0.1 Results and Discussion

Make chapter

This chapter includes presentation and analysis of the results obtained in this thesis. Section 0.2 presents results from testing different parameter settings for the genetic algorithm, and briefly discusses each results. Section 0.3 presents the main results obtained when running the different population distributed genetic algorithms. Section 0.4 contains a discussion and comparison of the results obtained in section 0.3.

0.2 Parameter settings

Parameter settings are crucial for obtaining good results with the genetic algorithm. In order to find the right settings, simulations were run to find the best adult selection method, parent selection method, crossover method, crossover rate and mutation rate for the given problem. Even though it would take much shorter time and less effort to test these settings on a toy problem such as One Max, the decision was made to test them on the real problem. The reason behind this decision is that different settings might work better on different problems and the nature of the wind farm layout optimization problem is unique because of the importance of the wind turbine positions relative to each other. It is important to note that even though effort was made to find good settings for the genetic algorithm, it is impossible to obtain the optimal ones. Just imagine trying the test every single value for the continuous parameter crossover rate against every possible value of each of the other settings, just that would be impossible! Therefore, the values that are tested for each parameter setting is largely tested against values that are based on the authors previous experience with genetic algorithm and values close to those.

For each simulation one of the parameters were tested while the others were kept fixed as shown in table 1. As can be seen in this table, other settings such as wind scenario, population size, number of generations, whether elitism should be used and mutation rate for interchange mutation and inversion mutation could also be tested, but since evaluation of each farm takes a lot of time, even on an 8 core computer running in parallel, simulations are

Table 1: Values kept fixed while one by one was changed in order to find the best settings for the wind farm layout optimization problem.

Parameter	**Value**
Wind scenario	00.xml
Evaluator	KusiakLayoutEvaluator
Population size	100
Generations	100
Elitism	true
Flip mutation rate	0.01
Inversion mutation rate	0.0
Interchange mutation rate	0.0
Parent selection	Tournament selection
Tournament size	5
Epsilon	0.0
Crossover method	Uniform crossover
Crossover rate	0.9

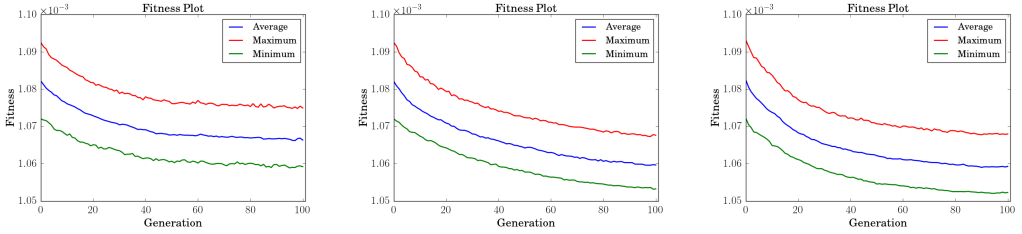
not run to set these values. These settings are set using the authors previous experience and educated guessing. Testing different parameters on a single wind scenario might lead to values that are tailored for the given scenario and that are not as well suited for some of the others. However, there is no time to test each value for every single wind scenario, and this should not make that much of a difference. Population size is a value that influence the performance of the genetic algorithm largely. Greater population size often leads to better performance since more solutions increase the probability of finding the global optimal solution. The population size was set to 100 for two reasons. First of all, a population size of 100 is large enough so that many different solutions are explored. Second when the population size is kept at 100, the adult selection method "Overproduction" will produce twice as many children, so that 200 individuals has to be evaluated for each generation something that will double the evaluation time, the step that already is the bottleneck of the algorithm. The number of generations was kept at 100 for each run, evaluating each population for 100 generations takes time and it the main reason why the algorithm were not run for more generations, however, as can be seen on the graphs in the sections below the fitness looks like it is starting to flat out after a 100 generations indicating that a 100 generations

are sufficient for the purpose of setting the parameter values. Elitism was set to true for every run without testing, meaning that the best individual of each generation will survive. This decision is based on the authors previous experience with genetic algorithm where experiments has shown that elitism leads to better results because the best individual is not lost due to coincidences. Different mutation methods were implemented for the genetic algorithm, but only the flip mutation rate was tested to find the best value. Usually, flip mutation is the only mutation method used with genetic algorithms and it is therefore also used as the main for of mutation in this thesis. Inversion mutation and interchange mutation are implemented, but they will only happen seldom to introduce more randomness to the algorithm. In the main simulations, they will be assigned extremely low probabilities so that their occurrence is so rare that they will introduce too much randomness, but hopefully make occasional "jumps" to different solutions spaces so that the algorithm do not get completely stuck in a local optimal solution.

0.2.1 Adult Selection

Figure **??** shows the results of running the genetic algorithm with the different adult selection methods. Figure **??**, **??**, and **??** shows the results for full generational replacement, generational mixing and overproduction respectively.

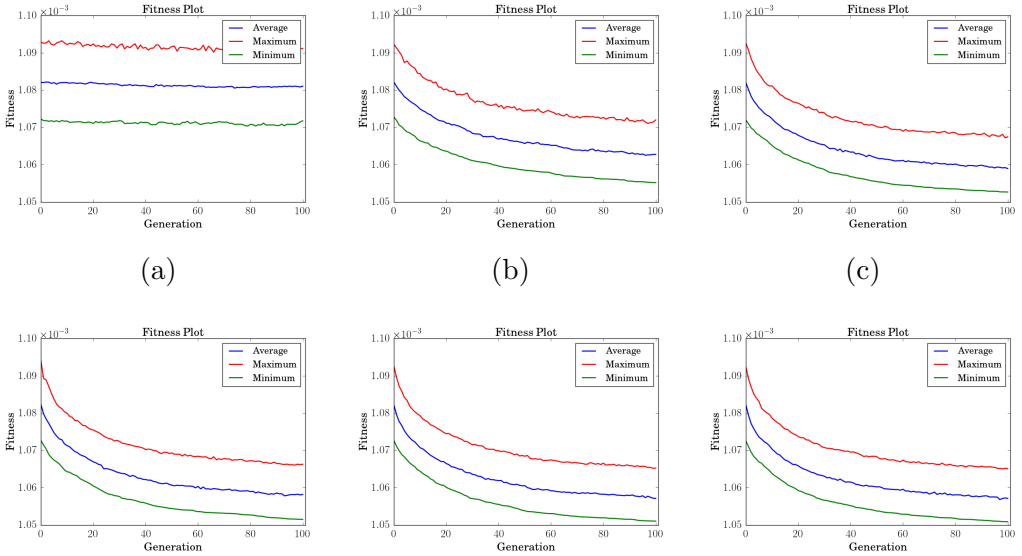
As can be seen in the figures, full generational replacement ends up with lower fitness than both generational mixing and overproduction, and overproduction ends up with best fitness. These results are as expected. Note that generational mixing could never do worse than full generational replacement. If the results of reproduction leads to a population of individuals with strictly better fitness, generational mixing will replace the entire previous generation with the new one, as full generational replacement. However, if the new generation produces individuals with worse fitness than some of the individuals in the previous generation these old individuals will be kept instead, making sure that the new population has equal or better fitness than the previous one. Overproduction does not provide the same "newer worst" Check word! guarantee, however, as shown in figure **??** it still outperforms generational mixing. Even though overproduction wipes out the entire previous population, it generates twice as many children at the reproduction step so its probability of getting it right is doubled.



(a) (b) (c)

Figure 1: Adult selection methods: (a) Full generational replacement, (b) generational mixing and (c) overproduction averaged over 10 runs.

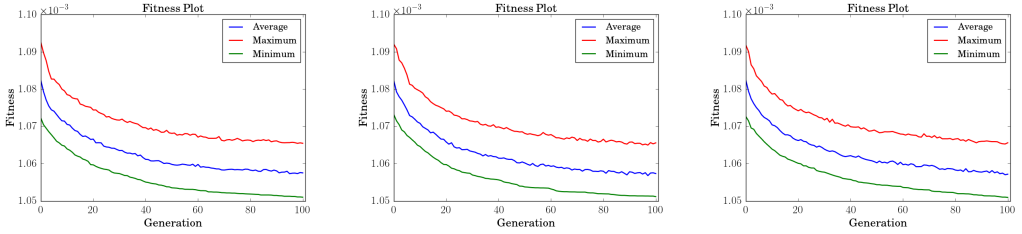
0.2.2 Parent Selection



(a) (b) (c)

(d) (e) (f)

Figure 2: Parent selection methods: (a) Roulette wheel, (b) Tournament selection averaged over 10 runs, tournament size 5 (c) tournament selection, tournament size 10, (d) tournament selection, tournament size 15, (e) tournament selection, tournaments size 20, and (f) tournament selection, tournament size 25.



(a) (b) (c)

Figure 3: Different epsilon values averaged 10 runs when tournament size was kept at 20: (a) Epsilon 0.05, (b) epsilon 0.10 and (c) epsilon 0.15.

0.2.3 Crossover Methods

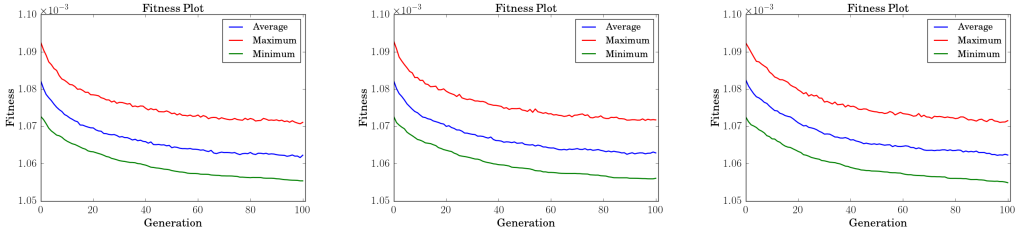
Figure **??** displays the results from running the genetic algorithm with single point crossover, two point crossover and uniform crossover showed in sub figures **??**, **??** and **??** respectively. As can be seen in the figure, there is no crossover method that stands out, but end up with similar fitness. Single point crossover ends up with an averaged best fitness of averaged best fitness, two point crossover ends up with an averaged fitness of averaged best fitness and uniform crossover ends up with an averaged fitness of averaged best fitness. Intuitively, one would expect uniform crossover to perform worse than the others because it mixes up the relative positions between the wind turbines more than single- and two point crossover, however it actually obtains slightly better fitness. Even though it obtains slightly better fitness this could be a coincidence since the different is so small, and the simulations are only averaged over ten runs. Explanation behind the results, talk to Keith. Stupid paragraph change everything!

0.2.4 Crossover Rate

0.2.5 Mutation Rate

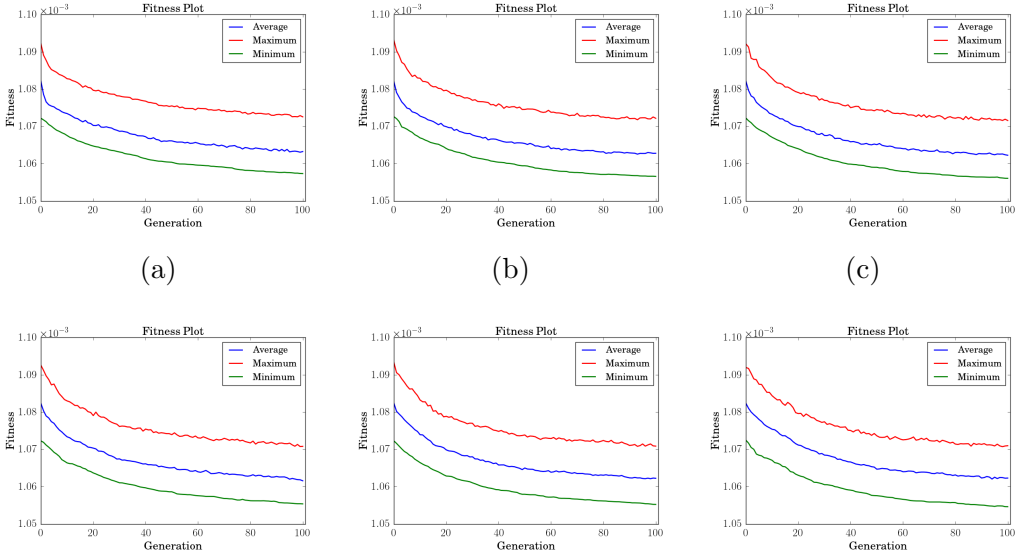
0.3 Results

In table 2 the parameter values used when running the master slave model is shown. The crossover method, crossover rate, mutation rate, adult selection



(a) (b) (c)

Figure 4: Crossover methods averaged over 10 runs: (a) Single point crossover, (b) two point crossover and (c) uniform crossover.



(a) (b) (c)

(d) (e) (f)

Figure 5: Crossover rates: (a) Crossover rate 0.0, (b) crossover rate 0.2, (c) crossover rate 0.4, (d) crossover rate 0.6, (e) crossover rate 0.8, and (f) crossover rate 1.0.

mechanism, parent selection mechanism and parent selection parameters are those that proved to work best for the wind farm layout optimization problem as shown in the previous section. The population size was set to 100 and the number of generations to 200. A population size of 100 is quite small,

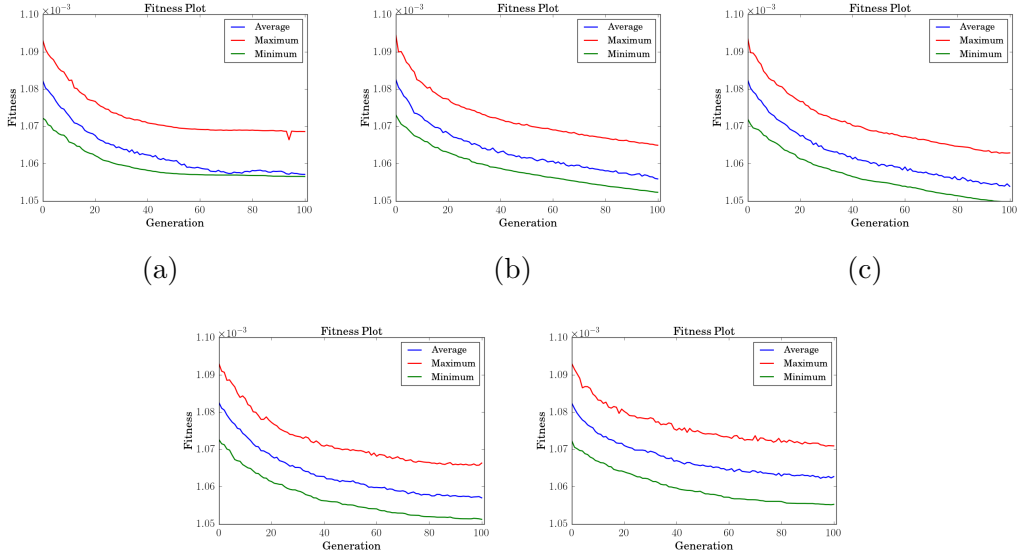


Figure 6: Mutation rates: (a) Mutation rate 0.0001, (b) mutation rate 0.0005 and (c) mutation rate 0.001, (d) mutation rate 0.005 and (e) mutation rate 0.01s.

but since overproduction is the selected adult selection method a population size of 100 leads to evaluation of 200 individuals each generation. Since evaluation is the bottle neck, by far, for wind farm layout optimization a larger population size would not be possible to evaluate for 200 generations. A possibility could be to pick a larger population size and a smaller number of generations, but since the objective of this thesis is to explore population distributed genetic algorithms a large number of generations is crucial since population distributed genetic algorithms need more time to find good solutions since they explore different search spaces.

Table 3 shows the parameters from the Island model that differs from the ones in table 2. The deme size, number of individuals on each Island, is set to 26 resulting in a total population size of 104, not 100 as in table 2. The reason behind this is that the implementation requires a population size of even numbers. The deme count, number of Islands, is set to four and the topology is circular as shown in figure (Make topology figure). In order to

Table 2: Parameter values used for the master slave model.

Parameter	**Value**
Population size	100
Generations	200
Crossover method	Single point crossover
Crossover rate	0.9
Elitism	True
Flip mutation rate	0.001
Inversion mutation rate	0.000001
Interchange mutation rate	0.000001
Adult selection mechanism	Overproduction
Parent selection mechanism	Tournament selection
Tournament size	20% of population size
Epsilon	0.1

Table 3: Parameter values used for the Island model.

Parameter	**Value**
Deme size	26
Total population size	104
Deme count	4
Migration rate	2
Number of migrations	10
Migration interval	20
Topology	Circular reference figure

let the populations on the different Islands explore different search spaces 20 generations are run before migration is performed. Migration is performed 10 times so that the total number of generations becomes 200 as for the master slave model.

As can be seen in table 5 the cellular model has a population size of 225. At a first glance this might seem like an odd decision since the models above has a population size of about 100 individuals, but the reason is simple. Since overproduction is the adult selection mechanism used for the models above 200 individuals are evaluated for each generation, however, overproduction is not used with the cellular model because each individual has its own position

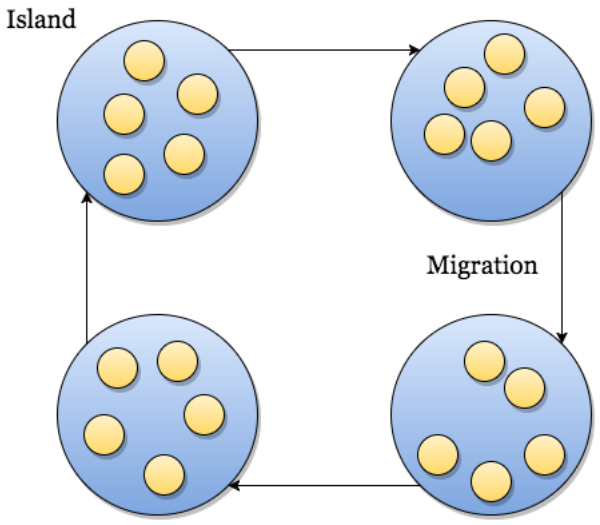


Figure 7: Island model topology. Individuals are only allowed to migrate in the directions indicated by the arrows.

Table 4: Parameter values used for the cellular model.

Parameter	**Value**
Population size	225
Topology	reference figure

in the grid so that growing and shrinking the population at different stages of the genetic algorithm does not make sense. By using a population size of about 200 the cellular model get approximately the same processor time as the algorithms above, and therefore the comparison becomes more fair than with a population size of 100. The reason why the population size is 225 and not 200 is because a quadratic grid is used to distribute the production and 225 is equal to 15^2. As can be seen in figure 8, the topology used is a simple square containing nine individuals, the individual in the middle square can only be replaced by individuals generated by recombining individuals within the square grid. The decision to make the neighborhood this small is that it will give different solutions the opportunity to dominate different areas of the grid, giving the algorithm the time to explore different solution spaces.

Table 5 shows the parameter values used by the pool model that differs from

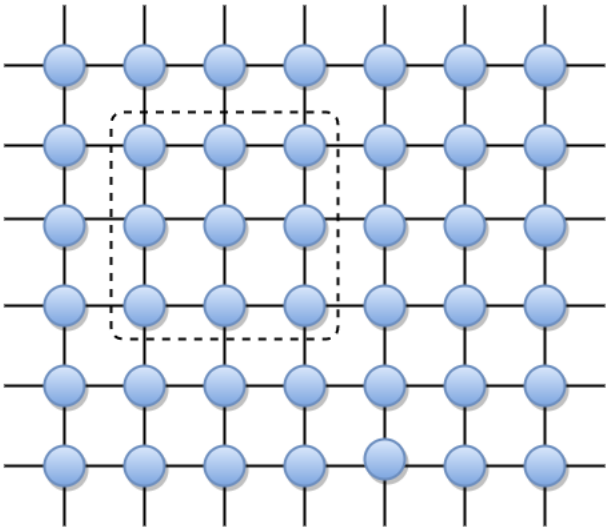


Figure 8: Cellular model topology. Only individuals within the marked square are allowed to become parents for the middle cell.

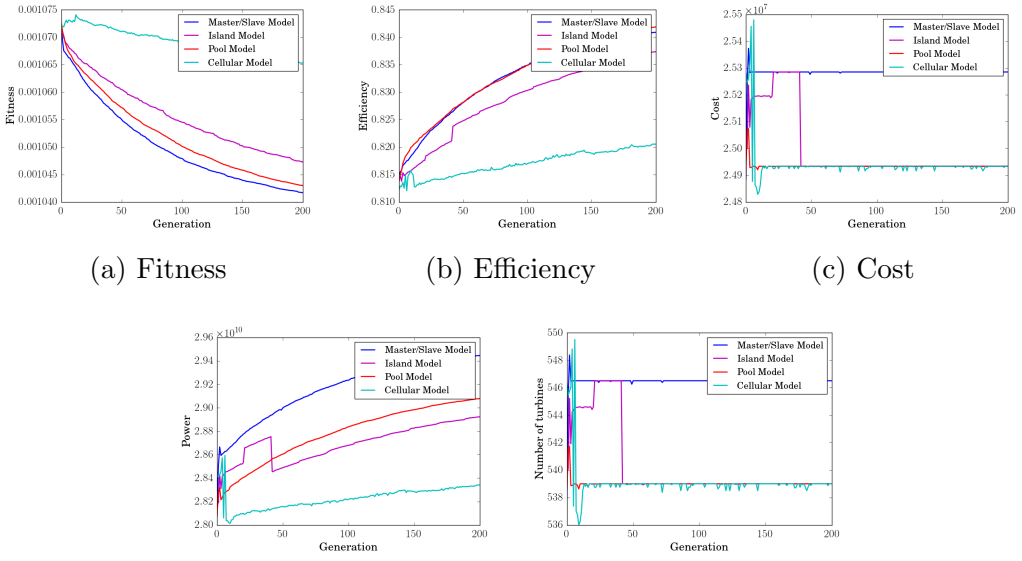
Table 5: Parameter values used for the pool model.

Parameter	**Value**
Population size	200
Number of workers	4

those in table 2. As for the cellular model, the population size is set to 200 so that 200 evaluations are performed for each of the 200 generations. Since child generation for the pool model consist of producing one individual for each position which the given thread is responsible for it makes no sense to use overproduction, therefore a population size of 200 gives the algorithm as much processor time as the three others.

Write about evaluation parameters. Fix graphs and display results.

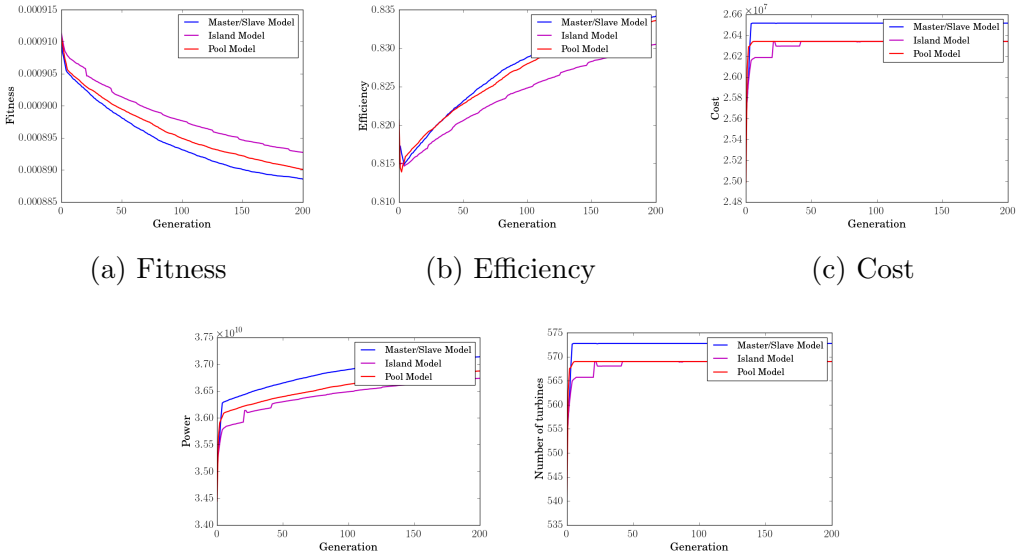
0.4 Discussion



(a) Fitness (b) Efficiency (c) Cost

(d) Power (e) Turbies

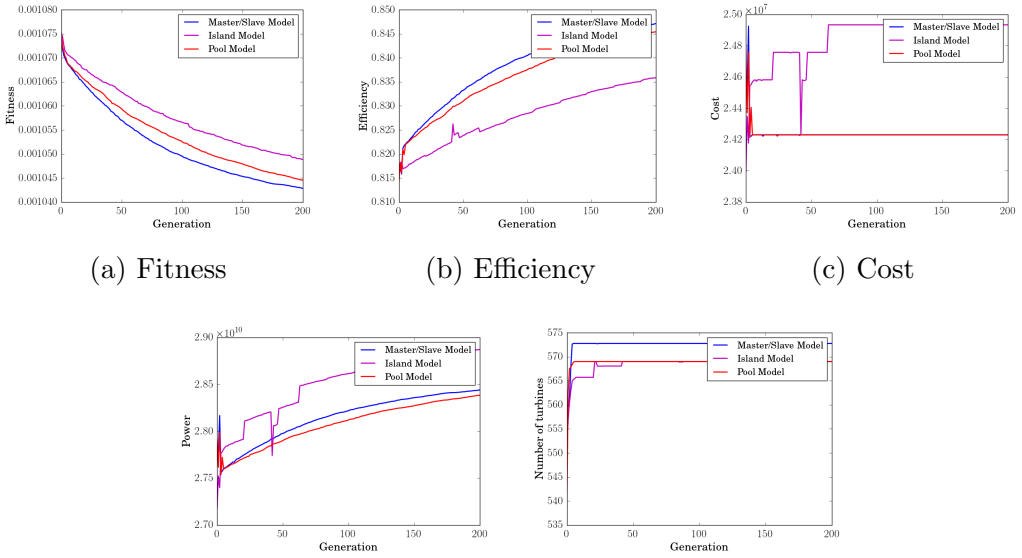
Figure 9: Scenario 00.xml averaged over 10 runs: (a) Fitness plot, (b) efficiency plot, (c) cost plot, (d) power plot, and (e) number of turbines.



(a) Fitness (b) Efficiency (c) Cost

(d) Power (e) Turbies

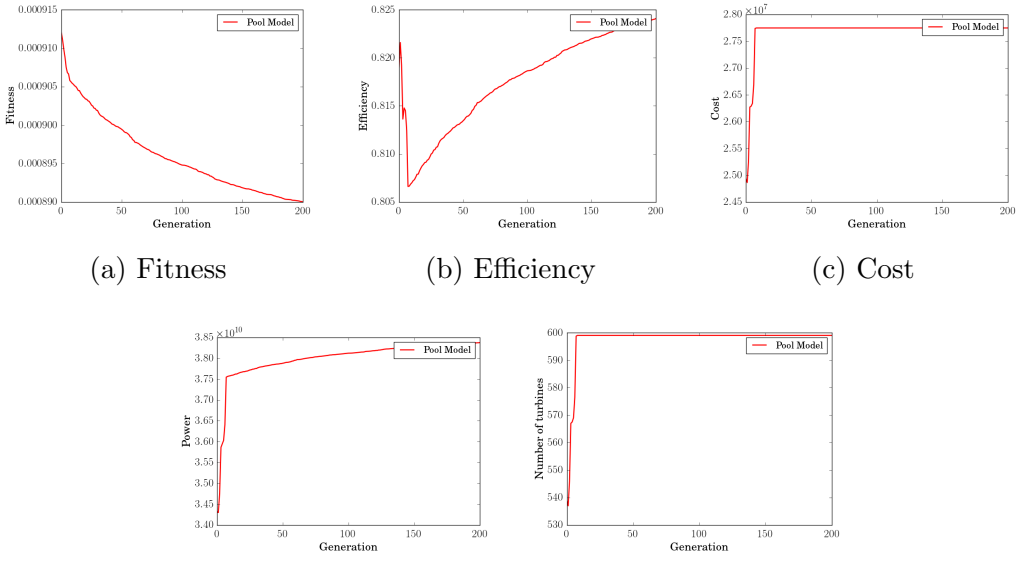
Figure 10: Scenario 05.xml averaged over 10 runs: (a) Fitness plot, (b) efficiency plot, (c) cost plot, (d) power plot, and (e) number of turbines.



(a) Fitness (b) Efficiency (c) Cost

(d) Power (e) Turbies

Figure 11: Scenario obs00.xml averaged over 10 runs: (a) Fitness plot, (b) efficiency plot, (c) cost plot, (d) power plot, and (e) number of turbines.



(a) Fitness (b) Efficiency (c) Cost

(d) Power (e) Turbies

Figure 12: Scenario obs05.xml averaged over 10 runs: (a) Fitness plot, (b) efficiency plot, (c) cost plot, (d) power plot, and (e) number of turbines.