Making Change with DERIVE: Different Approaches

by Adi Ben-Israel and Wolfram Koepf

RUTCOR, Rutgers University, New Brunswick, NJ 08903-5062, U.S.A., and
Konrad-Zuse-Zentrum, Heilbronner Str. 10, 10711 Berlin, Germany

1. MAKING CHANGE

We solve here the problem of making change, using a minimal number of coins. This problem is a special case of the so-called knapsack problem, see [2], [1].

Consider a currency with N distinct coins of **weights** (i.e. face values)

$$w_1 \; > \; w_2 \; > \; \cdots \; > \; w_N \; = \; 1\,.$$

For example, in the USA the coins are
$w_1 = 50$ cents (**half dollar**), $w_2 = 25$ cents (**quarter**), $w_3 = 10$ cents (**dime**),
$w_4 = 5$ cents (**nickel**), $w_5 = 1$ cent (**penny**).
If paper currency is included, then $w_1 = 10^6$ cents (ten thousand dollars), $w_2 = 10^5$ cents, etc.
Problem: Given a cash amount of y cents, represent it using the minimal number of coins.
Formulation: Let $\mathbb{N}_0$ denote the set of nonnegative integers. The minimal number of coins required to represent y is

$$V(y) := \min \sum_{i=1}^{N} x_i \quad \text{s.t.} \quad \sum_{i=1}^{N} w_i\, x_i \; = \; y \quad (x_i \in \; \mathbb{N}_0 \;,\; (i = 1, \ldots, N))\,.$$

We solve the problem initially for the USA currency: The given formulation, using USA coins, is

$$\begin{aligned} V(y) \;:=\; \min \quad & x_1 + x_2 + x_3 + x_4 + x_5 && \text{(1a)}\\ \text{s.t.} \quad & 50\,x_1 + 25\,x_2 + 10\,x_3 + 5\,x_4 + x_5 = y && \text{(1b)}\\ & x_1\,,\; x_2\,,\; x_3\,,\; x_4\,,\; x_5 \in \mathbb{N}_0\,. && \text{(1c)} \end{aligned}$$

Let the decision x_k be made in stage k, $k \; = \; 1, 2, 3, 4, 5$. In particular:
Stage 1: Decide how many half–dollars to use. The set of allowable decisions is

$$X_1(y) \; = \; \{x \in \mathbb{N}_0 \;:\; 50\,x \leq y\}\,.$$

The optimal value function $V_1(y)$ is the same as $V(y)$ of (1a).
Stage 2: The set of allowable decisions is

$$X_2(y) \; = \; \{x \in \mathbb{N}_0 \;:\; 25\,x \leq y\}\,.$$

The optimal value function is

$$\begin{aligned} V_2(y) \;:=\; \min \quad & x_2 + x_3 + x_4 + x_5\\ \text{s.t.} \quad & 25\,x_2 + 10\,x_3 + 5\,x_4 + x_5 = y\\ & x_2\,,\; x_3\,,\; x_4\,,\; x_5 \in \mathbb{N}_0\,. \end{aligned}$$

Stage 5: Decide on x_5 (the number of penny coins), with allowable decisions

$$X_5(y) \; = \; \{x \in \mathbb{N}_0 \;:\; x \leq y\}\,.$$

The optimal value function is $V_5(y) = y$.

We can compute the optimal value functions recursively, for all relevant $y \in \mathbb{N}_0$.

$$\begin{aligned}
V_5(y) &= y \\
V_4(y) &= \min\{x + V_5(y-5x) \ : \ 5x \le y \ , \ x \in \mathbb{N}_0\} \\
V_3(y) &= \min\{x + V_4(y-10x) \ : \ 10x \le y \ , \ x \in \mathbb{N}_0\} \\
V_2(y) &= \min\{x + V_3(y-25x) \ : \ 25x \le y \ , \ x \in \mathbb{N}_0\} \\
V_1(y) &= \min\{x + V_2(y-50x) \ : \ 50x \le y \ , \ x \in \mathbb{N}_0\}
\end{aligned}$$

No computation is required in stage 5, since $V_5(y) = y$ for all $y \in \mathbb{N}_0$. The optimal value function V_1 has to be computed only for the given initial cash, say $y = y_1$, however the intermediate o.v. functions (V_4, V_3 and V_2) have to be computed for all values of $y \le y_1$.
An alternative formulation: For symbolic computation we prefer recursion with a single function, rather than the 5 different functions V_k, in the above computation. It is indeed possible to use here the function $V(y)$ of (1a), and do recursion on the values of the state variable y. This recursion is

$$V(y) = 1 + \min\{V(y-1), V(y-5), V(y-10), V(y-25), V(y-50)\} \tag{4a}$$

$$V(0) = 0 \quad \text{(the boundary condition)} \tag{4b}$$

$$V(y) = \infty \quad \text{if } y < 0 \ , \ \text{(in order to exclude negative arguments of } V \text{ in (4a)).} \tag{4c}$$

2. FIRST DERIVE SOLUTION

We solve this problem with Derive Version 3.0. First introduce the USA coins

```
coins:=[1,5,10,25,50]
```

The following Derive program is a direct translation of (4a)–(4c).

```
V(n):=IF(n=0,0,
      1+MIN(VECTOR(IF(n>=ELEMENT(coins,k_),V(n-ELEMENT(coins,k_)),inf),
            k_,1,DIMENSION(coins))))
```

For example, `V(7)` simplifies to 3, one nickel and two pennies.
This function may not work in older versions of Derive. In Derive Version 2.60, we get "?" when simplifying `MIN(1,inf)`. This bug was corrected in Version 3.0. If you use an older version, replace ∞ in `V`(n) by a sufficiently large number, say 10^6.

A table with the first 10 values of n and `V`(n) is (recall that `x`‘ denotes the transpose of `x`)

`VECTOR([n,V(n)],`n, 1, 10)‘ giving $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & 1 \end{bmatrix}$.

3. SECOND DERIVE SOLUTION

However, the above program is slow, because the values of `V`(n) are computed from scratch each time they are needed, see also § 10.6 of the DERIVE manual. To avoid this we propose another program which records the previous values through the `APPEND` command

```
V1(n) := ITERATE(APPEND(g_,[[DIMENSION(g_),1+MIN(VECTOR(
              IF(DIMENSION(g_)-ELEMENT(coins,k_)<0,inf,
                 IF(DIMENSION(g_)-ELEMENT(coins,k_)=0,0,
                      ELEMENT(g_,DIMENSION(g_)-ELEMENT(coins,k_)+1,2))),
                             k_,1,DIMENSION(coins)))]]),g_,[[0,0]],n)
```

This generates a list of `[k,V(k)]` for $k = 0, 1, \ldots, n$. For example

`V1(10)`‘ simplifies to $\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & 1 \end{bmatrix}$.

The sought value `V(n)` is the `(n+1,2)`th of `V1(n)`. It is computed by

```
V2(n) := ELEMENT(V1(n),n+1,2)
```

For example, `V2(59)` gives `6`.

4. THIRD DERIVE SOLUTION

As shown in [2], the solution of the current problem agrees with the **greedy solution**, using the maximal number of the highest–valued coin before trying lower–valued coins. The greedy solution is computed by

$$\begin{aligned} x_1 &= \left\lfloor \frac{y}{50} \right\rfloor \\ x_2 &= \left\lfloor \frac{y - 50\,x_1}{25} \right\rfloor \\ x_3 &= \left\lfloor \frac{y - 50\,x_1 - 25\,x_2}{10} \right\rfloor \\ x_4 &= \left\lfloor \frac{y - 50\,x_1 - 25\,x_2 - 10\,x_3}{5} \right\rfloor \\ x_5 &= y - 50\,x_1 - 25\,x_2 - 10\,x_3 - 5\,x_4 \end{aligned}$$

where $\lfloor \alpha \rfloor$ denotes the largest integer $\leq \alpha$. Programming this solution (recursively) in Derive we get

```
GREEDY(n) := IF(n=0,0,1+GREEDY(MIN(VECTOR(
                  IF(n>=ELEMENT(coins,k_),n-ELEMENT(coins,k_),inf),
                                               k_,1,DIMENSION(coins)))))
```

For example, `GREEDY(59)` gives `6`, in agreement with `V2(59)`.

However, the optimality of the greedy solution is a property of the currency, not of the problem. To see this consider the currency obtained by adding a 20 cents coin to the USA coinage (see [2, Example 2]):

`coins` := [1,5,10,20,25,50]

then `V2(40)` gives `2` but `GREEDY(40)` is `3`.

Currencies for which the greedy solution is optimal are called **orderly**, see [3]. Necessary and sufficient conditions for orderliness are given in [2] and [1].

Another example of orderly currency is the coinage used in Germany

```
coins:=[1,2,5,10,50]
```

Here `V1(10)‘` gives $\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0 & 1 & 1 & 2 & 2 & 1 & 2 & 2 & 3 & 3 & 1 \end{bmatrix}$.

5. FINAL REMARKS

It is instructive to compare the running times of the various algorithms mentioned above. We suggest that you compare the running times for the calculations `VECTOR([n,V(n)],n,1,20)`, `V1(20)`, `V1(100)`, and `VECTOR([n,GREEDY(n)],n,1,100)` on your computer. Then you realize that additional information, such as that the greedy solution is applicable, can shorten the computation significantly. On slow machines this could be the difference between being able to solve a problem or not. You may try to program the greedy solution iteratively which (for large n) is much faster than our recursive approach. Interested readers may obtain such a solution from the authors.

Finally we consider the

Fibonacci currency: The Government of Fibonia decided to base its currency on the **Fibonacci numbers** $\{F_n\}$, defined by

$$F_1 = F_2 = 1 \ , \ F_n \ := F_{n-1} + F_{n-2} \ , \quad n \ \geq \ 3 \ .$$

These numbers can be computed recursively as follows (see § 10.6 in the DERIVE manual)

```
FIB_AUX(n,next,current) := IF(n=0,current,FIB_AUX(n-1,next+current,next))
                 FIB(n) := FIB_AUX(n,1,0)
```

The first 15 Fibonacci numbers are `VECTOR(FIB(n),n,1,15)` which simplifies to

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 6105] .
```

Every positive integer y can be represented as a sum of Fibonacci numbers

$$y \ = \ \sum_{i=1}^{k} F_i \, x_i \ , \quad \text{where } k \text{ is determined by } F_k \leq y < F_{k+1} \tag{5}$$

and all $x_i \ = \ 0$ or 1. Moreover, the minimal number of Fibonacci numbers representing y is given by the greedy solution, computed recursively by,

$$V(y) \ = \ 1 + V(y - F_k) \ , \tag{6}$$

see e.g. [4, p. 6]. The minimal representation (5) cannot have $x_{n-1} = x_n = 1$ for any n, since then F_{n+1} would be used instead of F_{n-1} and F_n.

For this reason the Fibonian currency uses every second Fibonacci number, i.e. the Fibonacci numbers with even indices $\{F_2 \, , \, F_4 \, , \, F_6 \, , \, F_8 \, , \, F_{10} \, , \ \ldots \ \}$. The first 5 values are `coins := VECTOR(FIB(2*n),n,1,5)` giving `[1, 3, 8, 21, 55]`. The Fibonacci currency is orderly, see [2], so we can use `GREEDY(n)` instead of the slower `V2(n)`.

As an exercise, we ask the reader to compute `VECTOR([n,GREEDY(n)],n,1,99)` for the US, German and Fibonacci currencies. Then plot the three graphs, and compare. Conclude that the Fibonacci currency is quite economical, in the sense that fewer coins are required, on the average, than in the other two currencies (why?).
Note: To plot set **Plot Options State** to **Connected**.

References

[1] T.C. Hu and M.L. Lenard, "Optimality of a heuristic solution for a class of knapsack problems", *Operations Res.* **24**(1976), 193–196

[2] M.L. Magazine, G.L. Nemhauser and L.E. Trotter, Jr., "When the greedy solution solves a class of knapsack problems", *Operations Res.* **23**(1975), 207–217

[3] S.B. Maurer, "Disorderly currencies", *Amer. Math. Monthly* **101**(1994), p. 419

[4] M.R. Schroeder, *Number Theory in Science and Communication*, Second Edition, Springer–Verlag, 1986