Performance in Combinatorial Optimization Techniques: Hill-Climber versus Microbial GA

Cand.No. 183708
Department of Informatics
University of Sussex

June 3, 2020

Abstract

The Knapsack problem (KP) is a common combinatorics problem that deals with resource allocation and is classified as NP-Hard. The cost-benefit ratio of sets of weighted items that could fit into a knapsack are constantly evaluated in an attempt to maximize benefit $\Sigma_i^N B_i$ whilst staying within the weight constraints set out by the KP: $\Sigma_i^N C_i \leq C$ where C is the maximum capacity of the knapsack. Using both the Hill-Climber (HC) and the Microbial Genetic Algorithm (MGA) in an attempt to find optimum solutions in the KP search space gives the opportunity to explore their differences, highlight the various limitations and benefits of each and examine the prevalence of their solutions.

Keywords— Knapsack, Hill-Climber, Microbial Genetic Algorithm

1 Introduction

In the context of the KP, the search for local and potentially global maxima within the problems solution space demonstrates the ways in which these algorithms differ. The first of these algorithms involves local search using a population of Hill-Climbers running in parallel. The second is a genetic algorithm, specifically the Microbial Genetic Algorithm which involves a sexually active population with crossover from a more successful "winner" genotype (candidate solution), to a "loser" genotype. Whilst both are affected by random mutations across generations it is only the MGA that experiences this type of genetic recombination. The performances of both systems are evaluated and their differences compared as to ascertain the importance of the mutation rate and genetic crossover for genetic diversity and success in a population.

2 The Hill-Climber

Simple Hill-Climbing is a local search optimization technique, which starts with initializing one randomly generated candidate solution and then repeatedly refining this solution through various mutations. In this way the HC separates itself from Random Search. Once a genotype has been mutated its fitness is evaluated and then compared to the fitness of its predecessor, if the child genotype returns a higher score then that solution is stored as the current working best. Otherwise random mutations continue until this situation arises. This simple HC process is shown below in figure 1.

2.1 The Hill-Climbing Algorithm

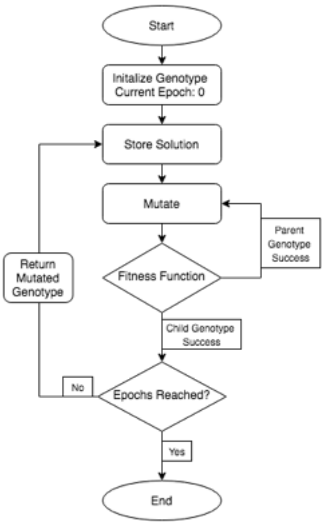


Figure 1: Simple Hill-Climber Flowchart

For this paper a uncomplicated implementation of the HC algorithm was selected as to better demonstrate its restrictions; Every HC instance acts as an independent object and there is no exchange of genetic material between the objects. Mathematically it can be expressed as trying to maximise the target function $f(x)$ where x is a vector containing discrete values. At each step the HC will amend its phenotype (the information encoded in each genotype) and through the fitness function decide whether this is a positive or negative change in the value of $f(x)$. And it is in this way the Simple HC differs from gradient descent, where all elements in x are adjusted depending on the first derivative of $f(x)$. Once no such change can be made that would improve our target function then x can be said to be a local optimum.

2.2 Stuck in Sub-Optimal Space

In vector space each x can be viewed as a vertex in a 3-dimensional plot. HC travels from vertex to vertex, locally incrementing $f(x)$. If our heuristic were upwards convex and a maximum had been found then one could make the claim that it is in fact the globally optimal solution.

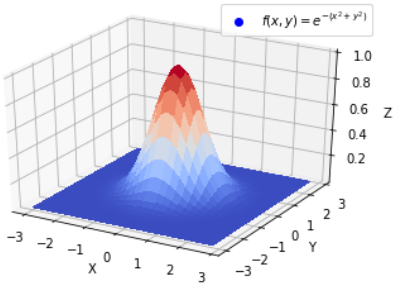


Figure 2: One Maximum

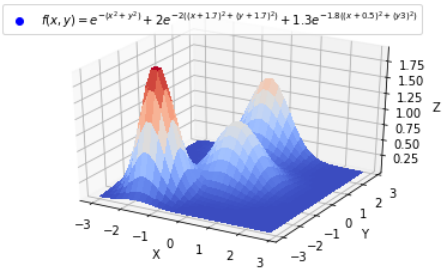


Figure 3: Three Local Maxima

It is in these simpler cases that HC performs well (Figure 2). However with the introduction of a space containing multiple maxima (Figure 3), if the HC were to start with a less than optimal initial solution (i.e. Our Hill-Climber started from a poor position) the algorithm may stop, having converged at a lower maximum, and so end never having reached the best solution. As many of these types of KPs contain functions that are not convex HCs can often fail to reach the global maximum.

2.3 Methodology

Other optimization algorithms such as simulated annealing or random walks try to deal with this problem however introducing a population of HCs builds upon the Simple HC algorithm to become Random-Restart Hill-Climbing. In this variant to find the optimal solution only the working best genotype across all parallel HCs is stored, until another better solution presents itself or the algorithm has run its course. This seemed to be very effective as it was thought exploring the solution space with a population could often yield a better result than endlessly refining one genotype.

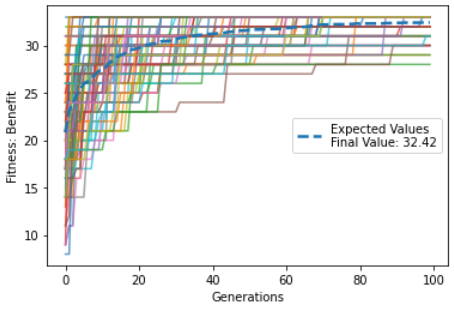


Figure 4: Hill-Climber Population
Average: 32.42

2.4 Method

A single HC object contained a binary array with a shape of an equivalent size to the number of items in the KP. Each binary value represented whether the HC chose that item to be in the KP or not: 0 encodes not that item and 1 selects that item. From this running totals were kept to check the overall benefit and ensure no illegal KPs were created. Figure 4 demonstrates a population of 100 HCs evolving independently for 100 generations and tracks each instances solution across each generation, as shown on the x-axis. The fitness displayed on the y-axis is a sum of benefits for each item chosen to be in the KP. This will hold true for all similar examples to come - for reference the KP here and in the following examples was found to have a maximum benefit of 33). The dashed blue line represents the average fitness of all HC objects at any one point in their evolution and the graphs legend also contains the final average value across all HCs to give a rough measure of the effects of its parameters (mutation rate, "generations to run for") on that populations success.

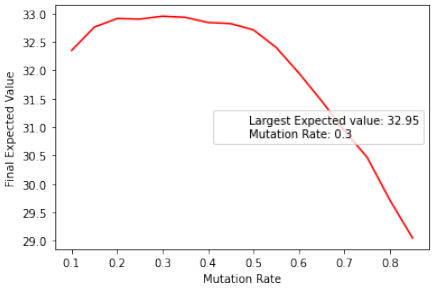


Figure 5: Final Expected Value of Hill-Climber Population against Mutation Rate

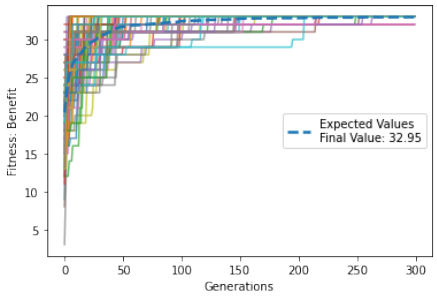


Figure 6: Ideal Mutation Rate: 0.3 Hill-Climber Population Average: 32.95

2.5 Mutation Rate in a Hill-Climber Population

Figures 5 and 6 explore the effect of mutation rate (with increments of 0.05) on the mean performance of HCs in a population and show the corresponding mutation rate that yielded the best performing population, with the evolution of said population, respectively. We can see from Figure 5 that in the 0.20 to 0.45 range a near plateau-like landscape with some mild peaks and then a steep drop off towards failure from 0.50 onward. This is a fairly intuitive result as too high a mutation rate means any successful saved states may have every individual gene flipped and so whatever progress that genotype had built up is lost. In other words increasing the probability of searching all areas the HC becomes unable to converge on any one optimal solution and the algorithm is reduced to being a primitive random search. Alternatively too low a setting means not enough chance for growth and the number of generations necessary to see a similar successful result becomes computationally inefficient. This is an effect of the mutation rate on the HCs performance - It is worth noting at this point the exact value of an ideal mutation rate is a problem specific value; In this case a recursive examination of the previously mentioned flat area showed that the optimal value appeared to lie in the 0.275 to 0.350 range. Increasing the number of generations for a HC population to evolve for shows individuals converging to the global optimum as seen in Appendix A. With a sufficient increase in generations a similar result can be achieved as with a population with a lesser mutation rate however this can quickly become computationally inefficient.

3 Microbial Genetic Algorithm

Genetic algorithms are an increasingly popular class of evolutionary algorithms and used in modern day computing and optimization. The GA algorithms heuristic was to simulate the theory of evolution of a population as proposed by Darwin, with the first GA having been introduced by John Holland and then later refined by his student David Goldberg (Gen & Cheng, 2000).

3.1 The Microbial GA

The MGA retains much of the same functionality as a conventional genetic algorithm would but is simplified to demonstrate a horizontal gene transfer (Figure 8). This is to mimic bacterial conjugation, a process in which bacteria transfer genetic material between cells through direct contact and was the inspiration for the original paper on MGAs (Harvey, 2009).

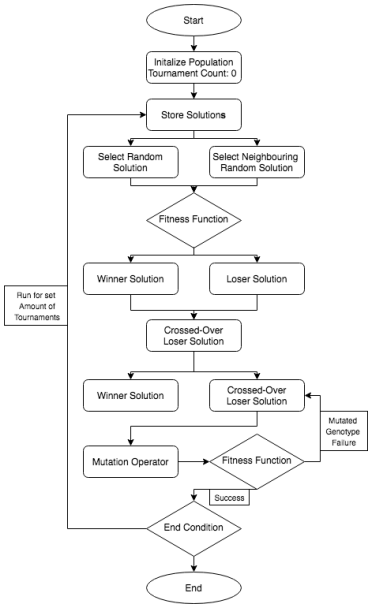


Figure 7: Microbial Genetic Algorithm Flowchart

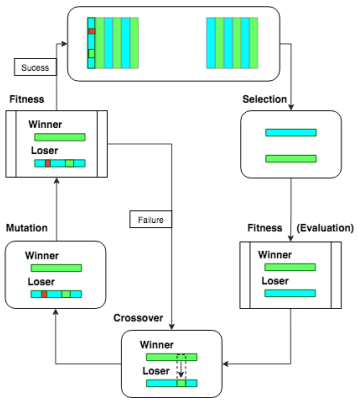


Figure 8: Operations on a single Population - Shown at the Top in the after and before Stages

The MGA featured in this paper makes use of selection, crossover, mutation and evaluation (Figure 7, Figure 8) to meet the required properties of Selection, Variation and Heredity that are necessary for evolution to take place (Harvey, 2009). Interestingly Hollands original paper on genetic algorithms did not feature crossover as one of the original operators; The original thesis made use of selection and probabilistic mutation only (Wilson, 2014). This crossover operation is however more commonplace in modern GAs.

3.2 Genetic Recombination in Microbials

The crossover is a stochastic method used to combine genetic information of two genotypes to create new solutions. In terms of this MGA and also asexual reproduction in bacteria, crossover can be viewed as an infection (Harvey, 2009), where the winning genotype copies a portion of itself over to the loser, whilst remaining unaffected by the operation. This function was carried out using three traditional crossover operators: One Point, Multi Point and Uniform. Below shows two initial binary arrays to represent genotypes and the sexual operations and changes depending on crossover operator - each individual item encodes a gene within the genotype.

1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0

3.2.1 One Point

One Point crossover uses a single randomly selected point for fragmentation and the head or tail of the parent as the portion to splice over to the child genotype.

1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1

3.2.2 Multi Point

Multi Point crossover (K-Point) segments the parent by creating K crossover points in the original shape and then carries over the genes split by those K points over to the child.

1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	0	0	0	0

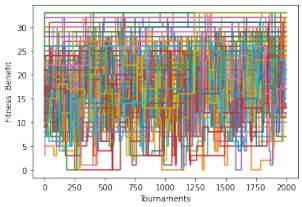
3.2.3 Uniform

Uniform crossover treats each gene separately, creating a random number for each as it iterates through the array and by passing some mutation rate dependent condition, copies that gene over to the child.

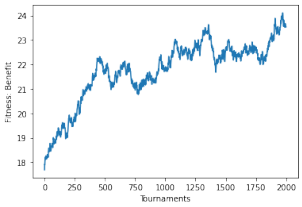
1	1	1	1	1	1	1	1	1	1
0	1	1	0	1	0	0	1	0	0

3.3 Method

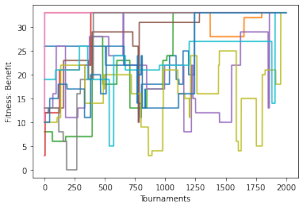
Figure 9 demonstrates an instance of the MGA algorithm (see Appendix B for parameters) during its evolution - for our analysis, Figure 9.d is most useful. This image shows the average evolution of all the most successful genotypes in a population at a set mutation rate. The most successful genotypes (best) here are solutions encoded with a binary array that give a value equal to the maximum benefit/fitness value of all solutions at any point in their evolution. I.e. a candidate optimal solution.



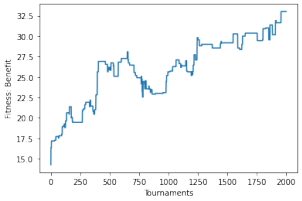
(a) Complete Evolution of Population



(b) Average Evolution of Fig.9a Population



(c) Complete Evolution of Best performing Genotypes



(d) Average Evolution of Fig.9c Best performing Genotypes

Figure 9: Microbial GA: Fitness against Tournament:

From there we can plot these averages as they are calculated over a range of mutation rates. Figure 10 shows these averages and the corresponding mutation rate and number of solutions which reached the maximum value (out of 100 total) found by this genetic algorithm.

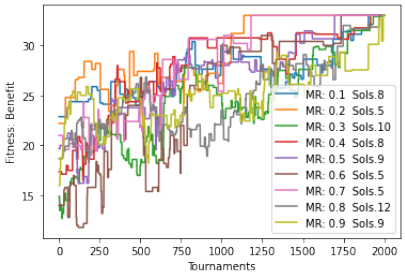


Figure 10: Hill-Climber Population
Average: 32.42

To account for the different conditions each instance of the MGA will encounter during evolution: Populations are all initialized at random, starting from unlike positions and the probabilities which dictate selection, crossover and mutation differ at every call. To navigate this problem at each mutation rate, multiple runs are recorded and those plots (individually as seen in Figure 9.d) are then averaged to create a graph of fitness against tournaments, with each line representing mean performance of an average of well-performing genotypes across mutation rates (Figure 11).

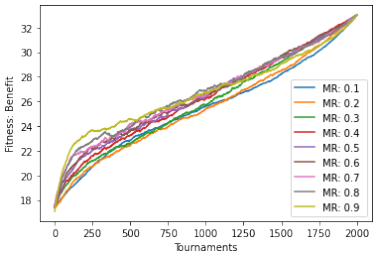


Figure 11: Average Successful Solutions across 250 Runs per Mutation Rate

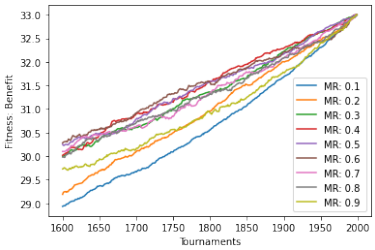


Figure 12: Last 20% Zoom of Figure 11

3.4 The Balance: Exploration versus Exploitation

In optimization algorithms the balance between exploration and exploitation is key. In other words the balance between exploring the search space fully and exploiting those areas in that space which are local to other previously visited points. Exploration is often attributed to the mutation rate, whilst crossover influences exploitation. With a balance of these two principles the algorithm can hope for convergence on a global optimum. This can sometimes be seen as an oversimplification of these principles' roles (Črepinšek et Al, 2013) as selection in this algorithm was based off neighbouring solutions chosen at random, rather than solutions' fitness. This made it difficult to maintain selection pressure throughout of a MGAs evolution, which has been shown to be a contributor to the levels of exploration and exploitation achieved (Back, 1994). However due to the relatively small feasible region of the KP, this papers focus on mutation and crossover and the MGAs original intention of partly being a tool for educational purposes this can be pushed aside for now.

3.4.1 Exploration

Figure 11 (parameters explicitly stated in Appendix C) shows a large amount of spread between the different mutation rates however this was to be expected due to the large range of values

under experiment (0.1 to 1 with a step of 0.1). Considering that, a convergence past the 1900/1950 tournament mark (Figure 12) can be made out. Figure 13 shows the average proportion of solutions per mutation rate that were successful in converging to the same optimum, as a percentage of the total population size (in red). Also on the same plot (opposing y axis in Blue) is the integral of each line (Area under curve: AUC) in Figure 12. These both are demonstrated as a loss function against varying mutation rates. These metrics were chosen as to ascertain in which region of the mutation rates range did the MGAs perform best.

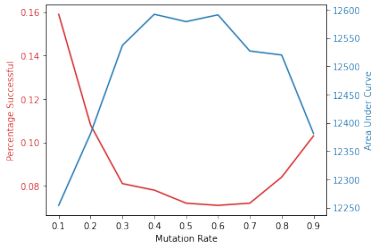


Figure 13: Loss Function of Successful % of Solutions and Area Under Curve against Mutation Rate

We can see from our AUC plot (Figure 13) that at mutation rates between 0.4 to 0.6 and within the 1600 - 2000 tournament range, the successful solutions in those MGAs spent more time performing with higher fitness values than the others. From our percentage successful plot (Figure 13) we can see that at mutation rates of [0.1, 0.2) were where the highest number of solutions in a MGA population were performing at their best and converging to the same optimum. Recursively searching this lower end of the spectrum showed the 0.0015 to 0.0135 mutation rate range to yield the best results (Figure 14).

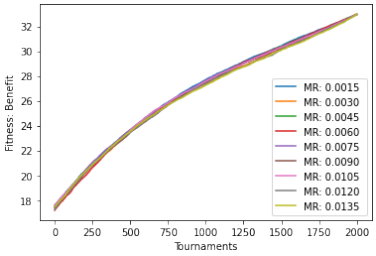


Figure 14: Average Successful Solutions across 250 Runs per Mutation Rate

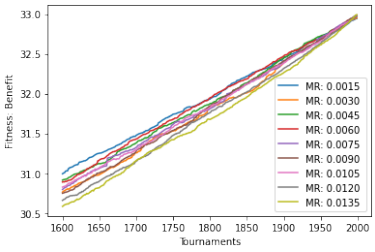


Figure 15: Last 20% A More Promising Convergence

We can see from the above figures a greater convergence on the global optimum. The spread of Figure 14 is much less, due to the much smaller range of mutation rate values and comparing the spread of Figure 15 against that of Figure 12 we can see that this lesser range of 0.0015 to 0.0150 shows a greater convergence. A risk of choosing a mutation rate too small can often lead optimization algorithms to converge on local optima and so it was encouraging the same max value was consistent across both ranges shown. Whilst the best mutation rate is a problem specific value, fairly low values are commonplace in genetic algorithms and so this was an indication of a good range to examine. The increase in performance is further supported by Figure 16: Whilst a trend in the percentage of successful solutions is hard to spot, if we take note of the scale on the y-axis

it peaks at 26% of the population successfully hitting the maximum rather than the 16% seen previously (Figure 13), showing an undeniable boost in performance.

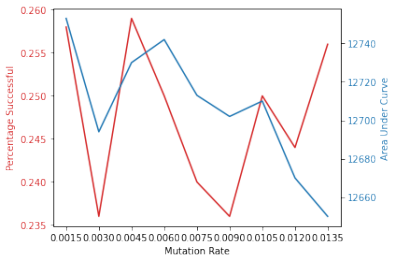
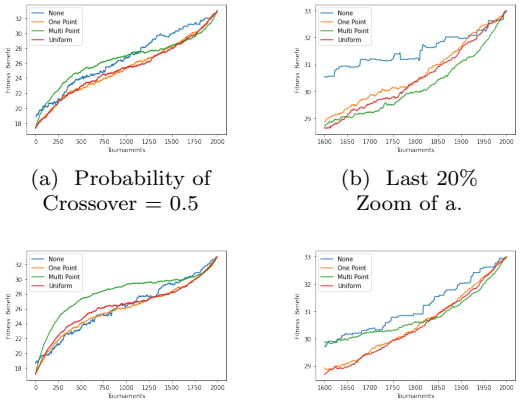


Figure 16: Loss Function of Successful % of Solutions
and Area Under Curve against Mutation Rate

To investigate whether this convergence is due to the smaller mutation rate allowing crossover to converge the population we must now compare results of the various crossover operators and even the lack of.

3.4.2 Exploitation

A high chance of crossover means constantly running the risk of inhibiting a good combination or even producing illegal genotypes (Umbarkar Sheth, 2015). Illegal genotypes here are dealt with by way of the fitness function however with a high probability of crossover to occur it can quite easily lead to potentially good or already good solutions having their phenotypes erased or copied over thus preventing the population from converging on the global optimum.



(a) Probability of Crossover = 0.5

(b) Last 20% Zoom of a.

(c) Probability of Crossover = 0.95

(d) Last 20% Zoom of c

Figure 17: Microbial GA: Fitness against Tournament:

Figure 17a and b show the effects of all crossover operators that were discussed and also of no crossover. Probability of crossover occurring was set to 0.5 and mutation rate set to 0.012 whilst 17c and d show the same except with a higher crossover chance of 0.95 (for parameters of both see Appendix D). From these figures we can see that even with no crossover, as denoted by the blue line ("None"), convergence on the global optimum is still achieved and the best performing

crossover operator ("Uniform") was a constant in both configurations (Figure 18). With crossover convergence seems slower but also smoother. The consistency of performances of the crossover operators at varying crossover rates, as seen in Figure 18, is an expected result however at the 0.95 rate we see a near double average boost in performance (Figure 18, disregarding no crossover). At this 0.95 level new solutions are created so frequently it now allows for the exploration of new genotypes, showing genetic recombination as a useful tool for the evolution of a population.

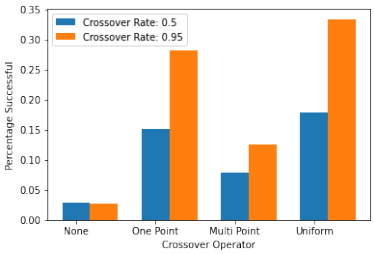


Figure 18: Percentage Solutions Successful per Crossover Rates

Although the paths taken by the mean best solutions in both sets of images differ slightly, overall their slope as tournaments progress retains a near identical shape - I.e. the choice of crossover operator or even a lack thereof seems to have little effect on a populations ability to converge.

4 Conclusion

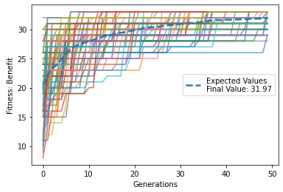
The ability to share genetic information amongst a population is a defining factor of Genetic Algorithms (and so by extension Microbial Genetic Algorithms) and is what leaves Hill-Climbing behind as a local search technique. This is not to down-play the Hill Climbing algorithm's importance as there are uses, for either algorithm, depending on the problem. As seen in this paper both the Hill-Climber and the Microbial Genetic Algorithm performed to reach the working global optimum but the Hill Climbers inability to backtrack and take up a weaker position can potentially create a limitation in its capability to explore the search space fully. It is however a consideration of this paper that the search space of this KP was too limited to explore on a grander scale the possibilities of the algorithms becoming trapped/converging in less than optimal solutions. Regardless, in the microbial population a successful performance can be attributed to a correct selection of the mutation rate, the crossover rate and the crossover operator as seen from the experiments above. The choice of crossover operator may not hold as much importance as the other two characteristics, mutation rate and crossover rate, as populations using any of the three recombination techniques reached the maximum value, but it is clear some form of genetic recombination is needed for high performance across the whole population otherwise the algorithm will stagnate with a slow mutation rate and rely on solutions that were randomly placed near the optimal location. The Hill-Climber, lacking some of these more complicated evolutionary functions places all its importance in its mutation rate and as observed earlier on, this can make a huge difference; If too high a rate is chosen the Hill-Climber finds difficulty in converging to the global optimum, too low and the Hill Climber instance stagnates and will never have the opportunity to explore, again relying on a lucky start in its climb to find the maximum. For the microbial population, an ideal mutation rate, when merged with the correct crossover operator and a good crossover rate meant more than a third of the population was able to converge on the global maximum. This success reinforces the idea that a problem specific selection of characteristics is paramount to the algorithm's performance and only with these three options in harmony does the algorithm achieve peak performance.

5 Bibliography

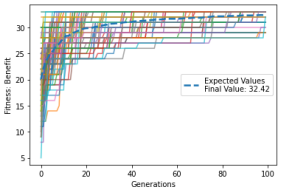
- Gen, M., Lin, L. (2007). Genetic algorithms. Wiley Encyclopedia of Computer Science and Engineering, 1-15.
- Harvey, I. (2009, September). The microbial genetic algorithm. In European conference on artificial life (pp. 126-133). Springer, Berlin, Heidelberg.
- Wilson, L. (2014). Re: What is the role of mutation and crossover probability in Genetic algorithms?. Retrieved from: https://www.researchgate.net/post/ What_is_the_role_of_mutation _and_crossover_probability_in_Genetic_algorithms
- Črepinšek, Matej Liu, Shih-hsi Mernik, Marjan. (2013). Exploration and Exploitation in Evolutionary Algorithms: A Survey. ACM Computing Surveys. 45. Article 35. 10.1145/2480741.2480752.
- Back, T. (1994). Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 57-62 vol.1.
- Umbarkar, Dr. Anantkumar Sheth, P.. (2015). CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. ICTACT Journal on Soft Computing (Volume: 6 , Issue: 1). 6. 10.21917/ijsc.2015.0150.

Appendices

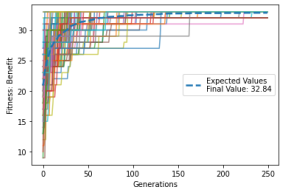
A Growth of separate HCs populations running for an increasing amount of generations.



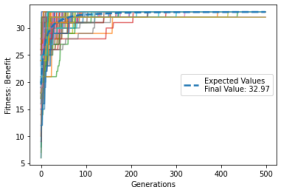
(a) HC Generations: 50
Population Average: 31.97



(b) HC Generations: 100
Population Average: 32.42



(c) HC Generations: 250
Population Average: 32.84



(d) HC Generations: 500
Population Average: 32.97

Figure 19: Incrementing Generations at default, ideal Mutation Rate: 0.3

B Parameters for Figure 9 under 3.3 MGA Methods:

Mutation Rate: 0.25, Crossover Operator: One Point
Tournament: 2000, Population Size: 100

C Parameters for Figure 11, 12 under 3.3 MGA Methods:

Mutation Rate: [0,1,step=0.1], Crossover Operator: One Point
Tournament: 2000, Population Size: 100

D Parameters for Figure 17, 18 under 3.4.2 Exploitation:

Mutation Rate: 0.0015, Crossover Operator: All[-1:2]
Tournament: 2000, Population Size: 100