ECEN 2703: Discrete Mathematics

Computer Project 3: Optimal Coin Denominations

Justin Daniel Casali

1) Analysis of the code:

The task at hand is to return an array of the amount of each coin given any change value and any set of coin denominations. To make any progress one has to realize that this problem can be modeled as a staged optimization problem. In this case the change is the starting point, zero is the ending point, and the denominations represent the cost to go at every stage in-between. One can then work backwards subtracting the cost to get a node from the optimal nodes before until zero is reached, at which point the routing is complete. This puzzle is similar to the second project except in this case the cost between nodes is fixed and the number of stages varies. The program uses two matrices to store information about the cost and route of the optimal path. These matrices, 'C' for the cost up to the node and 'R' the path up to the node, are initialized with all zeros to prevent resizing memory inefficiencies of the matrices at every stage, and their first elements are filled with the cost and path from the starting node to the first stage. For the intermediate stages the program run a while loop which terminates when a zero is reached or if there are more stages then the length of change, which is theoretically impossible to reach because the maximum stages possible occurs when every node is the one coin, which will then have the length of the change itself. At every stage the program looks backwards and calculates the optimum cost to itself given the choices of the nodes one stage before it. For example if the previous optimal nodes are [3, 4, 2, 9] then the optimal cost to the 5 coin would be 9 - 5 = 4. The other choices of [-2, -1, -3] are invalid since the change has gone over the amount needed. If the minimum cost to a node has no path resulting in a positive number then the value at the node is assigned to be negative infinity and the route to it becomes zero, making it a dead node, and ignoring it for the next stages. At the end of every stage, the stage counter 'k' is incremented by one to allow for the path data at the next stage to computed. Once the intermediate stages are completed the program then removes the leftover zeros from the 'C' and 'R' matrices from initialization to be more easily in the creation of the number_coin array. The cost to the change is always going to be the change itself and there for it is redundant for it to be

noted again. The route of the optimal path to the zero node is the matrix to be utilized. At every stage the first node is always fixed to the first coin, the second node fixed to the second coin, and so on for every coin in the denomination array. The number of each different type of coin is there for how many time the route passed through that node, representing the coin being used for change. Creating an array of how many times each node was passed through then results in an array of how many times each coin was used. Before this array is returned to the user it is then scalar multiplied by the value of each coin, and its sum is compared to the amount of change to check if the coins add up to the desired change.

2) Test cases:

Calculation — 11¢	1¢	5¢	8¢
Number of Coins	1	2	0

Coins	Start	Stage 1	Stage 2	End
1¢	11	10	2	0
5¢	-	6	1	-Inf
10¢	-	3	2	-Inf

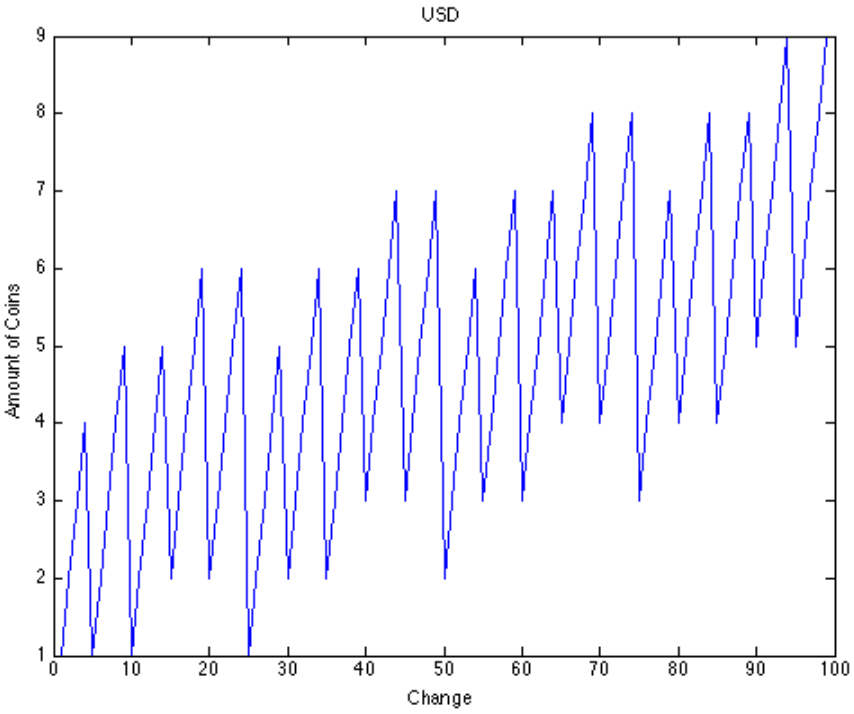
Calculation — 13¢	1¢	5¢	8¢
Number of Coins	0	1	1

Coins	Start	Stage 1	End
1¢	13	12	4
5¢	-	8	0
8¢	-	5	0

Calculation — 47¢	1¢	5¢	10¢	25¢
Number of Coins	2	0	2	1

Coins	Start	Stage 1	Stage 2	Stage 3	Stage 4	End
1¢	47	46	21	11	1	0
5¢	-	42	17	7	2	-Inf
10¢	-	37	12	2	-Inf	-Inf
25¢	-	22	12	-Inf	-Inf	-Inf

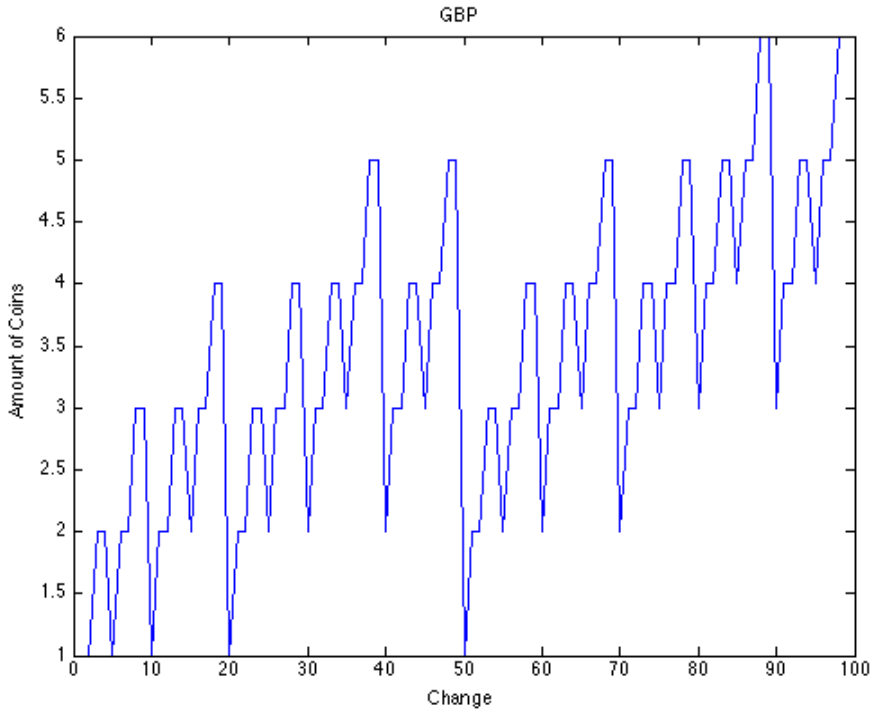
3) United States Dollar (USD):



Coin denomination:	{1, 5, 10, 25}
Largest number of coins:	9
Expected number of coins:	4.7475

This graph represents the pattern in number of coins needed to make change in USD from 1¢ to 99¢. The maximum amount of coins needed is 9 and the average amount of coins is 4.75. At every multiple of 5 and 10 the amount of coins needed plummets due to going from four pennies to one nickel, dime, or quarter. This system creates a dilemma of where people will hold on to their pennies that are otherwise worthless to manage just to use them not to get pennies back.

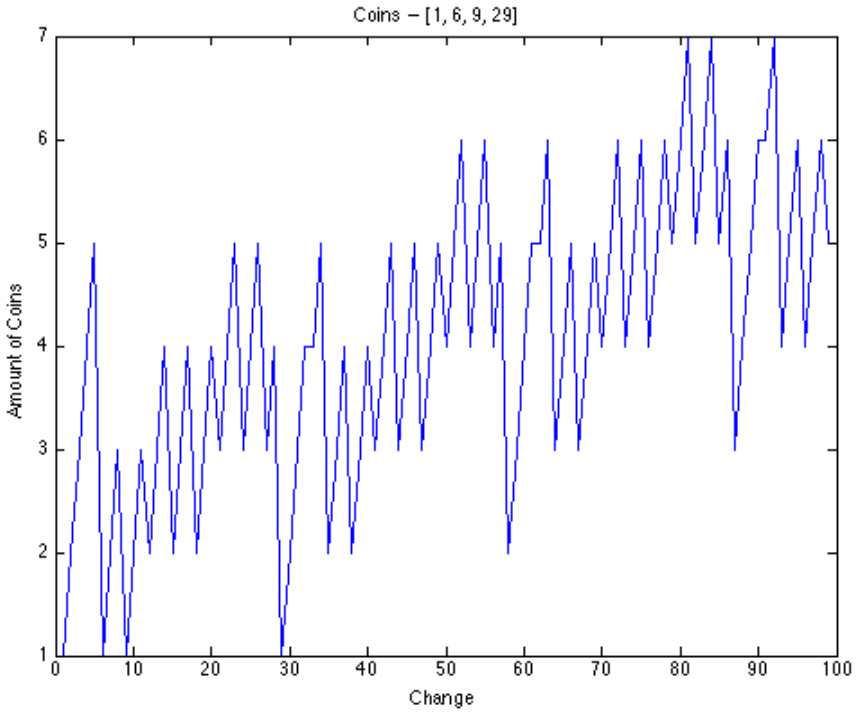
4) Pound Sterling (GBP):



Coin denomination:	{1, 2, 5, 10, 20, 50}
Largest number of coins:	6
Expected number of coins:	3.4343

As shown by the graph the British system is more efficient then the American one with a maximum of 6 and an average of 3.43 coins. The other obvious difference is that the American system uses only 4 coins while the British one uses 6, which raises the question is the efficiency worth having to circulate an extra two coins? It is also worth noting that most other countries include sales tax in the price, instead of when one pays, allowing pricing to match up more efficiently to multiples of the coins themselves, allowing for a reduction in the amount of coins needed, and elimination the need to hold onto 1¢ & 2¢ coins.

5&6) Optimal number and average amount of coins:



Coin denomination:	{1, 6, 9, 29}
Largest number of coins:	7
Expected number of coins:	4.0808

The minimum maximum number of coins was 7 coins, which was produced by 130 different sets of coins. The minimum average number of coins was 404 / 99 or 4.0808, which was uniquely produced by the set {1, 6, 9, 29}. The last set was also one of the 130 sets, so the coins 1¢, 6¢, 9¢, & 29¢ produce both the minimum maximum, and minimum average number of coins using 1¢ & 3 other coins on the range of 1¢ to 99¢. This system is too complicated to be efficient because few cashiers are going to bother returning the optimal amount of coins, instead giving the coins that they can manage to add up to the change the fastest. These numbers were calculated through exhaustive checking the maximum and average of 99^3 different coin denominations which took around 12 hours.

7) Efficiency of USD coin system:

Mathematically our current domination is not optimal in terms of producing the minimum expected number and the minimum maximum amount of coins as proven in question 5&6. Whether or not the system is efficient depends on how much inefficiency one allows. From a more practical and social standpoint the efficiencies of our system can be viewed differently. Our current system has a history to it and people will not be happy if we just changed it to coins with values that are more efficient that practical. The cost for a total overhaul of the coin system would be impossibly large, so coins would have to be phased out over time which would create periods of greater inefficient because of a mixing of the coin systems, not to mention the confusion of now having different ways to make the same change at different stage of the phase out. The numbers of the most efficient coins 1¢, 6¢, 9¢, & 29¢ are also confusing, do not work well with a base 10 system, and are too close to the original 4 coin values that it would just be a nightmare to convince people that this system is better in any way. Looking at the graphs for both system the new system also has unexpected points where the number of coins go down due to it being more efficient to use coins of a lesser value in combination then to pick the biggest coin that works until it all the change is reached. In reality if this system was implemented most cashiers are not going to bother giving out the optimal amount of change, instead they will give whatever coins are easiest to add, which in the case of the new coins would take a longer time with adding 1's, 6's, & 9's . The average efficient gain is 2/3 of a coin which is not worth an overhaul. Just phasing out pennies and rounding to the nearest 5 would increase the average efficiency by almost 2 coins, from 4.75 to 2.83 which is a far better option if any changes we to be made. For what the system is now, and given the fact they did not have computers to calculate the optimal coin denominations when the coins were created themselves, I would say the system is decently efficient mathematically, and very efficient in terms of usability and practicality.