What This Country Needs is an 18¢ Piece*

Jeffrey Shallit
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
shallit@graceland.uwaterloo.ca

September 3, 2002

Abstract

We consider sets of coin denominations which permit change to be made using as few coins as possible, on average.

1 Introduction

Most businesses in the United States currently make change using four different types of coins: 1¢ (cent),[1] 5¢ (nickel), 10¢ (dime), and 25¢ (quarter). For people who make change on a daily basis, it is desirable to make change in as efficient a manner as possible. One criterion for efficiency is to use the smallest number of coins. For example, to make change for 30¢, one could, at least in principle, give a customer 30 1-cent coins, but most would probably prefer receiving a quarter and a nickel.

Formally, we can define the *optimal representation problem* as follows: given a set of D integer denominations $e_1 < e_2 < \cdots < e_D$ and an integer $N \geq 0$, we wish to express N as a non-negative integer linear combination $N = \sum_{1 \leq i \leq D} a_i e_i$ such that the number of coins $S = \sum_{1 \leq i \leq D} a_i$ is minimized. In order that every number actually have a representation, we demand that $e_1 = 1$. If $(a_1, a_2, \ldots, a_D)$ is the D-tuple that minimizes S, then we say it is an optimal representation, and we define $\mathrm{opt}(N; e_1, e_2, \ldots, e_D) := S$.

The *optimal denomination problem* is to find denominations that minimize the average cost of making change. We assume that every amount of change between 0¢ and 99¢ is equally likely.[2] We then ask, what choice of D denominations minimizes the average number of coins needed to make change? More formally, solving the optimal denomination problem

*"What this country needs is a really good five-cent cigar." T. R. Marshall (US Vice-President), *New York Tribune*, January 4, 1920.

[1]Informally, a 1-cent coin is usually called a "penny", but this usage is frowned upon by numismatists.

[2]This assumption is probably inaccurate for several reasons, not least being the fact that many items have prices that end in the digit 9. Also, Benford's law may play a role; see Raimi [7].

for D denominations up to the limit L means determining the denominations $e_1, e_2, \ldots, e_D$ which minimize

$$\text{cost}(L; e_1, e_2, \ldots, e_L) := \frac{1}{L} \sum_{0 \leq i < L} \text{opt}(i; e_1, e_2, \ldots, e_D).$$

For the current system, where $(e_1, e_2, e_3, e_4) = (1, 5, 10, 25)$, a simple computation determines that $\text{cost}(100; 1, 5, 10, 25) = 4.7$. In other words, on average a change-maker must return 4.7 coins with every transaction.

Can we do better? Indeed we can. There are exactly two sets of four denominations that minimize $\text{cost}(100; e_1, e_2, e_3, e_4)$; namely, $(1, 5, 18, 25)$ and $(1, 5, 18, 29)$. Both have an average cost of 3.89. We would therefore gain about 17% efficiency in change-making by switching to either of these four-coin systems. The first system, $(1, 5, 18, 25)$, possesses the notable advantage that we only need make one small alteration in the current system: replace the current 10¢ coin with a new 18¢ coin. This explains the title of this article.

Figure 1 gives the optimal denominations of size D for $1 \leq D \leq 7$, and their associated costs.

D	$(e_1, \ldots, e_D)$	$\text{cost}(100; e_1, \ldots, e_D)$
1	(1)	49.5
2	$(1, 10)$ $(1, 11)$	9
3	$(1, 12, 19)$	5.15
4	$(1, 5, 18, 25)$ $(1, 5, 18, 29)$	3.89
5	$(1, 5, 16, 23, 33)$	3.29
6	$(1, 4, 6, 21, 30, 37)$ $(1, 5, 8, 20, 31, 33)$	2.92
7	$(1, 4, 9, 11, 26, 38, 44)$	2.65

Figure 1: Optimal denominations for change-making for $1 \leq D \leq 7$ denominations

Although the system $(1, 5, 18, 25)$ would be superior to the current $(1, 5, 10, 25)$ for change-making, it may be difficult to convince people to accept the removal of the popular dime. Thus it may be worthwhile to consider a different question: what single denomination could we *add* to $(1, 5, 10, 25)$ to achieve the maximum improvement in cost? The unique answer is 32¢; this improves $\text{cost}(100; 1, 5, 10, 25) = 4.7$ to $\text{cost}(100; 1, 5, 10, 25, 32) = 3.46$. If we also allow the infrequently-used 50¢ piece as a legitimate denomination, then the maximum improvement comes from adding an 18¢ piece; this improves $\text{cost}(100; 1, 5, 10, 25, 50) = 4.2$ to $\text{cost}(100; 1, 5, 10, 18, 25, 50) = 3.18$. Yet another reason to add an 18¢ piece to US coinage!

Other countries provide different problems. In Canada, the coin denominations currently in wide circulation are 1¢, 5¢, 10¢, 25¢, 100¢ (called a "loonie" for the loon on the reverse), and 200¢ (called a "toonie"). The smallest denomination of paper money in wide circulation is a \$5 bill. Assuming each amount of change between 0¢ and 499¢ is equally likely, the average cost of making change in Canada is $\text{cost}(500; 1, 5, 10, 25, 100, 200) = 5.9$. This can be best improved by adding an 83¢ coin; we have cost$(500; 1, 5, 10, 25, 83, 100, 200) = 4.578$. On the other hand, the new system of Euros introduced in Europe provides coins of denomination $.01, .02, .05, .1, .2, .5, 1$, and 2 Euros. For this system we have cost$(500; 1, 2, 5, 10, 20, 50, 100, 200) = 4.6$. This can be best improved (to average cost 3.92) by adding a coin of denomination 1.33 or 1.37 Euros.

2 Greedy methods for change-making

One nice feature of the current set of US denominations $(1, 5, 10, 25)$ is that the greedy algorithm determines the representation with the minimum number of coins. By the *greedy algorithm*, I mean the following procedure: given a number N to be represented as a non-negative integer linear combination of denominations $e_1 < e_2 < \cdots < e_D$, take as many copies a_D of the largest denomination e_D as possible, so that $a_D e_D \leq N$. Then set $N := N - a_D e_D$ and continue the procedure with the remaining smaller denominations. Use of the greedy algorithm provides a simple, easily-remembered method for making change. Not all sets of denominations have the property that the greedy method always determines the optimal representation. For example, with denominations $(1, 7, 10)$ the greedy algorithm gives the representation $14 = 10 + 1 + 1 + 1 + 1$, whereas $7 + 7$ uses fewer coins.

Unfortunately, none of the optimal sets of denominations in Figure 1 for $D \geq 3$ give optimal representations when used greedily. For example, when we try to greedily make change for 24¢ using the system $(1, 12, 19)$, we get $19 + 1 + 1 + 1 + 1 + 1$, a far cry from the optimal representation $12 + 12$.

This suggests considering a variation on the optimal denomination problem, where cost is replaced by the analogous function gcost, and we count only the cost of greedy representations. For the current system we still have gcost$(100; 1, 5, 10, 25) = 4.7$. Figure 2 displays the results for optimal sets of denominations.

We also might consider what single denomination could be added to the current US system $(1, 5, 10, 25)$ to best improve the greedy cost. It turns out that adding either a 2¢-piece or a 3¢-piece improves gcost$(100; -)$ from 4.7 to 3.9, and this is the best possible 1-coin improvement. It is interesting to note that the US actually had a 2¢-piece from 1864 to 1873, and two different 3¢-pieces: one made in silver from 1851 to 1873, and one made in nickel from 1865 to 1889.

3 Computational questions

So far we have focused on systems particular to the US, Canada, and Europe, but a good mathematician will want more general results. Let us examine the computational complexity of the problems we have studied, and some related ones.

1. Suppose we are given an amount of change to make, say N, and a system of denominations, say $1 = e_1 < e_2 < \cdots < e_D$. How easy is it to compute opt$(N; e_1, e_2, \ldots, e_D)$ or find an optimal representation $N = \sum_{1 \leq i \leq D} a_i e_i$, i.e., one which minimizes $\sum_{1 \leq i \leq D} a_i$?

D	$(e_1, \ldots, e_D)$	$\mathrm{gcost}(100; e_1, \ldots, e_D)$
1	(1)	49.5
2	$(1, 10)$ $(1, 11)$	9
3	$(1, 5, 22)$ $(1, 5, 23)$	5.26
4	$(1, 3, 11, 37)$ $(1, 3, 11, 38)$	4.1
5	$(1, 3, 7, 16, 40)$ $(1, 3, 7, 16, 41)$ $(1, 3, 7, 18, 44)$ $(1, 3, 7, 18, 45)$ $(1, 3, 8, 20, 44)$ $(1, 3, 8, 20, 45)$	3.46
6	$(1, 2, 5, 11, 25, 62)$ $(1, 2, 5, 11, 25, 63)$ $(1, 2, 5, 13, 29, 64)$ $(1, 2, 5, 13, 29, 65)$	3.13
7	$(1, 2, 5, 8, 17, 27, 63)$ ⋮ (30 other sets omitted) ⋮ $(1, 2, 5, 8, 19, 30, 67)$	2.86

Figure 2: Optimal denominations for greedy change-making

The answer depends on how N and the e_i are written down. If they are written in ordinary decimal notation, or in binary, then there is no fast algorithm known to solve this problem. In fact, it follows easily from results of Lueker [5] that this problem is NP-hard; roughly speaking this means it is at least as hard as many famous combinatorial problems, such as the travelling salesman problem, for which no polynomial-time algorithm is currently known.

If, on the other hand, N and the e_i are represented in unary, then a simple dynamic programming algorithm (e.g., [10]) solves the optimal representation problem in polynomial time.

2. Suppose we are given N and a system of denominations. How easy is it to determine if the greedy representation for N is actually optimal? Kozen and Zaks [4] have shown that this problem is co-NP-complete if the data is provided in ordinary decimal, or binary. This strongly suggests there is no efficient algorithm for this problem.

3. Suppose we are given a system of denominations. How easy is it to decide whether the greedy algorithm *always* produces an optimal representation, for all values of N? It turns

out that this problem *can* be solved efficiently; this surprising result is due to Pearson [6].

4. Suppose we are given N and a system of denominations. How easy is it to compute $\mathrm{cost}(N; e_1, e_2, \ldots, e_D)$? Since

$$\mathrm{opt}(N; e_1, e_2, \ldots, e_D) = (N+1)\mathrm{cost}(N+1; e_1, e_2, \ldots, e_D) - N\mathrm{cost}(N; e_1, e_2, \ldots, e_D),$$

any algorithm to compute cost would also provide an algorithm to compute opt. It follows that computing cost is NP-hard under Turing reductions.

5. Suppose we are given L and D and want to find an optimal set of denominations that minimizes the average cost of making change for all amounts from 0 to $L-1$? I don't know the computational complexity of this problem, but it seems quite hard. The data presented in Figure 1 were computed using a brute-force enumeration of possibilities, but with some tricks to speed up the computation.

6. A related problem is the *Frobenius problem*. Here we are given a set of D denominations $e_1 < e_2 < \cdots < e_D$ with $\gcd(e_1, e_2, \ldots, e_D) = 1$, and we want to find the largest integer N which *cannot* be expressed in the form $\sum_{1 \leq i \leq D} a_i e_i$ with the a_i non-negative integers. There is a huge literature on this problem (see, for example, Guy [2, pp. 113–114]), but only recently have researchers considered its computational complexity. Kannan [3] gave an algorithm for the Frobenius problem that runs in polynomial time if the dimension D is fixed. On the other hand, Ramírez-Alfonsín [8] has shown that the general Frobenius problem is NP-hard.

7. Another related problem is the *postage stamp problem*. There are two flavors. The "local" problem asks, given a set of D denominations $1 = e_1 < e_2 < \cdots < e_D$ and a bound h, what is the smallest integer N which cannot be represented in the form $N = \sum_{1 \leq i \leq D} a_i e_i$ where the a_i are non-negative integers and $\sum_{1 \leq i \leq D} a_i \leq h$? In the "global" version, we are given D and h and want to find the set of denominations that maximizes N. There is a large literature on these two problems (see Guy [2, pp. 123–127]).

I have recently shown [9] that the local postage stamp problem is NP-hard under Turing reductions, and that there is a polynomial-time algorithm for every fixed D.

4 Asymptotic results

Now we turn to some asymptotic estimates.

Let $\mathrm{optcost}(L, D)$ denote the minimum value of $\mathrm{cost}(L; e_1, e_2, \ldots, e_D)$ over all suitable values of $e_1, \ldots, e_D$. Can we find good upper and lower bounds on $\mathrm{optcost}(L, D)$?

One way to find an upper bound is as follows: let $k = \lceil L^{1/D} \rceil$, and define $e_i = k^{i-1}$ for $1 \leq i \leq D$. In this case, the greedy algorithm always finds the optimal representation for any N, and it turns out to be the base-k expansion of N. Letting $s_k(N)$ denote the sum of the digits in the base-k expansion of N, we find

$$\mathrm{cost}(L; e_1, e_2, \ldots, e_D) = \mathrm{gcost}(L; e_1, e_2, \ldots, e_D) = \frac{1}{L} \sum_{0 \leq i \leq L-1} s_k(i).$$

Hence

$$\operatorname{optcost}(L, D) \leq \frac{1}{L} S_{\lceil L^{1/D} \rceil}(L) \tag{1}$$

where $S_k(N) := \sum_{0 \leq i < N} s_k(i)$.

Now the quantity $S_k(N)$ has a long history; it is known that

$$S_k(N) = \frac{k-1}{2 \log k} N \log N + N F_k \left(\frac{\log N}{\log k} \right), \tag{2}$$

where F_k is a continuous, nonpositive, nowhere differentiable function of period 1; see, for example, [1]. Combining (1) and (2), we obtain the upper bound

$$\operatorname{optcost}(L, D) \leq \frac{D}{2} L^{1/D}.$$

Furthermore, using the identity

$$S_k(kN + a) = k S_k(N) + \frac{k(k-1)N}{2} + a s_k(N) + \frac{a(a-1)}{2}$$

one can compute $S_k(N)$ in time polynomial in the number of digits in k and N. This provides a fast way to compute the upper bound (1).

For a lower bound, one may reason as follows: fix a set of D denominations $e_1, e_2, \ldots, e_D$, and consider the number of different D-tuples $(a_1, a_2, \ldots, a_D)$ such that $\sum_{1 \leq i \leq D} a_i \leq k$. A simple combinatorial argument shows that this number is $\binom{D+k}{D}$. Now if $\binom{D+k}{D} \leq L/2$, it follows that for at least $L/2$ choices of N, $1 \leq N \leq L$, any representation for N must use at least $k+1$ coins, and hence $\operatorname{optcost}(L, D) \geq \frac{1}{2}(k+1)$. Now $\binom{D+k}{D} \leq \frac{(k+D)^D}{D!}$; if D is fixed and $L \to \infty$, this gives the lower bound of $\operatorname{optcost}(L, D) = \Omega(L^{1/D})$.

5 Acknowledgments

I obtained the results in Tables 1 and 2 in October 1999. Erik Demaine kindly pointed out that similar results were posted to the Usenet newsgroup `sci.math` by Jeffry Johnston and Bill Kinnersley in April 2000.

I thank Troy Vasiga, Ming-wei Wang, and Erik Demaine for their helpful comments.

References

[1] H. Delange. Sur la fonction sommatoire de la fonction "somme des chiffres". *Enseign. Math.* **21** (1975), 31–47.

[2] R. K. Guy. *Unsolved Problems in Number Theory.* Springer-Verlag, 2nd edition, 1994.

[3] R. Kannan. Lattice translates of a polytope and the Frobenius problem. *Combinatorica* **12** (1992), 161–177.

[4] D. Kozen and S. Zaks. Optimal bounds for the change-making problem. *Theoret. Comput. Sci.* **123** (1994), 377–388.

[5] G. S. Lueker. Two NP-complete problems in nonnegative integer programming. Technical Report TR-178, Computer Science Laboratory, Department of Electrical Engineering, Princeton University, March 1975.

[6] D. Pearson. A polynomial-time algorithm for the change-making problem. Technical report, Department of Computer Science, Cornell University, June 1994.

[7] R. A. Raimi. The first digit problem. *Amer. Math. Monthly* **83** (1976), 521–538.

[8] J. L. Ramírez-Alfonsín. Complexity of the Frobenius problem. *Combinatorica* **16** (1996), 143–147.

[9] J. Shallit. The computational complexity of the local postage stamp problem. *SIGACT News* **33**(1) (2002), 90–94.

[10] J. W. Wright. The change-making problem. *J. Assoc. Comput. Mach.* **22** (1975), 125–128.