CpE5310 Computational Intelligence

Evolutionary Algorithm Sampling

Prepared by: Nicholas Covert, William Lorey, and Ruicheng Xu

Prepared for: Dr. Steven Corns

December 13, 2018

Contents

Abstract

In this report, the authors explore evolutionary algorithms by implementing programs utilizing evolutionary algorithms to solve Light Up puzzles. Light Up puzzles are a popular pencil-and-paper puzzle that originated from Japan. Played on grid cells, each player has to fill in empty cells following a set of rules, which can be found here: `www.puzzle-light-up.com` along with online, playable puzzles. Throughout the experiment, authors compared a variety of EA setups to solve the puzzles, including random search, standard EA, constraint satisfaction EA, and multi-objective EA (MOEA).

Light Up puzzles are part of the NP-Complete problem space. They are hard to solve using brute force approaches as the search space is so huge that it is impossible to explore all possible solutions in reasonable time. The performance and behaviors of different evolutionary algorithms were also observed and analyzed. This experimentation was completed in part as an assignment series for Dr. Daniel Tauritz's Evolutionary Computing class.

Software Discussion

Introduction

As part of this project, software was written in Python 3 to implement, run, and analyze evolutionary algorithms that solve Light Up puzzles. For each of the four algorithm types explored (random search, standard EA, constraint satisfaction EA, and MOEA), a separate code base was maintained, with all code bases containing shared elements.

Note that the source code for this project was not included as part of this report document for concision, but can be viewed on GitHub by using the following link:

`https://github.com/wwlorey/ea-sampler/`

In the git repository there are four folders, each named in accordance with the algorithmic approach it contains. The following is a brief overview of how the general EA driver module was structured, information about EA configuration, and instructions on running the algorithms.

EA Driver Module

The heart of every EA implemented as part of this project was the EA driver module. Implemented for each EA type at the path `ea/ea_driver.py`, this module acted as the primary driver for all high-level evolutionary algorithm functioning for the algorithms and was slightly adjusted depending on the individual algorithm requirements.

Each EA driver contained functions to drive the main program logic, including, for example, the functions `evaluate, select_parents, recombine, mutate, select_for_survival,` and `decide_termination` among other helper functions. These functions interfaced with a lower-level driver, which contained the individual phenotype which in this case was the Light Up puzzle 2D board. This phenotype was acted upon by genotype classes containing sets of bulbs. These genotypes were mapped to a singular phenotype for each experiment run and individuals in the population were treated as the bulb sets, with the aforementioned EA functions acting on the genotype classes.

While all aspects of the EA driver and other related modules are not individually discussed for brevity, the code is thoroughly documented if one desires to peruse it.

Configuration Parameters

For easy configuration of EA experiments and for the sake of the programmers, configuration files were loaded with settings and fed into the algorithms to setup all configurable values. These files were stored in the `config` directory of each EA folder and the structure of the config files changed slightly from algorithm to algorithm.

An example of the configuration parameters of note available in the case of the standard EA are as follows:

- mu (population size)
- lambda (offspring pool size)
- num_experiment_runs
- num_fitness_evaluations
- n_termination_convergence_criterion
- termination_convergence_criterion_magnitude

- parent_population_size
- use_fitness_proportional_selection
- k_parent_selection
- use_truncation
- k_survival_selection
- n_point_crossover
- parent_selection_weight
- mutation_probability
- input_file_path
- log_file_path
- soln_file_path
- force_validity
- use_external_seed
- external_seed_value
- generate_uniform_random_puzzle
- black_square_placement_prob
- bulb_placement_prob
- min_random_board_dimension
- max_random_board_dimension
- override_random_board_dimensions
- override_num_rows
- override_num_cols

Choosing an optional configuration file to load is discussed in the following section.

Execution Instructions

A `run.sh` script has been included in the root of each EA directory on the git repository. If one wishes to run a particular EA, all they must do is run that script in the directory of the EA of their choice. Running this script without any command line arguments will load the default configuration (`default.cfg`), found in the `config` folder of each EA directory.

For example, the following command would run the EA in the folder containing it with a default configuration:

```
./run.sh
```

The following command would run the EA in the folder containing it with the configuration `website_puzzle.cfg`:

```
./run.sh config/website_puzzle.cfg
```

Random Search

Introduction

The Purpose of the Random Search Algorithm is to serve as a baseline to compare all of the other Evolutionary Algorithms performance and serve as an introduction to the actual puzzle itself. This Algorithm does not actually try to solve the puzzle, it just places down light bulbs on the map that gets generated for it. The random placement was made to satisfy the bulb shining constraint, but it pays no attention to the black cell constraint. The total number of bulbs placed is also random, sometimes a map will only have two or three bulbs placed on it, and sometimes a lot of bulbs are placed on the map.

Discussion

Unsurprisingly when the random search solutions are placed into the fitness function it performs the worst by far. These results show that the light up puzzle is one where using computational intelligence is very useful to solve this problem. While this problem is not a practical one, it is not to hard to imagine real world problems can lead to the same situation where using some form of computational intelligence can be very helpful, if not almost necessary to solve the problem.

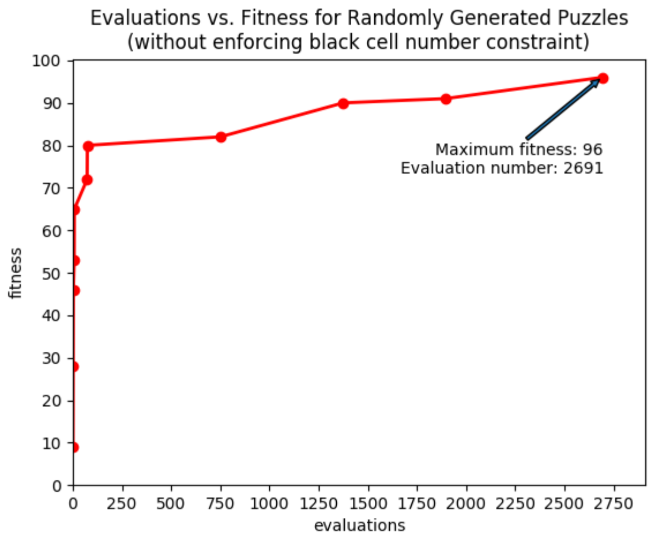

Figure 1: Evaluations vs. Fitness for Randomly Generated Puzzles (without enforcing black cell number constraint)

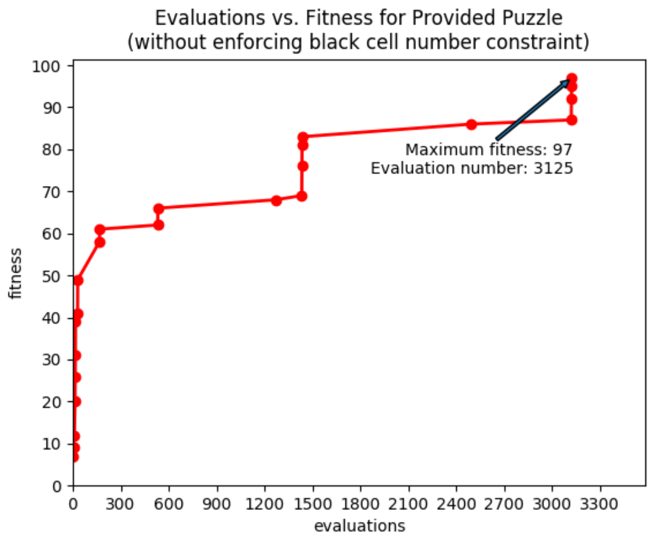

Figure 2: Evaluations vs. Fitness for Provided Puzzle (without enforcing black cell number constraint)

Standard EA

Introduction

The standard EA was a step up when compared to the random search algorithm. Is started off with the same randomized initial population as the random search. In addition, the standard EA utilized evolutionary processes, including fitness evaluations, parent selection, recombination of solutions, and survival selection. This evolutionary approach makes the algorithm perform a stochastic hill climb to find optimal solutions.

The fitness function for this EA reflects the quality of the solution considering the following aspects: number of cells being lit, whether the black cell adjacency constraint is satisfied, and whether any light bulb is shining on any other light bulb. Due to the coarse gradient of the fitness function, the EA was expected to perform poorly, thus the authors implemented another fitness evaluation function such that the black cell adjacency constraint was being ignored. This greatly reduced the difficulty of the puzzle and using this approach, and as such, improvements from the random search were observed.

Discussion

For both puzzle varieties (the provided puzzle and the generated puzzle) the evolutionary algorithm was compared to the random search algorithm to determine statistical superiority between the methods, in the context of this problem instance. The numerical analysis for each of these experiment classes can be found in the Statistical Analysis for Provided Puzzle and Statistical Analysis for Randomly Generated Puzzle sections.

In each case, since the distribution was not known to be normal and since the sample size was greater than 29, the variances were compared using the f-test. The f-test for both experiment comparisons yielded the conclusion that variances could be assumed equal. This was because the conditions

```
|mean(Random Search) < mean(EA)| and F > F Critical
```

both held true. Finally, the two-tailed two-sample t-test assuming equal variances was performed for both experiment comparisons. The following condition held true for both tests:

```
|t Stat| > |t Critical Two - Tail|
```

This condition's validity meant that the null hypothesis was rejected, meaning that the variable with the larger mean indicated a better algorithm (statistically) on this particular problem instance. In both cases, the standard evolutionary algorithm had a larger mean when compared to that of the random search. Because of this, the evolutionary algorithm is proven to be superior to the random search for this problem instance, in both the case of the provided puzzle and the randomly generated puzzle.

Table 1: F-Test Two-Sample for Variances for Provided Puzzle Analysis

	Random Search	EA
Mean	0.6966216216	0.9923423423
Variance	0.0009928906073	0.00004638842011
Observations	30	30
df	29	29
F	21.40384615	
P(F $\leq$ f) one-tail	0	
F Critical one-tail	1.860811435	

Table 2: t-Test Two-Sample Assuming Equal Variances for Provided Puzzle Analysis

	Random Search	EA
Mean	0.6966216216	0.9923423423
Variance	0.0009928906073	0.00004638842011
Observations	30	30
Pooled Variance	0.0005196395137	
Hypothesized Mean Difference	0	
df	58	
t Stat	-50.24308544	
P(T $\leq$ t) one-tail	0	
t Critical one-tail	1.671552762	
P(T $\leq$ t) two-tail	0	
t Critical two-tail	2.001717484	

Table 3: Standard Deviation for Provided Puzzle Analysis

Standard Deviation	sqrt(0.0009928906073)	0.03151016672

Table 4: F-Test Two-Sample for Variances for Randomly Generated Puzzle Analysis

	Random Search	EA
Mean	0.9038096235	0.9936490682
Variance	0.001202739143	0.00008180893878
Observations	30	30
df	29	29
F	14.70180595	
P(F $\leq$ f) one-tail	0	
F Critical one-tail	1.860811435	

Table 5: t-Test Two-Sample Assuming Equal Variances for Randomly Generated Puzzle Analysis

	Random Search	EA
Mean	0.9038096235	0.9936490682
Variance	0.001202739143	0.00008180893878
Observations	30	30
Pooled Variance	0.000642274041	
Hypothesized Mean Difference	0	
df	58	
t Stat	-13.7294299	
P(T $\leq$ t) one-tail	0	
t Critical one-tail	1.671552762	
P(T $\leq$ t) two-tail	0	
t Critical two-tail	2.001717484	

Table 6: Standard Deviation for Randomly Generated Puzzle Analysis

Standard Deviation	sqrt(0.001202739143)	0.03468052973

Table 7, Table 8, and Table 9 are the result of statistical analysis performed on the evolutionary algorithm run on a provided puzzle with one version initialized in a uniform random approach and the other initialized with validity enforcement also using a uniform random approach. The fitness distribution was not known to be normal and the sample size was larger than 29 so the variances were tested for equality using the f-test. Unequal variances were assumed due to the following conditions' validity.

```
|mean(Uniform Random) < mean(Validity Enforced Uniform Random)| and F < F
                                Critical
```

The two-tailed two-sample t-test assuming unequal variances was then performed. The following condition did not hold true for the test.

```
|t Stat| > |t Critical Two - Tail|
```

For this reason, the null hypothesis could not be rejected and the variable with the better mean did not indicate a statistically better algorithm (for this problem instance). Instead, both algorithms were comparable in producing fit solutions.

Table 7: Additional F-Test Two-Sample for Variances for Provided Puzzle Analysis

	Random Search	EA
Mean	0.9923423423	0.993018018
Variance	0.00004638842011	0.00004245275098
Observations	30	30
df	29	29
F	1.092707046	
P(F $\leq$ f) one-tail	0.4064710353	
F Critical one-tail	1.860811435	

Table 8: t-Test Two-Sample Assuming Unequal Variances for Provided Puzzle Analysis

	Random Search	EA
Mean	0.9923423423	0.993018018
Variance	0.00004638842011	0.00004245275098
Observations	30	30
Pooled Variance	0	
Hypothesized Mean Difference	58	
df	-0.3926374989	
t Stat	0.3480132753	
P(T $\leq$ t) one-tail	1.671552762	
t Critical one-tail	0.6960265505	
P(T $\leq$ t) two-tail	2.001717484	
t Critical two-tail	2.001717484	

Table 9: Standard Deviation for Provided Puzzle Analysis

Standard Deviation	sqrt(0.00004638842011)	0.0068109045

Table 10, Table 11, and Table 12 are the result of statistical analysis performed on the evolutionary algorithm run on a randomly generated puzzle with one version initialized in a uniform random approach and the other initialized with validity enforcement also using a uniform random approach. The fitness distribution was not known to be normal and the sample size was larger than 29 so the variances were tested for equality using the f-test. Unequal variances were assumed due to the following conditions' validity.

```
|mean(Uniform Random) > mean(Validity Enforced Uniform Random)| and F > F
                                Critical
```

The two-tailed two-sample t-test assuming unequal variances was then performed. The following condition did not hold true for the test.

```
|t Stat| > |t Critical Two - Tail|
```

For this reason, the null hypothesis could not be rejected and the variable with the better mean did not indicate a statistically better algorithm (for this problem instance). Instead, both algorithms were comparable in producing fit solutions.

Table 10: F-Test Two-Sample for Variances for Randomly Generated Puzzle Analysis

	Random Search	EA
Mean	0.9936490682	0.9904708897
Variance	0.00008180893878	0.000106774622
Observations	30	30
df	29	29
F	0.7661833615	
P(F ≤ f) one-tail	0.238872549	
F Critical one-tail	0.5373999648	

Table 11: t-Test Two-Sample Assuming Unequal Variances for Randomly Generated Puzzle Analysis

	Random Search	EA
Mean	0.9936490682	0.9904708897
Variance	0.00008180893878	0.000106774622
Observations	30	30
Pooled Variance	0	
Hypothesized Mean Difference	57	
df	1.267613926	
t Stat	0.1050449431	
P(T ≤ t) one-tail	1.672028888	
t Critical one-tail	0.2100898862	
P(T ≤ t) two-tail	2.002465459	
t Critical two-tail	2.001717484	

Table 12: Standard Deviation for Randomly Generated Puzzle Analysis

Standard Deviation	sqrt(0.00008180893878)	0.00904482939

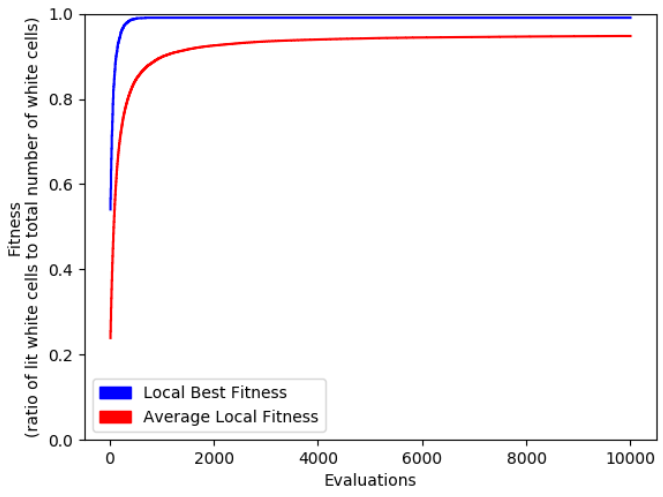

Figure 3: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for Uniform Random Initialized, Randomly Generated Puzzle, Averaged Over All Runs

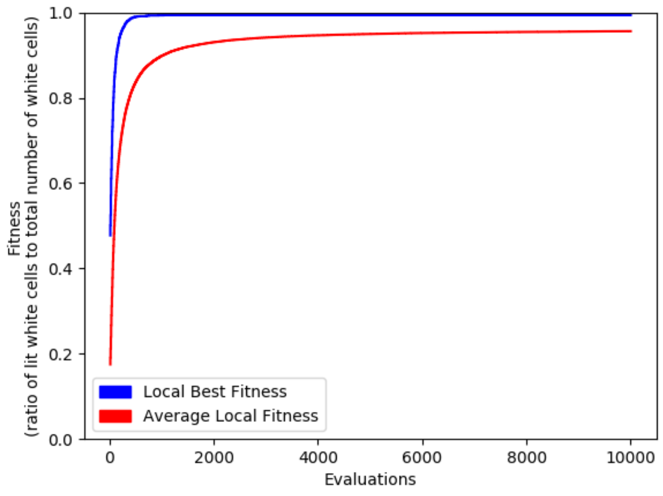

Figure 4: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for Validity Enforced Initialized, Randomly Generated Puzzle, Averaged Over All Runs

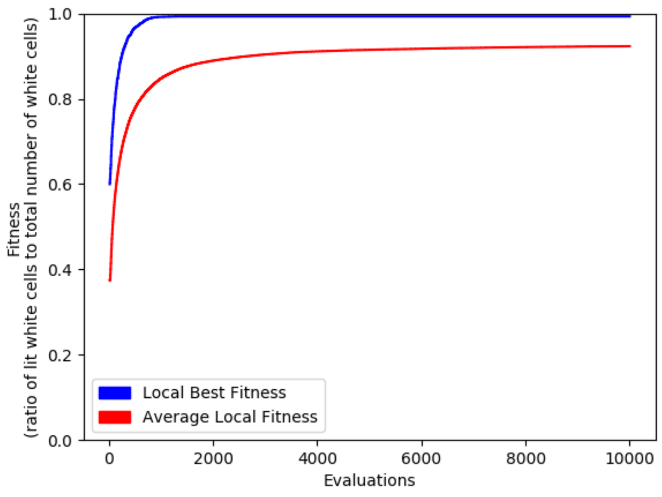

Figure 5: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for Uniform Random Initialized, Provided Puzzle, Averaged Over All Runs

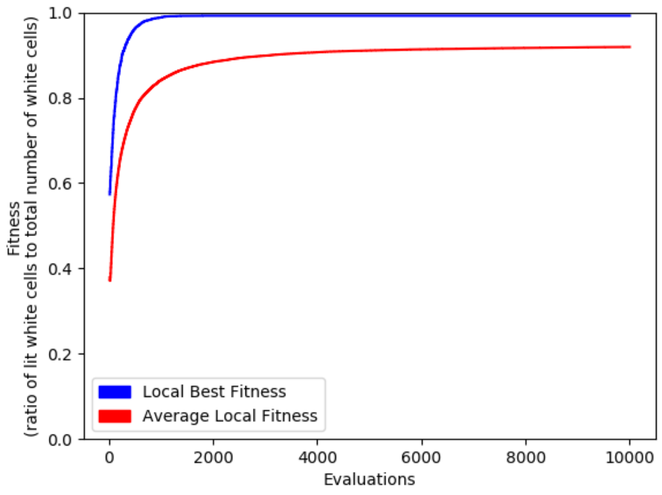

Figure 6: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for Validity Enforced Initialized, Provided Puzzle, Averaged Over All Runs

Constraint Satisfaction EA

Introduction

This section involved implementing an EA leveraging constraint satisfaction to more effectively solve Light Up puzzles. The performance of penalty function and repair function constraint satisfaction techniques as well as plain-vanilla EA performance are compared as part of this section. It also outlines a comparison between Validity Forced plus Uniform Random versus plain Uniform Random initialization for a plain-vanilla EA, a constraint satisfaction EA employing a penalty function, and a constraint satisfaction EA employing a repair function. The extent to which EA performance is affected by the penalty coefficient used in penalty function constraint satisfaction EAs will also be examined.

Constraint Satisfaction EA Performance Comparisons

For a baseline, an EA leveraging a penalty function was implemented. The penalty function subtracted from a given genotype's fitness the number of constraints violated multiplied by a penalty coefficient. Violated constraints involved bulbs shining on other bulbs and black cell constraints not being met.

This EA was compared to an EA implemented using the repair function constraint satisfaction technique. Figure 7 and Figure 8 show evaluations versus fitness for the penalty function EA and the repair function EA respectively tested against the Light Up puzzle provided on the course website. Figure 9 and Figure 10 show evaluations versus fitness for the aforementioned EAs tested against randomly generated puzzles. Note that for situations where the best fitness for a solution plateaus, there is possibility of encountering a decreasing average fitness as higher mutation factors are introduced to combat a stagnant population.

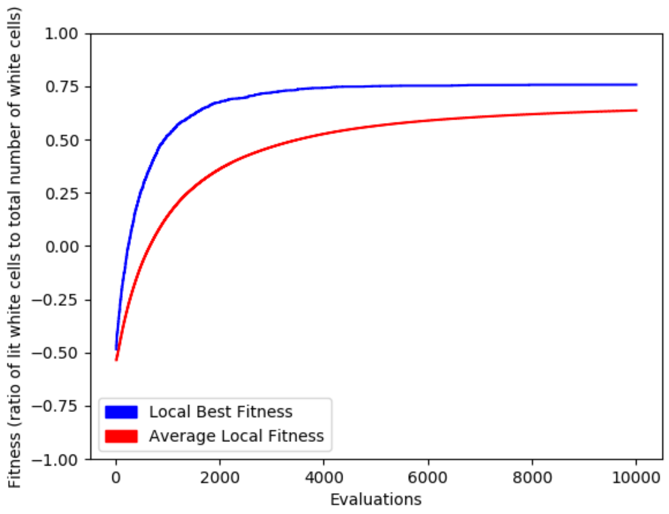

Figure 7: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Penalty Function EA with the Validity Enforced, Provided Puzzle**, Averaged Over All Runs

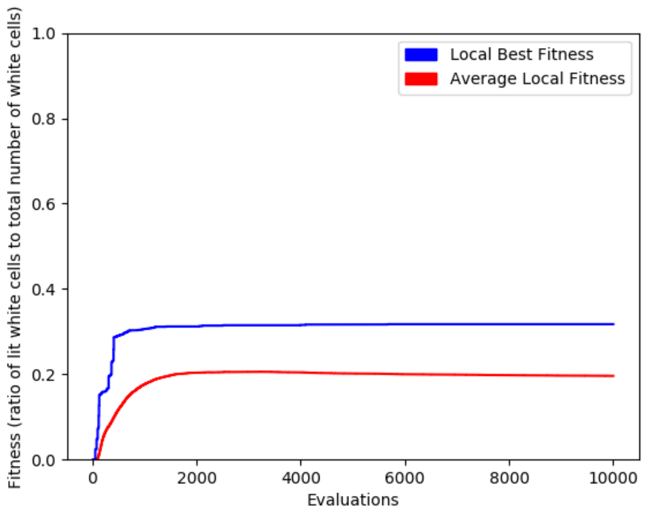

Figure 8: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Repair Function EA with the Validity Enforced, Provided Puzzle**, Averaged Over All Runs

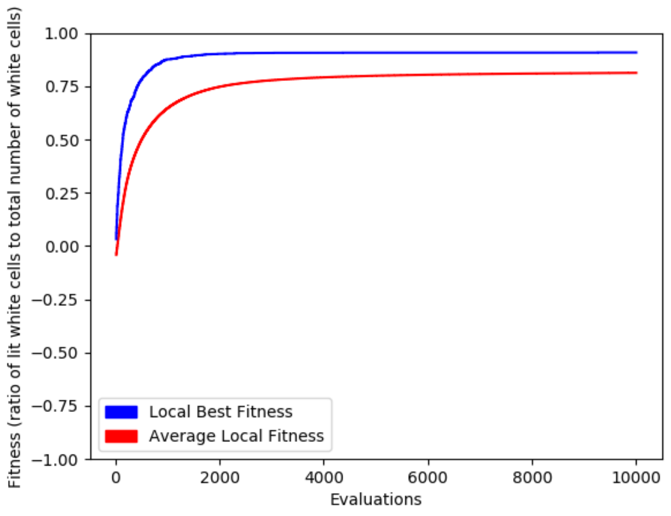

Figure 9: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Penalty Function EA with the Validity Enforced, Randomly Generated Puzzle**, Averaged Over All Runs

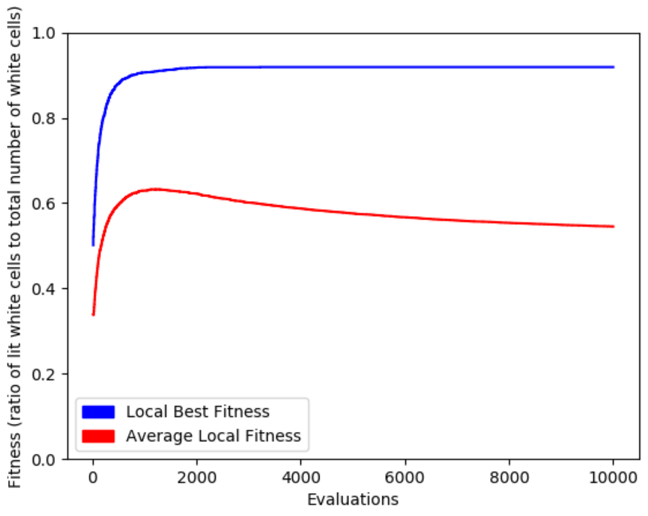

Figure 10: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Repair Function EA with the Validity Enforced, Randomly Generated Puzzle**, Averaged Over All Runs

Through visually examining each configuration's fitness plots, it appears that the penalty function EA outperforms the repair function EA. However, statistical analysis seen in Table

13 and Table 14 proves that for randomly generated puzzles, neither constraint satisfaction method yields a more effective EA. Note that the table headers correspond to the configuration files used to generate the data upon which statistical analysis was performed and that all statistical analysis consisted of the F-Test Two-Sample for Variances followed by the t-test Two-Sample Assuming either Unequal or Equal Variances depending on the F-Test output. A performance boost was proven when examining the provided puzzle tests, proving that the penalty function EA was in fact optimal when compared to the repair function EA for that problem.

Table 13: Statistical Analysis performed on the Validity Enforced Penalty Function and Repair function, Randomly Generated Puzzle, EA configurations

	random_gen_validity_enforced	random_gen_validity_enforced_repair
mean	0.9081384890909392	0.9188060294943169
variance	0.002217915192875166	0.060460786982912955
standard deviation	0.047094746977504466	0.24588775281195474
observations	30	30
df	29	29
F	0.036683531650059054	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	-0.2294580495633768	
P two-tail	0.8200141447652132	
t Critical two-tail	2.0395	
Nether random_gen_validity_enforced_repair nor random_gen_validity_enforced is statistically better		

Table 14: Statistical Analysis performed on the Validity Enforced Penalty Function and Repair function, Provided Puzzle, EA configurations

	website_puzzle_validity_enforced	website_puzzle_validity_enforced_repair
mean	0.7569819819819819	0.3173423423423423
variance	0.0033889801964126286	0.2015810912263615
standard deviation	0.05821494822133426	0.4489778293260832
observations	30	30
df	29	29
F	0.016811994497078302	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	5.229384989438686	
P two-tail	2.4339170289675772e-06	
t Critical two-tail	2.0017	
website_puzzle_validity_enforced is statistically better than website_puzzle_validity_enforced_repair		

The vanilla EA tested on a validity enforced initialization (Figure 11 and Figure 12) performed very poorly compared to the constraint satisfaction EAs. This is because with black cell adjacency constraints enforced, invalid solutions produce fitness values of zero. With many invalid solutions dominating the population, no search gradient is provided and the EA struggles to find any solution.

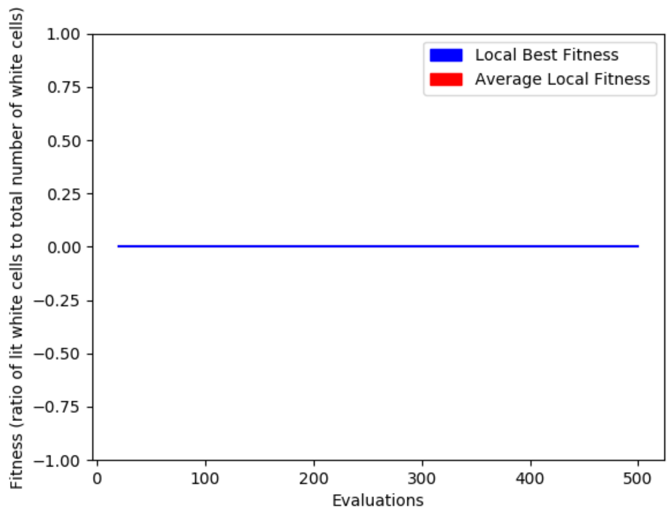

Figure 11: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Plain-Vanilla EA with the Validity Enforced, Provided Puzzle**, Averaged Over All Runs

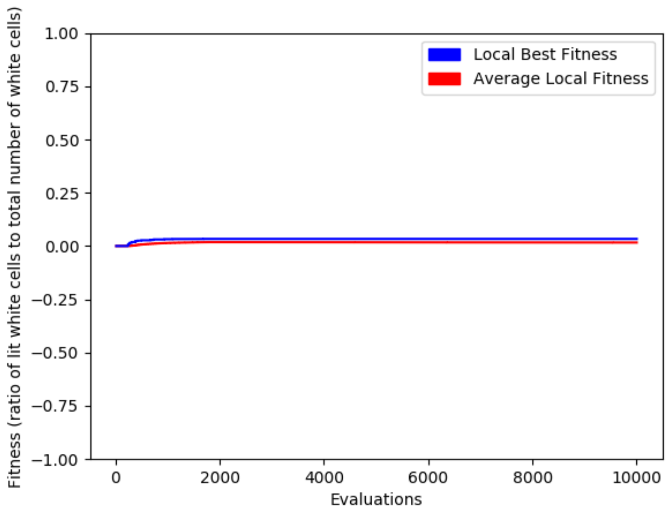

Figure 12: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Plain-Vanilla EA with the Validity Enforced, Randomly Generated Puzzle**, Averaged Over All Runs

Initialization Comparisons

The effect of Validity Forced plus Uniform Random versus plain Uniform Random was examined regarding the plain-vanilla EA, the penalty function EA, and the repair function EA.

The comparison for the plain-vanilla EA tested against the randomly generated puzzle (graphed in Figure 13 and Figure 12, and analyzed in Table 15) proved that neither initialization method led to quantifiable improvements. Granted, neither plain-vanilla EA configuration managed to reliably find meaningful solutions.

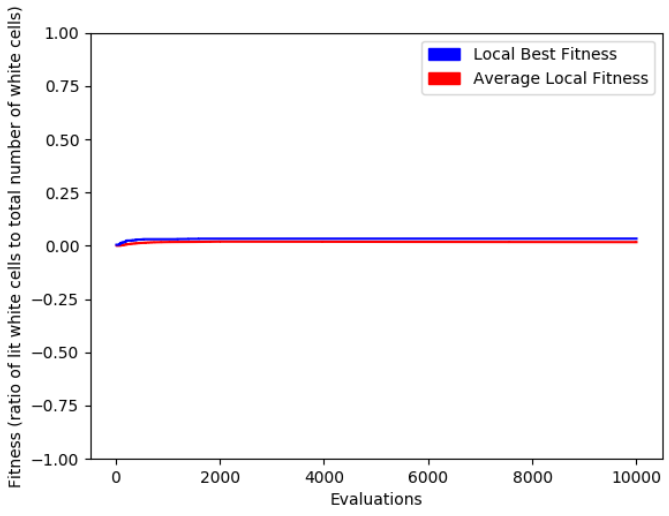

Figure 13: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Plain-Vanilla EA with the Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

Table 15: Statistical Analysis performed on the Uniform Random Initialized Vanilla and Validity Enforced Initialized Vanilla, Randomly Generated Puzzle, EA configurations

	random_gen_uniform_random_vanilla	random_gen_validity_enforced_vanilla
mean	0.03333333333333333	0.03333333333333333
variance	0.032222222222222215	0.03222222222222222
standard deviation	0.17950549357115012	0.17950549357115014
observations	30	30
df	29	29
F	0.9999999999999998	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	0.0	
P two-tail	1.0	
t Critical two-tail	2.0395	
Nether random_gen_validity_enforced_vanilla nor random_gen_uniform_random_vanilla is statistically better		

Figure 9 and Figure 14 illustrate the penalty function EA configuration tested with both Uniform Random and Validity Enforced plus Uniform Random for randomly generated puzzles. Figure 7 and Figure 15 illustrate the same comparison for the provided puzzle. The statistical analysis comparing these two configurations can be found in Table 16 and Table 17.

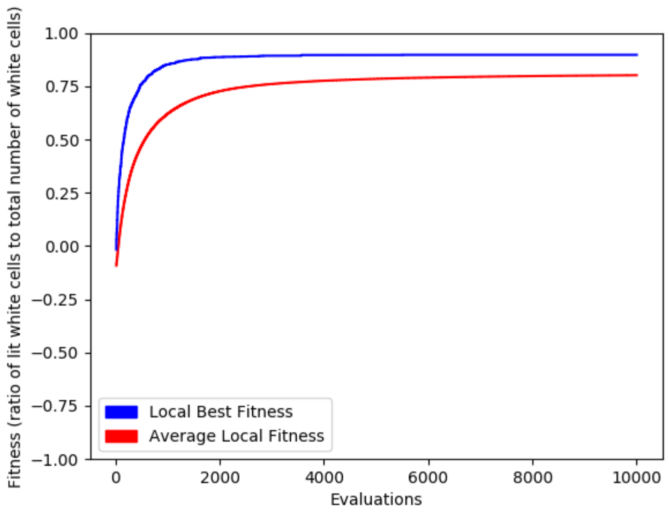

Figure 14: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Penalty Function EA with the Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

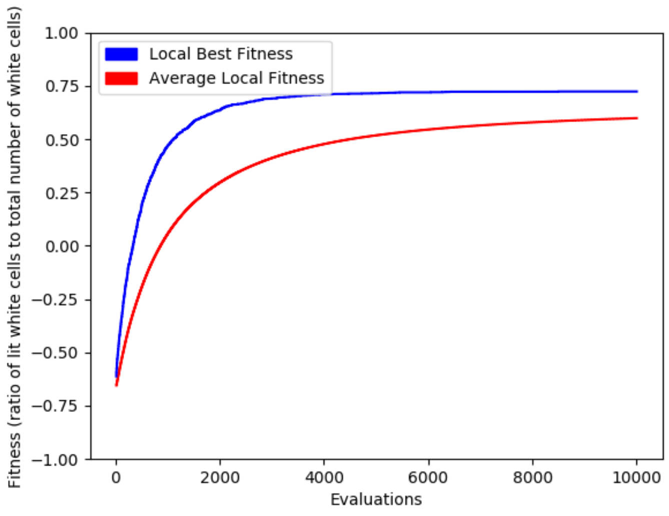

Figure 15: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Penalty Function EA with the Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

Table 16: Statistical Analysis performed on the Uniform Random Initialized Penalty Function and Validity Enforced Initialized Penalty Function, Randomly Generated Puzzle, EA configurations

	random_gen_uniform_random	random_gen_validity_enforced
mean	0.8971591167744307	0.9081384890909392
variance	0.0048536750468879545	0.002217915192875166
standard deviation	0.06966832168846868	0.047094746977504466
observations	30	30
df	29	29
F	2.1883952382308878	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	-0.7031014079094314	
P two-tail	0.48480534283767107	
t Critical two-tail	2.0017	
Nether random_gen_validity_enforced nor random_gen_uniform_random is statistically better		

Table 17: Statistical Analysis performed on the Uniform Random Initialized Penalty Function and Validity Enforced Initialized Penalty Function, Provided Puzzle, EA configurations

	website_puzzle_uniform_random	website_puzzle_validity_enforced
mean	0.7231981981981981	0.7569819819819819
variance	0.0018428394610827035	0.0033889801964126286
standard deviation	0.042928306058854726	0.05821494822133426
observations	30	30
df	29	29
F	0.5437740424194343	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	-2.515248538012428	
P two-tail	0.014687698277300898	
t Critical two-tail	2.0017	
website_puzzle_validity_enforced is statistically better than website_puzzle_uniform_random		

This statistical analysis showed that for randomly generated puzzles, neither initialization method made a quantifiable improvement on the EA's results. Randomly generated puzzles are generally less complicated to solve, so the Validity Enforced initialization's lack of impact in this case is justified.

The analysis also showed that the EA was more sensitive to initialization configuration for the provided puzzle, an objectively difficult puzzle to solve. The Validity Enforced Uniform Random initialization configured EA proved to be statistically better than the Uniform Random initialization configured EA for this problem instance. Since enforcing validity upon initialization shrinks the search space by ensuring the validity of conditions that can only be fulfilled one way (such as a '3' valued black cell placed along an edge leads to one single way to ensure that constraint is valid), and since the search space is more difficult to traverse for highly difficult puzzles, Validity Enforced Uniform Random initialization produces a higher performing EA for the provided puzzle.

The Penalty Coefficient and its Effect on Solution Quality

The penalty function EA, tested against the provided puzzle, was configured using three different penalty coefficients to test their effect on the EA's performance. The first coefficient tested was a small coefficient, equating to 0.25. The second was the medium coefficient, equating to 3. The third was the large coefficient, equating to 10. The fitness graphs for each of these coefficients may be found in Figure 16, Figure 17, and Figure 18 respectively.

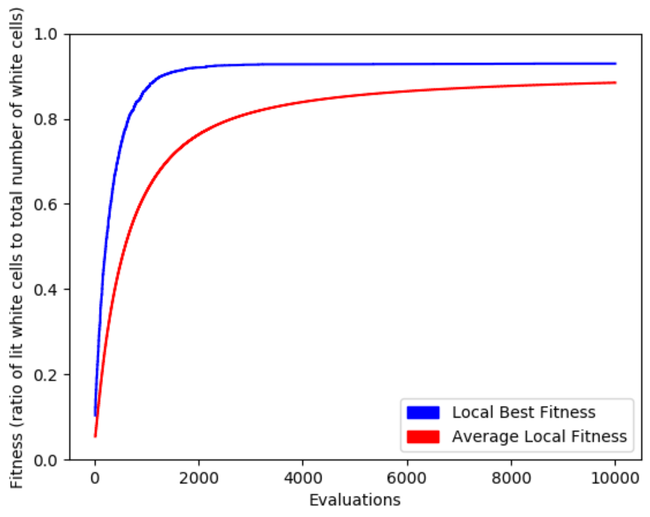

Figure 16: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Penalty Function (with Small Penalty Coefficient = 0.25) EA with the Validity Enforced, Provided Puzzle**, Averaged Over All Runs

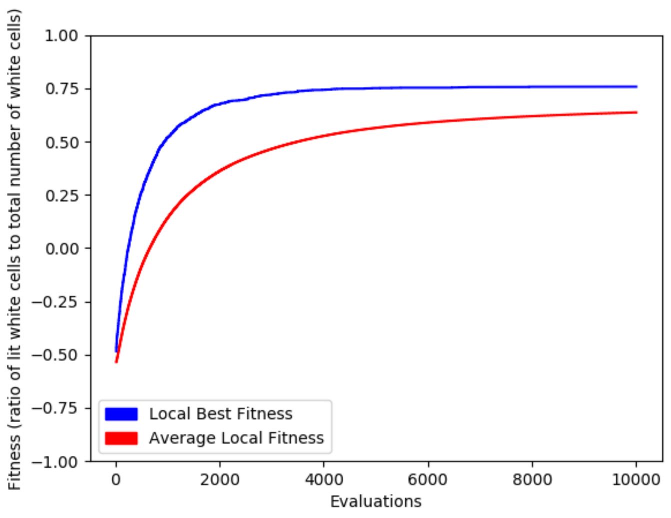

Figure 17: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Penalty Function (with Large Penalty Coefficient = 3) EA with the Validity Enforced, Provided Puzzle**, Averaged Over All Runs

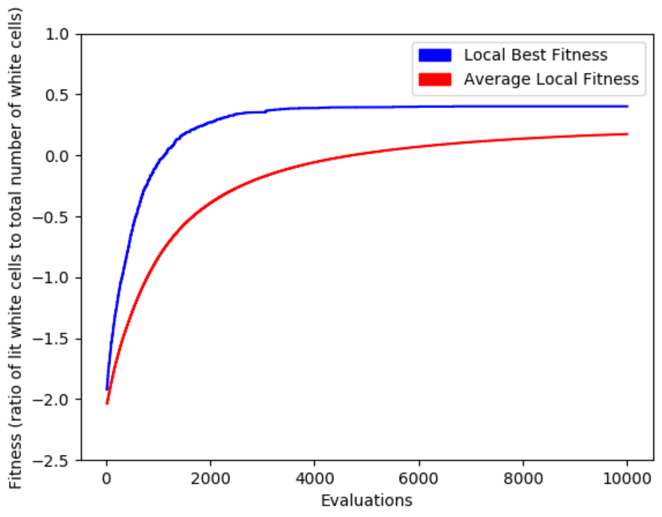

Figure 18: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Penalty Function (with Large Penalty Coefficient = 10) EA with the Validity Enforced, Provided Puzzle**, Averaged Over All Runs

In order to properly compare each configuration, pairwise statistical analysis was performed between each pair of coefficient types (small, medium, and large). Analysis comparing the small and medium coefficients (Table 18) showed that the small penalty coefficient of 0.25 led to an EA that outperformed the EA configured with the medium coefficient of 3. The comparison between the medium and large coefficients (Table 19) led to the conclusion that, again, the smaller coefficient configured EA outperformed the larger penalty coefficient configured EA. This conclusion was also drawn from the last comparison, comparing the small penalty coefficient to the large penalty coefficient (Table 20).

Table 18: Statistical Analysis performed on the Validity Enforced Initialized Penalty Function with Small Penalty and Medium Penalty, Provided Puzzle, EA configurations

	website_puzzle_validity_enforced_small_penalty	website_puzzle_validity_enforced
mean	0.9291666666666667	0.7569819819819819
variance	0.00010251805859913959	0.0033889801964126286
standard deviation	0.010125120177022077	0.05821494822133426
observations	30	30
df	29	29
F	0.030250415363199546	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	15.69233619401264	
P two-tail	1.5429129447240864e-22	
t Critical two-tail	2.0017	
website_puzzle_validity_enforced_small_penalty is statistically better than website_puzzle_validity_enforced		

Table 19: Statistical Analysis performed on the Validity Enforced Initialized Penalty Function with Medium Penalty and Large Penalty, Provided Puzzle, EA configurations

	website_puzzle_validity_enforced	website_puzzle_validity_enforced_large_penalty
mean	0.7569819819819819	0.4020270270270271
variance	0.0033889801964126286	0.024783144631117603
standard deviation	0.05821494822133426	0.15742663253438918
observations	30	30
df	29	29
F	0.13674536653260058	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	11.388392849154927	
P two-tail	2.0169179415154988e-16	
t Critical two-tail	2.0017	
website_puzzle_validity_enforced is statistically better than website_puzzle_validity_enforced_large_penalty		

Table 20: Statistical Analysis performed on the Validity Enforced Initialized Penalty Function with Small Penalty and Large Penalty, Provided Puzzle, EA configurations

	website_puzzle_validity_enforced_small_penalty	website_puzzle_validity_enforced_large_penalty
mean	0.9291666666666667	0.4020270270270271
variance	0.00010251805859913959	0.024783144631117603
standard deviation	0.010125120177022077	0.15742663253438918
observations	30	30
df	29	29
F	0.004136604136604133	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	17.994926171087094	
P two-tail	2.0818133467335364e-25	
t Critical two-tail	2.0017	
website_puzzle_validity_enforced_small_penalty is statistically better than website_puzzle_validity_enforced_large_penalty		

In each comparison case, the smaller penalty function led to a higher best fitness and a higher average population fitness when all run data was averaged. This is because when invalid individuals that do not meet all constraints are penalized, their fitness decreases by a small fraction for smaller penalty coefficients and by a larger fraction for larger penalty coefficients. The final population of a penalty function configured EA using a small penalty function may actually have the same number of invalid solutions as the population of an EA configured with a large penalty coefficient, but the small coefficient EA will read as having higher fitness values. With this fact in mind, it is also important to consider the reason a penalty function constraint satisfaction method is used. Employing a penalty function allows for invalid solutions to be examined with the hope that those invalid solutions lead to a valid solution after some exploration. This can be equated to traversing the 'sea' of validity on the way to a more optimal solution. Keeping the penalty coefficient relatively small incentivizes the traversal of invalid areas of the search space, possibly allowing for a more optimal solution to be discovered.

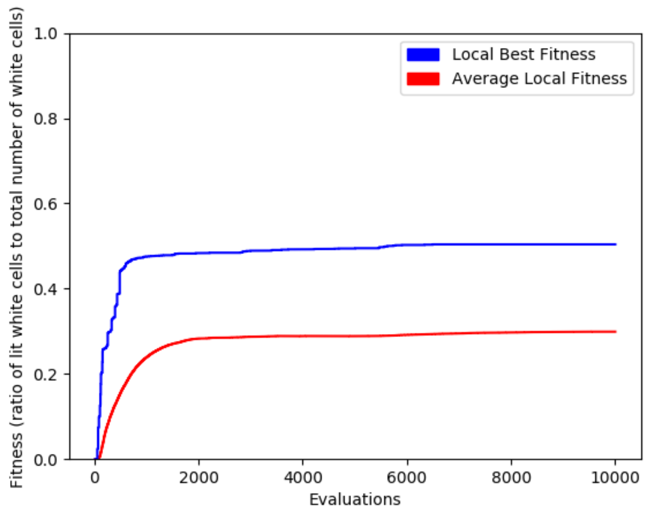

Figure 19: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Repair Function EA with the Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

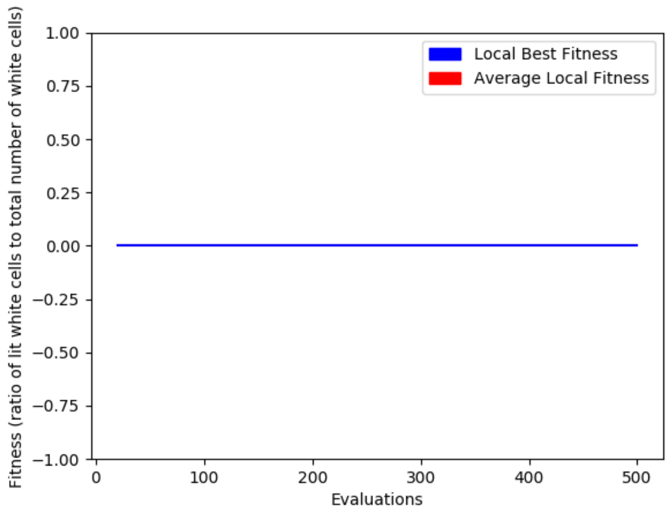

Figure 20: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Plain-Vanilla EA with the Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

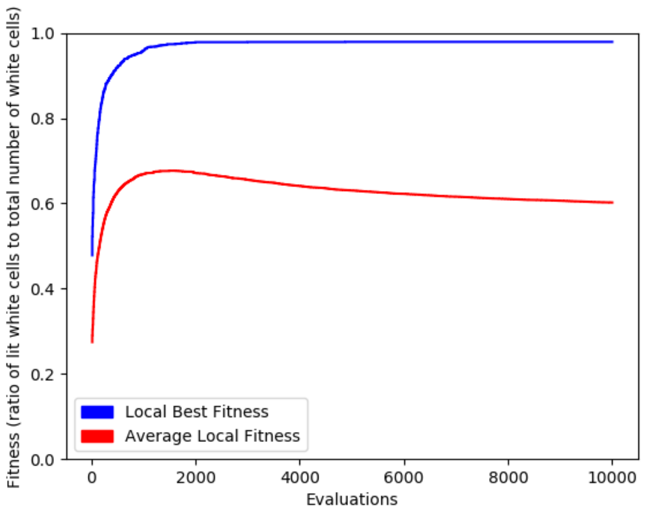

Figure 21: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Repair Function EA with the Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

Table 21: Statistical Analysis performed on the Uniform Random Initialized Repair Function and Validity Enforced Initialized Repair Function, Randomly Generated Puzzle, EA configurations

	random_gen_uniform_random_repair	random_gen_validity_enforced_repair
mean	0.9790060265443229	0.9188060294943169
variance	0.0002560072950964829	0.060460786982912955
standard deviation	0.016000227970141015	0.24588775281195474
observations	30	30
df	29	29
F	0.004234269976817107	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	1.315652066180613	
P two-tail	0.19346782059815182	
t Critical two-tail	2.0017	
Nether random_gen_validity_enforced_repair nor random_gen_uniform_random_repair is statistically better		

Table 22: Statistical Analysis performed on the Uniform Random Initialized Repair Function and Validity Enforced Initialized Repair Function, Provided Puzzle, EA configurations

	website_puzzle_uniform_random_repair	website_puzzle_validity_enforced_repair
mean	0.5038288288288288	0.3173423423423423
variance	0.2221989388036686	0.2015810912263615
standard deviation	0.47137982434939724	0.4489778293260832
observations	30	30
df	29	29
F	1.1022806625952564	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	1.5426809688332364	
P two-tail	0.12835995483574056	
t Critical two-tail	2.0395	
Nether website_puzzle_validity_enforced_repair nor website_puzzle_uniform_random_repair is statistically better		

Multi Objective EA (MOEA)

Introduction

This section involved implementing a Multi-Objective Evolutionary Algorithm (MOEA) to more effectively solve Light Up puzzles by balancing the fulfillment of three objectives:

1. maximize the number of cells lit up (represented in this implementation as a ratio of lit cells to the total number of white cells)
2. minimize the number of bulbs shining on each other
3. minimize the number of black cell adjacency constraint violations

Additionally, a fourth objective was added, namely minimizing the number of bulbs placed on the board. This objective was added for additional exploration of increased numbers of MOEA objectives and was not the main focus of this section.

This report outlines this solution's particular implementation of an MOEA, the impact of initialization strategies on the MOEA's performance, a comparison between the impact of parent selection, survival strategy, and survival selection strategies on MOEA performance, as well as the impact of increasing the number of objectives on MOEA performance.

MOEA Overview

The MOEA implemented as part of this section is based on the NSGA-II algorithm. It begins, similar to a standard evolutionary algorithm, by creating an initial population using either uniform random or validity enforced plus uniform random initialization, the settings for which are specified in the algorithm configuration file. That population is evaluated and the subfitnesses are determined and assigned to each individual in the population.

The population is then evaluated on the basis of non-domination. A list of Pareto fronts is created from the initial population where all genotypes in a given front are not dominated by any other genotypes in that front while genotypes in higher level fronts are dominated by genotypes in lower level fronts. The 'best' genotypes, those in the best level of non-domination, are assigned to level number one. Subsequent levels increase in increments of one for other levels of non-domination.

The fitness of each genotype is then set to its level in the list of Pareto fronts, with individuals exhibiting a smaller fitness (level number) are more fit. A binary tournament selection is performed to choose breeding parents. Then offspring are created using an n-point crossover recombination (with n determined in the configuration file). Following that, mutation is performed, completing the child population.

For the standard NSGA-II configuration (exhibited in the deliverables configuration folder), the plus survival strategy is exhibited, combining the children and parent populations into one large population from which to choose the new population. Individuals are then selected for survival using a binary tournament selection and the process is repeated using the new population until the end of the experiment.

Impact of Initialization on MOEA Performance

The effect of Validity Enforced plus Uniform Random versus Uniform Random initialization was examined in this experiment for both the provided puzzle and randomly generated puzzles. One would assume that an initialization method utilizing Validity Enforced initialization would outperform a solely Uniform Random initialization. After performing statistical analysis on the experiment data, it was concluded that there is in fact no tangible difference between the initialization methods for the tested puzzles. Table 23 and Table 24 each display statistical analysis supporting this finding.

Table 23: Statistical Analysis performed on the Uniform Random and Validity Enforced Uniform Random Initialized, Randomly Generated Puzzle, EA configurations, EA configurations

	random_gen	random_gen_uniform_random_init
mean	1.10325284795678	1.0764116503308314
variance	0.04970344216568983	0.04355857958196582
standard deviation	0.22294268807406498	0.20870692269775296
observations	30	30
df	29	29
F	1.141071234248146	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	0.47331304656977363	
P two-tail	0.6377741699825987	
t Critical two-tail	2.0395	
Nether random_gen_uniform_random_init nor random_gen is statistically better		

Table 24: Statistical Analysis performed on the Uniform Random and Validity Enforced Uniform Random Initialized, Provided Puzzle, EA configurations, EA configurations

	website_puzzle	website_puzzle_uniform_random_init
mean	0.8031737242867948	0.7959489651519909
variance	0.000603666878279215	0.00058947613524421
standard deviation	0.024569633254878164	0.02427912962287178
observations	30	30
df	29	29
F	1.02407348183676	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	1.1263571505364551	
P two-tail	0.26465388827990055	
t Critical two-tail	2.0395	
Nether website_puzzle_uniform_random_init nor website_puzzle is statistically better		

This statistical analysis for this section was performed on the average of all last best subfitnesses, giving thirty data points to compare for each MOEA initialization method. The analysis consisted of performing an f-test, which determined if variances could be treated as equal. In both cases, the f-test yielded that unequal variances should be assumed. Following the f-test, the two-tailed t-test was performed assuming unequal variances. This test yielded (in both cases) that neither initialization method was statistically better for the set of Light Up puzzles tested.

To visually interpret the data, plots of evaluations versus average local subfitness and evaluations versus local best subfitness were graphed for each of the subfitnesses collected in this experiment: ratio of lit cells to total number of white cells, number of bulbs shining on each other, and number of black cell constraints not met. For concision, figures pertaining to only the provided puzzle are discussed in this section as the randomly generated puzzle results are quite similar. Figures 22, 23, 24, 25, 32, and 33 depict experiment plots for the provided puzzle while figures 29, 26, 27, 28, 30, and 31 depict experiment plots for the randomly generated puzzle.

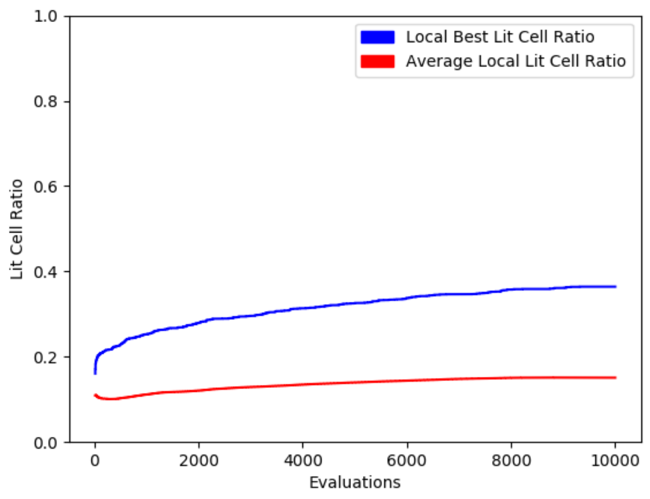

Figure 22: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Lit Cell Ratio Subfitness, Validity Enforced plus Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

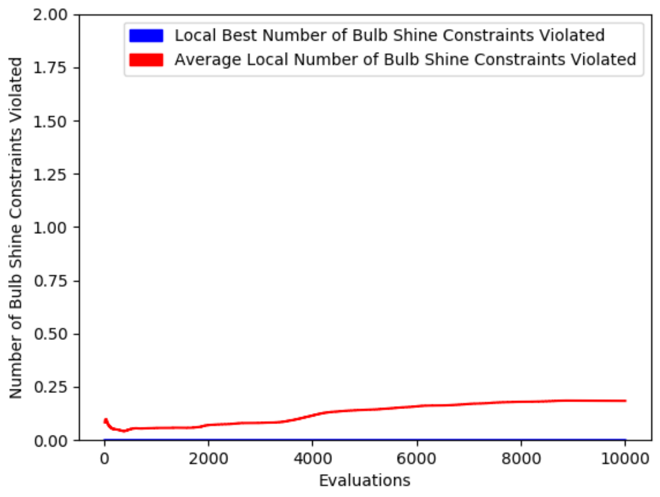

Figure 23: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Bulb Shine Constraint Subfitness, Validity Enforced plus Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

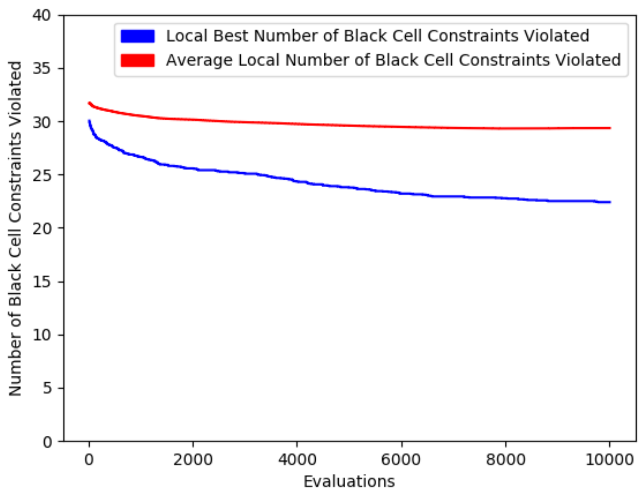

Figure 24: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Black Cell Constraint Subfitness, Validity Enforced plus Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

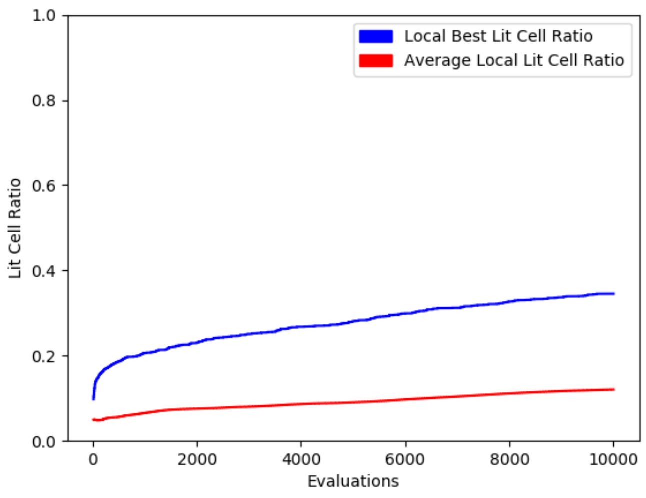

Figure 25: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Lit Cell Ratio Subfitness, Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

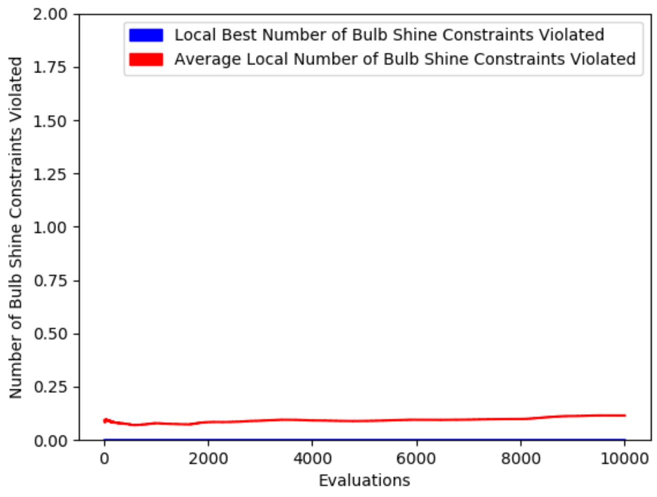

Figure 26: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Bulb Shine Constraint Subfitness, Validity Enforced plus Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

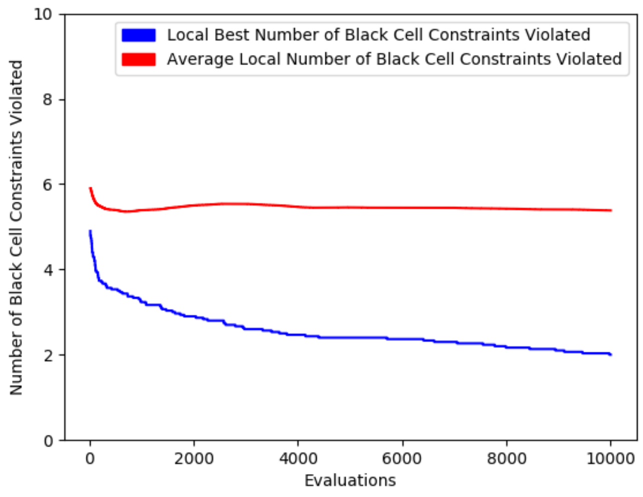

Figure 27: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Black Cell Constraint Subfitness, Validity Enforced plus Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

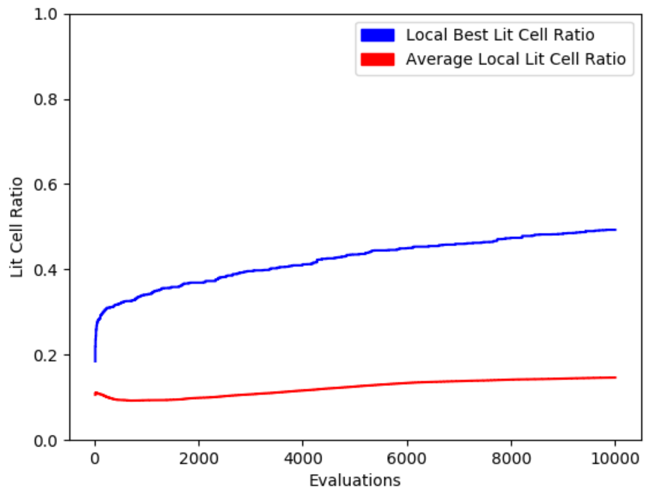

Figure 28: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Lit Cell Ratio Subfitness, Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

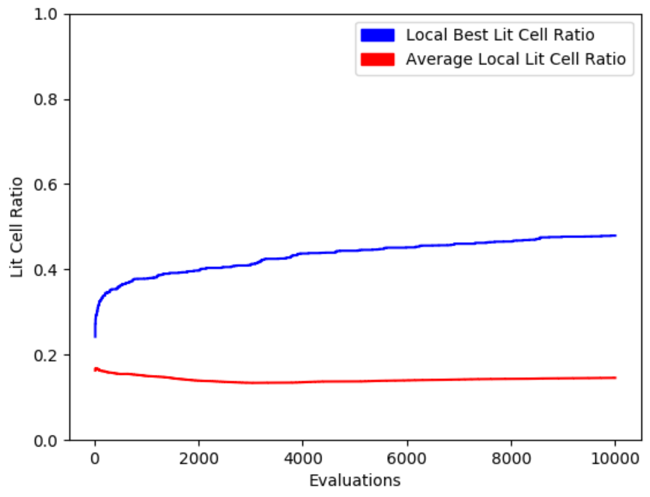

Figure 29: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Lit Cell Ratio Subfitness, Validity Enforced plus Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

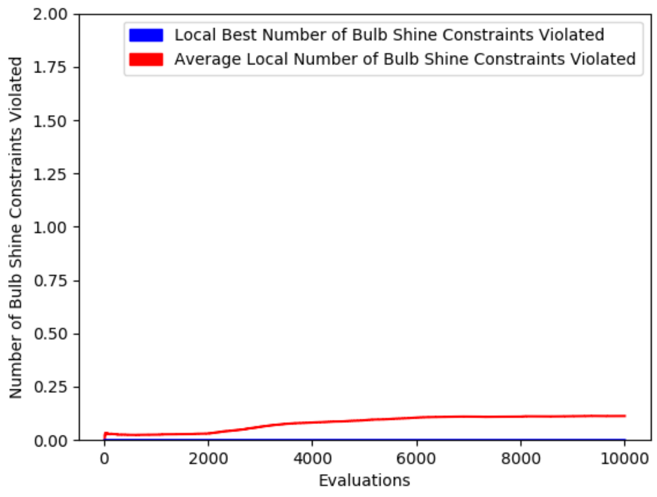

Figure 30: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Bulb Shine Constraint Subfitness, Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

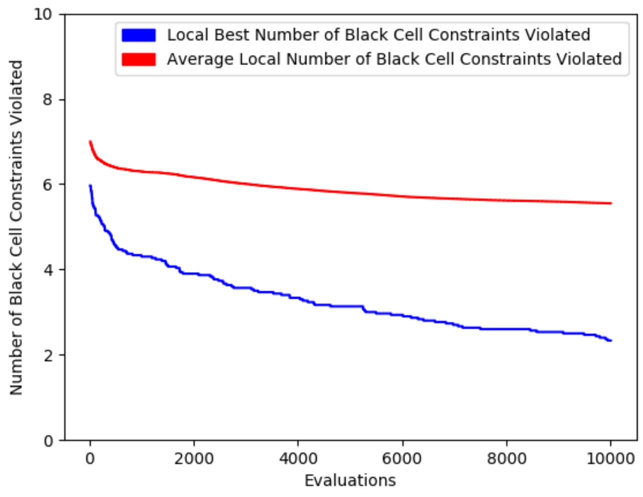

Figure 31: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Black Cell Constraint Subfitness, Uniform Random Initialized, Randomly Generated Puzzle**, Averaged Over All Runs

Figure 22 and Figure 25 depict the lit cell ratio subfitness plots for the Validity Enforced plus Uniform Random and Uniform Random initialized experiments, respectively. In the uniform random case (Figure 25), both the average local list cell ratio and the local best lit cell ratio started lower than those of the validity enforced experiment. However, as the experiments progressed, both the average and best subfitnesses of each experiment reached appreciatively the same point without the best subfitness plateauing. This implies that letting the experiment run for longer would produce a more fit solution, with respect to the lit cell ratio. Note that the lit cell ratio subfitness, the metric was maximized.

Figure 23 and Figure 32 depict the evaluations versus subfitness plots for the bulb shine constraint violations. These plots behaved quite similarly to the lit cell ratio plots in that the Validity Enforced plus Uniform Random plot started with a higher average number of bulb shine constraints violated while the plain Uniform Random plot had a lower number of initial constraint violations. This is logical as enforcing validity has the potential to place more bulbs, which creates opportunity for more bulbs to shine on each other, further adding to the number of bulb shine violations. The local best for both plots stayed right at zero bulb constraint violations, implying that most, if not all, experiments always had at least one member of the population with no bulb shine constraints. Because this subfitness is to be minimized as part of the MOEA, it is peculiar that the average number of bulb constraints increases as the experiments continue. This implies that as more bulbs are placed on the board, the multi-objective nature of the algorithm allows for more and more bulbs to shine on each other, keeping those individuals with more bulb-shine in the population so long as the levels of non-domination dictate it.

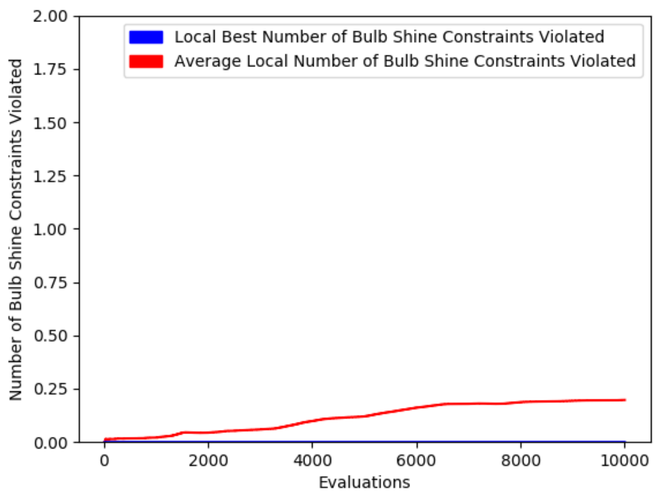

Figure 32: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Bulb Shine Constraint Subfitness, Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

The third subfitness, black cell constraint violations, was examined for both initialization schemes in Figure 24 and Figure 33. Before examining these plots, it was hypothesized that the Validity Enforced plus Uniform Random initialization method would be superior with respect to minimizing the black cell constraint violations when compared to the Uniform Random only initialization. While the statistical analysis was not granular enough to definitively prove this, the graphs provide anecdotal evidence that the Validity Enforced plus Uniform Random initialization method is in fact superior when it comes to the black cell constraint violations. This conclusion was drawn because the Validity Enforced plus Uniform Random method had lower average and local best black cell constraints violated at each point on the graph, across all experiments.

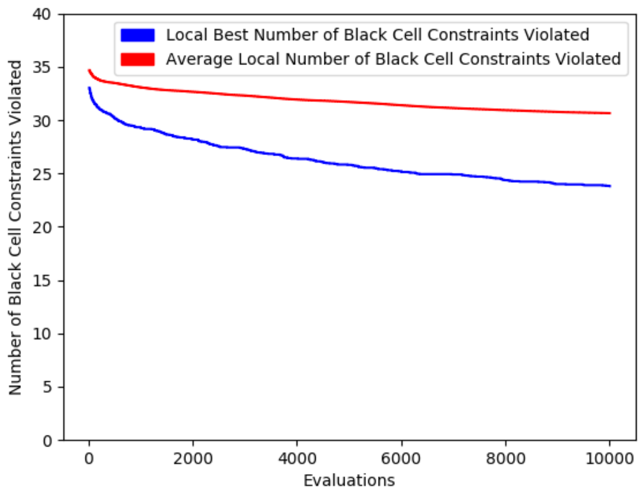

Figure 33: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Black Cell Constraint Subfitness, Uniform Random Initialized, Provided Puzzle**, Averaged Over All Runs

Comparison of Parent Selection Strategy, Survival Strategy, and Survival Selection Strategy MOEA Configurations

All combinations of the following MOEA configurations were compared, for both randomly generated puzzles and the provided puzzle, to determine which configuration combination was optimal.

- Parent Selection
  - Fitness Proportional
  - Binary Tournament
- Survival Selection
  - Fitness Proportional
  - Binary Tournament
- Survival Strategy
  - Plus

- Comma

Each combination of configurations resulted in $n = 2^3 = 8$ distinct configurations. Each configuration was tested against each other configuration (excluding itself), resulting in

$$n * (n - 1) = 8 * 7 = 56$$

comparisons per puzzle type. For each comparison, three statistical comparisons were made (each involving an f-test followed by a t-test, described in more detail below) between the three subfitnesses for each configuration. This resulted in $56 * 3 = 168$ statistical tests performed. For both the provided puzzle and the randomly generated puzzle, this resulted in $168 * 2 = 336$ comparisons total.

The explicit statistical analysis for each of the 336 individual comparisons is not tabulated in this report for brevity, however, an example of the statistical analysis performed for one of the dominant MOEA configuration comparisons for both the provided puzzle and the randomly generated puzzle can be found in tables 25, 26, and 27 for the provided puzzle experiments and in tables 28, 29, and 30 for the randomly generated puzzle experiments. Additionally, figures for one of the dominant MOEA configurations can be found in figures 34, 35, and 36 for the provided puzzle experiments and in figures 37, 38, and 39 for the randomly generated puzzle experiments. *Note that these dominating configurations are described at length below in this section.*

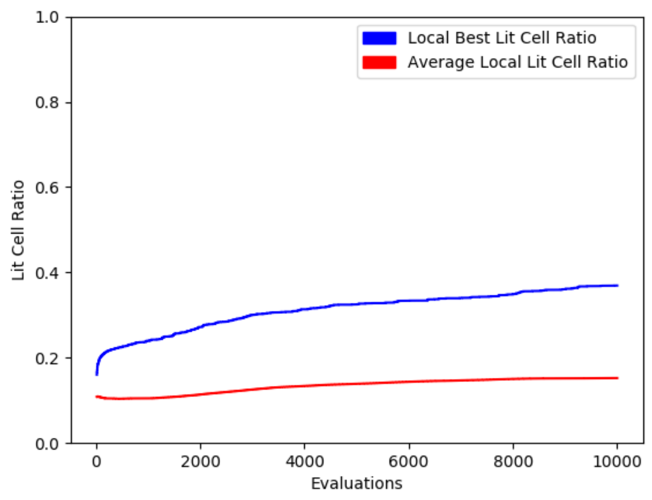

Figure 34: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Lit Cell Ratio Subfitness, Tournament Parent & Survival Selection, Plus Survival Strategy, Provided Puzzle**, Averaged Over All Runs

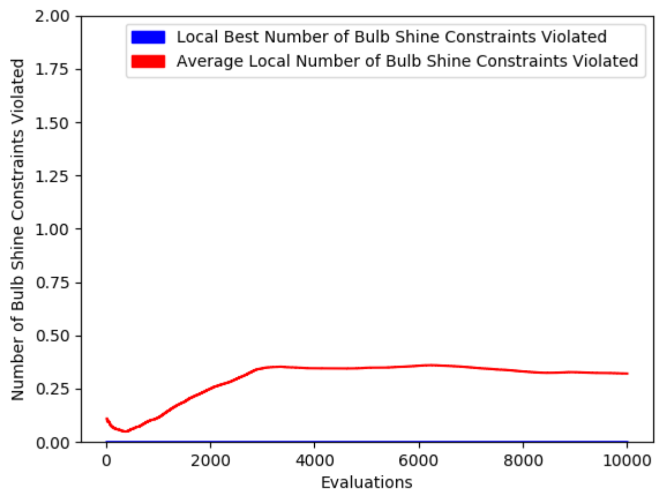

Figure 35: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Bulb Shine Constraint Subfitness, Tournament Parent & Survival Selection, Plus Survival Strategy, Provided Puzzle**, Averaged Over All Runs

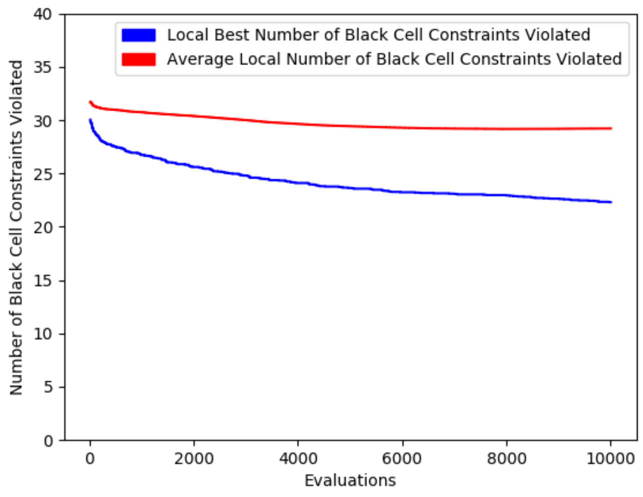

Figure 36: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Black Cell Constraint Subfitness, Tournament Parent & Survival Selection, Plus Survival Strategy, Provided Puzzle**, Averaged Over All Runs

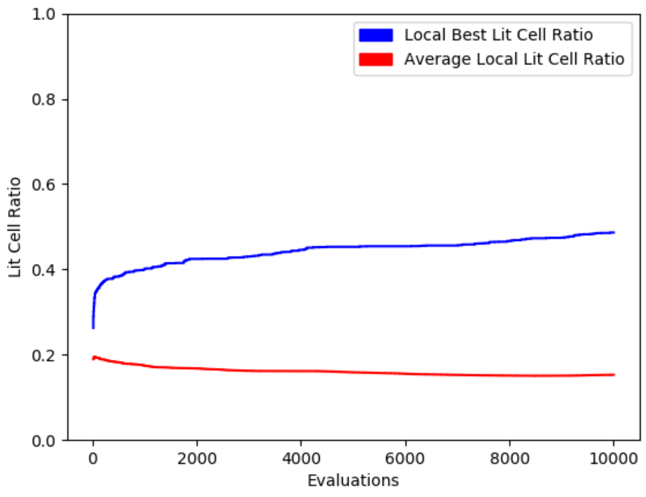

Figure 37: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Lit Cell Ratio Subfitness, Tournament Parent & Survival Selection, Plus Survival Strategy, Randomly Generated Puzzle**, Averaged Over All Runs

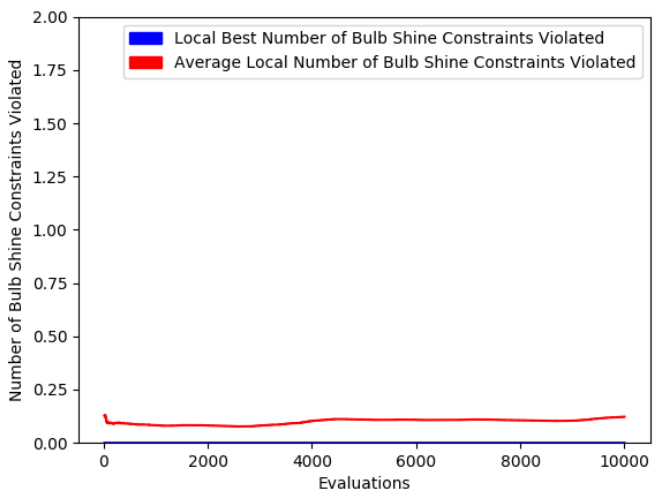

Figure 38: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Bulb Shine Constraint Subfitness, Tournament Parent & Survival Selection, Plus Survival Strategy, Randomly Generated Puzzle**, Averaged Over All Runs

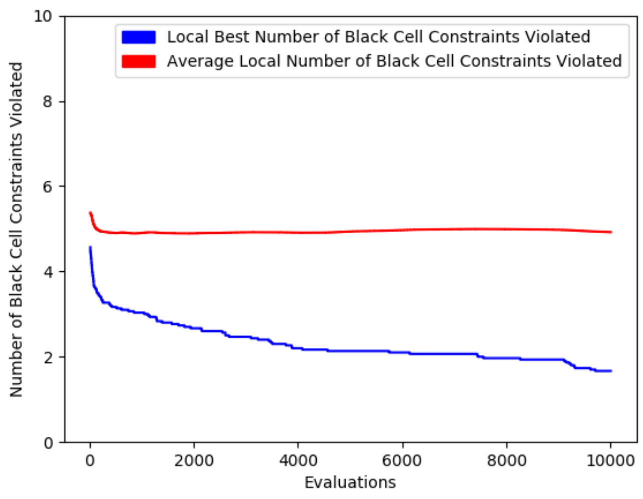

Figure 39: Evaluations versus Average Local Fitness and Evaluations versus Local Best Fitness for the **Black Cell Constraint Subfitness, Tournament Parent & Survival Selection, Plus Survival Strategy, Randomly Generated Puzzle**, Averaged Over All Runs

Table 25: Subfitness Statistical Analysis performed against the Tournament Parent & Survival Selection, Plus Survival Strategy, Provided Puzzle, EA Configuration

	website_puzzle__fitness_proportional_parent__fitness_proportional_survival__comma	website_puzzle__tournament_parent__tournament_survival__plus
mean	0.03557475942533414	0.045719133121307036
variance	1.0798324737368128e-06	4.716790367219106e-05
standard deviation	0.001039149880304479	0.006867889317118547
observations	30	30
df	29	29
F	0.022893374300487587	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	-7.864765318971465	
P two-tail	8.251713123765557e-09	
t Critical two-tail	2.0395	
website_puzzle__tournament_parent__tournament_survival__plus is statistically better than website_puzzle__fitness_proportional_parent__fitness_proportional_survival__comma		

Table 26: Subfitness Statistical Analysis performed against the Tournament Parent & Survival Selection, Plus Survival Strategy, Provided Puzzle, EA Configuration

	website_puzzle__fitness_proportional_parent__fitness_proportional_survival__comma	website_puzzle__tournament_parent__tournament_survival__plus
mean	0.33063063063063064	0.3691441441441441
variance	0.0004167681194708223	0.005361222709195682
standard deviation	0.020414899447972364	0.07322037086218344
observations	30	30
df	29	29
F	0.07773751289159707	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	-2.728498473507591	
P two-tail	0.01005918482437467	
t Critical two-tail	2.0395	
website_puzzle__tournament_parent__tournament_survival__plus is statistically better than website_puzzle__fitness_proportional_parent__fitness_proportional_survival__comma		

Table 27: Subfitness Statistical Analysis performed against the Tournament Parent & Survival Selection, Plus Survival Strategy, Provided Puzzle, EA Configuration

	website_puzzle__fitness_proportional_parent__fitness_proportional_survival__plus	website_puzzle__tournament_parent__tournament_survival__plus
mean	0.03543878854223681	0.045719133121307036
variance	6.924752843254994e-07	4.716790367219106e-05
standard deviation	0.0008321509985125893	0.006867889317118547
observations	30	30
df	29	29
F	0.014681069761719437	
F critical	0.5373999648406917	
Unequal variances assumed		
observations	30	
df	31	
t Stat	-8.002369566124022	
P two-tail	6.4560972436788905e-09	
t Critical two-tail	2.0395	
website_puzzle__tournament_parent__tournament_survival__plus is statistically better than website_puzzle__fitness_proportional_parent__fitness_proportional_survival__plus		

Table 28: Subfitness Statistical Analysis performed against the Tournament Parent & Survival Selection, Plus Survival Strategy, Randomly Generated Puzzle, EA Configuration

	random_gen__fitness_proportional_parent__fitness_proportional_survival__comma	random_gen__tournament_parent__tournament_survival__plus
mean	0.4827645502645503	1.0
variance	0.3007261451597099	0.5472222222222223
standard deviation	0.5483850336758926	0.7397447007057383
observations	30	30
df	29	29
F	0.5495503160278963	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	-3.024841182589063	
P two-tail	0.003703278639359185	
t Critical two-tail	2.0017	
random_gen__tournament_parent__tournament_survival__plus is statistically better than random_gen__fitness_proportional_parent__fitness_proportional_survival__comma		

Table 29: Subfitness Statistical Analysis performed against the Tournament Parent & Survival Selection, Plus Survival Strategy, Randomly Generated Puzzle, EA Configuration

	random_gen__fitness_proportional_parent__fitness_proportional_survival__plus	random_gen__tournament_parent__tournament_survival__plus
mean	0.561190476190476	1.0
variance	0.42458320105820113	0.5472222222222223
standard deviation	0.6516004919106501	0.7397447007057383
observations	30	30
df	29	29
F	0.7758880831520426	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	-2.397095764759235	
P two-tail	0.019767792049039033	
t Critical two-tail	2.0017	
random_gen__tournament_parent__tournament_survival__plus is statistically better than random_gen__fitness_proportional_parent__fitness_proportional_survival__plus		

Table 30: Subfitness Statistical Analysis performed against the Tournament Parent & Survival Selection, Plus Survival Strategy, Randomly Generated Puzzle, EA Configuration

	random_gen__fitness_proportional_parent__fitness_proportional_survival__plus	random_gen__tournament_parent__tournament_survival__plus
mean	0.35268032880882605	0.4864500836154015
variance	0.01582075351042015	0.00448740286063112
standard deviation	0.12578057684086263	0.06698807998913776
observations	30	30
df	29	29
F	3.525592419886963	
F critical	0.5373999648406917	
Equal variances assumed		
observations	30	
df	58	
t Stat	-5.055006060309308	
P two-tail	4.60221704581319e-06	
t Critical two-tail	2.0017	
random_gen__tournament_parent__tournament_survival__plus is statistically better than random_gen__fitness_proportional_parent__fitness_proportional_survival__plus		

The statistical analysis methodology is described as follows: First, the last best subfitness for each of the three subfitness were aggregated and written to separate files for each of the configuration schemes listed above. This resulted in $3 * 8 = 24$ data files to drive the statistical analysis for each the provided puzzle and the randomly generated puzzle (48 in total). Following this, each possible pairing (in which order does matter) was performed, comparing from each configuration each of the three last best subfitness lists using first an f-test to determine whether or not equal variances could be assumed followed by a two-tailed t-test assuming either equal or unequal variances (depending on the outcome of the f-test). The f-test would then yield if a configuration was better with respect to each subfitness. If the number of a configuration's subfitness' superiorities outnumbered the number of a configuration's subfitness' subordinates for a given comparison, that configuration was marked as being better than the other, incrementing a dominance count associated with that configuration. The configuration(s) with the highest dominance count are statistically better than all other configurations tested for the given test puzzles.

The randomly generated puzzle configurations yielded two configurations that were both equally as optimal (they both dominated two other configurations, while all other configurations dominated no other configurations). These were MOEAs configured with (1) fitness proportional parent selection, tournament survival selection, and plus survival strategy and (2) tournament parent selection, tournament survival selection, and plus survival strategy. Both of these solutions each were dominant over two other configurations out of all the other configurations. They were either at least as good as or subordinate to the other configurations. The outcome of these comparisons proved that for this MOEA tested against randomly generated puzzles, the plus method for survival strategy is optimal, as both dominant configurations employed this strategy for at least one selection. The results also showed that relatively *low* selection pressure for parent and survival selections led to better algorithm results. The binary tournament selection applies low selection pressure and is prevalent in both configurations - the first configuration used binary tournament selection for survival selection while the second used a binary tournament selection for both the parent and survival selections. This method of selection favors exploration over exploitation in traversing the solution search space for optimal bulb placements.

The provided puzzle configurations yielded one configuration that was optimal when compared to all other solutions. In fact, this configuration was the only solution found to be statistically better than all other configurations. This configuration involved a tournament parent selection and tournament survival selection with a plus survival strategy. Similarly to the above dominance results, the results for the provided puzzle showed that configurations employing a low selection pressure and a plus survival strategy produced better results. This dominance was more prevalent for the provided puzzle, as the winning configuration dominated all other configurations. Again, this proves that for this particular MOEA, an approach of exploration over exploitation is integral in achieving more optimal results.

Impact of Increasing Number of Objectives on MOEA Performance

A fourth subfitness was added to the MOEA, namely a constraint minimizing the number of bulbs placed on the board. This constraint directly contradicted one of the main measures of an EA's performance - the ratio of lit cells to the total number of white cells. For an MOEA to be successful with only the lit cell ratio, bulb shine constraint violations, and black cell constraint violations subfitnesses, it did not explicitly need to minimize the number of bulbs placed. In fact, the only thing *limiting* the number of bulbs placed on the board was the bulb shine constraint violation metric. By adding the fourth subfitness of minimizing the number of placed bulbs, the MOEA is more likely to be conservative regarding the placement of bulbs on a board, as now half of the four total subfitness metrics actively limit the number of placed bulbs. Although statistical analysis was not performed regarding the implication of an increased number of subfitness objectives and its impact on MOEA performance, the following hypothesis is offered instead: If a fourth subfitness objective is added which minimizes the number of bulbs placed on the board, the best level of non-domination will (overall, across many runs) prove to contain genotypes with lower lit-cell ratios as the incentive for placing more bulbs will be limited while the incentive for lighting up many cells will stay the same.

Conclusion

Knowing the varying ways to solve a problem is important when problem solving in general. There are many different techniques, especially in the field of computational intelligence, so it can save time and money to know which ones to use. Our results showcase the different performances between different approaches to solving the same problem. In our case, we found that our constraint satisfaction EA performed the best for the given problem space of Light Up puzzles. Despite this, it may be that with a differing implementation or perhaps a different problem space, a different algorithm would have performed better.

Random search was implemented as part of this project for comparison purposes. Additionally, a standard EA, constraint satisfaction EA, and MOEA were implemented alongside the random search to explore paradigms associated with evolutionary algorithms. After completing the software and reporting related to this project, the authors have a better understanding of implementation details regarding various EAs as well as a general view of the strengths and weaknesses of each algorithm implemented.

Each of these algorithms have strengths and weakness based on the way they evolve to solve the problem, and knowing the inner workings of many different EA's helps in the ultimate goal that lead to the creation computer programs and computational intelligence in the first place; solving problems.