Appendix A

R code

A.1 Data preparation.

The input of this code is a text file containing all the observations and variables, untreated. To verify the accuracy of the building pairs algorithm, we used a reduced data set and computed the result by hand and verified the concordance with the model's output.

```
#library(tidyverse)
#library(purrr)
#library(dplyr)

#setwd("~/Desktop/Master thesis/MT2/")

#Import data
mydata = as_tibble(read.csv("mydata.csv"))  # read csv file

#select appropriate variables
mydata = mydata %>%
  select(DSO, dOpex.sr, yEnergy.del.lv, yEnergy.del.hv,
   ↪  yConnections.lv,yConnections.hv, yPeakload.max, zTransf,
   ↪  zCircuitStn)

#NAs removal
mydata = mydata %>% drop_na()

#negative and zero values removal
mydata = mydata %>% filter_all(all_vars( . > 0))

#outliers removal
#require(FEAR)
```

```
#x.fear <- rbind(t(x[,1,7]),t(x[,2,7]))
#y.fear <- rbind(t(y[,1:3,7]))
#tap<-ap(X=x.fear,Y=y.fear,NDEL=12)
#outliers.table = print(cbind(tap$imat,tap$r0), na.print="",
↪  digit=2)
#outlier.ap.plot(tap$ratio,ylim = c(0,4),NLEN = 25,
#    xlab = "Number of DMUs deleted", ylab = "Log-Ratio R(p)")

outliers = c(88,146)
mydata = mydata %>% filter(!DSO %in% outliers)

#descriptive statistics
#statistics = mydata  %>%
#  pivot_longer(2:9) %>%
#  group_by(name) %>%
#  summarise_at(vars(value), list(min, mean, max, sd)) %>%
#  mutate_if(is.numeric, round, 2)

  #write.table(statistics, file = "sumstats.txt", sep = "&",
  #quote = FALSE, row.names = F)

#mean normalisation
#mydata = head(mydata,40)
mydata = mydata %>% mutate_at(vars(c(2:9)),funs(./ mean(.))) %>%
↪  mutate_at(vars(c(2:9)),funs(round(.*1000,digits=0)))
```

A.2 Building algorithm

```
library("vertexenum")
library("matrixStats")
library("rcdd")
library("abind")
library("data.table")
library("ecr")
library("dplyr")
library("rje")
library("Benchmarking")
library("Rglpk")

#_________________________data selection_________________________#
df1 = data.frame(cbind(mydata$DSO,mydata$dOpex.sr))
rownames(df1) <- NULL
colnames(df1) <- c("V1","V2")

df2 =
↪  data.frame(cbind(mydata$DSO,mydata$yEnergy.del.lv,mydata$yEnergy.del.hv,mydata
rownames(df2) <- NULL
colnames(df2) <- c("V1","V2","V3","V4","V5","V6")

df1_0 = df1
df2_0 = df2
index_0 = matrix(unique(df1_0[,1]),ncol=1)

#_________________________FDH & VRS computations_________________#
```

```
vrs = dea(as.matrix(df1_0$V2), as.matrix(df2_0[,2:6]),
 ↪  RTS="vrs",ORIENTATION="in", XREF=NULL,
 ↪  YREF=NULL,FRONT.IDX=NULL, SLACK=FALSE, DUAL=FALSE, DIRECT=NULL,
 ↪  param=NULL,TRANSPOSE=FALSE, FAST=FALSE, LP=FALSE, CONTROL=NULL,
 ↪  LPK=NULL)

fdh = dea(as.matrix(df1_0$V2), as.matrix(df2_0[,2:6]), RTS="fdh",
          ORIENTATION="in", XREF=NULL, YREF=NULL,
          FRONT.IDX=NULL, SLACK=FALSE, DUAL=FALSE, DIRECT=NULL,
           ↪  param=NULL,
          TRANSPOSE=FALSE, FAST=FALSE, LP=FALSE, CONTROL=NULL,
           ↪  LPK=NULL)

#_________________________initial dominance test_________________#
index <- matrix(unique(df2[,1]),ncol=1)
trashIndex <- vector()
l =
 ↪  apply(head(index,(length(index)-1)),1,function(x){print(c("x",x));
  remaining = tail(index,length(index)-as.numeric(match(x,index)));
  xin <- x

    i = TRUE;
    n = unlist(apply(remaining,1,function(z){if(i){
      result <- dominance(df1[df1[, 1] ==
       ↪  xin,,drop=F][,-1,drop=F],df1[df1[, 1] ==
       ↪  z,,drop=F][,-1,drop=F],df2[df2[, 1] ==
       ↪  xin,,drop=F][,-1,drop=F],df2[df2[, 1] ==
       ↪  z,,drop=F][,-1,drop=F],xin);

      if(result == 2){
        i<<- FALSE
        trashIndex <- c(trashIndex,z)
      };
     }}));
  })

toRemove = matrix(unique(trashIndex))

df1 <- df1 %>% filter(!V1 %in% toRemove)
df2 <- df2 %>% filter(!V1 %in% toRemove)

#_________________________Building algorithm_____________________#

running = 1
s = 0

while(running>0){
s=s+1
print(s)

trashIndex <- vector()
index <- matrix(unique(df2[,1]),ncol=1)
newIndex <- index


```

```
l =
  apply(head(index,(length(index)-1)),1,function(x){print(c("x",x));
  remaining = tail(index,length(index)-as.numeric(match(x,index)));
  xin <<- x

  m = apply(remaining,1,function(y){ print(c("y",y));
    #I create new pair 1, input and output
    new1_x <<- input.intersection(df1[df1[, 1] ==
     xin,,drop=F][,-1,drop=F],df1[df1[, 1] ==
     y,,drop=F][,-1,drop=F]);
    new1_y <<- output.union(as.matrix(df2[df2[, 1] ==
     xin,,drop=F][,-1,drop=F]),as.matrix(df2[df2[, 1] ==
     y,,drop=F][,-1,drop=F]));

    i = TRUE;
    n = unlist(apply(newIndex,1,function(z){if(i){
      result <<- dominance(df1[df1[, 1] ==
       z,,drop=F][,-1,drop=F],new1_x,df2[df2[, 1] ==
       z,,drop=F][,-1,drop=F],new1_y,z);
      if(result == 2){i<<- FALSE};
      return(result);}}));

    if(!(2%in%n)){
      print("New pair created!")
      newID = as.numeric(tail(newIndex,1)+1)
      print(newID)
      df1[nrow(df1)+1,] <<- as.data.frame(cbind(newID,new1_x))
      df2[nrow(df2) + seq_len(nrow(new1_y)),] <<-
       as.data.frame(cbind(newID,new1_y))
      newIndex <<- matrix(unique(df1[,1]),ncol=1)
      };

    #I create new pair 1, input and output
    new2_x <<- input.union(df1[df1[, 1] ==
     xin,,drop=F][,-1,drop=F],df1[df1[, 1] ==
     y,,drop=F][,-1,drop=F]);
    new2_y <<- output.intersection(as.matrix(df2[df2[, 1] ==
     xin,,drop=F][,-1,drop=F]),as.matrix(df2[df2[, 1] ==
     y,,drop=F][,-1,drop=F]));

    i = TRUE;
    n = unlist(apply(newIndex,1,function(z){if(i){
      result <<- dominance(df1[df1[, 1] ==
       z,,drop=F][,-1,drop=F],new2_x,df2[df2[, 1] ==
       z,,drop=F][,-1,drop=F],new2_y,z);
      if(result == 2){i<<- FALSE};
      return(result);}}));

    if(!(2%in%n)){
      print("New pair created!")
      newID = as.numeric(tail(newIndex,1)+1)
      print(newID)
      df1[nrow(df1)+1,] <<- as.data.frame(cbind(newID,new2_x))
      df2[nrow(df2) + seq_len(nrow(new2_y)),] <<-
       as.data.frame(cbind(newID,new2_y))
```

```
110       newIndex <<- matrix(unique(df1[,1]),ncol=1)
111       };
112     })
113 })
114 
115 toRemove = matrix(unique(trashIndex))
116 
117 df1 <<- df1 %>% filter(!V1 %in% toRemove)
118 df2 <<- df2 %>% filter(!V1 %in% toRemove)
119 running <<- length(toRemove)
120 }
121 
122 #_____________________functions_____________________________#
123 output.union = function(df1,df2){
124   b1 = c(rep(1, nrow(df1)), rep(0,ncol(df1)))     #right hand side
    ↪  of the inequalities
125   b2 = c(rep(1, nrow(df2)), rep(0,ncol(df2)))     #right hand side
    ↪  of the inequalities
126 
127   ab =
    ↪  enumerate.vertices(rbind(rbind(-df1,diag(ncol(df1))),rbind(-df2,diag(ncol(df
128   i2 = redundant(makeV(d2q(ab)), representation ="V")
129   i3 = scdd(i2$output)
130   i4 = as.data.frame(q2d(i3$output[,-c(1,2)])) %>%
    ↪  filter_all(all_vars( . != 0)) %>%
    ↪  mutate_all(funs(round(.,digits=0))) %>% distinct()
131   colnames(i4) = c("V2","V3","V4","V5","V6")
132   i4[order(i4$V2),]
133   rownames(i4) = c()
134   return(i4)
135 }
136 
137 output.intersection = function(df1,df2){
138 ab1 = enumerate.vertices(rbind(-df1,diag(ncol(df1))),c(rep(1,
    ↪  nrow(df1)), rep(0,ncol(df1)))) #the antiblocker of output
    ↪  space 1
139 ab2 = enumerate.vertices(rbind(-df2,diag(ncol(df1))),c(rep(1,
    ↪  nrow(df2)), rep(0,ncol(df2)))) #the antiblocker of output
    ↪  space 2
140 
141 i2 = redundant(makeV(d2q(rbind(ab1,ab2))), representation ="V")
    ↪  #remove non-vertices points
142 i3 = scdd(i2$output)                                  #find
    ↪  H-representation back
143 i4 = as.data.frame(q2d(i3$output[,-c(1,2)])) %>%
    ↪  filter_all(all_vars( . != 0)) %>%
    ↪  mutate_all(funs(round(.,digits=0))) %>% distinct()
144 colnames(i4) = c("V2","V3","V4","V5","V6")
145 i4[order(i4$V2),]
146 rownames(i4) = c()
147 return(i4)
148 }
149 
150 input.union = function(df1,df2){
151   union = data.frame(min(df1,df2))
```

```
152   colnames(union) = c("V2")
153   return(union)
154 }
155
156 input.intersection = function(df1,df2){
157   intersection = data.frame(max(df1,df2))
158   colnames(intersection) = c("V2")
159   return(intersection)
160 }
161
162 dominance = function(df1x,df2x,df1y,df2y,i){
163
164 #check if identical
165   equalitx = 0
166   equality = 0
167   if(df1x==df2x){equalitx=1}
168
    ↪  if(nrow(df1y)==nrow(df2y)){if(isTRUE(all.equal(df1y[do.call(order,
    ↪  df1y),],df2y[do.call(order, df2y),], check.attributes =
    ↪  FALSE))){equality=1}}
169
170   if((equalitx&&equality)==1){return(2)}
171
172 #dominance test inputs
173 if(equalitx==0){
174   if(df1x>df2x){domix = c(1,0)}else{if(df1x<df2x){domix=c(0,1)}}
175 }
176
177 #dominance test outputs
178 if(equality==0){
179 df1y = as.matrix(df1y,ncol = ncol(df1y))
180 df2y = as.matrix(df2y,ncol = ncol(df2y))
181 r1 = nrow(df1y)
182 r2 = nrow(df2y)
183 c  = ncol(df1y)
184
185 polytope1 = d2q(rbind(df1y,diag(colMaxs(df1y))))
186 polytope2 = d2q(rbind(df2y,diag(colMaxs(df2y))))
187
188 vrep = q2d(makeV(rbind(polytope1,polytope2,0)))
189 domi = matrix(redundant(vrep)$new.position)
190 domiy1 = domi[1:(r1+c),]
191 domiy2 = domi[(r1+c+1):((r1+c+r2+c)),]
192
193 if(0 %in% domiy1){domiy1 = 1}else{domiy1=0}
194 if(0 %in% domiy2){domiy2 = 1}else{domiy2=0}
195
196 if((domiy1&domiy2)==1){domiy=c(0,0)}else{domiy=c(domiy1,domiy2)}
197 }
198
199 #total dominance
200 if(equalitx==1){domix=domiy}
201 if(equality==1){domiy=domix}
202 domiAll = rbind(domix,domiy)
203
204 if(sum(domiAll[,1])==2){#the new pair dominates the other
```

```
205   trashIndex <<- c(trashIndex,i)
206   return(1)
207 }
208 if(sum(domiAll[,2])==2){#the new pair is dominated by the other
209   return(2)
210 }
211
212 if((sum(domiAll[,1])&&sum(domiAll[,2]))<=1){#no domination
213   return(1)
214 }
215 }
```

A.3 Computation of the productivity index.

```
1  #dimension of the constraints matrix
2  columns = 2*nrow(newIndex)
3
4  #productivity index vector
5  productivity = cbind(index_0,0)
6
7  t = 1
8  r = apply(index_0, 1,function(x){
9    p <<- 1
10   constraints_matrix <<- as.data.frame(matrix(1,ncol= columns))
11
12   x0 <<- as.matrix(df1_0[df1_0[, 1] == x,,drop=F][,-1,drop=F])
13   y0 <<- as.matrix(df2_0[df2_0[, 1] == x,,drop=F][,-1,drop=F])
14
15   s = apply(newIndex, 1, function(y){
16     Bi = as.matrix(1/df1[df1[, 1] == y,,drop=F][,-1,drop=F])
17     Ai = as.matrix((1/df2[df2[, 1] == y,,drop=F][,-1,drop=F])/5)
18     rowAi = nrow(Ai)
19     colAi = ncol(Ai)
20     loli = as.matrix(max(apply((-sweep(Ai,2,y0,`*`)),1,sum)+1))
21
22     x_line  = matrix(c(rep(0,(p -
       ↪  1)),x0*Bi,-1,c(rep(0,as.numeric(columns - p - 1)))), ncol =
       ↪  columns)
23     y_lines =
       ↪  matrix(cbind(matrix(0,nrow=nrow(loli),ncol=p),loli,matrix(0,nrow=nrow(loli
       ↪  - p - 1))), ncol = columns)
24     #y_lines =
       ↪  matrix(cbind(matrix(0,ncol=p,nrow=rowAi),matrix(-max(sweep(Ai,2,y0,`*`))+
25
26     constraints_matrix[nrow(constraints_matrix) +
       ↪  seq_len(nrow(rbind(x_line,y_lines))),] <-
       ↪  as.data.frame(rbind(x_line,y_lines))
27
28     p <<- p+2
29   })
30
31   obj = c(rep(c(1,0),nrow(newIndex)))
32   mat = as.matrix(constraints_matrix[-1,])
```

```
  mat = rbind(mat,rep(c(0,1),nrow(newIndex)))
  #print(mat)
  dir = c(rep(c(">="),nrow(mat)-1))
  dir = c(dir,"==")
  rhs = c(rep(0,nrow(mat)-1))
  rhs = c(rhs,1)
  max = FALSE
  #types = c(rep(c("C","C"),nrow(newIndex)))

  opti = Rglpk_solve_LP(obj, mat, dir, rhs, max = max)

  productivity[t,2] <<- opti$optimum
  t <<- t+1
})
```

A.4 Plots and results table.

```
library("ggplot")
all=cbind(productivity[,1],matrix(vrs$eff,nrow=40,ncol=1),productivity[,2],matrix(

eff.all = all[,2:4]
eff.all = eff.all[order(eff.all[,3]),]

p0 = matplot(eff.all, xlab = "DMUs", ylab = "Efficiency")

#histogram CP
p1 <- ggplot(as.data.frame(productivity[,2]),
 ↪  aes(x=productivity[,2])) +
  geom_histogram(color="white", fill="grey")+
  xlab("Convex Pairs approach efficiency distribution") +
   ↪  ylab("Count")+
  theme_bw()  #histogram

#histogram fdh
p2 <- ggplot(as.data.frame(fdh$eff), aes(x=fdh$eff)) +
  geom_histogram(color="white", fill="grey")+
  xlab("FDH efficiency distribution") + ylab("Count")+
  theme_bw()  #histogram

#histogram VRS
p3 <- ggplot(as.data.frame(vrs$eff), aes(x=vrs$eff)) +
  geom_histogram(color="white", fill="grey")+
  xlab("VRS efficiency distribution") + ylab("Count")+
  theme_bw()  #histogram

#efficiencies all
p4 = ggplot(data.frame(eff.all), aes(c(1:40),eff.all, colour =
 ↪  c("red","orange","green"))) +
  geom_line(data = data.frame(productivity[,2]), aes(x = c(1:40), y
   ↪  = productivity[,2]), color = "orange") +
  geom_point(data = data.frame(productivity[,2]), aes(x = c(1:40),
   ↪  y = productivity[,2]), color = "orange") +
```

```
31    geom_line(data = data.frame(vrs$eff), aes(x = c(1:40), y =
      ↪  vrs$eff), color = "red") +
32    geom_point(data = data.frame(vrs$eff), aes(x = c(1:40), y =
      ↪  vrs$eff), color = "red") +
33    geom_line(data = data.frame(fdh$eff), aes(x = c(1:40), y =
      ↪  fdh$eff), color = "green") +
34    geom_point(data = data.frame(fdh$eff), aes(x = c(1:40), y =
      ↪  fdh$eff), color = "green") +
35    geom_hline(yintercept=median(eff.all[,1]), color = "red") +
36    geom_hline(yintercept=median(eff.all[,2]), color = "orange") +
37    geom_hline(yintercept=median(eff.all[,3]), color = "green") +
38    theme_bw() + xlab("DMUs") + ylab("Efficiency")
39
40
41  diff_VRS_CP = productivity[,2]-vrs$eff
42
43  #comparison input size VS VRS-CP difference
44  p5 = ggplot() +
45    geom_col(data = data.frame(diff_VRS_CP), aes(x = c(1:40), y =
      ↪  diff_VRS_CP*(max(df1_0[,2])/max(diff_VRS_CP))))+
46    geom_line(data = data.frame(df1_0[,2]), aes(x = c(1:40), y =
      ↪  df1_0[,2]-mean(df1_0[,2])), color = "red") +
47    xlab('DMUs') +
48    ylab('Inputs - mean(inputs)')+
49    scale_y_continuous(sec.axis = sec_axis(~./max(df1_0),name = "CP -
      ↪  VRS efficiency"))+theme_bw()
50
51  #comparison output size VS VRS-CP difference
52  p6 = ggplot() +
53    geom_col(data = data.frame(diff_VRS_CP), aes(x = c(1:40), y =
      ↪  diff_VRS_CP*(max(rowSums(df2_0[,-1]))/max(diff_VRS_CP))))+
54    geom_line(data = data.frame(rowSums(df2_0[,-1])), aes(x =
      ↪  c(1:40), y = rowSums(df2_0[,-1])-mean(rowSums(df2_0[,-1]))),
      ↪  color = "red") +
55    xlab('DMUs') +
56    ylab('Outputs - mean(outputs)')+
57    scale_y_continuous(sec.axis = sec_axis(~./max(df1_0),name = "CP -
      ↪  VRS efficiency"))+theme_bw()
58
59  #table efficiencies
60  eff_results =
    ↪  cbind(c(1:20),round(data.frame(fdh$eff)[1:20,1],4),round(productivity[1:20,2],
61  write.table(format(eff_results, nsmall = 2), "eff_results.txt",
    ↪  quote=FALSE, eol="\\\\\n", sep=" & ")
```

Appendix B

Additional tables

B.1 FDH, CP and VRS efficiencies, n = 40.

DSO	FDH	CP	VRS	DSO	FDH	CP	VRS
1	0.5650	0.4829	0.4995	21	1.0000	1.0000	1.0000
2	1.0000	0.6897	0.7096	22	1.0000	1.0000	1.0000
3	1.0000	1.0000	0.9781	23	0.6126	0.5801	0.4838
4	0.8947	0.8947	0.7117	24	0.7267	0.3825	0.4247
5	1.0000	1.0000	1.0000	25	1.0000	0.8882	0.8849
6	1.0000	1.0000	1.0000	26	0.8349	0.5797	0.5530
7	0.9900	0.5731	0.7132	27	1.0000	0.5482	0.8949
8	0.8293	0.7088	0.7544	28	1.0000	1.0000	1.0000
9	0.5904	0.5046	0.5507	29	1.0000	1.0000	1.0000
10	0.5039	0.5039	0.4835	30	1.0000	1.0000	1.0000
11	0.5827	0.4980	0.5790	31	1.0000	0.4439	0.4098
12	1.0000	1.0000	0.9613	32	0.5961	0.5961	0.4162
13	0.8934	0.8934	0.7950	33	1.0000	1.0000	1.0000
14	0.7717	0.6595	0.7489	34	0.6157	0.5475	0.4645
15	1.0000	1.0000	0.7800	35	1.0000	0.9700	0.8152
16	1.0000	0.9623	1.0000	36	0.9710	0.6384	0.6273
17	0.8263	0.7348	0.7394	37	1.0000	1.0000	1.0000
18	0.7731	0.7731	0.5599	38	1.0000	0.5587	1.0000
19	0.8693	0.8693	0.7009	39	1.0000	0.8282	1.0000
20	1.0000	0.6863	1.0000	40	1.0000	1.0000	0.9956

Appendix C

Glossary

This short list provides information for the good understanding of the concepts developed in this thesis.

Convex polyhedron: in a convex polyhedron, the line segment joining any two vertices of the polyhedron lies entirely in the interior of the polyhedron. A convex polyhedron has no holes or indentations (Hui, 2001).

Convex union: "simply formed by merging extreme points, and intersection is formed by taking maximum and minimum coordinates, respectively, for input and output polyhedra" (Agrell et al., 2005a).

Determinant: in linear and multilinear algebra, a value, denoted det A, associated with a square matrix A of n rows and n columns. Designating any element of the matrix by the symbol a_{rc} (the subscript r identifies the row and c the column), the determinant is evaluated by finding the sum of n! terms, each of which is the product of the coefficient $(1)r + c$ and n elements, no two from the same row or column. Determinants are of use in ascertaining whether a system of n equations in n unknowns has a solution. If B is an nÖ1 vector and the determinant of A is nonzero, the system of equations $AX = B$ always has a solution" (Encyclopædia Britannica, n.d.-a).

Deterministic model: a deterministic model is one in which the values for the dependent variables of the system are completely determined by the parameters of the model. In contrast, stochastic, or probabilistic, models introduce randomness in such a way that the outcomes of the model can be viewed as probability distributions rather than unique values (Encyclopædia Britannica, n.d.-a).

Extreme points or vertex: "a vector x of a polyhedron P is called an extreme point if there are no two vectors $y \neq x$ and $z \neq x$ in P such that x is a convex combination of y and z. This means an extreme point is a vector which does not lie on the line connecting any two vectors in P. Note that we did not use the inequality description of P in this definition.
Definition 2. A vector x of a polyhedron P is called a vertex if there exists c such that $c'x < c'y$ for all $y \neq x$" (Hosten, 2003).

Monotonicity: "a monotonic function is a function which is either entirely nonincreasing or nondecreasing. A function is monotonic if its first derivative (which need not be continuous) does not change sign" (Royden & Fitzpatrick, 2018).

Polyhedra: polyhedron, in Euclidean geometry, are a three-dimensional object composed of a finite number of polygonal surfaces (faces). Technically, a polyhedron is the boundary between the interior and exterior of a solid. In general, polyhedrons are named according to number of faces. A tetrahedron has four faces, a pentahedron five, and so on; a cube is a six-sided regular polyhedron (hexahedron) whose faces are squares. The faces meet at line segments called edges, which meet at points called vertices" (Encyclopædia Britannica, n.d.-b).

Parametric model: a parametric model is a family of probability distributions that has a finite number of parameters (Penenberg, 2016).

Non-parametric models: differ from parametric models in that the model structure is not specified a priori but is instead determined from data. The term

non-parametric is not meant to imply that such models completely lack parameters but that the number and nature of the parameters are flexible and not fixed in advance (Richardson, 2015).

Recession cone: "let C be a non-empty convex set. The recession cone $O + C$ is then a convex cone containing the origin. It is the same as the set of vectors y such that C + y $\subseteq$ in C" (Rockafellar, 1970).