## 0.1 Determine the Evolutionary Algorithm Configurations

### 0.1.1 Experimental Setup

As explained in **??**, there are many possible configurations for an evolutionary algorithm. Therefore, we test all possible combinations of operators. To avoid confusion, we refer to each unique phase combination as a configuration. For instance, one configuration would consist of **[a DatabBasedInitiator, an ElitismSelector, a OnePointCrosser, SamplinBasedMutator and a FittestSurvivorRecombiner]**. We refer to a specific configuration in terms of its abbreviated operators. For instance, the earlier example is denoted as **[DBI-ES-OPC-SBM-FSR]**.

The configuration set contains **[144]** elements. We choose to run each configuration for **[50]** evolution cycles. For all configurations, we use the same set of **[5]** factual process instances, which are randomly sampled from the test set. We decide to return a maximum of **[1000]** counterfactuals for each factual case. Within each evolutionary cycle, we generate **[100]** new offsprings. We keep the mutation rate at **[0.1]** for each mutation type. Hence, across all cases that are mutated, the algorithm inserts, deletes, and changes **[1%]** of events.

### 0.1.2 Results

Figure 1 shows the bottom and top **[k]** configurations based on the viability after the final iterative cycle. We also show how the viability evolves for each iteration.**[change evolutionary cycle to iterative cycle]** The results reveal a couple of patterns
shows for the **[average feasibility for each configuration]**. It does not surprise, that the **[FactualInitiator]** remains at a low feasibility as deviations will often lead to infeasible counterfactuals. The evolutionary algorithm remains at a local optimum without exploring other solutions. Furthermore, we see that most configurations reach at most **[0.04]** feasibility, while the initialisation with the **[DataDistributionSample]** initiator reaches higher values.

In terms of viability Table 1 the overall mean between all configurations is **[2.49]**. We see that the factual initiator is clearly superior. However, the results for using the factual itself are nearly identical. In terms of the selection operators, we see a reduction of viability using either the **[RouletteWheelSelector or TournamentSelector]**. Meaning, **[the ElitismSelector]** yields better results when it comes to viability. The **[DefaultMu-**

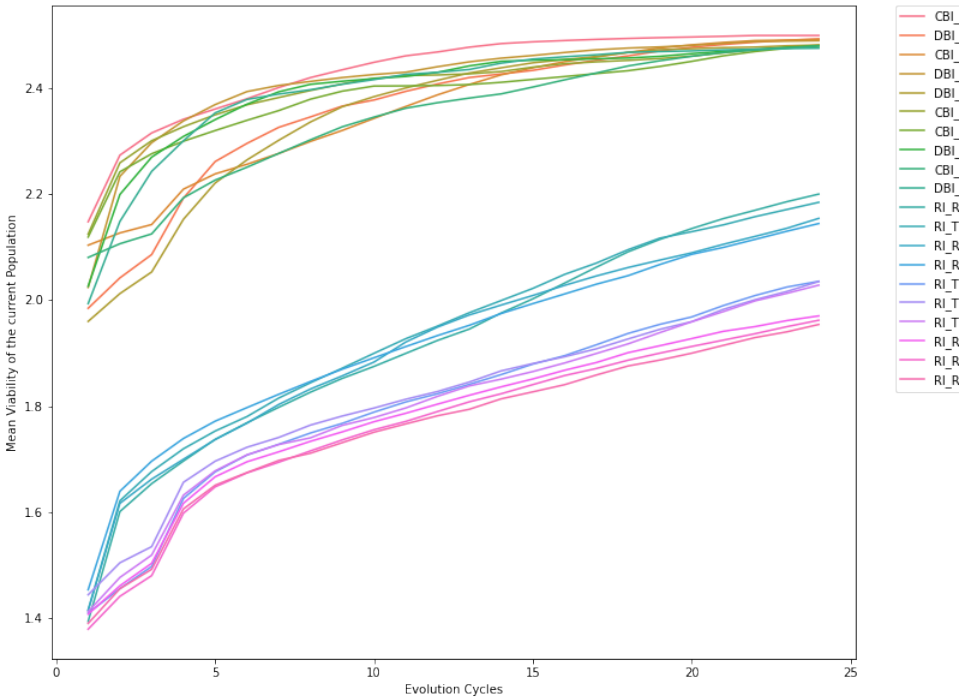


Figure 1: This figure shows the average viability of each configuration within the top or bottom 10 in terms of viability. The x-axis shows how the viability evolves for each evolutionary cycle.

Table 1: Table shows the result of the linear mixed model. It uses viability as dependent variable and evolutionary operators as independent categorical variable. The model is adjusted for general differences in combinations. The coefficient explains the effect of the model, while P explains the statistical significance. Only significant values are considerable effects.

| | Coef. | Std.Err. | p-value |
|---|---|---|---|
| Intercept | 2.490 | 0.017 | 0.000 |
| initiator[T.DataDistributionSampleInitiator] | -0.025 | 0.015 | 0.097 |
| initiator[T.DefaultInitiator] | -0.242 | 0.015 | 0.000 |
| initiator[T.FactualInitiator] | 0.494 | 0.015 | 0.000 |
| selector[T.RouletteWheelSelector] | -0.080 | 0.013 | 0.000 |
| selector[T.TournamentSelector] | -0.048 | 0.013 | 0.000 |
| mutator[T.DefaultMutator] | -0.024 | 0.011 | 0.024 |
| recombiner[T.FittestIndividualRecombiner] | 0.040 | 0.011 | 0.000 |
| crosser[T.TwoPointCrosser] | 0.003 | 0.013 | 0.833 |
| crosser[T.UniformCrosser] | -0.027 | 0.013 | 0.039 |
| Configuration Set Var | 0.004 | 0.006 | |

**tator]** also reduces the viability. In other words, we achieve better results with the **[DataDistributionMutator]**. The best recombination approach is the **[FittestIndividualRecombiner]**. If we evaluate the crossing methods, the **[TwoPointCrosser]** slightly edges out the **[OnePointCrosser]**. However, this effect is not significant. However, we see, that the **[UniformCrosser]** reduces viability and it is significant. The effect of each configuration set is comparable with the effect of using the **[TwoPointCrosser]**. Hence, the composition of configurations plays a comparatively minor role.

Now, we consider the mixed-effects linear model on feasibility and present the results in Table 2. When it comes to the initiator, we see a strong contrast when it comes to the effects on feaibility opposed to viability. Here, the **[DataDistributionSampleInitiator]** seems to improve the feasibility over other itnitiators. When it comes to the selectors, **[RouletteWheelSelector and TournamentSelector]** remain negative, although, slightly less significant. We also see that the effect of the **[UniformCrosser]** is significantly detrimental to the feasibility for both, the viability and the feasibility. Although, the **[TwoPointCrosser]** appears to be better than the **[OnePointCrosser]**, it seems, the model remain undecisive, due to the high p-value. The configuration set does not appear to have any influece on the viability if we judge the group coefficient.

Table 2: Table shows the result of the linear mixed model. It uses feasibility as dependent variable and evolutionary operators as independent categorical variable. The model is adjusted for general differences in combinations. The coefficient explains the effect of the model, while P explains the statistical significance. Only significant values are considerable effects.

| | Coef. | Std.Err. | p-value |
|---|---|---|---|
| Intercept | 0.023 | 0.005 | 0.000 |
| initiator[T.DataDistributionSampleInitiator] | 0.085 | 0.005 | 0.000 |
| initiator[T.DefaultInitiator] | -0.011 | 0.005 | 0.028 |
| initiator[T.FactualInitiator] | -0.010 | 0.005 | 0.033 |
| selector[T.RouletteWheelSelector] | -0.016 | 0.004 | 0.000 |
| selector[T.TournamentSelector] | -0.008 | 0.004 | 0.068 |
| mutator[T.DefaultMutator] | -0.002 | 0.003 | 0.563 |
| recombiner[T.FittestIndividualRecombiner] | 0.005 | 0.003 | 0.157 |
| crosser[T.TwoPointCrosser] | -0.001 | 0.004 | 0.830 |
| crosser[T.UniformCrosser] | -0.017 | 0.004 | 0.000 |
| Configuration Set Var | 0.000 | 0.001 | |

### 0.1.3 Discussion

The reasons for thisthe superiority of **[FactualInitiator]** are clear. If we start the model with the factuals as initial population, the factual will already have a viability of at least 2 as similiarity and sparcity have to be at their maximal value. As the prediction model tends to only assign scores close to the extremes, the favorable change of an event attribute often yields a strong bias which is often correct. Hence, the viabilities often reach a viability of around 3. The only way to reach a higher viability for factually inititiated counterfactuals is to approach the pareto-surface by increasing the feasibility. In other words, one would have to increase feasibility without significantly decreasing the scores for similarity, sparcity and the improvement. Similarly, it is no surprise, that the **[FactualInitiator]** has a negative effect on the feasibility, as it is difficult to find a case which is even more likely than a case that was directly sampled from the log.

Moving forward, we have to choose a set of configurations and also determine suitable hyperparameters for each. In the next experiment we consider a couple of configurations. We choose the **[DataDistributionSampleInitiator]** as it might increase our chances to generate feasible variables. Furthermore, we include the **[FactualInitiator]**, as it would be interesting whether we can reach better results, by changing parameters. For selection, we will use the **[ElitismSelector and RouletteWheelSelector]**. The former be-

cause it seems to be consistently better than the other selectors. The latter because, we suspect that the negative effect is highly bised by the results of the **[FactualInitiator]**. When it comes to the crossing operation, the results indicate, the difference in effect-sizes between **[OnePointCrosser]** and **[TwoPointCrosser]** are marginal and inconclusive. One can explain that by noting, that both operations are very similar in nature. Hence, we choose to move forward with the **[TwoPointCrosser]**, as it appears to yield better viability results on average. We do not consider the **[UniformCrosser]** as it has a negative effect on viability and feasibility. For mutation and recombination, we choose **[DataDistributionMutator]** and **[FittestIndividualRecombiner]**, respectively. Both consistenly outperformed their alternatives.

In the next experiment we vary the other parameters.