coordinated_ordering_3

Group B

08/06/2020

Coordinated Ordering

1. Deterministic/continous Demand: The same amount of demand is there every instance of time and demand are met at every instance of time, thus no time varying demand
2. Instantaneous replenishment: As soon as you place an order, the item arrives. Lead time= 0
3. No shortage: As soon as stock level on Y axis reaches 0, place an order for another quantity q. thus instanstaneous replenishment.

In the data set "Data_ordering.xlsx" parts are represented by material ID column. i =denotes part $i \in \mathcal{I} \quad \{i_1, i_2, ..., i_{62}\}$
d_i = demand per day for part i

data Extraction

```
library("readxl")
product_data <- read_excel("Data_ordering.xlsx",sheet = "product data")
box_data <- read_excel("Data_ordering.xlsx",sheet = "box data")

#rack data
Total_racks=8
levels_per_rack=4
rack_length= 6000
rack_width=1750
rack_height=300
```

Convert to Boxes

Convert demand d_i, quantity q_i and unit pricep_i all from part i to box bi which is the capacity of a box for part i

$de_i = \frac{d_i}{b_i} \rightarrow$ Demand for box with part i
$\sqrt{\ } r_i = b_i \cdot p_i \rightarrow$ Price for box with material i
$de_i = demand\_per\_year$
, $pr_i = box\_cost$

```r
#assuming there are 365 working days in a year
#converting everything to boxes

product_data$demand_per_year= ceiling((product_data$`demand per day` *365)/ product_data$`pieces/box`)

product_data$box_cost= product_data$`pieces/box` * product_data$price

product_data
```

```
## # A tibble: 62 x 7
##    `material ID` `demand per day` `box ID` `pieces/box` price demand_per_year
##    <chr>                    <dbl>    <dbl>        <dbl> <dbl>           <dbl>
##  1 7305667+74                  12  6203060           45  1.51              98
##  2 7305669+77                  30  6203059           15  7.62             730
##  3 7305670+77                  30  6203059           15  1.62             730
##  4 7305673+76                  30  6203059           16  1.51             685
##  5 7305674+76                  30  6203059           16  1.51             685
##  6 7305817+74                  60  6203060           16  4.55            1369
##  7 7305819+79                  30  6203059            6 11.1             1825
##  8 7305820+79                  30  6203059            6  2.86            1825
##  9 7305823+73                  30  6203059           30  4.05             365
## 10 7305824+73                  30  6203059           30  2.56             365
## # ... with 52 more rows, and 1 more variable: box_cost <dbl>
```

Append Ordering cost for each item

```r
order_cost <- double(length(product_data$`box ID`))

  for(j in 1: length(box_data$`box ID`)){
    for (k in 1:length(product_data$`box ID`)) {
      if(box_data$`box ID`[j]==product_data$`box ID`[k]){
        order_cost[k] <- box_data$`ordering cost (€)`[j]
      }
    }

  }

product_data$ordering_cost <- order_cost
product_data
```

```
## # A tibble: 62 x 8
##    `material ID` `demand per day` `box ID` `pieces/box` price demand_per_year
##    <chr>                    <dbl>    <dbl>        <dbl> <dbl>           <dbl>
##  1 7305667+74                  12  6203060           45  1.51              98
##  2 7305669+77                  30  6203059           15  7.62             730
##  3 7305670+77                  30  6203059           15  1.62             730
##  4 7305673+76                  30  6203059           16  1.51             685
##  5 7305674+76                  30  6203059           16  1.51             685
##  6 7305817+74                  60  6203060           16  4.55            1369
##  7 7305819+79                  30  6203059            6 11.1             1825
##  8 7305820+79                  30  6203059            6  2.86            1825
##  9 7305823+73                  30  6203059           30  4.05             365
```

```
## 10 7305824+73                  30  6203059           30  2.56              365
## # ... with 52 more rows, and 2 more variables: box_cost <dbl>,
## #   ordering_cost <dbl>
```

c_i^{or} = ordering cost for part i,

c_i^{sh} = stock holding cost rate based on unit price p_i and interest rate h,   . $c_i^{sh} = p_i \cdot h$

Separate Ordering (SO)

calculate EOQ for each part i:

$$q_i^* = \sqrt{\frac{2 \cdot de_i \cdot (c_i^{or} + c^{-or})}{pr_i \cdot h}}$$

A. Calculate The EOQ:

```r
#EOQ
#dei= demand_per_year, cori= ordering cost for box i
#cord= ordering cost, pri= box cost, h= interest rate
so_eoq_fun <- function(dei,cori,cord,pri,h){
  eoq<- sqrt((2*dei*(cori+cord))/(pri*h))

  return(eoq)
}

vec_so_eoq_fun <- Vectorize(so_eoq_fun(dei=product_data$demand_per_year,cori=product_data$ordering_cost
```

```
## Warning in formals(fun): argument is not a function
```

```r
#Append Vector eoq to the table
product_data$eoq <- ceiling(vec_so_eoq_fun)
```

B Number of lanes you occupy with EOQ

Constraints

$b_{i(sorting)}$ = sorting column lenght or width.This determines the rack width

$b_{i(-sorting)}$ =if sorted by length then the value is width and vice versa,

Add $b_{i(sorting)}$ and $b_{i(-sorting)}$ to product_data table

```r
#create a space holder
b_sorting <- double(length(product_data$`box ID`))
b_not_sorting <-double(length(product_data$`box ID`))

  for(j in 1: length(box_data$`box ID`)){
    for (k in 1:length(product_data$`box ID`)) {
      if(box_data$`box ID`[j]==product_data$`box ID`[k]){

        if(box_data$sorting[j]=="width"){
          b_sorting[k] <- box_data$width[j]
```

```
        b_not_sorting[k]<- box_data$length[j]

      }else{
        b_sorting[k] <- box_data$length[j]
        b_not_sorting[k]<- box_data$width[j]
      }
    }
  }

}


product_data$b_sorting <- b_sorting
product_data$b_not_sorting <- b_not_sorting
```

E.g Using box_ID 6203060, sorted by length $b_{i(sorting)} = 396$, $b_{i(-sorting)} = 297$ Also for material ID 7305667+74 with $q_i = 214$ therefore, $rack_{length} = 6000$

1. Number of boxes in a lane:

$$n_i = \frac{rack_{length}}{b_{i(-sorting)}} = \frac{6000}{297} = 20.20 = 20 boxes$$

2. How many lanes part i will occupy if you order some number of boxes

$$lane_i = \frac{q_i}{n_i} lane_i = q_i \cdot \frac{b_{i(-sorting)}}{rack_{length}} = 214 \cdot \frac{297}{6000} = 10.593 = 11 lanes$$

```
lane <- ceiling(product_data$eoq * (product_data$b_not_sorting/rack_length))
lane
```

```
##  [1]  11  30  65  63  63  39  62 122  21  26  25  73  50  89  50  29  21  21  19
## [20]  45  31  17  22  21  20  50  24  43  42  16  17   9  10  11  14  43 151  68
## [39]  20  15  78  60  62  40  40  42  44 132 196  15  28  36  28  32  20  15  56
## [58] 217  46  28  20  20
```

(C) Do we meet the capacity contraint?

3. Total number of lanes constraints

Collapsing Rack 4 levels in a rack and joining the 8 racks to become 1 level.

Total rack width available:

$$rack_{total_{Width}} = (rack_{width} \times 4 \times 8)$$

```
rack_total_width <- rack_width * 4 * 8
rack_total_width #56000 mm
```

```
## [1] 56000
```

Please note the coefficients are are different for every ith item except racklength which is the same for all items.Also the left and right hand side of the equation need to be in mm.

$$\sum_{i=1}^{n=62} lane_i \cdot b_{i(sorting)} \leq rack_{Total_{width}} \sum_{i=1}^{n=62} q_i \cdot \frac{b_{i(-sorting)} \cdot b_{i(sorting)}}{rack_{length}} \leq rack_{Total_{width}} \tag{1}$$

```r
#get the with of the lanes in mm
rack_width_occupied=sum(sum(lane*product_data$b_sorting))
rack_width_occupied
```

```
## [1] 1443664
```

```r
if(rack_width_occupied <= rack_total_width){
  print(past0("Capacity constraint fulfilled: ", rack_width_occupied, "<=",rack_total_width))

}else {
  violated <- rack_width_occupied - rack_total_width
  print(paste0("Capacity constraint violated by: ",violated ))
}
```

```
## [1] "Capacity constraint violated by: 1387664"
```

As seen above contraint was voilated by $1387664mm$, this means that we ordered too much, therefore we need to optimize q_i

#Prove equation 1 for single item using same figures above: Using our example to get number of the total width i.e (summation of lanes) in mm: Summation of lanes can simply be:

$$lanes_i \cdot b_{i(sorting)} = 10.593 \cdot 396 = 4194.828$$

Using equation (1) to prove this concept.

$$214 \cdot \frac{396 \cdot 297}{6000} = 4194.828 b_{isorting} = \frac{4194.828}{10.593} = 396$$

(D) Adjust Q by reducing it:

$\frac{de_i}{q_i}$ = no of orders for part i
$\frac{q_i}{2}$ = average inventory for part i

$$min \rightarrow (\sum_{i=1}^{n=62} c^{-or} + \sum_{i=1}^{n=62} \frac{de_i}{q_i} \cdot c_i^{or}) + (h \cdot \sum_{i=1}^{n=62} \frac{q_i}{2} \cdot pr_i)$$

Subject to:

$$\sum_{i=1}^{n=62} q_i \cdot \frac{b_{i(-sorting)} \cdot b_{i(sorting)}}{rack_{length}} \leq rack_{Total_{width}}$$

Calculating using the current EOQ

```r
#dei=product_data$demand_per_year,cori=product_data$ordering_cost,cord=1500,pri=product_data$box_cost,h

so_obj_funct<-function(cord,cori,dei,eoq,h,pri){

  obj <- (62*cord)+(sum((dei/eoq)*cori))+ (h*sum((eoq/2)*pri))
  return(obj)
}

so_obj_funct(1500,product_data$ordering_cost,product_data$demand_per_year,product_data$eoq,0.10,product_
```

```
## [1] 186649.1
```

Optimization with respect to capacity contraint

n= 62 Material items.

```
#library(Rglpk)
library(ROI)
```

```
## ROI: R Optimization Infrastructure
```

```
## Registered solver plugins: nlminb, alabama, glpk, quadprog, symphony.
```

```
## Default solver: auto.
```

```
# solver for non-linear, generally constrained programs
library(ROI.plugin.alabama)

n <- length(product_data$`material ID`) #number of materials
cori <- product_data$ordering_cost   #ordering cost for each items
cord <-  1500    #ordering cost whenever there is an order
dei <- product_data$demand_per_year
h<-0.10
box_cost <- product_data$box_cost #pri

# objective function --> I dropped the 1500*62 as it is not decision relevant
obj.fun <- function(q, d= dei, c.or = cori, c.h = h*box_cost  ) (sum((dei/q)*c.or)+ sum(c.h*(q/2)))
# benchmarks
obj.fun(max(product_data$eoq))
```

```
## [1] 570664
```

```
obj.fun(min(product_data$eoq))
```

```
## [1] 48918.81
```

```
# constraint function --> also contains the ceiling of lanes and a sum was missing
const.fun <- function(q, bns = product_data$b_not_sorting, bs = product_data$b_sorting, rl = rack_lengt
  sum(bs * ceiling( bns * q / rl))
  }

const.fun(max(product_data$eoq))
```

```
## [1] 6060320
```

```
const.fun(min(product_data$eoq))
```

```
## [1] 366272
```

```r
# try to figure out a freasible starting solution
const.fun(min(product_data$eoq)/10) -  rack_total_width
```

```
## [1] -3284
```

```r
qopt <- OP(
  objective = F_objective(F=obj.fun ,n=n),
  types = rep("C",n),
  bounds = V_bound(ub= rep(max(product_data$eoq),n) , lb= rep(1, n)),
  constraints = F_constraint(F=const.fun,
                             dir="<=",
                             rhs = rack_total_width)
)

# solve the problem with appropriate starting vlaue and proper solver
?ROI_solve
```

```
## starting httpd help server ...
```

```
##  done
```

```r
min(product_data$eoq) #173
```

```
## [1] 173
```

```r
rack_total_width #56000
```

```
## [1] 56000
```

```r
#This shows that minimum EOQ is too big and therefore will not meet the rack space constraint.
const.fun(min(product_data$eoq))#366272
```

```
## [1] 366272
```

```r
const.fun(min(product_data$eoq)/8)# 61428  still > 56000
```

```
## [1] 61428
```

```r
const.fun(min(product_data$eoq)/8.7)#52716 < 56000
```

```
## [1] 52716
```

```r
const.fun(round(min(product_data$eoq)/10))# 52716 < 56000
```

```
## [1] 52716
```

```r
round(min(product_data$eoq)/10) #17.3
```

```
## [1] 17
```

We will use min(product_data$eoq)/10 as our starting value because it seemed as the closest value to 56,000 which is the capacity

```r
copt_sol <- ROI_solve(qopt, start = rep(min(product_data$eoq)/10,n), solver = "alabama" )
# always check whether the algorithm converged
copt_sol# The objective value is: 1.313850e+05
```

```
## Optimal solution found.
## The objective value is: 1.313850e+05
```

```r
# solution
copt_sol$solution #vector of optimal Quantity that meets the space constraints and minimizes the Obj fu
```

```
##  [1] 19.37164 21.02634 21.24519 21.74656 21.74656 20.18584 22.13884 22.34026
##  [9] 21.35911 21.45702 20.20200 20.45919 20.44665 19.73786 22.26270 22.06554
## [17] 19.90296 19.91588 19.90694 19.56216 19.37097 19.96680 20.05206 19.92238
## [25] 19.90919 20.39379 20.26141 21.73427 21.73367 19.27732 19.30853 18.57280
## [33] 18.67564 19.44872 19.61198 21.21350 21.75563 20.64402 19.95120 19.76402
## [41] 20.46147 20.45473 20.62659 22.32568 22.32523 22.03786 22.04200 23.20436
## [49] 23.14804 20.94197 21.27054 20.31225 22.06817 21.66285 20.00371 19.81107
## [57] 20.27399 20.43553 20.69141 20.34752 20.20202 20.20202
```

```r
round(copt_sol$solution)
```

```
##  [1] 19 21 21 22 22 20 22 22 21 21 20 20 20 20 22 22 20 20 20 20 19 20 20 20 20
## [26] 20 20 22 22 19 19 19 19 19 20 21 22 21 20 20 20 20 21 22 22 22 22 23 23 21
## [51] 21 20 22 22 20 20 20 20 21 20 20 20
```

```r
copt_sol$objval #131385
```

```
## [1] 131385
```

```r
#########################################
# Now you need to fine tune the results
# --> there are rounding issues -> exactly determining the lane configuration per shelf level
# --> idea to coping with the common ordering cost
```

```r
const.fun(copt_sol$solution)#55884
```

```
## [1] 55884
```

```r
const.fun(round(copt_sol$solution))#55686
```

```
## [1] 55686
```

```r
#check obj function

obj.fun(copt_sol$solution)#131385
```

```
## [1] 131385
```

```r
obj.fun(round(copt_sol$solution))#132096.3  rounded q values
```

```
## [1] 132096.3
```

Joint Ordering (JO)

$de_i = Number of boxes demanded for item i$

$q_i^* = EOQ for box with item i$

$$q_i^* = \sqrt{\frac{2 \cdot de_i \cdot c_i^{or}}{pr_i \cdot h}}$$

$$\frac{de_i}{q_i} = Order frequency for item i$$

$k = \{\frac{de_1}{q_1}, ..., \frac{de_n}{q_n}\}$ number of unique ordering frequency

e.g there are 62 items assuming there are $k = 8$ unique ordering frequency that is item 1 and item 7 can have the same ordering frequency.

$$\sum_{i=1}^{m=k} \frac{de_i}{q_i} \cdot c^{-or} + \sum_{i=1}^{m=62} \frac{de_i}{q_i} \cdot c^{or}$$

$$min \rightarrow \sum_{i=1}^{m=62} \frac{q_i}{2} \cdot h \cdot pr_i + \sum_{i=1}^{m=k} \frac{de_i}{q_i} \cdot c^{-or} + \sum_{i=1}^{m=62} \frac{de_i}{q_i} \cdot c^{or}$$

Subject to:

$$\sum_{i=1}^{n=62} q_i \cdot \frac{b_{i(-sorting)} \cdot b_{i(sorting)}}{rack_{length}} \leq rack_{Total_{width}}$$

```r
#tinytex::install_tinytex()
tinytex:::is_tinytex()
```

```
## [1] TRUE
```