# The Exam Scheduling Problem

- Solve exam scheduling problem using a genetic algorithm, hillclimbing, random search, and a variant of simulated annealing (mutate search)
- Compare how each algorithm performs in small/medium/large data sets.
- Development tools: java and mathematica

# Fitness function

- We need a fitness function to determine how good a schedule is.
- the fitness of a schedule means how much each student or room likes their given timetable.
- For both a student and a room, the fitness function is $$q(TT) = \frac{1}{1 + penalty_{TT}}$$
- Runtime: a single evaluation of the fitness function took 0.2 seconds for a large data set.

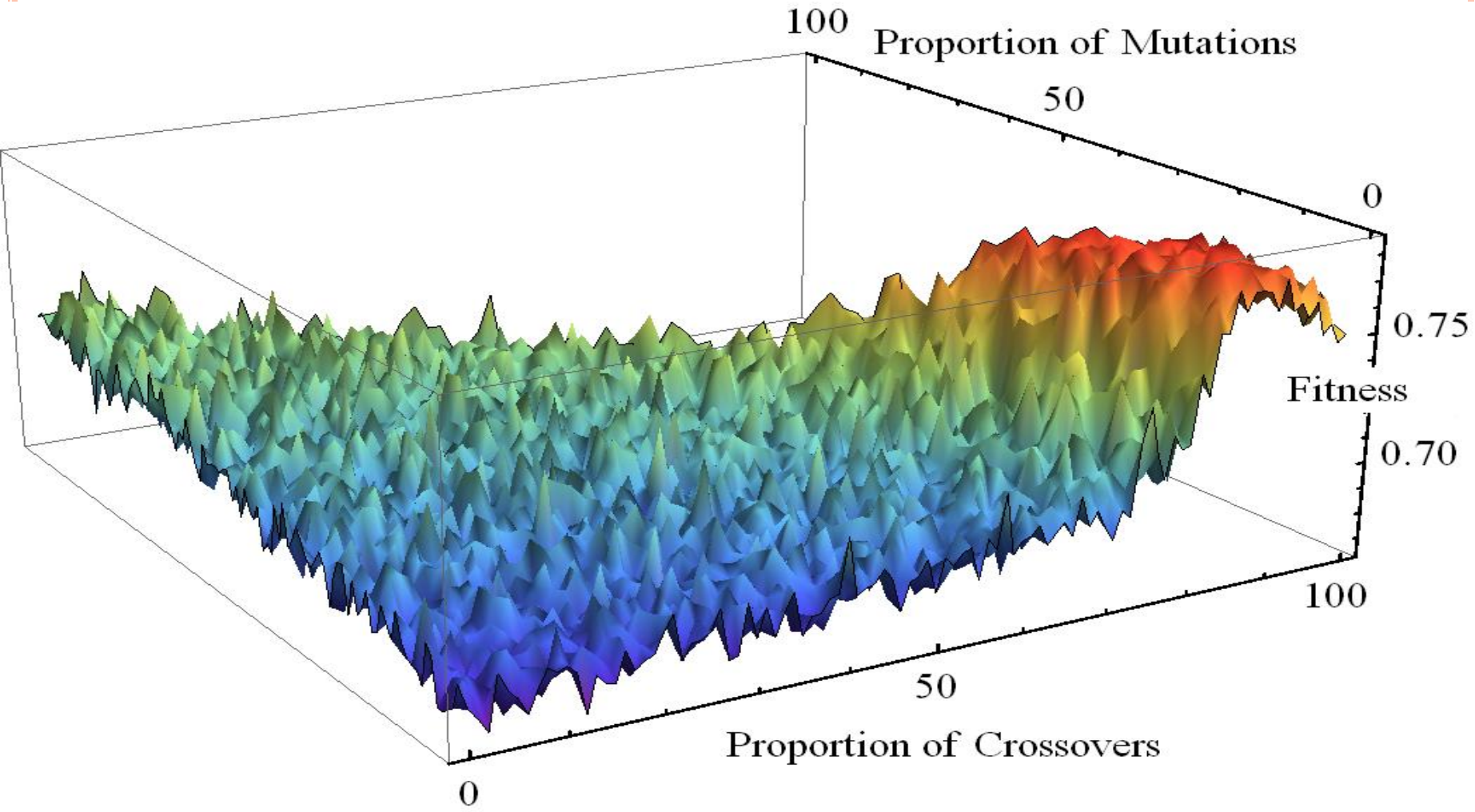
# Genetic Algorithm

- It maintains a population of schedules and constructs subsequent generations
- To generate a new schedule: random, copying, mutation, crossover
- Mutation/crossover: schedules are selected with a probability proportional to their fitness.
- free parameters: size of the population, number of generations, proportions of copy, random, mutation, and crossover

# Other search algorithms

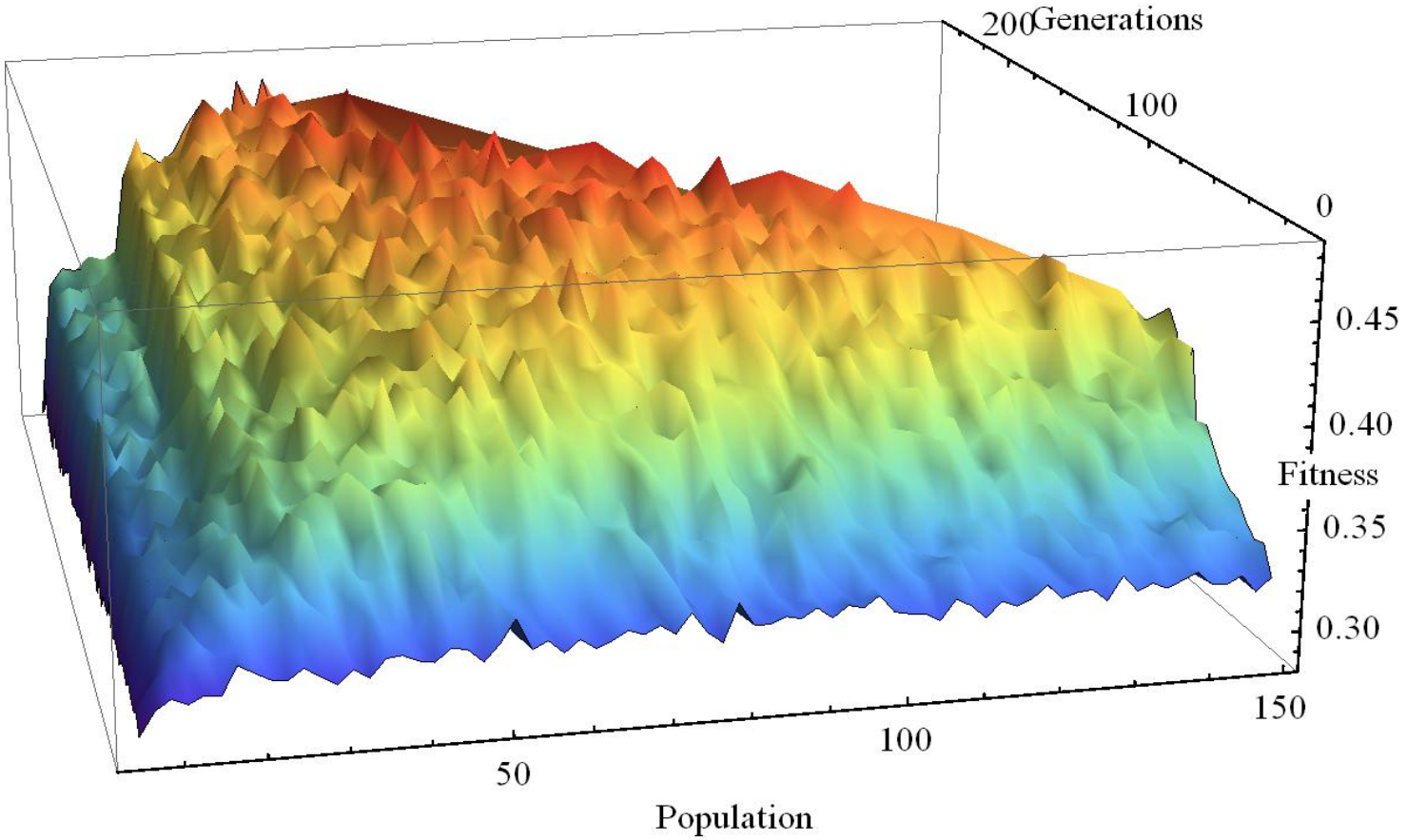
- Random Search: randomly generate some schedules, return the one with highest fitness.
- Mutate-search: Make 1% changes to the current best one and check the fitness. If the new one is better, replace the old one.
- Hill Climbing: examines every schedule in the neighborhood of a given schedule, then continues from the best one, until it reaches the maximum number of evaluation allowed.

# Generation Parameters for GA




Crossover is outperforming!

# Size Parameters for GA




the population per generation and the number of generation seem to both influence as much the fitness.

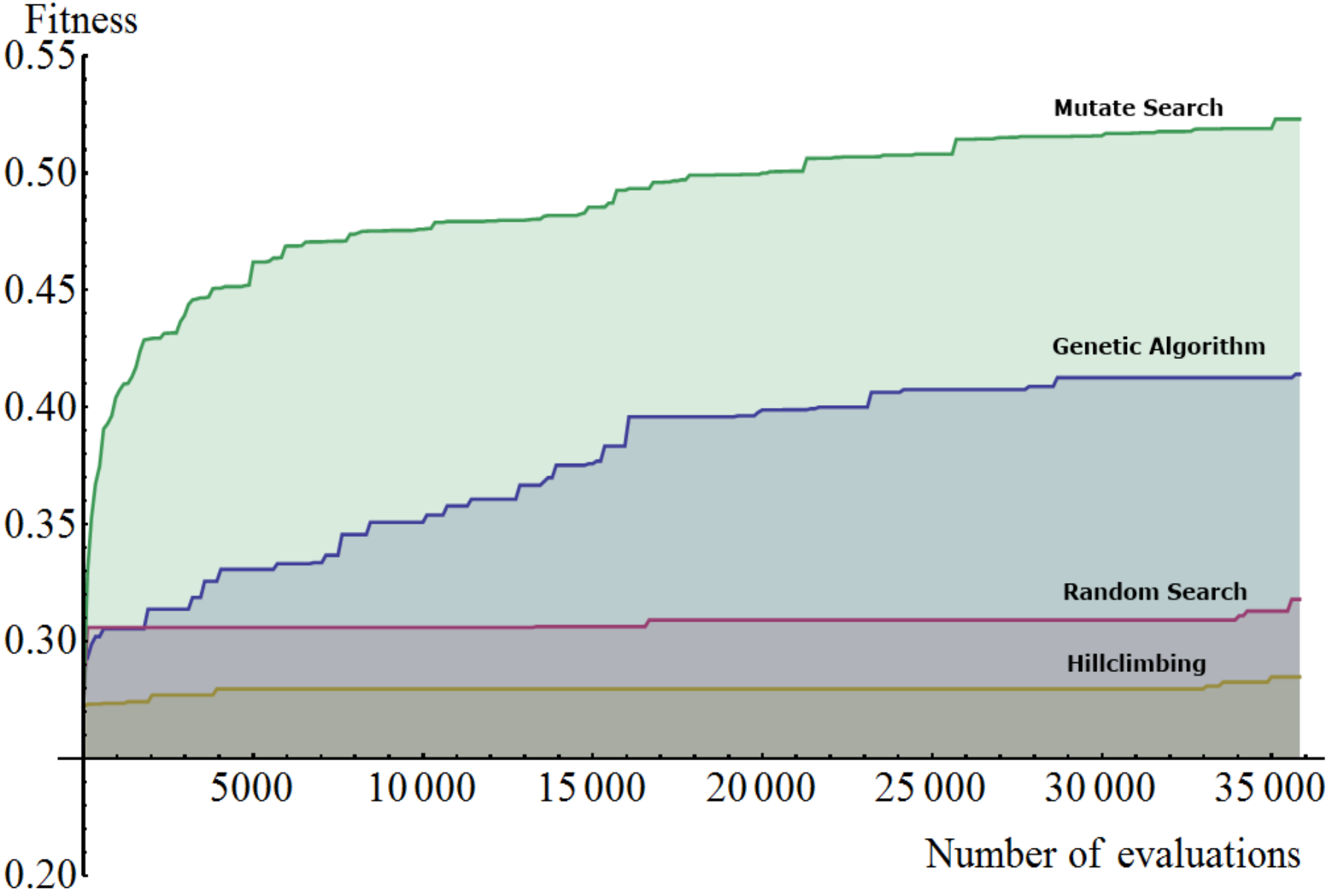
# How to test the algorithms?

- We would like to compare data sets which their courses are with and without correlation.
- We want to see how these algorithms perform with small/medium/large data sets.
- Thus we have 2 sets of 3 differently sized scheduling problem instances to test our algorithms.
- The large data set is actual U of T data.

# Testing data

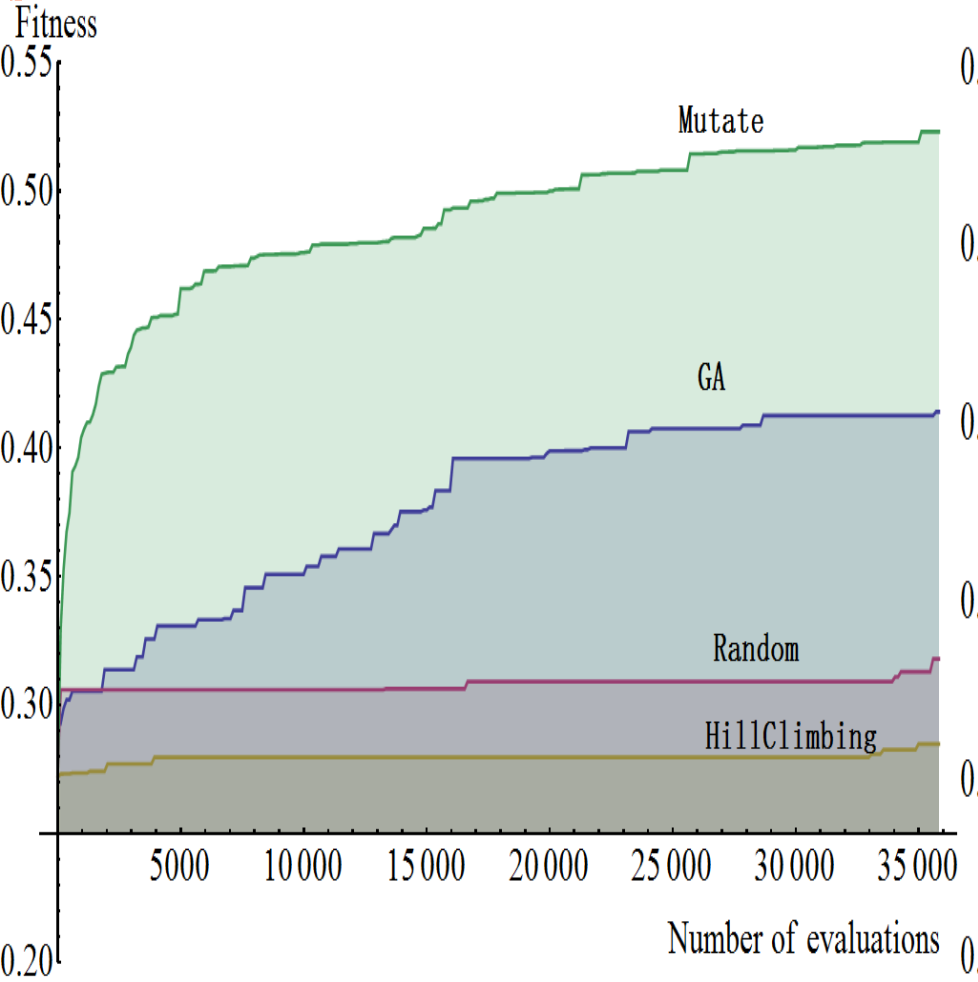
| | Small | Medium | Large | Small w/ Correlations | Medium w/ Correlations | Large w/ Correlations |
|---|---|---|---|---|---|---|
| Days | 7 | 10 | 7 | 7 | 10 | 7 |
| Time slots | 5 | 8 | 8 | 5 | 8 | 8 |
| Courses | 20 | 200 | 603 | 20 | 200 | 600 |
| Rooms | 4 | 10 | 43 | 4 | 10 | 43 |
| Students | 50 | 300 | 21945 | 50 | 300 | 21945 |
| Courses taking | 1-5 | 1-6 | 1-5 | 2-6 | 2-6 | 2-6 |
| Number of Majors | N/A | N/A | N/A | 4 | 10 | 20 |
| Search space size | $10^{182}$ | $10^{1840}$ | $10^{6695}$ | $10^{182}$ | $10^{1840}$ | $10^{6689}$ |

# Comparing Algorithms (large data)

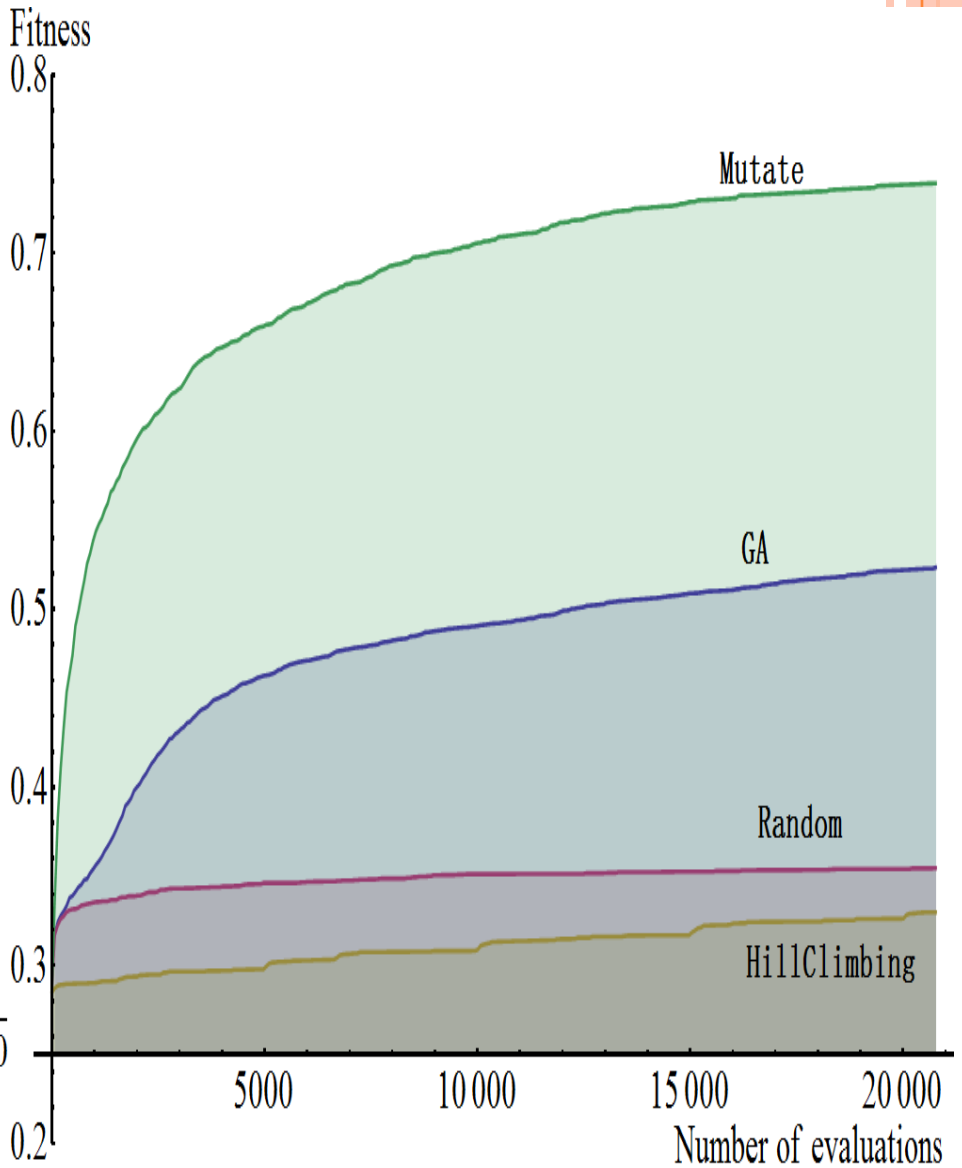


# Comparing Sizes (medium vs Large)

Large with correlation:

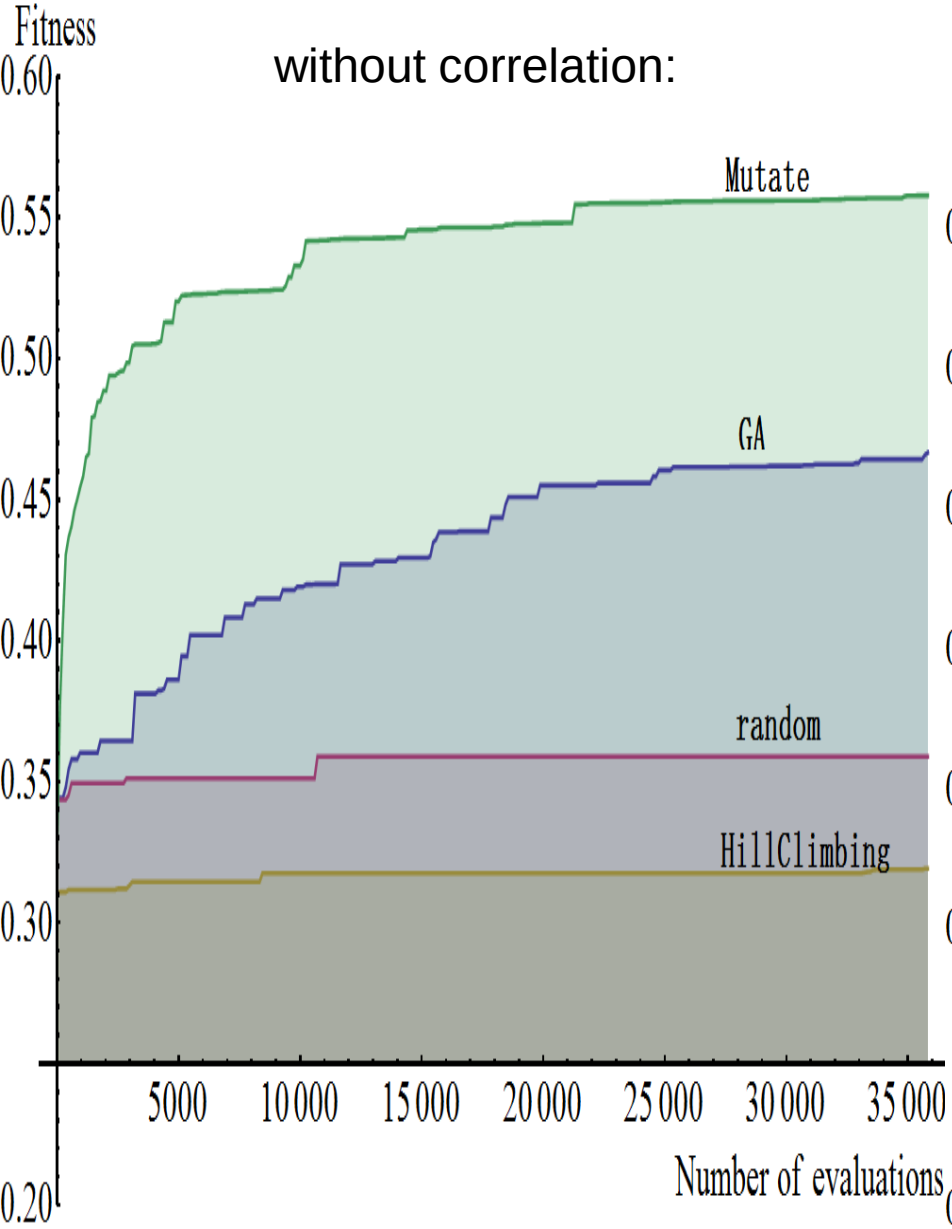



Medium with correlation:

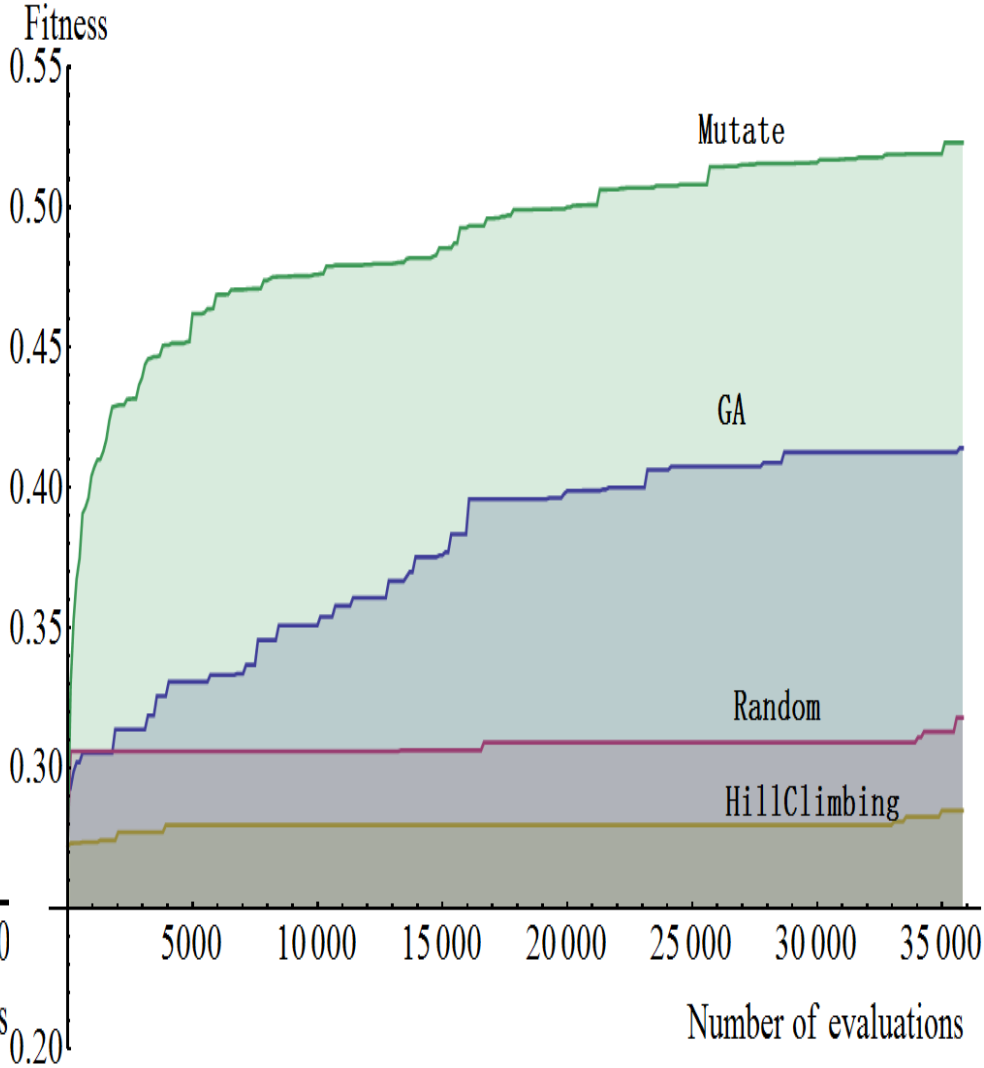


# WITH VS WITHOUT CORRELATION (LARGE)

without correlation:




with correlation:



# Conclusions

- Hillclimbing is not so valuable for this problem, if the data gets big, it will get stuck at a low local maxima.
- GA solves the problem pretty decently, but simulated annealing (mutate search) performs better.
- Searches perform better on data without course correlations than on those with correlations.