Verkfræðilegar Bestunaraðferðir

Day 9 - Group 2

T-423-ENOP

Arnar Gylfi Haraldsson
arnarh23@ru.is

Hafþór Árni Hermannsson
hafthorh20@ru.is

Ragnheiður Gná Gústafsdóttir
ragnheidurg@ru.is

May 16, 2024

Exercise 1 - Multiobjective Evolutionary Algorithm

Description

Implement a multiobjective evolutionary algorithm using a floating point representation that utilizes the following mechanisms:

1. Selection based on Pareto dominance tournaments
2. Assessment of individuals based on Pareto ranking
3. Dynamic sharing[1]
4. Mating restriction[2]
5. Shared-fitness-based archiving procedure[3]
6. Termination condition based on convergence measure $p^{(k)}$[4]
7. Visualization procedures to observe current population and elitist individuals

[1] Initial implementation may use static sharing with predefined σ_{share}.

[2] Initial implementation may use regular mating with no restrictions.

[3] Initial implementation may archive all non-dominated individuals.

[4] Initial implementation may use maximum number of generations as the termination condition.

Test the algorithm using functions f_f and f_r, which are found in the lecture slides.
Perform experiments in order to tune control parameters, particularly crossover and mutation probability, mutation distribution, mating range, etc.
Perform experiments in order to test the importance of different components of the algorithm, in particular sharing and mating restrictions.

Functionality

Since this algorithm is quite complex and has a lot of attributes we have split the functionality section into three subsections: Multi-Objective Evolutionary Algorithm, Supporting functions and Utility functions.

Main Function: MOEA (Multi-Objective Evolutionary Algorithm)

- **Initialization**: Initializes the population within the given bounds.
- **Evaluation**: Evaluates the objective values for the population.
- **Update Pareto Front**: Updates the current Pareto front and set.
- **Archive**: Archives non-dominated individuals using shared fitness.
- **Selection**: Selects individuals based on Pareto dominance and tournament selection.
- **Crossover and Mutation**: Generates offspring through crossover and mutation.
- **Dynamic Sharing Adjustment**: Adjusts the sharing and mating distances dynamically.
- **Termination**: Stops the algorithm if the algorithm has converged. We decided to make that if the differences from the last two generations are less than 0.1, then convergence has been reached.

0.1 Supporting Functions

- **initialize_population**:
  - Generates an initial population of size N with specified dimensions.
  - Each individual's variables are initialized randomly within the given bounds.
- **evaluate_population**:
  - Evaluates the objective functions for each individual in the population.
  - Returns a matrix of objective values.
- **update_pareto_front**:
  - Determines non-dominated solutions in the population.
  - Updates the Pareto front and Pareto set as the non-dominated individuals in this generations population.
- **update_archive**:
  - Adds current non-dominated individuals to the archive.
  - Removes dominated individuals from the archive to maintain a set of non-dominated solutions.
- **update_archive_shared_fitness**:
  - Adds current non-dominated individuals to the archive.
  - Computes shared fitness values to promote diversity.
  - Sorts individuals based on shared fitness and updates the archive.
  - Removes dominated individuals to maintain non-dominated solutions.
- **shared_fitness**:
  - Computes fitness values considering niche formation.
  - Penalizes fitness based on proximity to other individuals within a specified sharing distance, σ_{share}.
- **selection**:
  - Forms the mating pool using tournament selection based on Pareto dominance.
  - Ensures selected individuals are within the mating distance, σ_{mate}.
- **crossover_and_mutation**:
  - Produces new offspring by applying crossover with a specified probability.
  - Mutates offspring with a specified probability to introduce variability.
  - Ensures offspring variables remain within the given bounds.
- **dynamic_sharing**:
  - Adjusts the sharing distance dynamically based on the diversity of the Pareto front.
  - Uses distances between Pareto front individuals to compute new sharing distance.
- **convergence_measure**:
  - Calculates convergence based on changes in the Pareto front between generations.
  - Uses the set difference to measure how much the Pareto front has shifted.
- **convergence_measure_p**:
  - Measures convergence based on the ratio of non-dominated individuals between successive generations.
  - Helps determine if the algorithm is converging to a stable Pareto front.

Utility Functions

- **dominates**: Checks if one solution dominates another.
- **ff**: Wrapper function to evaluate the `ff` objective functions.
- **ft**: Wrapper function to evaluate the `ft` objective functions.

Results

We would like to start this results section by saying that we are not happy with these results but we do not know how to better the code further without help, which is unfortunately not available due to the professors sudden illness.

FF function

We tested the following values of P_c, P_m, σ_{share} and σ_{mate}:

Table 1: Tested Values for Crossover Probability, Mutation Probability, Sharing Distance, and Mating Distance

Parameter	**Tested Values**
Crossover Probability (p_c)	0.5, 0.7, 0.9
Mutation Probability (p_m)	0.05, 0.1, 0.2
Sharing Distance (σ_{share})	0.1, 0.3, 0.5
Mating Distance (σ_m)	0.1, 0.3, 0.5

Figures 1 to 15 show the resulting graphs, through visualization we determined that the best values for each variable are:

- $P_c = 0.7$
- $P_m = 0.1$
- $\sigma_{share} = 0.5$
- $\sigma_{mate} = 0.3$

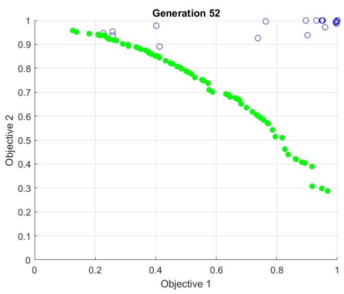

Figure 1: $P_c = 0.5$

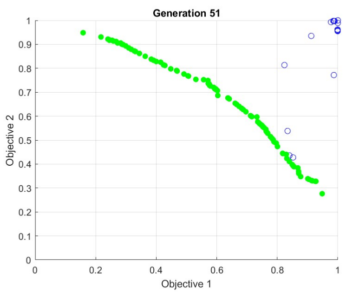

Figure 2: $P_c = 0.7$

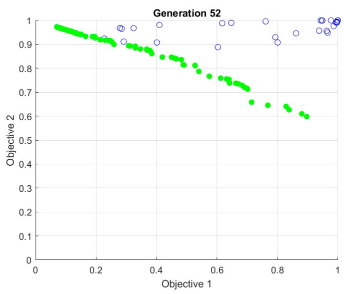

Figure 3: $P_c = 0.9$

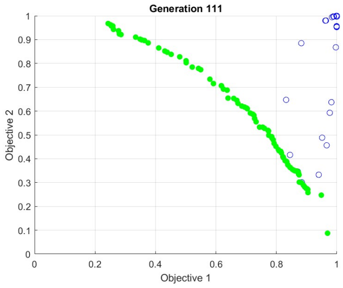

Figure 4: $P_m = 0.05$

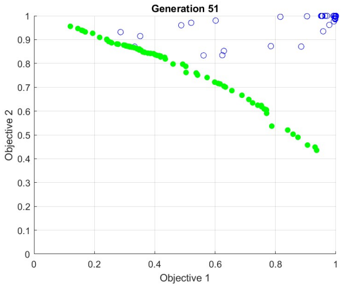

Figure 5: $P_1 = 0.1$

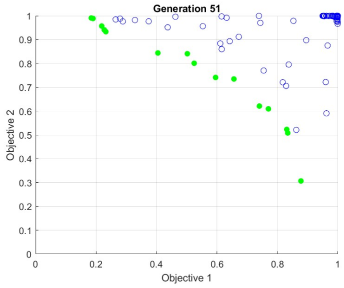

Figure 6: $P_c = 0.2$

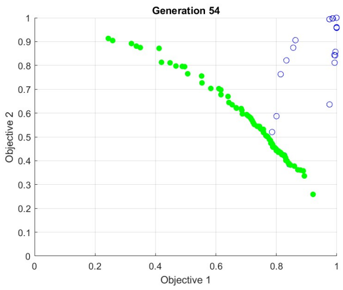

Figure 7: $\sigma_{share} = 0.1$

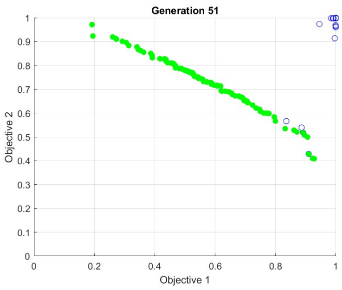

Figure 8: $\sigma_{share} = 0.3$

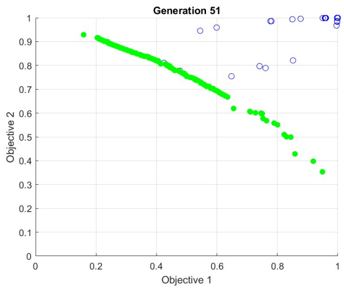

Figure 9: $\sigma_{share} = 0.5$

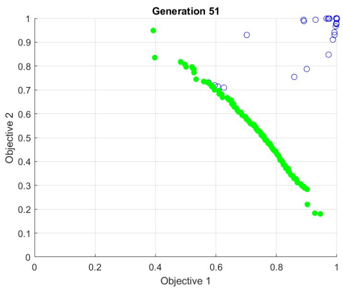

Figure 10: $\sigma_{mate} = 0.1$

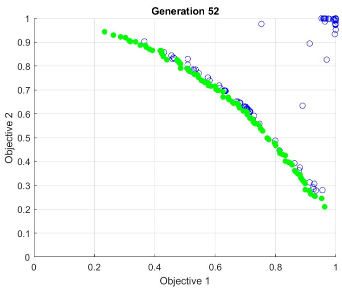

Figure 11: $\sigma_{mate} = 0.3$

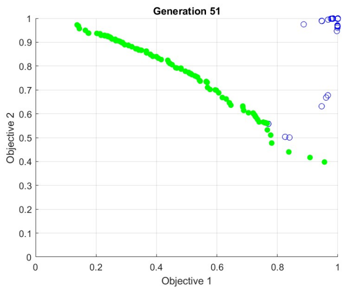

Figure 12: $\sigma_{mate} = 0.5$

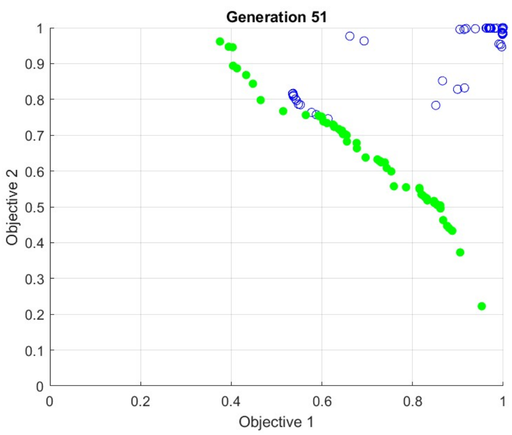

Figure 13: Mating and sharing restriction applied.

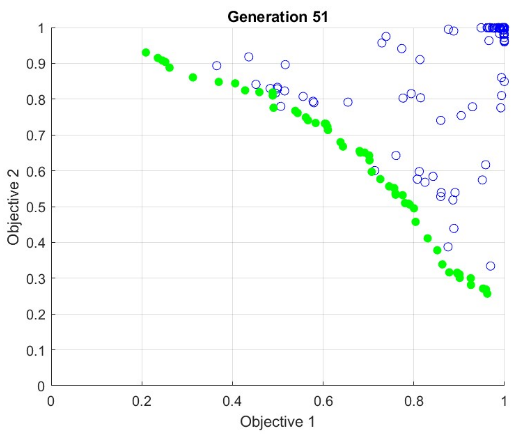

Figure 14: Neither mating nor sharing restriction applied.

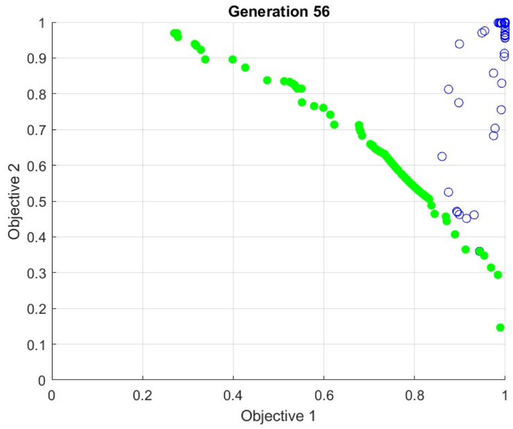

Figure 15: Sharing restriction applied but not mating restriction.

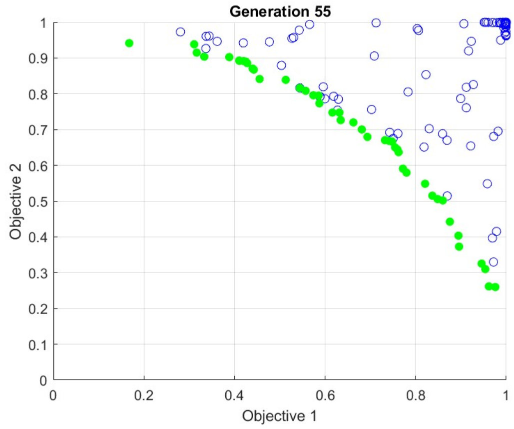

Figure 16: Mating restriction applied but not mating restriction.

FT function

We tested the following values of P_c, P_m, σ_{share} and σ_{mate}:

Table 2: Tested Values for Crossover Probability, Mutation Probability, Sharing Distance, and Mating Distance

Parameter	Tested Values
Crossover Probability (p_c)	0.5, 0.7, 0.9
Mutation Probability (p_m)	0.05, 0.1, 0.2
Sharing Distance (σ_{share})	0.1, 0.3, 0.5
Mating Distance (σ_m)	0.1, 0.3, 0.5

Figures 1 to 15 show the resulting graphs, through visualization we determined that the best values for each variable are:

- $P_c = 0.9$
- $P_m = 0.05$
- $\sigma_{share} = 0.3$
- $\sigma_{mate} = 0.3$

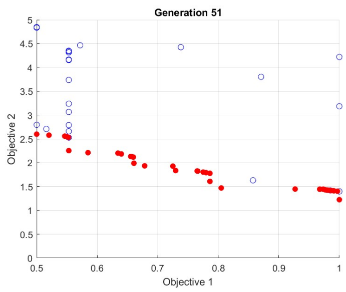

Figure 17: $P_c = 0.5$

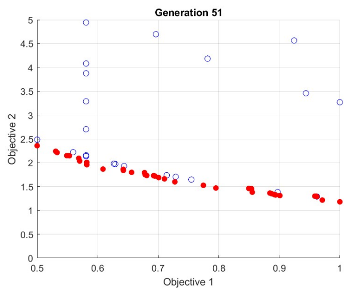

Figure 18: $P_c = 0.7$

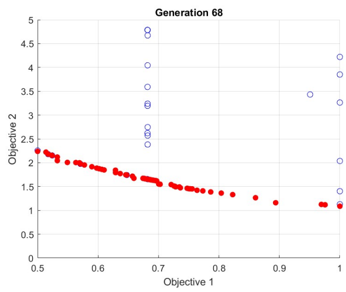

Figure 19: $P_c = 0.9$

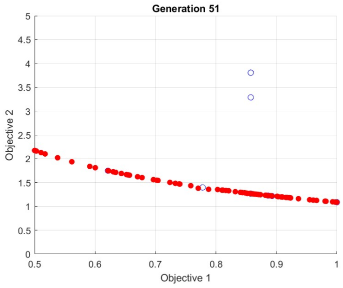

Figure 20: $P_m = 0.05$

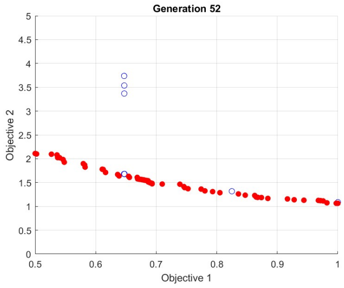

Figure 21: $P_1 = 0.1$

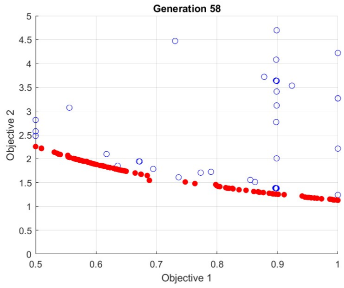

Figure 22: $P_c = 0.2$

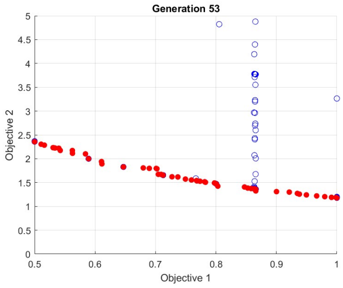

Figure 23: $\sigma_{share} = 0.1$

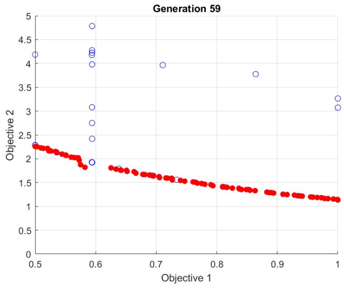

Figure 24: $\sigma_{share} = 0.3$

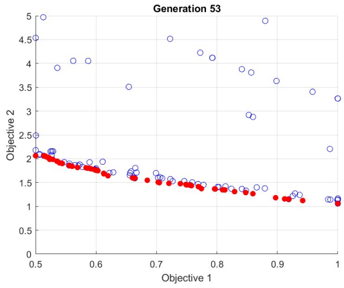

Figure 25: $\sigma_{share} = 0.5$

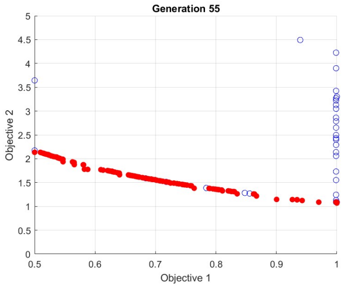

Figure 26: $\sigma_{mate} = 0.1$

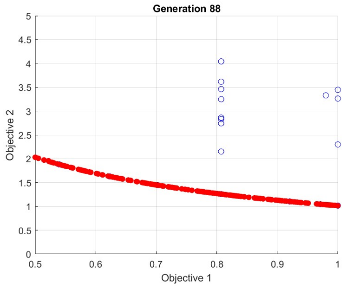

Figure 27: $\sigma_{mate} = 0.3$

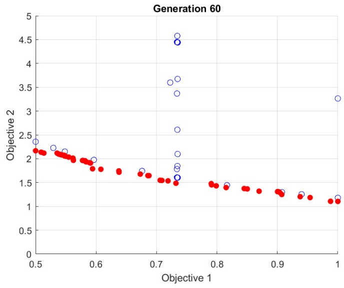

Figure 28: $\sigma_{mate} = 0.5$

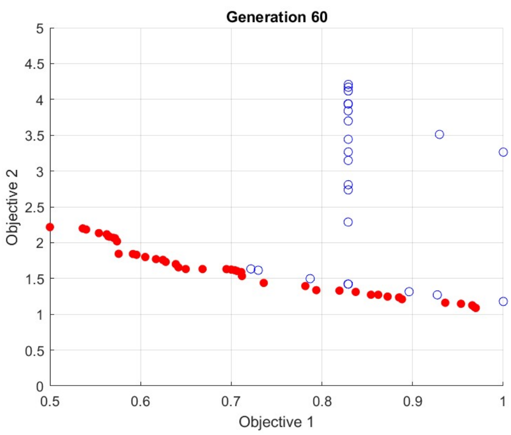

Figure 29: Mating and sharing restriction applied.

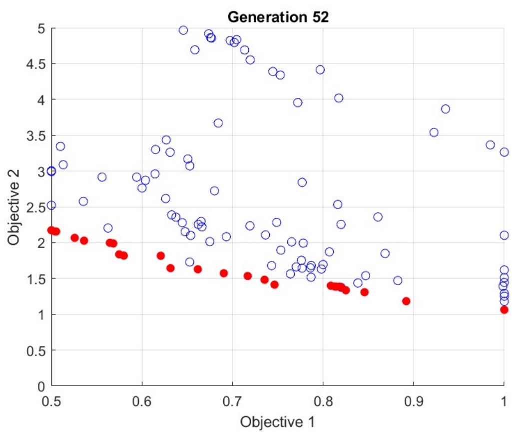

Figure 30: Neither mating nor sharing restriction applied.

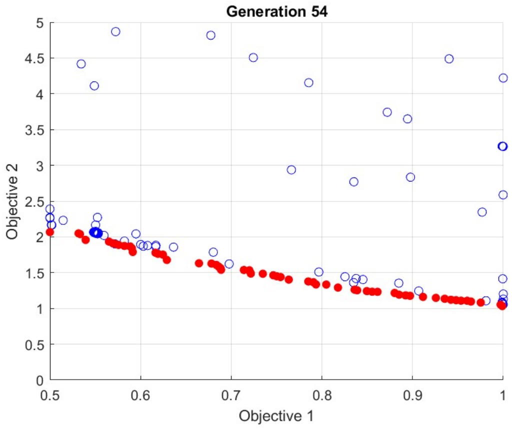

Figure 31: Sharing restriction applied but not mating restriction.

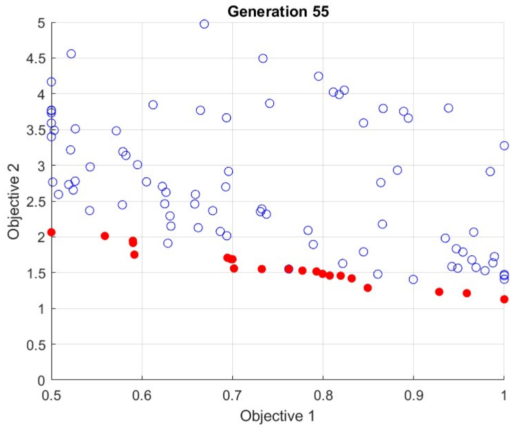

Figure 32: Mating restriction applied but not mating restriction.