Hill-Climbing vs Simulated Annealing
in minimizing functions

Robert Gaina

November 2nd, 2020

Abstract

In optimization problems we are looking for either the largest or the smallest value that a function can take. It is difficult doing this manually for some of the functions so there are some heuristic algorithms used to solve this problem like Hill-Climbing and Simulated annealing. This paper compares these methods used to solve minimizing certain functions.

1 Introduction:

The goal of this report is to compare non-deterministic methods of minimizing a function. There will be presented non-deterministic methods based on local search optimization for finding the global minimum for four different functions with different characteristics.

1.1 Motivation:

Hill-Climbing and Simulated Annealing are non-deterministic searching techniques that can be used for solving optimization problems. For some functions the optimization process cannot be done in a reasonable time using deterministic algorithms but can be aproximated with non-deterministic aproaches.

1.2 The problem:

Minimizing a function is the process of finding a value within an allowed range of parameters for which the function has the lowest value possible.

2 Methods:

2.1 Algorithms

In order to find the minimum of a function there have been used non-deterministic algorithms. These were applied on four different functions that vary from the number or distribution of local minima to the domain of them.

The **Iterative Hill Climbing** method searches for a solution by iteratively saving the best option from a set of candidates. These candidates always result from an improvement operation

performed on an initial randomly selected point. Based on the type of the improvement function the search is either improved with the **best** evolution or the **first** evolution.

Best-Improvement Hill Climbing aproach will select a random candidate solution and evaluate it. After the selection step we evaluate all the neighbours of the current candidate and select the best one out of them (the neighbour with the smallest value of the function). This process of improving repeats until we arrive in a state where the evaluation of all the neighbours is either smaller or equal than the current solution. At this point we are into a local minimum or in a plateau section. By repeating this process of generating local minima for a number of iterations and remembering the lowest minimum found we are able to aproximate the solution.

The only difference between **First-Improvement Hill Climbing** aproach and the previously described one is that in the process of verifying the neighbours we select the first one found.

Since these methods only allow downhill moves in guarantees that the solution is either a local or global minimum. In order to avoid getting stuck into a local minimum **Simmulated Annealing** could be a technique to use that allows uphill jumps. The concept of annealing comes from metallurgy; where metals are heated to a high temperature and cooled at a controled rate to avoid defects in the structure. The idea behind this implementation is that a high temperature allows jumps out of local maxima sacrificing a better solution at a moment for a better one later in the evolution process. The algorithm will start with a random candidate solution and a temperature compared to Hill Climbing where we randomly generate a number of random candidates. There are two conditions that should be estabilished: the **termination-condition**(inner-loop) and the **halting criterion**(outer-loop). The halting criterion is usually related with the temperature; it starts with the initial value of the temperature that decreases at each iteration towards an equilibrium state. The three main elements of this loop are the initial temperature, the cooling rate and the stopping criterion. In the inner-loop a random neighbor of the solution is evaluated. If it is better it is saved as the new solution and if not we are evaluating its chances of replacing the current value. Usually the acceptance criteria is chosen such that the probability of accepting a worse solutionn decreases with the temperature.

2.2 Implementation

The implementation of the methods follows the guidelines from 2.1 section with these additions:

- The solutions are saved in the binary format
- Whenever we evaluate a solution we are converting it to their decimal value
- The notion of neighbour means neighbour in Hamming space of distance one
- Neighbour evaluation is done in the following order:
  (First $\longrightarrow$Last (component) and for each *component* Most significant bit$\longrightarrow$ Least significant bit)
- For Hill Climbing aproaches we are iterating through 10.000 local minima.

For Simulated Annealing the conditions chosen are:

- the initial temperature is 100
- the halting condition is $temperature < 10e - 8$

- temperature decreases by 0.5% at every outer loop iteration
- inner loop is an iteration of 10.000 steps

3 Experiments

The experiments were conducted on the following 4 functions:

1. DeJong's function
2. Schwefel's function
3. Rastrigin's function
4. Michalewicz's function

For all the functions the algorithms were executed 30 times with a precision of 5 decimals. The search spaces chosen were 5, 10 and 30.

3.1 DeJong Function

$$f(x) = \sum_{i=1}^{n} x_i^2 \qquad -5.12 \leq x_i \leq 5.12$$

Global minimum: $f(x) = 0, x(i) = 0, i = 1 : n$.
DeJong's function is also known as a sphere model. It is continuos, convex and unimodal.

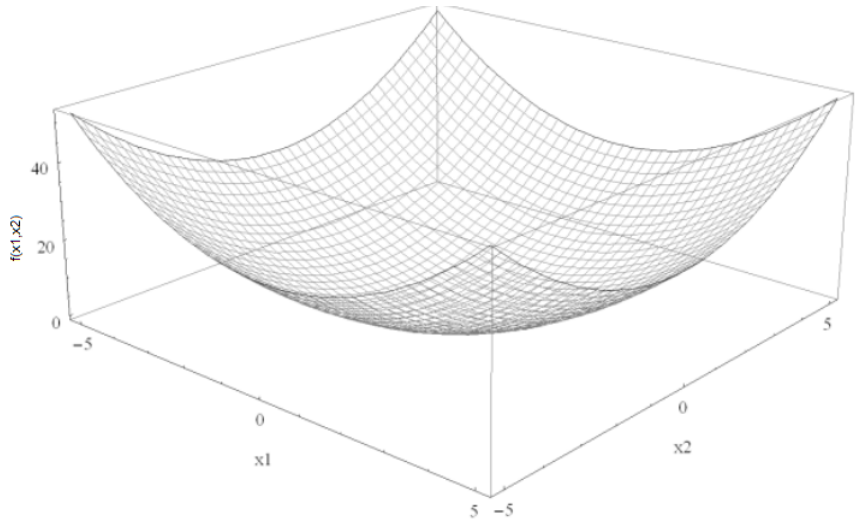

Figure 1: DeJong function 1 graph for n=2

3.1.1 Best Improvement

Since this function has no local minima and only one global minima the Best Improvement method is slow on this function. On a big space there is a lot of time wasted finding the best neighbour to be picked (eg. For n=30 it would have to check 599 neighbours at every iteration)

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	0	0	0	0	21.836	20.8	23.351
10	0	0	0	0	120.381	117.643	123.974
30	0	0	0	0	2182.296	1943.08	2284.86

Table 1: DeJong function 1 result on best improvement
*Actual values were smaller than 10^{-5} so they were approximated to 0

3.1.2 First Improvement

As we can see the first improvement aproach saves a lot of time by picking the improved candidate faster.

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	0	0	0	0	11.197	10.326	12.065
10	0	0	0	0	65.630	61.093	79.351
30	0	0	0	0	1034.764	989.381	1102.63

Table 2: DeJong function 1 result on first improvement
*Actual values were smaller than 10^{-5} so they were approximated to 0

3.1.3 Simulated Annealing

Compared to the Hill Climbing method where the minima was computed at the very first iteration the Simulated Annnealing doesn't come to a stop point in a local minima it rather ends with an aproximation when the temperatures arrives at a stability point.

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	0	0	0	0	38.727	36.646	44.175
10	0	0	0	0	53.113	51.968	54.348
30	0	0	0	0	94.423	91.848	96.619

Table 3: DeJong function 1 result on Simulated Annealing
*Actual values were smaller than 10^{-5} so they were approximated to 0

3.1.4 Remarks

Based on the results, it seems that for unimodal functions there is almost no reason to apply a Best Improvement Hillclimber since it is most of the time it is slower than First Improvement that saves a lot of time by picking the first improved neighbour. Simulated Annealing also returns an acceptable result for our precision in an impressive time compared to both other methods. If we are looking for an exact result for this type of function First Improvement seems to be the best choice and if we only want a fast aproximation we can use the Simulated Annealing technique with a fast cooling rate.

3.2 Schewefel's function

$$f(x) = \sum_{i=1}^{n} -x_i * sin(\sqrt{|x|}) \qquad -500 \leq x_i \leq 500$$

Global minimum: $f(x) = -n * 418.9829, x(i) = 420.9686, i = 1 : n$

Schwefel's function global minimum is geometrically distant over the parameter space, from the next best local minima. Because of this the search algorithms are potentially prone to converge in the wrong direction.

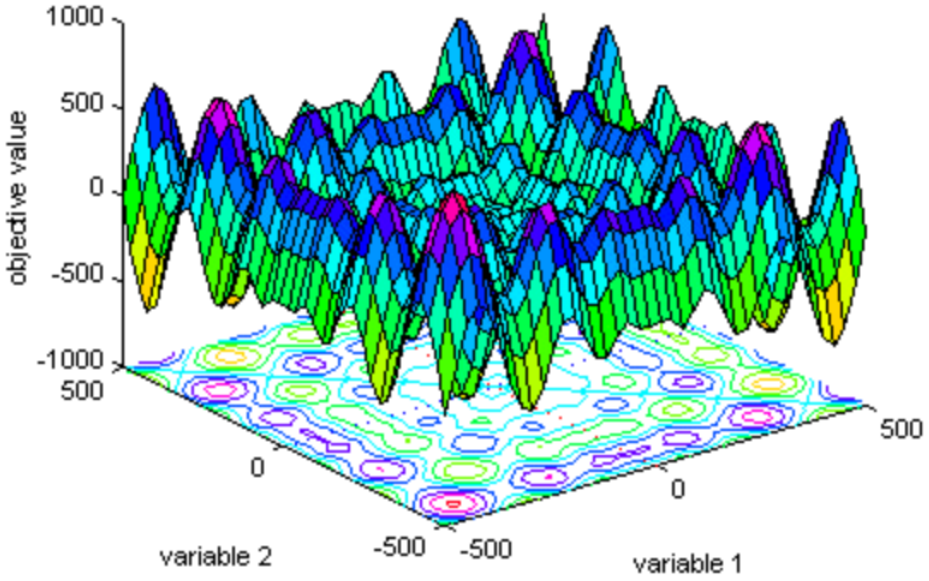

Figure 2: Schwefel's graph for n=2

3.2.1 Best Improvement

The Best Improvement approach seems to find the global minimum in a small search space (eg. for n=5 out of 30 runs 25 returned the global minimum). However when we are increasing the number of dimensions the result will rather be an aproximation of the minima (eg. for n=30 the global minima should be -12569.487 while the result obtained is a 7% bigger value)

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	-2094.893	0.03790	-2094.91	-2094.81	45.932	42.382	64.042
10	-4154.385	27.07185	-4189.52	-4071.08	276.169	267.5	287.133
30	-11536.53	160.1539	-11794.6	-11344.6	5470.997	5384.02	5555.07

Table 4: Schwefel's function results on Best Improvement

3.2.2 First Improvement

When using the First Improvement approach the aproximation is even worse than previous method. A cause of these results could be cause by the order of checking the neighbours. (as mentioned in 2.2 the method used is negating all the bits from the first component from the most significant to the least one and repeating this for every component until we find a better solution)

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	-2094.765	0.07578	-2094.81	-2094.6	26.204	24.598	27.852
10	-4059.537	55.01682	-4155.18	-3968.48	145.746	142.284	149.966
30	-10986.56	75.23497	-11093.4	-10896.6	2972.681	2839.15	3632.21

Table 5: Schwefel's function results on First Improvement

3.2.3 Simulated Annealing

For this type of function Simulated Annealing gives the best results since it favours jumps from a local minima. Even if it is still an aproximation it is a better one then those computed by the previous methods

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	-1803.254	152.9006	-2060.68	-1384.89	40.332	38.334	43.402
10	-3525.621	191.2386	-3849.34	-2937.74	58.431	56.83	60.721
30	-11141.36	436.0493	-11822.4	-10010	118.64	116.062	120.705

Table 6: Schwefel's function results on Simulate Annealing

3.2.4 Remarks

When using a function as Schwefel's with many local minima in a short space it is hard for optimization algorithms to find the solution. With a standard deviation increasing with the size dimension other aproaches should be used for solving this optimization problem.

3.3 Rastrigin's function

$$f(x) = 10 * n + \sum_{i=1}^{n} (x_i^2 - 10 * cos(2 * \pi * x_i)) \qquad -5.12 \leq x_i \leq 5.12$$

Global minimum: $f(x) = 0, x(i) = 0, i = 1 : n$

Rastrigin's function is multimodal and the location of the minima are distributed. The value of the function increases with the distance from the origin so the global minimum can only be obtained in the origin.

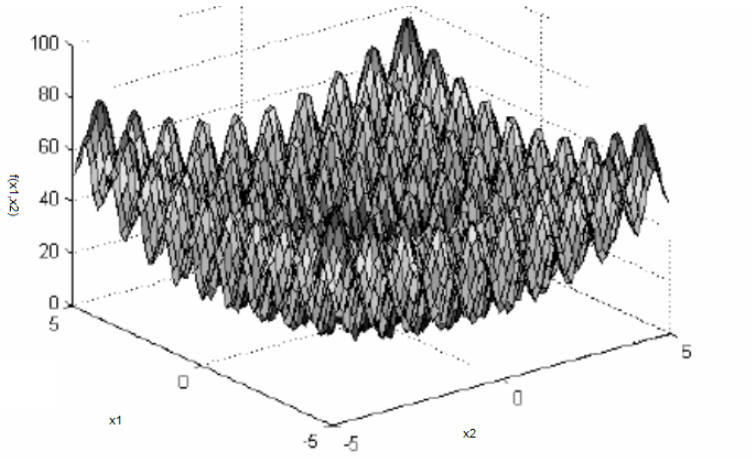

Figure 3: Rastrigin graph for n=2

3.3.1 Best Improvement

As expected from the previous function the error of the minima computed increases with the number of dimensions.

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	0	0	0	0	19.642	18.361	28.642
10	2.89747	0.47946	2.2308	3.23081	116.143	113.059	157.081
30	23.49357	0.0858	23.3647	23.5488	2020.405	1857.17	2571.65

Table 7: Rastrigin's function results on Best Improvement
*Actual values were smaller than 10^{-5} so they were approximated to 0 for size=5

3.3.2 First Improvement

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	0.49748	0.50598	0	0.99496	10.631	9.717	16.072
10	3.73195	0.4475	3.46663	4.46159	59.679	56.547	83.926
30	32.17243	0.31623	31.7638	32.409	1069.546	1006.45	1321.43

Table 8: Rastrigin's function results on First Improvement

3.3.3 Simulated Annealing

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	7.72053	4.22126	0	16.9143	35.794	34.313	37.131
10	15.1123	4.40189	7.21065	27.6269	50.687	49.478	52.211
30	48.21071	9.55763	29.7761	66.332	93.749	91.006	107.478

Table 9: Rastrigin's function results on Simulated Annealing

3.3.4 Remarks

For a function where the local minima values are directly proportional with the distance from the local minima it seems that HillClimbing techniques computed better results overall and the standard deviation seems to be inversely proportional with the dimensions for this method. Even if the Simulated Annealing remains faster at bigger dimension the results becomes worse and worse with an increasing error.

3.4 Michalewicz's function

$f(x) = -\sum_{i=1}^{n} sin(x_i) * (sin(\frac{i*x_i^2}{\pi}))^{2*m}$, $i = 1 : n, m = 10, 0 \leq x_i \leq \pi$

Michalewicz's function is a multimodal function with n! local minima where m defines the steepness of the valleys and ridges. Since the global minima becomes difficul to search when m reaches a larger value its recommended value is 10.

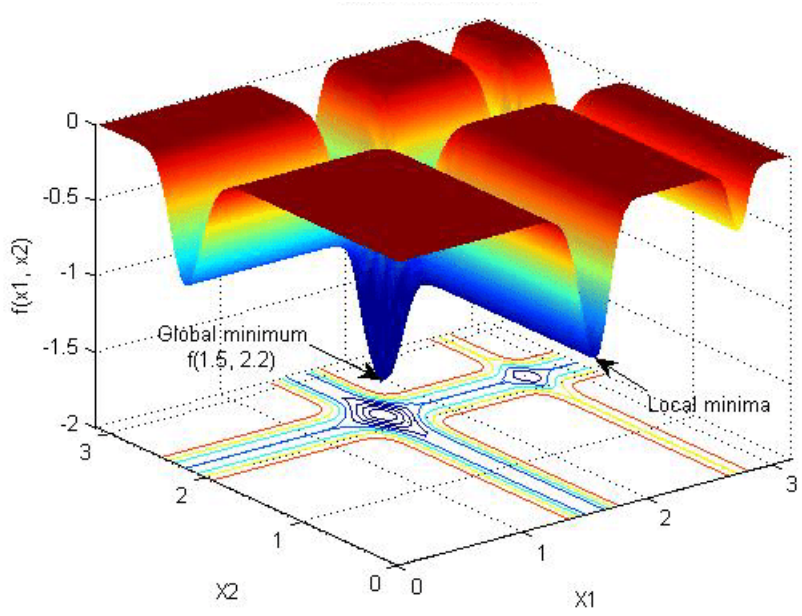

Figure 4: Michalewicz's function for n=2

3.4.1 Best Improvement

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	-4.68764	0.00003	-4.68766	-4.68759	27.72	25.415	29.132
10	-9.52279	0.07680	-9.61507	-9.38802	165.7	161.111	213.811
30	-27.316	0.12857	-27.4798	-27.1037	3164.465	3112.06	3334.44

Table 10: Michalewicz's function results on Best Improvement

3.4.2 First Improvement

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	-4.68756	0.00018	-4.68766	-4.68692	15.714	14.512	20.485
10	-9.40519	0.05764	-9.49299	-9.29939	98.57	95.138	127.377
30	-26.86171	0.22096	-27.1615	-26.6019	1978.305	1958.54	1996.58

Table 11: Michalewicz's function results on First Improvement

3.4.3 Simulated Annealing

Size	Mean	σ	Min	Max	Avg Time(s)	Min Time(s)	Max Time(s)
5	-4.22592	0.28523	-4.68093	-3.58642	49.915	49.037	50.858
10	-8.2083	0.53272	-9.15585	-6.81808	77.68	76.443	78.595
30	-25.2042	0.96744	-26.9699	-22.8067	171.097	168.14	180.606

Table 12: Michalewicz's function result on Simulated Annealing

Function	Algorithm	Size	Expected Result	Actual Result	Error(Act-Exp)
DeJong	HCB	5	0	0	0
	HCF			0	0
	SA			0	0
	HCB	10	0	0	0
	HCF			0	0
	SA			0	0
	HCB	30	0	0	0
	HCF			0	0
	SA			0	0
Schwefel	**HCB**	**5**	**-2094.9145**	**-2094.91**	**0.0045**
	HCF			**-2094.81**	**0.1045**
	SA			**-2060.68**	**34.325**
	HCB	**10**	**-4189.829**	**-4189.52**	**0.309**
	HCF			**-4155.18**	**34.649**
	SA			**-3849.34**	**340.389**
	HCB	**30**	**-12569.487**	**-11794.6**	**774.887**
	HCF			**-11093.4**	**1476.087**
	SA			**-11822.4**	**747.087**
Rastrigin	HCB	5	0	0	0
	HCF			0	0
	SA			0	0
	HCB	10	0	2.2308	2.2308
	HCF			3.46663	3.46663
	SA			7.21065	7.21065
	HCB	30	0	23.3647	23.3647
	HCF			31.7638	31.7638
	SA			29.7761	29.7761
Michalewicz	**HCB**	**5**	**-4.68765**	**-4.68766**	**0.00001**
	HCF			**-4.68766**	**0.00001**
	SA			**-4.68093**	**0.00672**
	HCB	**10**	**-9.66015**	**-9.61507**	**0.04508**
	HCF			**-9.49299**	**0.16786**
	SA			**-9.15585**	**0.5043**
	HCB	**30**	**-29.63088**	**-27.4798**	**2.15108**
	HCF			**-27.1615**	**2.46938**
	SA			**-26.9699**	**2.66098**

Table 13: Comparison of minimas obtained vs expected results (out of 30 runs each)

4 Discussion

The problem of optimization for a function becomes hard for the previously used algorithms based on certain criteria like: function dimension, the number and distribution of local minima, the precision or even the neighbour generation method.

For smaller dimensions any of the aproaches studied generate a great aproximation. However, if we want to expand the dimension, there are two main criteria based on which we can choose

a method: **precision** and **time**. If we want a better precision and we are willing to spend more time for the computation period, Best Improvement Hill Climbing might return the best values meanwhile for a fast aproximation Simulated Annealing with a fast cooling rate seems to be a faster way. To meet in the middle First Improvement Hill Climbing gives decent results almost two times faster then the Best Improvement method.

For the **Hill Climbing** method a higher number of iterations might return a better value but it mostly depends on the distribution and number of local minima. For this experiment the number of 10.000 iterations proved to be too much for functions that have only one local and global minima like DeJong function, where the best result was computed in the first iteration and too low for a function with several closed local minimas as the Schwefel function. Another factor that must be kept in mind is the method of improving the solution. For the Best Improvement the process of generating neighbours is less important since all must be evaluated anyway but there is a huge impact when talking about First Improvement. Depending on what is the order of evaluating the the neighbours (eg. most important bit from each component first to the least...(1), least important bit from each component first to the first...(2), try to modify the first/last component before going to the next one...(3) etc...) the algorithm might return different values. Either way any implementation of Hill Climbing only allows **downwards** movements.

When talking about **Simulted Annnealing** technique there are several things to keep in mind. Firstly, the halting-criterion and termination condition will decide the time and number of evaluations required. The halting criterion used in this experiment ($t < 10e^{-8}$) where the initial temperature of 100 was dropping by 0.5% at each outer loop iteration in combination with a inner loop iteration of 10.000 steps proved to be a fast but quite imprecise method. Another important factor is the notion of temperature and it's evolution. The initial temperature needs to be high enough to guarantee some time for the solution evaluation and the cooling process has to be slow enough to guarantee an equlibrium state at the final steps of the iteration. Lastly, the temperature has a direct influence on the acceptance criteria for worse intermediate solutions. This must be balanced such that, the algorithm will accept worse solutions at high temperatures while lowering this probability later. This way the algorithm will mostly move **downwards** but **not** necessarily **always**.

There are also some factors that directly have an influence upon all of the algorithms tested. Since all the algorithms are non-deterministic the notion of random needs to be defined precisely. In common, the notion of randomness is the apparent lack of patterns or predictability in a certain event. Since there is no actual way of generating random numbers and we are actually using pseudo-generations we must choose a way that avoids any pattern. The algorithms need to use a method of generating random numbers with equal probability. For this experiment there was used the rand() function with a different seed at each run. Other methods could include the Mersene Twister algorithm, Xorshift, WELL... The notion of neighbour also has a direct influence. For this experiment the numbers were represented in binary so a neighbour is denoted as a number in Hamming distance equal with 1 rather than the clasic decimal representations. The precision chosen also dictates the size of your solution and implicitily the number of neighbours to be evaluated as well as the space complexity required.

By looking at the **function** we want to evaluate we can sometimes see what approach will be the best for our situation. For example a function with only one global minimum it will be better to use First Improvement Hill Climbing since it can't get stuck in a local minima(because there isn't one) and it quarantees no uphill movement. For other functions that have a high distribution and number of minimas however none of the methods seem to generate acceptable solutions for a big dimension. There are also some functions where the random jumps out of local minimas from

Simulated Annealing might cause better results and even in a shorter time as we could see for the Schwefel function for size 30.

5 Conclusion

This paper compared the Hill Climbing and Simulated Annealing and their capacity of solving the problem of minimizing certain functions. This showed the strong(easy to implement) and weak points(long time of computation, inneficient results, abstract criterias) of each technique as well as underlying problems in computer science like scalability and the notion of randomnness. There were used two istances of Hill Climbing algorithm: First and Best Improvement and a Simulated Annealing algorithm with linear cooling time. From this experiment we can conclude that minimizing a function becomes a hard problem for these algorithms and alternatives solutions should be used for certain searching spaces. The poor performance of these non-deterministic algorithms at an increased dimension size shows that not even these can aproximate a decent solution in reasonable time. Further research are needed in studying the optimal annealing schedule or adaptive heuristic methods.

6 Bibliography:

- https://profs.info.uaic.ro/~eugennc/teaching/ga/
- http://www.geatbx.com/docu/fcnindex-01.html
- https://kenndanielso.github.io/mlrefined/blog_posts/5_Zero_order_methods/5_0_Motivation.html
- https://academicjournals.org/journal/AJMCSR/article-full-text-pdf/EC495AE4687
- http://rbanchs.com/documents/THFEL_PR15.pdf
- https://www.researchgate.net/post/how_can_I_determine_temperature_parameters_accurately_for
- https://www.seas.upenn.edu/~cis391/Lectures/informed-search-II.pdf
- https://en.wikipedia.org/wiki/Randomness
- https://bib.irb.hr/datoteka/653421.Eurocon2013_ICT_S13_2.pdf
- https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/
- https://www.geeksforgeeks.org/multithreading-in-cpp/
- http://www.cplusplus.com/reference/thread/thread/get_id/
- Hill Climbing Algorithm & Artificial Intelligence - Computerphile