A greedy approach to the coin exchange problem

Ming Yu-Hsieh Min-Zheng Shieh Shi-Chun Tsai
National Chiao-Tung University
Email: {myuhsieh, mzhsieh, sctsai}@csie.nctu.edu.tw

Abstract-*We consider a variation of the well known coin changing problem:* Suppose that we buy some merchandise priced at x dollars in a currency with n kinds of coins valued at $c_1, c_2, \cdots, c_n$ dollars. What is the fewest possible number of coins required to be exchanged? *In this paper, we propose a greedy algorithm, which finds optimal solutions for the case when $c_i = c^{i-1}, i = 1, ..., n$. Also if we apply our greedy algorithm to general input, then we can determine the solution to be optimal or not in $O(nc_n)$ time.*

Keywords: Coin change, coin exchange, greedy algorithm.

1 Introduction

The *coin changing* problem[4, 2] is: *given a set of coin denominations $\{c_1, c_2, \cdots, c_n\}$ and an integer x, find the way to make change for x cents (or dollars) using the fewest possible number of coins.* This famous problem is *NP*-complete and it can be solved in polynomial time for some special sets of coin denominations by a greedy algorithm. Kozen and Zaks[4] showed that if the greedy algorithm is not optimal for some set of coin denominations $1 = c_1 < c_2 < \cdots < c_n$, then the minimal counter example x lies in $[c_3 + 1, c_n + c_{n-1}]$. This problem is a variation of the integer knapsack problem[2, 5]. It can be formulated as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i x_i = x \\ & x_i \in \mathbf{N} \cup \{0\}, \forall i \in [n] \text{ and } x \in \mathbf{N}. \end{aligned}$$

Now we consider the following case: when we buy some merchandise priced at x cents, how should we pay to make the number of coins *exchanged* as few as possible? Formally, the *coin exchange* problem is: *given a set of coin denominations $\{c_1, c_2, \cdots, c_n\}$ and an integer x, find the way to buy some merchandise priced at x cents (or dollars) by exchanging the fewest number of coins.* This problem is a special case of the *jug measuring* problem [6] and it's also a generalization of the *extended GCD* problem [3] under ℓ_1-norm, which are proved to be *NP*-hard [6, 3]. WLOG, we assume that $c_1 < c_2 < \cdots < c_{n-1} < c_n$, and in general this problem can be formulated as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} |x_i| \\ \text{s.t.} \quad & \sum_{i=1}^{n} c_i x_i = x \\ & x_i \in \mathbf{Z}, \forall i \in [n] \text{ and } x \in \mathbf{N}. \end{aligned}$$

Intuitively, $|x_i|$ is the number of the c_i-coin exchanged. We pay $|x_i|$ c_i-coins if x_i is positive, otherwise we get $|x_i|$ c_i-coins. The goal is to find a vector $(x_1, x_2, \cdots, x_n)$, with $\sum_{i=1}^{n} c_i x_i = x$, such that its ℓ_1-norm, $\sum_{i=1}^{n} |x_i|$ is minimized. For example, suppose the denominations are 1-dollar, 10-dollar, and 100-dollar bills, and we would like to buy a book priced at $x = 98$ dollars. Then, we can pay 98 dollars consisting of nine 10-dollar bills and eight 1-dollar bills, and we get no change; seventeen bills are exchanged. Or we can pay one 100-dollar bill, and we get 2 dollars change consisting of two 1-dollar bills; only three bills are used. It is obvious that the latter is a better solution. In other words, this problem can be stated as representing a given value with the fewest number of coins (or bills) from a given set of denominations.

The coin changing problem and the coin ex-

change problem share a strong resemblance in their forms and both are NP-hard[5, 7, 6, 3]. While the integer knapsack problem can be approximated efficiently[1]. However, our problem has been proved by Havas and Seifert [3] to be NP-hard even to approximate within $n^{1/O(\log^c n)}$, where c is an arbitrary small positive constant.

In this paper, we give an efficient greedy algorithm runs in $O(n)$ time for a special case of this problem (where each $c_i = c^{i-1}$), and we prove if the greedy algorithm does not give the optimal solution for some set of denominations then the minimal counterexample $x \in (0, c_n + c_{n-1})$. At last, we can determine whether the answer is optimal or not for general input in $O(nc_n)$ time, which is less than the time required by exhaustive search, $O(n^2 c_n)$.

The rest of the paper is organized as follows. In section 2, we propose a greedy algorithm to solve a special case efficiently. This algorithm can give an optimal solution for the specific case in linear time. In section 3, we apply the greedy algorithm to more general input and we can check whether the greedy algorithm produces an optimal solution or not in $O(nc_n)$ time, which is analogous to the result for the knapsack problem by Kozen and Zaks [4].

2 A greedy algorithm

For convenience, we use the following notations. A *coin system* is a vector of positive integers $\vec{c} = (c_1, c_2, \cdots, c_{n-1}, c_n)$ with $c_1 < c_2 < \cdots < c_{n-1} < c_n$. A vector $\vec{s} = (s_1, s_2, \cdots, s_{n-1}, s_n)$ is a *solution* if $\sum_{i=1}^{n} s_i c_i = x$. We say a solution is *optimal* if its ℓ_1-norm, $\|\vec{s}\|_1 = \sum_{i=1}^{n} |s_i|$, is minimum. First we consider a special case for the coin system $\vec{c} = (c^0, c^1, \cdots, c^{n-2}, c^{n-1})$, where c is an integer greater than 1. We propose a greedy algorithm that finds an optimal solution for this special case.

Theorem 2.1 *(Main result) Given a coin system $\vec{c} = (c^0, c^1, \cdots, c^{n-2}, c^{n-1})$, where $c \in N$ and any objective value $x \in N$, an optimal solution $\vec{s} = (s_1, s_2, \cdots, s_{n-1}, s_n)$ can be found in $O(n)$ time.*

We need the following observations and facts for the correctness of theorem 2.1.

Lemma 2.2 *If $\vec{s} = (s_1, \cdots, s_n)$ is an optimal solution, then $|s_i| \leq \lceil c/2 \rceil$, for i = 1, $\cdots$, n-1.*

Proof. By contradiction, suppose there exists an s_i such that $s_i > \lceil c/2 \rceil$. Let $s'_i = s_i - c$, $s'_{i+1} = s_{i+1} + 1$, then we can get a solution $\vec{s'} = (s_1, s_2, \cdots, s_{i-1}, s'_i, s'_{i+1}, s_{i+2}, \cdots, s_{n-1}, s_n)$. It is clear that $\|\vec{s'}\|_1 < \|\vec{s}\|_1$, which is a contradiction. □

We can improve the previous lemma especially when c is an odd number.

Lemma 2.3 *There exists an optimal solution $\vec{s} = (s_1, s_2, \cdots, s_{n-1}, s_n)$ such that $|s_i| \leq \lfloor c/2 \rfloor$, $\forall$ i = {1, 2, $\cdots$, n-1}.*

Proof. We only need to consider the case when c is odd, since $\lfloor \frac{c}{2} \rfloor = \lceil \frac{c}{2} \rceil$ when c is even. Suppose there exists an optimal solution $\vec{s}$ has an entry $s_i = \frac{c+1}{2}$, then we can obtain $\vec{s'} = (s_1, s_2, \cdots, s_{i-1}, s'_i, s'_{i+1}, s_{i+2}, \cdots, s_{n-1}, s_n)$ where $s'_i = -\frac{c-1}{2}$ and $s'_{i+1} = s_{i+1} + 1$. It is clear that $\vec{s}$ and $\vec{s'}$ have the same ℓ_1-norm. Similarly, if $\vec{s}$ has an entry $s_i = -\frac{c+1}{2}$, then we can obtain another optimal solution where the i-th entry satisfying the constraint. We can simply adjust $\vec{s}$ sequentially from the first entry to the $(n-1)$-th entry and obtain an optimal solution satisfying the constraints in the lemma. This proves the claim.□

The following corollary is an immediate result from Lemma 2.2. It will be used to prove the greedy choice property of the greedy algorithm.

Corollary 2.4 *Given an optimal solution $\vec{s} = (s_1, s_2, \cdots, s_{n-1}, s_n)$, we have $|\sum_{i=1}^{j} s_i c_i| < c_{j+1}$, $\forall j = 1, 2, \cdots, n-1$.*

Proof. By lemma 2.2, we have $|\sum_{i=1}^{j} s_i c_i| \leq \lceil \frac{c}{2} \rceil \cdot \frac{c^j - 1}{c-1}$. It is easy to verify that for $c \geq 2$, $\lceil \frac{c}{2} \rceil \leq c - 1$. Thus $|\sum_{i=1}^{j} s_i c_i| < c_{j+1}$. □

Before we propose the greedy algorithm, we fix the range of the objective integer x first. Actually, for Theorem 2.1, we only need to consider the case of $0 < x < c^{n-1}$. For $x \geq c^{n-1}$, we

can solve the case $x' = x \ mod \ c^{n-1}$ obtaining an optimal solution $(s'_1, s'_2, \cdots, s'_{n-1}, s'_n)$, and $(s'_1, s'_2, \cdots, s'_{n-1}, s'_n + \lfloor \frac{x}{c^{n-1}} \rfloor)$ is an optimal solution for the original case. Suppose this is not true. We know that $s'_n \geq 0$ and $s_n \geq \lfloor \frac{x}{c^{n-1}} \rfloor$ from Corollary 2.4. So that $(s_1, s_2, \cdots, s_{n-1}, s_n - \lfloor \frac{x}{c^{n-1}} \rfloor)$ will be a better solution for x', a contradiction. For $x < 0$, we can just solve the case $x' = -x$ obtaining an optimal solution $(s'_1, s'_2, \cdots, s'_{n-1}, s'_n)$, and $(-s'_1, -s'_2, \cdots, -s'_{n-1}, -s'_n)$ is an optimal one for x. Now, we propose our greedy algorithm.

Algorithm EXCHANGE(x, $\vec{c}$)
Input: $\vec{c} = (c_1, c_2, \cdots, c_n)$: the coin system
x: the objective value, $0 < x < c_n$
Output: $\vec{g} = (g_1, g_2, \cdots, g_n)$, the solution,
which is initialized to be a zero vector

1. while(x ! = 0){
2. find an i such that $|c_i - |x||$ is minimum;
3. if($x > 0$){
4. $g_i = g_i + 1$;
5. $x = x - c_i$;}
6. else{
7. $g_i = g_i - 1$;
8. $x = x + c_i$;}}

Figure 1: Algorithm EXCHANGE

In each iteration of Algorithm EXCHANGE, it is clear that $|x|$ is strictly decreasing in the cases when $c_1 = 1$. So, we know this algorithm will finally terminate, although it may fall into an infinite loop in some cases when $c_1 \neq 1$.

Lemma 2.5 *The output $\vec{g}$ of algorithm* EXCHANGE *is a solution.*

Proof. In each iteration, $g_i = g_i + 1$ when $x = x - c_i$, or $g_i = g_i - 1$ when $x = x + c_i$. Because the algorithm terminates when $x = 0$ and $\vec{g}$ is initialized to be a zero vector, we have $\sum_{i=1}^{n} g_i c_i = x$. Moreover, after each iteration $|c_i - |x||$ is smaller than $|x|$, because $c_1 = 1$, and after the if statement $|x|$ is equal to $|c_i - |x||$. So we know that the algorithm will terminate and produce a solution. □

Next, we show the algorithm EXCHANGE gives an optimal solution. There are two parts to prove: the greedy choice property of the algorithm and the optimal substructure. Before proving the greedy choice property, we define the leading coefficient which is important to the proof.

Definition 2.6 *For a solution $\vec{s} = (s_1, s_2, \cdots, s_{n-1}, s_n)$, we say s_i is the leading coefficient if $s_i \neq 0$ and $\forall j > i, \ s_j = 0$.*

It is clear that $s_\ell x > 0$, where s_ℓ is the leading coefficient, by Corollary 2.4. In fact, we can assume $s_\ell > 0$ because it is sufficient to consider the situation $x > 0$. We prove the greedy choice property as follows.

Lemma 2.7 *For an objective value $x > 0$, there exists an optimal solution $\vec{s} = (s_1, s_2, \cdots, s_{n-1}, s_n)$ such that the leading coefficient s_i has $|c_i - |x||$ minimum, and it is determined in the first iteration (line 2) of the algorithm* EXCHANGE.

Proof. Assume we have an optimal solution, satisfying Lemma 2.3, with s_j as the leading coefficient and i as chosen in line 2 of EXCHANGE, which implies $|c_i - |x||$ is minimum. Notice that $|c_i - |x||$ is minimum implies $|x|$ is closest to c_i, thus $|x| \geq \frac{c_i + c_{i-1}}{2}$ and $|x| \leq \frac{c_i + c_{i+1}}{2}$. If $j = i$, then it is done. Thus we consider $j \neq i$:

(i) **Case $j < i$:**
If $j < i - 1$, then by Corollary 2.4 we have $x < c_{i-1}$, which contradicts the fact that $|x| \geq \frac{c_i + c_{i-1}}{2}$. Thus $j = i - 1$. Since c can be even or odd, we consider the following cases.

- **Case "c is even":** We claim $s_j = s_{j-1} = \frac{c}{2}$. Suppose not, then $x = s_j c_j + s_{j-1} c_{j-1} + \sum_{k=1}^{j-2} s_k c_k < \frac{c}{2} c_j + (\frac{c}{2} - 1) c_{j-1} + c_{j-1} = \frac{c}{2} c_j + \frac{c}{2} c_{j-1}$. But since $x \geq \frac{c_i + c_{i-1}}{2} = \frac{c}{2}(c_{j-1} + c_j)$, we get a contradiction. Since $s_j = s_{j-1} = \frac{c}{2}$, we can adjust $s_{j+1} = s_i = 1$, $s_j = 1 - \frac{c}{2}$, $s_{j-1} = -\frac{c}{2}$ to satisfy this lemma without losing the optimality.
- **Case "c is odd":** Then we have $x = s_j c_j + s_{j-1} c_{j-1} + \sum_{k=1}^{j-2} s_k c_k < \frac{c-1}{2}(c_j + c_{j-1}) + c_{j-1} < \frac{c-1}{2}(c_j + c_{j-1}) + \frac{c_j + c_{j-1}}{2} =$

$\frac{c}{2}(c_{j-1}+c_j)=\frac{c_i+c_{i-1}}{2}\le x$, a contradiction. Thus for $j=i-1$, c cannot be odd.

(ii) **Case $j>i$:**
Since $c_j \ge c_{i+1} > \frac{c_{i+1}+c_i}{2} \ge x = s_jc_j + \sum_{k=1}^{j-1} s_kc_k > (s_j-1)c_j$, we know $s_j=1$.
We claim $j=i+1$. If $j>i+1$, then $x=s_jc_j+s_{j-1}c_{j-1}+\sum_{k=1}^{j-2}s_kc_k > c_j+(s_{j-1}-1)c_{j-1}$. When $c=2$, s_{j-1} can be 0 or 1. (Note that s_{j-1} cannot be -1, otherwise, we can adjust s_j's such that the leading coefficient becomes s_{j-1}.) Hence $x>c_j-c_{j-1}\ge c_{i+1}$, a contradiction. When $c>2$, since $s_j \ge -\lfloor\frac{c}{2}\rfloor$, $x>c_j-(\lfloor\frac{c}{2}\rfloor+1)c_{j-1}\ge c_{i+1}$, a contradiction. Thus j must be $i+1$. Now consider the following cases:

- **Case "c is even":** If $s_{j-1}=-\frac{c}{2}$, we can set $s_j=0$ and $s_{j-1}=\frac{c}{2}$ and get a better solution. Hence $s_{j-1}>-\frac{c}{2}$. We claim $s_{j-1}=-\frac{c-2}{2}$ and $s_{j-2}=-\frac{c}{2}$. Suppose not, then $\frac{c_i+c_{i+1}}{2}\ge x = c_j+s_{j-1}c_{j-1}+s_{j-2}c_{j-2}+\sum_{k=1}^{j-3}s_kc_k > c_j-\frac{c-2}{2}c_{j-1}-(\frac{c}{2}-1)c_{j-2}-c_{j-2}=\frac{c}{2}(c_{j-1}+c_{j-2})=\frac{c_i+c_{i+1}}{2}$, a contradiction. Thus we can set $s_j=0$, $s_{j-1}=\frac{c}{2}$, $s_{j-2}=\frac{c}{2}$ to satisfy this lemma.
- **Case "c is odd":** $x=c_j+s_{j-1}c_{j-1}+\sum_{k=1}^{j-2}s_kc_k > c_j+(s_{j-1}-1)c_{j-1}$ implies $s_{j-1}=-\lfloor\frac{c}{2}\rfloor$, otherwise we have $x>c_j+(s_{j-1}-1)c_{j-1}>\frac{c_{i+1}+c_i}{2}$, a contradiction. Thus we know there exists another solution $\vec{s'}=(s_1,s_2,\cdots,s_{i-1},\lceil\frac{c}{2}\rceil,0,0,\cdots,0)$ satisfying the lemma.

□

Lemma 2.8 *If $\vec{s}=(s_1,\cdots,s_n)$ is an optimal solution for x, then, for $s_i\neq 0$, $\vec{s'}=(s_1,\cdots,s_{i-1},s_i-sgn(s_i),s_{i+1},\cdots,s_n)$ is an optimal solution for the objective value $x-sgn(s_i)\cdot c^{i-1}$.*

Proof. We only have to deal with the case $s_i\neq 0$. Let $\|\vec{s}\|_1=t$. Suppose $\vec{s'}$ is not optimal, but $\vec{u}=(u_1,u_2,\cdots,u_{n-1},u_n)$ is the optimal solution for objective value $x-\mathrm{sgn}(s_i)\cdot c^{i-1}$. Thus $\|\vec{u}\|_1<\|\vec{s'}\|_1$. However, $\vec{v}=(u_1,u_2,\cdots,u_{i-1},u_i+\mathrm{sgn}(s_i),u_{i+1},\cdots,u_{n-1},u_n)$ would be a solution for objective value x. But $\|\vec{v}\|_1\le\|\vec{u}\|_1+1<\|\vec{s}\|_1$, a contradiction. □

Theorem 2.9 *Algorithm* EXCHANGE *gives an optimal solution.*

Proof. Immediately from *Lemma* 2.7 and *Lemma* 2.8. □

Algorithm EXCHANGE2(x, $\vec{c}$)
Input: $\vec{c}=(c_1,c_2,\cdots,c_n)$, the coin values
x, the objective value, $0<x<c_n$
Output: $\vec{g}=(g_1,g_2,\cdots,g_n)$,
which is initialized to be a zero vector

```
1. for(i = n; i > 1 and x ≠ 0 ; i = i − 1){
2.     if( c_{i−1} ≤ |x| ≤ c_i ){
3.         if( c_i − |x| ≤ |x| − c_{i−1} ){
4.             g_i = g_i + sgn(x);
5.             x = x − c_i · sgn(x);}
6.         g_{i−1} = the integer part of x/c_{i−1};
7.         x = x − g_{i−1} · c_{i−1};}}
```

Figure 2: Improved algorithm EXCHANGE2

In fact, algorithm EXCHANGE can be improved by removing the unnecessary steps as follows. In each iteration of the algorithm, because $|c_i-|x||$ becomes smaller and smaller and $|x|=|c_i-|x||$ after each iteration, we know that once c_i is chosen, c_j will not be chosen any more $\forall j>i$. That is why the algorithm uses a for-loop instead of a while-loop. Besides, we can still shorten the steps of $g_i=g_i+1$ or $g_i=g_i-1$ to at most 2 steps. This is not hard to understand in the algorithm. Moreover, algorithm EXCHANGE2 runs in time $O(n)$. This completes the proof of Theorem 2.1.

3 Determine the optimality of the greedy algorithm

We have proved that algorithm EXCHANGE gives an optimal solution for a special case. Moreover, algorithm EXCHANGE may produce optimal solutions for some other cases. In this section, we give

a proof for the upper bound of the existence of positive counterexample and we propose a method to check whether our greedy algorithm gives an optimal solution or not for general cases. For convenience, we define $m(x)$ and $g(x)$ as follows:

Definition 3.1 *Given a coin system, let m(x) denote the value of ℓ_1-norm of the optimal solution of the objective value x, and let g(x) denote the value of ℓ_1-norm of the solution by the greedy algorithm to the objective value x. When the greedy algorithm does not terminate, $g(x) = \infty$. We call a system canonical if $g(x) = m(x)$ for all x. If a system is not canonical, then a value x for which $m(x) < g(x)$ is called a counterexample for the system.*

The following lemma is the major tool to prove the upper bound of the existence of positive counterexample.

Lemma 3.2 *Let $c_1 < \cdots < c_n$ be any system of coins. For all x, coins c_i and an integer $k \in \{-1, 1\}$, $m(x) \leq m(x - k \cdot c_i) + 1$, with equality holding if and only if there exists an optimal solution $\vec{s} = (s_1, s_2, \cdots, s_n)$ for objective value x where $s_i k > 0$.*

Proof. For the inequality, by contradiction, suppose there exists a solution $\vec{t} = (t_1, t_2, \cdots, t_n)$ for $x - k \cdot c_i$ where $\|\vec{t}\|_1 < m(x) - 1$. Thus we have a solution $\vec{t'} = (t_1, t_2, \cdots, t_{i-1}, t_i + k, \cdots, t_n)$ for x and $\|\vec{t'}\|_1 \leq \|\vec{t}\|_1 + 1 < m(x)$, a contradiction.

If there exists an optimal solution $\vec{s} = (s_1, s_2, \cdots, s_n)$ for objective value x where $s_i k > 0$, it is easy to verify the equality holds by the existence of a solution $(s_1, s_2, \cdots, s_{i-1}, s_i - k, \cdots, s_n)$ for $x - k \cdot c_i$. Assume the equality holds. If there does not exist an optimal solution $(s_1, s_2, \cdots, s_n)$ for x such that $s_i k > 0$, then for every optimal solution $\vec{u} = (u_1, u_2, \cdots, u_n)$ for $x - k \cdot c_i$, $u_i k < 0$. Thus $\vec{s'} = (u_1, u_2, \cdots, u_{i-1}, u_i + k, u_{i+1}, \cdots, u_n)$ is a solution for x and $\|\vec{s'}\|_1 = \|\vec{u}\|_1 - 1 = \|\vec{s}\|_1 - 2 < \|\vec{s}\|_1$, a contradiction. So we complete the proof. □

The following theorem gives a tight upper bound for the existence of a positive counterexample. It also implies that we only need to check $O(c_n)$ objective values to decide whether our greedy algorithm works for any coin system.

Theorem 3.3 *Given a coin system $c_1 < c_2 < \cdots < c_{n-1} < c_n$, the smallest positive counterexample lies in the interval $(0, c_n + c_{n-1})$, if exists.*

Proof. Suppose $x \geq c_n + c_{n-1}$ is the smallest positive counterexample, then $m(x) < g(x)$ and for all positive integer $y < x$, $g(y) = m(y)$. Suppose $\vec{s} = (s_1, s_2, \cdots, s_n)$ is an optimal solution for objective value x and $s_i > 0$. If $i = n$,

$$\begin{array}{rcll} g(x) & = & g(x - c_n) + 1 & \text{by definition of } g \\ & = & m(x - c_n) + 1 & \text{by hypothesis} \\ & = & m(x) & \text{by Lemma 3.2} \end{array}$$

a contradiction. If $i < n$, assume $m(x) = t$,

$$\begin{array}{rcccll} \Rightarrow & m(x - c_n) & \geq & t & \text{by Lemma 3.2} \\ \Rightarrow & m(x - c_n - c_i) & \geq & t - 1 & \text{by Lemma 3.2} \end{array}$$

but because c_i is a coin used in the optimal solution of x, so

$$\begin{array}{rcccll} & m(x - c_i) & = & t - 1 & \text{by Lemma 3.2} \\ \Rightarrow & g(x - c_i) & = & t - 1 & \text{by hypothesis} \\ \Rightarrow & g(x - c_i - c_n) & = & t - 2 & \text{by definition of } g \\ \Rightarrow & m(x - c_i - c_n) & = & t - 2 & \text{by hypothesis} \end{array}$$

a contradiction. Moreover, for the three-coin system 1, 3, 4, $x = c_n + c_{n-1} - 1 = 4 + 3 - 1 = 6$ can be checked easily to be the smallest positive counterexample. So, we have completed the proof and the bound is tight. □

Definition 3.4 *A witness is an x for which $g(|x|) > g(\|x| - c_i|) + 1$ for some coin c_i.*

Lemma 3.5 *(1) Every witness is a counterexample. (2) Suppose $|x|$ is a counterexample but not a witness, then $\|x| - c_i|$ is also a counterexample, such that there exists an optimal solution $(s_1, s_2, \cdots, s_i \neq 0, \cdots, s_n)$ for objective value x. (3) If $|x| < c_n + c_{n-1}$, then $\|x| - c| < c_n + c_{n-1}$, where c is any coin value.*

Proof.

(1) Suppose x is a witness, then for some coin c_i, $m(|x|) \leq m(\|x| - c_i|) + 1 \leq g(\|x| - c_i|) + 1 < g(|x|)$. So, x is a counterexample.

(2) Since $|x|$ is a counterexample but $|x|$ is not a witness,we have $m(\|x| - c_i|) = m(|x|) - 1 < g(|x|) - 1 \leq g(\|x| - c_i|)$. Thus, $\|x| - c|$ is also a counterexample.

(3) It follows immediately from the fact that $0 \leq |x| < c_n + c_{n-1}$ and $0 \leq c < c_n + c_{n-1}$.□

Theorem 3.6 *For a given system, it is canonical if and only if that there is no witness in the interval* $(0, c_n + c_{n-1})$.

Proof. If the given system is canonical, then it is clear that there is no counterexample. Thus there does not exist any witness by Lemma 3.5-1. If the given system is not canonical, we know that there exist counterexamples and the smallest positive counterexample x with $0 < x < c_n + c_{n-1}$. If $x = \sum_{j=1}^{n} s_j c_j$ is not a witness, by Lemma 3.5, for some $s_i > 0$, $|x - c_i|$ is also a counterexample and $m(|x - c_i|) < m(x)$. If $|x - c_i|$ is still not a witness, then let $x = |x - c_i|$, and repeat this until finding a witness, since it is guaranteed that we will find a witness because x will eventually converge to 0 which can't be a counterexample. □

By the theorem above, we only need to check the existence of a witness in the range of the smallest counterexample to determine whether the greedy algorithm gives an optimal solution or not. Moreover, the algorithm EXAMINE takes time $O(nc_n)$.

Algorithm EXAMINE($\vec{c}$)
Input: $\vec{c} = (c_1, c_2, \cdots, c_n)$: the coin system
Variable: $\vec{g} = (g_0, g_1, \cdots, g_{c_n + c_{n-1} - 1})$,
which is defined in Definition 3.1

```
1. call EXCHANGE2 to compute g⃗;
2. for(i = 1 ; i < c_n + c_{n-1} ; i++ ){
3.       for(j = 1 ; j <= n ; j++ ){
4.                if(g_i > g_{|i-c_j|} + 1){
5.                         return NO;}}}
6. return YES;
```

Figure 3: Algorithm EXAMINE

4 Conclusions and Remarks

We give a linear time greedy algorithm for a special case and we can check whether it gives an optimal solution for general case in $O(nc_n)$ time. A natural possible extension is: Can we find a greedy algorithm for more general input?

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela and M. Protasi Complexity and Approximation, combinatorial optimization problems and their approximability properties, Springer-Verlag, New York, 1999.

[2] T. Cormen, C. Leiserson, R. Rivest and C. Stein Introduction to Algorithms, 2nd Ed, The MIT Press, 2001.

[3] G. Havas and J. Seifert, The complexity of the extended GCD problem, MFCS 1999: pp. 103-113.

[4] Dexter Kozen, Shmuel Zaks, Optimal Bounds for the Change-Making Problem, Theoretical Computer Science, 123 (2), pp. 377-388, 1994.

[5] C. Papadimitriou and K. Steiglitz Combinatorial Optimization, algorithms and complexity, Prentice-Hall Inc., 1982.

[6] M.-Z. Shieh, S.-C. Tsai, Jug measuring: complexity and algorithm, 2004, submitted for publication.

[7] M. Sipser, Introduction to the Theory Computation, PWS Publishing Company, 1997.