TABLE OF CONTENTS

Optimization algorithms, a comparison between between a hill climber and microbial genetic algorithm.

Hypothesis

The Microbial GA will outperform the Hill Climber algorithm.

Abstract

Optimization tasks usually have the objective to find the global maximum (or minimum) depending on the problem. In some cases, deterministic search methods such as the hill climber method tend to get stuck in a local optimum and fail to find the global solution. On the other hand, stochastic methods, such as genetic algorithms (GA), can improve search methods by helping the algorithm to escape these local optimum solutions in order to move closer towards the global optimal solution[1]. This paper outlines the different characteristics and components between a conventional hill climber algorithm and a microbial genetic algorithm. It will also compare the performance between the two methods when solving a simple knapsack problem and discuss the results in terms of the hypothesis.

Introduction

Genetic algorithms (GAs) have been utilized in a wide variety of optimization problems ranging from protein structure[2] to the traveling salesman problem[3]. The first idea of GAs as a class of evolutionary algorithms was introduced by John Holland and his colleagues in the 1970s at the University of Michigan[4]. The main goals of their research have been to explain the adaptive processes of natural systems and to design artificial systems that retains the important mechanisms of natural systems. Likewise, hill climbing techniques have been used to learn behaviours in robots[5].

Darwinian evolution is emobodied by a population of replicating individuals with three major properties: Heredity, Variation, and Selection.
Heredity means that for evolution to occur, you require a population of individuals where the new individuals are similar to the old individuals. Hence, the offspring share similar characteristics to their parents which allows beneficial mutations to accumulate. Variation means that the population is not uniform, hence, the new individuals will not be completely identical to those they replace.

---

[1] "GA: A Package for Genetic Algorithms in R - Journal of Statistical ...." 29 Apr. 2013, https://www.jstatsoft.org/article/view/v053i04/v53i04.pdf. Accessed 22 Mar. 2019.
[2] "An Analysis of Selection Procedures with Particular Attention Paid to ...." https://dl.acm.org/citation.cfm?id=657604. Accessed 22 Mar. 2019.
[3] "Genetic Algorithms for the Traveling Salesman Problem." https://dl.acm.org/citation.cfm?id=657094. Accessed 22 Mar. 2019.
[4] "Adaptation in Natural and Artificial Systems | MIT CogNet." http://cognet.mit.edu/book/adaptation-natural-and-artificial-systems. Accessed 22 Mar. 2019.
[5] "Learning hill-climbing functions as a strategy for generating behaviors ...." https://www.semanticscholar.org/paper/Learning-hill-climbing-functions-as-a-strategy-for-Pierce/d10c1f731f6ac31c5d3cc18a981f9459ff55ca19. Accessed 22 Mar. 2019.

Selection implies some idea of discrimination in terms of choice of deciding which new individual will replace an old one, this makes the few beneficial mutations likely to stick around[6]. With these three key features, one could generate an evolutionary algorithm to stimulate important parts of evolution to solve problems.

Compared with other evolutionary algorithms, the distinguishing features proposed by John Holland and his colleagues were: (i) bit strings representation; (ii) proportional selection; and (iii) crossover as the major operator[7]. In conjunction to such ideas, during the two laboratory classes, two major optimization algorithms for search have been investigated: a basic hill-climber and a full microbial GA. The aim of this paper is to analyze the hypothesis of if the Microbial GA will outperform the Hill Climber algorithm when solving a simple knapsack problem, for two reasons: Firstly theoretical, to compare each algorithm alongside its advantages and disadvantages; secondly didactic, to come towards a suitable conclusion as to whether the hypothesis is true, and if not, explain the cause.

Method

Lab 3 - A hill climbing GA

The main objective of the task was to implement a population of hill climbers to solve a resource allocation or knapsack problem. In order to do so, we designed a suitable fitness function and selection method to investigate the effects of the mutation rate.

The knapsack (KP) problem is an example of a combinatorial optimization problem. It is concerned with a knapsack that has positive integer volume (or capacity) **V.** There are **n** distinct items that may potentially placed in the knapsack. Item **i** has a positive integer volume **Vi** and positive integer benefit **Bi**. In the most basic form of the problem we will consider there are only one of each item available (0-1 KP). Hence, the goal is to maximize the benefit values whilst being under the volume capacity[8].

In this case, the knapsack has a capacity of 20 cubic inches and N = 10 items of different sizes and benefits. Our given benefit values are indexed from a-j [5,6,1,9,2,8,4,3,7,10], likewise, our respective volumes were [3,2,4,5,8,9,10,1,6,7]. As mentioned above, the goal of the problem was to include items in the knapsack that will yield the greatest total benefit whilst fitting within the maximum volume capacity constraint.

A basic hill climbing algorithm is initialized by selecting random individual in the search space. The population then expands by applying mutations to this single individual. Furthermore, the parent and the offspring are considered as main candidates for the next generation and the fittest one is selected.

---

[6] "The Microbial Genetic Algorithm - University of Sussex." http://users.sussex.ac.uk/~inmanh/MicrobialGA_ECAL2009.pdf. Accessed 22 Mar. 2019.

[7] "Untitled - STU." http://www2.fiit.stuba.sk/~kvasnicka/Free%20books/Goldberg_Genetic_Algorithms_in_Search.pdf. Accessed 22 Mar. 2019.

[8] “CW 3.” - Christopher Buckley

A hill climbing algorithm

1.Randomise the genotype g0
2. Copy Genotype to temporary genotype
    g1← g0
3. Mutate g1 (increment the angle of one allele)
4. Evaluate fitnesses
5. If (Fitness(g1) > Fitness(g0))
        g0← g1
    else
        do nothing
    end
6. Goto 1.

Now utilizing the pseudo code above, we could create a hill climber function to solve the knapsack problem given. However, before we do so, we need to define the mutation function and the fitness function. The former will start with an empty sack (array of 0s), mutate a random genome within the sack with a 1 if 0 else if the genome has a value of 1 replace it with a 0. The mutation will return a randomly generated genotypes. The latter will take the sum of the benefits where the genome has a value of 1 and return the total benefits if the volume of the indexed genomes are under the total volume capacity. When running hill climber algorithm on an empty knapsack, it yields a maximum fitness value of 30.

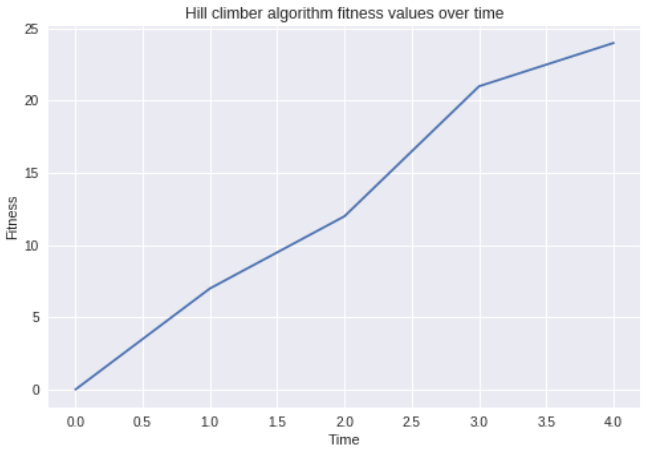


Figure 1. Hill climber algorithm fitness function over time. Fix as generation

However, when running the algorithm different iterations, although the trend present in the graph is universal, the maximum values vary greatly. ***Figure 2.*** below represents the maximum fitness values from the hill climber algorithm ran on 5 iterations.

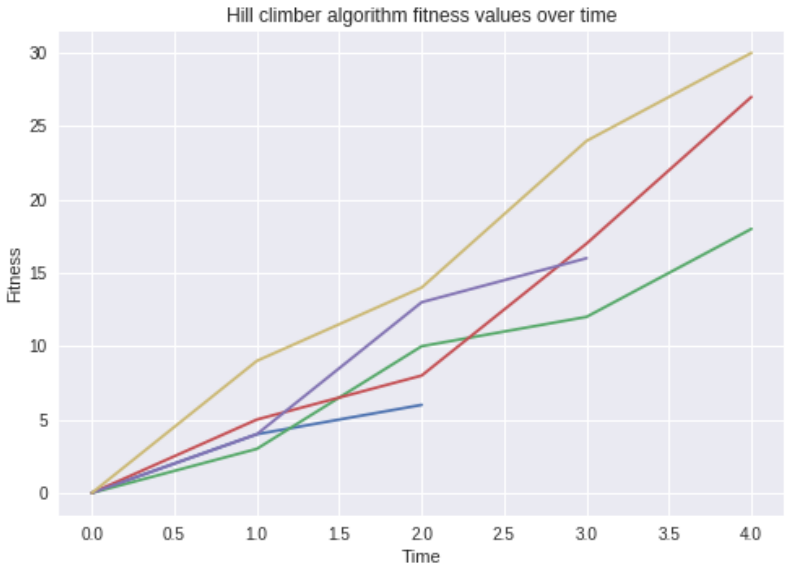


Figure 2. Hill climber algorithm fitness function over time, 5 iterations

As we can see, the maximum fitness values vary from 30 to 6. This is one of the major disadvantages of the hill climbing algorithm. Hill climbing methods such as, gradient descent, exploit local information about a search space to find optima. This method performs particularly well for unimodal functions, however, most real world functions aren't. The algorithm can get stuck in a local optimum (foothill problem), it can get stuck in mostly flat surfaces with few sharp peaks (the plateau problem), or it can get stuck because the direction of ascent is not within our directions of motions (the ridge problem)[9]. This can be represented in the graph below:

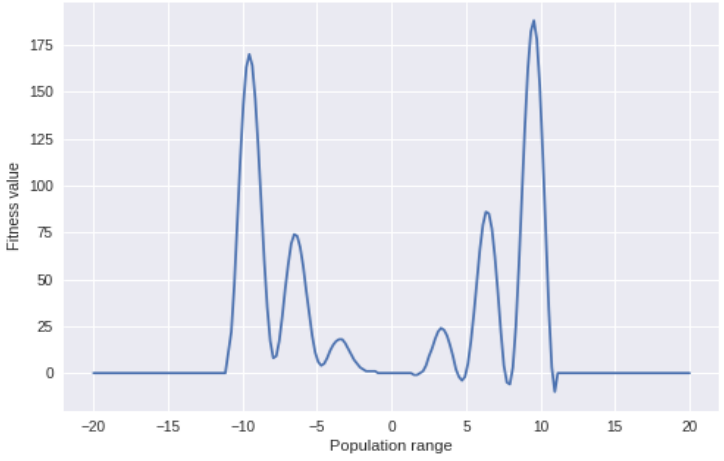


Figure 2. Fitness function landscape graph

[9] "Dynamic Hill Climbing: Overcoming the limitations of ... - ResearchGate." 22 Nov. 2012, https://www.researchgate.net/publication/2688657_Dynamic_Hill_Climbing_Overcoming_the_limitations_of_optimization_techniques. Accessed 22 Mar. 2019.

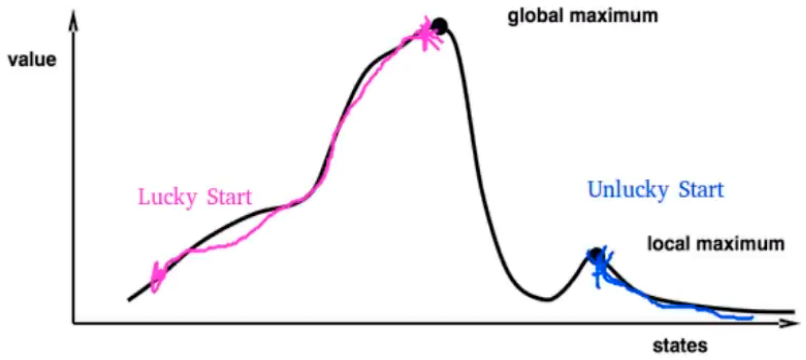


Figure 3. Representation of the hill climber's limitations[10]

Figure 2. represents a simple fitness function where the x-axis represents the population range and the fitness values as the y-axis. Now, in conjunction to ***figure 2.*** ,***Figure 3.*** represents an example of how the hill climbing algorithm can be stuck in local a local optima. As shown above, from a random starting point, if the algorithm follows a gradient ascent path "Luck start", it will eventually reach the global maximum. However, if the algorithm commences from the "Unlucky Start" it will only reach a local maximum and fail to reach the global one.

Although the hill climbing algorithm may have disadvantages, its main advantage is that it is easy to implement and relatively fast to execute. Furthermore, there exist different measures to combat against the problems born with the algorithm. For instance, the ridge problem can be solved by adopting a dynamic coordinate frame (dynamic hill climbing algorithm paper) and the foothill problem can be fought by exploiting local optima. However, the plateau problem is a harder problem to solve as by nature, the space is not providing any information about its structure. In this case, running multiple iterations with random starting points might allow the algorithm to perform better.

Lab 4 - "An optimization algorithm" - full Microbial GA

The main objective of this lab was to implement a fully functional microbial GA or other optimization algorithms to solve the simple knapsack problem. This allowed us to investigate the effects of population size, mutation rate and recombination rate on evolution.

The principle behind a full microbial GA comes from the notion of *bacterial conjugation.* Compared to many beings in nature, microbes such as bacteria do not undergo sexual reproduction, instead they reproduce by binary fission. *Bacterial conjugation* is the process where chunks of DNA and plasmids are transferred from one bacterium to the next when they are in direct contact with each other. This behaviour can now be adapted to create a full microbial genetic algorithm to solve the knapsack problem, the pseudo code is as below:

[10] "Introduction to Monte Carlo Methods | Udemy." https://www.udemy.com/introduction-to-monte-carlo-methods/. Accessed 22 Mar. 2019.

A full microbial GA

1. Initialize a random population P
2. Evaluate entire population
3. Pick 2 individuals at random & evaluate them. Assign the fitter individual parent one W1 (winner) and less fit L1 (loser)
4. Go along the genotype of W1 and with some probability (Pcrossover) copy the gene to L1
5. Mutate L1 (containing partial W1 genes)
6. Evaluate new fitness of L1
7. Run until success or give up, goto 3.

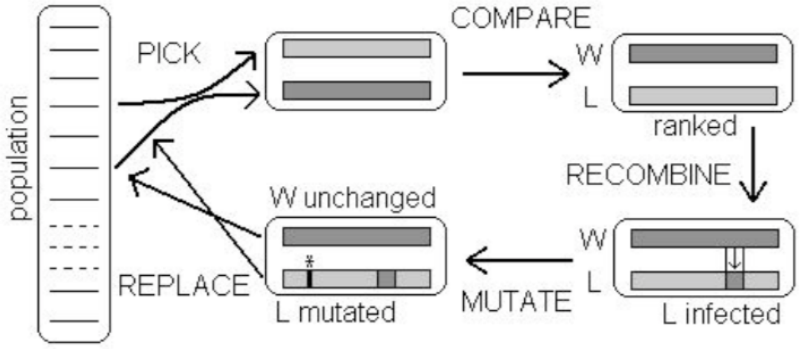


Figure 4. Bacterial conjugation, the genotypes of the population are represented as a pool of strings. One single cycle of the Microbial GA is represented by the operations PICK (two at random), COMPARE (their fitnesses to determine Winner = W, Loser = L), RECOMBINE (where some proportion of Winner's genetic material 'infects' the Loser), and MUTATE (the revised version of Loser).[11]

Figure 4. illustrates the recombination described as the winner 'infecting' the loser with its genetic material and it is important to mention that the rate of infection is varied. As recommend in Inman Harvey's "The microbial Genetic Algorithm" (I. Harvey, 1996), the rate should be around ~50% although in bacterial conjugation, the rate of infection is rather low. Furthermore, in principle, we may want explore different rates of infection where we can assign any value between 0% and 100%. [12]

To execute the lab and proceed further in this investigation whether the the Microbial GA will outperform the Hill Climber algorithm, the pseudo code from Inman Harvey's "The microbial Genetic Algorithm" has been implemented. The original code is in C and the implementation used in this investigation is in python. [13]

---

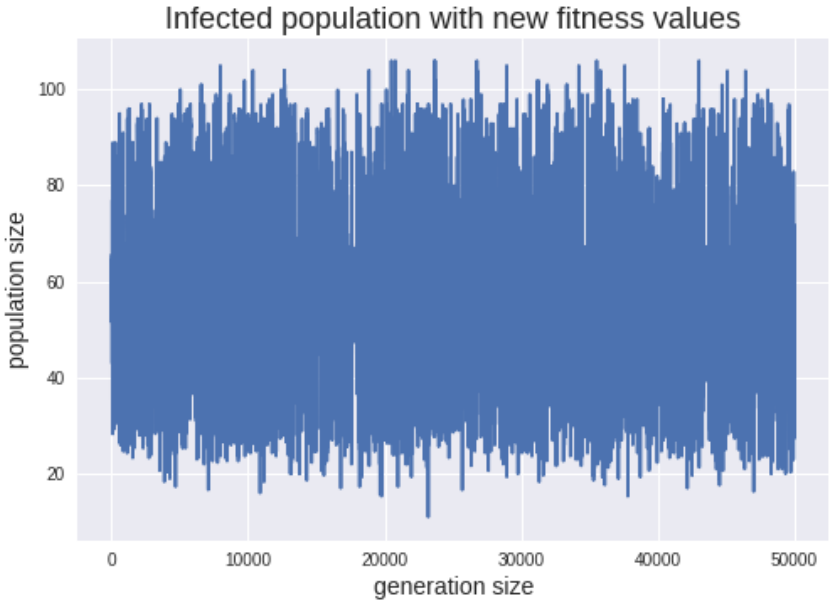
[11] "The Microbial Genetic Algorithm - University of Sussex." http://users.sussex.ac.uk/~inmanh/MicrobialGA_ECAL2009.pdf. Accessed 22 Mar. 2019.
[12] "The Microbial Genetic Algorithm - University of Sussex." http://users.sussex.ac.uk/~inmanh/MicrobialGA_ECAL2009.pdf. Accessed 22 Mar. 2019.
[13] "The Microbial Genetic Algorithm - University of Sussex." http://users.sussex.ac.uk/~inmanh/MicrobialGA_ECAL2009.pdf. Accessed 22 Mar. 2019.

```
void microbial_tournament(void) {
  int A,B,W,L,i;
  A=P*rnd();                          // Choose A randomly
  B=(A+1+D*rnd())%P;                  // B from Deme, %P..
  if (eval(A)>eval(B)) {W=A; L=B;} // ..for wrap-around
  else {W=B; L=A;}                    // W=Winner L=Loser
  for (i=0;i<N;i++) {                 // walk down N genes
    if (rnd()<REC)                    // RECombn rate
      gene[L][i]=gene[W][i];          // Copy from Winner
    if (rnd()<MUT)                    // MUTation rate
      gene[L][i]^=1;                  // Flip a bit
  }
}
```

Figure 5. Full Microbial GA in C[14]

"There is a population of size P of binary genotypes length N, stored in an array of the form gene[P][N]. This code represents a single tournament, that is repeated as many times as is considered necessary. We assume a pseudo- random number function rnd() that returns a real number in the range [0.0,1.0). REC is the recombination or 'infection' rate (suggested value 0.5), and MUT is the (per locus) mutation rate. D is the deme size."[15]

The algorithm implemented will take as inputs a population size and a generation size. In this case the algorithm is ran with a population size of 50000, for a course of 1000000 generations, a crossover rate of 50% and mutation rate of 1/20 was implemented. Using the code above in Figure 5. as inspiration, the algorithm yielded the graph below:



[14] "The Microbial Genetic Algorithm - University of Sussex." http://users.sussex.ac.uk/~inmanh/MicrobialGA_ECAL2009.pdf. Accessed 22 Mar. 2019.
[15] "The Microbial Genetic Algorithm - University of Sussex." http://users.sussex.ac.uk/~inmanh/MicrobialGA_ECAL2009.pdf. Accessed 22 Mar. 2019.

Figure 6. Full microbial GA algorithm output

Figure 6. illustrates the new fitness values of the 'infected' population containing 50% of the winner's genotypes. In order to clearly visualize the data, a threshold has been set to obtain the maximum values of the population returned, this has been represented in the graph below:

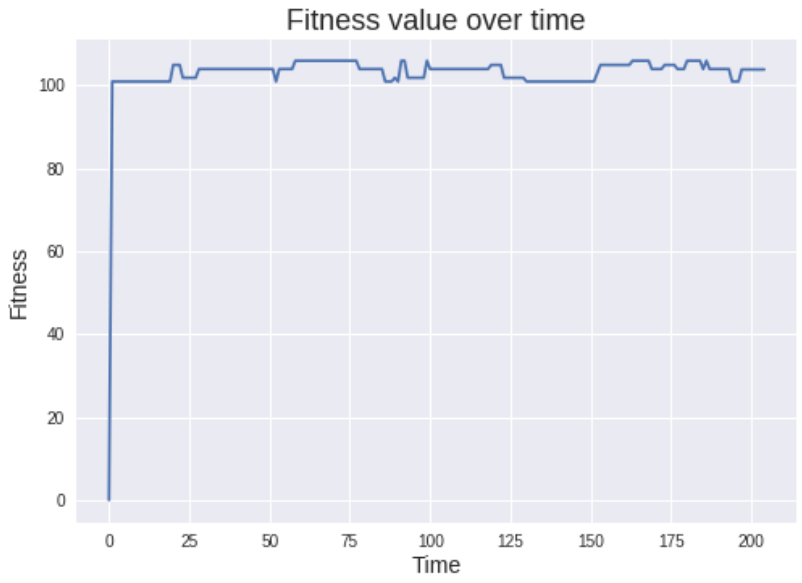


Figure 7. Full microbial GA algorithm output with threshold

A max value threshold has been set on the full Microbial GA algorithm's output to produce figure 7. We are able to observe that over time, the fitness value shows an increase and revolves around 100. It is crucial to mention that it is sometimes unfavourable to set your population as panmictic, since after multiple iterations or generations, the population could converge too quickly and become uniform[16]. To combat this issue, it is common to introduce a geographical distribution, if a population is considered to be distributed over a virtual landscape, usually a 2-D array, the population will contain more diversity and variation within the local context.

Now, having analyzed the different inner workings of the two optimization algorithms we can run a quick experiment. Using the same knapsack problem size and parameters, we can run the hill climber and microbial GA to see which one outputs the greater fitness function. Here are the graphs:

[16] "The Microbial Genetic Algorithm - University of Sussex." http://users.sussex.ac.uk/~inmanh/MicrobialGA_ECAL2009.pdf. Accessed 22 Mar. 2019.

Hill Climber Performance:

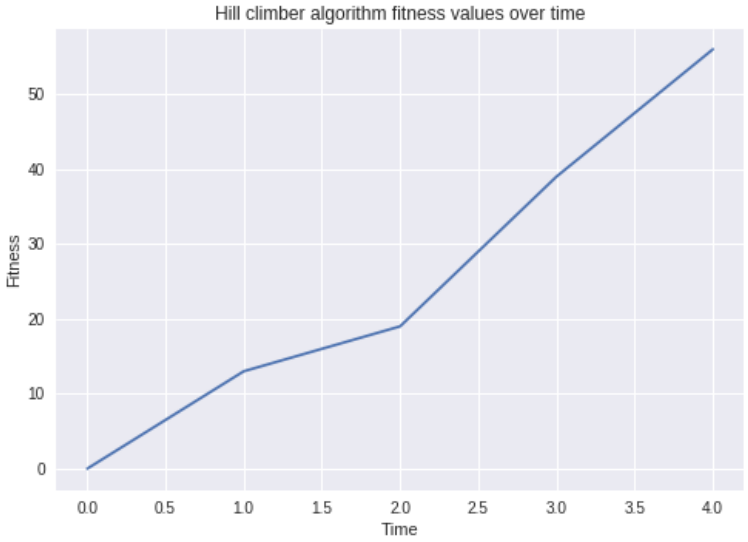


Figure 8. Fitness value output of the hill climber algorithm on knapsack problem.

Full Microbial GA:

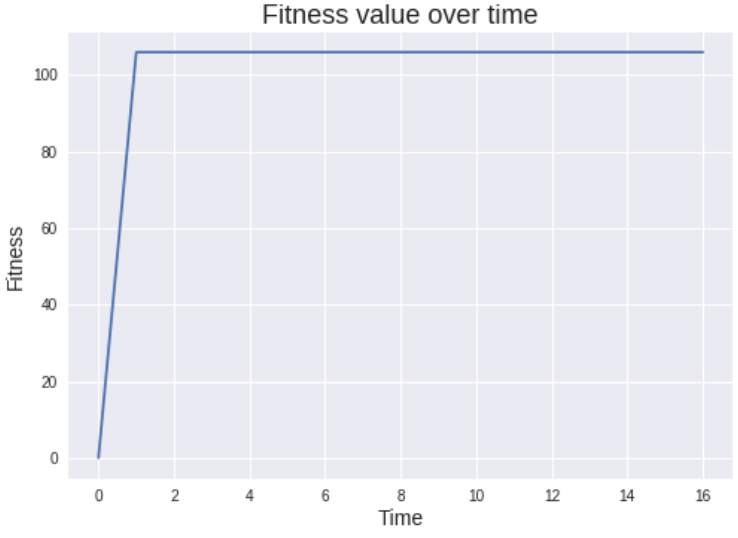


Figure 8. Fitness value output of the Full Microbial GA on knapsack problem.

Discussion

We can observe in figure 7. the hill climber algorithm returned an upward trending graph of the fitness value which yielded a maximum value of 63.0. On the other hand, figure 8. where a full microbial GA was implemented, a similar upward trend fitness value is also visible, however the maximum value obtained is over 100. This represents that, given the same problem size and parameters, the microbial GA clearly outperformed the hill climber algorithm which may have been stuck in a local optima, failing to reach the global solution, thus confirming the hypothesis "The Microbial GA will outperform the Hill Climber algorithm" to be true. To support this claim further, the a conventional GA inspired by Luca Scrucca's "GA: A Package for genetic Algorithms in R" was implemented on figure 2. fitness landscape[17].

The algorithm implemented was run through 100 generations yielding a top fitness value of 189.0. Even though this GA implementation differs from the full microbial GA and takes different inputs as parameters, it nevertheless shows how it is able to reach global optima outperforming the hill climber algorithm, which is shown by the blue marker at the peak (9.5499).

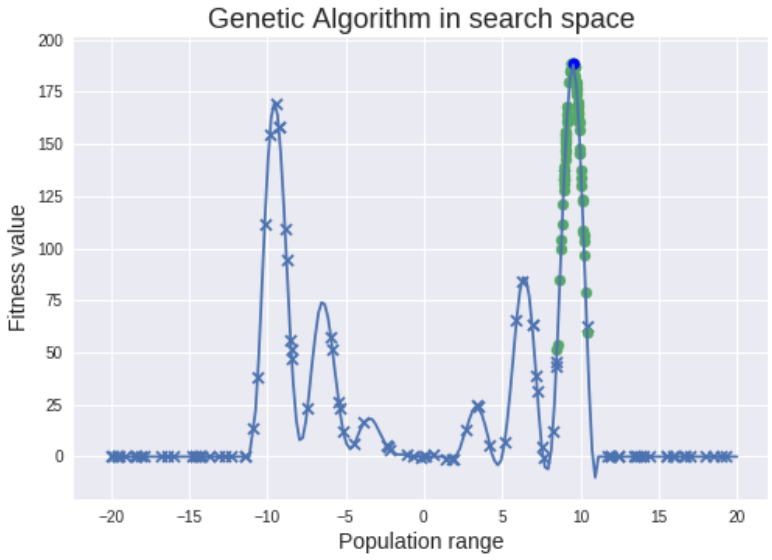


Figure 9. GA implementation on fitness landscape

[17] "(PDF) GA: A Package for Genetic Algorithms in R - ResearchGate." https://www.researchgate.net/publication/257428155_GA_A_Package_for_Genetic_Algorithms_in_R. Accessed 22 Mar. 2019.

References

1. Holland, John H. *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 2010.
2. Grefenstette, John & Gopal, Rajeev & J. Rosmaita, Brian & Van Gucht, Dirk. (1985). Genetic Algorithms for the Traveling Salesman Problem.
3. De la Maza, Michael & Tidor, Bruce. (1993). An Analysis of Selection Procedures with Particular Attention Paid to Proportional and Boltzmann Selection.
4. Hill climbing for robots - David Pierce and Benjamin Kuipers. 1991. Learning hill-climbing functions as a strategy for generating behaviors in a mobile robot. In Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats, Jean-Arcady Meyer and Stewart W. Wilson (Eds.). MIT Press, Cambridge, MA, USA, 327-336.
5. Holland, John H. *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 2010.
6. Goldberg D (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Boston.
7. Navarrete, Jonathan. "Introduction to Monte Carlo Methods." *Udemy*, www.udemy.com/introduction-to-monte-carlo-methods/#instructor-1.
8. Scrucca, Luca. (2013). GA: A Package for Genetic Algorithms in R. Journal of Statistical Software. 53. 1-37. 10.18637/jss.v053.i04.