0.1 Determine the Evolutionary Algorithm Configurations

As explained in **??**, there are many possible configurations for an evolutionary algorithm. Therefore, we run simulations for all possible combinations of operators. We use the results, to filter out operators for subsequent evaluation steps that may not be promising.

0.1.1 Experimental Setup

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. First, all of the top-**[k]** algorithms use **[either CaseBasedInitiator or SampleBasedInitiator]** as initiation operation. In contrast, the bottom-**[k]** all use **[RandomInitiator]** as initialisation. Second, we see that most of the top-**[k]** algorithms use the **[ElitismSelector]**.

The complete table of results is in **??**.

ATTACHMENT: tbl:average-viability

In Figure 2, we see the effects of each operator type, except the mutation operation.

Starting with some commonalities accross operator-type and measure, the figure shows that the initiator heavily determines the starting point for

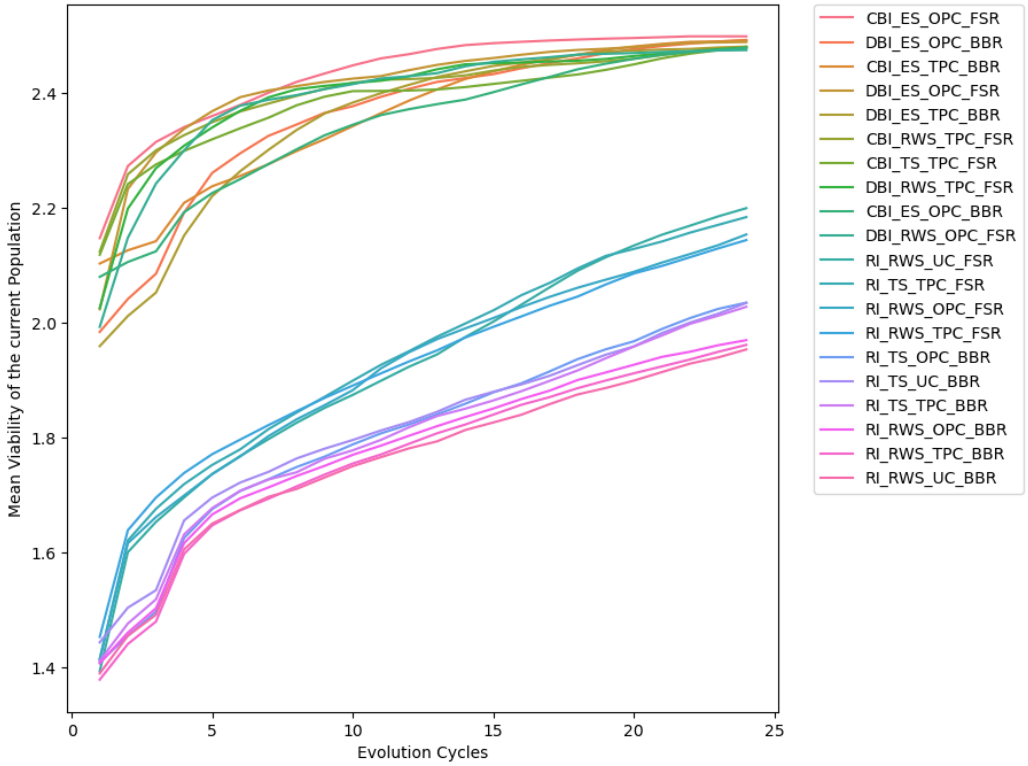

Figure 1: This figure shows the average viability of the [10] best and worst configurations. The x-axis shows how the viability evolves for each evolutionary cycle.

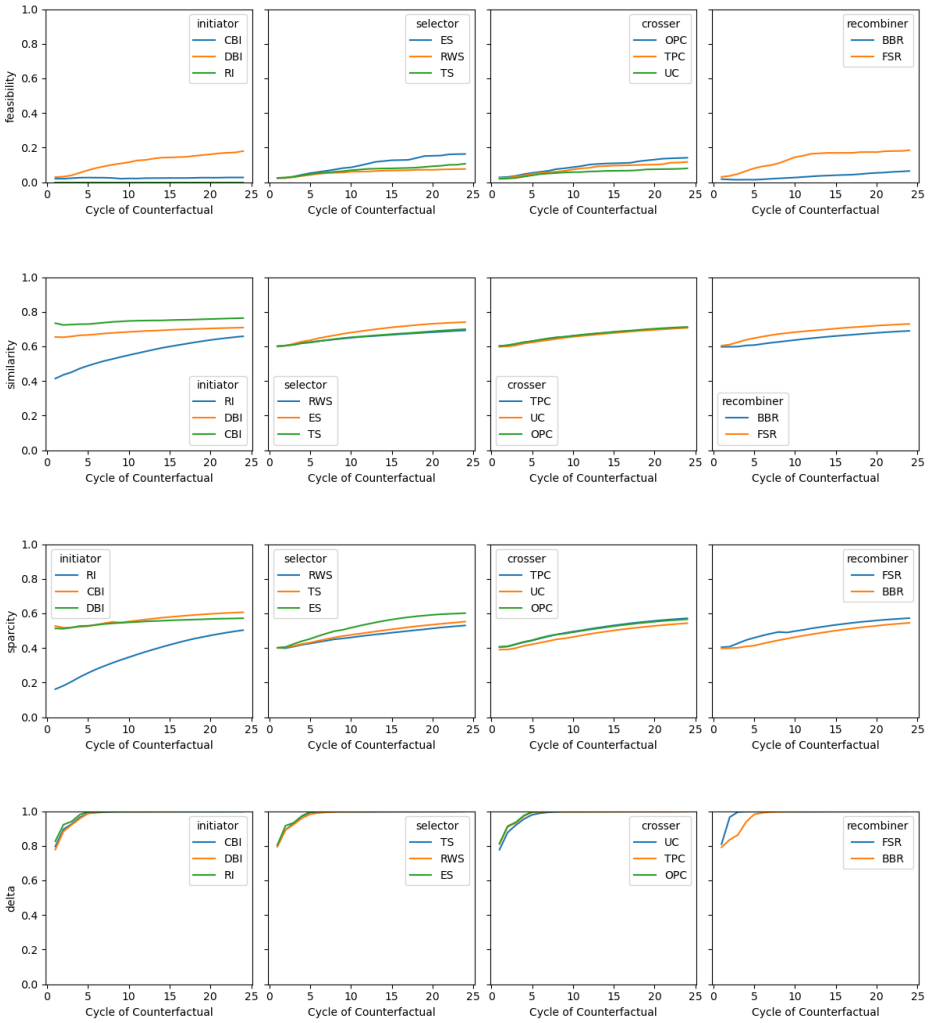

Figure 2: Shows the evolution of each viability measure over the entire span of iterative cycles. Each figure adjust the respective operator type by taking the average over all other operator types. **[Make sure, that the legends are aligned in color.]**

each measure. For instance, the [RandomInitiator] starts well below the other initiator operations when it comes to sparcity and similarity. Another noteworthy general observation is the delta measure[**Change some of the names to align with Dice4EL**]. Here, for each operator type we see a movement towards the highest possible delta value. Hence, most configurations are capable of changing the source class to the desired class.

In terms of viability Figure **??**, shows an increase only for cases, in which the [SampleBasedInitiator] is used. Similar holds for recombination with [FittestSurvivorRecombiner].

The results for the selection operator type are undeniably in favor of [ElitismSelector] for all viability measures. The same holds for the recombination operation. Here, the [FittestSurvivorRecombiner] yields better results.

When it comes to the crossing operation, the results indicate, the differences between [OnePointCrosser] and [TwoPointCrosser] are inconclusive for all viability measures except feasibility. One can explain that by noting, that both operations are very similar in nature. However, cutting the sequence only once retains produces less impossible sequences for the child sequences.

0.1.3 Discussion

Moving forward, we have to choose a set of configurations and also determine suitable hyperparameters for each. In the next experiment we consider the following configurations: We choose the [SampleBasedInitiator] as it might increase our chances to generate feasible variables.

For selection, we will use the [ElitismSelector], as it generally appears to return better results.

Furthermore, we choose to move forward with the [OnePointCrosser]. This crossing operation is slightly better in yielding feasible results.

For selection and recombination, we use the [ElitismSelector] and the [FittestIndividualRecombiner], respectively. They all outcompete their alternatives for all similarity, sparcity and feasibility.

In the next experiment we vary mutation rates, using [SBI-ES-OPC-SBM-FIR].