Greedy Algorithm

Luis David Bedon Gomez

5 9 2017

The Greedy Algorithm is an optimization algorithm, that picks up the objects with the greatest value until the constraint is reached. See e.g. Data Science MIT

If there are more objects with a subsequent value and the value of the objects chosen so far is less then the constraint, the objects with the now highest value are chosen and so on.

Building a greedy algorithm:

Generate/import file with the objects and values to optimize for given a constraint:

```
# Generate the data
set.seed(120)
nobjects<-15
table1<-data.frame(object=letters[1:nobjects],
                   number=rpois(nobjects,3),
                   value=round(1+abs(rnorm(nobjects,mean=nobjects*.6,sd=nobjects*.6)),0))

head(table1,8)
```

```
##   object number value
## 1      a      2     5
## 2      b      1    11
## 3      c      2     2
## 4      d      4     8
## 5      e      1    19
## 6      f      3     6
## 7      g      7     6
## 8      h      1     7
```

Greedy-Algorithm

Algorithm

The Greedy-Algorithm should:

- Calculate the global total value
- Arrange by total value
- Begin the computation

This can be achieved as follows:

1. Reorganize objects by total value in descent order:

```
library(dplyr)
table2<-table1 %>% mutate(total.value=number*value) %>% arrange(desc(total.value))
print(table2)
```

```
##    object number value total.value
## 1       o      6    21         126
## 2       j      5    23         115
## 3       i      5    11          55
## 4       l      6     8          48
## 5       g      7     6          42
## 6       d      4     8          32
## 7       k      3     8          24
## 8       e      1    19          19
## 9       f      3     6          18
## 10      m      1    15          15
## 11      b      1    11          11
## 12      a      2     5          10
## 13      h      1     7           7
## 14      c      2     2           4
## 15      n      1     2           2
```

Note, that by creating the new column *total.value* and sorting it from high to low, the order changes in respect to the value for each object!!

2. Set a constrain:

```
constraint<-sum(table2$total.value)/1.1
print(constraint)
```

```
## [1] 480
```

3. Run the iteration:

```
k<-1
result<-{}
resvalue<-0
for(i in 1:length(table2$object)){
  if(resvalue<=constraint){
    for(j in 1:table2$number[i]) {
       if((resvalue+table2[i,3])<=constraint) {
         resvalue<-resvalue+table2[i,3]
         result[k]<-as.character(table2$object[i])
         k<-k+1
       }
    }
  }
}
print(resvalue)
```

```
## [1] 479
```

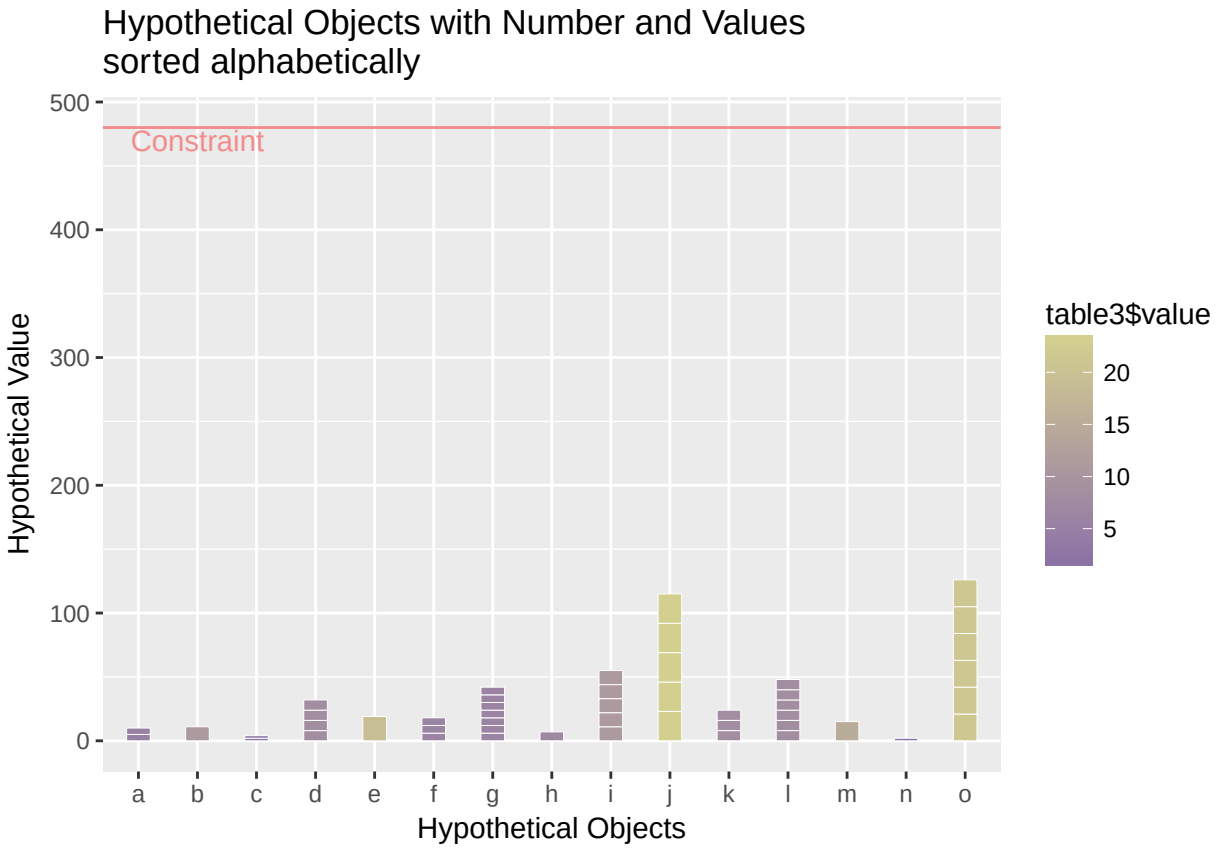
```
print(result)
```

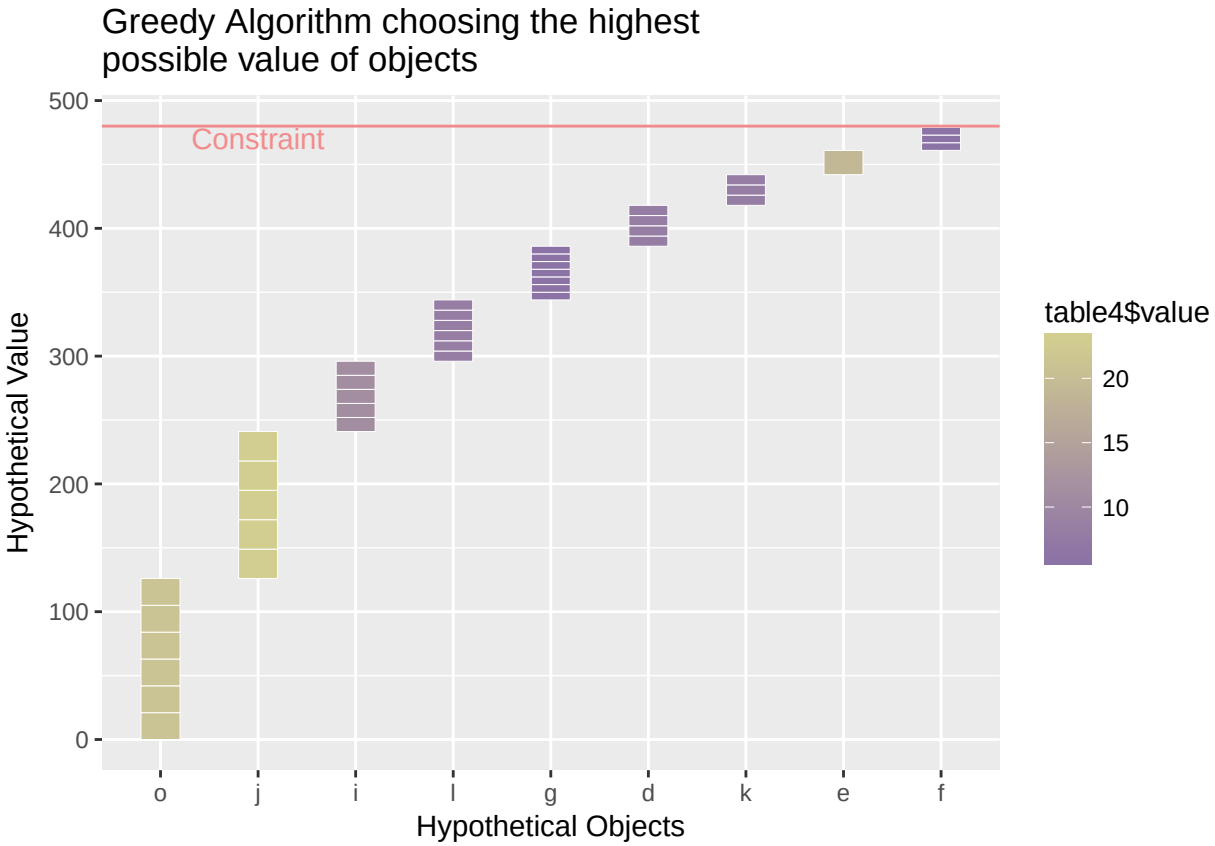
```
##  [1] "o" "o" "o" "o" "o" "o" "j" "j" "j" "j" "j" "i" "i" "i" "i" "i" "l"
## [18] "l" "l" "l" "l" "l" "g" "g" "g" "g" "g" "g" "g" "d" "d" "d" "d" "k"
## [35] "k" "k" "e" "f" "f" "f"
```

```
print(paste("Constraint=",constraint,"vs. Reached Value=",resvalue))
```

```
## [1] "Constraint= 480 vs. Reached Value= 479"
```

Visualization





Some considerations:

- The greedy algorithm should fail by maximal the modulo of the highest possible value to the respective highest value available.
- The greedy algorithm resambles a decomposition in prime numbers
- It represents a up-down algorithm with no branches
- Possible uses could be:
  - Choosing coins and notes to give the change back after paying cash

Appendix

The code for the plots:

```
library(ggplot2)
library(extrafont)
#font_import(pattern="Interstate-Regular")

# Cumulative table with all objects and values
numobject<-function(x) as.character(rep(table1$object[x],table1$num[x]))
cumobject<-sapply(1:length(table1$object),numobject)
table3<-data.frame(object=unlist(cumobject))
table3$value<-sapply(1:length(table3$object),
                     function(x) table1$value[table1$object==table3$object[x]])
```

```
subset_t3<-function(x) table3[table3$object==as.character(x),2]
cumsubset_t3<-function(y) {
              sapply(1:length(subset_t3(y)),
                     function(x) sum(subset_t3(y)[1:x]))
              }
cumallsubset<-sapply(as.character((table1$object)),cumsubset_t3)
table3$cum<-unlist(cumallsubset)

# Cumulative table after algorithm results
s<-(sapply(result,function(x) grep(x,table2$object)))

table4<-data.frame(object=table2$object[s],value=table2$value[s])
table4$cum<-sapply(1:(length(table4$value)),function(x) sum(table4$value[1:x]))

lresult<-length(table(result))

# Plot with all objects and values
ggplot()+
  geom_tile(aes(x=table3$object,
                y=table3$cum-table3$value/2,fill=table3$value),
            width=.4,
            height=table3$value,
            colour="white")+
  geom_hline(yintercept = constraint,colour=rgb(.95,.55,.55))+
  annotate("text",x=2,y=constraint-10,label= "Constraint",colour=rgb(.95,.55,.55))+
  scale_fill_gradient2(low=rgb(.55,.45,.65,1),
                       high=rgb(.35,.65,.25,.8),
                       midpoint = 25,mid=rgb(.85,.85,.55,.2))+
  ggtitle(label="Hypothetical Objects with Number and Values\nsorted alphabetically")+
  xlab("Hypothetical Objects")+
  ylab("Hypothetical Value")#+
```

```
  #theme(text=element_text(family="Interstate"))

# Plot showing the algorithm resuts for the given constraint
ggplot()+
  geom_tile(aes(x=reorder(table4$object,table4$cum),
                y=table4$cum-table4$value/2,fill=table4$value),
            width=.4,
            height=table4$value,
            colour="white")+
  geom_hline(yintercept = constraint,colour=rgb(.95,.55,.55))+
  annotate("text",x=2,y=constraint-10,label= "Constraint",colour=rgb(.95,.55,.55))+
  scale_fill_gradient2(low=rgb(.55,.45,.65,1),
                       high=rgb(.35,.65,.25,.8),
                       midpoint = 25,mid=rgb(.85,.85,.55,.2))+
  ggtitle(label="Greedy Algorithm choosing the highest\npossible value of objects")+
  xlab("Hypothetical Objects")+
  ylab("Hypothetical Value")
```