Comparing the performance of a genetic algorithm against a population of hillclimbers

S. Kuznetsov

April 04, 2022

Abstract

Evolutionary algorithms are a type of optimization algorithms inspired by natural selection principles. They use populations of solutions and gradually evolve and combine them until an optimal solution is found. The quality of the solution is determined by the fitness function, where the maximum fitness is usually unknown, and the search for the optimal solution is carried out by gradually exploring the fitness landscape and trying to find the highest point. In this report, we explore Hillclimbers and Genetic algorithms, and evaluate their performance on a simple optimization task. We show that both algorithms are efficient, but Hillclimbers are more likely to get stuck in a local optimum and perform better with fewer iterations of the algorithm, while GAs converge more slowly but achieve better results in the long run.

1 Introduction

Evolutionary algorithms are a set of optimization algorithms inspired by natural selection that follow three properties: Selection, Variation and Heredity. Selection means that new better individuals replace old ones, Variability means that new individuals will differ to some extent, and Heredity implies that new individuals retain some of the properties of their parents.

Optimization itself is the process of finding the optimal point in the search space. Genetic algorithms and Hillclimbers can be applied for many various types of problems, but for this case we will consider the **zero-one knapsack problem**: given a set of items with corresponding values and weights, determine the number of each item to include in a knapsack, so that the total weight is less than or equal to the given limit and the total value is maximized as much as possible. This can be formulated as follows:

$$\text{maximize } \sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{subject to } \sum_{i=1}^{n} r_i x_i \leq b_i \tag{2}$$

$$x \in \{0, 1\} \tag{3}$$

We will consider a single genotype as a potential solution to the problem. The genotypes are encoded as binary strings or *chromosomes* and each individual is made up of the same number of genes, in our case 50. This genotype can then be mutated and recombined with other individuals to obtain a better solution at each iteration of the algorithm. In this report we will examine the results of running both types of algorithms on a given problem and demonstrate the comparison between them. Finally, we will discuss what further optimization can be applied to achieve better results for both types of algorithms.

2 Hillclimbing

Hillclimbing is a common method for solving optimization problems. The goal of a Hillclimbing algorithm is to find the maximum of a given objective function starting with a sub-optimal random solution and slowly improving itself until the optimal solution is found. By taking small steps in the search space it attempts to find the point where the function has its maximum fitness. In this section, we will discuss how Hillclimber can be applied to the knapsack problem, what problems can arise, and what are some potential ways to avoid them.

2.1 Single Hillclimber

The value of the next possible location for a single Hillclimber is evaluated by the fitness function $Fit(g)$ and the terminating point of an algorithm is represented by a given number of generations. The mutation function $Mut(g)$ is where the decision is made whether to move or stay in the same place in the landscape (4). This can be considered as a greedy approach, with the only difference being that backtracking is allowed. For the knapsack problem, the mutation function will update the genotype and make a step in the search space only if the new fitness is greater and the new weight does not exceed the given weight limit.

$$Mut(g) = \begin{cases} Fit_{t+1}(g) & \text{if} \quad Fit_{t+1}(g) > Fit_t(g) \\ Fit_t(g) & \text{otherwise} \end{cases} \tag{4}$$

Because Hillclimber's starting location is initialized randomly, this makes it a local search problem, making it prone to getting stuck in local optima. The modality(i.e number of local optima) of a fitness landscape is determined by the complexity of the function used which can be very high. It should be noted that the optimization process can be complex even for unimodal functions that have only one local optimum. The problem may occur when function forms an isolated peak on otherwise flat landscape. For this reason a multimodal problem can sometimes be more easily solved than a unimodal [1]. In knapsack case, we are dealing with a multimodal function so there is a certain high risk for a single Hillclimber to arrive at the local optima.

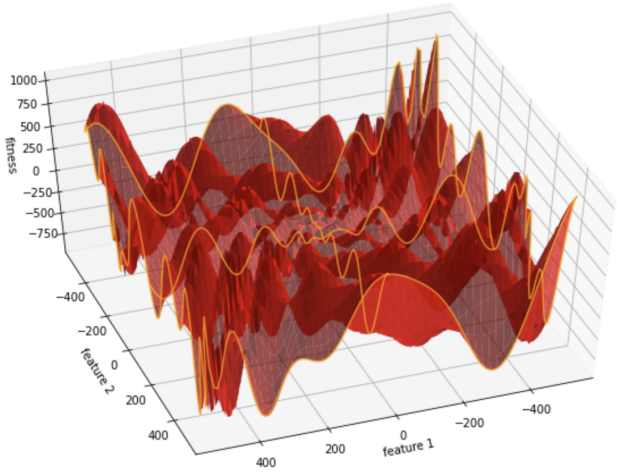

Figure 1: Fitness Landscape formed by the Eggholder function

2.2 Population of Hillclimbers

One possible solution to the problem of local optimal convergence is to initialize several Hillclimbers at different locations in the landscape, which together will form a population. These Hillclimbers can then try to solve the given problem simultaneously, slowly moving towards the best solution from different locations in the landscape. Fig. 5 shows how the fitness of three different populations of 20 Hillclimbers changes over time.

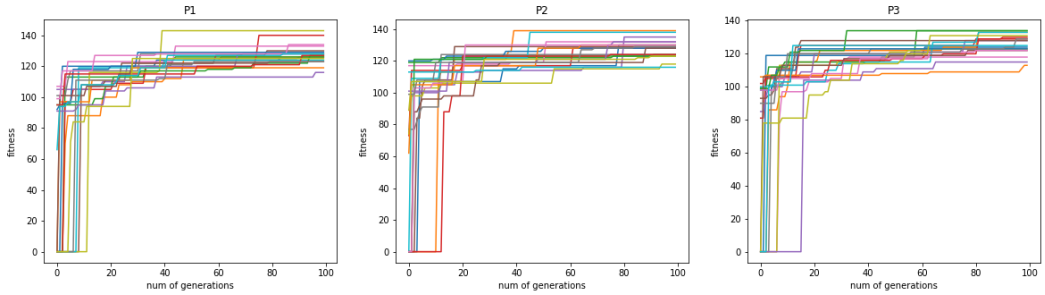

Figure 2: Population of hillclimbers

If we consider the best individual from each population, we can see that two individuals from different populations achieve roughly the same fitness, and the best individual from the third population ends up with a slightly lower result.

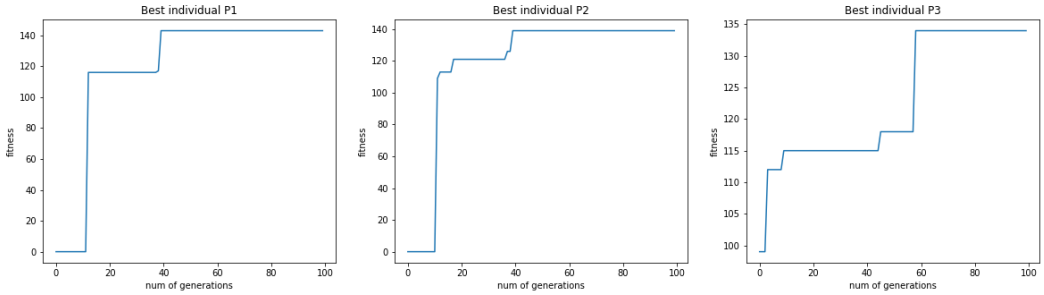

Figure 3: Best individual in the population

Although this approach may result in a relatively good fitness, it still does not guarantee that local optima is avoided. Due to selection, population diversity decreases over time and Hillclimbers coming from different locations may end up on the the same local optima.

Algorithm 1 Population of Hillclimbers

Generate population of genotypes P
N ← *number of individuals in P*
M ← *number of generations*
FOR j=1 to M:
  FOR i=1 to N:
    $I_{new} \leftarrow Mut(P[i])$
    $P[i] \leftarrow I_{new}$
  END FOR
END FOR

3 Genetic Algorithms

A Genetic Algorithm is a type of evolutionary algorithm which can be seen as a more "intelligent" and biologically plausible interpretation of Darwinian natural selection. Similarly as with the population of hillclimbers, each individual is encoded as sequence of genes which represents a single possible solution to the problem. After that two individuals from the population of genotypes are selected and their fitnesses are compared. The offspring of two chosen individuals is the result if their genes recombination. This process is called *tournament selection* and is repeated until a certain satisfaction condition is reached, so that the population gradually evolves as the number of tournaments get bigger. Each new offspring generated replaces the "weakest" parent, which broadly mimics natural selection.

The process of preserving the genetic code of well-performing individuals and excluding poorly-performing individuals from the population is called "elitism". Algorithm maintains stochasticity by randomly selecting the first individual from the population. For more complex tasks, *non-linear* selection methods can be introduced and the number of individuals participating in one tournament can be increased. However, *microbial* tournament selection with a minimum size of individuals is preferred for a minimalist GA [2]. The process of tournament selection can be described as follows:

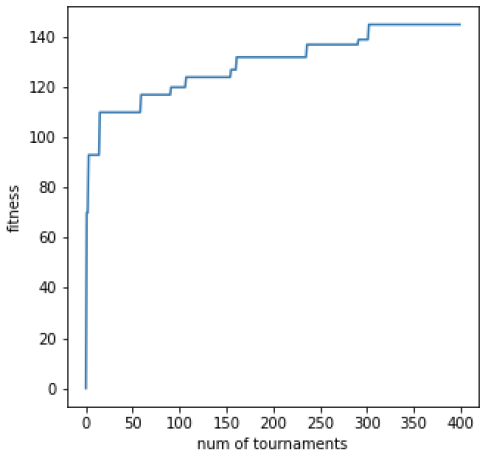

Figure 4: Example of a GA optimization

The main privilege of a GA over a Hillclimber is the ability to get out the local optima by using mutation together with crossover. This enhances the recombinative effect of GA and is the main factor that makes it so powerful [3]. The crossover function $Crs(g_1, g_2)$ receives two genotypes and replaces the genes of the second one with the genes of the first with a certain probability. As a result, we get an offspring that replaces the weakest parent in the population, and the strongest parent remains unchanged. In my experiments we used crossover rate of 0.5, but this can be adjusted depending on the type of problem and the design of the algorithm.

Algorithm 2 Microbial Tournament Selection

Generate population of genotypes P
$M \leftarrow$ *number of tournaments*
FOR i=1 to M:
  Evaluate fitness of each genotype in P
  Pick first individual I_1 *from* P *at random position* x_i
  Pick second individual I_2 *from* P *in range* $(x_i + 1, x_i + k)$
  IF $Fit(I_1) > Fit(I_2)$:
    $Winner \leftarrow I_1$
    $Loser \leftarrow I_2$
  ELSE:
    $Winner \leftarrow I_2$
    $Loser \leftarrow I_1$
  END IF
  $I_{new} \leftarrow Crs(Winner, Loser)$
  $I_{new} \leftarrow Mut(I_{new})$
  Insert I_{new} *back into* P
END FOR

The genetic algorithm is metaheuristic and cannot guarantee that the best solution will be found because it does not explore the entire problem landscape, which can often be very computationally inefficient. It is still prone to local optima although to a lesser extent than a hillclimber. GAs usually require tuning since their performance is greatly affected by the settings of a crossover and mutation rates as well as choosing the right technique for the stochastic sampling. It is also important to add that the same GA can be coded in different ways which will can considerably affect the performance [4]. However it is considered a general optimization technique for a range of domains because of its robustness and relative ability to avoid local optimum.

4 Experimental Results

One way to see how two algorithms behave in relation to each other is to start optimization from zero values for all genes. This will eliminate the randomness that occurs when initializing genotypes.

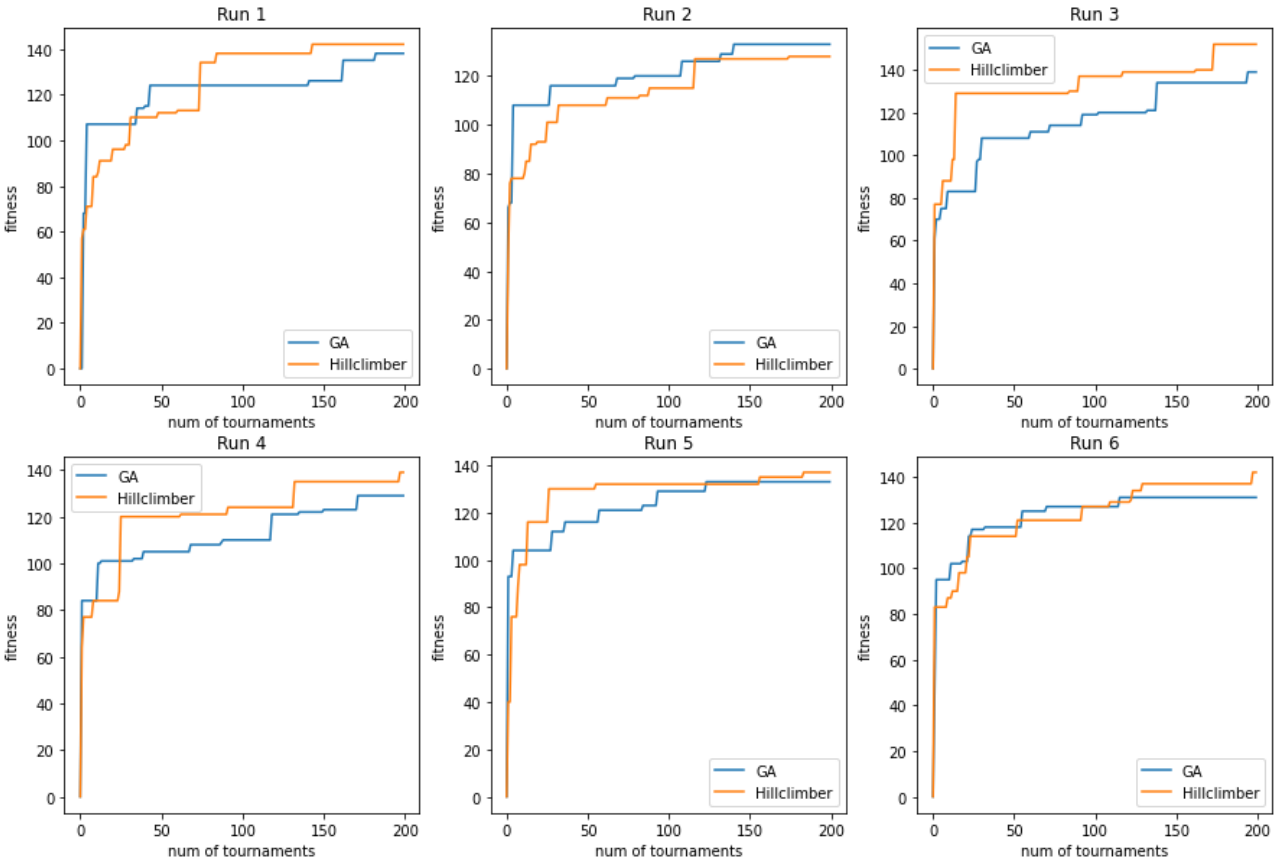

Figure 5: GA vs Hillclimber

We can see that both algorithms successfully optimize to the higher fitness, however, this does not tell us which approach is more suitable for the task. For better comparison we can run each of them a certain number of times and evaluate the performance. Fig. 6 shows the results of running Hillclimber and GA 100 times with different number of algorithm iterations, i.e. 100 and 400. We can see that Hillclimber significantly outperforms GA when the tournaments number is limited to 100 which may be due to the slower optimization of GA comparing to hillclimbing. This may be improved by empirically trying different crossover and mutation rates, however, in this report, we will focus more on final fitness with preselected parameters. The second figure shows that as soon as the number of tournaments increases to 400, the performance of the GA starts to improve, albeit with a large spread. The median of the GA result increases from ≈ 125 to ≈ 147, while for the hillclimber it remains slightly lower. We can make as assumption that in a long run GA can outperform the hillclimber.

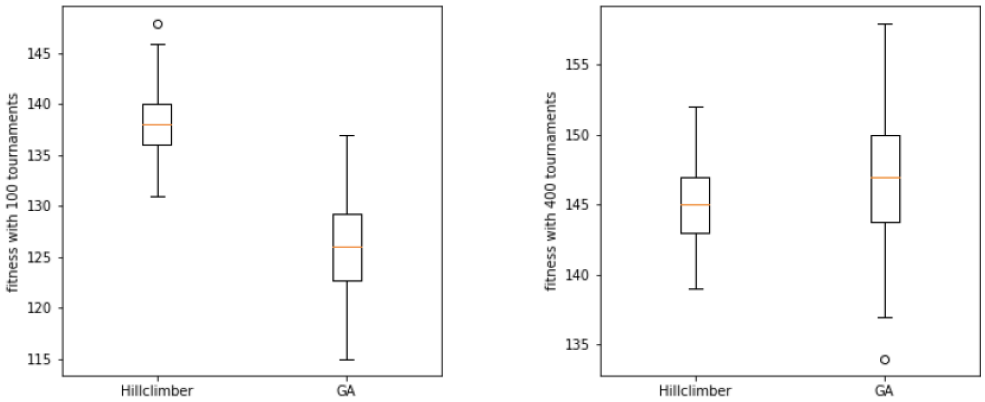

Figure 6: Fitness over 100 runs

To be more precise in our conclusions, we need to consider different numbers of iterations for each algorithm and evaluate the average performance over several runs. This time we do fewer runs for the measurements to reduce the computation time. Table 4 shows the results of measuring the performance using 10 runs and different numbers of tournaments. We see that, as mentioned above, the fitness of the hillclimber is significantly better than the GA for fewer generations, but with larger number of iterations GA starts to outperform. One possible reason for this is a higher tendency of a hillclimber to get stuck in a local optima.

Table 1: Fitness of a Hillclimber vs GA

Iterations	Hillclimber Mean	GA Mean	Hillclimber St.d	GA St.d over 10 runs
100	139.7	129.4	3.38	4.34
200	142.6	137.6	2.69	6.36
300	143.3	145.7	3.40	4.86
400	143.6	146.3	2.41	3.32

Finally, we can conclude that, in the long run, the GA typically achieves higher fitness than the hillclimber, but if for some reason a smaller number of algorithm iterations is used, hillclimber may be a better option.

5 Further Optimization

As mentioned earlier, both GA and Hillclimber can achieve quite good results depending on the initialization point and parameter values. Up until know we have only discussed the *unimodal* variation of a problem. Often we may need to identify several appropriate good solutions and choose one of them at a later stage or use both. There are a few possible solutions to a *multimodal* problem. One approach called *sequential niching* is described by Beasley et.al (1993)[5]. In this case, the algorithm is run several times and the solutions for the problem are found serially with each run of a GA. Once a solution is found, fitness function is depressed for the points within a certain radius from the solution. This prevents the GA from searching the same part of a landscape again. The problem that may arise is that individuals from the entire population may begin to recombine and search for more fit optima. One way to improve this is to use a *restricted tournament selection* and encourage distancing between groups of genotypes. The idea is to define several niches like in the previously mentioned approach, one for each potential solution. Two individuals I_1 and I_2 are selected crossed and mutated to get two new individuals $I_1^{'}$ and $I_2^{'}$. After that both new individuals are inserted back into the population and closest individual is selected for each of them. Then $I_1^{'}$ competes with it's closest individual $I_1^{''}$ and $I_2^{'}$ competes with $I_2^{''}$. New winners are then inserted back into the population. This is one possible option to restrict an individual from competing with other individuals that are too different from it [6].

Interestingly, Hillclimber can be modified to avoid local optima in certain cases. An approach called *simulated annealing* was suggested by Rutenbar (1989) [7]. The algorithm is similar to the one described in Section 2, with the only difference that steps to the lower points of the landscape can also be taken with a certain probability $p(t)$, which starts at 1 and gradually decreases towards zero. In that case when the algorithm starts running any move is accepted, but once the value of t gets smaller the probability of taking a step to the lower point reduces respectively. It means that some negative moves will be accepted and may help to avoid local optima, but not many of them since the hillclimber will just never reach the global one.

6 Discussion

The main purpose of this report was to explore and compare two optimization approaches for the knapsack problem. First, we looked at how an evolving population of hillclimbers could be used to solve the problem in parallel. However, many of the individuals get stuck in local optima and only a few continue to evolve in further generations. After that, the microbial version of the genetic algorithm and the benefits of using the crossover function were discussed. In the next section we compared the performance of both algorithms under different constraints. Final results show that both algorithms can achieve relatively good results on this task, but genetic algorithm shows better results in a long term. Finally, several techniques for improving GAs and Hillclimbers were discussed.

7 Conclusion

In this report, we have discussed and compared two types of evolutionary algorithms. We have seen that both approaches show good results, but neither of them can be considered a universal solution. Since much of our understanding of evolutionary algorithms is obtained empirically, we may need to try different methods and parameters and compare which of them result in better performance on a given task.

References

[1] Jeffrey Horn and David E Goldberg. Genetic algorithm difficulty and the modality of fitness landscapes. In *Foundations of genetic algorithms*, volume 3, pages 243–269. Elsevier, 1995.

[2] Inman Harvey. The microbial genetic algorithm. In *European conference on artificial life*, pages 126–133. Springer, 2009.

[3] Colin R Reeves. Using genetic algorithms with small populations. In *ICGA*, volume 5, pages 90–92. Citeseer, 1993.

[4] Mohsen Mosayebi and Manbir Sodhi. Tuning genetic algorithm parameters using design of experiments. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1937–1944, 2020.

[5] David Beasley, David R Bull, and Ralph R Martin. A sequential niche technique for multimodal function optimization. *Evolutionary computation*, 1(2):101–125, 1993.

[6] Georges R Harik et al. Finding multimodal solutions using restricted tournament selection. In *ICGA*, pages 24–31. Citeseer, 1995.

[7] Rob A Rutenbar. Simulated annealing algorithms: An overview. *IEEE Circuits and Devices magazine*, 5(1):19–26, 1989.

8 Appendix

see *Optimization.ipynb*.