Simulated Annealing and Ant Colony Optimization Algorithms

University of Malta

Faculty of Information & Communication Technology

Bachelor of Science in Information Technology (Honours) (Artificial Intelligence) 2nd Year

ICS2207 – Machine Learning: Introduction to Classification, Search and Optimisation

Mr Kristian Guillaumier

Francesca Maria Mizzi

I.D. Card Number: 118201(L)

Table of Contents

Simulated Annealing (SA) & Ant Colony Optimization (ACO)

Simulated Annealing

Simulated annealing is an optimization algorithm which is used in order to find the global minimum in a large search space. The name of the algorithm is taken from the annealing process in metals. This process involves the slow and controlled cooling down of a liquidized metal, as opposed to quickly cooling it down, in order for the ions in the metal to settle in a more uniform matter so that the cooled metal is easier to manipulate.

Simulated annealing works by generating different states of the problem and concluding an acceptance probability for the algorithm to accept the current state. The higher the probability (1 being the maximum) the higher the chance of the algorithm accepting that state as having a better result. This happens by having an initial problem state and calculating the total energy or desired result. This initial problem state is then transformed into another state of the problem and the new energy or result is calculated. Should the result of this new problem state be better than the previous state, the acceptance probability is 1 and therefore is accepted by the algorithm and this process is then repeated with the new problem state. When the result of the new problem state is worse than the previous state is when simulated annealing comes into play. The algorithm calculates a probability as to whether or not this new and worse problem state should still be accepted. This is done so that the algorithm can avoid settling into a final result which is not the best it could be.

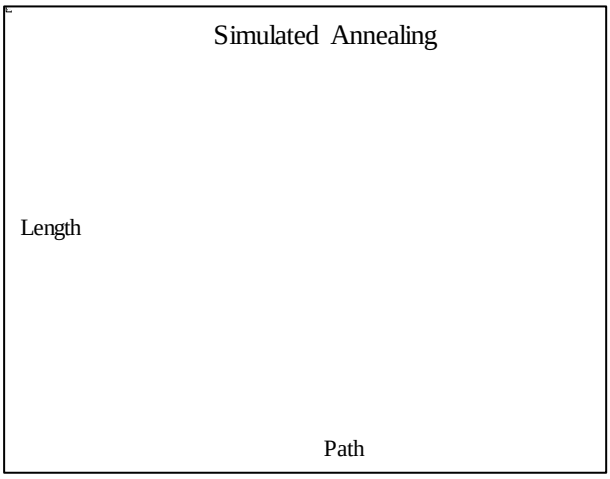

Figure 1 - Graph showing where the calculated shortest path (A) would be if SA completely rejected larger lengths

In the case of the result of the new problem state be worse than that of the previous state, the acceptance probability of this state is calculated by taking the result of the previous state and subtracting from it the result of the new state, dividing it by T and finding the exponential of that result.

$$P(A) = e^{\frac{(L1-L2)}{T}}$$

T is the variable “temperature” which is used to narrow down the possible solutions. Every set number of new states, T is exponentially decreased so that the closer the algorithm is to the best solution, the less likely it is to accept a worse state.

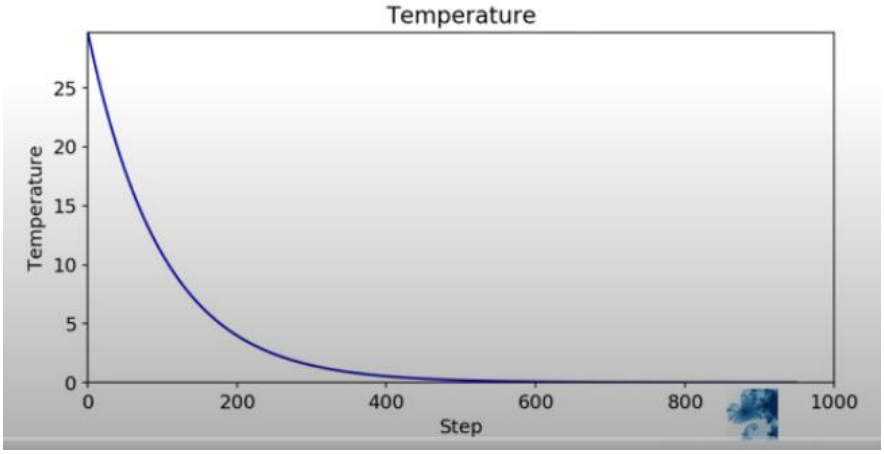

Figure 2 - A graph showing the exponential decrease of the variable temperature

As the algorithm approaches a final result, it is less likely to accept states which are worse than prior states. This is because towards the end of the algorithm, T will be very close to 0 which means that the calculated acceptance probability will be very close to 0.

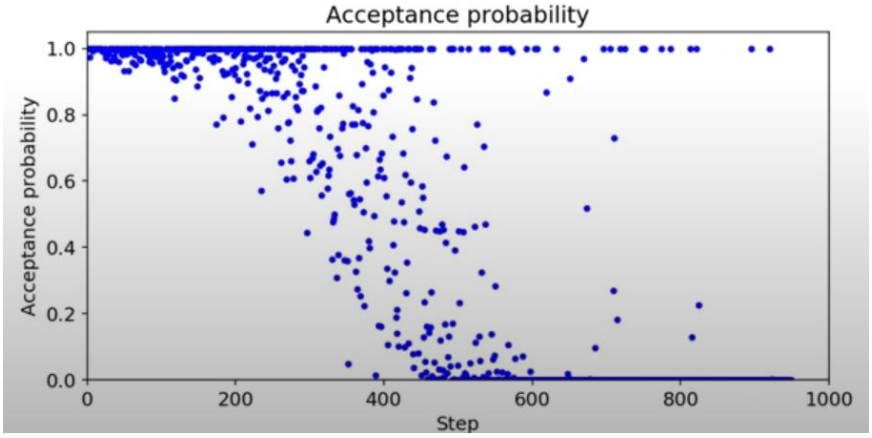

Figure 3 - A graph showing the acceptance probability of paths as iterations take place

In the above graph we can see this in practice. In the above simulation of simulated annealing, we can see that in the first 200 steps, most of the new states have a very high acceptance probability. This is because T will not yet have started decreasing yet. In the next 400 steps, we can see how the acceptance probability is gradually decreasing due to the exponential decrease in T. In the final 400 steps, we can see how most of the states are either immediately accepted (1 acceptance probability) or immediately rejected (0 acceptance probability).

If we take the example of the travelling salesman problem, the states of the problem would be the different possible combination of paths while the desired result will be the lowest possible path length. As each possible path combination is generated, they are compared with the prior combination to see if this new combination requires less length than the old combination. If that is the case, the new paths are accepted. If not, the acceptance probability is calculated to check whether the new combination will still be accepted.

Ant Colony Optimization (ACO)

Ant colony optimization is a swarm-intelligence based algorithm which, through the use of graphs, bases its behaviour on real ants which find the best path for the desired journey. In the case of ant colony optimization, it focuses on the communication between ants through the use of pheromones. Ants make use of pheromones by releasing them as they journey to attract other ants; this pheromone then evaporates over time. The behaviour of real ants is emulated by the virtual ants in the ant colony optimization algorithm.

Initially, the virtual ants in the ACO algorithm randomly choose a path. There is an aspect of randomness and probability since the shorter path is more likely to be chosen than a longer path. After the journey of the initial ants is completed, the next generation of ants will then base their journey off of the results of the previous ants which in this case is the pheromone left behind. The new generation of ants is more attracted to the paths which have a high concentration of pheromones present. Once the new generation has completed its journey, another generation is created which not only uses the pheromones of the previous generation but of all the generations before it up until the initial generation. The more generations are created, the more pheromones are present and the closer the ants get to finding the shortest journey. However, it should be noted that as mentioned before, there is still an aspect of randomness and probability. This means that just because a path has a high concentration of pheromone on it, there is never a 100% guarantee that an ant will choose that path over another path. The probability is higher when there is more pheromone however it is not guaranteed since the ant might choose a path to a closer point or even to a further point.

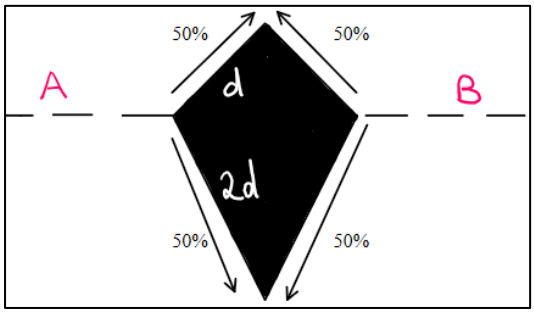

Figure 4 - A diagram showing the 50/50 split of ants when choosing an initial path

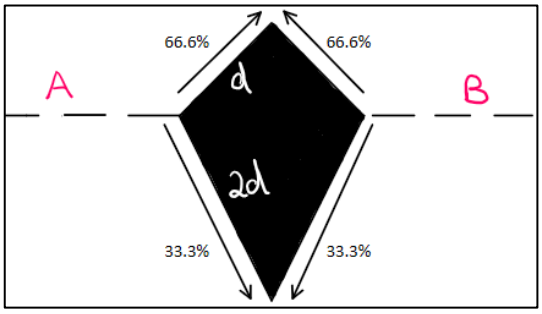

Figure 5 - A diagram showing the more biased path an ant would choose after being traversed a few times

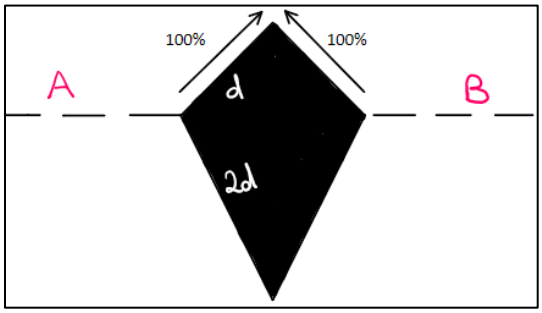

Figure 6 - A diagram showing the ants choosing the shortest path

As an example, imagine there are 100 ants trying to cross from point A to point B and 100 ants trying to cross from point B to point A however they are obstructed by a body of water such as in figure 4. Two sides of the body of water have length d while the other two sides have length 2d. Initially, the ants will evenly split 50/50 across the sides since there is no pheromone present. After 1 second, all the ants manage to traverse distance d and after 2 seconds, they manage the distance 2d. This means that after 2 seconds, all the ants on the upper path have managed to reach their destination while the ants on the bottom half are still half way and require another 2 seconds. When the next generation of ants is created, it is aware that the upper path has more concentrated pheromone so more ants from this new generation are likely to choose the upper path as seen in figure 5. Eventually, as more and more generations are created, all the ants will choose the upper path since the concentration of pheromone is very high.

When considering the travelling salesman problem, the ant colony optimization can be easily implemented since the algorithm is mainly used to find the shortest path. When implementing the algorithm, the virtual ants have distinctive advantages over real ants: virtual ants have memory which means that they know which cities they have visited, meaning that they know not to visit them again while real ants would not know not to visit a path they have already taken. Virtual ants also know the distance between their current city and the nearby cities which means that they are more likely to choose the shorter path rather than the longer ones. However, if the paths between the closest cities are they same, they will choose the path with the most pheromone.

SA to solve Travelling Salesman Problem (TSP) (Code)

```
# importing the required libraries
import tsplib95
import time
from satsp import solver


problem = tsplib95.load('problems/berlin52.tsp.txt') # loading in the data

start_time = time.time()  # starting timer to see how long algorithm takes

cities = []
templist = problem.node_coords  # getting the coordinates for all the
cities

for x in range(0, len(templist)):
    city = [x+1, templist[x+1][0], templist[x+1][1]]
    cities.append(city)  # individually adding each parsed city to a new
list


solver.Solve(cities, stopping_count=300)  # running the algorithm
print("Time taken: %s seconds"%(time.time() - start_time))
solver.PrintSolution()
```

The above code is my chosen method of implementing simulated annealing to solve the travelling salesman problem in python 3.9. I made use of two main libraries in order to implement the algorithm, “tsplib95” as well as “satsp”. “tsplib95” is used in order to be able to read the .tsp file containing all the data required to solve the travelling salesman problem while “satsp” is library containing the simulated annealing algorithm.

ACO to solve Travelling Salesman Problem (TSP) (Code)

```
import tsplib95
import acopy

# setting up the environment
solver = acopy.Solver(rho=.03, q=1)
colony = acopy.Colony(alpha=1, beta=3)

# adding a timer to record the total time to find the best path
timer = acopy.plugins.Timer()
solver.add_plugin(timer)

# setting it up so that is prints out each iteration
printout = acopy.plugins.Printout()
solver.add_plugin(printout)

problem = tsplib95.load('problems/berlin52.tsp.txt')  # loading in the
dataset
G = problem.get_graph()  # changing the dataset so that it can be used by
the algorithm

tour = solver.solve(G, colony, limit=100)  #running the algorithm

# printing out relevant information
print("Shortest tour: ", tour.cost)
print("Best tour: ", tour.nodes)
print("Time to complete: ", timer.duration)
```

The above code is my chosen method of implementing the ant colony optimization algorithm to solve the travelling salesman problem in python 3.9. I once again made use of two main libraries in order to implement the algorithm, “tsplib95” as well as “acopy”. “tsplib95” is used in order to be able to read the .tsp file containing all the data required to solve the travelling salesman problem while “acopy” is library containing the ant colony optimization algorithm.

Application of SA and ACO to other scenarios

Simulated Annealing Algorithm

There have been many uses of simulated annealing in order to solve modern problems such as Askarzadeh who had to deal with the optimal sizing of a photovoltaic and wind hybrid system. He used simulated annealing combined with the chaotic and harmony approaches in order to present simple and efficient discrete optimization methodology.

Similarly, a simulated annealing and Tabu search method hybrid configuration was considered for the sizing problem of hybrid power systems. After multiple tests with many different configurations, the conclusion was that the hybrid configuration using SA gave more sensitive results than one of the techniques used separately.

A simulated annealing based-strategy for maintaining the energy of a smart grid when referring to generation, storage and direction phases of the system energy was proposed by Sousa et al. With their strategy, they successfully gained very favourable results in the compatible management of distributed energy sources as well as in the execution time. Another hybrid algorithm using SA and GA algorithms for the allocation of the sources in the distribution network was presented by Gandomkar et al. Similarly, in order to test the microgrid in terms of trading profit and utilization rate of photovoltaic systems embedded into the microgrid, Velik and Nicolay presented an altered version of the simulated annealing algorithm. Chen also adopted simulated annealing methodology to regulate integrating energy capacity into the main power system for a wind-thermal coordination algorithm. Thanks to this algorithm, they found the best scheduling problem for the isolated wind power source.

Simulated annealing was also used to optimize the design of a 10-bar truss by Elperin. Balling studied the design optimization of steel frames using this SA. The simulated annealing algorithm applied for 2D steel framewoks was also written by May and Balling.

As we can see, there are countless other uses for simulated annealing which aren’t as obvious as finding the shortest path between a number of cities such as the travelling salesman problem. The algorithm can be applied to much more complicated problems and they don’t just work, in most cases, they also present the optimal result.

Ant Colony Optimization Algorithm

The travelling salesman problem is not the only problem the ant colony optimization problem is suitable for. The algorithm has been used in all sorts of combinatorial optimization problems starting from quadratic assignment to routing vehicles or protein folding. In fact, there have been a lot of methods which have been adjust to suit dynamic problems in real variables, multi-targets, stochastic problems, and parallel implementations. Ant colony optimization has an advantage over the simulated annealing algorithm when the graph can change dynamically since the ACO algorithm runs continuously and changes can be made in real time which is not as easy to do in simulated annealing. The ant colony optimization algorithm is also suitable for urban transport systems and network routing.

The ant colony optimization algorithm can also be applied to the job shop scheduling problem which is described as follows: there are n jobs and within each job there is a set of operations which need to be processed in a predetermined specific order. Each operation can only be carried out on a specific machine and only one operation per job can be processed at a given time. The job shop scheduling problem is an optimization problem which the ant colony optimization algorithm can easily be applied to.

There is also the split delivery vehicle routing problem which is suitable for the ant colony optimization algorithm. In the problem, a number of vehicles must service customer in a least-cost way. Savings in costs are very possible when allowing multiple vehicles to service the same customer however, customers might prefer to be services in a single visit rather than split visits so multiple visits would be an inconvenience to them. Since this problem is path based and very similar to the travelling salesman problem, it can very easily by optimized by the ant colony optimization algorithm.

Experiments and their evaluation

In order to get an appropriate feel for both the simulated annealing algorithm and the ant colony optimization algorithm, I ran both algorithms on the same 4 datasets. This was done so that both algorithms have the same amount of data to process to get an accurate and equal read from both of them.

Instance name: berlin52.tsp

SA method

Temperature = initial temperature calculated automatically by satsp at 3810.522028821813

Alpha = cooling rate alpha is automatically set by satsp at 0.99

Epoch Length = automatically set by satsp at 100

ACO method

Number of ants = set by acopy as 52

Pheromones = 1

Alpha = 1

Beta = 3

Rho = 0.3

Iterations = 100

Results

After running both algorithms, the following results were achieved:

	Simulated Annealing (SA)	Ant Colony Optimization (ACO)
Time Taken to Complete	22.095304250717163 sec*	14.103989124298096 sec
Best Distance	7812.098192766203	8315
Best Path	[1, 44, 16, 29, 50, 20, 23, 30, 42, 2, 7, 17, 21, 31, 18, 3, 45, 19, 41, 8, 9, 10, 43, 33, 51, 11, 52, 14, 13, 47, 26, 27, 28, 12, 25, 4, 6, 15, 5, 24, 48, 46, 37, 38, 40, 39, 36, 34, 35, 49, 32, 22]	[23, 30, 20, 50, 29, 16, 46, 44, 34, 35, 36, 39, 38, 40, 37, 24, 48, 6, 15, 5, 4, 25, 12, 27, 28, 26, 47, 14, 13, 52, 11, 51, 33, 43, 10, 9, 8, 41, 19, 45, 32, 49, 1, 22, 18, 31, 21, 3, 17, 2, 42, 7]

**Simulated annealing terminated after 940 epochs.*

In the above table, we see the results of both the simulated annealing algorithm as well as the ant colony optimization algorithm side by side when given a dataset of 52 cities.

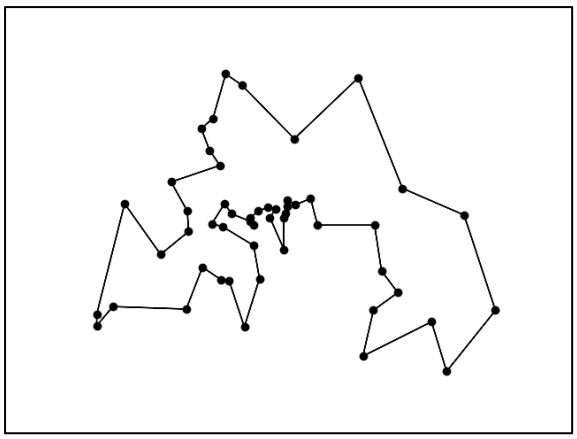

Figure 7 - A graph showing the plotted best path for the TSP using SA on the berlin52 dataset

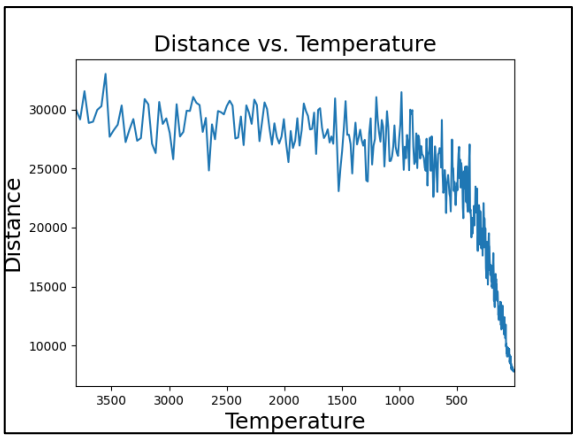

Figure 8 - A graph showing the shortest path being found as the variable temperature decreases using TSP on the berlin52 dataset

When looking at the simulated annealing algorithm, we can see that it took approximately 22.1 seconds to run through the set number of iterations. It also found that the shortest path had an approximate length of 7812.1. On the other hand, the ant colony optimization algorithm took approximately 14.1 seconds to run thought the set number of iteration and found that the shortest path had a length of 8315.

Comparing the results from both algorithms, we can see that although the ant colony optimization algorithm took less time than the simulated annealing algorithm, the latter algorithm successfully found a shorter path than the ant colony optimization algorithm.

When looking more closely at the paths that both algorithms resulted in, we can see some similarities in both results. In both cases, we can see that the paths between city 50, 20, 30 and 23 are present in both situations as well as the path between city 16 and city 29. However, the two sets of cities have almost identical paths in both cases of the algorithms are between the cities 37, 38, 40, 39, 36, 34, 35 and between the cities 45, 19, 41, 8, 9, 10, 43, 33, 51, 11, 52, 14, 13, 47, 26, 27, 28, 12, 25, 4, 6, 15, 5. When looking at figure 7 above, we can see that these cities are most likely the collection of cities very close together in the middle.

Instance name: kroD100.tsp

SA method

Temperature = initial temperature calculated automatically by satsp at 9539.125625078275

Alpha = cooling rate alpha is automatically set by satsp at 0.99

Epoch Length = automatically set by library as 100

ACO method

Number of ants = set to default by acopy

Pheromones = 1

Alpha = 1

Beta = 3

Rho = 0.3

Results:
After running both algorithms, the following results were achieved:

	Simulated Annealing (SA)	Ant Colony Optimization (ACO)
Time Taken to Complete	35.34899616241455 sec*	134.9614417552948 sec
Best Distance	23286.569968979587	26135
Best Path	[1, 27, 24, 88, 13, 79, 41, 84, 73, 39, 15, 63, 54, 91, 77, 43, 71, 4, 93, 44, 60, 53, 62, 35, 30, 100, 51, 37, 19, 2, 48, 92, 5, 3, 83, 40, 59, 6, 25, 20, 10, 9, 55, 87, 26, 95, 56, 21, 47, 28, 12, 82, 18, 85, 64, 16, 96, 86, 65, 14, 68, 33, 45, 99, 78, 31, 57, 72, 80, 76, 7, 75, 67, 36, 29, 58, 22, 42, 74, 61, 69, 46, 52, 8, 66, 81, 34, 38, 23, 11, 89, 70, 49, 98, 97, 32, 94, 17, 90, 50]	[34, 81, 38, 66, 8, 52, 46, 61, 74, 7, 67, 75, 36, 29, 58, 22, 42, 23, 89, 11, 90, 94, 17, 70, 49, 98, 97, 32, 50, 1, 27, 24, 88, 79, 13, 84, 41, 73, 39, 91, 77, 43, 71, 4, 93, 44, 60, 5, 53, 3, 6, 83, 40, 59, 85, 18, 25, 20, 9, 10, 21, 47, 12, 28, 56, 95, 26, 87, 55, 92, 19, 100, 51, 37, 62, 35, 30, 2, 48, 64, 96, 86, 16, 63, 69, 15, 54, 82, 68, 14, 33, 65, 45, 99, 31, 78, 76, 80, 57, 72]

**Simulated annealing terminated after 1151 epochs*

In the above table, we see the results of both the simulated annealing algorithm as well as the ant colony optimization algorithm side by side when given a dataset of 100 cities.

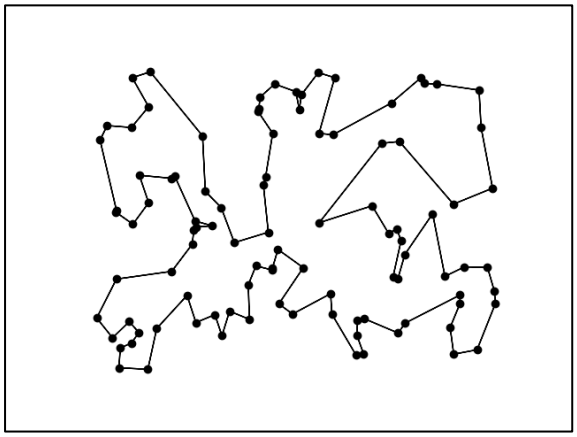

Figure 7 - A graph showing the plotted best path for the TSP using SA on the kroD100 dataset

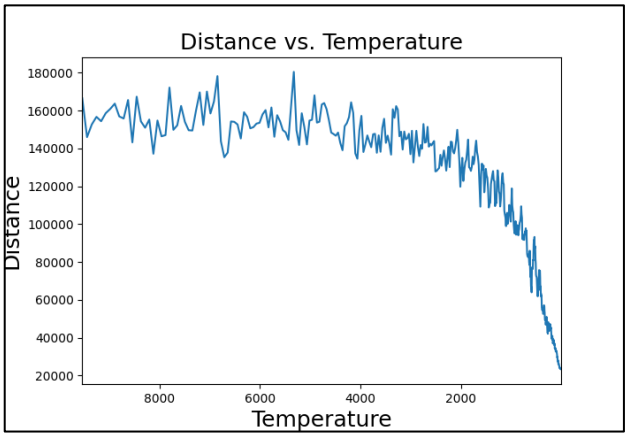

Figure 8 - A graph showing the shortest path being found as the variable temperature decreases using TSP on the kroD100 dataset

When looking at the simulated annealing algorithm, we can see that it took approximately 35.3 seconds to run through the set number of iterations. It also found that the shortest path had an approximate length of 23286.6. On the other hand, the ant colony optimization algorithm took approximately 135 seconds to run thought the set number of iteration and found that the shortest path had a length of 26135.

Comparing the results from both algorithms, we can see that in this test, the ant colony optimization algorithm took much more time than the simulated annealing algorithm. However, this test also resulted in the simulated annealing finding the shorter path.

When looking more closely at the paths that both algorithms resulted in, we can see some similarities in both results. The two sets of cities that have almost identical paths in both cases of the algorithms are between the cities 1, 27, 24, 88, 79, 13, 84, 41, 73, 39 and between the cities 91, 77, 43, 71, 4, 93, 44, 60.

Instance name: lin105.tsp

SA method

Temperature = initial temperature calculated automatically by satsp at 5736.846881182152

Alpha = cooling rate alpha is automatically set by satsp at 0.99

Epoch Length = automatically set by library as 100

ACO method

Number of ants = set by acopy as 105

Pheromones = 1

Alpha = 1

Beta = 3

Rho = 0.3

Results:

After running both algorithms, the following results were achieved:

	Simulated Annealing (SA)	Ant Colony Optimization (ACO)
Time Taken to Complete	29.62398862838745 sec*	161.9491605758667 sec
Best Distance	36478.170682675605	17495
Best Path	[1, 28, 40, 49, 48, 53, 54, 36, 42, 50, 26, 78, 72, 85, 84, 80, 79, 71, 77, 102, 101, 92, 91, 90, 74, 75, 73, 64, 68, 93, 88, 98, 99, 94, 87, 61, 66, 60, 65, 52, 57, 69, 86, 89, 76, 62, 67, 105, 59, 81, 100, 95, 97, 96, 82, 83, 56, 23, 29, 13, 17, 9, 25, 16, 11, 6, 3, 8, 33, 43, 41, 104, 51, 47, 46, 58, 44, 19, 18, 27, 70, 63, 30, 22, 15, 24, 21, 10, 45, 55, 103, 20, 7, 32, 12, 37, 35, 38, 39, 5, 4, 14, 34, 31, 2]	[44, 47, 51, 54, 55, 50, 40, 45, 48, 49, 104, 36, 37, 42, 46, 43, 41, 52, 58, 53, 59, 56, 105, 57, 63, 62, 70, 74, 69, 81, 75, 76, 80, 73, 64, 67, 68, 78, 71, 82, 83, 84, 85, 91, 92, 96, 97, 102, 101, 93, 86, 79, 77, 72, 89, 90, 99, 98, 100, 94, 95, 87, 88, 66, 65, 60, 61, 38, 39, 34, 35, 14, 13, 5, 4, 9, 8, 17, 25, 18, 16, 26, 24, 27, 19, 28, 20, 23, 103, 22, 29, 21, 15, 10, 11, 7, 6, 1, 2, 3, 12, 30, 31, 32, 33]

**Simulated annealing terminated after 300 epochs.*

In the above table, we see the results of both the simulated annealing algorithm as well as the ant colony optimization algorithm side by side when given a dataset of 105 cities.

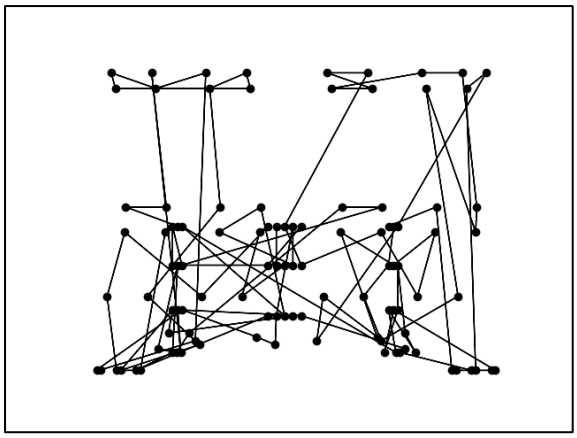

Figure 11 - A graph showing the plotted best path for the TSP using SA on the lin105 dataset

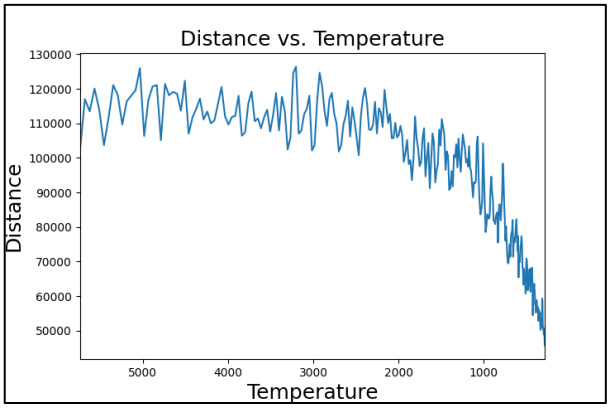

Figure 12 - A graph showing the shortest path being found as the variable temperature decreases using TSP on the lin105 dataset

When looking at the simulated annealing algorithm, we can see that it took approximately 29.6 seconds to run through the set number of iterations. It also found that the shortest path had an approximate length of 36478.2. On the other hand, the ant colony optimization algorithm took approximately 162 seconds to run thought the set number of iteration and found that the shortest path had a length of 17495.

Comparing the results from both algorithms, we can see that in this test, while the ant colony optimization algorithm still took time than the simulated annealing algorithm, this time the ACO algorithm found the much shorter path than the SA algorithm.

As was the case in the other two tests, we can find instances between both algorithms where they chose the same paths such as between cities 48 and 49 and between cities 38 and 39.

Instance name: pr124.tsp

SA method

Temperature = initial temperature calculated automatically by satsp at 32006.16572873719

Alpha = cooling rate alpha is automatically set by satsp at 0.99

Epoch Length = automatically set by library as 100

ACO method

Number of ants = set by acopy as 124

Pheromones = 1

Alpha = 1

Beta = 3

Rho = 0.3

Results:
After running both algorithms, the following results were achieved:

	Simulated Annealing (SA)	Ant Colony Optimization (ACO)
Time Taken to Complete	37.293567419052124 sec*	285.87916326522827 sec
Best Distance	98943.48735630691	77453
Best Path	[1, 30, 63, 64, 82, 107, 84, 121, 67, 117, 123, 115, 120, 78, 83, 124, 89, 85, 86, 90, 99, 92, 79, 33, 34, 6, 36, 59, 61, 60, 3, 32, 65, 98, 80, 93, 118, 97, 96, 69, 62, 40, 41, 14, 16, 49, 51, 111, 101, 91, 102, 110, 113, 114, 112, 103, 70, 77, 75, 74, 81, 71, 119, 72, 116, 100, 68, 66, 73, 76, 95, 94, 122, 31, 25, 22, 38, 88, 106, 108, 105, 104, 9, 48, 13, 15, 12, 10, 23, 29, 56, 58, 37, 39, 45, 18, 109, 52, 50, 11, 43, 20, 44, 47, 53, 54, 46, 87, 35, 26, 2, 4, 27, 24, 57, 55, 42, 21, 19, 17, 7, 28, 8, 5]	[113, 114, 112, 111, 110, 109, 108, 107, 106, 105, 104, 89, 88, 87, 86, 85, 90, 103, 102, 91, 101, 84, 83, 82, 81, 67, 66, 65, 63, 68, 69, 64, 70, 71, 74, 72, 73, 75, 77, 76, 78, 79, 80, 92, 93, 94, 95, 97, 98, 99, 100, 124, 123, 122, 121, 120, 119, 118, 117, 116, 115, 96, 57, 56, 55, 58, 59, 61, 60, 62, 35, 36, 29, 7, 1, 8, 2, 3, 4, 5, 30, 34, 33, 6, 32, 31, 27, 28, 26, 38, 37, 24, 25, 23, 22, 39, 40, 41, 42, 21, 20, 19, 18, 45, 43, 44, 54, 53, 52, 46, 47, 48, 49, 50, 51, 13, 12, 11, 14, 15, 10, 9, 16, 17]

**Simulated annealing terminated after 300 epochs.*

In the above table, we see the results of both the simulated annealing algorithm as well as the ant colony optimization algorithm side by side when given a dataset of 124 cities.

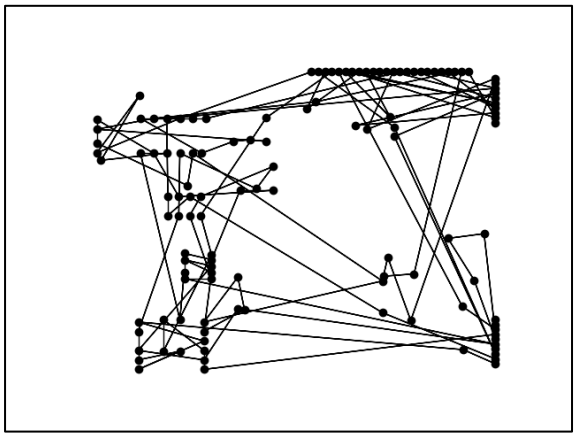

Figure 13 - A graph showing the plotted best path for the TSP using SA on the pr124 dataset

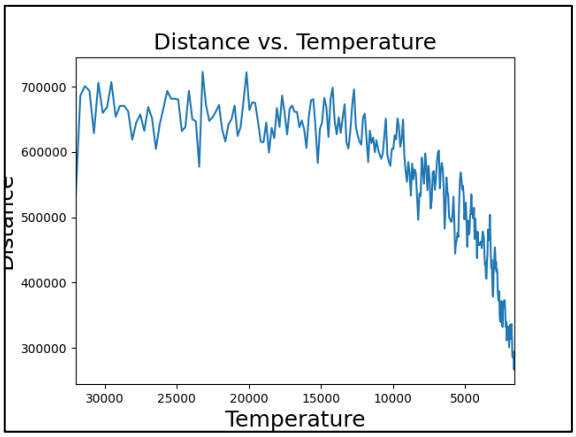

Figure 14 - A graph showing the shortest path being found as the variable temperature decreases using TSP on the pr124 dataset

When looking at the simulated annealing algorithm, we can see that it took approximately 37.3 seconds to run through the set number of iterations. It also found that the shortest path had an approximate length of 98943.5. On the other hand, the ant colony optimization algorithm took approximately 285.9 seconds to run thought the set number of iteration and found that the shortest path had a length of 77453.

Comparing the results from both algorithms, we can see that also in this test, while the ant colony optimization algorithm still took time than the simulated annealing algorithm, this time the ACO algorithm found the much shorter path than the SA algorithm.

As was the case in the other two tests, we can find instances between both algorithms where they chose the same paths such as between cities 113 and 114 and between cities 95 and 94.

Overall Conclusions

From the four tests carried out using both the simulated annealing algorithm and ant colony optimization algorithm, in three tests, the SA algorithm performed much faster than the ACO algorithm however, in two of the cases, the ant colony optimization algorithm found a much shorter path than the simulated annealing algorithm. This implies that should the ant colony optimization algorithm have more iterations, it is likely that it would take more time than SA but it will probably find a shorter path

It is also to be noted that the two times the ACO algorithm found the shorter path, the dataset given to the algorithms was larger than the datasets given to the two sets where the simulated annealing algorithm found the shorter path. This implies that when given a much larger dataset, it is more likely that the ant colony optimization algorithm will find a shorter path.

Although both algorithms had similar results when dealing with a small dataset and produced very similar results, it is clear that then dealing with a larger dataset, the ant colony optimization algorithm will perform better.

Statement of Completion

Item	Completed
SA and ACO technical discussion	Yes
Artifact 1: TSP with SA	Yes
Artifact 2: TSP with ACO	Yes
Application of SA and ACO to other scenarios	Yes
Experiments and their evaluation	Yes
Overall conclusions	Yes

Plagiarism Declaration Form

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I, the undersigned, declare that the assignment submitted is my work, except were acknowledged and referenced.

I understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

Francesca Maria Mizzi
Student Name


Signature

ICS2207
Course Code

Simulated Annealing and Ant Colony Optimization Algorithms
Title of work submitted

29/01/2021
Date

References

[1]
"Simulated Annealing - an overview | ScienceDirect Topics," *Sciencedirect.com*, 2011. https://www.sciencedirect.com/topics/materials-science/simulated-annealing (accessed Jan. 27, 2021).
[2]
Wikipedia Contributors, "Simulated annealing," *Wikipedia*, Jan. 27, 2021. https://en.wikipedia.org/wiki/Simulated_annealing#:~:text=Simulated%20annealing%20(SA)%20is%20a,space%20for%20an%20optimization%20problem. (accessed Jan. 28, 2021).
[3]
"TSPLIB 95 — TSPLIB 95 0.7.1 documentation," *Readthedocs.io*, 2018. https://tsplib95.readthedocs.io/en/stable/pages/readme.html (accessed Jan. 28, 2021).
[4]
"satsp," *PyPI*, Sep. 26, 2018. https://pypi.org/project/satsp/ (accessed Jan. 28, 2021).
[5]
"MP-TESTDATA - The TSPLIB Symmetric Traveling Salesman Problem Instances," *Elib.zib.de*, 2021. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html (accessed Jan. 28, 2021).
[6]
"Usage — ACOpy 0.7.0 documentation," *Readthedocs.io*, 2018. https://acopy.readthedocs.io/en/latest/usage.html# (accessed Jan. 28, 2021).
[7]
Wikipedia Contributors, "Ant colony optimization algorithms," *Wikipedia*, Jan. 20, 2021. https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms (accessed Jan. 28, 2021).
[8]
"Ant Colony Optimization - an overview | ScienceDirect Topics," *Sciencedirect.com*, 2019. https://www.sciencedirect.com/topics/engineering/ant-colony-optimization (accessed Jan. 28, 2021).
[9]
G. V. Batista, C. T. Scarpin, J. E. Pécora, and A. Ruiz, "A New Ant Colony Optimization Algorithm to Solve the Periodic Capacitated Arc Routing Problem with Continuous Moves," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–12, Jul. 2019, doi: 10.1155/2019/3201656.
[10]
Wikipedia Contributors, "Job shop scheduling," *Wikipedia*, Jan. 11, 2021. https://en.wikipedia.org/wiki/Job_shop_scheduling (accessed Jan. 29, 2021).
[11]
D. Gulczynski, B. Golden, and E. Wasil, "The split delivery vehicle routing problem with minimum delivery amounts," *Transportation Research Part E: Logistics and Transportation Review*, vol. 46, no. 5, pp. 612–626, Sep. 2010, doi: 10.1016/j.tre.2009.12.007.