• U C •

FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE DE COIMBRA

Performance and Behaviour Analysis of Stochastic and Heuristic Algorithms

PL1

Oleksandr Yakovlyev, 2015231448, oy@student.uc.pt

Ricardo Guedes, 2016241802, uc2016241802@student.uc.pt

Ricardo Santos, 2013143324,uc2013143324@student.uc.pt

Abstract

Using different strategies of optimization, we compare suggested search algorithms (random search, hill climbing and simulated annealing), analyzing experiment results. Their performance and scalability are evaluated based on tests with different initial conditions (parameters) and seeds for each given map. We further try to optimize simulated annealing using different cooling schedules.

Keywords

Simulated annealing, search algorithm, heuristic, stochastic, cooling schedule, optimization.

Table of Contents

1. Introduction

In this study we will address a search problem using different search algorithms in various conditions. Similar to classic problems such as the traveling salesman, D31 is going to be our study subject in order to evaluate both the behavior and performance of said algorithms. In order to understand and further investigate them, we experimented with stochastic (random search and simulated annealing) and heuristic algorithms (hill climber) in different conditions (iterations, seeds, maps). Within the simulated annealing algorithm, we used distinct cooling functions (logarithmic, linear, geometric, sinusoidal) with the goal of optimizing temperature variation.

We evaluated the average cost of different seeds to analyze the performance and plotted some tests to obtain conclusions regarding their behavior over time and the effects of their cooling schedule.

2. Implementation

In the following experiment we applied three different search algorithms:

1. Random Search (already implemented in the given code);
2. Hill Climbing;
3. Simulated Annealing.

In the Simulated Annealing implementation we tested three annealing schedules:

1. Logarithmic;

$$T_i = \frac{\alpha}{log_{10}(1+i)} \qquad (1)$$

2. Linear;

$$T_i = T_{init} - \alpha t \qquad (2)$$

3. Geometric;

$$T_i = T_{init}\alpha^i \qquad (3)$$

We also tested our own cooling function:

$$T_i = T_{init}\alpha^i + \frac{sin(\alpha x)x+x}{d} \qquad (4)$$

3. Experimental Setup

For each map, except for the simple ones like *NoObstacles* and *ObstaclesSmall*, we tested each algorithm with 100, 1000, 10.000 and 100.000 iterations.

Map	# Boxes
1. NoObstacles	3
2. ObstaclesSmall	6
3. ReturnTo2b	9
4. ObstaclesSmallManyBox	12
5. ReturnTo2bHarder	17
6. Bomberman	20
7. PacManOriginal	24

Table 1: Maps and corresponding complexity expressed in the number of boxes the robot must collect.

To decide an appropriate initial temperature in the Simulated Annealing algorithms we first created table 1 to gain some insight as to how the temperature, *T,* and the cost difference, ΔC, impact the probability of choosing a worse solution and possibly enabling the escape of a local maxima.

T / ΔC	0.1	1	2	5	10	20
-50	0.00	0.00	0.00	0.00	0.67	8.21
-20	0.00	0.00	0.00	1.83	13.53	36.79
-10	0.00	0.00	0.67	13.53	36.79	60.65
-5	0.00	0.67	8.21	36.79	60.65	77.88
-4	0.00	1.83	13.53	44.93	67.03	81.87
-3	0.00	4.98	22.31	54.88	74.08	86.07
-2	0.00	13.53	36.79	67.03	81.87	90.48
-1	0.00	36.79	60.65	81.87	90.48	95.12
0	100.0	100.0	100.0	100.0	100.0	100.0

Table 2: Probability of choosing a worse solution given the Temperature *T* and the cost difference ΔC.

After analyzing the table we experimented with a variety of temperature ranges and parameter values. Firstly we explored these combinations almost randomly, but we quickly realized that lower temperatures (50 and below) and slower cooling scheduling worked better.

After that we iterated in a hill-climber-like method: keeping all the parameters fixed but one, we would run tests to find the best value.

We defined each test as a combination of map, algorithm, iteration count, starting temperature, and parameters (in the case of Simulated Annealing). Given that all these tests have a stochastic component, we ran each test ten times, each in a different seed ranging from 0 to 9. To compare the algorithms we then used the averages of said tests.

In addition, we also used a plotting tool (in the supplementary material) to visualize our cooling schedules beforehand.

All the tests were run in the unity project solution. We made some small quality of life features that simplified and enabled faster parameterization of each Simulated Annealing function directly in the editor. Still, the tests were time-consuming due to the simple fact that every test must be started manually and the simulation needs to be reset afterward.

All the tests and additional tables can be found in our spreadsheet linked in the supplementary material.

4. Experimental Results and Analysis

In this section we will broadly analyze the results. Firstly let's look at the main table, comparing side by side every algorithm.

# Map	# Iter.	Algorithm				
		Rand	HC	Linear	Log	Geo
1	100	17	17	17	17	17
2	100	29.2	29.8	28.9	29.4	29
	1k	25	29.8	25.4	27.8	25.5
3	100	85.5	68.3	71.8	72.8	73.5
	1k	74.1	66.7	69.1	66	67.7
	10k	66.6	66.7	64.3	64.7	63.6
	100k	63.9	66.7	64.4	64.5	63.8
4	100	58.5	44.6	47.3	46.2	46.3
	1k	49	35.1	35.3	35.6	32.9
	10k	46.7	35.1	31.4	33.6	31.5
	100k	44.4	35.1*	31.5	34.1	31.4
5	100	163	123.4	122.5	121.7	119.6
	1k	145.4	89.9	96.2	94.9	95.8
	10k	133.6	88.3	84.2	86.1	83.2
	100k	127*	87*	79.5	85	80.2
6	100	156	114.8	121.7	121.2	121.5
	1k	143.8	89.4	87.5	91.4	89.3
	10k	134.1	87.6	80.5	82.6	80.5
	100k	118.5*	90.5	78	80.4	78.2
7	100	417.2	322.4	351.1	363	347.9
	1k	396	249.4	255.1	264.6	247.2
	10k	379.6	235.8	218.6	229.9	221.8
	100k	355.4	235.8	211.8	227.8	211.4

Table 3: Table with the best cost for each map. The names were abbreviated for compactness. Rand - Random, HC - Hill Climber. Linear, Log, and Geo correspond to the Simulated Annealing algorithm with the Linear, Logarithmic (base 10), and Geometric cooling schedules, respectively.

As we can see, in the first map - with only three boxes and no obstacles - with 100 iterations every algorithm reached the optimal solution. We could have done this with as little as 10 as there are only 6 possible combinations. There's not much we can extract from this map.

In the second map (6 boxes), at 1.000 iterations, we have an interesting but logical result: the Random algorithm can find the optimal solution, but the Hill Climber and the other Simulated Annealing algorithms fail to do so. This is because at 6 boxes, we have 6! = 720 possible combinations, which is less than the 1.000 iterations. This means that the solution space is small enough for the random search to be able to find, by luck, the best sequence, but big enough to have many local maxima in which the rest of the algorithms eventually get stuck.

The third map is also quite interesting. At 100, 1.000, and 10.000 iterations, the Random search is beaten by every other search, but at 100.000 every search apart from S.A. Geometric performs worse, and even the Geometric is better only by 0.1 in average, which is negligible. The HC search is easy to explain: it gets stuck in local maxima early in the search phase (in fact, it gets stuck in the same iteration in the last three tests, yielding the same value of 66.7).

But why do the SA algorithms perform better than the HC but worse than Random?

We have two possible explanations:

1) The parameters were wrongly configured

2) As the number of iterations approaches the total number of possibilities (in this case, 100k iterations vs 360k possibilities, roughly ¼) random search becomes more viable because it does not have the disadvantage of getting stuck at local minima. It ends up exploring more possibilities than the other algorithms.

The rest of the maps all follow the same tendency: Random search is worse in every test, Hill Climber is better at 100 and 1.000 iterations but pales from thereon. In other words all three Simulated Annealing searches perform better with 10.000 and 100.000 iterations. This can be visually perceived from the following figures:

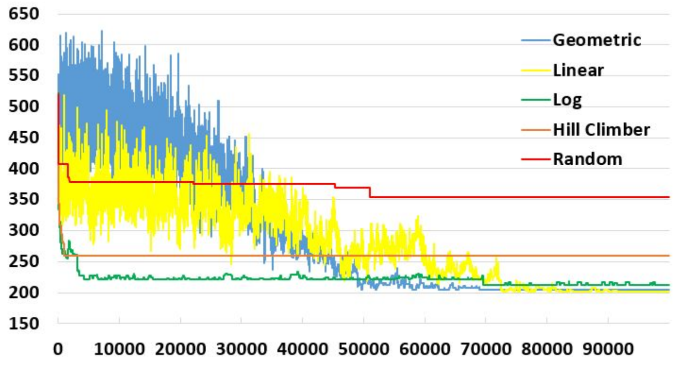

Figure 1: Plot of best cost (y-axis) per iteration (x-axis). Data extracted from the logs generated for the *PackmanOriginal* map at 100k iterations, data chosen from the same seed (8) of the best performing configuration.

We can also see that the SA algorithms, namely the Linear and Geometric, oscillate immensely within the first 30 thousand iterations. The Logarithmic on the other hand drops very quickly and stays there, being slightly overrun by the Geometric and Linear near the end of the simulation.

Bellow, we have a similar graph but on a smaller scale:

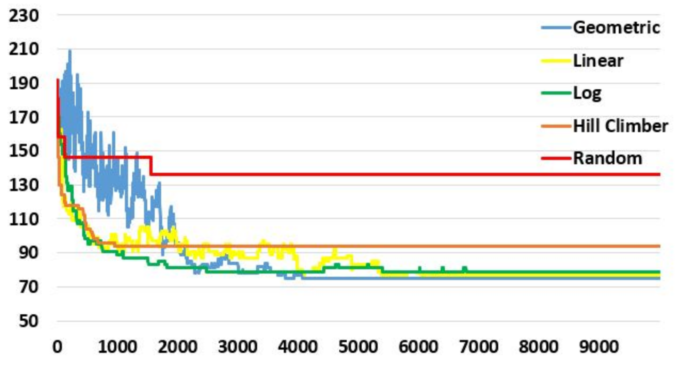

Figure 2: Same as the previous figure 1 but for the *Bomberman* map at 10k iterations, seed 5.

We can see that the Geometric and Linear functions have much heavier oscillations in the cost than the Logarithmic - which is expected (table 2 and figure 3) as the initial temperature is higher, enabling wider leaps in terms of choosing a worse solution.

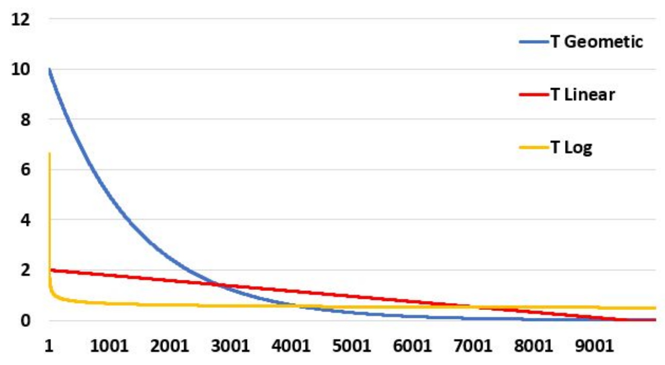

Figure 3: Temperatures in reference to figure 2.

In both figure 1 and figure 2 the difference is clearly visible in the effectiveness of the random search and Hill Climber search when the solution space is magnitudes bigger than the iteration count.

There was also another cooling schedule that we tried to apply but because of time constraints and a few unpromising tests we ended up not including it in our final results. In any case here are some figures to illustrate how it compares to the other SA schedules.

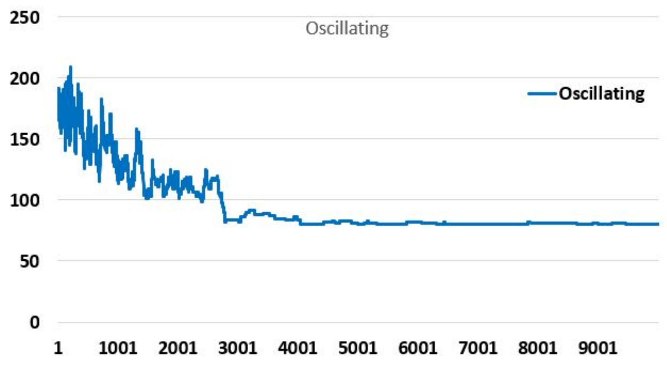

Figure 4: Oscillating function in the same conditions as in figure 2.

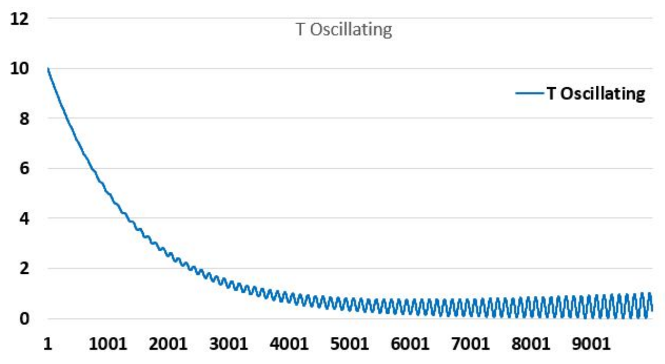

Figure 5: Oscillating function temperature.

5. Discussion

In this section we compare, in detail, each algorithm and state our findings, namely to answer the following questions: which algorithm is the best? Is there one algorithm to rule them all? Is the performance dependent on the scale of the problem? What are the best parameters and initial temperatures for the SA cooling schedules?

I. Algorithm Comparison

In this study, our robot is like the traveling salesman, trying to find the shortest path that passes through all the boxes. In this model, our solutions are sequences representing the order in which D31 picks up his targets. Usually neighbor solutions are also close to each other in terms of cost - meaning we could take advantage of this property. Hill Climbing and Simulated Annealing work on this principle as they try to generate a neighbor solution, hoping it is better than the current one. The main problem with this approach is that, by repeatedly choosing the best neighbor, we eventually end up with the best one around - but this doesn't mean we have found the best possible solution. We have just reached the local maxima, which could be much worse than the global, best solution possible. On the other hand, Random search is just like the name implies, random - it just generates solution after solution, always saving the best one found yet. In most domains, random search is usually the worst algorithm possible (imagine a random sorting algorithm). It is simple to implement - but, in its simplicity, this type of search is limited because it cannot exploit the properties of the environment, as the Hill Climbing and Simulated Annealing do.

As we were making these tests we found that even though Random Search is simple it's often a sensible choice, in the same manner, that Hill Climber can also perform really well, depending on the scale and complexity of the problem. For example, if we were making a game with many simple independent bots that needed to react quickly, RC or HC would be enough.

Now, the SA algorithms clearly have an advantage over more uncertain environments. Independently of the cooling schedule, with enough iterations these showed better results. Simulated Annealing can be seen as a compromise between the randomness that doesn't care about local maxima and the informed search that chooses the best available neighbor. The tricky part is getting the right parameters. If the temperature doesn’t cool off at the right time, the algorithm just accepts the next solution without question, often performing worse than random search. If it cools off too soon, it becomes a HC, never reaping the advantages of the randomness that higher temperatures allow.

To summarize: there is no best algorithm, it’s dependent on the problem. One example we found out during testing was: SA would sometimes perform worse than HC. Upon examining the situation further, we reached a conclusion: for that specific map, the best option would often be “find the best neighbor”. A fitting analogy would be that, in a mountainous world where each mountain is 500 m tall, if you only have the strength for 100 steps, there’s no point in going down in hopes of finding a taller mountain. Your best bet is climbing the one at your feet.

II. Scalability

We previously mentioned where each type of search performs best. For problems with small solution spaces (or a small number of possible solutions) random algorithms perform better. But very quickly these become infeasible as the number of possibilities grows exponentially.

Hill Climbing is mediocre at best because it often gets stuck. One possible way to overcome this, would be to run Hill Climber over and over again with random seeds, also known as Iterated Hill Climber(which again, due to time constraints, we were not able to implement in this study).

Finally, we have the various flavors of Simulated Annealing. Each one works better than RS and HC for problems of considerable size. They consistently achieved better results and we can safely conclude that these are the algorithms to go by in most real-world scenarios.

III. Best Parameters

Considering the obtained results in the suggested maps, we can verify that every single map has its own tweaks in the parameter, and each function follows specific rules.

For the Logarithmic schedule: this function is very slow. It does converge to zero but it takes too much time. In our implementation the initial temperatures have no meaning as it solely scales of the α parameter. Choosing α as big as 5 mean then the temperature would converge very slowly. The table below illustrates this well.

step \ α	1	2	2.5	5	10
100	0.50	1.00	1.25	2.49	4.99
1k	0.33	0.67	0.83	1.67	3.33
10k	0.25	0.50	0.62	1.25	2.50
100k	0.20	0.40	0.50	1.00	2.00
1M	0.17	0.33	0.42	0.83	1.67

Table 4: The temperatures of the logarithmic function (1) illustrating the slow temperature decay

For the Linear function we had to choose a very slow starting temperature with a very slow decay, otherwise it would be too much random search.

Finally, for the Geometric function, we had more freedom to experiment. The initial temperature could be set to 50 or even 100, which is pretty high compared to the linear and logarithmic (usually 10, 20 at best). The α parameter must be very close to 1 (for example 0.995, 0.99995, depending on the scale) to be useful and converge to 0 at the end.

We can also verify that optimal parameters depend on map characteristics.

	α	Average Cost
Bomberman	2.5	83.9
	5	88.5
PacManOriginal	2.5	237.7
	5	229.9

Table 5: Results and parameters of Logarithmic SA experiments.

Taking table 4 as an example, we can realize that, in the Bomberman map, Logarithmic SA gets better results when α=2.5 is applied, compared to α=5. However in the PacManOriginal map, we get the opposite outcome. This example shows that the average difference in cost, being relatively different for each map, heavily influences the results since it is related to the temperature in calculating the probability.

6. Conclusions

In this study, we tested the performance of different search algorithms in various conditions. In order to do so, we ran around 2000 tests in total, from which we collected data that was further analysed. Having several other studies into consideration and the obtained results, we find that random search and hill climber algorithms are better for a lower number of iterations. Furthermore, simulated annealing based algorithms often yield better results, despite requiring a more sensible parameterization.

Overall, the results met our expectations and allowed us to better understand the differences between the various optimization methods of finding the best or at least a good solution (within reasonable time) where classical deterministic algorithms (like A star) cannot be applied, be it because of time or memory constraints.

7. References

[1] Costa, Ernesto, and Anabela Simões. 2004. *Inteligência Artificial: Fundamentos E Aplicações*.

[2] Gallo, Crescenzio, and Vito Capozzi. 2019. "A Simulated Annealing Algorithm for Scheduling Problems." *Journal of Applied Mathematics and Physics* 7 (11): 2579–94.

[3] Ingber, L. 1993. "Simulated Annealing: Practice Versus Theory." *Mathematical and Computer Modelling* 18 (11): 29–57.

[4] Mahdi, Walid, Seyyid Ahmed Medjahed, and Mohammed Ouali. 2017. "Performance Analysis of Simulated Annealing Cooling Schedules in the Context of Dense Image Matching." *Computación Y Sistemas* 21 (3): 493–501.

[5] Michalewicz, Zbigniew, and David B. Fogel. 2013. *How to Solve It: Modern Heuristics*. Springer Science & Business Media.

[6] Nourani, Yaghout, and Bjarne Andresen. 1998. "A Comparison of Simulated Annealing Cooling Strategies." *Journal of Physics A: Mathematical and General*. https://doi.org/10.1088/0305-4470/31/41/011.

Supplementary Material

Plotting tool - https://www.desmos.com/calculator

Testing Spreadsheet - http://tiny.cc/test_spreadsheet