A Learned Polyalphabetic Decryption Cipher

Chaminda Hewage[1*], Ambikesh Jayal[1], Glenn Jenkins[1], Ryan J. Brown[1]

[1] Cardiff School of Technology, Cardiff Metropolitan University, Llandaff Campus, CF5 2YB, Cardiff, UK;
chewage@cardiffmet.ac.uk

SNE 28(4), 2018, 141 - 148, DOI: 10.11128/sne.28.tn.10441
Received: November 20, 2018; Revised: December 6, 2018;
Accepted: December 8, 2018
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. This paper examines the use of machine learning algorithms to model polyalphabetic ciphers for decryption. The focus of this research is to train and evaluate different machine learning algorithms to model the polyalphabetic cipher. The algorithms that have been selected are: (1) hill climbing; (2) genetic algorithm; (3) simulated annealing; and (4), random optimisation. The resulting models were deployed in a simulation to decrypt sample codes. The resulting analysis showed that the genetic algorithm was the most effective technique used in with hill climbing as second. Furthermore, both have the potential to be useful for larger problems.

Introduction

Stamp [1] states that: 'Cryptology is the art and science of making and breaking "secret codes"'. Martin [2] defines cryptology as a loosely used term to describe, 'the design and analysis of mechanisms based on mathematical techniques' to secure data and information.

There are two types of studies in cryptology. "Cryptography" describes the fundamentals of securing data by using such mechanisms to design an algorithm [2][15]. "Cryptanalysis" is the opposite of cryptography and uses an 'analysis of such mechanisms' to decrypt its encryption [2][14]. Cryptology is therefore a way of transforming an original message into cipher text that an interceptor may not be able to read and understand. However, the true recipient of the message could transform the message back to its original readable message by using a suitable decryption technique.

The purpose of cryptanalysis is to uncover or exploit encrypted information, it is a study and science of recovering the original plaintext without knowing the key [3]. Cryptanalysis is perceived today as 'code-breaking' or 'hacking', but maybe better known as an 'attack' [3][13]. Its primary concern is identifying and attacking the vulnerabilities in weak methods to gain knowledge of the plaintext [4].

As a cryptanalyst, it is important to understand what type of algorithm is used before attempting to unravel the cipher and give meaning to the content. According to Schneier [3] if a cryptanalyst cannot break the algorithm used, having known the background information of the algorithm, then they are unlikely to be successful at breaking it. Therefore, before any cryptanalyst can 'attack' an encrypted message, it is important to discover and analyse the type of method used for the cryptosystem.

The primary aim of this research is to explore the application of machine learning algorithms to the modelling polyalphabetic substitution ciphers for decryption. The focus of this paper is the application of well-known machine learning techniques as a first step to exploring more sophisticated machine learning techniques. The paper is structured as follows, first literature related to the cryptography techniques and machine learning algorithms are considered. Next the methodology used in is reported followed by the results. These are discussed and conclusions drawn in the final section.

1 Machine Learning Algorithms

Machine Learning is a branch of artificial intelligence, and its purpose is finding out if a computer can develop a model without prior learning and then improve this model, just like a human. The computer learns over time, which helps in finding a better solution to a problem [5], i.e. improving the model. More importantly, it learns 'without being explicitly programmed', which means that it has the ability to learn (create a model) based one dataset and apply this model to other datasets, the result is more flexible solutions [5]. Machine Learning research is a popular approach to problems today, such as: discovering new medicines and accurately diagnosing patients; working out better solutions to a specific problem (e.g. the travelling sales man); and finally, using machine learning to better understand cryptology.

Some machine learning algorithms are inspired by nature as these can provide a useful way of looking at a particular problem.

This means that nature-inspired algorithms can be employed to achieve solutions to difficult tasks [6][12]. For example, the ant colony optimisation is a nature inspired algorithm analysing ants 'social behaviour in finding shortest paths' [6]; this type of algorithm could also help by solving other real world issues that require finding the shortest route.

This research takes into consideration three main machine learning nature-inspired algorithms. These are outlined below.

1.1 Hill Climbing Algorithm

This algorithm is inspired by nature as it 'resembles trying to find the top of Mount Everest in a thick fog while suffering from amnesia' [7]. The purpose of this type of algorithm is to find and improve on the best local solution to the problem after each step, checking whether the neighbouring results are better or worse than the current position (also known as a 'local search') [9]. A problem with this algorithm, is that it can reach a local maximum quite quickly. It may have found a good enough solution, but not the best (global maxima). Ways in which to resolve these issues include using multi-starts or simply allowing the algorithm to accept negative moves [8]. In turn, the hill climbing algorithm has more scope with the data, and has a better chance of finding a best solution.

1.2 Genetic Algorithm

Genetic Algorithm (GA) is another well-known nature inspired method; it is also known as an evolutionary approach. In particular, GA is inspired by biological evolution. For example, in the gene selection stage it takes both parents genes to produce a mutation or crossover to a new set of genetic composition [6]. It is a very useful algorithm for solving 'local search, optimisation and machine learning problems' [6]. This type of algorithm works by finding the best successor (result) from a combination of parents that are modified in ways of either mutation or crossover [7].

1.3 Simulated Annealing

The simulated annealing algorithm is based upon the process of heating up metals and glass to very high temperatures and slowly cooling them to the shape required [7]. This is quite similar to the hill climbing algorithm, but is implemented to prevent reaching a local maximum.

While the temperature is still high the probability allows the annealing process to accept worse answers, as well as a better ones.

This improves the scope in which the algorithm can search and as it leads towards a good solution the probability of accepting worse solutions are discarded.

2 Methodology

The aim of this research was to ascertain which of the selected machine learning algorithms were best at modelling this type of encryption. The following sections introduce the encryption algorithm used, test data selected, machine learning algorithms and metrics.

2.1 Encryption

A message from Winston Churchill to Franklin D. Roosevelt was used to create a sense of realism to the projects goal of uncovering sensitive data. For this to work, all grammar and spaces were removed from the text when encrypting, making the output text (cipher text) a single block of unreadable script. This ensured that the text is more difficult to comprehend and provided slightly more protection.

Encryption used a polyalphabetic cipher (Vigenere cipher) to encrypt the message, implemented based on well known sources [2,9]. This encryption focused on using a stream cipher to look at each individual bit of character of the text, and changed it to a new corresponding letter. An alternative method considered used an array of alphabetic letters that linked to an index; however, this method seemed to be less practical and consequently was not used.

2.2 Decryption

The procedure of decryption is very similar to encryption, as it performs a sequential search through each character of the cipher text, discovering the cipher text and keys represented decimal numbers of ASCII at each iteration. The outputted cipher text was used for decrypting to help analyse the machine learning algorithms, checking whether the cipher text can be decrypted. The purpose of the research was to find out whether the algorithms used can decrypt the cipher text with the same key as encrypted. Therefore, getting the algorithm to understand a way of finding the best decryption key. It is also worth pointing out that the key size has been hard coded into the program enabling the algorithms to work at an efficient rate, and focus on the decryption of the cipher text.

Algorithm	Parameter Name	Description	Value
Hill Climbing	Repeats	Number of times to repeat the experiment.	10
Genetic Algorithm	Repeats	Number of times to repeat the experiment.	10
	Population Size	Size of the population of each generation.	100
	Mutation Probability	Probability of mutation.	0.6
	Elite	Proportion of population used for crossover.	0.4
	Iterations	Number of generations over which to evolve the population.	120
Simulated Annealing	Repeats	Number of times to repeat the experiment.	10
	Initial Temp.	Initial temperature.	1.0×10^5
Random Optimization	Repeats	Number of times to repeat the experiment.	10
	Iterations	Maximum number of training iterations.	1000

Table 1: The training parameters of the experiment.

2.3 Scoring

To test the algorithms, it was important to utilise a scoring method, which could assess each of the algorithms performances in modelling the key. The fitness of the decrypted text was scored using quad grams [9]. Quad grams were chosen over trigrams or bigrams in order to reduce the time taken for assessing score for the selected scope and length of the cipher text. This provided a metric to assess results of the English language decryption of the cipher text. Big O notation [11], provides a measure of complexity and gives an indication of how well an algorithm will scale. This was used to analyse the machine learning techniques and provide an indication of the algorithms potential.

2.4 Machine Learning Algorithms

The machine learning algorithms were implemented based on the work of Segaran [10], the training parameters used (shown in Table 1) were identified by experimentation.

3 Results

The results reported below are based on the the cipher score. Here each key modelled by the selected machine learning techniques is used to decrypt the cipher text and find its fitness score (closer to 0 represents the English language). The results showing the key used, completion time and the number of keys identified throughout the process (i.e. the number of times the best key changed). Each experiment was repeated ten times to ensure consistency of the results, these are recorded individually below so the keys can be visually compared.

3.1 Random Hill Climbing

The hill climbing algorithm has had the ability to get very close to the correct key. For example, this can be shown in repeat five of Figure 1, where the best score is a model with the an output (WINSTUN). This bares close resemblance to the key used (WINSTON) to encrypt, and the decrypted text will have close similarity to the text used. Considering all the outputs from this algorithm in Table 2, they all reasonably close. This would suggest that using these keys for decryption would help a cryptanalysis to identify some words or letters in the cipher text. Consequently, this type of algorithm has fared considerably well in the short amount of time taken to nearly decrypt the message. This suggests that if this runs longer period of time, it could potentially model the key used and uncover the original plaintext.

The results in Table 2 suggests that this algorithm has the capability produce the best result due to the number of good models it can find in quite a short period. For example, repeats 8 with an output of 'ACNSTON' has the shortest completion time of 48.30 (s), but has found more than six thousand better models throughout the process. This implies that with only two letters wrong in this key that this decryption could be somewhat successful as some words or letters would be recognisable.

Also, due to the nature of this algorithm constantly striving to find a better result it has nearly achieved its full potential. Although it did not find the correct key within the repeats tested, it nevertheless came close to solving it.

The graphs in Figure 1 represent each of the repeats from Table 2. It is interesting to note that the graphs can be represented as sound waves, but are a collection of all the different hill climbing points within the scope.

All the lowest points in the graphs represent the starting point from where the algorithm grows to a better result, which is shown at the peaks throughout the processes.

Fascinatingly, it is not clear where the best result will be found and image 5 of Figure 1 clearly shows this, since within the process a sudden move to the best result noted. This is better than any other keys that have been found. Observing these graphs has helped to clarify the progress of the algorithms at each repeat.

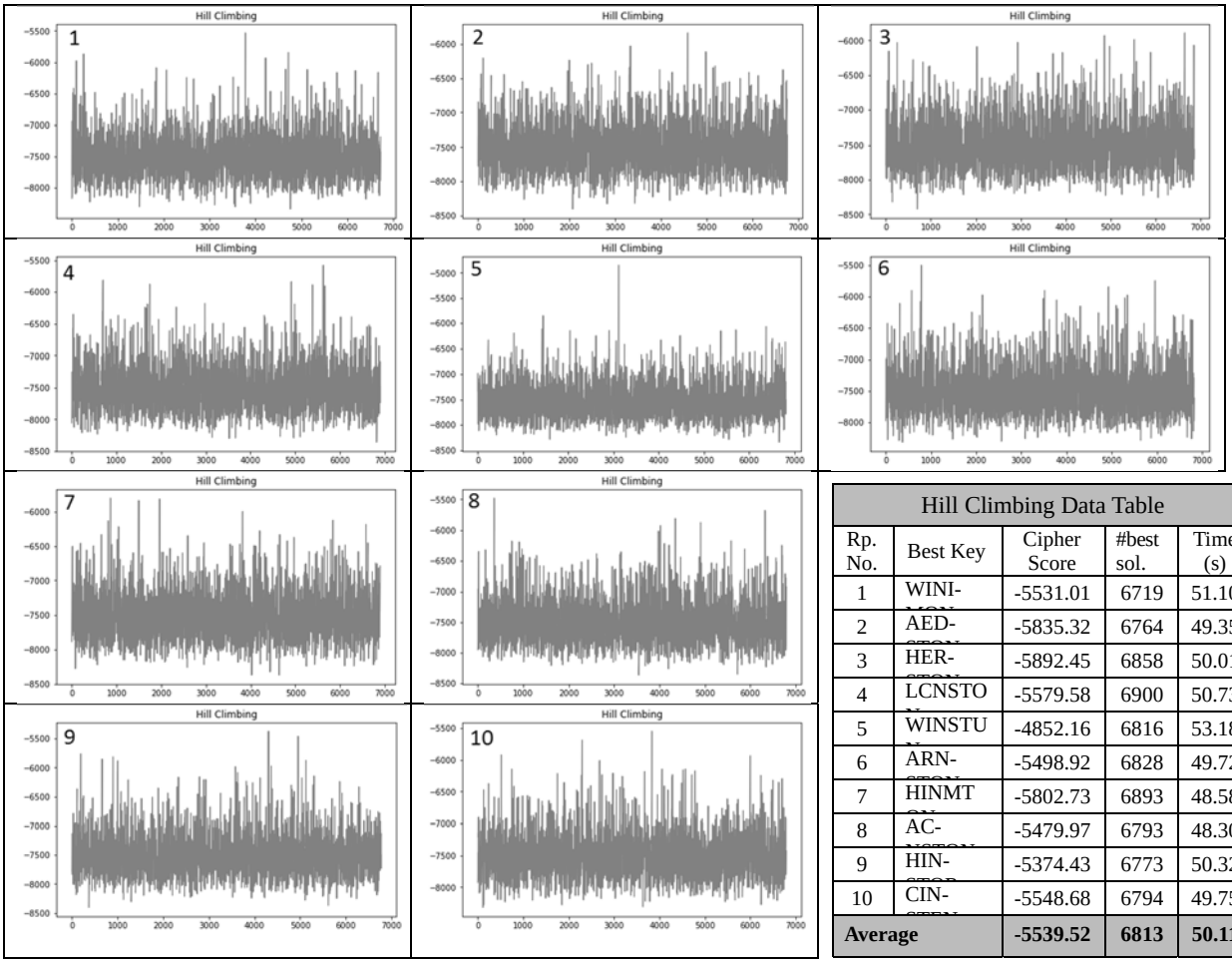

Figure 1: Random Hill Climbing Graphs (Y-axis: Score; X-axis: No. of Best Solutions Found).

Hill Climbing Data Table				
Rp. No.	Best Key	Cipher Score	#best sol.	Time (s)
1	WINI-	-5531.01	6719	51.10
2	AED-	-5835.32	6764	49.35
3	HER-	-5892.45	6858	50.01
4	LCNSTO	-5579.58	6900	50.73
5	WINSTU	-4852.16	6816	53.18
6	ARN-	-5498.92	6828	49.72
7	HINMT	-5802.73	6893	48.58
8	AC-	-5479.97	6793	48.30
9	HIN-	-5374.43	6773	50.32
10	CIN-	-5548.68	6794	49.75
Average		**-5539.52**	**6813**	**50.11**

Table 2: Random Hill Climbing data.

Assessing the implementation of the algorithm the complexity is of quadratic-time $O(N^2)$. This suggests a increase in the size of the input (the key size) would results in a significant increase in complexity and therefore time to complete. This suggests that while the algorithm is suitable for the small keys used in this research there may be issues when scaling to larger keys.

3.2 Genetic Algorithm

The genetic algorithm results shown in Table 3, produces an accurate model of the key on three occasions. It can find the key used to encrypt the polyalphabetic cipher. Also, considering the other keys, it has a success rate correctly modelling at least five out of seven letters of the key. This largely suggests that using any of these keys could uncover some words or phrases in the decrypted text.

The results also show that for each repeat the number of best solutions has been reasonably consistent, this is ascribed to the underlying algorithm and its implementation. As shown in Table 3, this algorithm finds a good number of best solutions in a short period on average 23.40 (s).

The below graphs from Figure 2 represent the data collected in Table 3. There is a clear pattern to the graphs, the results demonstrate a consistent improvement in the result. For example, repeat 1 in Figure 2 has a rapid growth of finding the best result after 1000 other better solutions. However, a plateau appears once it has found its best solution.

This rectangular shape is a fluctuation between one best solution and another, and suggests that it cannot improve beyond this point (as shown in all other repeats). Looking back at the data in Figure 2 shows that once it is found what it thinks is the best solution of best keys, it does not have the capabilities to be able to progress beyond that.

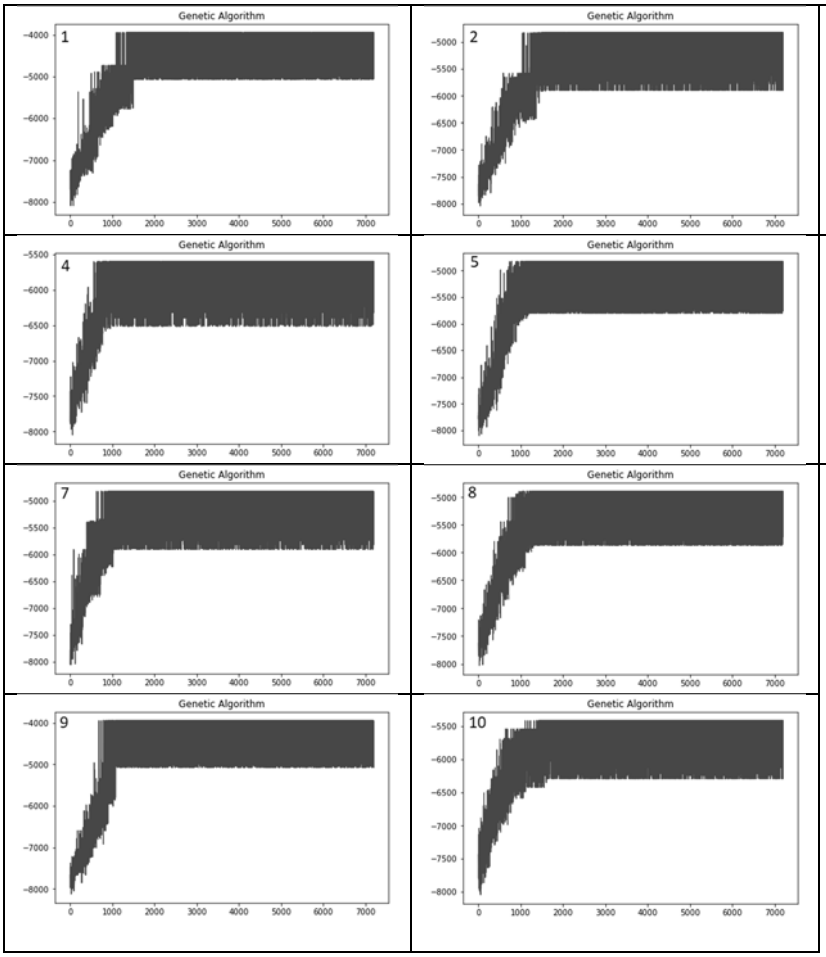

Figure 2: Genetic Algorithm Graphs (Y-axis: Score; X-axis: No. of Best Solutions Found).

Hill Climbing Data Table				
Rp. No.	Best Key	Cipher Score	#best sol.	Time (s)
1	WINI-	-5531.01	6719	51.10
2	AED-	-5835.32	6764	49.35
3	HER-	-5892.45	6858	50.01
4	LCNSTO	-5579.58	6900	50.73
5	WINSTU	-4852.16	6816	53.18
6	ARN-	-5498.92	6828	49.72
7	HINMTO	-5802.73	6893	48.58
8	AC-	-5479.97	6793	48.30
9	HINSTOR	-5374.43	6773	50.32
10	CINSTEN	-5548.68	6794	49.75
Average		**-5539.52**	**6813**	**50.11**

Table 3: Genetic Algorithm data.

Nevertheless, the scores that have been collected prove that this algorithm has the efficiency and reliability to find a solution to this problem in a small amount of time. The number of best solutions found throughout this process make this algorithm reliable.

Looking at the implementation of this algorithm the complexity is quadratic-time $O(N^2)$. This again suggests that this algorithm might be better suited to modeling shorter keys.

3.3 Random Optimisation

The random optimisation algorithm performs reasonably well for a random search, though it should be noted it is not as reliable as the other algorithms. The results in Table 4, demonstrate some reasonable results. It is also clear that it takes on average less time to complete and can get some good results. This can be shown by looking at the best keys, where some of the letters represent the original key used to encrypt: thus, it is quite close to finding the correct key. Not once though has the algorithm completed the process successfully. The data show that it has the potential to search a much wider scope if the number of repeats were increased in the algorithm. The best keys here could have the potential to uncover some small phrases, which could be useful to a cryptanalysis.

As mentioned previously, the inconsistency of this algorithm is what makes it weak, and ultimately unusable. On the other hand, when observing the graphs in Figure 3, they all show an improving state; and one which represents the hill climbing event, where it tries to find a better solution to find its best result within the scope. Each repeats has a different climb and different number of best solutions found (as displayed in Figure 3). Interestingly, they all bare similar qualities of aiming for better solutions.

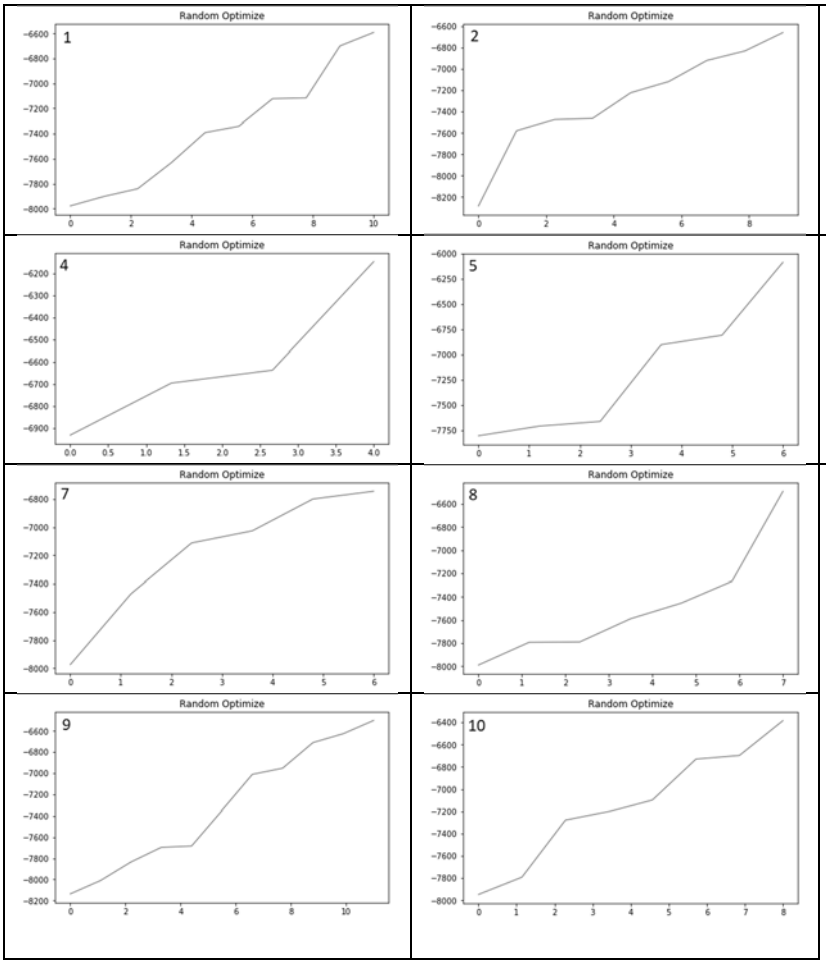

Figure 3: Random Optimisation graphs (Y-axis: Score; X-axis: No. of Best Solutions Found).

Random Optimise Results				
Rp. No.	Best Key	Cipher Score	#best sol.	Time (s)
1	JYNYTOA	-6591.53	10	10.01
2	IKERTON	-6659.19	9	9.24
3	WINSKGM	-6325.53	23	9.63
4	BEPSTON	-6147.19	4	9.18
5	WEHSTOT	-6085.54	6	9.28
6	EFUSTOJ	-6573.54	11	9.73
7	ZETSTEN	-6743.52	6	9.22
8	WCNQION	-6488.86	7	9.17
9	HXNVTON	-6500.34	11	9.21
10	FINKTOR	-6383.89	8	9.21
Average		**-6449.91**	**9.5**	**9.39**

Table 4: Random Optimisation Results

However, they do this through different paths because of its randomness and unreliability. For smaller problems, this algorithm could be useful, although it has shown many weaknesses within this research; and for larger keys used, the chance of solving them is minimal.

Due to its simplicity the random optimisation algorithm has a complexity is of linear-time O (N) which means that for input (keys) there would be a proportional increase in the complexity and therefore the time taken.

3.4 Simulated Annealing

The simulated annealing algorithm struggles to model the keys, as shown in Table 5, all the best keys do not bare any resemblance to the key used (WINSTON) to encrypt the text. Simulated annealing does not produce any good results: the average cipher score implies that a text deciphered using a key with this score would result in a failure, since the message would still be unclear and not readable English. It is noteworthy that even though this algorithm was not successful it has the ability to find many different though weak solutions throughout the process in a reasonable amount of time.

The graphs in Figure 4 show an alternative view of the performance of the algorithm. Each have a bad start, with some results getting worse like repeats 6. However, towards the end of the process the algorithms are now adjusting themselves to only accept better results. This is because the probability is less likely to accept the worse results; thus, it gradually ascends to a result that is better than the others. A good example of this can be shown in repeats 6 (Figure 4). It starts off unpredictable but at the end a curve appears where it begins to increase only using the conditions to accept better results. As it is shown in Figure 4, that all of the repeats exceed during the end of its process.

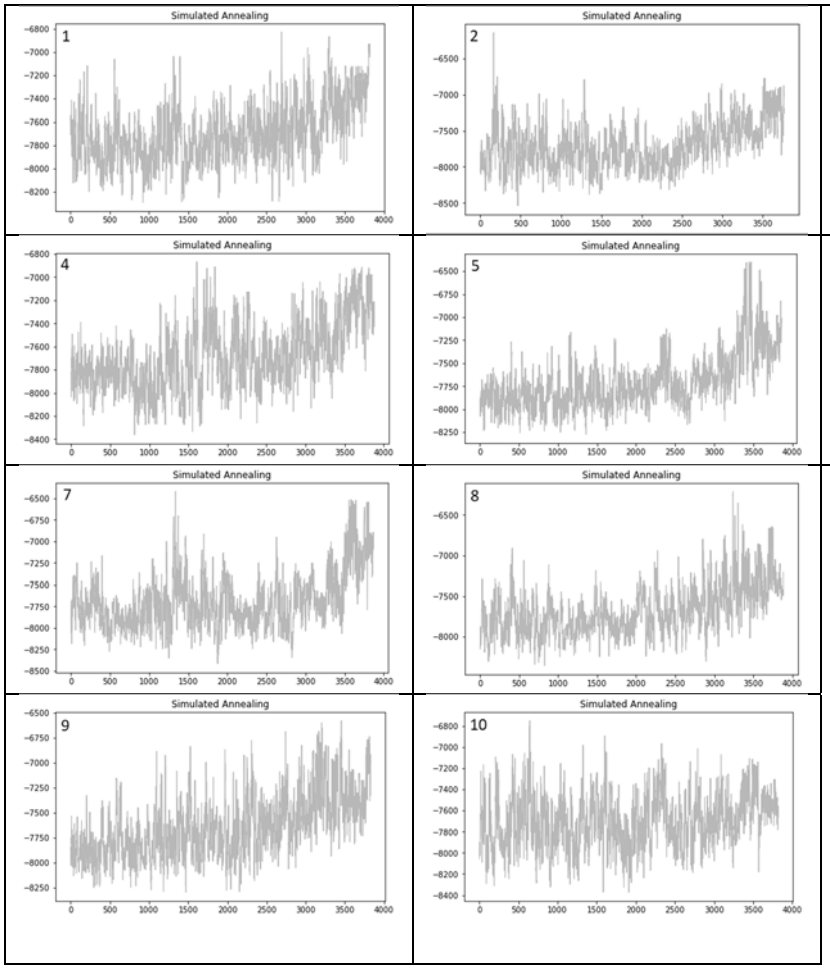

Figure 4: Simulated Annealing Graphs (Y-axis: Score; X-axis: No. of Best Solutions Found).

Simulated Annealing Results				
Rp. No.	Best Key	Cipher Score	#best sol.	Time (s)
1	JVNITAC	-6930.51	3817	66.28
2	IGGHTEJ	-7192.06	3762	75.85
3	AYGOTOC	-6858.18	3842	67.73
4	YVAFTZZ	-7214.97	3878	68.61
5	WUJOEOD	-7012.83	3865	70.50
6	COZETAN	-6897.47	3811	66.40
7	HMNOIOC	-6899.66	3867	66.27
8	WBSDIOB	-7210.67	3885	66.38
9	FBCHTOC	-6919.98	3826	66.49
10	EOFOEBU	-7559.53	3819	67.14
Average		**-7069.59**	**3837**	**68.17**

Table 5: Simulated Annealing Results

In consequence, it can be suggested that with larger sized inputs this algorithm works a lot more efficiently in finding better solutions. Although, this is something that could be tested in further research in this area. The implementation of this algorithm has a complexity of O(N), this is linear-time, and implies that the increase in performance is dependent upon the size of the input given. Therefore, this could make this algorithm work more functionally with larger problems.

3.5 Discussion

Comparing the results for the algorithms tested as displayed in Figure 5, as well as analysing the results of all the tests, it has been shown that the genetic algorithm has been the most effective and efficient way for finding the correct solutions.

It has been the only algorithm to successfully find a correct solution. Finding the solution however is not the only factor and the time taken and number of best solutions found in the process are considered. Thus, due to all these contributors the genetic algorithm has been the most effective.

This is shown in Figure 5, as it illustrates that the genetic algorithm has nearly found the best solutions in under processing 1000 better solutions. On the contrary, the other algorithms are took longer to produce worst keys, and did not get the best result possible. However, it must be acknowledged that the hill climbing algorithm also performed well, nearly achieving the correct solution through widening the search scope. In this respect, it can also be considered a very effective algorithm and could be useful for discovering other solutions.

4 Conclusion

This paper has investigated the use of machine learning algorithms for decrypting polyalphabetic substitution ciphers. Four well known machine learning algorithms were applied; hill climbing, genetic algorithms, simulated annealing and random optimisation.

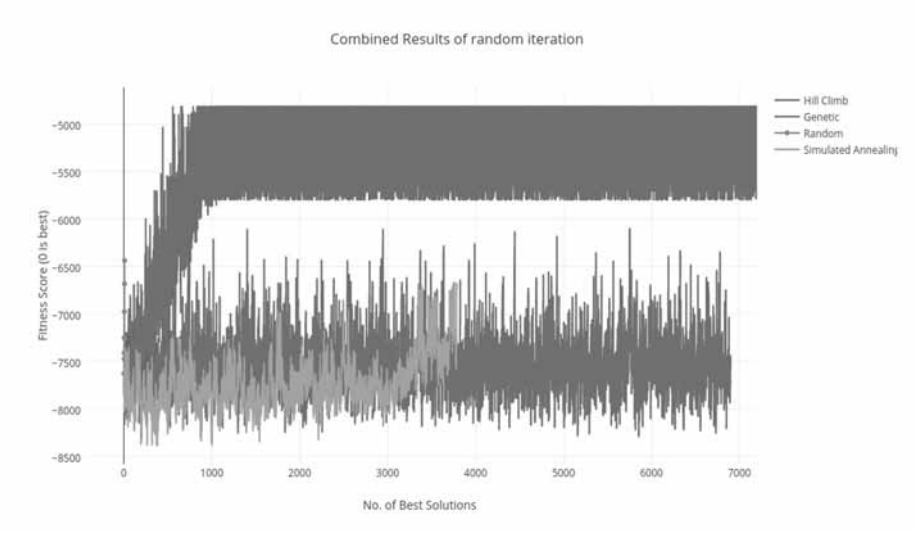

Figure 5: Comparing the graphs for a single repeat of each technique.

It can be concluded that the genetic algorithm has been the most effective algorithm used in this research, with hill climbing as second. Furthermore, they both have the potential to be useful for larger problems. The main reason why genetic algorithm has performed best is that it has achieved the correct key used in three out of ten repeats, with an average time of 23.40 (s), and a consistent number of best solutions found.

It is possibly the nature by which the algorithm works that makes it so successful; however, this would be an ideal opportunity to exploit this algorithm further in extended research.

Throughout this work the polyalphabetic technique has been adopted for use of encryption and decryption. Using this difficult type of cipher demonstrates the power of the algorithms used, though some are clearly superior to others. While every effort was made to optimise the implementation of the machine learning algorithms, further improvements may be possible. Limited experimentation was undertaken to identify suitable training parameters and additional experimentation may yield improved results.

References

[1] Stamp M. Information security: Principles and practice. 2nd edition. Oxford: Wiley, John & Sons, 2011.

[2] Martin KM. Everyday cryptography: Fundamental principles and applications. Oxford: Oxford University Press, 2012.

[3] Schneier B. Applied cryptography, second edition. 2nd ed. John Wiley & Sons, 1996.

[4] Bergmann K. Cryptanalysis Using Nature-Inspired Optimization Algorithms. Master of Science. University of Calgary, 2007.

[5] Bell J. Machine learning. 1st ed. Indianapolis, Indiana: John Wiley & Sons, Inc., 2015.

[6] Zang H, Zhang S, Hapeshi K. A Review of Nature-Inspired Algorithms. Journal of Bionic Engineering, 7, pp. S232-S237, 2010.

[7] Russell S, Norvig P. Artificial intelligence. 3rd ed. Pearson, pp.120-129, 2016.

[8] Lones M. Metaheuristics in nature-inspired algorithms. Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14, 2014.

[9] Practical Cryptography. Quadgram Statistics as a Fitness Measure. Available at: http://practicalcryptography.com/cryptanalysis/text-characterisation/quadgrams/#a-python-implementation (Accessed: 13 February 2017), 2009.

[10] Segaran T. Programming collective intelligence: Building smart web 2.0 applications. United States: O'Reilly Media, Inc, USA, 2007.

[11] Bell R. *A beginner's guide to Big O notation.* Robbell.net (open access article). Available at: https://robbell.net/2009/06/a-beginners-guide-to-big-o-notation/ (Accessed: 12 April 2018), 2018.

[12] Conway D, White JM. Machine learning for hackers. Sudbury, MA, United States: O'Reilly Media, Inc, USA, 2012.

[13] Kahn D. The codebreakers: The comprehensive history of secret communication from ancient times to the Internet. New York, NY: Simon & Schuster Adult Publishing Group 1997.

[14] Churchhouse RF. (2001) Codes and ciphers: Julius Caesar, the enigma and the internet. Cambridge: Cambridge University Press, 2001.

[15] Gollmann D. Computer security. 3rd ed. Chichester, United Kingdom: Wiley, John & Sons, 2010.