AI: Using Simulated Annealing
To break Playfair Cipher

Alan Heanue

Built with Eclipse IDE

Playfair Cipher

The Playfair Cipher The Playfair Cipher is a symmetric polygram substitution cipher invented by the Victorian scientist Sir Charles Wheatstone in 1854 (of Wheatstone Bridge fame). The cipher is named after his colleague Lord Playfair, who popularised and promoted the encryption system. Due to its simplicity, the Playfair Cipher was used at a tactical level by both the British and US forces during WWII and is also notable for its role in the rescue of the crew of PT-109 in the Pacific in 1943. Polygram substitution is a classical system of encryption in which a group of n plain-text letters is replaced as a unit by n cipher-text letters. In the simplest case, where n = 2, the system is called digraphic and each letter pair is replaced by a cipher digraph. The Playfair Cipher uses digraphs to encrypt and decrypt from a 5x5 matrix constructed from a sequence key of 25 unique letters. Note that the letter J is not included. For the purposes of this assignment, it is only necessary to implement the decrypt functionality of the Playfair Cipher. It should be noted however, that the 5x5 matrix described below can be augmented with auxiliary data structures to reduce access times to O(1). For example, the letter ‘A’ has a Unicode decimal value of 65. Thus, an int array called rowIndices could store the matrix row of a char val at rowIndices[val – 65]. The same principle can be used for columns… The encryption / decryption process works on diagraphs as follows:

What is Simulated Annealing?

The simulated annealing algorithm was originally inspired from the process of annealing in metal work. Annealing involves heating and cooling a material to alter its physical properties due to the changes in its internal structure. As the metal cools its new structure becomes fixed, consequently causing the metal to retain its newly obtained properties. In simulated annealing we keep a temperature variable to simulate this heating process. We initially set it high and then allow it to slowly 'cool' as the algorithm runs. While this temperature variable is high the algorithm will be allowed, with more frequency, to accept solutions that are worse than our current solution. This gives the algorithm the ability to jump out of any local optimums it finds itself in early on in execution. As the temperature is reduced so is the chance of accepting worse solutions, therefore allowing the algorithm to gradually focus in on a area of the search space in which hopefully, a close to optimum solution can be found. This gradual 'cooling' process is what makes the simulated annealing algorithm remarkably effective at finding a close to optimum solution when dealing with large problems which contain numerous local optimums. The nature of the traveling salesman problem makes it a perfect example.

Advantages of Simulated Annealing.

You may be wondering if there is any real advantage to implementing simulated annealing over something like a simple hill climber. Although hill climbers can be surprisingly effective at finding a good solution, they also have a tendency to get stuck in local optimums. As we previously determined, the simulated annealing algorithm is excellent at avoiding this problem and is much better on average at finding an approximate global optimum.
To help better understand let's quickly take a look at why a basic hill climbing algorithm is so prone to getting caught in local optimums.
A hill climber algorithm will simply accept neighbour solutions that are better than the current solution. When the hill climber can't find any better neighbours, it stops
In the example above we start our hill climber off at the red arrow and it works its way up the hill until it reaches a point where it can't climb any higher without first descending. In this example we can clearly see that it's stuck in a local optimum. If this were a real world problem we wouldn't know how the search space looks so unfortunately we wouldn't be able to tell whether this solution is anywhere close to a global optimum.

Simulated annealing works slightly differently than this and will occasionally accept worse solutions. This characteristic of simulated annealing helps it to jump out of any local optimums it might have otherwise got stuck in.