Measuring Spatio-Temporal Efficiency: An R Implementation for Time-Evolving Units

Georgios Digkas[1], Konstantinos Petridis[1,*], Alexander Chatzigeorgiou[1], Emmanouil Stiakakis[1], Ali Emrouznejad[2]

[1]Department of Applied Informatics, University of Macedonia, 54006 Thessaloniki, Greece

[2]Aston Business School, Aston University, Birmingham B4 7ET, United Kingdom

Email: mai153@uom.edu.gr, k.petridis@aston.ac.uk, achat@uom.gr, stiakakis@uom.gr, a.emrouznejad@aston.ac.uk

Abstract

Classical Data Envelopment Analysis (DEA) models have been applied to extract efficiency when time series data are used. However, these models do not always yield realistic results, especially when the purpose of the study is to identify the peers of the Decision Making Unit (DMU) under investigation. This is due to the fact that apart from the spatial distance of DMUs, which is the basic on which efficiency is extracted, the distance in time between DMUs is also important in identifying the most suitable peer that could serve as a benchmark for the DMU under investigation. Based on these two dimensions, i.e. the spatial and the temporal, the concept of Spatio-Temporal efficiency is introduced and a Mixed Integer Linear Programming (MILP) model is proposed to obtain its value. This model yields a unique past peer for benchmarking purposes based on both dimensions. The implementation has been performed in the R language, where the user can provide, through a graphical interface, the data (inputs and outputs for successive versions of a DMU) for which the Spatio-Temporal efficiency is measured. Applications to the real world and particularly from the discipline of software engineering are provided to show the applicability of the model to temporally arranged data. Profiling results of the code in the R language are also provided showing the effectiveness of the implementation.

Keywords: DEA, LP, MILP, R platform, Spatio-Temporal efficiency, Computational economics

[*]Corresponding author, e-mail: k.petridis@uom.edu.gr, Tel: +30 2310 891 728

1. Introduction

Evolution occurs in all business, economic, and technological systems. They evolve as their constituent parts, such as means of production, market mechanisms, processes etc., change over time. Computer software can also be regarded as an evolving system since it gradually undergoes maintenance in order to correct errors and accommodate changing and new requirements. This evolution is clearly evident in software systems from the multitude of releases that they offer over time. In order to study the evolution of a software (or any other evolving) system which comprises a series of successive versions, the current version of the system should be compared with temporally previous versions. What would be useful in this case is to identify a previous, but near in time, version with similar functionality which could be regarded as more efficient than the current version, in terms of its ability to maximize output (e.g. certain software metrics) for a given input (e.g. software functionality). In such a case and in the context of software, the previous version could serve as a benchmark for the current version, meaning that its characteristics could be used as a guide to examine more carefully the characteristics of the current software version. A similar reasoning applies to the temporal analysis of system evolution in other domains, such as economies or businesses.

One of the most widely known methodologies in order to measure efficiency is Data Envelopment Analysis (DEA). In DEA, the entity under study is called a Decision Making Unit (DMU). For each DMU, a virtual input and a virtual output are formed (input and output items respectively, multiplied by weights) and then the weights are determined, using linear programming, so as to maximize the ratio of the virtual output over the virtual input. What is actually measured in DEA is the relative efficiency of a DMU against the other DMUs. The efficient units, i.e. the units with the maximum efficiency, form a surface named as “efficient frontier” which envelops all the inefficient units. The efficient DMUs can be used as benchmarks for the inefficient ones in order to improve their efficiency.

As already mentioned, the right choice, when time series data are examined, is to compare the entity under study with preceding entities. In that way, the efficiency that should be measured is not only determined according to the spatial distance of DMUs from the efficient frontier (as usually occurs), but also according to their time distance (how close the DMUs are in terms of time). Thus, efficiency is considered with regard to both the spatial and temporal dimensions. In the context of this study, this is called Spatio-Temporal efficiency. The new concept of Spatio-Temporal efficiency is graphically depicted in Figure 1. Each DMU is arranged temporally and spatially, as well. In this hypothetical case, three DMUs are

considered; DMU(6) which is DMU under investigation and its peers, DMU(2) and DMU(3). The horizontal axis represents the closeness ofa unit to other units (their spatial distance), while the vertical axis represents the temporal difference of a unit from the other units. In this example, λ_2 equals to 0.7 while λ_3 equals to 0.3. However, DMU(2) is temporally more distant than DMU(3) (the temporal distance is 4) and DMU(3) which is temporally closer to DMU(6) (with temporal distance equal to 3). As aim of the proposed approach is to find the DMU which is closer to the DMU under investigation, values in the horizontal axis are presented in a decreasing order; the largest the lambda value of a peer, the more resemblance it bears with the inputs/outputs of the DMU under study. Based on this example, there is not a rule of thumb to assist in making the decision on which unit, 2 or 3, should be selected as a single peer because the DMU under investigation should be compared with a temporally and spatially closer DMU. The previous illustrative example indicates the need for a unique peer selection in terms of both the spatial and temporal dimensions. The resulting efficiency is the Spatio-Temporal efficiency, which is analyzed more extensively in the next sections of the paper. This type of efficiency cannot be addressed by conventional DEA methods or other DEA techniques that deal with time series data.

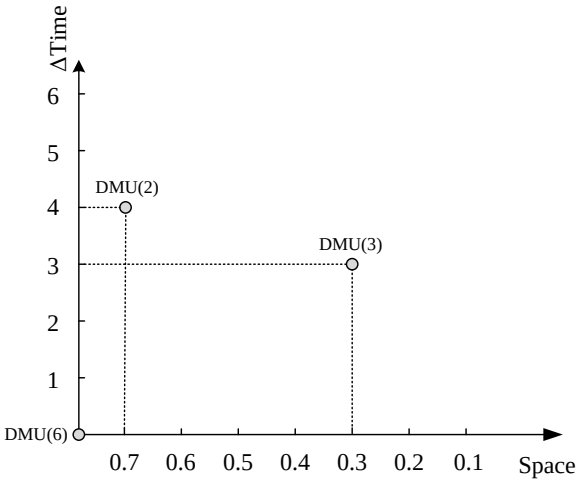

Figure 1: Units arranged over space and time

The rest of the paper is organized as follows: in Section 2, a literature review with relevant works on DEA is presented, also identifying the gap in the literature. In Section 3, the R implementation is analytically described, demonstrating all the stages of the proposed methodology. The interface and the functional characteristics of the proposed R package are demonstrated in Section 4, while a performance analysis is presented, with an application in a

real-life example from the discipline of software engineering, in Section 5. Finally, conclusions are drawn in Section 6.

2. Literature review

A Spatio-Temporal DEA model has not yet been addressed in the literature. The nearest concept to Spatio-Temporal DEA is that of dynamic and two-stage DEA models. Dynamic models take into account time providing information for efficiency changes over time. The first dynamic DEA model has been proposed by Färe and Grosskopf [1]. In the dynamic context, Malmquist index is considered in order to measure efficiency changes over time [2]; applications to healthcare[3][4]and electricity generation companies [5] have been demonstrated. Dynamic DEA models have also been used in software maintenance [6]. Moreover, the dynamic and two-stage DEA models have been used in various contexts, as for instance to assess jet fighters based on their capabilities [7]. Later work extended Technological Forecasting Data Envelopment Analysis (TFDEA) model in an application for the assessment of jet fighters and commercial airplane development.

Recently, due to the openness and popularity of the R programming language a number of mathematical and statistical approaches are offered in the form of R packages. Essentially R packages are collections of functions, data, and compiled code that extends the capabilities of the R language; they are accompanied with a documentation explaining the utilization of the functions. Those packages are developed by individual programmers or communities. One of the subjects those packages are often dealing with is modeling Operation Research (OR) techniques. There is a wide variety of packages for solving optimization problems [8]. Due to its ability, R language can easily install and built upon external packages. A wide variety of mathematical programming models have been implemented in R platform through different packages.

In his work, Konis [9] created the lpSolveAPI package for R; this package supports a Mixed Integer Linear Programming (MILP) solver with the support of other OR models. Theuss [10]created the R Optimization Infrastructure (ROI) package; ROI is a package that integrates different Mixed Integer Programming(MIP) solvers into a single interface. Cantyand Ripley [11] created the boot R package for bootstrapping techniques. In his work, Buttrey created the lp Solve[12] package for R. This package contains functions for solving linear, integer, and mixed integer problems. Henningsen [13] proposed the linprog package for R; the package is built on the simplex algorithm to solve Linear Programming or Linear Optimization problems. Turlach [14] proposed the quadprog package, which contains functions for solving Quadratic

Programming Problems. Gilbert[15]created the BB package, which contains functions that implement the Barzilai-Borwein spectral methods for solving nonlinear problems. Rudy[16] created the CLSOCP package that contains functions for solving Second Order Cone Problems (SOCP). Karatzoglou [17] created the package kernlab; it contains methods for solving Quadratic Programming problems using Interior Point Methods (IPM). Soetaert [18] proposed the package limSolve, which contains functions for solving Linear and Quadratic programming problems. Geyer [19] created the package rcdd; this is an R interface that uses the GNU Multiple Precision (GMP) library to solve linear programming problems. Hayfield and Racine (2008) [20] created the R package called np for estimating non linear production function.

An overview of R packages that specifically aim at addressing the problem related to the application of DEA are also presented. In their work, Bogetoft and Otto [21] presented the Benchmarking package. Benchmarking provides methods for frontier analysis and contains functions for different DEA technologies, such as fdh, vrs, drs, crs, irs, add/frh, and fdh+. Another package that provides functions for frontier efficiency analysis in R is the FEAR package by Wilson [22] [23]. Dong-hyun [24] proposed an R package called nonparaeff which contains nonparametric DEA functions for measuring efficiency and productivity of DMUs. Jaak [25] in his work, created the rDEA package that estimates DEA scores with or without environmental variables and performs Returns To Scale (RTS) tests. Finally, Shott [26] has proposed TFDEA package, an R package that has implemented the TFDEA algorithm which provides functions for technology forecasting. An overview of R packages which can be used as optimization solvers is presented in Table 1, contrasting each packages with the supported mathematical programming models, while packages supporting DEA technologies are summarized in Table 2.

Table 1: R packages used as optimization solvers

Mathematical Programming models / Solvers	Non-Linear Programming (NLP)	Linear Programming (LP)	Mixed Integer Linear Programming (MILP)	Semi – Continuous (SC)	Quadratic Programming (QP)	Special Ordered Sets (SOS)
lpSolveAPI[9]		✓	✓	✓		✓
ROI[10]	✓	✓	✓		✓	
boot[11]		✓				
lpSolve[12]		✓	✓			

linprog[13]		✓				
quadprog[14]					✓	
BB[15]	✓					
CLSOCP[16]						
kernlab[17]					✓	
limSolve[18]		✓			✓	
rcdd[19]		✓				

Table 2: R packages supporting DEA technologies

	DEA technologies							
R Package	FDH	VRS	DRS	CRS	IRS	Add/FRH	FDH+	SFA
Benchmarking[21]	✓	✓	✓	✓	✓	✓	✓	✓
FEAR [22] [23]	✓						✓	
nonparaeff [24]	✓	✓	✓	✓	✓			
rDEA[25]		✓		✓				
TFDEA[26]		✓	✓	✓	✓			

Classical DEA models cannot handle by default time series data and extract efficiency. This is attributed to the fact that when using time series data, the construction of the reference set of a DMU may contain temporally subsequent DMUs. Also, even in most of the dynamic DEA models, a sequential modification is implemented to take the aforementioned shortfall in the modeling into account; the analyses presented are extracting the efficiency based only on the temporal dimension. In this paper, a DEA model is presented considering two aspects based on the time series data that are to be handled; the first aspect is the spatial and the second is the temporal. The formulation presented in this paper allows firstly constructing the reference set of each DMU based on an iterative DEA model [28]. The information, taken from this initial step, is then analyzed on a second stage and the Spatio-Temporal efficiency is calculated. The implementation inthe R language is shown throughout the paper, while the R package that models the Spatio-Temporal DEA (ST-DEA) approach is available and can be downloaded from: http://se.uom.gr/index.php/projects/stdea/.To our knowledge, such an application has not yet been proposed in the literature.

3. Spatio-Temporal DEA and its R implementation

In this section, the proposed model along with its implementation in the R language [27] is

presented. The R language is a GNU project for statistical computing and graphics[1].The proposed model consists of two stages; the first stage aims at solving the conventional DEA iteratively providing a solution using a Linear Programming (LP) model. The second stage filters the solution from the first stage through a MILP model in order to derive the Spatio-Temporal frontier. The mathematical formulation and parts of the R-coding are shown in the next sub-sections.

3.1 Iterative LP model

Conventional DEA models fail to capture the dimension of time in their formulation. When time series data are fed in the model as inputs and outputs, a modified DEA approach is needed. Assuming we have t Decision Making Units (DMUs) temporally arranged such as: $DMU(1), DMU(2), \ldots, DMU(t-1), DMU(t)$. The following LP model guarantees that in the reference set of a DMU (for instance $DMU(8)$), a temporally subsequent DMU like $DMU(9)$ will not appear in its reference set. Due to the fact that DEA loses its discrimination power when the inputs and the outputs are less than $\mu = \max\left\{n \cdot m,\ 3 \cdot (n+m)\right\}$, where n is the number of inputs and m the number of outputs, a minimum number of DMUs (μ)is required in order for the technique to work. In the following mathematical model, φ represents the efficiency to be calculated and $|I|$ the number of DMUs.

Stage 1

For $\tau = \mu, ..., |I|$

$$\text{Max} \quad \varphi \tag{1}$$

s.t.

$$\sum_{i \le \tau} \lambda_i \cdot x_{ij} - x_{io} \le 0, \forall j \tag{2}$$

$$\sum_{i \le \tau} y_{ri} \cdot \lambda_i - y_{r\tau} \cdot \varphi \ge 0, \ \forall r \tag{3}$$

$$\sum_{i \le \tau} \lambda_i = 1 \tag{4}$$

$$\lambda_i \ge 0, \ i \le \tau$$
$$\varphi \ \textit{free} \tag{5}$$

End For

[1] http://www.r-project.org/about.html

Mathematical formulation (1) – (5) is the iterative LP model and presents an output-oriented DEA model, with Variable Returns to Scale (VRS) as derived by constraint (4). In this formulation, there are $j = 1,...,|I|$ DMUs, x_{ij} inputs and y_{rj} outputs. The target is to maximize ϕ as shown in objective function (1). As it can be seen in the mathematical formulation (1) – (5), the summation is performed with respect to i such that $i \le \tau$ whereas τ is an index which iteratively takes values from μ up to $|I|$. With this procedure, it can be ensured that only temporally previous peers will appear in the reference set of a DMU. Assuming that there are 10 DMUs, and $n = 1$ input and $m = 2$ outputs then the minimum point from which the iterative LP model will begin is $\mu = \max\{1 \cdot 2,\ 3 \cdot (1+2)\} = 9$. Thus, the iterative LP model is solved only for DMUs 9 and 10.

The iterative LP model starts the summation from the DMU that equals to the variable `minimumDMUs` which corresponds to μ. The variable `imported.data` is used to store the imported data, which R parses from any type of plain text files or spreadsheets; the most common format for data import is a .csv file, although different types of source can be used (.txt, .xls, .ods, etc.). Variables x and y are vectors representing inputs and outputs respectively. Variables `numberOfInputs` and `numberOfOutputs` correspond to the number of inputs and outputs which are defined as shown in Figure 2.

```
imported.data <- read.csv(...)
x <- with(imported.data, cbind(Inputs))
y <- with(imported.data, cbind(Outputs))
numberOfInputs <- ncol(x)
numberOfOutputs <-ncol(y)
```

Figure 2: The R commands that separate the imported data into Inputs and Outputs

Based on the code fragment shown in Figure 3, the iterative LP model is solved; the `while` loop is initiated with the value that is assigned to `stopDMU`, and in the first iteration is equal to the value of the variable `minimumDMUs`, whereas in the next iterations it serves as a counter. In order to provide a correspondence between the fragments of code presented and the example presented in Table 3, the initial value of variable `stopDMU`is 9 (based on the formula of minimum DMUs for 1 input and 2 outputs). Variables `TempX` and `TempY` are sub-vectors of X and Y and contains the rows 1 to stopDMU of those vectors. After this, the variables `TempX` and `TempY` are assigned to DEA model as Input and Output correspondingly. Finally, `stopDMU` variable is increased by one, and sub-vectors X, Y are calculated again.

```
while(stopDMU <= numberOfDMUs){
    tempX <- x[1:stopDMU,]
    tempY <- y[1:stopDMU,]
    e <- dea(tempX, tempY, RTS="vrs", ORIENTATION = orient)
    phiDEA[stopDMU - minimumDMUs +1, 1]<- stopDMU
    phiDEA[stopDMU - minimumDMUs +1, 2]<- eff(e)[stopDMU]
    l <- lambda(e)

    columnNames <-colnames(l)
    dataFrameLambdas <- data.frame(l)
```

Figure 3: While loop that calculate the lambdas values for each DMU

Inputs and outputs that correspond to `stopDMU` are assigned to variables `tempX` and `tempY` correspondingly through the function `dea(tempX, tempY, RTS="...", ORIENTATION =`

"..."). The results of that iterative LP DEA model are stored in variable `e`. The model can be solved for each orientation and each Returns To Scale (RTS) technology. In this example, an output-oriented DEA model with Variable Returns to Scale (VRS) technology is shown for illustrative reasons.

The procedure is graphically illustrated in Figure 4. As the model is solved using `lpSolveAPI`, the resulting LP model must be introduced in a data frame form, stating the coefficients at the right hand side of the constraints. In the first column of the data frame, variable ϕ is stored, whereas the reference set of each DMU is assigned to variable `l` as seen in Figure 4, starting from `stopDMU`. The model is solved until the last DMU is reached. With this procedure, only temporally past DMUs can appear as peers (lambdas) in the reference set as depicted in Figure 4; the grey area indicates that DMU(*t*) is not taken into account in the calculations. Figure 4demonstrates the example in Table 3 (1 input and 2 outputs), thus `stopDMU` is defined as DMU(9) which has as reference set any combination from $\lambda_1,\ldots\lambda_9$ while DMU(10) has as reference set any combination of $\lambda_1,\ldots\lambda_{10}$.

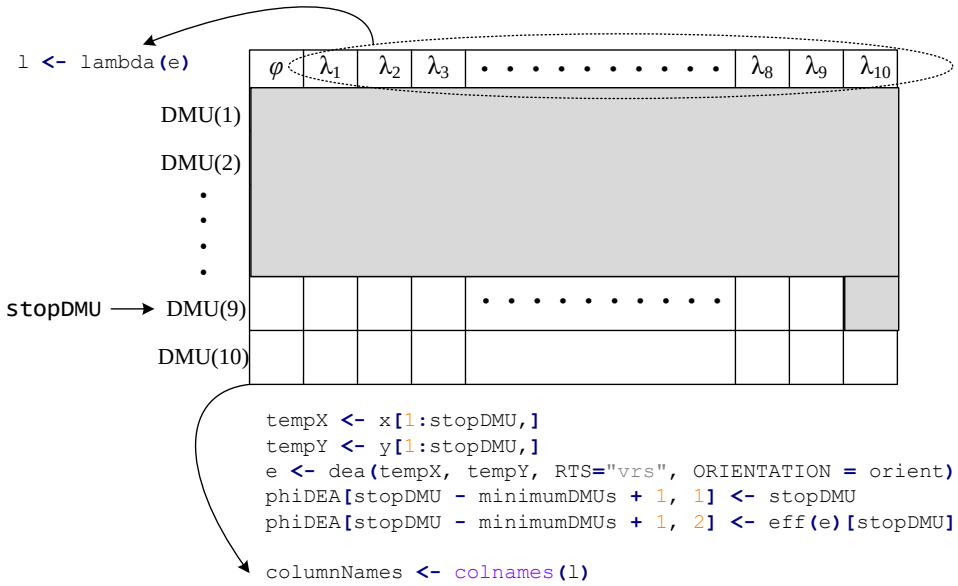

Figure 4: Graphical representation of the iterative LP models solved for DMU(9) and DMU(10).

```
#Alpha matrix

for(i in1:length(columnNames)){
     alphaMatrix[stopDMU, as.integer(substr(columnNames[i], 2,
nchar(columnNames[i])))]<-
 dataFrameLambdas[nrow(dataFrameLambdas), i]
}

    stopDMU <- stopDMU +1
}

#Delta matrix
    for(i in minimumDMUs:numberOfDMUs){
for(j in1:numberOfDMUs){
if(alphaMatrix[i,j]>0){
           deltaMatrix[i,j]<-(i - j)
}
       }
}
```

Figure 5: The R code that defines the A and Δ matrices

In order to examine the Spatio-Temporal dimension of each DMU and to construct the reference set taking into account both dimensions, two matrices are constructed. In the first matrix, as indicated by the block of R code shown inFigure5, the lambda values of each DMU are stored in the ***Λ***matrix, whereas the distances in the temporal dimension is captured by ***Δ*** matrix. Both matrices are explained next.

Matrix ***Λ*** reflects the spatial dimension as the lambda value (peer) of each DMU demonstrates the level of similarity with the DMU under investigation. For instance, if DMU(10) has DMU(5) and DMU(8) as peers with $\lambda_5 = 0.543$ and $\lambda_8 = 0.457$ respectively, then it can be said that DMU(10) exhibits a greater spatial similarity in terms of inputs/outputs with DMU(5) rather than DMU(8). However, DMU are also temporally arranged, DMU(5) is further away in terms of time than DMU(8) which is closer to DMU(10). More specifically, DMU(5) has a temporal distance of 5 time units to the DMU of interest, while DMU(8) has a distance of only 2 time units. The temporal distance of each DMU is stored in the***Δ***matrix.

Figure 6shows the ***Λ*** and ***Δ*** matrices with dimensions $|I| \times |I|$. As it can be observed, for each DMU that has temporally precedent peers the non-zero lambda values at each row (DMU) of the***Λ***matrix correspond to the temporal differences in ***Δ*** matrix. The peers that do not appear in a DMU's reference set (zero elements in ***Λ*** matrix) correspond to a very large number (`M<-1E5`) in ***Δ*** matrix in order to exclude selection.

Vectors $\left(\lambda_\tau^{MAX}\right)^{\mathrm{T}}$ and $\left(\delta_\tau^{MIN}\right)^{\mathrm{T}}$ with dimensions $|I| \times 1$, select the maximum lambda and minimum temporal distance correspondingly and are used for normalization purposes of the

next stage objective function. Figure 7 shows the procedure and the corresponding R code, by which vectors $\left(\lambda_{\tau}^{MAX}\right)^{\mathrm{T}}$ and $\left(\delta_{\tau}^{MIN}\right)^{\mathrm{T}}$ are created.

Rows 1,...,`minimumDMUs` (`minimumDMUs` = μ) of matrix ***A*** are empty; from row `minimumDMUs` + 1,...,`numberOfDMUs`, non-zero columns represent the optimal lambda values(peers) of DMU under investigation. For example, assuming that DMU 6 peers are 2 and 3 with corresponding lambda values 0.3 and 0.7 correspondingly; *A* matrix would have in positions ***A***[6,2] = 0.3 and***Δ***[6,3] = 0.7. Rows 1,...,`minimumDMUs` (`minimumDMUs` = μ), of matrixΔ (similar to A matrix) are empty; rows `minimumDMUs` + 1,...,`numberOfDMUs` contain either a very big positive number(`M`), or the temporal distance of DMU under study with its peers. Following the previous example, the values of cells ***Δ***[6,2]= 6 – 2 = and ***Δ***[6,3] = 6 – 3= 3, representing the temporal distance of DMU 6 with its peers (2 and 3). The rest of cells of DMU 6 of matrix ***Δ*** have `M` value (***Δ***[6,1]=***Δ***[6,4]=…=***Δ***[6,n]=M).

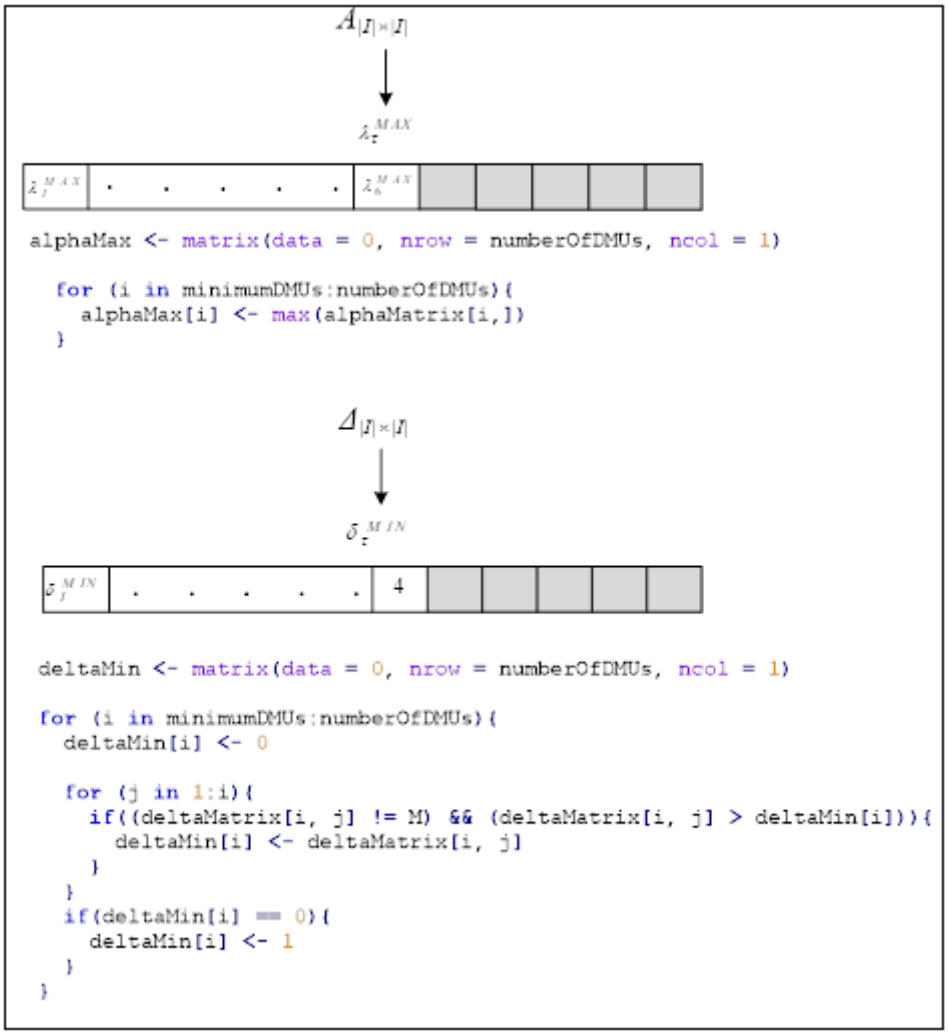

Figure 6: Construction of vectors λ_{τ}^{MAX} and δ_{τ}^{MIN}

From the ***A*** matrix, the maximum of each row (DMU) is selected while from the ***Δ*** matrix the maximum non-*M* value is selected. This is formally represented in relations (6) and (7). In (7), function *ORD* returns the order of the current DMU under investigation minus the order of its peers, as shown in Figures 3 and 4.

$$\lambda_{\tau}^{MAX} = \max_{l}\left\{a_l^{\tau}\right\},\ \forall \tau \tag{6}$$

$$\delta_l^{\tau} = \begin{cases} ORD(\tau) - ORD(l),\ a_l^{\tau} \neq 0 \\ M, a_l^{\tau} = 0 \end{cases} \tag{7}$$

As the aim of the proposed mathematical formulation is to examine the Spatio-Temporal efficiency of the temporally arranged DMUs, ***A*** and ***Δ*** matrices are introduced into the following Multi-Objective Mixed Integer Linear Programming (MO-MILP) model since the two objectives should be taken into consideration at the same time:

Stage 2

For $\tau = \mu,...,|I|$

For $j = 1,...J$

$$\max \mathrm{w}_{sp}^{j} \cdot \frac{1}{\lambda_{\tau}^{MAX}} \cdot \sum_{l} a_l^{\tau} \cdot \zeta_l - \frac{1}{\delta_{\tau}^{MIN}} \cdot \mathrm{w}_{t}^{j} \sum_{l} \delta_l^{\tau} \cdot \zeta_l \tag{8}$$

s.t.

$$-y_{r\tau} \cdot \hat{\varphi} + \sum_{l \le \tau} y_{rl} \cdot \zeta_l \geq 0,\ \forall r \tag{9}$$

$$\hat{\varphi} \geq 1 \tag{10}$$

$$\sum_{l \le \tau} \zeta_l = 1 \tag{11}$$

$$\mathrm{w}_{sp}^{j} + \mathrm{w}_{t}^{j} = 1,\ \forall j \tag{12}$$

$$\zeta_l \in \{0,1\}^{|I|} \qquad \hat{\varphi}\ free \tag{13}$$

End For

End For

Objective function (8) consists of two terms: the first one uses the elements of *A* matrix (a_l^τ) multiplied by binary variables (ζ_l) and is normalized with vector λ_τ^{MAX}, whereas the second term uses the elements of $\mathbf{\Delta}$ matrix (δ_l^τ) multiplied by binary variables (ζ_l) and is normalized with vector δ_τ^{MIN}.The aim of the MO-MILP is to select the temporally closest and spatially most similar unique peer to the DMU under investigation, weighting the two objectives with w_{sp}^j and w_t^j; this Weighted Sum Model (WSM) is subjected to constraints (9) – (13).

Constraint (9) is introduced to calculate the new Spatio-Temporal efficiency $\hat{\varphi}$; it resembles constraint (3) of the initial iterative LP model except for lambda values which have been replaced with binary variables ζ_l. As the proposed model provides a unique peer, then constraint (11) guarantees that only one peer will be selected. However, selecting only one peer, as Spatio-Temporal reference set may cause $\hat{\varphi}$ to take values less than one. Thus constraint (10) is introduced in order to exclude such cases. Finally, constraint (12) states that the weights that are assigned to the temporal and spatial dimensions in the objective function are complementary. In Figure 7, the mathematical formulation of the WSM model is demonstrated in a table form as introduced to `lpSolveAPI` package. Reformulating objective function (8), the coefficients of binary variables ζ_l are shown in (14). The coefficients of constraints (9) – (14)are shown in Figure 8.

$$\sum_l \left(\frac{1}{\lambda_\tau^{MAX}} \cdot \mathrm{w}_{sp}^j \cdot a_l^\tau - \frac{1}{\delta_\tau^{MIN}} \cdot \mathrm{w}_t^j \cdot \delta_l^\tau \right) \cdot \zeta_l \qquad (14)$$

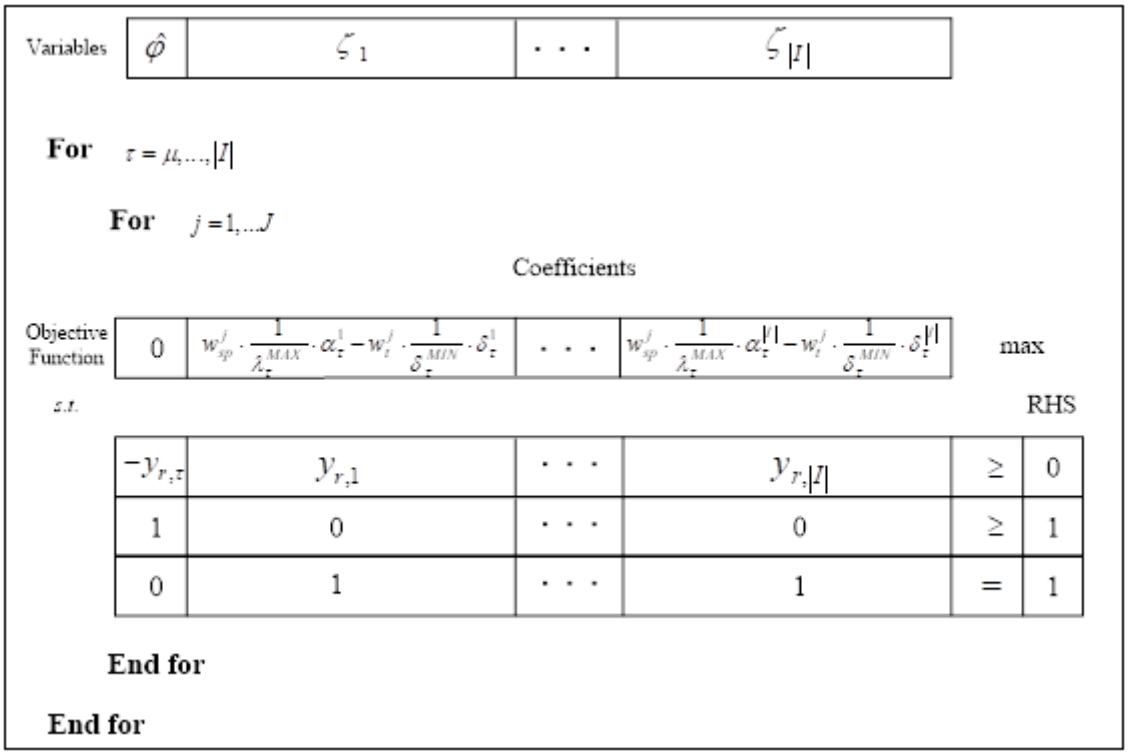

Figure 7: Formulation of MO-MILP model.

3.2 Extensions of Spatio-Temporal to find virtual outputs

Consideringthe Spatio-Temporal efficiency, DMUs are now projected on a new frontier. Thus, in addition to determining the unique and most appropriate peer of each DMU, the proposed model can also be extended in order to obtain virtual inputs and outputs on the new space and time dependent frontier, with the addition of slack variables. Thus, in addition to determining the unique and most appropriate peer of each DMU, the proposed model can also be extended in order to obtain virtual inputs and outputs on the new Spatio-Temporal frontier, with the addition of slack variables. These slack variables will be denoted as $\hat{s}_r^+$. The following model with objective function (15) and constraints (16 – 20) extends the previous MO-MILPmodel (8 – 13) with the addition of a slack variable $\hat{s}_r^+$.

For $\tau = \mu,\ldots,|I|$

For $j = 1,\ldots J$

$$\max \mathrm{w}_{sp}^j \cdot \frac{1}{\lambda_\tau^{MAX}} \cdot \sum_l a_l^\tau \cdot \zeta_l - \frac{1}{\delta_\tau^{MIN}} \cdot \mathrm{w}_t^j \sum_l \delta_l^\tau \cdot \zeta_l - \varepsilon \cdot \sum_r \hat{s}_r^+ \qquad (15)$$

s.t.

$$-y_{r\tau}\cdot\hat{\varphi}+\sum_{l\le\tau}y_{rl}\cdot\zeta_l-\hat{s}_r^{+}=0,\ \forall r \tag{16}$$

$$\hat{\varphi}\ge 1 \tag{17}$$

$$\sum_{l\le\tau}\zeta_l=1 \tag{18}$$

$$\mathrm{w}_{sp}^{j}+\mathrm{w}_{t}^{j}=1,\ \forall j \tag{19}$$

$$\zeta_l\in\{0,1\}^{|I|}$$
$$\hat{\varphi}\ \textit{free} \tag{20}$$
$$\hat{s}_r^{+}\ge 0,\ \forall r$$

End For

End For

In the objective function (15), the summation of slack variables that corresponds to the outputs is weighted with ε which is a relatively small positive number $\left[10^{-3},\ 10^{-5}\right]$. Also, constraint (9) changes with the addition of the slack variable and turns into constraint (16). The projected outputs are calculated based on the following formula:

$$\hat{y}_r^{ST}=\hat{\varphi}\cdot y_r+\hat{s}_r^{+} \tag{21}$$

3.3 Architecture of the Spatio-Temporal DEA implementation

The proposed Spatio-Temporal DEA approach has been implemented in R, which is, as already mentioned, an open-source language initially developed for statistical computing and graphics[28]. The implementation is based on two packages; the first is the `Benchmarking` package which has already been introduced and the second is the `Rgtk2`package [29]which is required to build the Graphical User Interface (GUI). The dependencies among packages are graphically depicted in Figure 9.

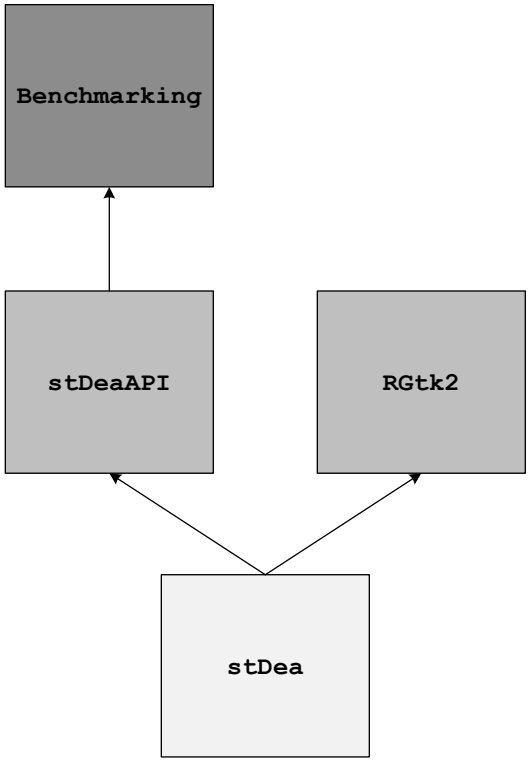

Figure 8: Package dependency of `stDea`

The core of the proposed functionality has been implemented in the `stDeaAPI` package which has been structured as an API (Application Programming Interface), that is, as a set of well-defined functions with explicit parameters. The rationale for developing an individual `stDeaAPI` package is to allow R users to reuse this functionality either in R scripts or in the context of other packages. The `stDeaAPI` is based on the `Benchmarking` package utilizing functions like `lambda(e)` for the construction of $\boldsymbol{A}$ and $\boldsymbol{\Delta}$ matrices, as described in Section 3.1.

Package `stDea` is essentially a front-end offering access to the Spatio-Temporal DEA functionality by means of a simple user interface. This package, whose functionality is

discussed in detail in the following subsection, relies on the`RGtk2` package for the construction of user screens and event handling.

The proposed R implementation is available: http://se.uom.gr/index.php/projects/stdea/ and the corresponding source code can be downloaded from https://github.com/digeo/stDeaAPI.

3.4 Function of `stDEA` package

Assuming that the data are imported by a csv external (the package imports the data in the following formats as well .txt, .xls), semi-column separated file then the command that is used for importing data is the following:

```
imported.data <- read.csv(file= file.choose(), header =TRUE, sep =";")
```

For sake of illustration, the following toy example is considered (Table 3). In this example there are 9 DMUs, one input (X1) and two outputs (O1 and O2).

Table 3: Inputs and outputs for toy example

DMU	Input (X1)	Output1 (O1)	Output2 (O2)
DMU(1)	1	6	4
DMU(2)	1	13	7
DMU(3)	1	3	9
DMU(4)	1	4	11
DMU(5)	1	11	12
DMU(6)	1	15	4
DMU(7)	1	6	12
DMU(8)	1	16	9
DMU(9)	1	10	8

The data are declared as follows in R. As it can be seen, input X1 contains only ones, thus it is modeled by the following vector:

```
X1<-rep(1, 9)
```

Outputs 1 and 2 are declared by the following commands:

```
O1<-c(6,13,3,4,11,15,6,16,10)
```

```
O2<-c(4,7,9,11,12,4,12,9,8)
```

Inputs and outputs are grouped as shown in the following commands; `x` indicates the inputs (in this case only `X1`) while `y` the outputs (`O1`, and `O2`).

```
x <-with(imported.data, cbind(X1))
y <-with(imported.data, cbind(O1, O2))
```

The stDEA method is conducted with the following. The results are assigned to variable `e`.

```
e <-stDEA(x,y)
e <-stDEA(x, y, RTS ="vrs", ORIENTATION ="out", stp =0.01)
```

In `stDEA` function, RTS option stands for Returns To Scale and the following options are available: `fdh`, `vrs`, `drs`, `crs`, `irs`, `irs2`, `add`, `fdh+`, `fdh++`, `fdh0`. The `ORIENTATION` and for the `ORIENTATION` can choose either `out` or `in`.

The following options are available after running the `stDEA` function.

The first is $`eff.DEA` which returns the result of the LP model solved iteratively (1 – 5). An example is presented below:

$$
\begin{aligned}
&\max\ \phi \\
&\quad s.t. \\
&\quad \lambda_1+\lambda_3+\lambda_3+\lambda_4+\lambda_5+\lambda_6+\lambda_7+\lambda_8+\lambda_9=1 \\
&\quad 6\lambda_1+13\lambda_2+3\lambda_3+4\lambda_4+11\lambda_5+15\lambda_6+6\lambda_7+16\lambda_8+10\lambda_9\geq 10\phi \\
&\quad 4\lambda_1+7\lambda_2+9\lambda_3+11\lambda_4+12\lambda_5+4\lambda_6+12\lambda_7+9\lambda_8+8\lambda_9\geq 8\phi \\
&\quad \lambda_1,\lambda_3,\lambda_3,\lambda_4,\lambda_5,\lambda_6,\lambda_7,\lambda_8,\lambda_9\geq 0 \\
&\quad \phi\ free
\end{aligned}
\qquad (22)
$$

The result after solving LP model (22) can be derived with the following:

```
$eff.DEA
[1] 1.328571
```

The peers (λ_j) after solving LP model (22) are the following:

```
          L1      L2      L3      L5        L6        L8
[9,] 0.3333333  0 0.00000000 0.5 0.1666667 0.0000000
```

As there is a restriction in the number of DMUs that are needed for the technique to maintain the discrimination power, only one model is solved for 1 input and 2 outputs due to minimum DMU formula $\mu=\max\{n\cdot m,\ 3\cdot(n+m)\}$.

The second information is derived after running `stDEA` function is spatio-temporal efficiency (`$eff.stDEA`). The results present the efficiency for different weights in temporal and spatial dimension. It can be seen that the efficiency changes as per time and space as the single past DMU that is selected, changes when temporal dimension is weighted in the range [0.83, 1]. The spatiotemporal efficiency for weights in the region [0, 0.82] in spatial dimension is 1.125 whereas the spatiotemporal efficiency for weight in the range [0.83, 1] is 1.1.

```
$eff.stDEA
  [1] 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [12] 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [19] 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [30] 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [37] 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [48] 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [55] 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [66] 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [73] 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125 1.125
 [84] 1.100 1.100 1.100 1.100 1.100 1.100 1.100
 [91] 1.100 1.100 1.100 1.100 1.100 1.100 1.100 1.100 1.100 1.100 1.100
```

The DMUs that are selected for each weight in time or space is presented with the following "triangle" plot (Figure 9). The reference set of DMU(9) presented in Figure 9, shows the reference set. In MILP model (8) – (13), the peers (λ_j) of each DMU are replace by binary variables ζ_l for selection of each past peer, based on spatial or temporal dimension. Due to constraint $\sum_l \zeta_l = 1$, a single past peer is selected. From Figure 9, for DMU(9), it can be seen that $\zeta_8 = 1$ for $0 \le W_{sp} \le 0.82$ ($0.18 \le W_t \le 1$) while $\zeta_5 = 1$ for $0.82 \le W_{sp} \le 1$ ($0 \le W_t \le 0.18$).

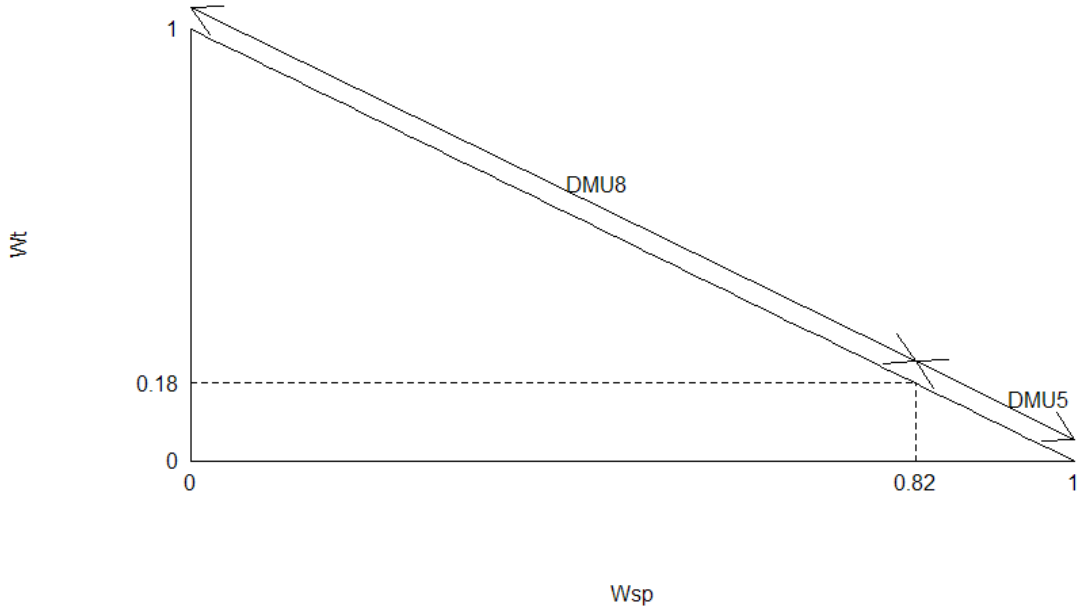

Figure 9: Spatiotemporal reference set for DMU(9).

4. Profiling analysis

To assess the performance of the `stDea` package that has been developed, 104 successive versions of the Eureka project[2] have been selected. Eureka is an open-source REST (Representational State Transfer) based service developed by Netflix for load balancing middle-tier servers at Amazon Web Services (AWS) Cloud. Each version of this software system is considered as a distinct DMU and the goal is to apply the Spatio-Temporal DEA approach to find for each selected version the most appropriate past peer.

For the analysis of each temporal DMU, selected software metrics (i.e. measures that quantify particular aspects of design quality in a software system) have been used as outputs. In particular, we have used the Coupling Between Objects (CBO), Lack of Cohesion (LCOM), and Weighted Methods per Class (WMC)[31]. These metrics quantify the coupling between software modules, the cohesion of each module and the complexity of each module, respectively. The evolution of these metrics for the examined software versions is shown in Figure 10. As already explained, for the sake of simplicity, a constant input value of 1 has been set to all DMUs.

[2]Lang ML and DT. RGtk2: R bindings for Gtk 2.8.0 and above. 2014.

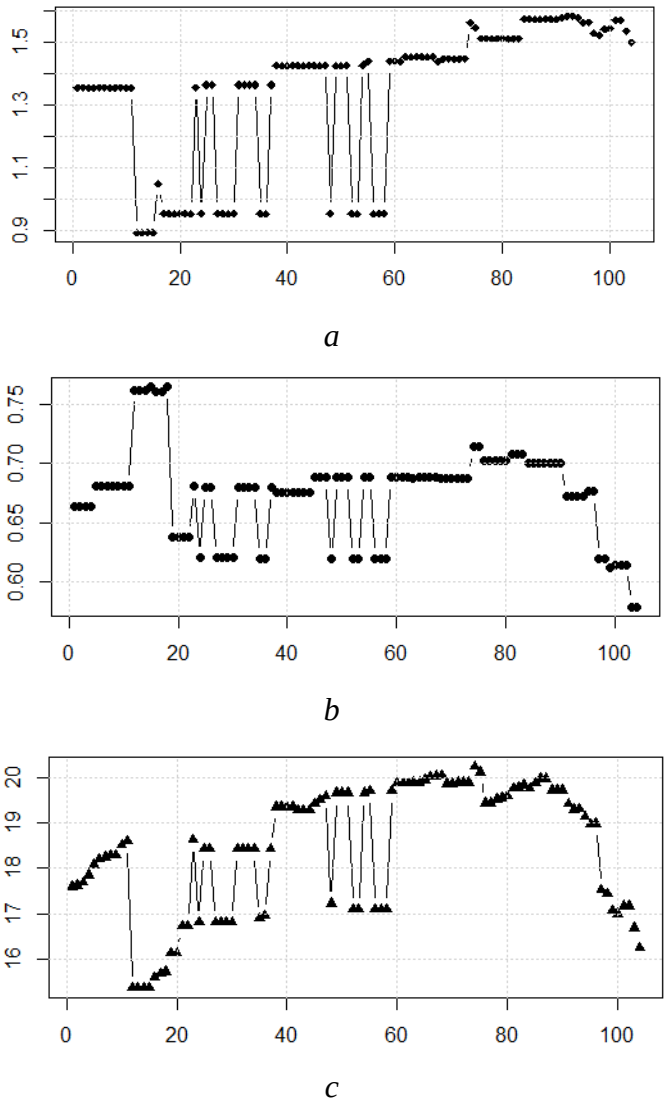

Figure 10: Metrics multiple for versions of Eureka project: a) CBO, b) LCOM, c) WMC

The results of the profiling are summarized in Table 6. To organize the results, we assume that an interested research would like to assess the performance of the software system at distinct time points, namely at the 20th, 40th, 60th, 80th and 104th version. (We note that when a version of the system is treated as a DMU under investigation, only its past versions are available. For example, for the 20th version only versions 1..20 are available).

The second and third columns list the number of LP and MILP problems solved and the fourth column indicates the corresponding CPU time. The experiments have been conducted on a desktop computer running on an Intel Core 2 Duo E8400 @ 3.00 GHz with 4 GB DDR2 RAM.

Table 4: Profiling information for each experiment set

Number of DMUs	Number of LP models	Number of MILP models	Time elapsed (CPU sec)
20	8	909	1.715
40	28	2929	6.709
60	48	4949	13.382
80	68	6969	21.679
104	76	9393	33.879

The last column of Table 6 corresponds to the average time elapsed for solving the LP and MILP models. Extending the results, a generic formula for the number of LP and MILP models solved is the following:

$$\#LP = |I| - \mu \tag{23}$$

$$\#MILP = \left(|I| - \mu + 1\right) \cdot \left(\frac{1}{h} + 1\right) = \left(|I| - \max\left\{n \cdot m, 3 \cdot (n+m)\right\} + 1\right) \cdot \left(\frac{1}{h} + 1\right) \tag{24}$$

In (22), $|I|$ stands for the number of DMUs to be analyzed (i.e. versions of the DMU under study), while the number of LP models solved ($\#LP$) equals to the number of DMUs minus the minimum number of DMUs (μ) that is needed for the DEA technique to provide robust results. In (23), the number of MILP models solved ($\#MILP$) is a function of the number of DMUs ($|I|$), μ, inputs (n), outputs (m) and the incremental step based on which weights assigned to temporal and spatial dimension increase (h). The value of h is assumed in this case to be equal to 0.01.

5. Conclusions

This study was an implementation of R language in the case of evolving systems, such as computer software. It is a fact that, classical DEA models fail to assess the units' efficiency in terms of the temporal distance between units. This shortfall is attributed to the fact that in classical DEA models, the construction of the reference set is not possible for units that are temporally arranged since it is based only on the spatial distance of units from the efficient frontier. In this context, different versions of a software product are assessed based on metrics that characterize the efficiency of each version. In this paper, the Spatio-Temporal efficiency

is examined through a two phase DEA model. Firstly, an iterative DEA model is solved for all the examined DMUs. The output of this phase is the reference set of a DMU containing only past units. Based on that reference set, the two types of information are transferred into spatial resemblance (λ values) and temporal difference (difference in time of the DMU under investigation with the units that form its reference set). Using this information, a new DEA MO-MILP model is formed, weighting spatial and temporal dimensions, which extracts the Spatio-Temporal efficiency for different combinations of weights assigned to each dimension. Based on this model, Spatio-Temporal projections are also provided based on the derived Spatio-Temporal efficiency. The model selects only a unique past peer for each DMU under investigation for any combination of weights.

The methodology has been modeled using R language, which is a platform that supports a wide variety of functions. The R package, called `stDEA`, can provide information and graphical illustration for each DMU and has no limitations concerning the number of DMUs, as well as the number of inputs and outputs. The applicability of the proposed methodology is demonstrated through a real life example from the discipline of software engineering. The evolution of Eureka project has been analyzed with `stDEA` R package, taking into account 3 metrics from 104 versions of the software. Based on this real life example, a profiling analysis of `stDEA` R package is also conducted, providing information on the number of problems solved and the CPU time elapsed.

References

[1] Färe R, Grosskopf S. Dynamic Production Models. Intertemporal Prod. Front. Dyn. DEA, Springer; 1996, p. 151–88.

[2] Emrouznejad A, Thanassoulis E. Measurement of productivity index with dynamic DEA. Int J Oper Res 2010;8:247–60.

[3] Ouellette P, Vierstraete V. Technological change and efficiency in the presence of quasi-fixed inputs: A DEA application to the hospital sector. Eur J Oper Res 2004;154:755–63. doi:10.1016/S0377-2217(02)00712-9.

[4] Prior D. Efficiency and total quality management in health care organizations: A dynamic frontier approach. Ann Oper Res 2006;145:281–99.

[5] Nemoto J, Goto M. Measurement of dynamic efficiency in production: an application of data envelopment analysis to Japanese electric utilities. J Product Anal 2003;19:191–210.

[6] Banker RD, Slaughter SA. A field study of scale economies in software maintenance. Manag Sci 1997;43:1709–25.

[7] Inman OL, Anderson TR, Harmon RR. Predicting U.S. jet fighter aircraft introductions from 1944 to 1982: A dogfight between regression and TFDEA. Technol Forecast Soc Change 2006;73:1178–87. doi:10.1016/j.techfore.2006.05.013.

[8] Borchers ST and HW. CRAN Task View: Optimization and Mathematical Programming 2014.

[9] Konis K, others. lpSolveAPI: R Interface for lp_solve Version 5.5. 2.0, R Package Version 5.5. 2.0-5. 2011.

[10] Hornik K, Meyer D, Theussl S. ROI: R Optimization Infrastructure. 2013.

[11] Canty A, Ripley B. boot: Bootstrap R (S-Plus) functions. R Package Version 2012;1.

[12] Berkelaar M, others. lpSolve: Interface to Lp solve v. 5.5 to solve linear or integer programs. R Package Version 2007;5.

[13] Henningsen A. linprog: Linear Programming. Optim R Package Version 09-0 2010.

[14] Turlach BA, Weingessel A. quadprog: Functions to Solve Quadratic Programming Problems. R Package Version 1.5-3. 2010.

[15] Varadhan R, Gilbert P. BB: An R package for solving a large system of nonlinear equations and for optimizing a high-dimensional nonlinear objective function. J Stat Softw 2009;32:1–26.

[16] Rudy J, Rudy MJ, Liu S, Matrix D. Package “CLSOCP” n.d.

[17] Karatzoglou A, Smola A, Hornik K, Zeileis A. kernlab–Kernel Methods. R Package Version 06-2 URL HttpCRAN R-Proj Org 2005.

[18] Soetaert K, Van den Meersche K, van Oevelen D. limSolve: Solving linear inverse models. R Package Version 2009;1.

[19] Geyer CJ, Meeden GD. R package rcdd (C Double Description for R), version 1.1. Incorporates code from (4). http://http://www. stat. umn. edu/geyer/rcdd; 2008.

[20] Hayfield T, Racine JS. Nonparametric econometrics: The np package. J Stat Softw 2008;27:1–32.

[21] Bogetoft P, Otto L. Benchmarking with DEA, SFA, and R. New York: Springer; 2011.

[22] Wilson PW. FEAR: A software package for frontier efficiency analysis with R. Socioecon Plann Sci 2008;42:247–54.

[23] Wilson PW. FEAR: A software package for frontier efficiency analysis with R. Socioecon Plann Sci 2008;42:247–54.

[24] Oh D, Oh MD. Package “nonparaeff” 2013.

[25] Simm J, Besstremyannaya G, Simm MJ. Package “rDEA” 2014.

[26] Shott T, Lim D-J. TFDEA: Technology Forecasting using DEA. 2015.

[27] Team RC, others. R: A language and environment for statistical computing 2012.

[28] Petridis, K., Chatzigeorgiou, A., & Stiakakis, E. (2016). A spatiotemporal Data Envelopment Analysis (ST DEA) approach: the need to assess evolving units. *Annals of Operations Research*, 238(1-2), 475-496.
[29] The R Project for Statistical Computing n.d. http://www.r-project.org/ (accessed February 26, 2015).
[30] Netflix/eureka. GitHub n.d. https://github.com/Netflix/eureka (accessed March 13, 2015).
[31] Chidamber SR, Kemerer CF. A metrics suite for object oriented design. IEEE Trans Softw Eng 1994;20:476–93. doi:10.1109/32.295895.