Factory Planning Algorithm

Alex Loomis

May 25, 2020

Given a collection of recipes, create a matrix M where each column of M is a recipe, and each row corresponds to an item. For instance, if mining creates two iron per second and smelting consumes five iron to create one steel,

$$M = \begin{pmatrix} 2 & -5 \\ 0 & 1 \end{pmatrix}.$$

Multiplying by a column of recipes, R, yields a column of outputs produced, P. Mining two times and smelting once produces one steel with a deficit of one iron:

$$R = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad P = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad MR = P.$$

The user can specify up to rank M values of P, however, they must be chosen such that the submatrix of corresponding rows has linearly independent rows. If they specify fewer, the remaining choices can be assumed to be zero, to maximize self-sufficiency of the production line. Our goal then is to solve for R.

If the rank of M is less than the number of columns of M then several recipes are linearly dependent, so we can choose rank M linearly independent columns. The system now has a unique solution, so we solve.

This has been implemented in Haskell. As a test input, try

```
>> solutions = filter (all (>= 0)) . solveFactory angIron $ ironOut
>> solutions

[ fromList [ (Name "Molten iron", 0.0)
           , (Name "Iron plate", 0.0)
```

```
           , (Name "Iron ingot", 0.0)
           , (Name "Process iron", 1.875) ]
, fromList [ (Name "Molten iron", 0.9375)
           , (Name "Iron plate", 2.8125)
           , (Name "Iron ingot", 0.46875)
           , (Name "Process iron", 1.875) ]]

>> outputs angIron $ solution !! 0

fromList [ (Item "iron ore", -7.5)
         , (Item "coke", 0.0)
         , (Item "processed iron", 3.75)
         , (Item "molten iron", 0.0)
         , (Item "iron plate", 0.0)
         , (Item "iron ingot", 0.0) ]

>> outputs angIron $ solution !! 1

fromList [ (Item "iron ore", -7.5)
         , (Item "coke", -0.9375)
         , (Item "processed iron", 0.0)
         , (Item "molten iron", 0.0)
         , (Item "iron plate", 11.25)
         , (Item "iron ingot", 0.0) ]
```

where

```
procIron :: Recipe Double
procIron = ("Process iron",)
  [ ("iron ore", -4)
  , ("processed iron", 2) ]

ironIngot :: Recipe Double
ironIngot = ("Iron ingot",)
  [ ("processed iron", -8)
  , ("coke", -2)
  , ("iron ingot", 24) ]
```

```
moltenIron :: Recipe Double
moltenIron = ("Molten iron",)
  [ ("iron ingot", -12)
  , ("molten iron", 120) ]

ironPlate :: Recipe Double
ironPlate = ("Iron plate",)
  [ ("molten iron", -40)
  , ("iron plate", 4) ]

angIron :: Cookbook Double
angIron = [procIron, ironIngot, moltenIron, ironPlate]

ironOut :: Constraint Double
ironOut = [ ("iron ore", Just (-7.5)), ("iron plate", Nothing) ]
```

(output reformatted for clarity).