In parameter controlling approaches, parameters are allowed to change

during a run. A popular example is allowing the mutation step size to change during a run [16]. In evolution strategies with the 1/5 success rule, the mutation step size can be increased if the number of successful iterations is too large or vice versa. This prevents an algorithm from searching too locally or too globally. Even though many of these methods are effective in improving algorithms' performance, they can make the algorithms much more complicated and often lack an insight in how algorithm dynamics are impacted by these changes. In this thesis, we suggest additional instruments based on methods from network science to help visualise and understand how changes in parameters affect the dynamic behaviour of GAs.

2.3 Network Science

In many domains we can find large systems of interacting agents that exhibit complex behaviour. The word complex refers to non-trivial behaviour that is neither completely random nor completely regular [17]. By modelling these systems as graphs, we can quantify this non-trivial structure by analysing the graph's topological features. A graph is a mathematical representation of a network, made of nodes or vertices that are connected by links or edges. In this thesis, we will use 'network' and 'graph' interchangeably.

Complex network theory can be found in and applied to many different fields. For example, it can be used to detect vulnerabilities in an electricity grid, to study the spread of diseases or information in a social network, and to develop drugs with minimal side effects by studying molecular networks [18]. Interestingly, such networks tend to be governed by similar organising principles, despite the differences in their origin. In particular, many real-world networks are scale-free, which means that few highly connected hubs coexist with a large number of poorly connected nodes. Moreover, they are small-world, which means that even in very large networks, the shortest distance between any two nodes is unexpectedly small [18].

Of course there are also quantifiable differences between different classes of networks. Social networks, for example, are often highly assortative [18]. Assortativity is the extent to which a node tends to connect to similar nodes, e.g. similar in terms of degree, the number of connections. High-degree hubs, individuals with many connections, often tend to connect to other hubs in social networks. In contrast, protein-protein networks tend to be more disassortative, hub proteins avoid each other. A power grid network is often more neutral [18]. By quantifying topological features like a network's scale-free property or assortativity, we are able to quantify and reveal the complex underlying organising principles that shape these networks and interactions. A better understanding of the behaviour of complex systems has many practical applications, such as identifying potential drug target molecules in a metabolic pathway or identifying and fixing vulnerabilities in a power grid or a server network. As shown in this thesis, by visualising and modelling genetic algorithms as networks, we can apply similar analyses to those network representations to better understand GA dynamics and interactions.

2.4 Random Networks as Null Models

Complex networks are partly governed by stochastic processes. As a consequence, most network topology features are not particularly informative on their own [19]. Therefore, it is important to determine whether trends in network features are due to chance or due to some underlying organising principle. We can contextualise network features using random null models. Comparing to null models allows us to draw conclusions on whether the values we observe are actually of interest, caused by the algorithms we study, or whether these values are random deviations.

There are two common ways to create a random network model: generative methods that build a random graph from scratch and randomisation-based methods that randomise an existing graph. The choice of null model has important consequences for our analysis. We can design random networks using known mechanisms to express specific topological properties [20]. When we compare these to real-world networks, we can deduce the relevance of these hard-coded mechanisms for real-world networks and to what extent topological features are correlated. Many network features depend on other, simpler features, such as the number of nodes, the proportion of possible edges that exist, and the number of edges incident on each node [21]. If we pick the right restrictions for our random

networks, we can keep these basic properties fixed and determine if a certain feature is due to the specific topology of our network rather than one of these basic properties.

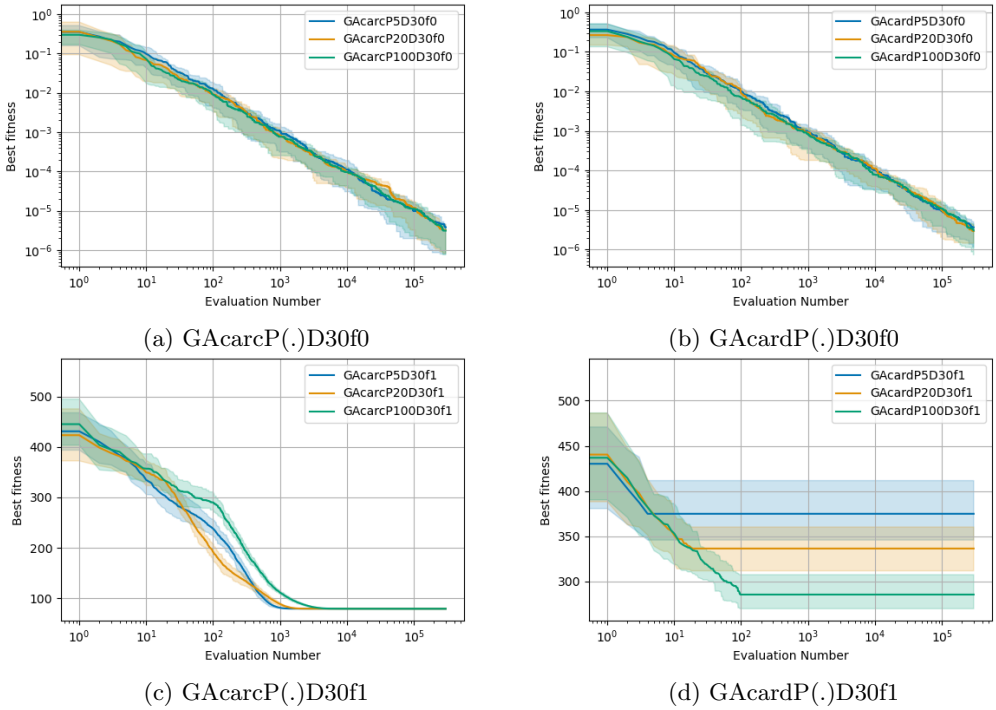
For each topological feature we study, we can normalise its value with respect to the average value in an ensemble of comparable random networks. This allows us to quantify the extent to which each feature deviates from what would be expected from chance alone [19]. Moreover, normalising different networks that have similar topological features with their respective random ensembles can sometimes reveal differences that would not be apparent without normalisation. Section 4.3 explains how we construct and apply random null networks in this thesis.

2.5 Applying Networks Analysis to Population-based Algorithms

Recently, researchers have turned to network analysis to deepen the understanding of the underlying interactions and dynamics in population-based algorithms. For example, Zelinka et al. showed that different population-based algorithms can be represented as complex networks [22]. In subsequent work, it has been shown that complex network analysis can excellently represent interconnections and dynamics within an EA population for differential evolution algorithms (DE) [5], self-organising migrating algorithms (SOMA) [4], and genetic algorithms [6]. Such network analysis can contribute to better control of these populations. More specifically, complex networks can be visualised as coupled map lattice (CML) systems. Algorithm dynamics can be controlled based on the state of the CML, which improves algorithm performance [23, 24]. For example, CMLs can be used to identify individuals that are contributing the least to improvements, which can be regenerated into more useful individuals.

A study by Metlicka et al. shows another example of how complex network analysis can improve the performance of population-based algorithms, specifically the artificial bee colony (ABC) algorithm [3]. In this study, the node centrality in the network is used as a measure of the transfer of information between solutions. Solutions with a low centrality are considered unimportant to population development and are replaced. As a result, the new, adaptive ABC variants outperformed the ABC algorithm, especially on higher-dimensional problems. Similar enhancements have been applied to differential evolution algorithms using closeness centrality or node strength [25, 26, 27].

An important characteristic of the studies mentioned above is their philosophy of dynamics in population-based algorithms. These studies do not focus on the philosophy that a bad parent is replaced by a better offspring, but accept the philosophical interpretation that an individual is moving to the better position [6]. This interpretation seems preferable in algorithms like SOMA, ABC, or DE, where each individual exchanges information with its peers to update into a better solution. In these cases, the resulting network representation reflects the exchange of information during the algorithm. However, this interpretation may not accurately reflect the exchange of information in genetic algorithms. Such a representation would collapse the interactions of several individuals over multiple generations into a single node in the network. As a result, this representation would fail to present how genetic information can be inherited and spread throughout the network over multiple generations of the algorithm. Instead of drawing links for each information exchange, we propose adding a new node for each new solution and drawing a link for each parent-child pair. Moreover, in their representation, Zelinka et al. seem to require a complex network structure for the visualisation with a CML. Although it is not entirely clear by what criteria this is determined, they remark that algorithm settings and objective function play an important role in whether or not a complex network structure emerges [6]. In Section 4, we describe our method of representation that conserves the hereditary patterns of GAs and can be used to analyse GAs with a large variety of settings and objective functions.



(a) GAcarcP(.)D30f0

(b) GAcardP(.)D30f0

(c) GAcarcP(.)D30f1

(d) GAcardP(.)D30f1

Figure 2: GA performance on a random function (f_0) and a non-random function (f_1). The figures for all GA variants can be found in Appendix A. The legend labels containing regular expressions that represent groups of GA variants are explained in Section 3.3. Every line is an average of 50 different runs. Drop shadows indicate the interquartile range of the different runs.

3 GA Data

In this thesis, we construct network representations from data that were gathered from several real-valued GA variants run on two different test objective functions. In this section, we define the objective functions, explain the differences between GA variants, and describe a variant-naming system, used to refer to groups of variants in the rest of this thesis. The data were generated by Dr. Fabio Caraffini at Swansea University.

3.1 Objective Functions

Multiple GA variants were used to minimise several objective functions. In this thesis, we create and discuss the network representation for one random function (f_0) and one non-random function (f_1). Figure 2 shows how a sample of GA variants progresses on both objective functions. Figures for all GA variants can be found in Appendix A. As both functions are minimised, more fit solutions will always have a lower fitness than less fit solutions. The number of dimensions (D) was 30 for both functions.

Function f_0 returns a uniformly random fitness from the interval $[0, 1]$ for any input.

$$f_0 : [0, 1]^D \to [0, 1] \tag{1}$$

Function f_1 is the sphere function from the BBOB suite [28]. The BBOB suite comprises a number of benchmark functions each with different properties, designed to asses different algorithm performance metrics and allows to compare the performance of many different algorithms. The sphere function is useful to determine the optimal convergence rate of the optimising algorithm.

$$f_1(x) = ||z||^2 + f_{opt} \tag{2}$$

Here, $||z||$ is the Euclidean norm of vector $z = x - x^{opt}$ with optimal solution x^{opt}, which is drawn uniformly from $[-4, 4]^D$ and input $x \in [-5, 5]^D$. f_{opt} is drawn from a Cauchy distribution with zero

median and roughly 50% of the values between -100 and 100 [28]. The same random optimum is used for all GA variants. The coordinates of the optimum can be found in Appendix F.

3.2 GA parameters

In this section, we explain how GA variants differ in their mutation operator, crossover operator, method of parent selection, correction method, and population size. We introduced the general framework of GAs in Algorithm 1. This section describes its detailed implementation.

3.2.1 Mutation

Two types of mutation operators are studied: Gaussian mutation and Cauchy mutation [29]. In real-valued stochastic algorithms, a normal Gaussian distribution is typically used to mutate solutions. The Gaussian probability distribution is given by

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(z-\mu)^2}{(2\sigma)^2}\right)$$

where μ and σ^2 are the mean and variance, respectively. The Cauchy distribution is similar to the Gaussian distribution, however, it is a bit wider. This means that the probability of generating larger values is slightly higher [29]. As a result, algorithms that apply Cauchy mutation are able to shed local optima more easily and improve over a larger scope [30]. The Cauchy probability distribution is given by

$$p(z) = \frac{1}{\pi} \frac{t}{t^2 + z^2}$$

where $t > 0$ is the proportion parameter.

In our setup, each coordinate is mutated individually with $\mu = 0$ and $\sigma = t = 0.01 \times r$, where r is the width of the search domain of the coordinate.

3.2.2 Crossover

We compare discrete and arithmetic crossover. In discrete crossover, every child coordinate has a 50% chance to be from either parent. In arithmetic crossover, children are generated with the following formula:

$$child = p_1 + \alpha \times (p_2 - p_1)$$

Here, p_1 and p_2 are parent the two selected parents and α is a random number from the uniform distribution $\mathcal{U}[-d, 1+d]$ with $d = 0.25$.

3.2.3 Parent Selection

Two types of parent selection are compared in this thesis: random selection and tournament selection. In random selection, each individual has the same probability of being selected as parent. In tournament selection, two individuals are randomly selected as candidates. The individual with the best fitness is selected as parent.

3.2.4 Correction

Objective functions are typically constrained to a finite domain in optimisation problems. Previous studies have shown that the strategy of dealing with solutions outside of the domain can introduce structural bias and greatly impact algorithm performance [1, 2, 31]. In this thesis, we compare four correction strategies for out-of-domain solutions [32].

Firstly, we consider the dismiss strategy, where an infeasible solution is simply discarded and a new one is regenerated.

Secondly, any coordinate outside of the domain is reflected off the opposite domain boundary in the toroidal correction strategy. This would be equivalent to periodic boundaries.

Thirdly, in the saturation strategy, coordinates are moved to the closest amongst their upper and lower boundary.

Finally, we consider the complete one-tailed normal correction strategy (COTN) [32]. Here, an infeasible coordinate is moved towards the domain by a random number drawn from a one-tailed normal distribution $|N(0, \frac{1}{3})|$. This is done recursively until the coordinate falls within the domain. The goal of this method is to map out-of-domain solutions back into the domain non-deterministically, close to the domain boundaries.

3.2.5 Population Size

We compare GAs with three different population sizes: 5, 20, and 100.

3.2.6 Additional GA settings

For all GA variants, the maximum evaluation budget was $10000 \times D$, where $D = 30$ is the function dimensionality. Each GA was run 50 times on the same function to capture the stochastic variability of the GA dynamics. Thus, for each variant, we create 50 different network representations, one for each run.

Conventionally, duplicate solutions should be avoided. However, in this thesis, we only consider real-valued GAs. As a result, we do not have to check if the newly generated solution is identical to a pre-existing solution, as this is extremely unlikely.

3.3 GA Variant Names

The data comprise GA variants with two types of mutation operators, two types of crossover operators, two methods of parent selection, four methods of out-of-bound correction, and three different population sizes. This results in 96 possible GA variants for each objective function we study. We will often refer to groups of variants. For this, we use a notation based on regular expressions. Each GA parameter is abbreviated as listed below:

- Mutation:
  - c: Cauchy mutation [30]
  - g: Gaussian mutation [29]
- Crossover:
  - a: arithmetic crossover
  - d: discrete crossover
- Parent selection:
  - r: random selection
  - t: tournament selection
- Correction [32]:
  - c: COTN strategy
  - d: dismiss strategy
  - t: toroidal strategy
  - s: saturation strategy
- Population size (popsize): $P5$, $P20$ or $P100$
- Objective function:
  - $f0$: random objective function (Equation 1)
  - $f1$: sphere function (Equation 2) [28]

The name of each variant group will have the following format:

$$GA(mutation)(crossover)(selection)(correction)(popsize)(function)$$

If two or more variants are combined in the same group, their parameters will be separated by a vertical line (|). A wildcard (.) means that all possible variants are included in the group. For example the following GA group name:

$$GAg.r(t|s)P5f0$$

The variants in this group have the following properties:

- g: They all use Gaussian mutation.
- .: They either use discrete or arithmetic crossover.
- r: They all use random selection.
- $(t|s)$: They either use toroidal or saturation correction.
- $P5$: They all have a population size of 5.
- $f0$: They all optimise the random objective function f_0.

4 Methods

In this thesis, we analyse a dataset that comprises information from a variety of GAs used to optimise different objective functions. We transform these data into network representations and analyse their topological features. In this section, all procedures used to create and analyse GA networks are explained. Firstly, in Section 4.1, we design a network representation that expresses information on the dynamics and performance of the corresponding GA. Secondly, in Section 4.2, we describe several network topological features to help us gain insight into the underlying algorithm dynamics. Thirdly, in Section 4.3, we describe a rewiring process that we can use to create null models to assess the significance of the patterns that emerge in our representations. Finally, in Section 4.4, we construct a visualisation method to intuitively generalise and compare network features and dynamics. All algorithms and calculations in this section were implemented using the NetworkX Python package version 2.5 [33].

4.1 Constructing Networks from Genetic Algorithms

Table 1 shows an example of two consecutive rows of a GA data file. A single file corresponds to a single run of a single GA variant and every row corresponds to a successful iteration of the algorithm in which a new solution replaces an old solution in the population. For every row in the data file, we create a new node (v_i) in the network representation and store its fitness as a node attribute. If the new node is not part of the initial population, we draw an edge from each parent to the new node, or only one edge if the same parent was selected twice. When constructing the network, it is important to assign each node a unique ID. This is needed for the chosen implementation of random rewiring as explained in Section 4.3. Each network $\mathcal{G}$ has a set of vertices V and a set of edges $E \subseteq \{(v_i, v_j) | (v_i, v_j) \in V^2 \wedge v_i \neq v_j\}$. The number of vertices and edges are $|V|$ and $|E|$, respectively. The set of nodes without incoming edges (M) are called maximal and the set of nodes without outgoing edges (μ) are called minimal. Figure 3 shows an example of what such a network representation looks like.

Many initial solutions are quickly replaced and are unable to create any edges. These solutions do not contribute to topological features due to their lack of edges. Therefore, we use the nodes in the largest component to calculate topological features. The largest component is the largest set of vertices that is connected to each other by an undirected path.

We create a new network representation for each file. This means that we create 50 network representations for each variant. Each row only corresponds to successful iterations; we ignore iterations that did not generate a solution that was added to the GA's population. As a result, each data file can have a different number of rows. Consequently, the network representations we create have a different number of nodes and edges for each run. In the rest of this thesis, when we use the term 'iteration', we refer only to successful iterations that add a node to the GA network. By analysing each representation and updating the network representation each iteration, we obtain a time series for each run of each variant that reveals how its topological features evolve while the algorithm is running.

Previous studies that employed dynamic network representations of population-based algorithms used a fixed number of nodes and add links over time [6]. With this interpretation, new and good solutions do not replace old solutions. Instead, the old solution is considered to move to a better position. This approach allows for the comparison of a wide range of population-based algorithms. Nevertheless, this representation may not express GA specific dynamics accurately.

There are three important differences between our method of analysis and the methods used previous in previous studies. Firstly, instead of analysing the interactions and information exchange between individuals, our analysis is more focused on how genetic information is inherited and passed on between different generations. We expect that this approach is more suited to represent GA

Table 1: Example of two consecutive rows in a GA data file.

New ID	Parent 1 ID	Parent 2 ID	Fitness Value	Fitness Evaluation Number	Replaced ID	Coordinates
39	37	36	3.78574965e-05	24324	38	1.71738996e-01, ..., 2.66379118e-01
40	32	37	1.05745735e-04	28590	37	1.33588664e-01, ..., 2.46944369e-01

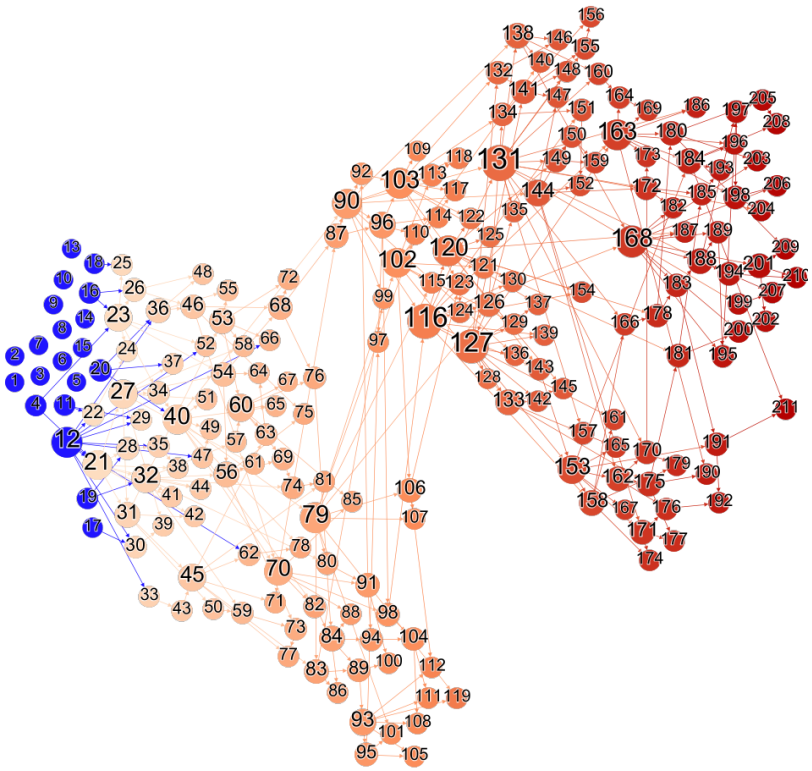


Figure 3: One run of a genetic algorithm with population size 20 (GAgdtcP20D30f0 as explained in Section 3.3) visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 211 nodes and 363 edges. Other visualisation examples can be found in Appendix B.

dynamics specifically. Secondly, we only consider successful iterations. In previous studies that focus on exchange of information, iterations of algorithms that produce bad solutions are used to decrease edge weights. Using edge weights in this way is beyond the scope of our analysis, however, may present an opportunity to produce a more comprehensive network representation. Thirdly, in this thesis, we focus on analysing and understanding trends in algorithm dynamics instead of controlling and improving the algorithm while it is running. Nevertheless, the methods presented in this thesis can be calculated in real-time and provide potential targets to control algorithm behaviour while it is running.

In summary, the chosen method of network construction has several important consequences for our analysis.

1. We only consider iterations where a good new solution was found. As a result, runs of the same algorithm with an equal evaluation budget may produce networks of different sizes.

2. Our networks grow in edges and nodes over time. Especially in early iterations, we are dealing with very small networks. The low number of nodes in early iterations can cause uninformative artefacts, as most network analysis measures are only well defined for large networks. Properties such as assortativity and centralisation (explained in Section 4.2) are not very informative when a network only has a handful of nodes and edges. We expect that most properties make large jumps in the early iterations but stabilise over time.

3. Only a fixed number of nodes in the network can generate new children and edges. Every iteration, a new solution replaces an old solution in the population and the old solution is marked 'dead'. Consequently, the dead node cannot create new children.

4. The networks produced with this method belong to the class of directed acyclic graphs (DAGs).

The unique properties of this class of graphs open up new possibilities of network analysis that were not considered previously. Specifically, leaf-leaf distance and treeness (explained in Section 4.2) are unique to DAGs. In some cases, like leaf-leaf distances, it is also possible to consider the networks as being undirected graphs.

4.2 Network Topological Features

Here, we define several topological features of our network representations. In our experiments, we observe these properties and their evolution over time to attain an understanding of how these properties relate to underlying algorithm dynamics.

4.2.1 Assortativity

The *assortativity coefficient* ($r \in \{\mathbb{R} \cap [-1, 1]\}$) is the correlation coefficient of a specific attribute between pairs of nodes [34]. It measures the extent to which nodes tend to connect preferentially to other nodes that are like them in some way. If the assortativity of a network is 1, there is perfect assortative mixing. An assortativity of 0 indicates that there is no specific form of mixing and an assortativity of -1 means that there is a perfect negative correlation between connected nodes. Commonly, the Pearson correlation coefficient is used to calculate assortativity. However, the Pearson correlation has two big disadvantages. Namely, it does not compare well between networks of varying sizes and it does not quantify non-linear correlations accurately. In this thesis, we use Spearman's correlation coefficient instead. Spearman correlation is able to quantify non-linear correlations in networks of different sizes more accurately [35]. To calculate Spearman's correlation coefficient, values for source nodes and values for target nodes are ranked individually. The Spearman correlation is calculated by calculating the correlation between the rankings. For a formal definition of Spearman correlation, we refer to Kokoska et al. [36]. In this thesis, we consider both degree and fitness assortativity.

Degree assortativity is a common tool in network analysis. It is informative of the type of graph and its underlying forming mechanisms. For example, many social networks are found to have a positive degree assortativity, while biological and technological networks tend to be slightly disassortative [34]. A notable property of networks with a high degree assortativity is their resilience to the removal of nodes. A high degree assortativity in a GA network could indicate that the algorithm is able to progress through many different paths and solutions. On the other hand, a low degree assortativity could indicate that the algorithm heavily relies on one or a few nodes with a high degree that act as a bottleneck in the algorithm's progression. We expect that GAs express a higher degree assortativity on objective functions with a smooth fitness landscape, compared to objective functions with a rugged fitness landscape. On a fitness landscape with many discontinuities and local minima, GAs can make sudden big jumps in fitness. The new and good solution provides access to a new part of the search space. If the jump in fitness is big enough, this may result in a substantially high selection frequency of this individual and a low degree assortativity.

We also calculate *fitness assortativity*. Fitness assortativity can be an indicator of the average rate of improvement and convergence of the algorithm's solutions. A low assortativity means that low fitness nodes are connected to high fitness nodes. Thus, the algorithm is progressing at a high rate. In general, we expect that fitness assortativity increases over time, as new solutions are closer together in fitness as the algorithms converge.

4.2.2 Centralisation

Network centrality measures determine the relative importance of each node. These measures can be used to identify nodes that are disproportionately important to the structure of the network. In order to generalise centrality for an entire network, we use the Freeman centralisation ($C \in \{\mathbb{R} \cap [0, 1]\}$) [37] given by

$$C = \frac{\sum_{i=1}^{|V|} |C(v^*) - C(v_i)|}{\max \sum_{i=1}^{|V|} |C(v^*) - C(v_i)|}$$

Here, $C(v_i)$ is one of the node centrality measures for node v_i and $C(v^*)$ is the largest value for any node in the network. $\max \sum_{i=1}^{|V|} |C(v^*) - C(v_i)|$ is the largest possible sum of differences in

node centrality for any network of N nodes. In other words, the Freeman centrality determines the dominance in centrality of the most central node with respect to all other nodes, relative to a network of the same size where this dominance has the largest possible value. In our calculations, this means that we compare our most central node to the dominance of the most central node in a star graph.

There are many different choices for centrality measures, each quantifying different structural network properties. We examine three centrality measures: degree centrality, out-degree centrality, and betweenness centrality.

Degree centrality uses the number of edges connected to a node as its centrality measure. The degree centralisation can be used as an indicator of convergence. High values show exploration of a single solution which can be a sign of premature convergence if this occurs in early iterations. We expect that GAs with high selection pressure will generally have higher degree centralisation.

Out-degree centrality is the fraction of all total nodes the outgoing edges of one node are connected to. Although this measure is similar to degree centrality, we expect out-degree centrality to have a slightly different relation to network size.

Betweenness centrality measures the relative number of shortest paths that pass through each node [38]. Betweenness centralisation can be used as an indicator of bottlenecks. In other words, high values suggest that many evolutionary pathways are directed through only one or a couple of nodes. We expect that GAs that are able to make large jumps in fitness will have a large increase in betweenness centralisation. Such a large jump could be caused, for example, by an objective function that contains discontinuities or local minima instead of a smooth fitness landscape.

Low values of both degree and betweenness centralisation mean that all nodes are equally centralised. This can be a sign of exploration because each node is contributing equally to newly added solutions.

4.2.3 Laplacian Eigenvalues

Eigenvalues of the Laplacian matrix of networks can contain a lot of information about the structure of the network. We use the adjacency matrix to calculate the Laplacian matrix (L). L is given by subtracting the adjacency matrix (A) from the diagonal matrix of node degrees (D):

$$L = D - A$$

The normalised Laplacian is then given by:

$$N = D^{-1/2} L D^{-1/2}$$

The eigenvalues of this matrix (λ_i) can be used for several topological metrics [39]. We consider two, namely, the second smallest non-zero eigenvalue, also called the *algebraic connectivity* ($\lambda_2 \in \mathbb{R}^+$) and the difference between the two largest eigenvalues, also called the *spectral gap* ($\lambda_{|V|} - \lambda_{|V|-1} \in \mathbb{R}^+$) [40]. Both of these are measures for the network's degree of connectivity. Connectivity indicates the robustness of a network. The higher the connectivity, the more nodes need to be removed to split a network into two isolated networks. Connectivity offers a more generalised measure of a network robustness compared to degree assortativity.

4.2.4 Leaf-leaf Distance

In order to gauge the exploitation of GAs, we measure the lengths of undirected paths between all leaf nodes. Leaf nodes are nodes with no outgoing edges. If exploitation is dominating the optimisation, we expect low leaf-leaf distances as we expect many common ancestors between newly added solutions. We calculate both the *maximum leaf-leaf distance* ($\max(d_{leaf}) \in \mathbb{Z}^+$) and the *mean leaf-leaf distance* ($\bar{d}_{leaf} \in \mathbb{R}^+$). The maximum leaf-leaf distance conveys to what extent the most distant new solutions are separated. The mean leaf-leaf distance will do the same for the average.

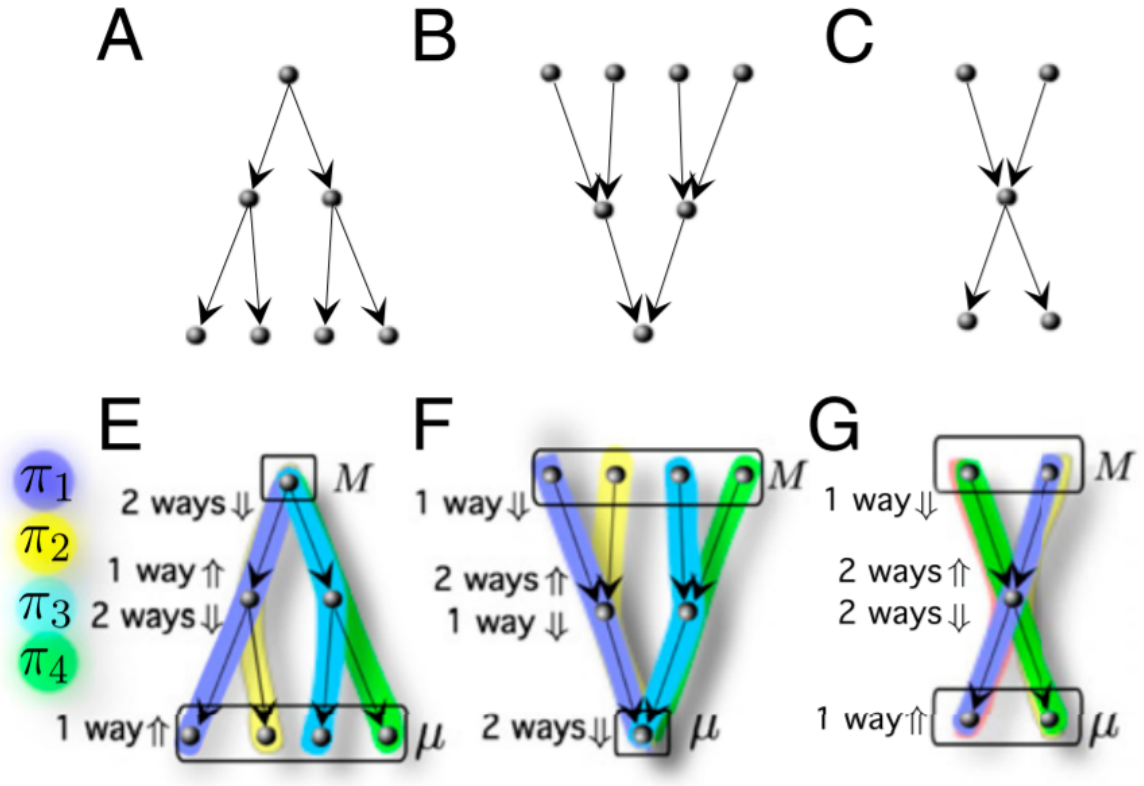


Figure 4: Example of how treeness is calculated in a network. A: A forward tree network with a treeness of 1. B: A backward tree network with a treeness of -1. C: A non-hierarchical network with a treeness of 0. E-G: Each network has four paths (π_1, π_2, π_3, π_4) from the top nodes (M) to the bottom nodes (μ). E: The forward uncertainty of paths is $H_f = \log 4$ while there is no uncertainty when the paths are reversed. F: The opposite of E: $H_f = 0$ and $H_b = \log 4$. G: Forward and backward uncertainties are equal: $H_f = H_b = \log 2$. Figure adapted from [41].

4.2.5 Treeness

Treeness ($T \in \{\mathbb{R} \cap [-1, 1]\}$) expresses how pyramidal the structure of a graph is or how reversible a graph is [41]. The reversibility is defined as the difference between the forward and the backward uncertainty of a graph. Treeness compares the creation of alternative paths in a forward exploration of a graph to the uncertainty in reversing those paths. In other words, if a random walker was to traverse the network, the uncertainty to follow a given path is compared to the uncertainty of following the reversed path. If the forward uncertainty is greater than the reversed or backward uncertainty, the graph shows a pyramidal structure. Figure 4 shows an example of three different networks and the calculation of their treeness.

For a directed acyclic graph ($\mathcal{G}$) with maximal nodes M and minimal nodes μ, the uncertainty ($h(v_i)$) to follow a given path (π_k) from $v_i \in M$ to a node in μ is given by [42]

$$h(v_i) = -\sum_{\pi_k \in \Pi_{M\mu}} \mathbb{P}(\pi_k|v_i) \log \mathbb{P}(\pi_k|v_i)$$

Here, $\mathbb{P}(\pi_k|v_i)$ is the probability that the path π_k is followed starting from node v_i. The average uncertainty to follow a path from some node v_i in the group of nodes M is

$$H_f(\mathcal{G}) = \frac{1}{|M|} \sum_{v_i \in M} h(v_i)$$

The treeness is then given by the normalised difference between the forward and backward entropies:

$$T = \frac{H_f - H_b}{\max(H_f, H_b)}$$

Due to our method of network construction, we know that each newly added node is the leaf node of a backward tree (Figure 4B). Because the entire network is a union of several backward trees, we can expect a negative treeness. Nodes that produce a lot of children increase the forward uncertainty in the network and, consequently, they also increase the treeness of the network. Therefore, we expect similar trends between the betweenness centralisation and treeness.

4.2.6 Shortest Path Length

Shortest path lengths can be used to identify small-world networks. The small-world property is ubiquitous in real-world networks [18]. In small-world networks, the *average shortest path length* ($\bar{d} \in \mathbb{R}^+$) scales logarithmically with the number of nodes [43]. As a result, even when the number of nodes becomes very large, the average distance between nodes can stay surprisingly small. An average shortest path length that scales less than linearly with the number of nodes and stays small, is an indication that shortcuts are created between distant nodes or distant neighbourhoods of nodes.

We also calculate the *longest shortest path length* ($\max(d) \in \mathbb{Z}^+$) or diameter. The diameter gives an indication of the network's scale. For our newtorks specifically, the diameter is the largest minimum number of generations involved in the creation of each solution.

4.2.7 Subtree Size

At each iteration, we create a subtree with the best node (v_{opt}) and all its predecessors. The *best subtree size* ($|V_{v_{opt}}| \in \mathbb{Z}^+$) is the number of different solutions used to generate the current best solution.

4.3 Random Null Models

When creating a network representation of a genetic algorithm, the objective function and choice of algorithm settings can have a large effect on the structure of the network. When analysing the network structure, it is important to determine whether topological features are caused by a specific algorithm setting, a property of the objective function, a combination of the two, or can simply be explained by chance. We use a generative model to assess the effects of an objective function on network structure and a randomisation-based model to assess the effects of algorithm settings.

We create a generative null model by running the GA variants on a random objective function. A random objective function randomises fitness selection and allows us to capture the behaviour of our GA variants without the influence of the fitness landscape of a non-random function.

We also create a randomisation-based model by randomly rewiring our network representations. This disrupts the structure caused by the parent selection of the genetic algorithm and allows us to determine a base line that is inherent to our method of representation.

A common way to compare null models and actual networks is normalisation [19]. Here, every data point is divided by the average value of a corresponding null model that has been matched for the relevant properties. To prevent dividing by 0, we normalise treeness and assortativity features using $\frac{value_o+1}{value_s+1}$, where $value_0$ and $value_s$ are the original value and the shuffled value, respectively.

We propose a rewiring algorithm that conserves edge degrees and the typical directed acyclic graph (DAG) structure of our GA networks (Algorithm 2). The rewiring process is based on Maslov-Sneppen rewiring [44]. Maslov-Sneppen rewiring preserves size, connection density, and degree distribution. With such a rewiring process, the shuffling can be constrained to achieve a better match between the properties of the empirical and null networks. This algorithm is a directed adaptation of an undirected edge shuffling function found in the NetworkX package [33]. The original version was modified to create directed edges and conserve a DAG structure.

Every iteration, two random edges ($u \to v$ and $x \to y$) are selected, weighted by the degree of their respective source nodes (Line 8-11). The endpoints of edges are swapped in a way that conserves the network structure (Line 15-18). As shown in Figure 3, nodes are ordered by the iteration at which they are added to the network. This makes it trivial to check whether the DAG structure is conserved during rewiring. As long as edges are connected from lower-order to higher-order nodes, the network conserves its DAG structure.

During every iteration, we calculate network properties for every network and a shuffled instance of the same network where edges have been swapped $100 \times |E|$ where $|E|$ is the number of edges

Algorithm 2 Algorithm to randomly rewire directed network edges, adapted from the undirected version implemented in the NetworkX package [33].

function DIRECTEDDOUBLEEDGESWAP(graph G, n_swaps, max_tries)
  $pdf \leftarrow normalize(G.out\_degrees())$
  while $swap\_count < n\_swaps$ **do**
    if $n\_tries \geq max\_tries$ **then**
      return G
    end if
    $n\_tries \leftarrow n\_tries + 1$
    $u \leftarrow select\_random(pdf)$
    $x \leftarrow select\_random(pdf)$
    $v \leftarrow select\_random(G.neighbors(u))$
    $y \leftarrow select\_random(G.neighbors(x))$
    if $u == x \vee v == y$ **then**
      continue
    end if
    $s_{ux} = \min\{u, x\}$
    $t_{ux} = \max\{u, x\}$
    $s_{vy} = \min\{v, y\}$
    $t_{vy} = \max\{v, y\}$
    if $(s_{ux}, t_{ux}) \notin E \wedge (s_{vy}, t_{vy}) \notin E$ **then**
      $E \leftarrow E \setminus \{(u, v), (x, y)\}$
      $E \leftarrow E \cup \{(s_{ux}, t_{ux}), (s_{vy}, t_{vy})\}$
      $swap\_count \leftarrow swap\_count + 1$
    end if
  end while
  return G
end function

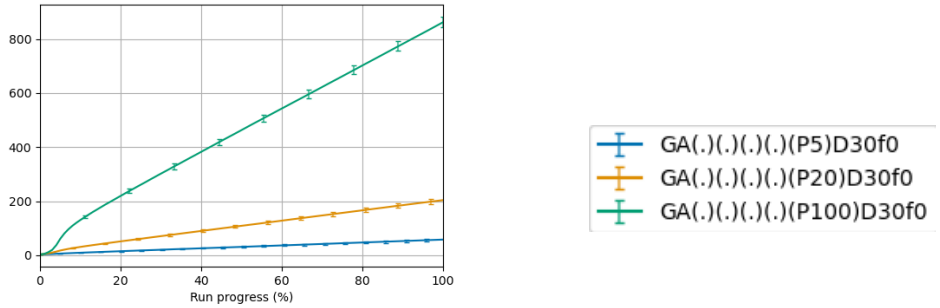
in the original network. According to previous studies, this should be enough iterations to create sufficiently random versions of most graphs [19, 45].

4.4 Visualisation of Topological Time Series

We measure network properties for each iteration of the growing networks, resulting in 50 time series for each run of each genetic algorithm. The stochastic nature of these data leaves us with some important choices to make in order to compare algorithm variants directly, as we have to consider extreme outliers and variations in algorithm progression.

Firstly, due to the stochastic nature of the algorithms, not all networks have an equal number of iterations which result in producing a solution fitter than a solution already present in the population. To align different time series, we resample the time series using linear interpolation, implemented with the `interp` function of Numpy version 1.20.3 [46]. This allows us to compare GAs directly, with respect to the relative run progression, despite the possible introduction of artefacts due to resampling. All trajectories start on the iteration where the first edges are added to the network because many topological features are not well-defined for a network without edges. This effectively reduces the number of function evaluations by the algorithm's population size.

Secondly, we generalise the results of 50 runs by visualising median values and error bars indicating the interquartile range, the range between the median of the lower half of the values and the median of the higher half of the values. This allows us to compare general GA behaviour while simultaneously taking stochastic variation into account without excessive impact of extreme outliers.



(a) Largest component size.

Figure 5: Median largest component size time trajectories for different population sizes on random objective function f_0 (Equation 1). The legend labels containing regular expressions that represent groups of GA variants are explained in Section 3.3. The error bars show the interquartile range of each group of variants included in the plot and the trajectories are rescaled to reflect run progress, as explained in Section 4.4. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.

5 Results

Here, we analyse times series of the topological features of our network representations of different genetic algorithm variants that are created using the procedures described in Section 4. In Section 5.1, we analyse the random objective function f_0 (Equation 1) without reference to a null model. In Section 5.2, we compare and normalise a limited sample of GA variants run on f_0 with respect to their shuffled counterparts. Finally, in Section 5.3, we analyse how topological features differ between random objective function f_0 and non-random objective function f_1 (Equation 2) for a limited sample of GA variants.

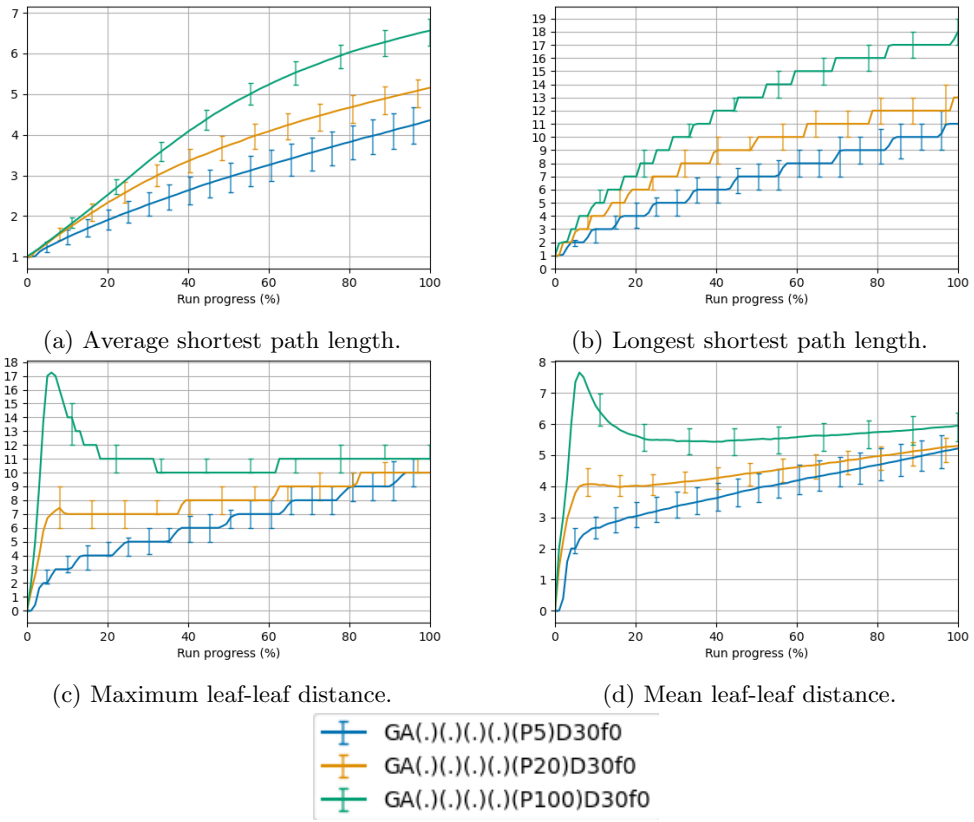
5.1 Random Function f_0

In this section, we analyse the time trajectories of features we calculate for network representations of GAs that optimise a random objective function f_0 (Equation 1). We compare different GA parameters and their impact on the network features described in Section 4.2. We discuss the figures that present discernible trends and interesting outliers in GA behaviour. The figures containing all other calculated time trajectories can be found in Appendix C. We discuss the impact on topological features of the following factors:

1. Network size
2. Population size
3. Parent selection strategy
4. Mutation operator
5. Crossover operator
6. Correction strategy

5.1.1 Effect of Network Size

Before we analyse network features with respect to different algorithm parameters, it is important to realise that our network representations grow in size over time (Figure 5). As a consequence, it is expected that measures that scale with network size, increase with population size. Moreover, with a larger network size, each added node has a smaller impact on averaged and network-wide measures. Unsurprisingly, our trajectories often show more deviations and larger jumps in early iterations than in later iterations. Variants with a smaller population size and thus a smaller number of nodes produce less smooth trajectories and more variance in between different runs.



(a) Average shortest path length.

(b) Longest shortest path length.

(c) Maximum leaf-leaf distance.

(d) Mean leaf-leaf distance.

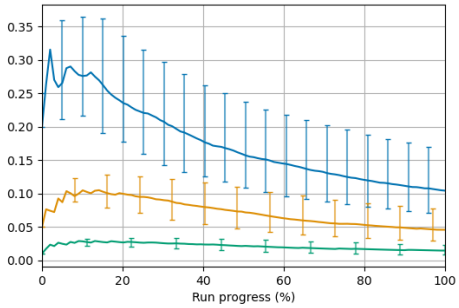
Figure 6: Median path length time trajectories for different population sizes on random objective function f_0. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.

5.1.2 Effect of Population Size

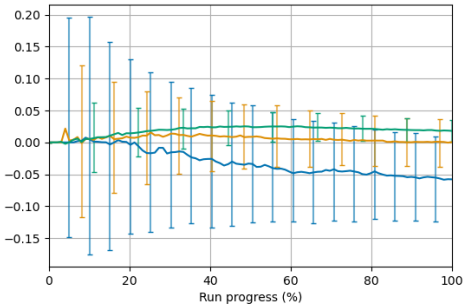
Figures 6a and 6b show how shortest path lengths are related to population size. As the networks increase in size, nodes are farther away from each other which results in higher shortest path lengths. Figures 6c and 6d show the maximum and mean distances between leaf nodes. Interestingly, these distances seem to converge to similar values for each population size. Despite of larger network sizes, the leaf nodes are still densely connected with larger population sizes. In contrast to shortest path lengths, leaf-leaf distances decrease after an initial spike for larger population sizes. This behaviour can be explained by exploration and exploitation competing to dominate the optimisation process. When leaf-leaf distances are increasing, different branches of the network are developed individually, increasing the distance between leaf nodes of each branch. This is a characteristic of exploration. When the leaf-leaf distances are decreasing, different branches are connected together to exploit their shared search space. This creates new, shorter paths between leaf nodes in different branches. The opposing forces of exploration and exploitation appear to reach a similar dynamic equilibrium between 9 and 12 for the maximum leaf-leaf distance and around 5.5 for the mean leaf-leaf distance for all population sizes.

Figure 7a shows that degree centralisation tends to be lower in larger population sizes compared to smaller population sizes. According to Figure 7b, there is hardly any assortative mixing with respect to node degree. The negative values indicate slight trend of disassortative mixing in population size 5, even though the spread in the time trajectories is very large. Figure 7c shows that all population sizes are highly assortative with respect to fitness. As expected, over time, as the algorithms progress, fitness assortativity converges to very high values.

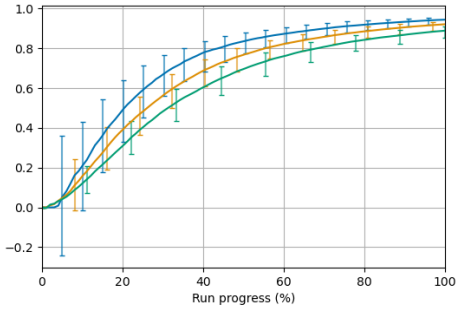
While the trajectories of GA variants with different population size behave differently in early iterations, the value of the algebraic connectivity converges to similar values for each population size (Figure 8). Together with Figure 6, this shows that the degree of connectivity is constant with



(a) Degree centralisation.



(b) Degree assortativity.



(c) Fitness assortativity.

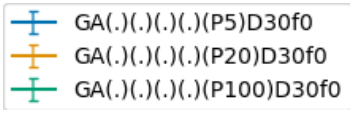
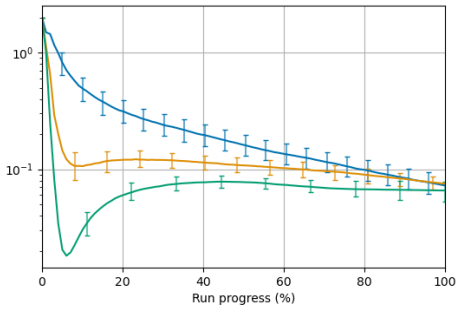


Figure 7: Median centralisation and assortativity time trajectories for different population sizes on random objective function f_0. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.

respect to population size and quite low, as shortest path lenghts do scale with respect to network size.

As expected, our network representations produce negative values for treeness (Figure 9a). Even so, some trajectories report positive values for a big part of the algorithms' lifetime before reaching negative values. This shows that in early iterations, the production of new branches is dominating the optimisation process, similar to the peak in leaf-leaf distances in Figures 6c and 6d. In later iterations, the consolidation of separate branches into common end points dominates the representation's extension, consequently lowering the treeness into negative values. Unsurprisingly, we can see that the best subtree sizes increase with population and network size (Figure 9b).



Algebraic connectivity.

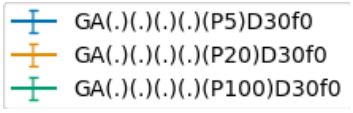
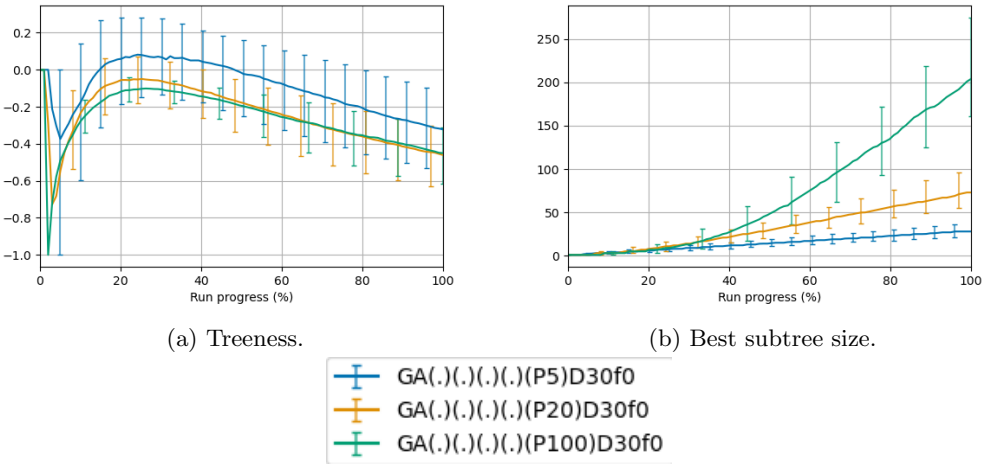
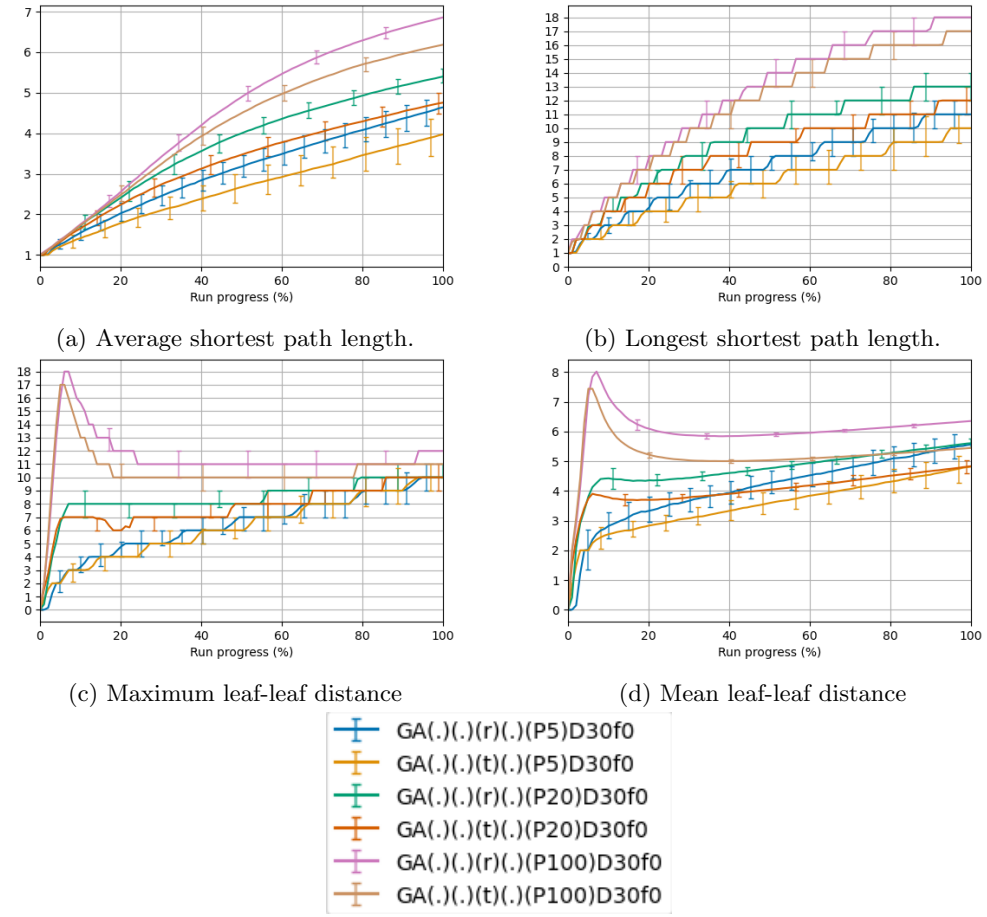


Figure 8: Median algebraic connectivity time trajectories for different population sizes on random objective function f_0. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.



(a) Treeness. (b) Best subtree size.

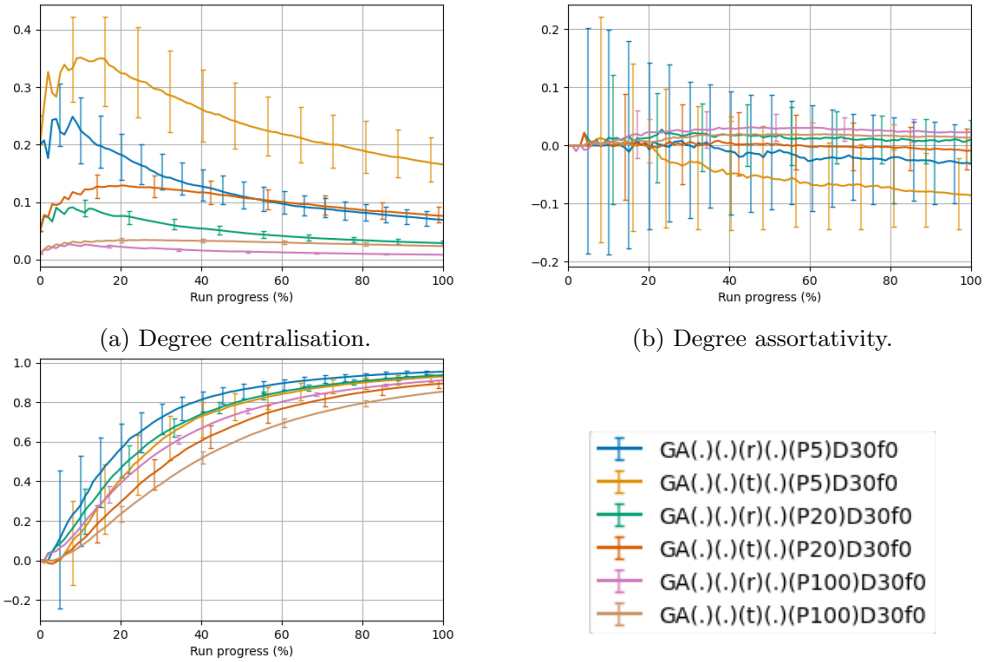
Figure 9: Median treeness and subtree size time trajectories for different population sizes on random objective function f_0. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.



(a) Average shortest path length. (b) Longest shortest path length.

(c) Maximum leaf-leaf distance (d) Mean leaf-leaf distance

Figure 10: Median path length time trajectories for different parent selection strategies on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.



(a) Degree centralisation.

(b) Degree assortativity.

(c) Fitness assortativity.

Figure 11: Median centralisation and assortativity time trajectories for different parent selection strategies on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

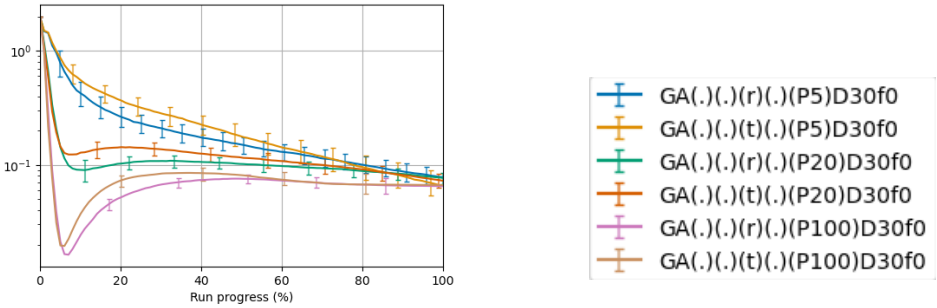
5.1.3 Effect of Parent Selection

The method of parent selection affects to which nodes new links are added. Figure 10 shows how the choice between random or tournament selection can change distances within our network representation, even if the network size hardly changes (Appendix Figure C.2i). Tournament selection correlates with smaller shortest path lengths (Figure 10a). A possible explanation is that, during tournament selection, highly fit nodes have a higher probability of being selected. As a result these nodes are able to accumulate more connections. Compared to random selection that would explore each branch of the network more equally, tournament selection reduces the diameter of the network (Figure 10b). A similar effect is observed in leaf-leaf distances. However, despite tournament selection causing lower maximum leaf-leaf distances compared to random selection in early iterations, both methods of selection still converge to similar values (Figure 10c). In contrast, mean leaf-leaf distances are lower with tournament selection throughout the entire duration of the algorithms' runs (Figure 10d).

The centralisation trajectories confirm how parent selection affects the distribution of edges in the network representations (Figures 11a). Random selection produces consistently lower centralisation scores, indicating a flatter centrality distribution for random selection. This confirms that higher parent selection pressure correlates with higher degree centralisation. Unintuitively, tournament selection also seems to lower fitness assortativity (Figure 11c). With tournament selection, more fit nodes are selected as parents than in random selection. As a result, it would be expected that the new, fitter node is connected to more fit parents, thus increasing the fitness assortativity compared to random selection. It is unclear what mechanisms cause the lower fitness assortativity in tournament selection.

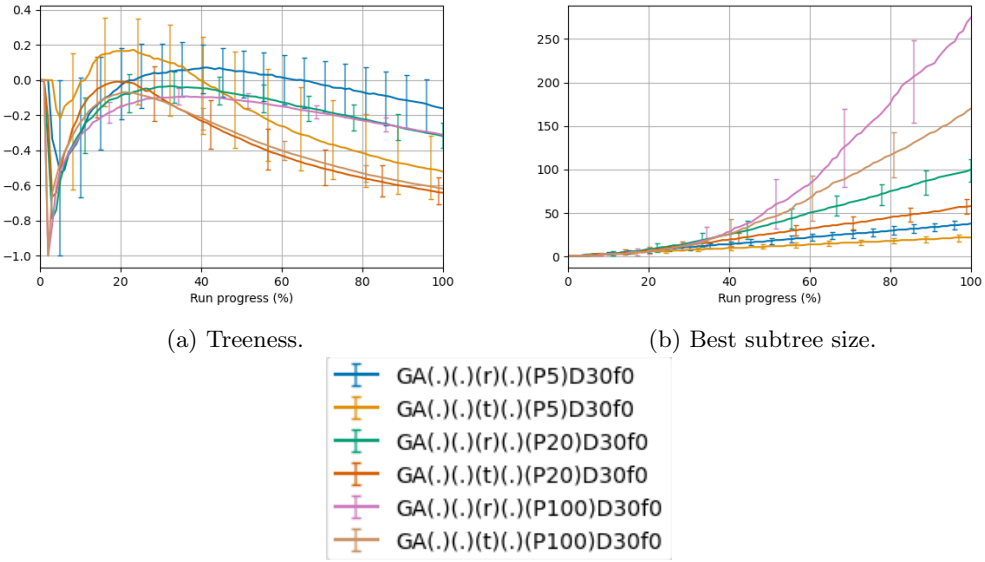
In correlation with the smaller distances and greater centralisation, tournament selection causes greater algebraic connectivity (Figure 12).

Parent selection strategy has a big impact on the treeness of the network representation (Figure 13a). The representations resulting from tournament selection resemble the structure of a backwards tree more than those resulting from random selection. This implies that random selection is continuously creating new branched paths, while tournament selection reduces forward uncertainty



Algebraic connectivity.

Figure 12: Median algebraic connectivity time trajectories for different parent selection strategies on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.



(a) Treeness.

(b) Best subtree size.

Figure 13: Median treeness and subtree size time trajectories for different parent selection strategies on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

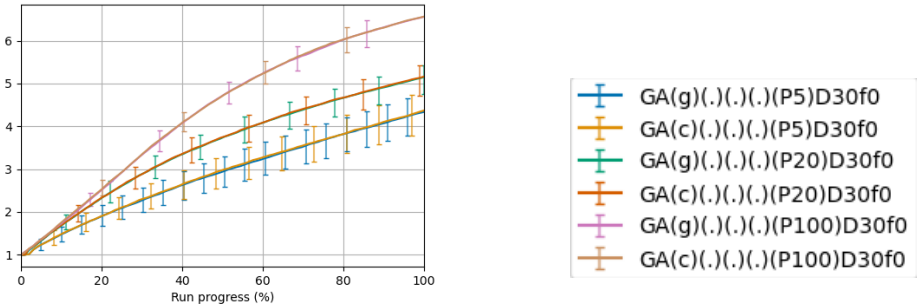
(as explained in Section 4.2.5) by combining nodes into a common child. Tournament selection also decreases the best subtree size (Figure 13b). When more fit nodes are selected more frequently with the increased selection pressure, only the subtrees containing those nodes are involved in creating new solutions. This may explain the smaller number of nodes in the best subtrees.

5.1.4 Effect of Mutation

As expected, we did not see a difference in trajectories for GAs using Cauchy or Gaussian mutation on a random objective function. When comparing the mutation operators, the trajectories seem almost identical (Figure 14, Appendix C.3).

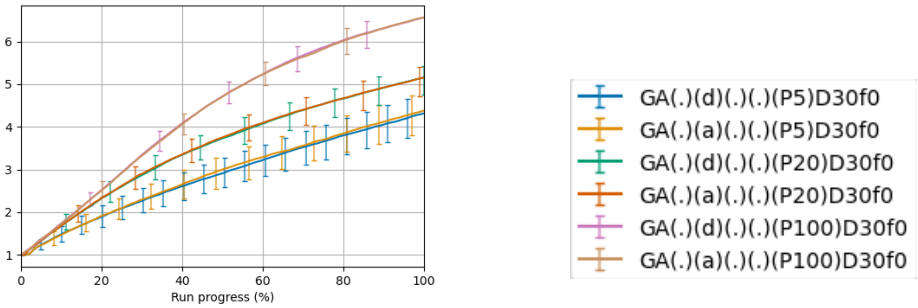
5.1.5 Effect of Crossover

Similar to the mutation operators, we did not observe the crossover method to have a noticeable impact on our time trajectories for a random objective functions (Figure 15, Appendix C.4).



(a) Average shortest path length.

Figure 14: Median average shortest path length time trajectories for different mutation operators and different population sizes on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs. There is no discernible difference caused by changing mutation operators.

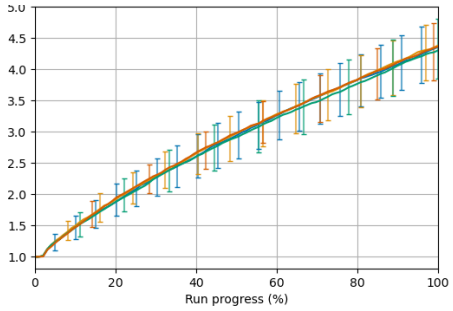


(a) Average shortest path length.

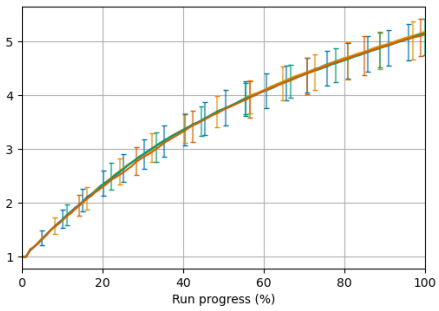
Figure 15: Median average shortest path length time trajectories for different crossover operators and different population sizes on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs. There is no discernible difference caused by changing crossover operators.

5.1.6 Effect of Correction

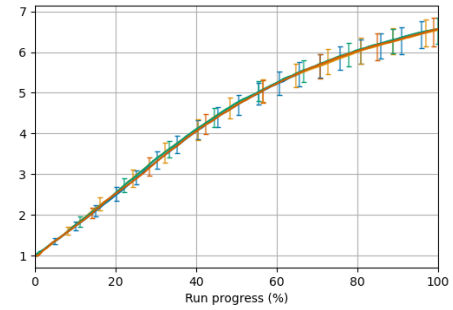
As expected, a change in correction strategy did not change the trajectories of topological features for GAs on a random objective function (Figure 16, Appendix C.5, C.6, C.7).



(a) Average shortest path length for population size 5.

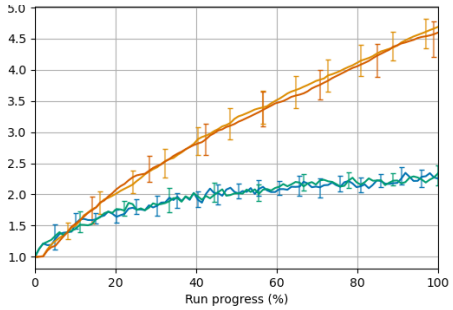


(b) Average shortest path length for population size 20.

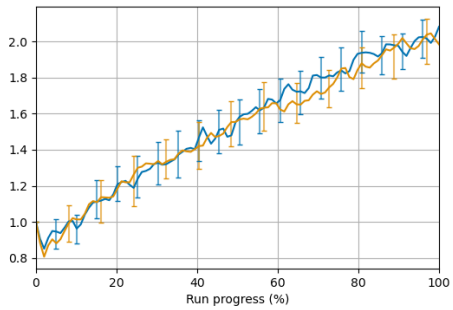


(c) Average shortest path length for population size 100.

Figure 16: Median average shortest path length time trajectories for different correction strategies and different population sizes on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs. There is no discernible difference caused by changing correction strategies.



(a) Average shortest path length.



(b) Normalised average shortest path length.

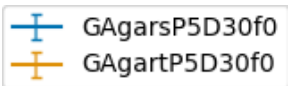


Figure 17: Median average shortest path length time trajectories on random objective function f_0 before and after normalisation. Every data point is divided by the mean of its respective shuffled counterpart. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.

5.2 Shuffled Networks on Function f_0

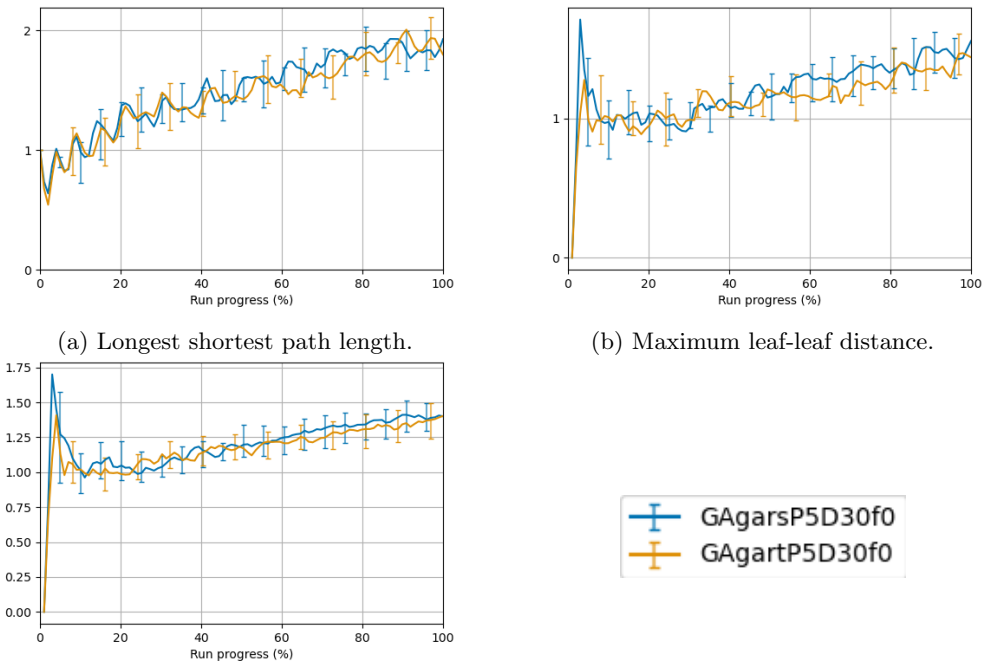
For each GA variant run, we shuffle the edges of its network representations $100 \times |E|$ times, where $|E|$ is the number of edges, to produce a randomised null model, as explained in Section 4.3. We compare the topological features of our networks to their respective null model to verify whether the features we observe are caused by algorithm dynamics or simply caused by our method of representation. Due to computational and time constraints, we were unable to shuffle and analyse all GA variants. In this section, we selected two GA variants for which 30 runs were shuffled. Because of this limited analysis, we cannot generalise our conclusions for general GA behaviour. However, this preliminary analysis offers some interesting insights into the degree to which our network representations resemble random behaviour. Figure 17 shows how we normalise each variant with respect to its shuffled version. All plots without normalisation can be found in Appendix D.

Figures 17 and 18 show normalised trajectories of path lengths. While path lengths keep increasing as the network representations grow for both the original and shuffled networks, path lengths seem to increase at a higher rate in the original representations than in the shuffled representations. This suggests that the original network representations are less small-world-like than would be expected from the random models.

Figure 19a shows that, even though there is a large spread, median centralisation scores are similar between shuffled and original networks. This is expected as the shuffling is designed to preserve degree distribution. Degree assortativity increases when the networks are shuffled (Figure 19b), implying that shuffling increases the robustness and connectivity of the representations. Fitness assortativity is higher in the original networks, compared to the shuffled networks (Figure 19c) as pairs of highly fit nodes are broken up and connected to less fit nodes by the shuffling process.

In congruence with the longer path lengths and degree assortativity, the algebraic connectivity is lower in the original representations than in their shuffled counterparts (Figure 20). Together with the increased best subtree size (Figure 21b), these topological features show that shuffling increases the connectedness and decreases the scale of the network representations.

Shuffling our network representations seems to remove its hierarchical structure as the treeness

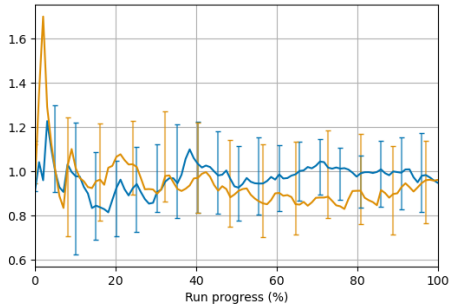


(a) Longest shortest path length.

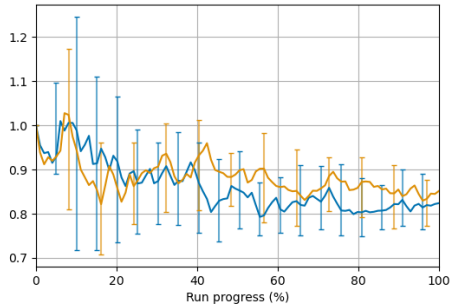
(b) Maximum leaf-leaf distance.

(c) Mean leaf-leaf distance.

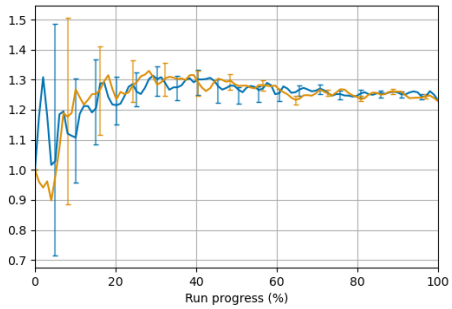
Figure 18: Normalised median path length time trajectories on random objective function f_0. Every data point is divided by the mean of its respective shuffled counterpart. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.



(a) Degree centralisation.



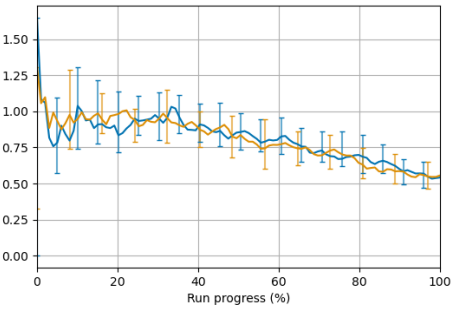
(b) Degree assortativity.



(c) Fitness assortativity.

Figure 19: Normalised median centralisation and assortativity time trajectories on random objective function f_0. Every data point is divided by the mean of its respective shuffled counterpart. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.

of shuffled representations stays close to 0 (Appendix Figure D.1o). Figure 21a shows that the GAs cause negative treeness in our representations.



(a) Algebraic connectivity.

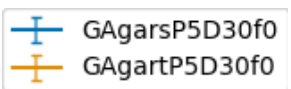
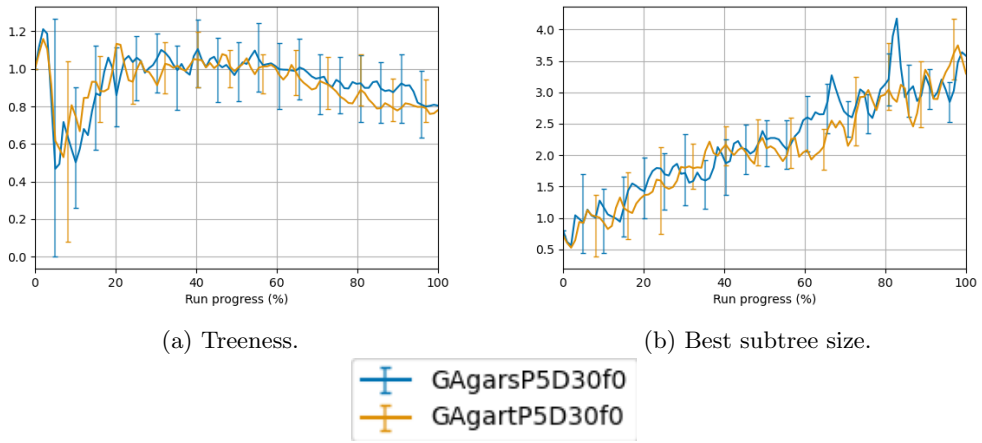
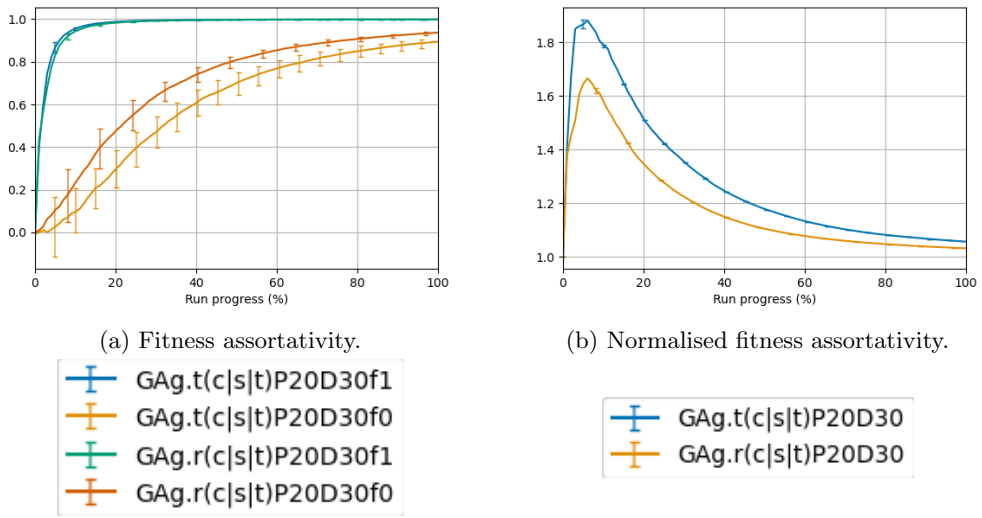


Figure 20: Normalised median algebraic connectivity time trajectories on random objective function f_0. Every data point is divided by the mean of its respective shuffled counterpart. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.



(a) Treeness. (b) Best subtree size.

Figure 21: Normalised median treeness and subtree size time trajectories on random objective function f_0. Every data point is divided by the mean of its respective shuffled counterpart. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.



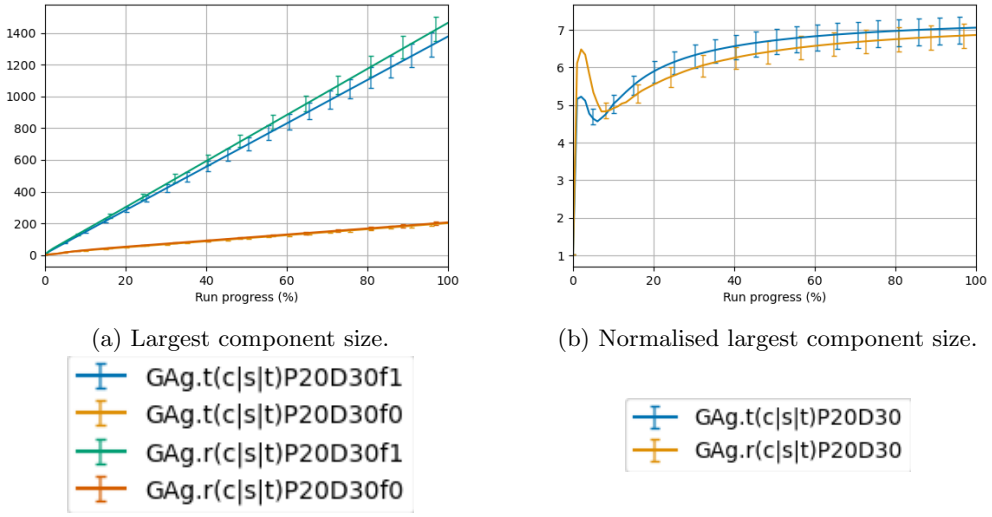
(a) Fitness assortativity. (b) Normalised fitness assortativity.

Figure 22: Median fitness assortativity time trajectories on objective function f_1 before and after normalisation. Every data point on f_1 is divided by the mean of its respective f_0 counterpart. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs. The normalisation reveals differences between random and tournament parent selection.

5.3 Sphere Function f_1

We use our GAs run on random objective function f_0 as a null model to compare to non-random objective function f_1, as explained in Section 4.3. Due to computational and time constraints, we were unable to analyse all GA variants on objective function f_1. In this section, we selected two groups of GA variants for which 33 runs were completed. Even with this limited analysis, we can still observe some interesting effects that the choice of objective functions has on the topological features of our network representations. All plots without normalisation can be found in Appendix E.

Figure 22 shows how normalisation of results on function f_1 with respect to function f_0 can sometimes reveal differences that are not apparent without normalisation. In Figure 22a, the fitness assortativity seems almost identical between variants with random or tournament parent selection. While both groups of variants produce higher values of fitness assortativity on f_1 compared to f_0, the increase for the variants with tournament selection is higher as the original expectation was lower (Figure 22b). In other words, according to Figure 22, objective function f_1 has a greater impact on algorithm dynamics for variants with tournament selection. In general, we observe the same pattern



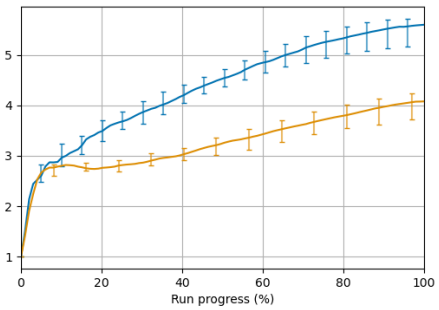
(a) Largest component size.

(b) Normalised largest component size.

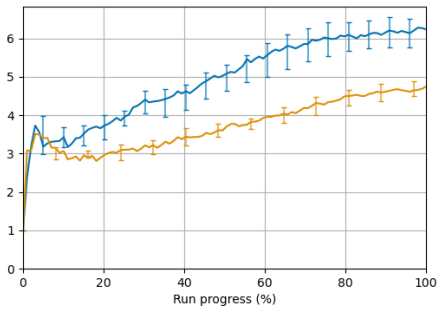
Figure 23: Median largest component size time trajectories on random objective function f_1 before and after normalisation. Every data point on f_1 is divided by the mean of its respective f_0 counterpart. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs. The normalisation reveals differences between random and tournament parent selection.

for other topological features (Appendix E). If a network feature increases or decreases between f_0 and f_1, the change is more dramatic for the variants using tournament selection. This shows that the increased parent selection pressure caused by tournament selection interacts more dramatically with the fitness landscape of function f_1.

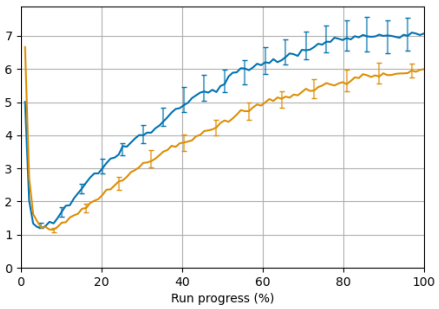
The network representations of GAs run on f_1 are larger (Figure 23), even less small-world-like (Figure 24) and less densely connected (Figure 25) than the networks for f_0. This difference is not very pronounced in the earlier iterations when the algorithms progress very quickly (Figure 27c, Appendix A.2). However, in later iterations, when progress slows down, the representations lose their hierarchical structure (Figure 26) and express much higher path lengths (Figure 24), much lower connectivity (Figure 25) and much lower centralisation (Figure 27a). This indicates that during the iterations with slow progress, regardless of the choice of parent selection, the solutions are contributing equally to the fitness improvements and exploration is dominating the minimisation process. We expected an increase in degree assortativity for a smooth objective function such as f_1. Even though there is an increase, it is not very large (Figure 27b).



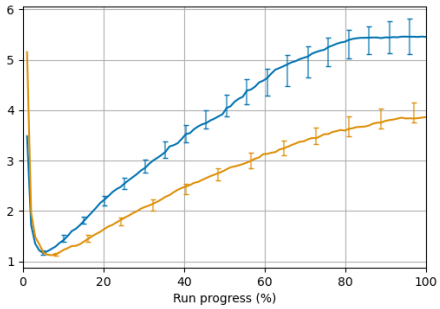
(a) Average shortest path length.



(b) Longest shortest path length.

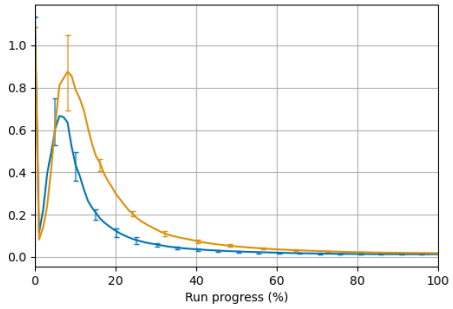


(c) Maximum leaf-leaf distance.



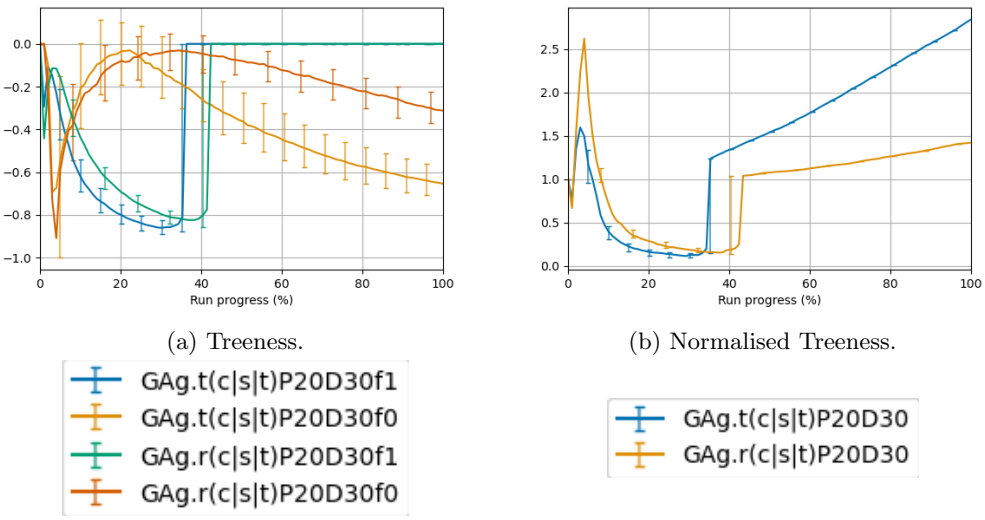
(d) Mean leaf-leaf distance.

Figure 24: Normalised median path length time trajectories on sphere function f_1. Every data point on f_1 is divided by the mean of its respective f_0 counterpart. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs.



(a) Algebraic connectivity.

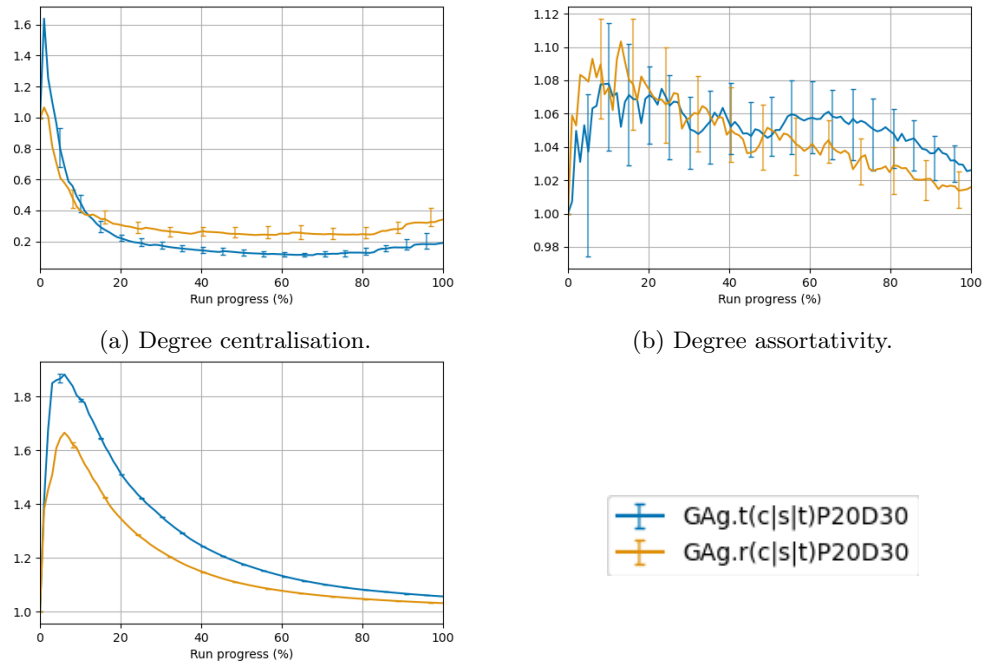
Figure 25: Normalised median Laplacian time trajectories on sphere function f_1. Every data point on f_1 is divided by the mean of its respective f_0 counterpart. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs.



(a) Treeness.

(b) Normalised Treeness.

Figure 26: Median treeness time trajectories on sphere function f_1 before and after normalisation. Every data point on f_1 is divided by the mean of its respective f_0 counterpart. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs. f_1 time trajectories lose their hierarchical structure during the run.



(a) Degree centralisation.

(b) Degree assortativity.

(c) Fitness assortativity.

Figure 27: Normalised median centralisation and assortativity time trajectories for different population sizes on sphere function f_1. Every data point on f_1 is divided by the mean of its respective f_0 counterpart. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs.

6 Discussion

In this thesis, we constructed network representations for a variety of genetic algorithms. The topological features of these representations are able to express the dynamic behaviour of the underlying GAs. However, there are several limitations to the analysis presented in this thesis.

For example, because there are no contemporary random network models with a structure similar to our GA networks, we compared our networks to randomly rewired networks, as explained in Section 4.3. It is important to realise that these shuffled networks are not extensively studied and may have their own biases. For instance, it has been shown before that the impact of shuffling edges depends on the structure and size of the original networks [47]. Without further investigation, it is not necessarily clear whether the networks are shuffled enough to remove the structure imposed by the GAs and to create a sufficiently random null model to compare against.

Another limiting factor in our analysis was the computational cost of calculating topological features and shuffling networks. Because our representations grow for each successful iteration, they are able to grow in size. Theoretically, the number of nodes can grow up to the number of function evaluations. Even though our representations are sparse, i.e. the number of edges is relatively small, the computation time of some topological features, such as treeness, scale badly with network size. Moreover, edge shuffling can become very slow for large networks, as the amount of shuffling needed depends on the size of the network. For these networks, it may be advantageous to implement a more efficient shuffling algorithm [48].

The analysis of only successful iterations is another limitation for three reasons. Firstly, unsuccessful iterations may provide valuable information about the interactions inside the GA population [6]. Secondly, these unsuccessful iterations provide information about algorithm runtime and convergence speed. Thirdly, this choice produces networks and time series that differ in size. Including unsuccessful iterations would prevent the need to interpolate and align time series of different lengths (Section 4.4). Interpolating may introduce noise or artificial data points that are not representative of the actual feature values. We considered several methods of aligning time series of different lengths. We decided not to discard data points to match the shortest run because this may result in a loss of valuable information. We also decided not to time series to match the length of the longest run. Our time series express similar highly random behaviour in the early iterations. In addition, the final iteration of each time series corresponds to the termination of each algorithm run. If we were to append values either to the beginning or end of our shorter time series, we would undo the alignment of these similar periods in different runs. Therefore, we opted to preserve the alignment of the beginning and end of our series by scaling and interpolating the time series to be equal in length.

In our experiments, we only analyse network-wide features and disregard their distribution with respect to individual nodes. As shown, these network-wide generalisations of features can easily be compared and used to identify trends, even if they exhibit large deviations. Because of the spread in network representation sizes, it becomes complicated to visualise and analyse distributions of individual node values. Unfortunately, by analysing network-wide abstractions, we may be discarding informative details. Especially centrality measures can be very informative when analysed as a distribution. For example, the degree distribution of nodes in a network can be used to identify scale-free networks [18]. Other centrality measures can be used to identify and count hub nodes in networks [5]. Analysing the change of distributions over time and comparing between variants may reveal more details about GA dynamics. Including node distributions may be an interesting opportunity for future work to create even more comprehensive network representation analyses.

In this thesis, we focused on the analysis of network topological features, but there are more features in our data that can be analysed. For example, using the real-valued chromosomes as coordinates (Table 1) of each solution, it is possible to calculate pairwise distances between solutions as an indicator of step size, or it is possible to calculate the size of the subdomain that is being explored. Furthermore, there are potential correlations between features that can be explored, e.g. the correlation between fitness and centrality measures. These additional features may be an interesting avenue for future research.

7 Conclusion

Genetic algorithms (GAs) and other population-based algorithms use an interacting population of solutions to optimise complicated objective functions. Even though they are widely applied to complex real world problems, their dynamic behaviour is not yet thoroughly understood. Recent research uses analysis of network representations of these algorithms to gain insight into the interactions and dynamics of population-based algorithms. While this approach has been used successfully to improve parameter selection or for real-time performance improvements, previous network representations may not accurately capture GA dynamics.

In this thesis, we constructed network representations, specifically for real-valued genetic algorithms, designed to capture the spread and inheritance of genetic information in GA populations. We defined a set of network topological features and related these features to GA dynamics by comparing their values for different GA parameters and objective functions. We showed how the effect of GA parameters can be quantified by analysing the temporal evolution of these features. We created ensembles of random graphs to use as null models using a random objective function or random rewiring. Next, we used these null models to observe to what extent changes in topological features differ from what can be expected by chance.

On a random objective function, we found that increasing population size increases the number of nodes and increased path lengths in the network representations, while connectivity decreased. Tournament parent selection decreased path lengths and treeness, and increased connectivity and centralisation of the networks compared to random parent selection. These effects can be explained by the higher parent selection pressure in GAs with tournament selection. Unexpectedly, tournament selection also decreases fitness assortativity. As expected, we did not observe any effect of changing mutation operators, crossover operators or correction strategy on a random objective function.

We found that GAs increase path lengths and fitness assortativity, while they decrease degree assortativity, connectivity, and treeness of their representations, compared to their randomly shuffled counterparts for some variants. This shows that the topological features we observe are not caused by our method of representation or by chance, but can be attributed to the underlying genetic algorithm.

The network representations increased in size, path lengths, and treeness while decreasing in connectivity, centralisation, and degree assortativity when the GAs optimised a non-random objective function, compared to a random objective function. While we observed these effects for GA variants using both parent selection methods, these differences were greater for variants with tournament selection than for variants with random selection.

We suggest the following points of interest for future research: Firstly, the analysis of the time evolution of node degree and node centrality distributions can be used to perform a more detailed analysis of network representations. Secondly, the network representations in this thesis could be extended to include unsuccessful iterations. Thirdly, we suggest increasing the scope of analysis by analysing solution coordinates or correlations between topological features. Fourthly, a deeper analysis of shuffled or other random network null models may be needed to validate the significance of GA network features. Finally, we suggest expanding this method of analysis to more GA variants and objective functions to further characterise GA behaviour. We think that our network-based approach will help improve the understanding of GAs and may contribute to GA performance improvements by providing additional opportunities for real-time control in future work.

References

[1] F. Caraffini, A. V. Kononova, and D. W. Corne. "Infeasibility and structural bias in Differential Evolution". In: *Information Sciences* 496 (May 2019), pp. 161–179.

[2] A. V. Kononova et al. "Structural bias in population-based algorithms". In: *Information Sciences* 298 (2015), 468–490. ISSN: 0020-0255.

[3] M. Metlicka and D. Davendra. "Ensemble centralities based adaptive Artificial Bee algorithm". In: *2015 IEEE Congress on Evolutionary Computation (CEC)*. 2015, pp. 3370–3376.

[4] D. Davendra et al. "Complex Network Analysis of Discrete Self-organising Migrating Algorithm". In: *Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems*. Cham: Springer International Publishing, 2014, pp. 161–174. ISBN: 978-3-319-07401-6.

[5] D. Davendra et al. "Complex Network Analysis of Evolutionary Algorithms Applied to Combinatorial Optimisation Problem". In: *Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014*. Ed. by P. Kömer, A. Abraham, and V. Snášel. Cham: Springer International Publishing, 2014, pp. 141–150. ISBN: 978-3-319-08156-4.

[6] I. Zelinka et al. "Evolutionary algorithms dynamics and its hidden complex network structures". In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. 2014, pp. 3246–3251.

[7] S. Katoch, S. S. Chauhan, and V. Kumar. "A review on genetic algorithm: past, present, and future". In: *Multimedia Tools and Applications* 80.5 (2020), 8091–8126.

[8] G. Guariso and M. Sangiorgio. "Improving the Performance of Multiobjective Genetic Algorithms: An Elitism-Based Approach". In: *Information* 11.12 (2020). ISSN: 2078-2489.

[9] X.-S. Yang. "Chapter 5 - Genetic Algorithms". In: *Nature-Inspired Optimization Algorithms*. Ed. by X.-S. Yang. Oxford: Elsevier, 2014, pp. 77–87. ISBN: 978-0-12-416743-8.

[10] D. Ashlock, C. McGuinness, and W. Ashlock. "Representation in Evolutionary Computation". In: *Advances in Computational Intelligence: IEEE World Congress on Computational Intelligence, WCCI 2012, Brisbane, Australia, June 10-15, 2012. Plenary/Invited Lectures*. Ed. by J. Liu et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 77–97. ISBN: 978-3-642-30687-7.

[11] R. Soman and P. Malinowski. "A Real-Valued Genetic Algorithm for Optimization of Sensor Placement for Guided Wave-Based Structural Health Monitoring". In: *Journal of Sensors* 2019 (2019), 1–10.

[12] F. Akdeniz et al. "Application of real valued genetic algorithm on prediction of higher heating values of various lignocellulosic materials using lignin and extractive contents". In: *Energy* 160 (2018), pp. 1047–1054. ISSN: 0360-5442.

[13] E. Michielssen, S. Ranjithan, and R. Mittra. "Optimal multilayer filter design using real coded genetic algorithms". English. In: *IEE Proceedings J (Optoelectronics)* 139 (6 1992), 413–420(7). ISSN: 0267-3932.

[14] E. G. Johnson et al. "Advantages of genetic algorithm optimization methods in diffractive optic design". In: *Diffractive and Miniaturized Optics: A Critical Review*. Ed. by S. H. Lee. Vol. 10271. International Society for Optics and Photonics. SPIE, 1993, pp. 56 –76.

[15] A. Eiben and S. Smit. "Parameter tuning for configuring and analyzing evolutionary algorithms". In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 19–31. ISSN: 2210-6502.

[16] A. Eiben, R. Hinterding, and Z. Michalewicz. "Parameter control in evolutionary algorithms". In: *IEEE Trans. Evolutionary Computation* 3 (Aug. 1999), pp. 124–141.

[17] S. Boccaletti et al. "Complex networks: Structure and dynamics". In: *Physics Reports* 424.4 (2006), pp. 175–308. ISSN: 0370-1573.

[18] A.-L. Barabási and M. Pósfai. *Network science*. Cambridge University Press, 2017.

[19] A. Fornito, E. T. Bullmore, and A. Zalesky. *Fundamentals of Brain Network Analysis*. Elsevier, 2016.

[20] F. Klimm et al. “Resolving structural variability in network models and the brain”. In: *PLoS Computational Biology* 10.3 (2014).

[21] F. Váša and B. Mišić. “Null models in network neuroscience”. In: *Nature Reviews Neuroscience* 23.8 (2022), 493–504.

[22] I. Zelinka et al. “Do Evolutionary Algorithm Dynamics Create Complex Network Structures?” In: *Complex Systems* 20.2 (2011), 127–140.

[23] I. Zelinka, D. Davendra, and L. Skanderova. “Visualization of Complex Networks Dynamics: Case Study”. In: *NETWORKING 2012 Workshops*. Ed. by Z. Becvar, R. Bestak, and L. Kencl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 145–150. ISBN: 978-3-642-30039-4.

[24] I. Zelinka et al. “A novel approach on evolutionary dynamics analysis – A progress report”. In: *Journal of Computational Science* 25 (2018), pp. 437–445. ISSN: 1877-7503.

[25] L. Skanderova, T. Fabian, and I. Zelinka. “Differential Evolution Enhanced by the Closeness Centrality: Initial Study”. In: *2015 International Conference on Intelligent Networking and Collaborative Systems*. 2015, pp. 346–353.

[26] L. Skanderova, T. Fabian, and I. Zelinka. “Differential Evolution Dynamics Modeled by Longitudinal Social Network”. In: *Journal of Intelligent Systems* 26.3 (2017), pp. 523–529.

[27] L. Skanderova, T. Fabian, and I. Zelinka. “Differential evolution based on node strength”. In: *International Journal of Bio-Inspired Computation* 11.1 (2018), p. 34.

[28] N. Hansen et al. “Real-Parameter Black-Box Optimization Benchmarking 2009: Noisy Functions Definitions”. In: (Jan. 2009).

[29] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. 2nd. Springer Publishing Company, Incorporated, 2015. ISBN: 3662448734.

[30] Q. Wu and R. Law. “Cauchy mutation based on objective variable of Gaussian particle swarm optimization for parameters selection of SVM”. In: *Expert Systems with Applications* 38.6 (2011), pp. 6405–6411. ISSN: 0957-4174.

[31] A. V. Kononova et al. “Can Single Solution Optimisation Methods Be Structurally Biased?” In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. 2020, pp. 1–9.

[32] A. V. Kononova, F. Caraffini, and T. Bäck. “Differential evolution outside the box”. In: *Information Sciences* 581 (2021), pp. 587–604. ISSN: 0020-0255.

[33] A. A. Hagberg, D. A. Schult, and P. J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by G. Varoquaux, T. Vaught, and J. Millman. Pasadena, CA USA, 2008, pp. 11 –15.

[34] M. E. J. Newman. “Mixing patterns in networks”. In: *Physical Review E* 67.2 (2003). ISSN: 1095-3787.

[35] N. Litvak and R. van der Hofstad. “Uncovering disassortativity in large scale-free networks”. In: *Physical Review E* 87.2 (2013).

[36] S. Kokoska. *CRC standard probability and statistics tables and formulae*. CRC Press, 2017.

[37] L. C. Freeman. “Centrality in social networks conceptual clarification”. In: *Social Networks* 1.3 (1978), pp. 215–239. ISSN: 0378-8733.

[38] U. Brandes. “On variants of shortest-path betweenness centrality and their generic computation”. In: *Social Networks* 30.2 (2008), pp. 136–145. ISSN: 0378-8733.

[39] N. Jovanovic, Z. Jovanovic, and A. Jevremovic. “Complex Networks Analysis by Spectral Graph Theory”. In: Apr. 2017.

[40] A. E. Brouwer and W. H. Haemers. *Spectra of graphs*. 2011.

[41] B. Corominas-Murtra et al. “On the origins of hierarchy in complex networks”. In: *Proceedings of the National Academy of Sciences* 110.33 (2013), pp. 13316–13321. ISSN: 0027-8424.

[42] B. Corominas-Murtra et al. “Measuring the hierarchy of feedforward networks”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 21.1 (2011), p. 016108. ISSN: 1089-7682.

[43] D. J. Watts and S. H. Strogatz. “Collective dynamics of ‘small-world’ networks”. In: *Nature* 393.6684 (1998), 440–442.

[44] S. Maslov and K. Sneppen. “Specificity and stability in topology of protein networks”. In: *Science* 296.5569 (2002), 910–913.

[45] R. Milo et al. “On the uniform generation of random graphs with prescribed degree sequences”. In: (2003).

[46] C. R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362.

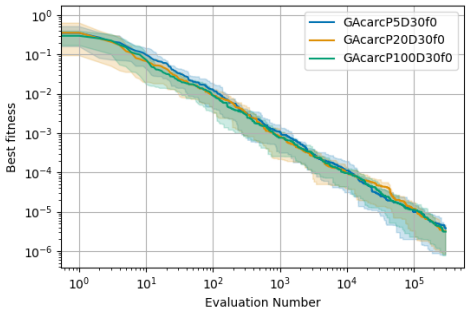
[47] M. Tantardini et al. “Comparing methods for comparing networks”. In: *Scientific Reports* 9.1 (2019), p. 17557. ISSN: 2045-2322.

[48] F. Viger and M. Latapy. “Efficient and simple generation of random simple connected graphs with prescribed degree sequence”. In: *Lecture Notes in Computer Science* (2005), 440–449.

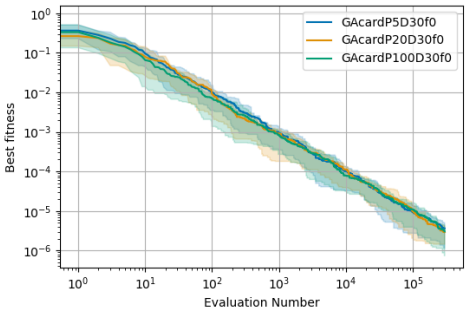
Appendices

Appendix A GA Performance Plots

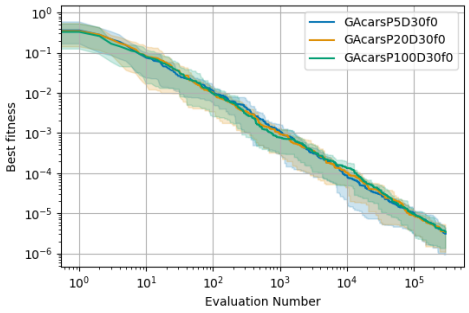
A.1 GA Performance on f_0



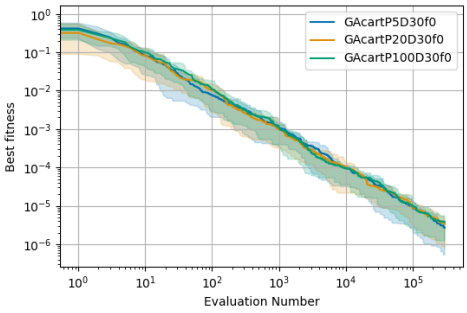
(a) GAcarcP(.)D30f0.



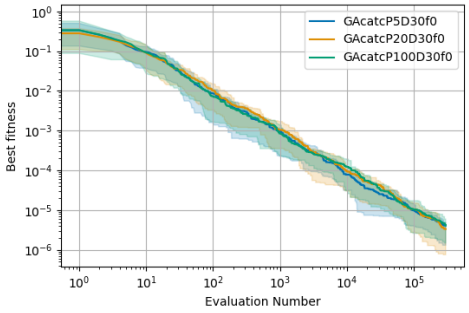
(b) GAcardP(.)D30f0.



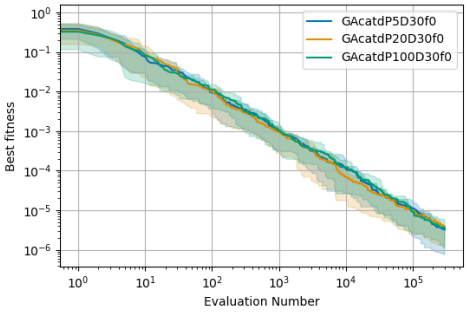
(c) GAcarsP(.)D30f0.



(d) GAcartP(.)D30f0.

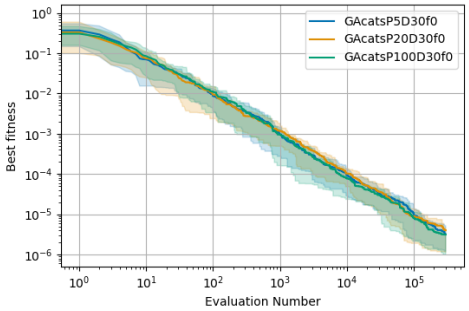


(e) GAcatcP(.)D30f0.

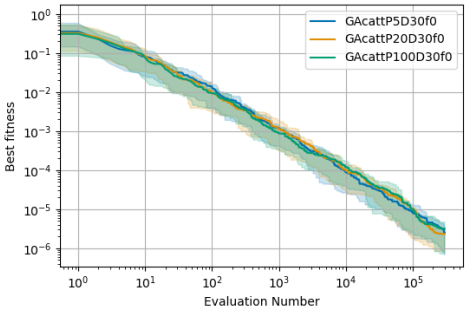


(f) GAcatdP(.)D30f0.

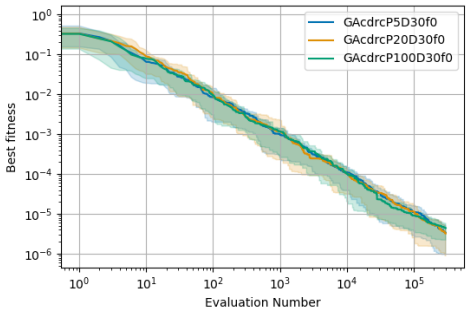
Figure A.1: Performance of GAs with Cauchy mutation on f_0. The legend labels containing regular expressions that represent groups of GA variants are explained in Section 3.3.



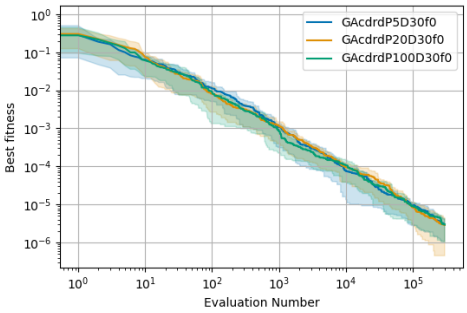
(g) GAcatsP(.)D30f0.



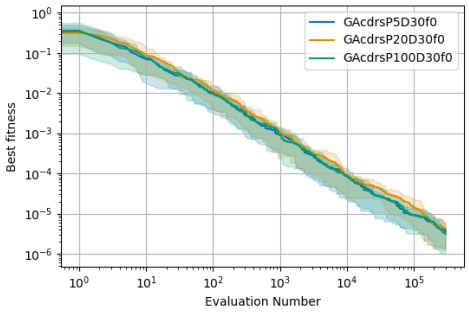
(h) GAcattP(.)D30f0.



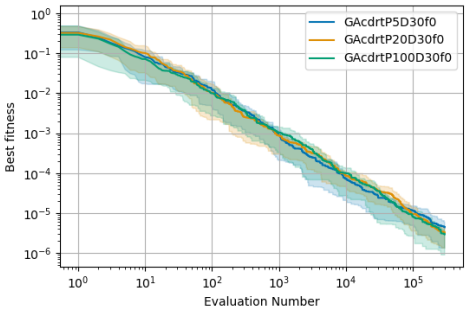
(i) GAcdrcP(.)D30f0.



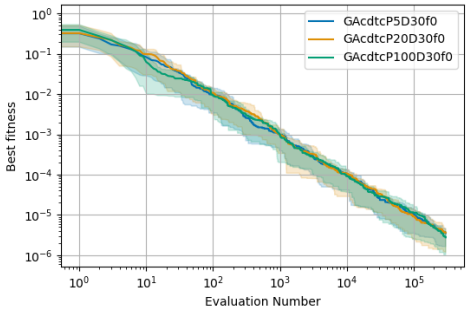
(j) GAcdrdP(.)D30f0.



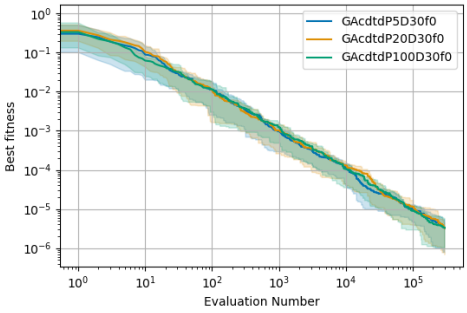
(k) GAcdrsP(.)D30f0.



(l) GAcdrtP(.)D30f0.

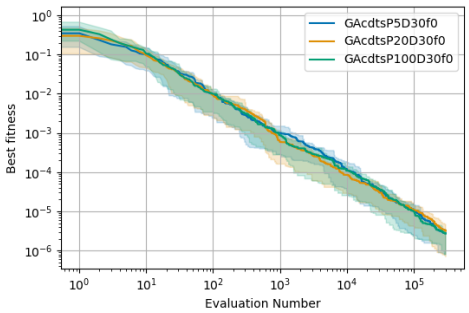


(m) GAcdtcP(.)D30f0.

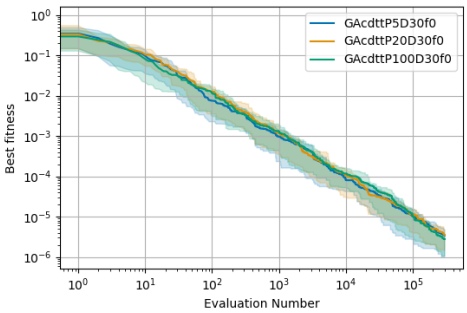


(n) GAcdtdP(.)D30f0.

Figure A.1: Performance of GAs with Cauchy mutation on f_0.

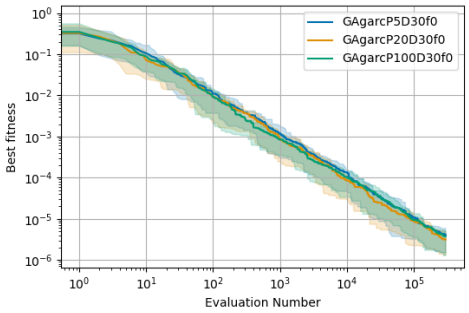


(o) GAcdtsP(.)D30f0.

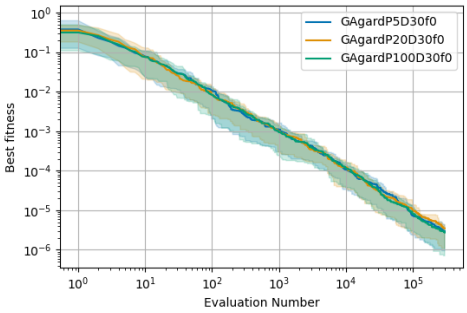


(p) GAcdttP(.)D30f0.

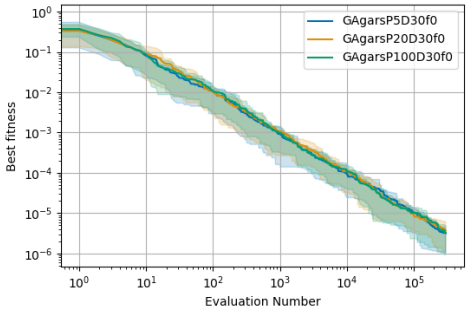
Figure A.1: Performance of GAs with Cauchy mutation on f_0.



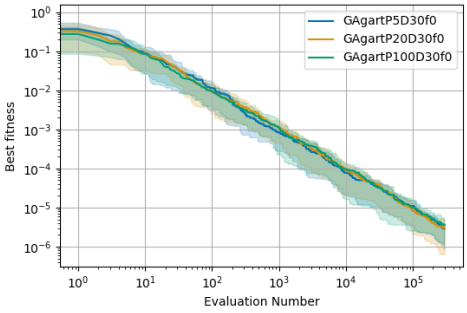
(a) GAgarcP(.)D30f0.



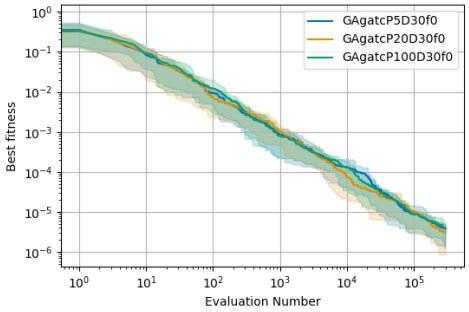
(b) GAgardP(.)D30f0.



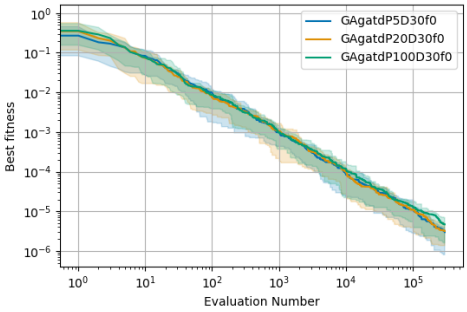
(c) GAgarsP(.)D30f0.



(d) GAgartP(.)D30f0.

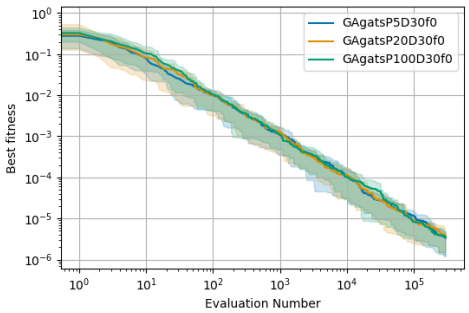


(e) GAgatcP(.)D30f0.

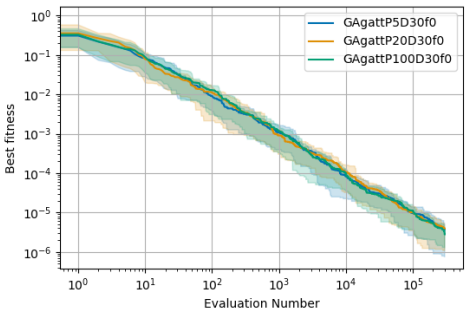


(f) GAgatdP(.)D30f0.

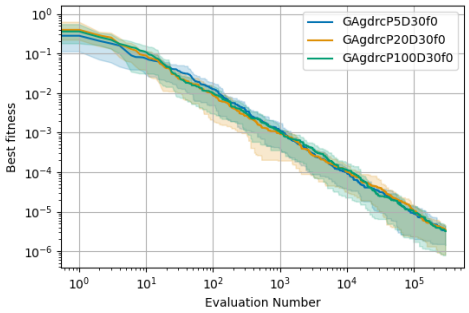
Figure A.2: Performance of GAs with Gaussian mutation on f_0.



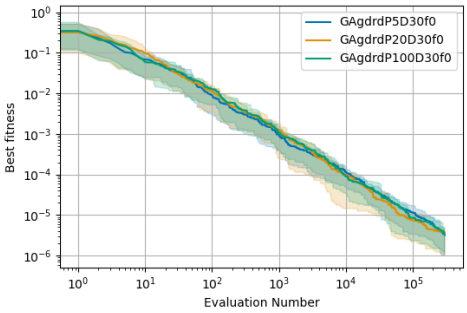
(g) GAgatsP(.)D30f0.



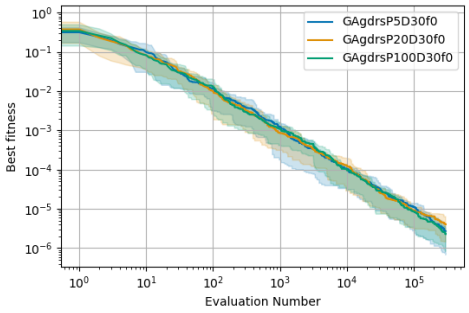
(h) GAgattP(.)D30f0.



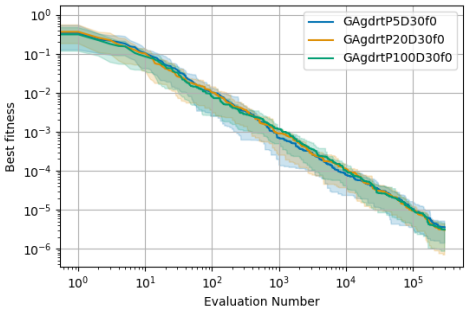
(i) GAgdrcP(.)D30f0.



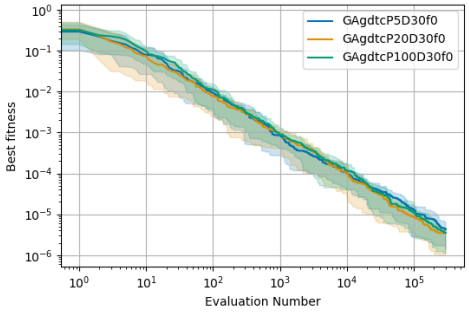
(j) GAgdrdP(.)D30f0.



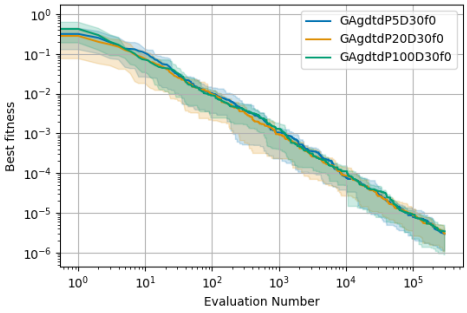
(k) GAgdrsP(.)D30f0.



(l) GAgdrtP(.)D30f0.

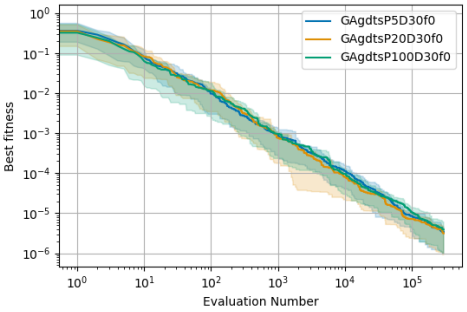


(m) GAgdtcP(.)D30f0.

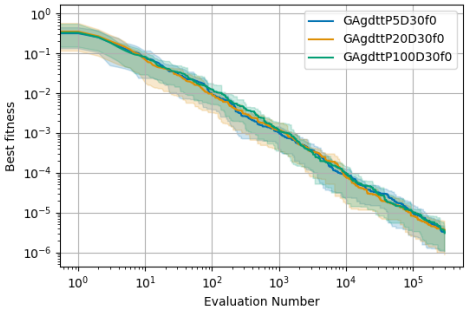


(n) GAgdtdP(.)D30f0.

Figure A.2: Performance of GAs with Gaussian mutation on f_0.



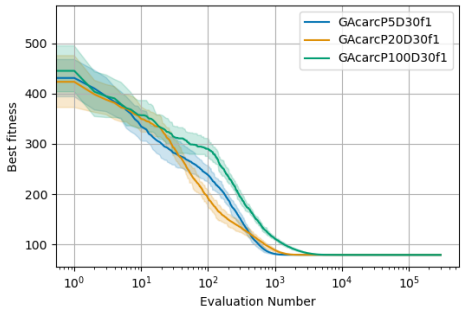
(o) GAgdtsP(.)D30f0.



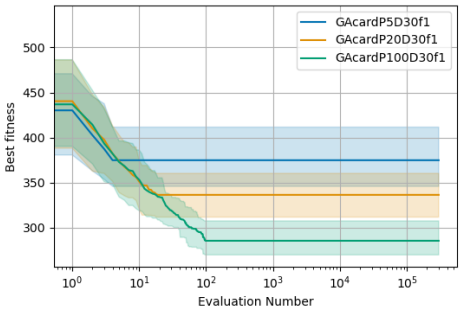
(p) GAgdttP(.)D30f0.

Figure A.2: Performance of GAs with Gaussian mutation on f_0.

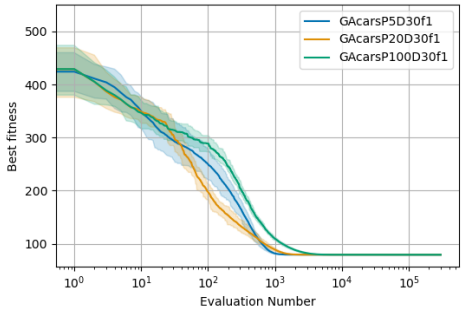
A.2 GA Performance on f_1



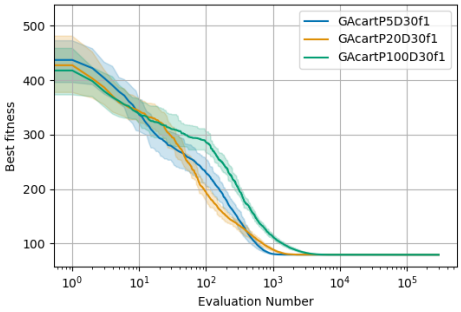
(a) GAcarcP(.)D30f1.



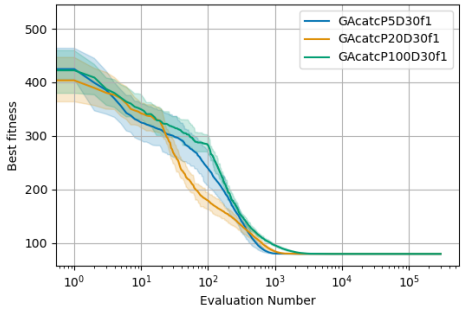
(b) GAcardP(.)D30f1.



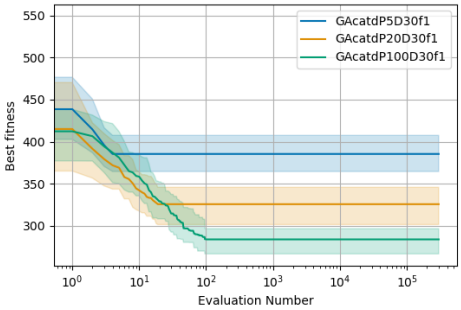
(c) GAcarsP(.)D30f1.



(d) GAcartP(.)D30f1.

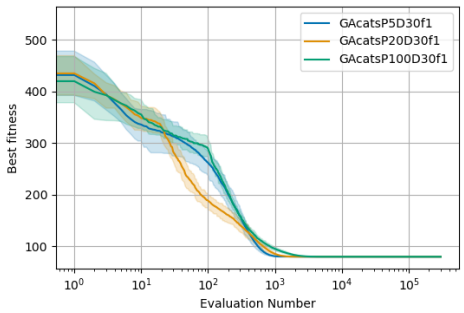


(e) GAcatcP(.)D30f1.

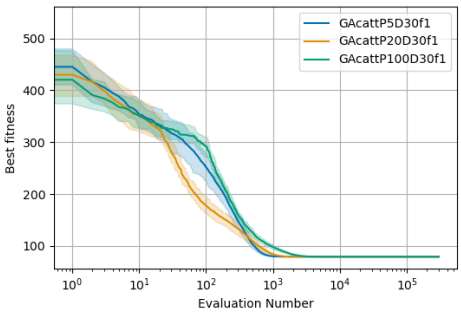


(f) GAcatdP(.)D30f1.

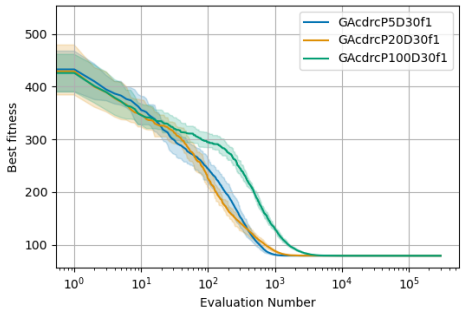
Figure A.3: Performance of GAs with Cauchy mutation on f_1. The legend labels containing regular expressions that represent groups of GA variants are explained in Section 3.3.



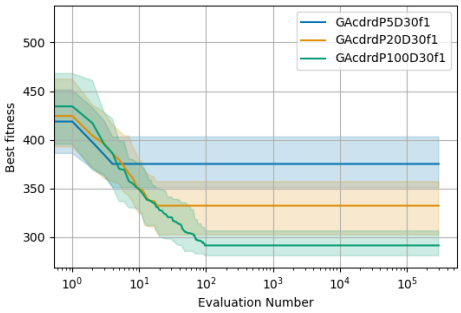
(g) GAcatsP(.)D30f1.



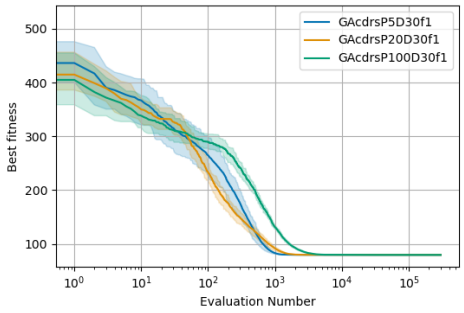
(h) GAcattP(.)D30f1.



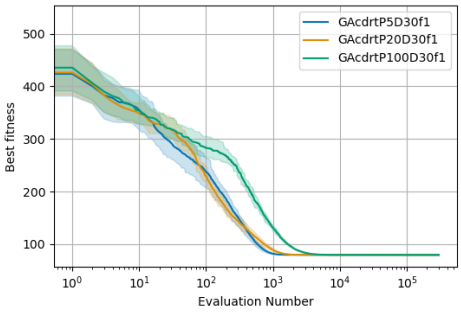
(i) GAcdrcP(.)D30f1.



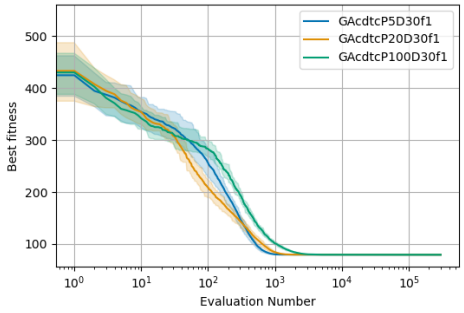
(j) GAcdrdP(.)D30f1.



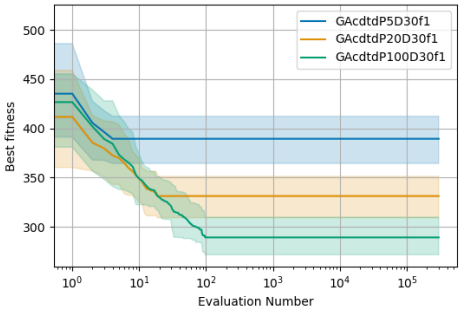
(k) GAcdrsP(.)D30f1.



(l) GAcdrtP(.)D30f1.

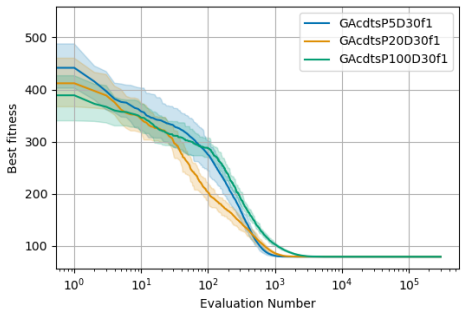


(m) GAcdtcP(.)D30f1.

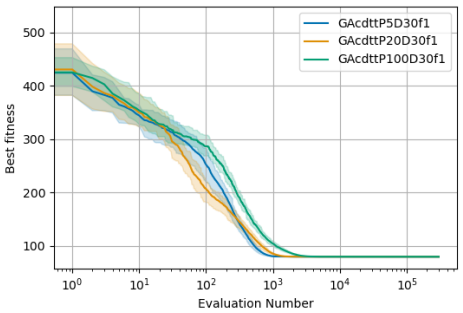


(n) GAcdtdP(.)D30f1.

Figure A.3: Performance of GAs with Cauchy mutation on f_1.

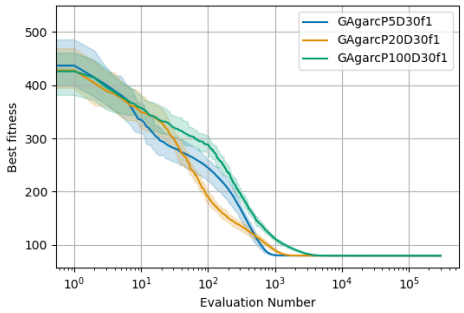


(o) GAcdtsP(.)D30f1.

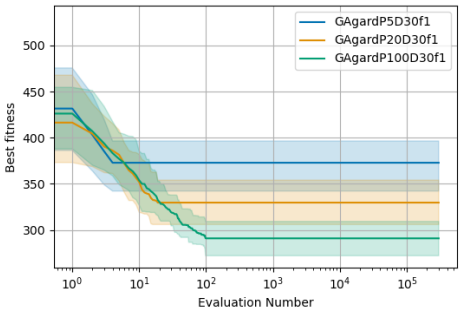


(p) GAcdttP(.)D30f1.

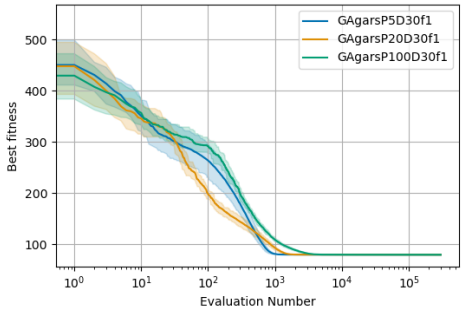
Figure A.3: Performance of GAs with Cauchy mutation on f_1.



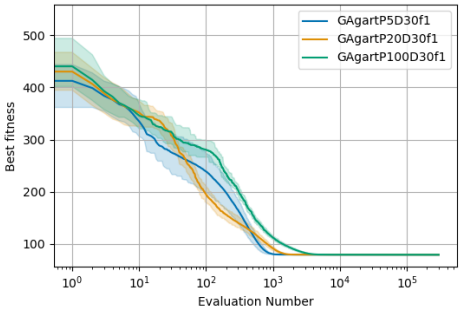
(a) GAgarcP(.)D30f1.



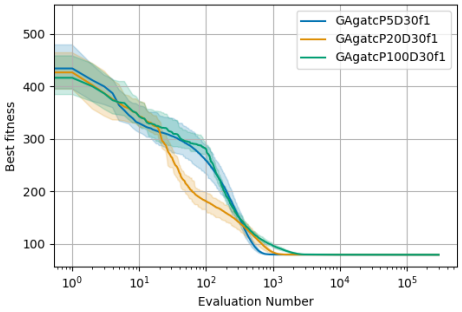
(b) GAgardP(.)D30f1.



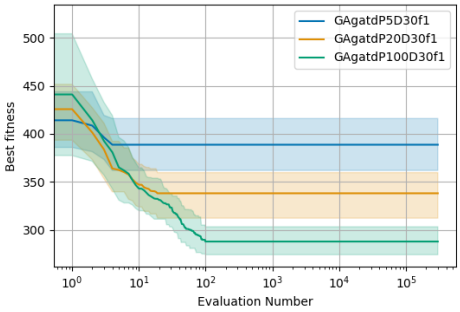
(c) GAgarsP(.)D30f1.



(d) GAgartP(.)D30f1.

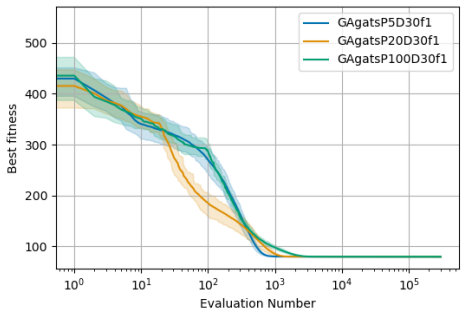


(e) GAgatcP(.)D30f1.

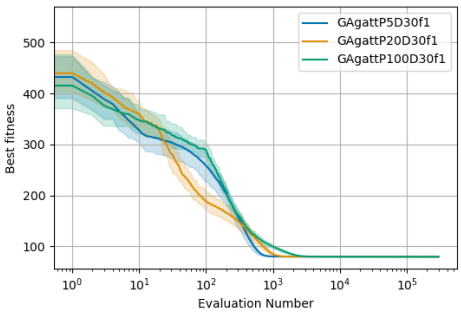


(f) GAgatdP(.)D30f1.

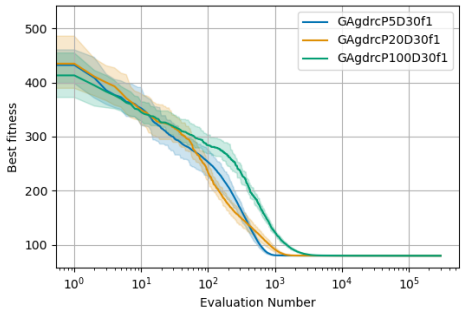
Figure A.4: Performance of GAs with Gaussian mutation on f_1.



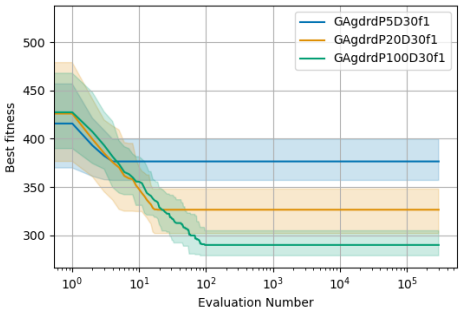
(g) GAgatsP(.)D30f1.



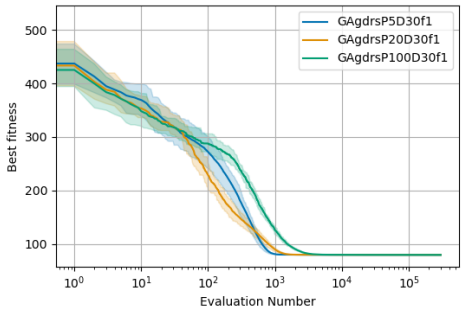
(h) GAgattP(.)D30f1.



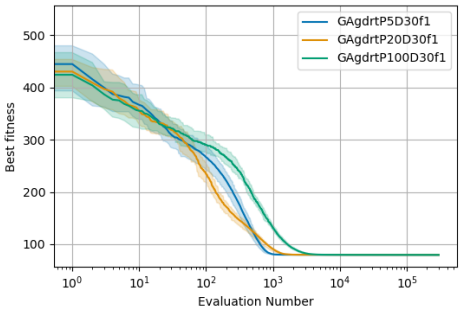
(i) GAgdrcP(.)D30f1.



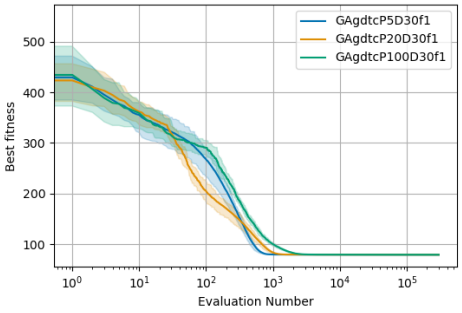
(j) GAgdrdP(.)D30f1.



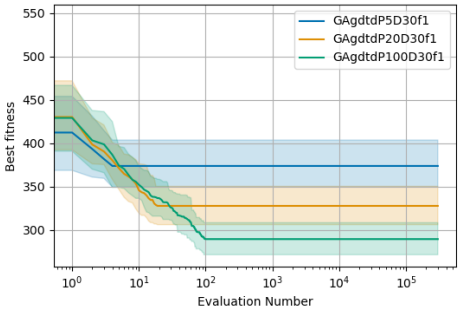
(k) GAgdrsP(.)D30f1.



(l) GAgdrtP(.)D30f1.

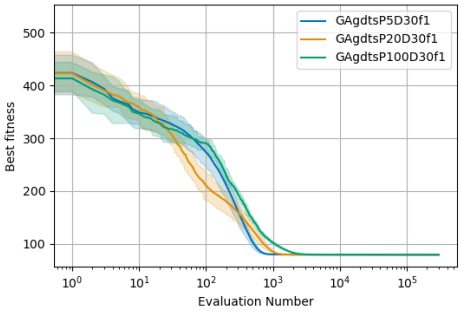


(m) GAgdtcP(.)D30f1.

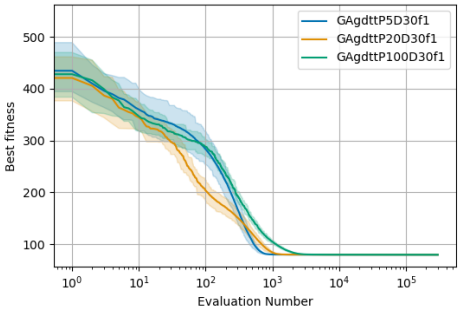


(n) GAgdtdP(.)D30f1.

Figure A.4: Performance of GAs with Gaussian mutation on f_1.



(o) GAgdtsP(.)D30f1.



(p) GAgdttP(.)D30f1.

Figure A.4: Performance of GAs with Gaussian mutation on f_1.

Appendix B Network Pictures

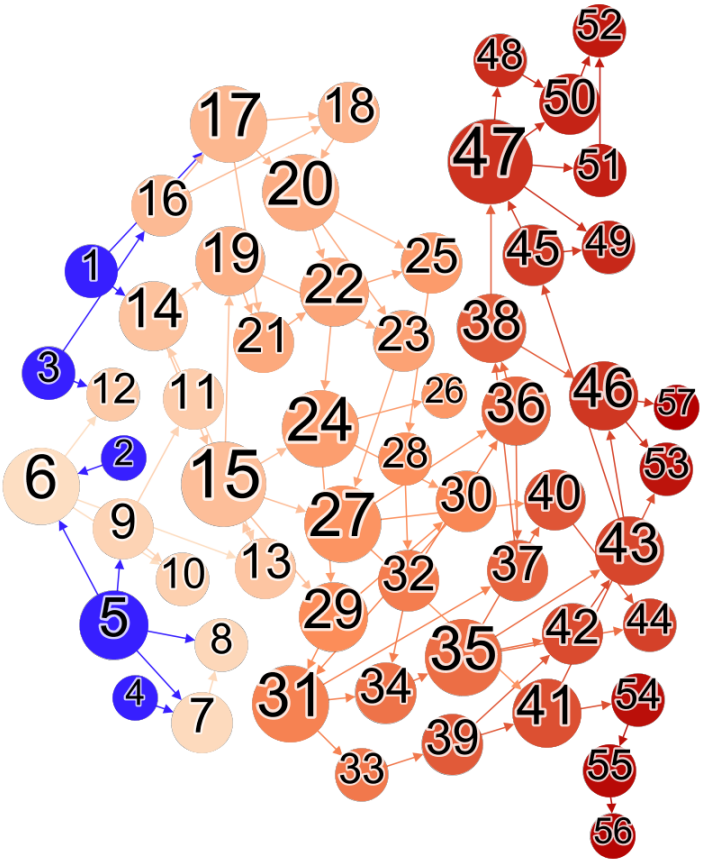


Figure B.1: One run of GA variant GAcarsP5D30f0 (as explained in Section 3.3) visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 57 nodes and 88 edges.

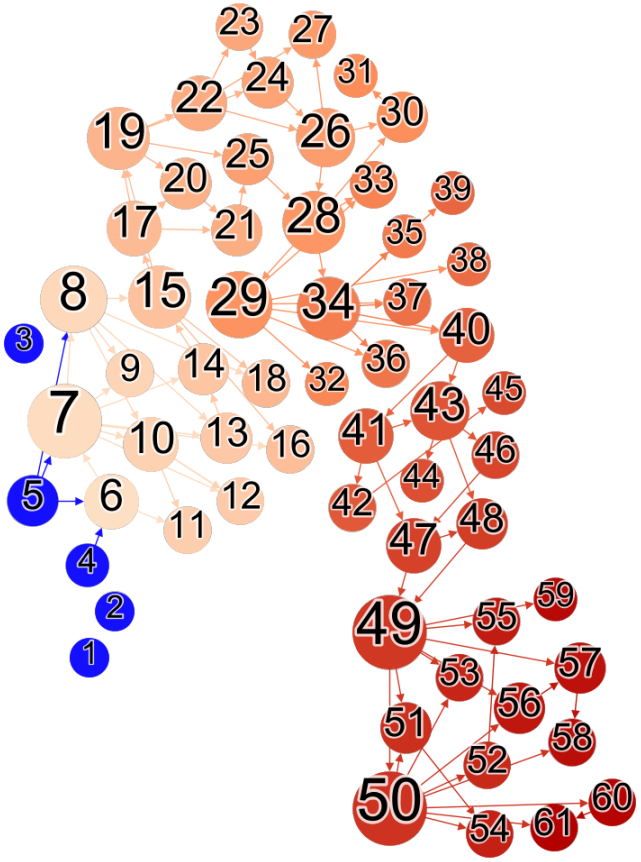


Figure B.2: One run of GA variant GAcatsP5D30f0 visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 61 nodes and 93 edges.

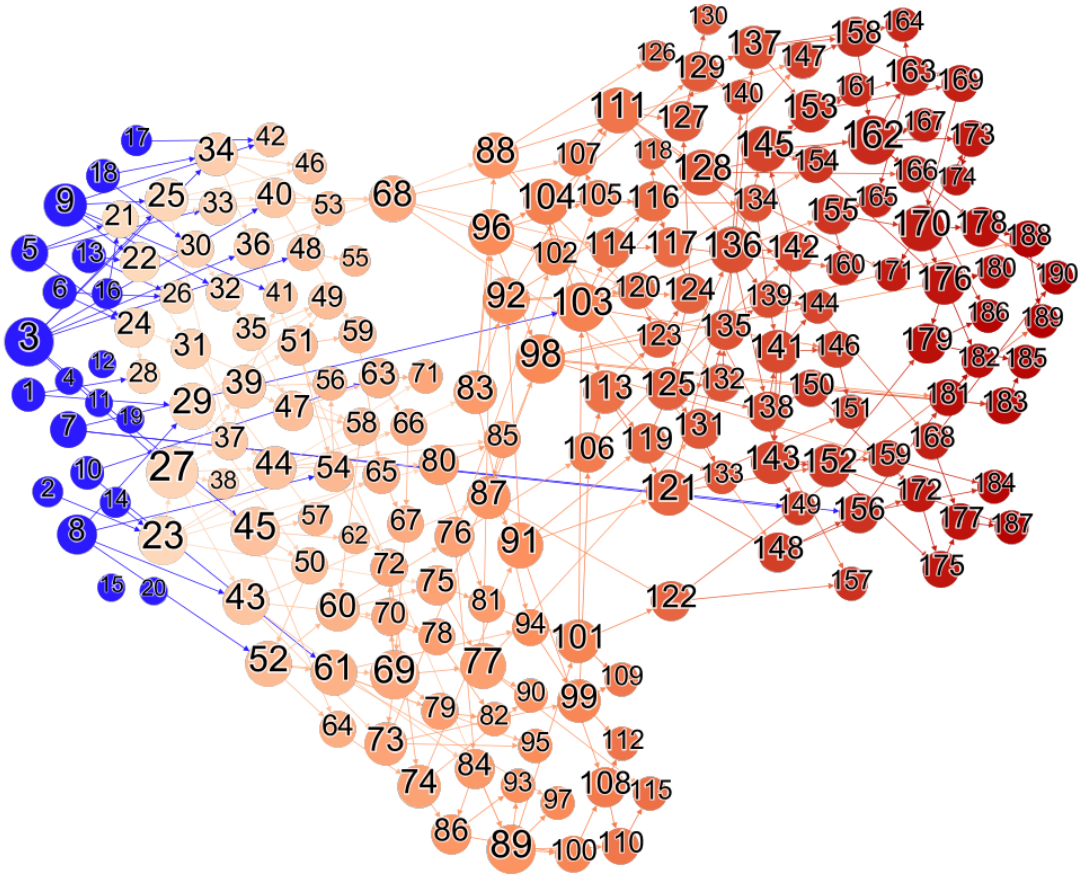


Figure B.3: One run of GA variant GAgdrcP20D30f0 visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 190 nodes and 326 edges.

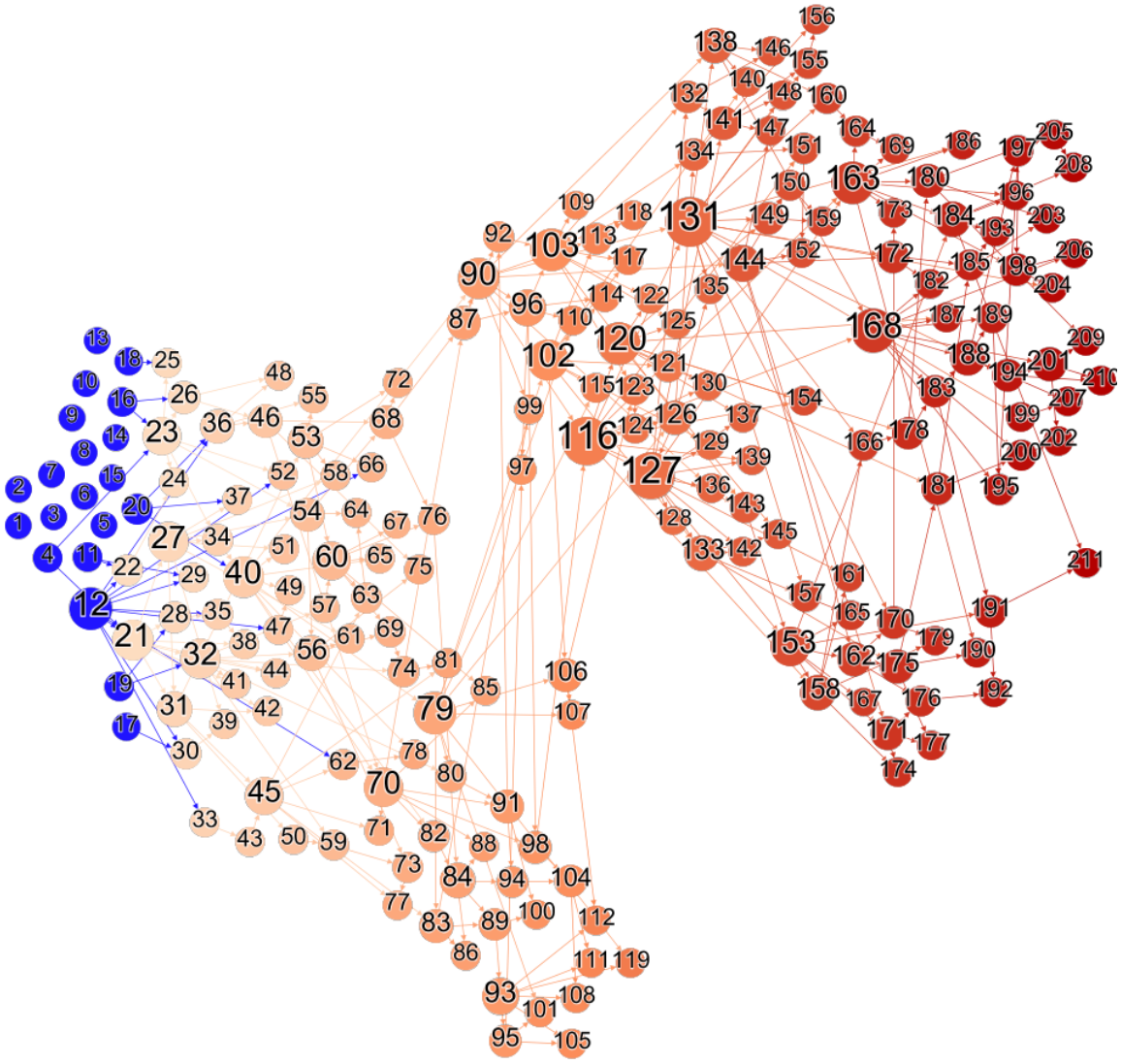


Figure B.4: One run of GA variant GAgdtcP20D30f0 visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 211 nodes and 363 edges.

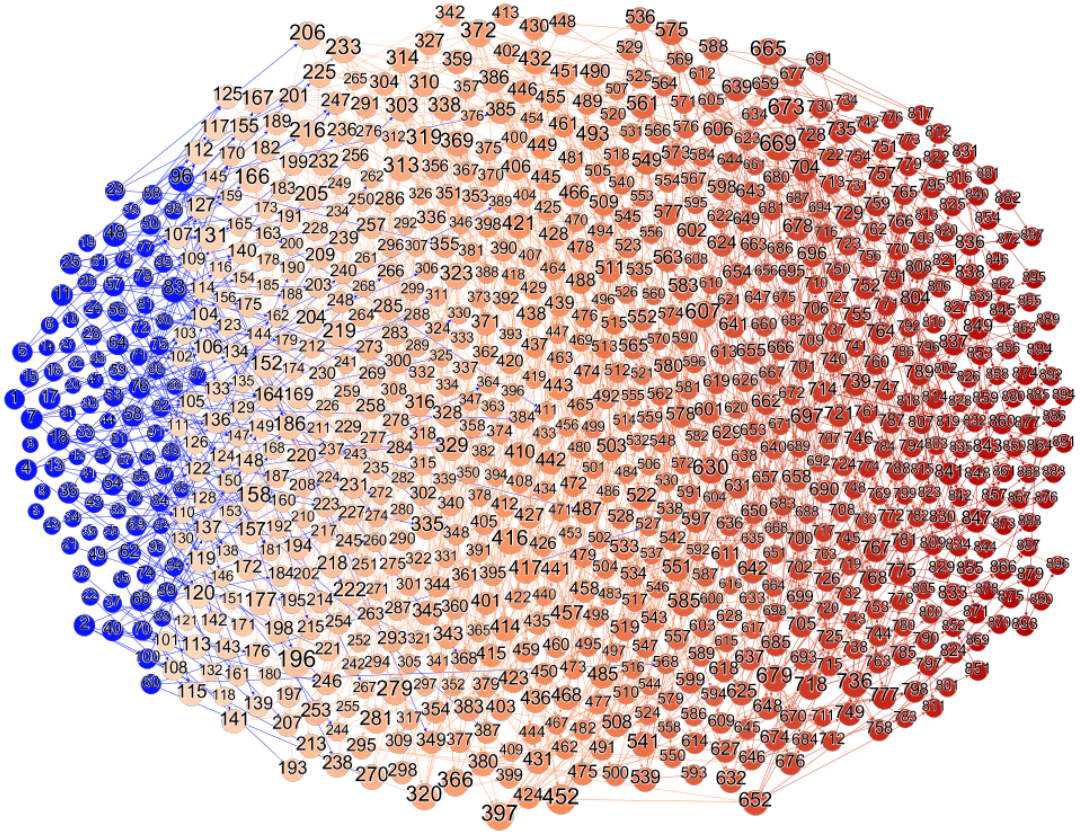


Figure B.5: One run of GA variant GAcdrtP100D30f0 visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 897 nodes and 1574 edges.

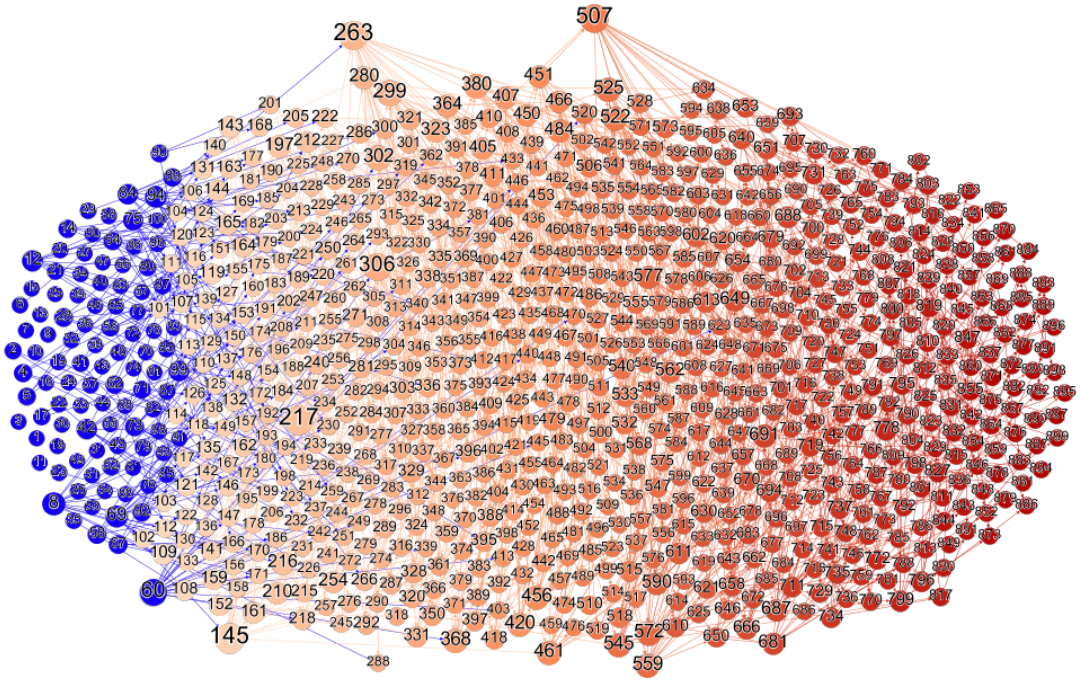


Figure B.6: One run of GA variant GAcdttP100D30f0 visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 899 nodes and 1592 edges.

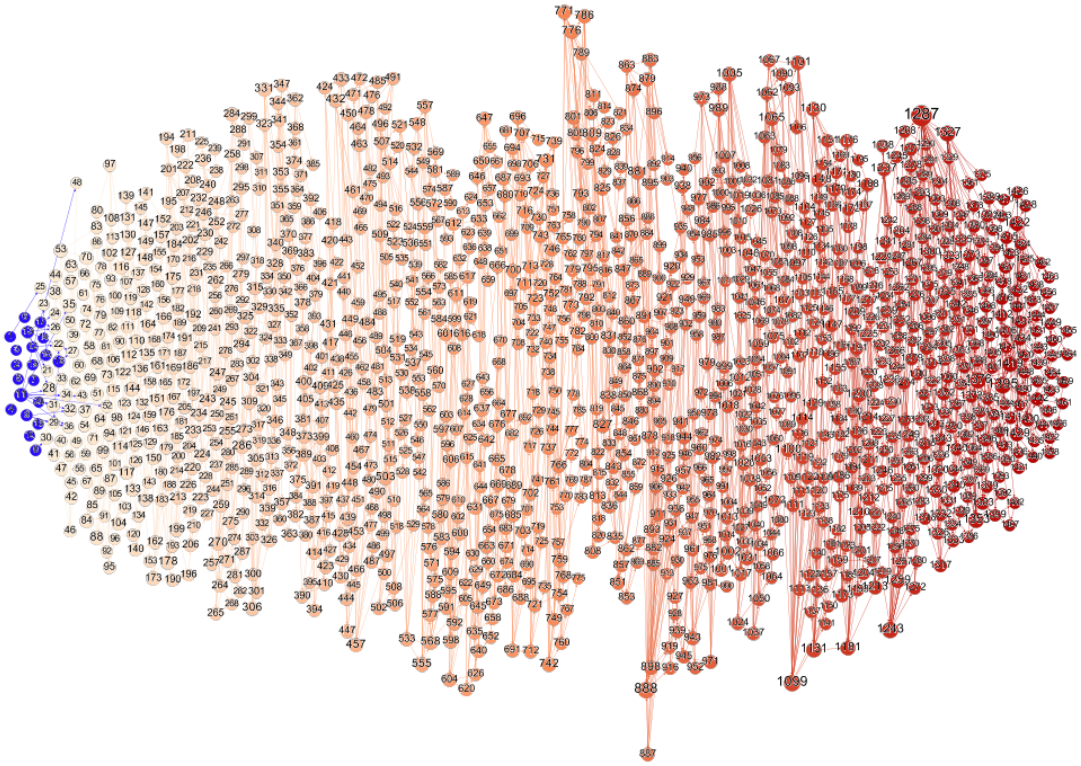
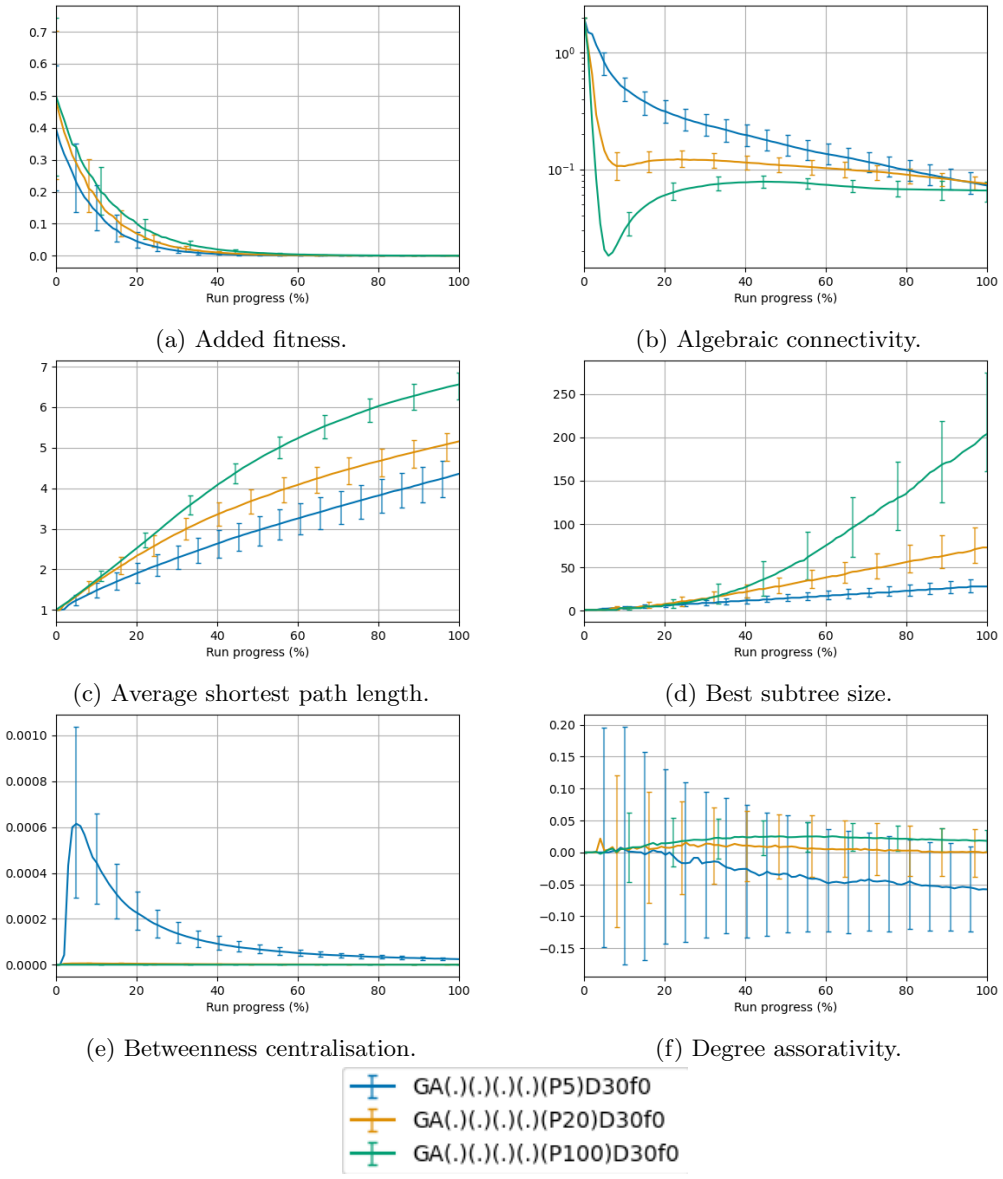


Figure B.7: One run of GA variant GAgdtcP20D30f1 visualised as a network. Nodes are numbered and coloured by the iteration at which they are added to the network. The 20 nodes that comprise the initial population are coloured blue. The size of the nodes is proportional to their combined in and out-degree. The network has 1466 nodes and 2786 edges.

Appendix C Plots for Random Function f_0

C.1 Effect of Population Size



(a) Added fitness.

(b) Algebraic connectivity.

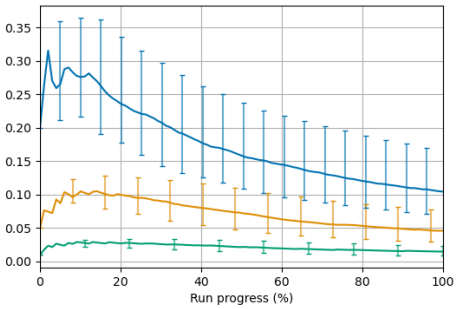
(c) Average shortest path length.

(d) Best subtree size.

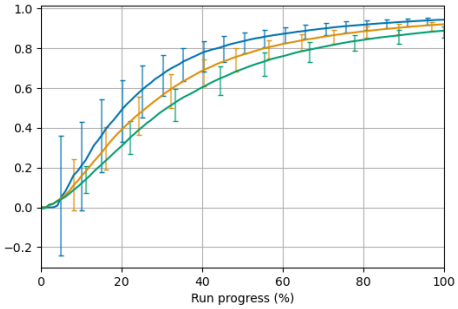
(e) Betweenness centralisation.

(f) Degree assorativity.

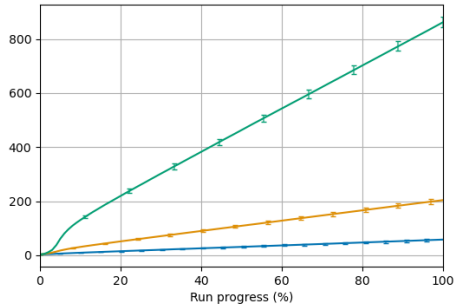
Figure C.1: Median time trajectories for different population sizes on random objective function f_0. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.



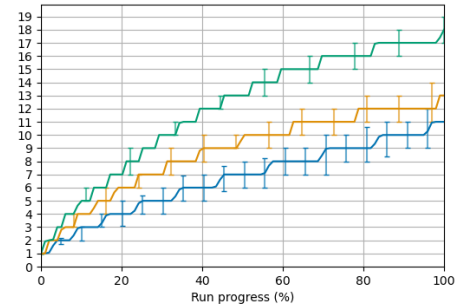
(g) Degree centralisation.



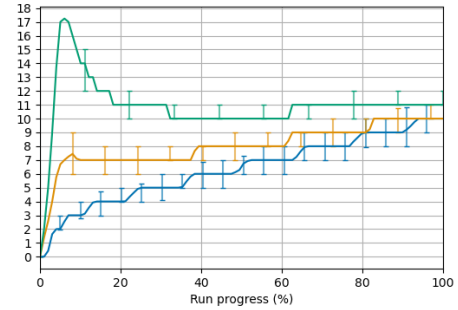
(h) Fitness assorativity.



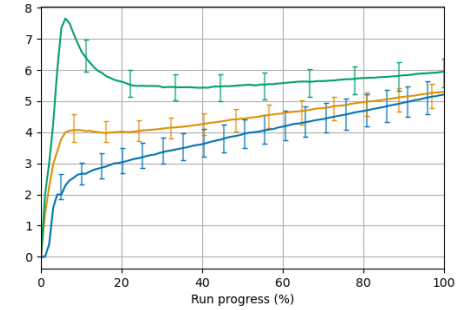
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

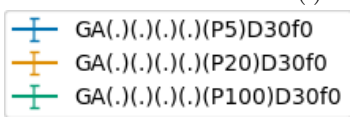
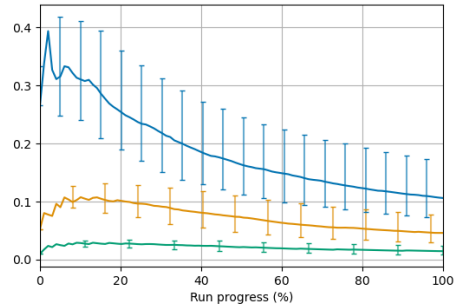
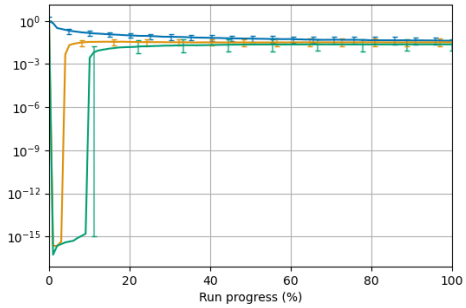


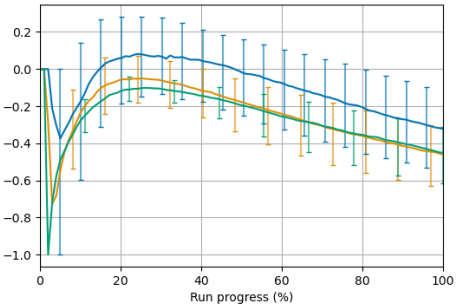
Figure C.1: Median time trajectories for different population sizes on random objective function f_0. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.



(m) Out-degree centralisation.



(n) Spectral gap.



(o) Treeness.

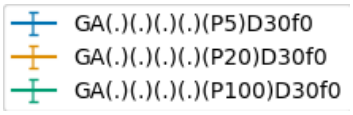
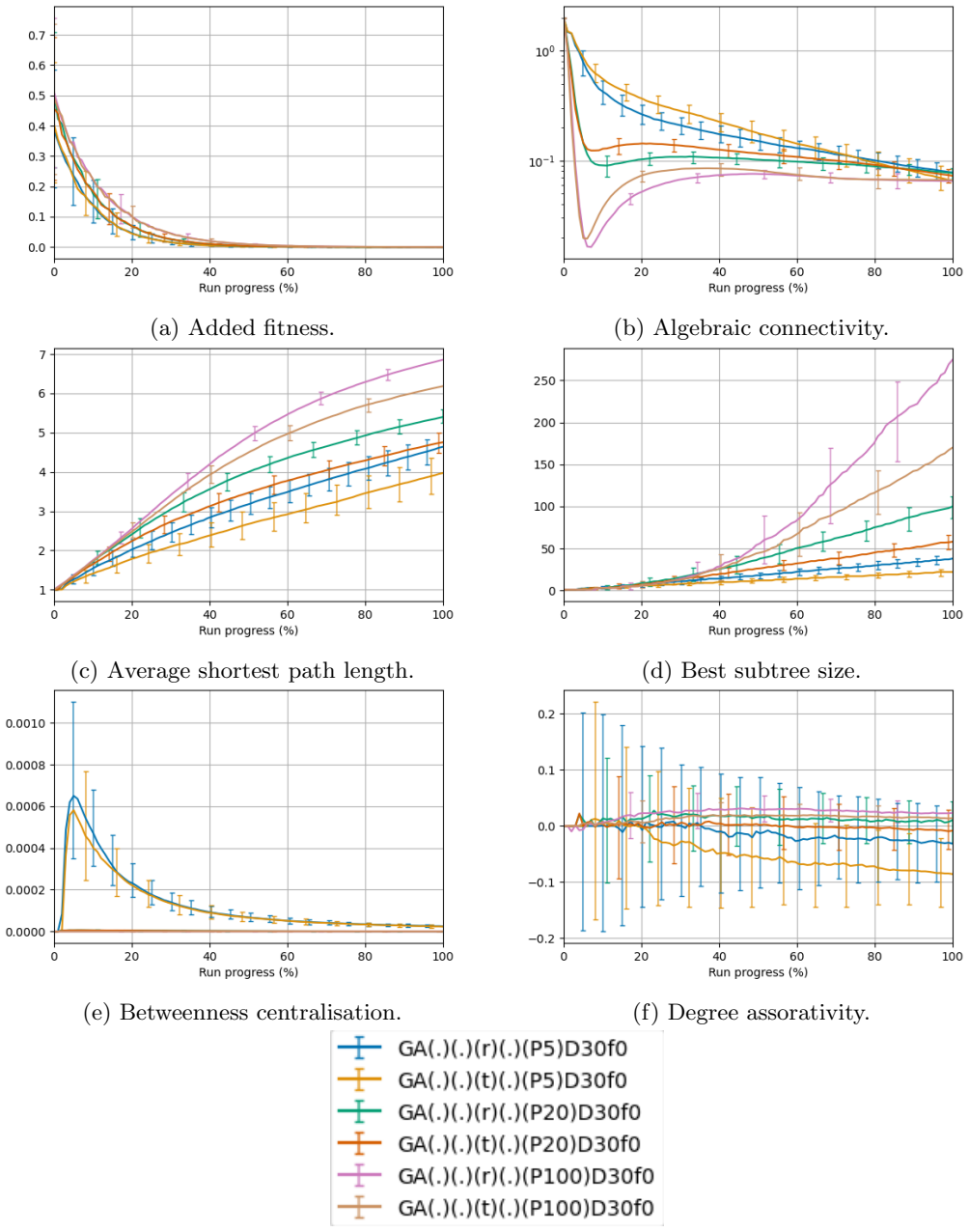


Figure C.1: Median time trajectories for different population sizes on random objective function f_0. Each line shows the median and spread of 1600 time trajectories from 32 variants, each comprising 50 runs.

C.2 Effect of Parent selection



(a) Added fitness.

(b) Algebraic connectivity.

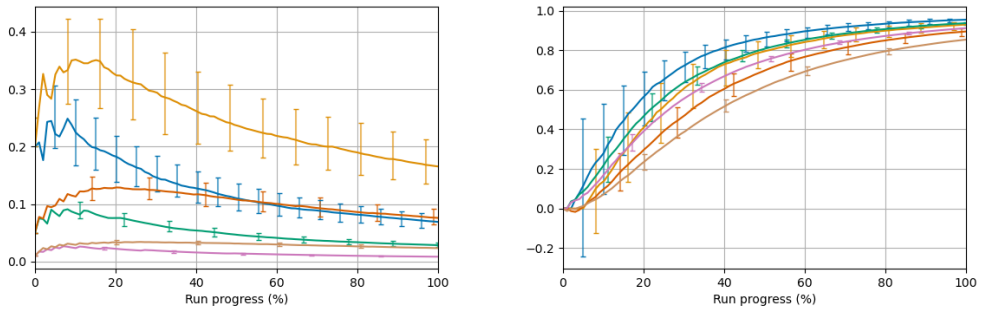
(c) Average shortest path length.

(d) Best subtree size.

(e) Betweenness centralisation.

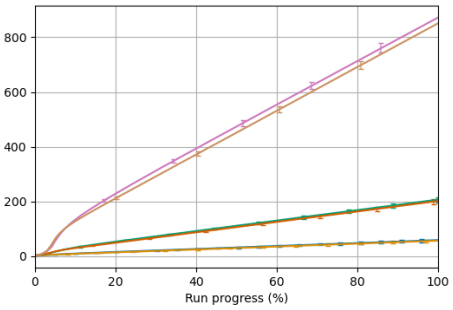
(f) Degree assorativity.

Figure C.2: Median time trajectories for different parent selection strategies on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

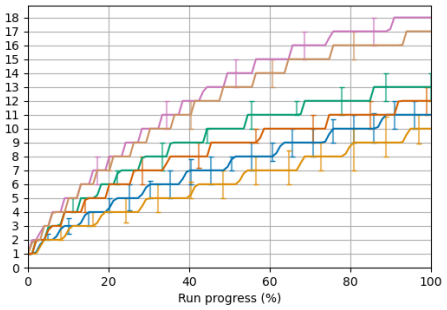


(g) Degree centralisation.

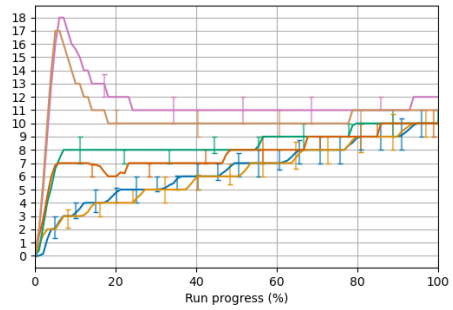
(h) Fitness assorativity.



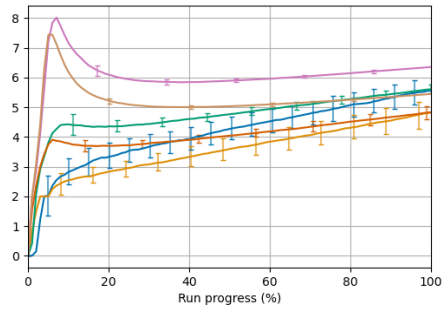
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

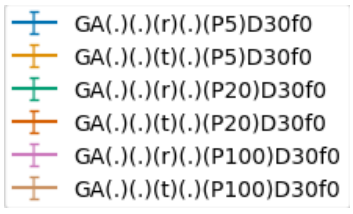
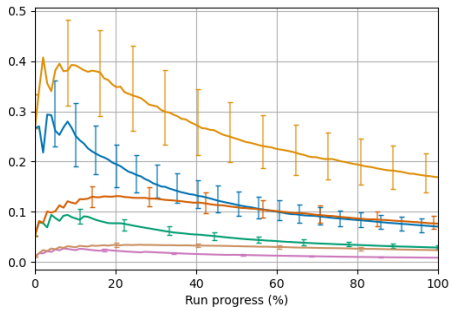
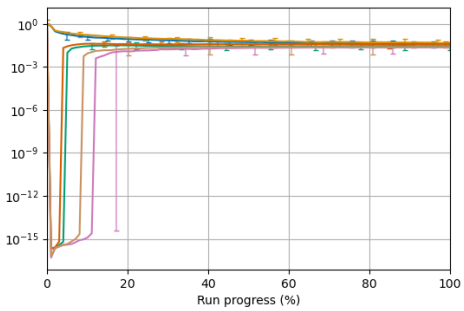


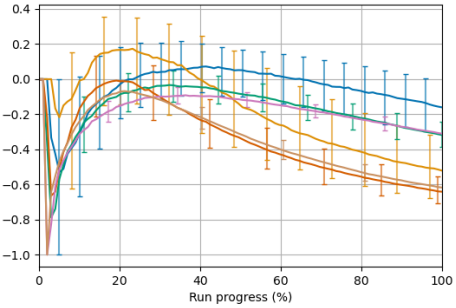
Figure C.2: Median time trajectories for different parent selection strategies on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.



(m) Out-degree centralisation.



(n) Spectral gap.



(o) Treeness.

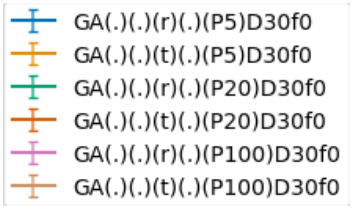
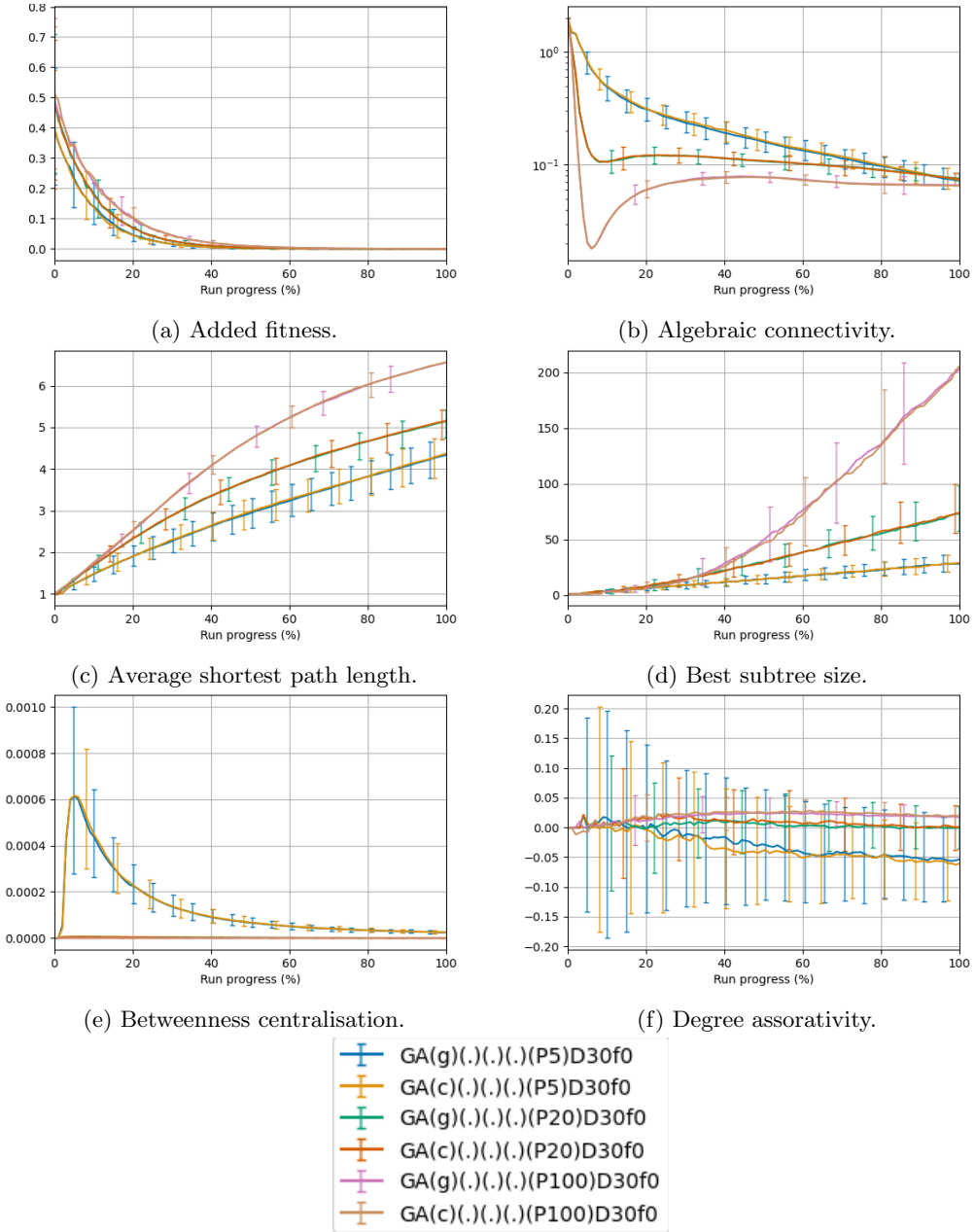


Figure C.2: Median time trajectories for different parent selection strategies on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

C.3 Effect of Mutation



(a) Added fitness.

(b) Algebraic connectivity.

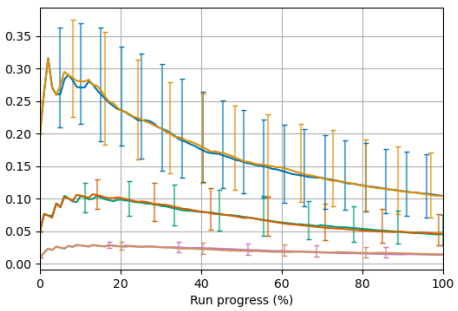
(c) Average shortest path length.

(d) Best subtree size.

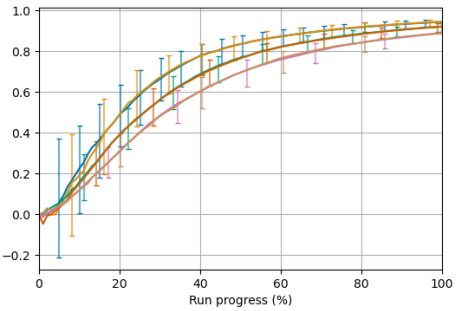
(e) Betweenness centralisation.

(f) Degree assorativity.

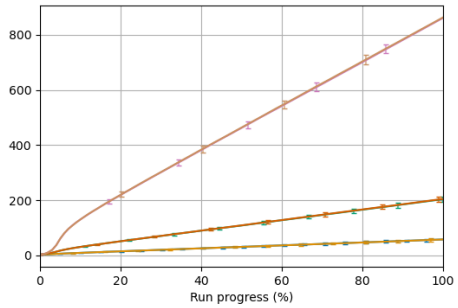
Figure C.3: Median time trajectories for different mutation operators on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.



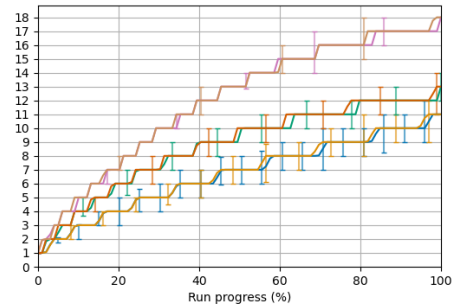
(g) Degree centralisation.



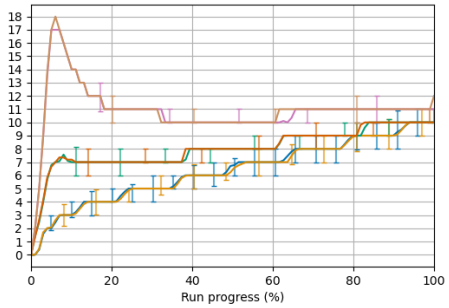
(h) Fitness assorativity.



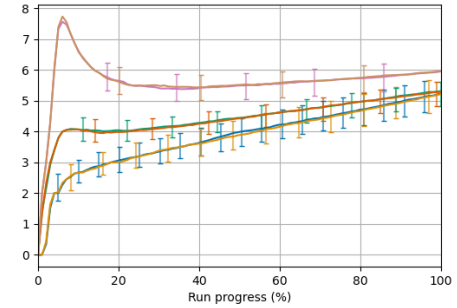
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

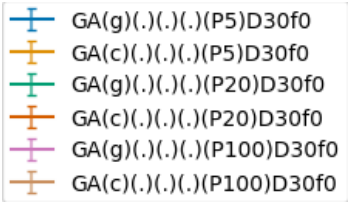
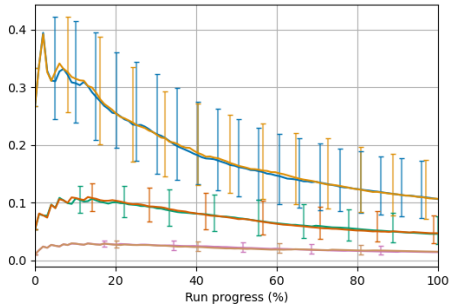
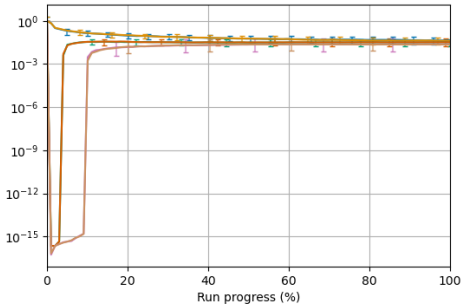


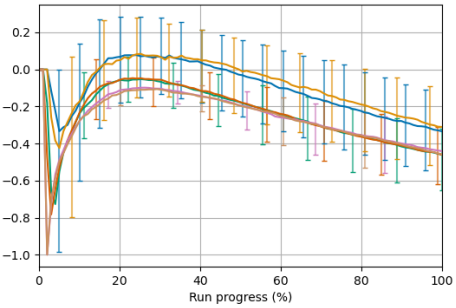
Figure C.3: Median time trajectories for different mutation operators on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

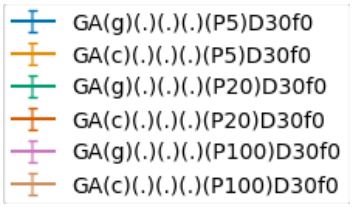


(m) Out-degree centralisation.



(n) Spectral gap.

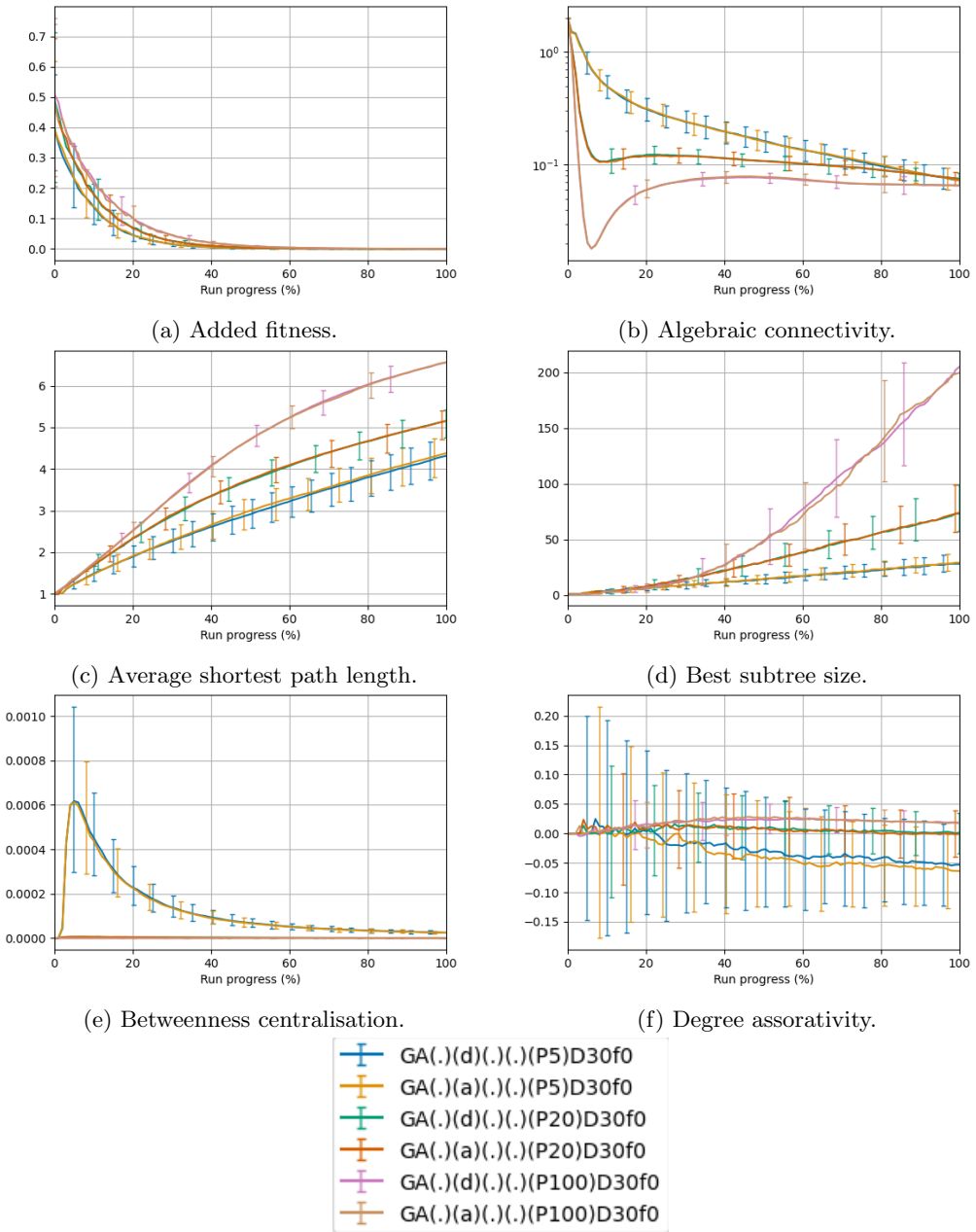




(o) Treeness.

Figure C.3: Median time trajectories for different mutation operators on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

C.4 Effect of Crossover



(a) Added fitness.

(b) Algebraic connectivity.

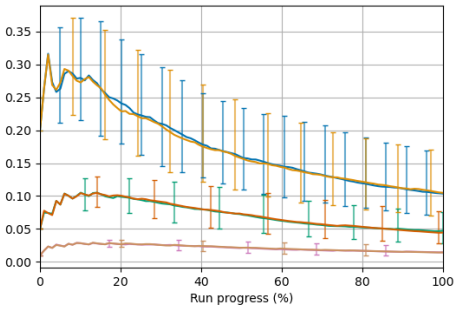
(c) Average shortest path length.

(d) Best subtree size.

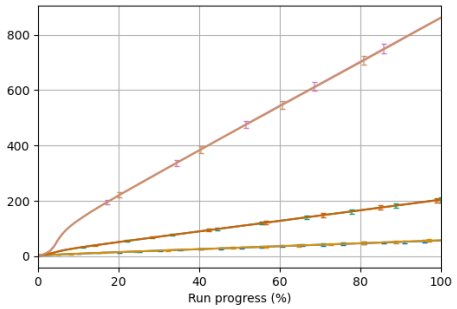
(e) Betweenness centralisation.

(f) Degree assorativity.

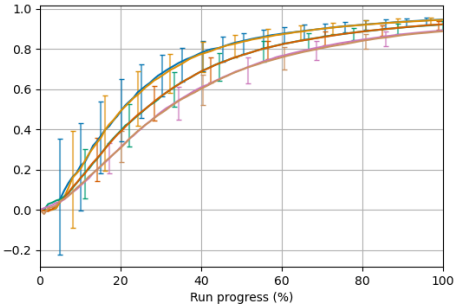
Figure C.4: Median time trajectories for different crossover operators on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.



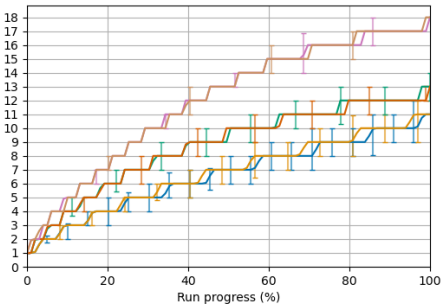
(g) Degree centralisation.



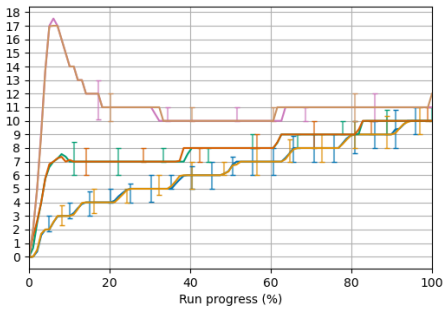
(h) Fitness assorativity.



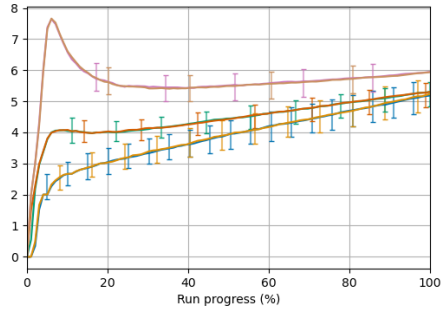
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

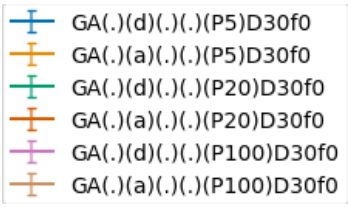
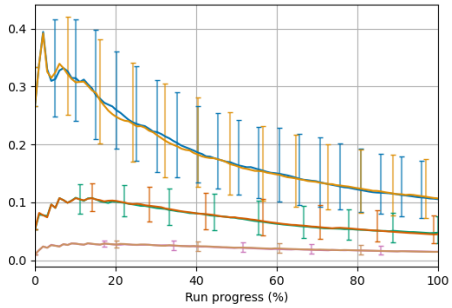
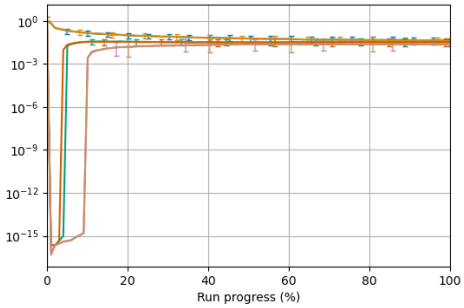


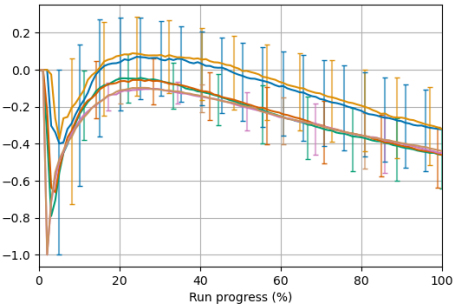
Figure C.4: Median time trajectories for different crossover operators on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

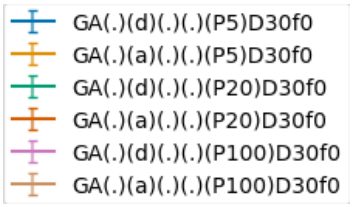


(m) Out-degree centralisation.



(n) Spectral gap.

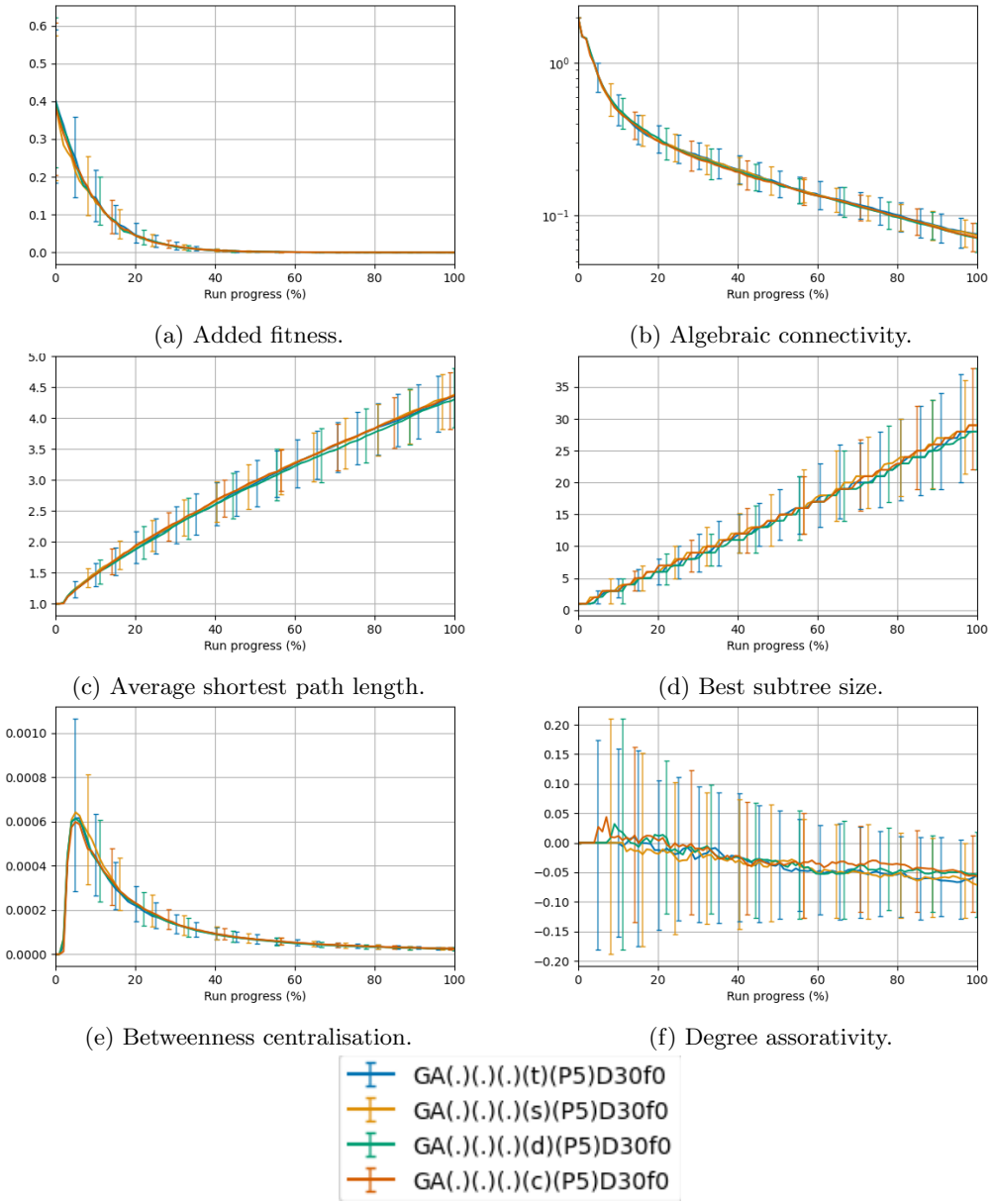




(o) Treeness.

Figure C.4: Median time trajectories for different crossover operators on random objective function f_0. Each line shows the median and spread of 800 time trajectories from 16 variants, each comprising 50 runs.

C.5 Effect of Correction (Population Size 5)



(a) Added fitness.

(b) Algebraic connectivity.

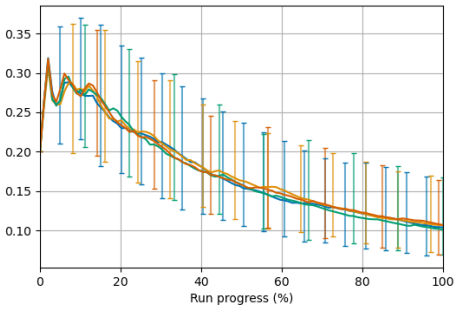
(c) Average shortest path length.

(d) Best subtree size.

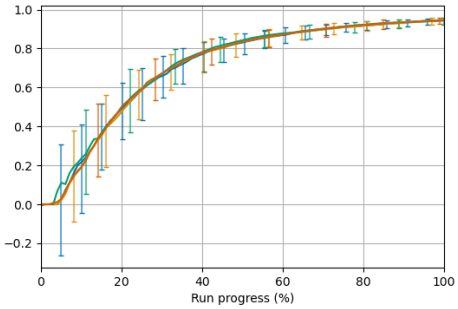
(e) Betweenness centralisation.

(f) Degree assortativity.

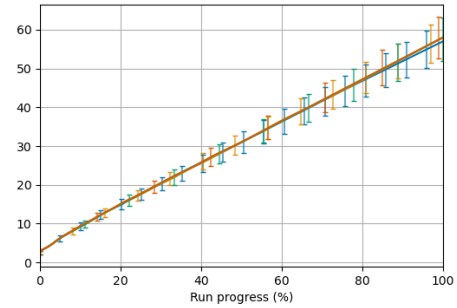
Figure C.5: Median time trajectories for different correction strategies and population size 5 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.



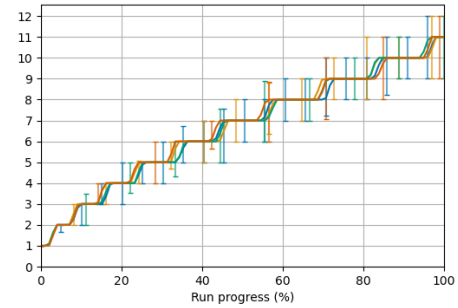
(g) Degree centralisation.



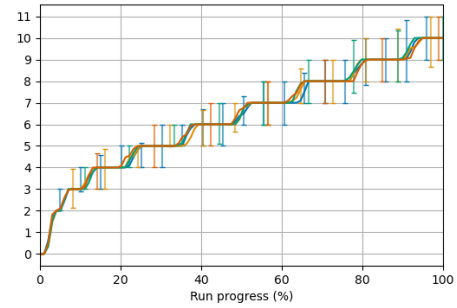
(h) Fitness assorativity.



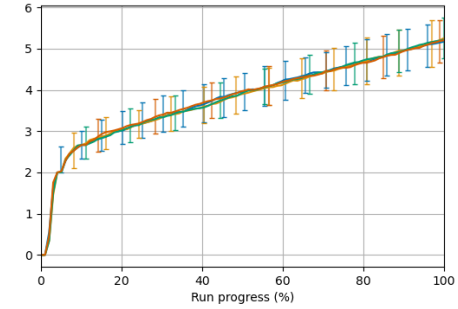
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

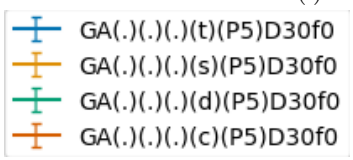
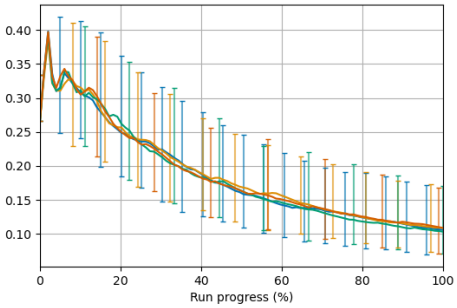
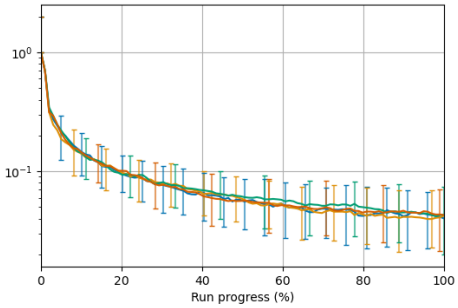


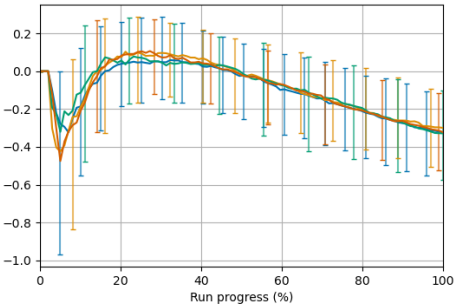
Figure C.5: Median time trajectories for different correction strategies and population size 5 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.



(m) Out-degree centralisation.



(n) Spectral gap.



(o) Treeness.

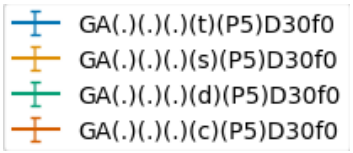
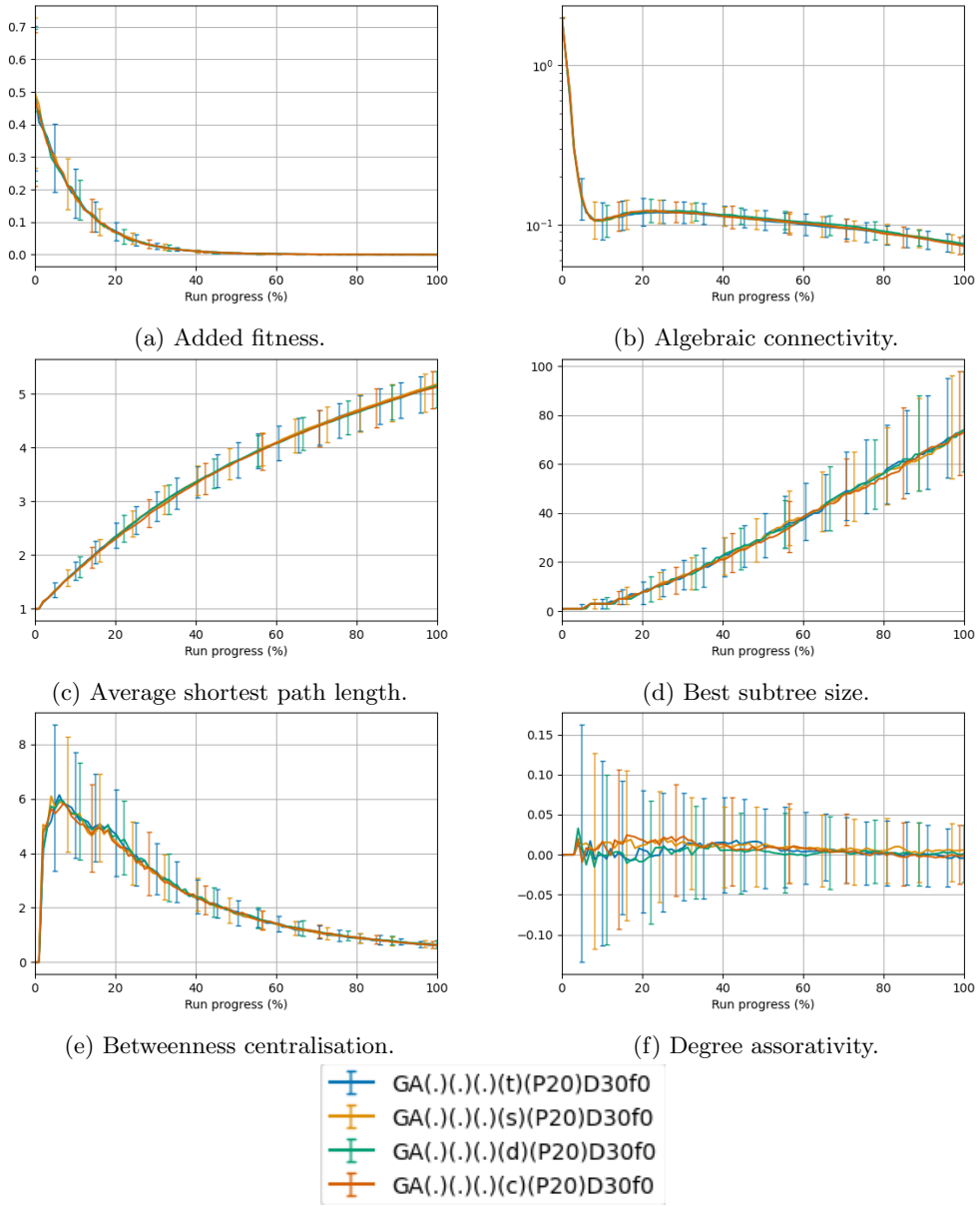


Figure C.5: Median time trajectories for different correction strategies and population size 5 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.

C.6 Effect of Correction (Population Size 20)



(a) Added fitness.

(b) Algebraic connectivity.

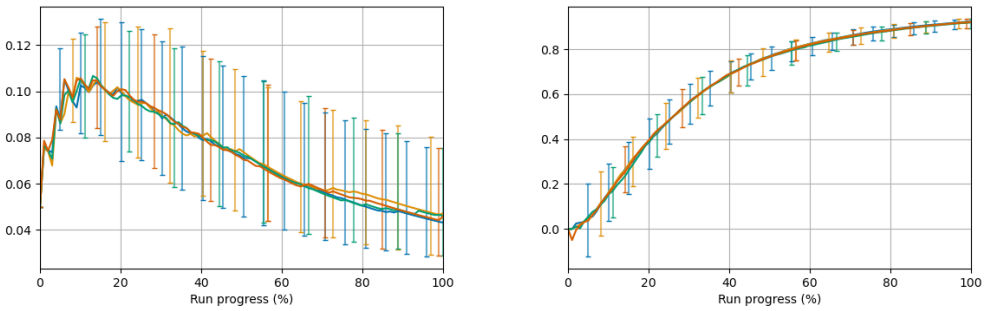
(c) Average shortest path length.

(d) Best subtree size.

(e) Betweenness centralisation.

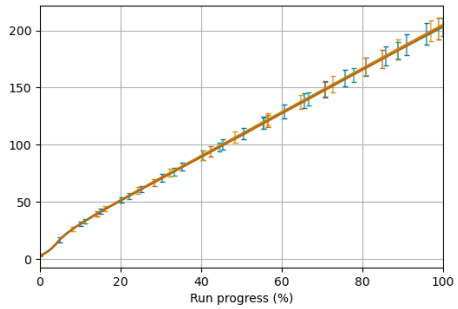
(f) Degree assortativity.

Figure C.6: Median time trajectories for different correction strategies and population size 20 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.

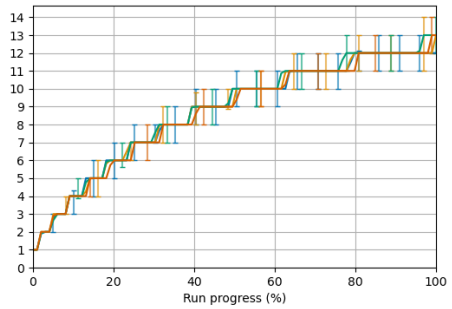


(g) Degree centralisation.

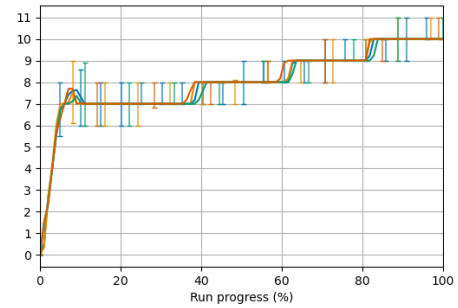
(h) Fitness assorativity.



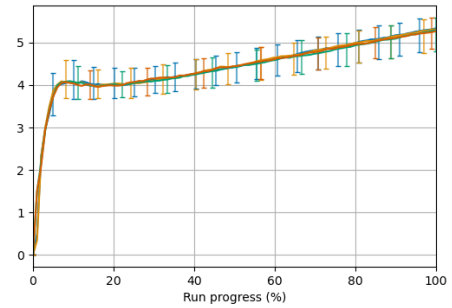
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

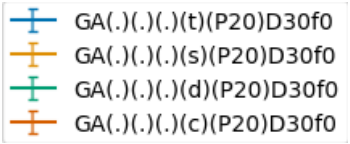
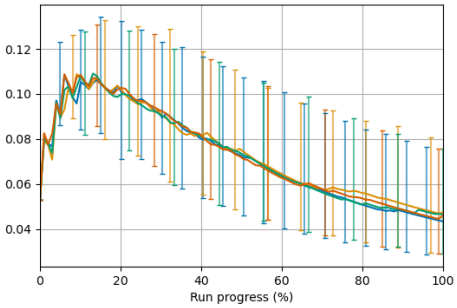
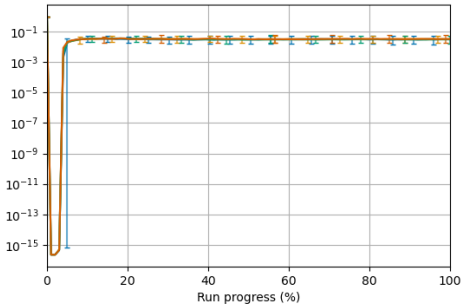


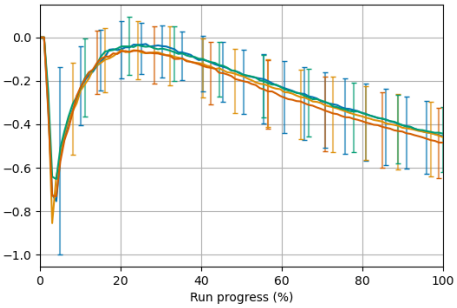
Figure C.6: Median time trajectories for different correction strategies and population size 20 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.



(m) Out-degree centralisation.



(n) Spectral gap.



(o) Treeness.

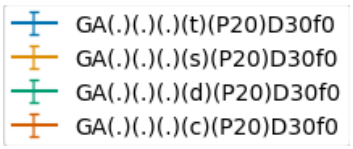
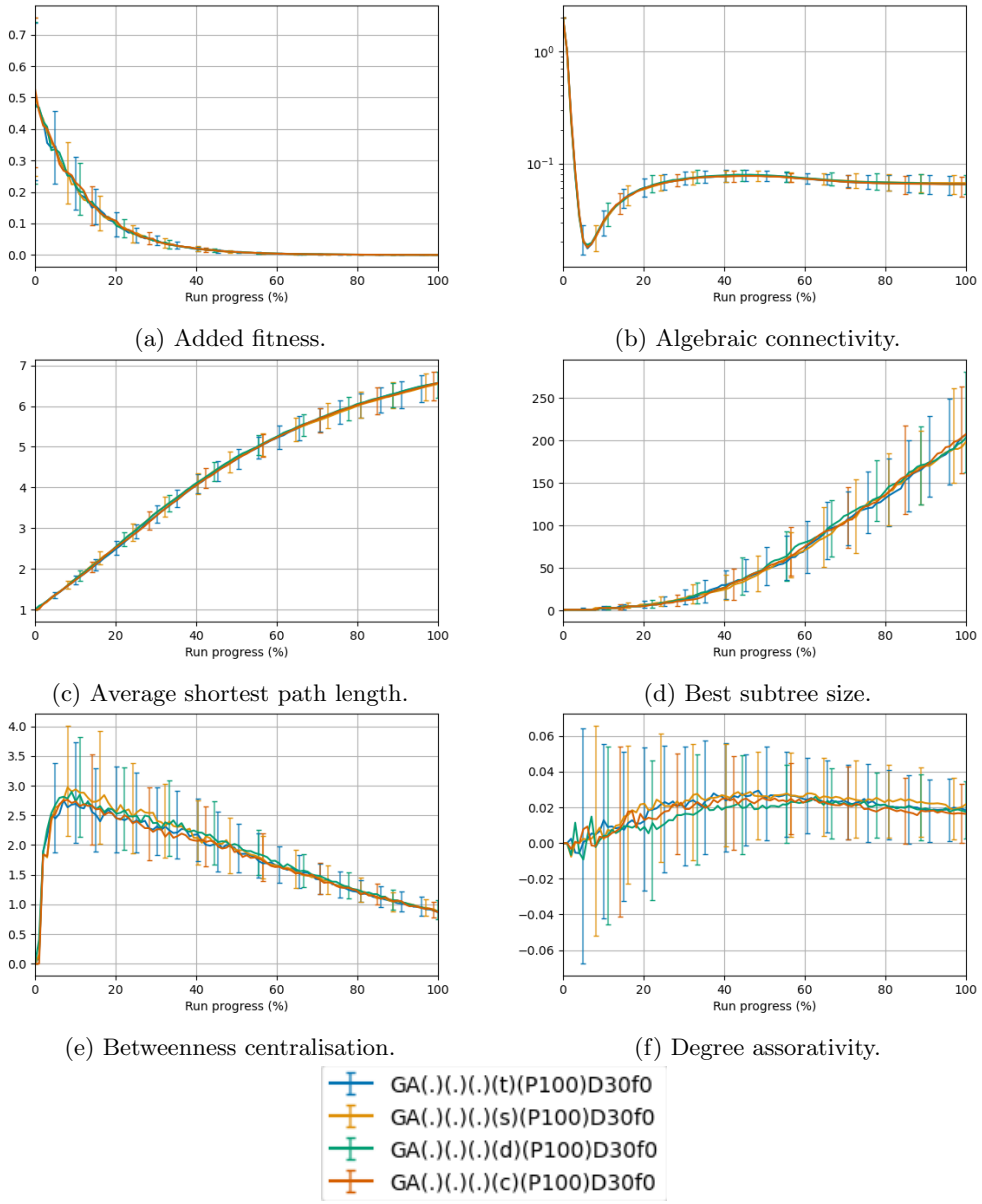


Figure C.6: Median time trajectories for different correction strategies and population size 20 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.

C.7 Effect of Correction (Population Size 100)



(a) Added fitness.

(b) Algebraic connectivity.

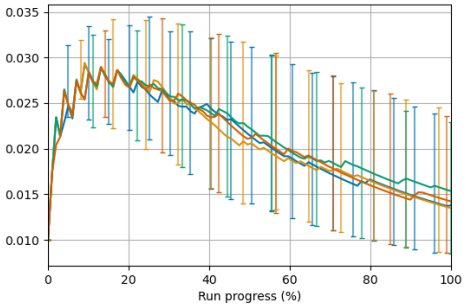
(c) Average shortest path length.

(d) Best subtree size.

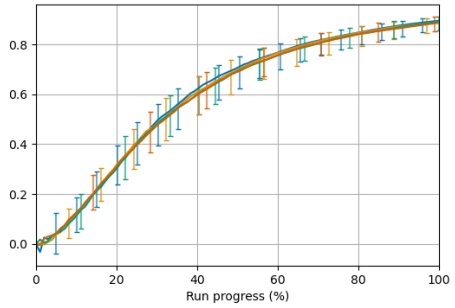
(e) Betweenness centralisation.

(f) Degree assorativity.

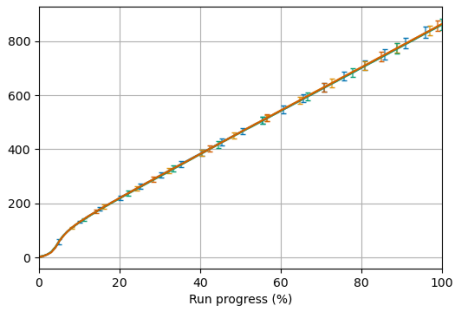
Figure C.7: Median time trajectories for different correction strategies and population size 100 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.



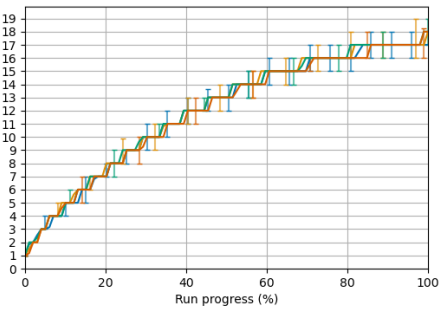
(g) Degree centralisation.



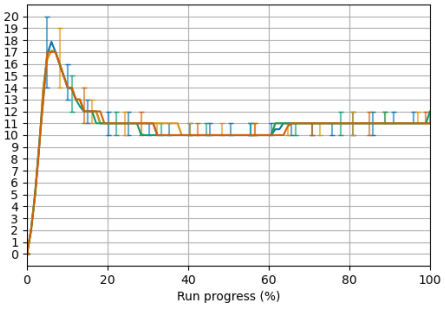
(h) Fitness assorativity.



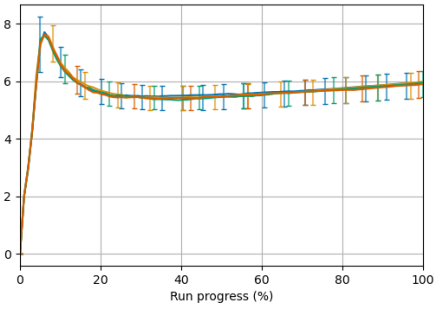
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

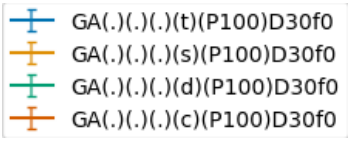
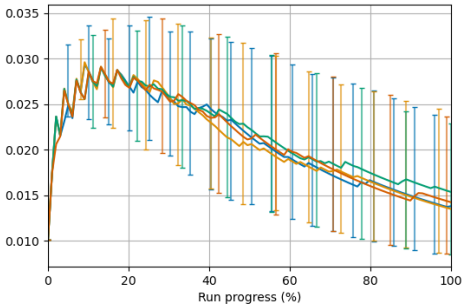
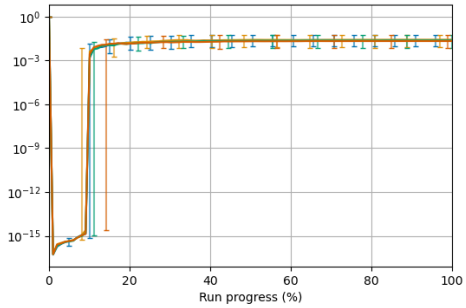


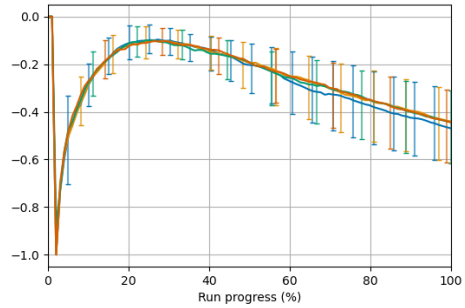
Figure C.7: Median time trajectories for different correction strategies and population size 100 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.



(m) Out-degree centralisation.



(n) Spectral gap.



(o) Treeness.

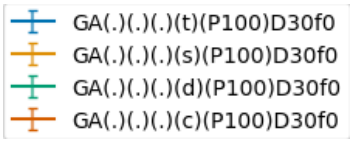
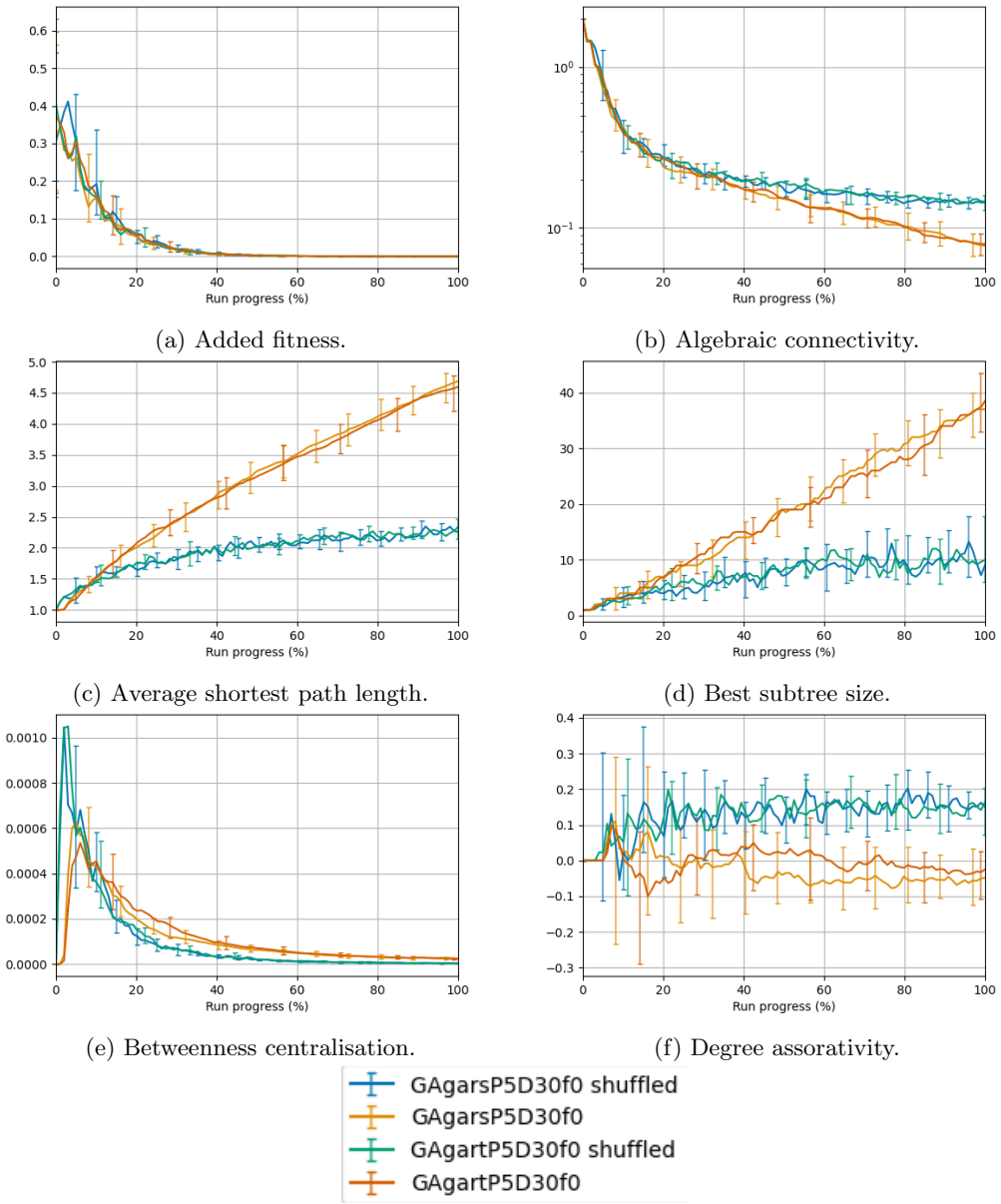


Figure C.7: Median time trajectories for different correction strategies and population size 100 on random objective function f_0. Each line shows the median and spread of 400 time trajectories from 8 variants, each comprising 50 runs.

Appendix D Shuffled networks



(a) Added fitness.

(b) Algebraic connectivity.

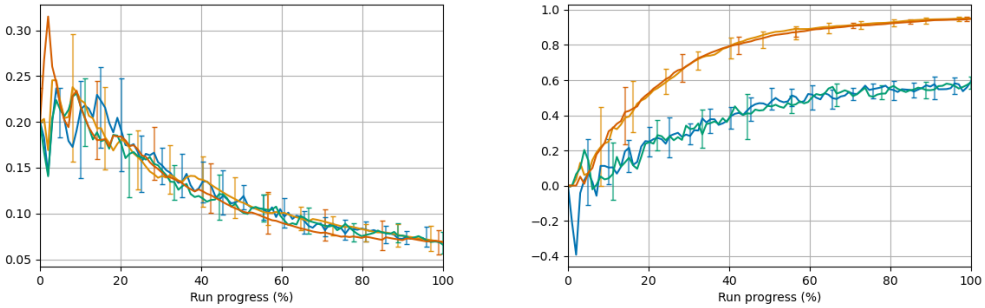
(c) Average shortest path length.

(d) Best subtree size.

(e) Betweenness centralisation.

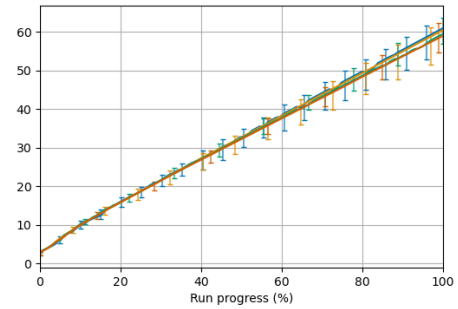
(f) Degree assorativity.

Figure D.1: Median time trajectories for original network representations and their shuffeld counterparts on random objective function f_0. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.

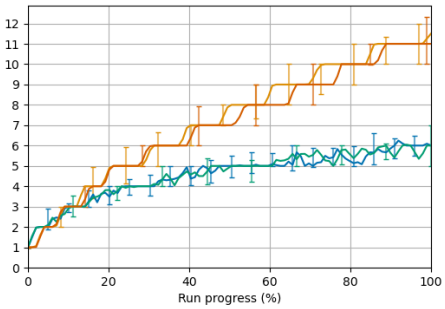


(g) Degree centralisation.

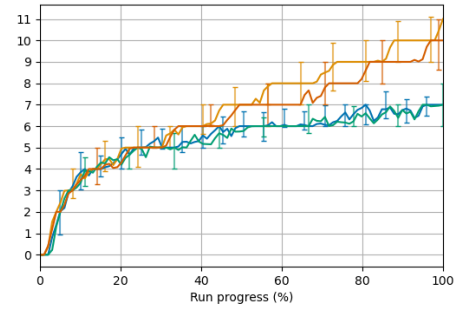
(h) Fitness assorativity.



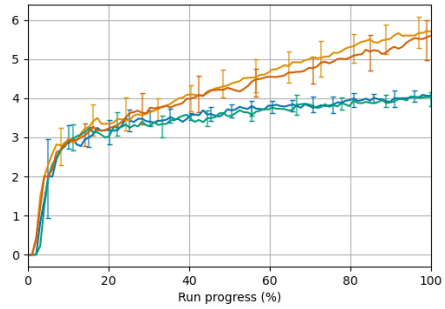
(i) Largest connected component size.



(j) Longest shortest path length.

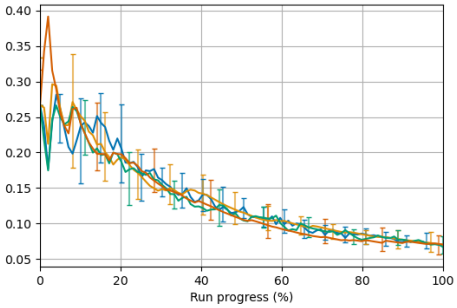


(k) Maximum leaf-leaf distance.

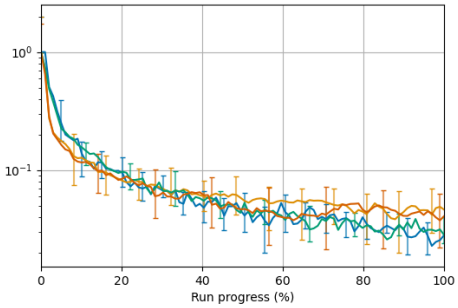


(l) Mean leaf-leaf distance.

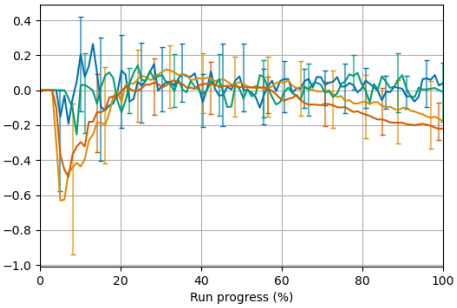
Figure D.1: Median time trajectories for original network representations and their shuffeld counterparts on random objective function f_0. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.



(m) Out-degree centralisation.



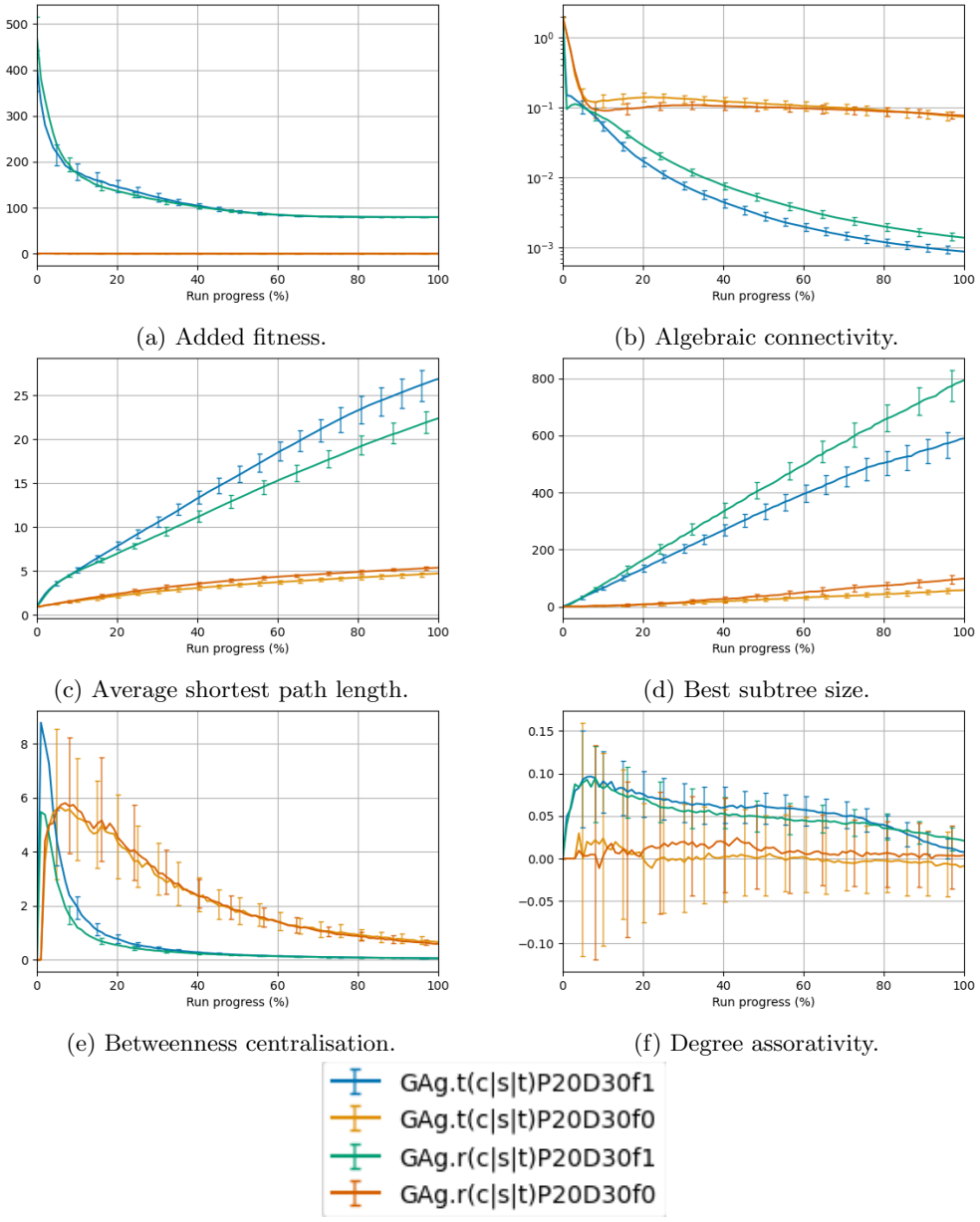
(n) Spectral gap.



(o) Treeness.

Figure D.1: Median time trajectories for oirigin network representations and their shuffled counterparts on random objective function f_0. Each line shows the median and spread of 30 time trajectories from 1 variant, comprising 30 runs.

Appendix E Plots for Sphere Function f_1



(a) Added fitness.

(b) Algebraic connectivity.

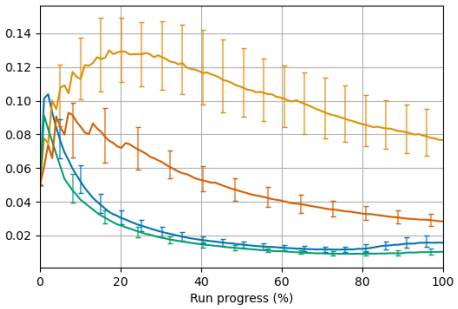
(c) Average shortest path length.

(d) Best subtree size.

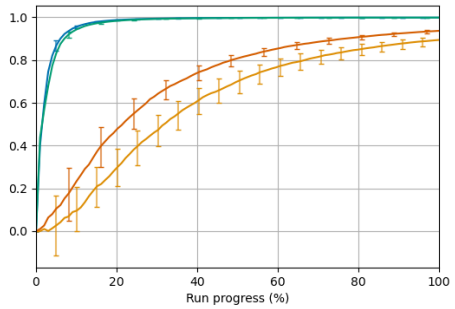
(e) Betweenness centralisation.

(f) Degree assorativity.

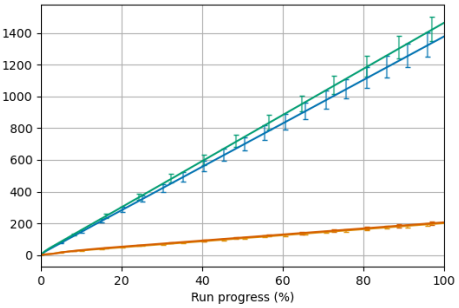
Figure E.1: Median time trajectories on objective functions f_0 and f_1. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs.



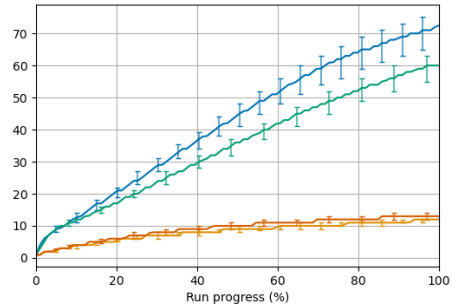
(g) Degree centralisation.



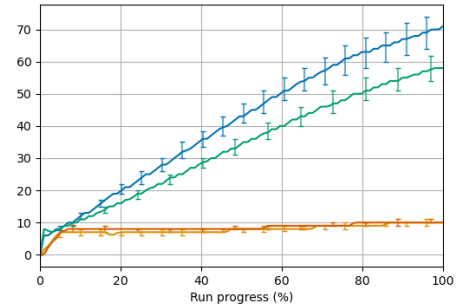
(h) Fitness assorativity.



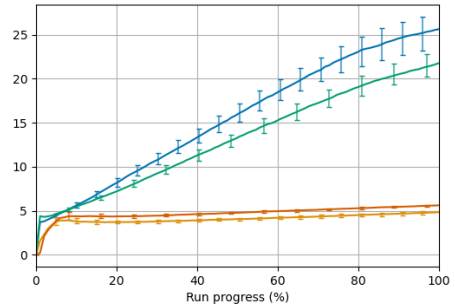
(i) Largest connected component size.



(j) Longest shortest path length.



(k) Maximum leaf-leaf distance.



(l) Mean leaf-leaf distance.

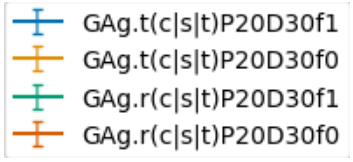
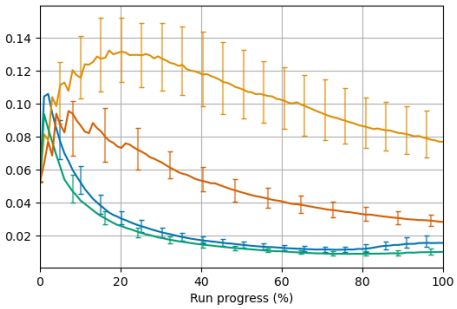
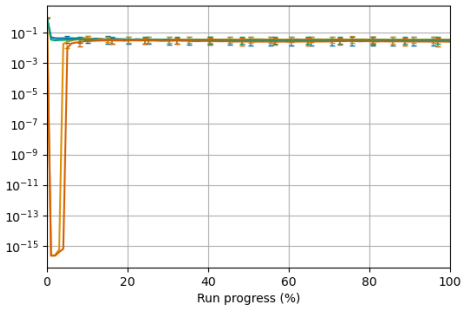


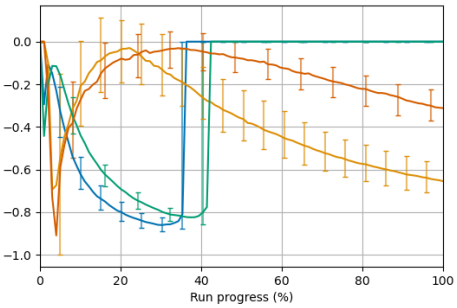
Figure E.1: Median time trajectories on objective functions f_0 and f_1. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs.

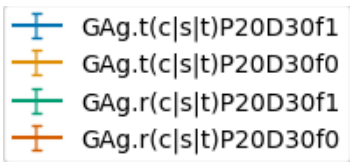


(m) Out-degree centralisation.



(n) Spectral gap.





(o) Treeness.

Figure E.1: Median time trajectories on objective functions f_0 and f_1. Each line shows the median and spread of 198 time trajectories from 6 variants, comprising 33 runs.

Appendix F Optimum Coordinates of f_1

The following coordinates were set using instance ID 1 for the sphere function in the BBOB suite [28].

$$x^{opt} = \begin{bmatrix} 0.2528 \\ -1.1568 \\ -0.7240 \\ 1.9264 \\ -2.6808 \\ 0.4392 \\ -0.1168 \\ 0.1200 \\ -1.6376 \\ -3.0512 \\ -3.8536 \\ 2.0160 \\ 1.0864 \\ -0.0864 \\ 3.6472 \\ 2.9088 \\ -0.1248 \\ -3.9280 \\ 2.3624 \\ 1.3584 \\ 0.3192 \\ 3.8304 \\ 1.1464 \\ -1.2536 \\ 0.8568 \\ -1.7824 \\ 1.9400 \\ 2.6976 \\ 2.0448 \\ 3.8568 \end{bmatrix}, f_{opt} = 79.48$$