Proceedings of the 2nd International Conference on Computing Innovation and Applied Physics (CONF-CIAP 2023)
DOI: 10.54254/2753-8818/5/20230547

Bilateral optimal denomination problem

Xiangmiao Yin

Department of Mathematics, University College London, London, WC1H 0AY, UK

xiangmiao.yin.21@ucl.ac.uk

Abstract. A common problem about currency denomination is the change-making problem. In this paper, a generalized denomination-related problem inspired by the optimal denomination problem is discussed, namely the bilateral optimal denomination problem. The solution maximizes a certain monetary value and thus can be applied in pricing strategies. Two key issues need to be addressed. One is how to properly formulate the problem in mathematical language. The other is how to compute the corresponding result with an algorithm. The paper learns from existing denomination algorithms and provides a solution to the problem from construction to implementation. Some limitations are needed to improve the practicality of the solution. An example of threshold discounts is used to illustrate the real-life application of the proposed problem.

Keywords: change-making problem, optimal denomination problem, canonical Denominations, pricing strategy.

1. Introduction

In-game purchases are very common in video games nowadays, and generally consumers have to pay for in-game currencies as a medium of exchange [1]. On the one hand, for some gaming companies, it is desirable to make players pay as much as possible for their currencies when purchasing in-game contents. On the other hand, consumers usually aim to minimize the cost of purchases in as convenient a manner as possible. One criterion for convenience is to make the minimum number of times of top-ups. We assume that all consumers are willing to pay for convenience, which is not true in reality [2].

The *bilateral optimal denomination problem* is to find denominations that maximize the average surplus of a goal amount, given that consumers try to minimize both surplus and number of payments in every individual purchase. It combines the *change-making problem* algorithm applied by consumers with the *optimal denomination problem* algorithm applied by companies [3]. Although the original problems are strongly related to change-making, whereas the proposed problem is not, they indeed share the same nature, as both the process of making change and top-ups are adding up different values of money. Compared to the original problem which is NP-hard, the proposed problem is even more complicated [4]. The solution to the problem is divided into two parts, namely, the proper mathematical formulation of the problem and a practical algorithm correspondingly. The paper is organized as follows. Section 2 gives a brief introduction to the basic concepts used in the proposed method, including the change-making problem and the optimal denomination problem. Section 3 develops the proposed method in detail. In Section 4, the proposed method is evaluated under more practical limitations. Section 5 concludes the paper. The original intention of this paper is to introduce

DOI: 10.54254/2753-8818/5/20230547

a new perspective to currency design, but the research further offers businesses a reference to their pricing strategies, which applies to both revenue and sales maximization. It also introduces its inverse problem, which helps consumers save money [5].

2. Related work

2.1. The change-making problem

The change-making problem addresses the question of finding the minimum number of certain denominations that add up to a given amount of money. It is a special case of the integer knapsack problem.

Definition ①. "Linear combination": Given two sets of non-negative integers $\{a_i\}$ and $\{b_i\}$, define $lc(\{a_i\},\{b_i\}) \coloneqq \sum a_i b_i$.

Definition ②. "Representation": A representation of a non-negative integer W with respect to a set of D non-negative integer denominations $\{e_1, e_2, \dots, e_D : e_i < e_{i+1}\}$ is the non-negative integer linear combination $W = \sum_{i=1}^{D} a_i e_i$. In order that every number actually has a representation, we demand that $e_1 = 1$.

Definition ③. "Optimal representation": Let the number of coins used be $S = \sum_{i=1}^{D} a_i$. If $\{e_1, e_2, \dots, e_D\}$ is the set that minimizes S, then we say it is an optimal representation, and we define:

$$opt(W; e_1, e_2, \dots, e_D) \coloneqq \{a_i\} \tag{1}$$

$$opts(W; e_1, e_2, \dots, e_D) \coloneqq S \tag{2}$$

where $W := lc(\{a_i\},\{e_i\})$

(For simplicity $\{e_i\}$ and $\{e_1, e_2, \dots, e_D\}$ are equivalent for the rest of the paper).

It can be noticed that (1) may not be unique, but an arbitrary output could be chosen without affecting further computations.

There are several methods of solving this problem, including the greedy method and dynamic programming. By simply picking the largest acceptable item first, the greedy method always generates a result, but it may not be optimal. For example, given $\{1, 3, 4\}$, the greedy method will represent 6 as $4 + 1 + 1$ whereas the optimal solution is $3 + 3$.

The problem is weakly NP-hard, but may be solved optimally in pseudo-polynomial time by dynamic programming [6-7]. The simple dynamic programming strategy will be introduced, which guarantees optimal solution disregard of canonicality of the coin system provided.

Let N be the amount needed. Then for each of the N sub-problems, i.e., change-making for 1 to N, sort all the denomination and start scanning from the smallest. Denote the minimum number of making a change of amount j with denominations up to d_i (the i_{th} denomination) $C[i, j]$.

In every iteration, if current denomination d_i is acceptable for the amount j, then 1 coin is added to the solution for that less amount. Hence, $C[i, j] = 1 + C[i, j - d_i]$. If d_i is larger than j, then we have to skip the denomination and continue with the previously calculated solution. Hence, $C[i, j] = C[i - 1, j]$.

We aim to find minimum number of coins, so at each step we choose the smaller value. Hence, the problem is formulated mathematically as $C[i, j] = min\{C[i - 1, j], 1 + C[I, j - d_i]\}$. We will obtain a matrix of size $n \times N$ where n is the number of different denominations, and $C[n, N]$ is the solution to the problem.

To get the exact coins used, we can trace back with the method described below:

Compare $C[i, j]$ and $C[i - 1, j]$, if they are equal, it means nothing was added in iteration i. Otherwise, it means d_i is used and we continue trace the amount $j - d_i$. Repeat the process until $j = 0$. As we are constructing a matrix of size $n \times N$, the time and space complexity are both $O(n, N)$. More efficient methods can be found in *Symposium on Simplicity in Algorithms* by Chan and He, which lowers the time complexity to $O(t \cdot log(t))$ by the means of a randomized algorithm [8].

DOI: 10.54254/2753-8818/5/20230547

2.2. The optimal denomination problem

Following the change-making problem, the optimal denomination problem is to find the sets of denominations that permit change to be made using as few coins as possible, on average.

"Optimal denomination": We assume that every amount of change between 1 and a limit L is equally likely, then let $\{W_i\} = \{1, 2, \dots, L\}$ be the set of all the possible amounts of change. The optimal denomination is the set $\{e_1, e_2, \dots, e_D\}$ which minimize:

$$avgs(L; e_1, e_2, \dots, e_D) \coloneqq \frac{1}{L}\sum_{i=1}^{L} opts(i; e_1, e_2, \dots, e_D) \tag{3}$$

In fact, it is not easy to find this optimal representation. We can easily estimate that, if we try all combinations of denomination choices and see which results in the lowest cost (using the brutal-force search), the running time will lie within a polynomial factor of $O(MCD)$, assuming M is the upper bound of e_i, which is considerably large even if $M = 100$.

Jeffrey Shallit concluded that the efficiency depends on how L and e_i are expressed [3]. On the one hand, it is proved by Lucker that this problem is NP-hard if they are written in ordinary decimal notation or in binary. On the other hand, J.W. Wright invented a simple dynamic programming algorithm to solve the problem in polynomial time [9]. However, in real life, many choices of denomination will be discarded. For example, the difference between two denominations should neither be too large nor too small. Some possible limitations which reduce the time complexity will be mentioned.

3. Proposed method

As an extension to the above problems, since they share the same nature of combination of denominations, the process of purchases will be referred to instead of change-making, so that several limitations can be ignored, such as the boundary of 100 and the necessity of the denomination $e_1 = 1$.

Definition ④. "Surplus": Let the amount required be P. Since the denomination of $e_1 = 1$ is not guaranteed, it might be impossible to find a representation for P, and therefore the actual amount will be $W > P$. Define $surplus = W - P$.

Definition ⑤. "Cost": The $cost$ of one purchase is a weighted mean of $surplus$ and S of this purchase. Without loss of generality, we assume it is a weighted arithmetic mean, i.e.,

$$cost = a \cdot surplus + (1 - a)S \tag{4}$$

In our problem, since P may not be represented by a linear combination of integers, consumers will try to find the few $W_1 < W_2 < \cdots < W_k$ closest to P which validate the representation, and then choose the one leading to minimum $cost$. Here, "closeness" implies that we need an upper limit for the choice of W: $W_k \le e_D \left\lceil \frac{P}{e_D} \right\rceil$. This is true because any W_b greater than this limit can be expressed as $W_a + e_i$ for an $a < b$, which means that we have an alternative reducing both $surplus$ and S.

Definition ⑥. "Optimal amount": The optimal amount $optW$ of a purchase of amount P is the $W \le e_D \left\lceil \frac{P}{e_D} \right\rceil$ linearly combined by integers which minimizes $cost(P)$.

Definition ⑦. "Extended optimal representation": Let the representation of the optimal amount be $optW = \sum_{i=1}^{D} a_i e_i$, then the extended optimal representation is the optimal representation

$$eopt(P; e_1, e_2, \dots, e_D) \coloneqq opt(optW; e_1, e_2, \dots, e_D) \coloneqq \{a_i\} \tag{5}$$

$$\text{where } optW \coloneqq lc(\{a_i\}, \{e_i\}).$$

Definition ⑧. "Bilateral optimal denomination": We assume that every amount between 1 and L is equally likely, then let $\{P_i\} = \{1,\ 2, \dots, L\}$ be the set of all the possible amounts. The bilateral optimal denomination is the set $\{e_1, e_2, \dots, e_D\}$ which maximize average surplus:

DOI: 10.54254/2753-8818/5/20230547

$$as(L; e_1, e_2, \dots, e_D) = \frac{1}{L}\sum_{i=1}^{L} surplus_i$$

$$= \frac{1}{L}\sum_{i=1}^{L} (optW_i - P_i)$$

$$= \frac{1}{L}\left(\sum_{i=1}^{L} lc(eopt(P_i; \{e_i\}), \{e_i\}) - \frac{(1+L)L}{2}\right)$$

$$= \frac{1}{L}\sum_{i=1}^{L} lc(eopt(P_i; \{e_i\}), \{e_i\}) - \frac{1+L}{2} \#(6)$$

Now we have a naïve approach for the problem as following in pseudocode.

```
for {e_i}:
    ts = 0
    for P ≤ L:
        for W ≤ e_D ⌈P/e_D⌉:
            cost = ∞
            try: change − making(W)                (*)
                if successful:
                    cost0 = a(W − P) + (1 − a)S0
                    if cost0 < cost:
                        cost = cost0
                        S = S0
                        break
                else:
                    W = W + 1
        ts = ts + s
    as = ts/L
    if as < optimalas:
        optimalas = as
        optimal = {e_i}
return optimal
```

One possible improvement to the algorithm is that, instead of repeating (*) in every loop, we can compute $change - making(W)$ for all W in the domain, and refer the value when needed.

For better understanding, if we regard the proposed problem as a "global" problem, the corresponding "local" problem would be as follows. Given a set D of denominations $\{e_1, e_2, \dots, e_D : e_i < e_{i+1}\}$ and a bound P, what is the smallest integer W represented in the form $W = \sum_{i=1}^{D} a_i e_i$ which is greater than P?

4. Real-life applications

The algorithm of the change-making problem is applied in the proposed method, and as mentioned above, if it is the greedy algorithm, we will get an output that is not guaranteed to be optimal. In such a case, the data will be invalid for further computation, so the dynamic programming algorithm has to be applied. However, in real life people are likely to apply the greedy method due to limited computing power. Now to make up for the defect of such a method, the term "canonicality" or "orderliness" is introduced [10].

DOI: 10.54254/2753-8818/5/20230547

Given a set of denomination D and goal amount W, denote the output of greedy method and optimal outputs $G(D,W)$ and $O(D,W)$ respectively. By definition, D is called canonical if $G(D,W) = O(D,W)\ \forall W$. In other words, the denomination system is canonical if we always get the minimal representation by simply applying the greedy method. For instance, the U.S. paper currency system $\{1,5,10,20,50\}$ is canonical, but the system $\{1,3,4,20,50\}$ is non-canonical, with a counterexample $6 = 4+1+1 = 3+3$.

If we only allow canonical denominations in the iteration, the method will be more reflective on consumers' real behavior, and the time complexity will be reduced, as we have a checker for canonicality that runs in $O(n^3)$ time, where n is the number of denominations in the system [11-12].

To what extent does this limitation help with the time complexity? We can make an estimation according to Xuan Cai's method of checking if a coin system starting with 1 with four or five coins is canonical [13]. From his work we know that with few exceptions, if $\{1,e_2,e_3\}$ is non-canonical, then $\{1,e_2,e_3,e_4\}$ and $\{1,e_2,e_3,e_4,e_5\}$ are also non-canonical. To check if a coin system with three coins is canonical, the following theorem is provided: the coin system $\{1,e_1,e_2\}$ is non-canonical if and only if $0 < r < e_2 - q$ where $e_3 = q \cdot e_2 + r$ and $r \in [0, e_2 - 1]$. With some algebra, $q > 1$ and $r = e_3 - q \cdot e_2 \le e_2 - 1 \Rightarrow e_3 \le (1+q)\cdot e_2 - 1 < 2e_2 - 1$. This shows that any system including $e_3 < 2e_2 - 1$ is non-canonical. This implies that at least half of combinations should be discarded. In particular, for a system of three denominations with upper bound M:

$$\left[M - \left(\left\lceil\frac{M}{2}\right\rceil + 1\right) + 1\right]\cdot M + \sum_{i=2}^{\left\lfloor\frac{M}{2}\right\rfloor} 2i - 2$$

$$= \begin{cases} \dfrac{M^2}{2} + \dfrac{(4+M)\left(\frac{M}{2}-2+1\right)}{2} - 2\left(\dfrac{M}{2}-2+1\right), even\ M \\ \dfrac{M(M+1)}{2} + \dfrac{(5+M)\left(\frac{M+1}{2}-2+1\right)}{2} - 2\left(\dfrac{M+1}{2}-2+1\right), odd\ M \end{cases}$$

$$= \begin{cases} \dfrac{3}{4}M^2 - \dfrac{1}{2}M, even\ M \\ \dfrac{3}{4}M^2 + \dfrac{1}{2}M - \dfrac{1}{4}, odd\ M. \end{cases}$$

This many combinations are skipped in the algorithm. For $M = 1000$, the proportion of discarded combinations of three is 75.12%.

Let's now look at a specific example of threshold discounts, which are given to customers when the total for a transaction reaches a specified amount. In this case, the denominations are the prices of the goods to buy, and the goal amount is the discount threshold to reach. The proposed method can then be utilized as a revenue or sales maximizing strategy. For example, if a takeaway restaurant sets several discount thresholds on its online platform and expects consumers to tend to reach these amounts, a denomination adjustment could be made by changing the price of an item or introducing a new item. Applying the proposed method enables the restaurant to set the price which maximize expected revenue, and by changing the "average $surplus$" in the last step of the proposed method to "average S", which is the average number of items bought in this situation, it manages to maximize expected sales instead.

On the other hand, customers can find the optimal combination of products to reach the threshold with the least amount of money by solving the "local" problem mentioned above. This problem does not involve average and thus is more explicit and easier. In particular, this is an instance of the knapsack problem and will not be discussed here.

DOI: 10.54254/2753-8818/5/20230547

5. Conclusion

This paper extended the change-making problem and optimal denomination problem to find the optimal denomination system maximizing the average difference between goal amount and actual amount obtained. The proposed method addresses two key issues: the mathematical formulation and the algorithmic implementation of the problem. The complexity of the proposed method is evaluated under real-life assumptions. It should be pointed out that the term "denomination" referred to in the problem is not limited to that of currencies, either real or virtual. For example, it can be generalized to a set of prices of goods. The author will be looking for non-monetary applications of the problem and researching the algebraic and statistical properties of denominations, including but not limited to canonicality and variations under certain distributions.

Acknowledgement

Firstly, I would like to show my deepest gratitude to my teachers and professors in my university, where I gained the background knowledge involved in the research. Secondly, I'm grateful for my off-campus instructors, who have provided me with valuable guidance in every stage of the writing of this thesis. Further, I would like to thank my parents and friends for their encouragement and support. Without all their enlightening advice and impressive kindness, I could not have completed my paper.

References

[1] Asadi, A. R., & Hemadi, R. (2018, November). Understanding Virtual Currencies in Video Games: A Review. In 2018 2nd National and 1st International Digital Games Research Conference: Trends, Technologies, and Applications (DGRC) (pp. 109-117). IEEE.

[2] Boden, J., Maier, E., & Wilken, R. (2020). The effect of credit card versus mobile payment on convenience and consumers' willingness to pay. Journal of Retailing and Consumer Services, 52, 101910.

[3] Shallit, J. (2003). What this country needs is an 18c piece. Mathematical Intelligencer, 25(2), 20-23.

[4] Lueker, G. S. (1975). Two NP-complete problems in nonnegative integer programming. Princeton University. Department of Electrical Engineering, Princeton University.

[5] Keller J B. (1976). Inverse problems. Amer Math Monthly, 83: 107–118.

[6] Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms. MIT Press. Problem 16-1, p. 446.

[7] Goodrich, M. T., & Tamassia, R. (2015). Algorithm design and applications (Vol. 363). Hoboken: Wiley.

[8] Chan, T. M., & He, Q. (2020). On the change-making problem. In Symposium on Simplicity in Algorithms (pp. 38-42). Society for Industrial and Applied Mathematics.

[9] Wright, J. W. (1975). The change-making problem. Journal of the ACM (JACM), 22(1), 125-128.

[10] Adamaszek, A., & Adamaszek, M. (2010). Combinatorics of the change-making problem. European Journal of Combinatorics, 31(1), 47-63.

[11] Owlree — The Change-Making Problem. (n.d.). owlree.blog. https://owlree.blog/posts/change-making-problem.html.

[12] Pearson, D. (2005). A polynomial-time algorithm for the change-making problem. Operations Research Letters, 33(3), 231-234.

[13] Cai, X. (2009, August). Canonical coin systems for change-making problems. In 2009 Ninth International Conference on Hybrid Intelligent Systems (Vol. 1, pp. 499-504). IEEE.