広島経済大学研究論集
第40巻第3号　2017年12月
http://dx.doi.org/10.18996/kenkyu2017400303

Fundamental Review on the Formulation of Large Lattice Spatial Neighbor Matrices

Gigih Fitrianto[*] and Shojiro Tanaka[**]

Abstract

To calculate the impact of each location within an observation area we need to calculate the dependences of that location. In order to specify the model to explain this condition, we must define the neighbor relation for each location. This important information is described by a spatial neighbor matrix (Cressie, 1991: Ch. 6). By using Spatial Matrix $D_{r \times c}$, which is extracted from polygon structure of spatial lattice $D_{\mathcal{M}}$, we can construct Neighbor Relation Matrix, **W**. There should be several methods to construct **W** matrix, such as: 1) Direct Arrow Reading (DAR); 2) Inner-Outer Neighbor Matrix (ION); and 3) Kronecker Product.

In this research, we verified the algorithm performances based on their time and space efficiency. All of them were calculated based on the complexity and real execution. We found that Kronecker product method became the best method to construct **W** matrix. That method can be used efficiently both in terms of computational time and space.

1. Introduction

In order to capture the influences for each location for the other location within an area or region, we must define the neighbor relation within them. By definition, the neighbor relation is defined inter-relation of one element position of (x_i, y_i) and another element position of (x_j, y_j) within $\mathcal{M}$ map, such that $\mathcal{M}$ is a two-dimensional manifold that charted from world map, $\mathbb{W} : \mathbb{W} \subseteq \mathbb{R}^3$. As stated on Cressie (1991) and Cressie and Wikle (2011: p. 167) based on a site $D_{\mathcal{M}}$ as subset of $\mathbb{R}^d$ provide an geoinformation for each $s(x_i, y_i)$ or $s_i(x, y)$ location, which is stored as lattice $\mathbf{Z} = \left(Z(s_1), \ldots, Z(s_Z)\right)^{\mathrm{T}}$, $i = 1, \ldots, Z$. Hence if there is no relation between $s_i(x, y)$ with itself, then we will have zero down the diagonals and $h_{rc} = 1$ if there is a relation between $s_i(x, y)$ and $s_j(x, y)$ (Cressie and Wikle, 2011). In order to describe inter-relation between s_i on $D_{\mathcal{M}}$ then we need a spatial neighbor relation matrix, **W**, to represent a spatial structure within the analysis (Getis and Aldstaldt, 2004; and Aldstald and Getis, 2010).

Hence this importance of **W** matrix, there should be researches which are focused on how to develop this matrix. Based on our attempt to search precedent research of spatial neighbor matrix, we found there are 180 researches focused on this subject, by using keyword: *allintitle*: "spatial * matrix" on Google Scholar on 16-20 August 2017. This attempt is used considering a research, which is focused on the development of spatial neighbor matrix, will highlight this subject in the title of their research. The result is shown in **Appendix I.A**.

However, based on our keywords, we found

[*] Graduate School of Economics, Hiroshima University of Economics
[**] Professor, Hiroshima University of Economics

several researches which are not directly related to the development of the **W** matrix. We also found several researches from the other subjects that are using "spatial * matrix" as their term. For example, based on **Appendix I.A**: Tegmark (1996), Gershman *et al* (1997), Roberts (1999), Roberts (2000), and Snell and Fuller (2010). Therefore, we eliminate and summarize the related topics in **Appendix I.B**. As shown in **Appendix I.B**, we found 32 of 180 papers, which are considered to have an impact on the development of the matrix formulation.

Furthermore, if we deepen our analysis of that result, we found Aldstald and Getis (2006) that do not only provide theoretical construction but also practical technique to apply their theoretical formulation. Unfortunately, the paper that we found is using their technique on the distance-based **W** matrix. Therefore, based on our search attempts, we have not found a precedent research of the formulation of binary spatial neighbor matrix **W** in lattice data structure, which provides us a theoretical and practical formulation. Row-standardized **W** matrix can be easily derived from the binary **W** matrix.

We considered that the presentation of practical as well as the theoretical formulation of spatial neighbor matrix will provide more knowledge and an easier attempt of replication for further research. Therefore, in this paper, we aim to provide not only the theoretical but also practical formulation of spatial neighbor matrix, **W**. Furthermore, in the practical section of this paper (Section **III** and **IV**) we provide several techniques to formulate **W** matrix. Besides that, we also try to pursue the best and the most efficient computational technique that can be used.

The most efficient method that provides in this research aims to provide a future reference for spatial analysis. This condition is directly corresponding to provide minimize cost to do spatial analysis research. For example, if we want to analyze satellite data, then we have to deal with large lattice data structure. For example, satellite data for Jakarta and its neighborhood areas from Landsat Data 8[1]. The satellite image has 2061×1688 pixels dimension which can be translated into a 2061×1688 dimension matrix. Later section (**Section 4**) in this research we show that the efficient methods will have a big different impact to deal the large data structure.

Therefore, in order to accomplish our objective, we try to formulate an effective construction of weighted neighbor relation matrix based on a rectangular $\mathcal{M}$. There are four methods, that will be compared regarding their efficiency, such as 1) Direct Arrow Reading (DAR); 2) Inner-outer neighbor method (ION); 3) 'cell2nb' and 'poly2nb' function on *spdep* package in R; and 4) Kronecker product (KP). *Spdep* package is chosen hence the package is regularly used by the applications of spatial analysis with R language program, such as Bivand *et al* (2008), Arbia (2014), and Kelejian and Piras (2017).

The methods will be discussed further in section III in this paper. On the other hand, all methods will be evaluated based on three computational criteria: 1) time complexity; 2) space complexity, and; 3) actual simulation of elapsed time and object size.

2. Weighted Neighbor Relation Matrix

Construction of Weighted Neighbor Matrix started from constructing polygon structure object from the raster map. At this beginning step, we charted an open set (region), $\mathbf{M} \subseteq \mathbb{R}^3$ into two-dimensional system, which is defined any element $m \in \mathbf{M}$ inherits position $m(x,y,z) \in \mathbb{R}^3$ that will be charted into $m'(x,y) \in \mathbb{R}^2$. Collection of $m \in \mathbf{M}$ called map of **M**. However, in this research any elements $m(x,y) \in \mathbf{M}$ not only inherit latitude and longitude position on $\mathbb{R}^3$ but also carry on geoinformation on any *m*. We define this as spatial data of $\mathcal{M}$. The natural geomorphology with water bodies and dessert that carried on in the lattice of rectangular **D** image matrix of **M**, can be easily converted by using projection matrix (Tanaka and Nishii, 2009).

Based on this projected $\mathcal{M}$ map we can construct the lattice formation as follows,

$$D_{\mathcal{M}} = \{m_i(x_i, y_i) : i = 1, \dots, Z\} \tag{1}$$

where, *x* is longitude and *y* is latitude, and *Z* is the last cell formed by an (x,y) formation on $\mathcal{M}$ map. Hence, subset cell $D_{\mathcal{M}}$ can be constructed by any method of projection within **M** region, then we can have those subset cell m_i. Let, $s(x,y)$ is a notation for each cell, then each $s(x,y)$ constructed by the value of (x,y) coordinate formation as follow

$$s(x,y) = (x_i, y_i) : i = 1, \dots, Z \tag{2}$$

By substitute equation (2) into the lattice formation on equation (1) then we will have,

$$D_{\mathcal{M}} = \{s(x_i, y_i) : i = 1, \dots, Z\} \tag{3}$$

Equation (3) creates a spatial lattice, $D_{\mathcal{M}}$ based on grid cells. In this research, we try to use hyperbolic rectangle projection of **M** region, which implies $\mathcal{M}$ become a rectangular map with $l = (x_{max} - x_{min})$ and $w = (y_{max} - y_{min})$. Based on this map we can easily extract geo-information value of spatial $\mathcal{M}$ into spatial lattice matrix, $\mathbf{D}_{(r \times c)}$, where $\boldsymbol{r}$ and $\boldsymbol{c}$ are represented the row and column matrix for **D**. Each element represents 1 km square grid area of the map. This process demonstrated by Tanaka and Nishii (2009).

On the other hand, the neighbor relation between each element $\mathbf{D}_{(r \times c)}$ also represents each rectangular lattice within $D_{\mathcal{M}}$. We are able to retrieve the neighborhood information for each $s(x_i, y_i)$. If $i > 1$ on equation (2), then there will exists a neighbor for each $s(x_i, y_i)$. Cressie (1991: p. 384-385) defined that each spatial lattice *D* there will be always any N_i neighborhood from sub-area *i* on *D*.

$$N_i = \{k : k \text{ is a neighbor of } i\} \quad i = 1, \dots, Z \tag{4}$$

$$D_N = \{(i; N_i) : i = 1, \dots, Z\} \tag{5}$$

where, D_N is a spatial lattice which contains neighborhood information for each *i*. Besides, the *k* neighbor defines by Euclidian distance from *i* sub-area.

Translate (5) into (4) structure then we will have,

$$N_i = \\ \{s(x_j, y_j) : s(x_j, y_j) \text{ is a neighbor cell of } s(x_i, y_i)\}, \\ i \neq j,\ i = 1, \dots, Z \tag{6}$$

where, N_i constructs as follows,

$$N_i = \{s(x_j, y_j) : (x,y) \in D_{\mathcal{M}}\} \tag{7}$$

where, *j* is neighbor index of $s(x_i, y_i)$ and *k* is cell-distance for $k = s(x_j, y_j) - s(x_i, y_i)$ on cell unit. The coordinate for $s(x_j, y_j)$ constructed by cell which are separated *k*-cell away from $s(x_i, y_i)$. Under this argument then,

$$\{x, y | (x_i, y_i)\} = \begin{cases} x_j^+ \ or \ y_j^+ \ for \ x_i + k \ or \ y_i + k \\ x_j^- \ or \ y_j^- \ for \ x_i - k \ or \ y_i - k \end{cases}$$

(8)

In this research we using $k=1$ and using the nearest neighbor cell. Therefore, there will be $N_i = 8$, which are found by following properties:

$$N_i = \begin{cases} s\left(x_j^+, y_i\right) = \left(x_i + 1, y_i\right) \\ s\left(x_j^+, y_j^+\right) = \left(x_i + 1, y_i + 1\right) \\ s\left(x_i, y_j^+\right) = \left(x_i, y_i + 1\right) \\ s\left(x_j^-, y_j^+\right) = \left(x_i - 1, y_i + 1\right) \\ s\left(x_j^-, y_i\right) = \left(x_i - 1, y_i\right) \\ s\left(x_j^-, y_j^-\right) = \left(x_i - 1, y_i - 1\right) \\ s\left(x_i, y_j^+\right) = \left(x_i, y_i - 1\right) \\ s\left(x_j^+, y_j^-\right) = \left(x_i + 1, y_i - 1\right) \end{cases} \tag{9}$$

Thus, we can construct relation on equation (9) into following figure based on Tanaka and Nishii

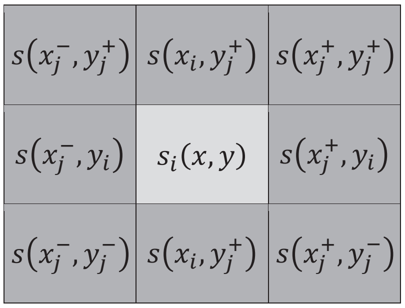


Figure 1　k=1 Neighbors Cell System of $s(x_i, y_i)$

(2009) as follows,

Figure 1 shows us the possibility of neighbor cell construction of $s\left(x_j, y_j\right)$ from cell distance $k=1$. Construction of N_i on equation (9) and **Figure 1** is defined as *Nearest Neighbor Set,* N_i (Cressie, 1991), which is a Neighborhood construction that captures all possible neighbor cells with $k=1$. N_i structure itself can be partitioned into two nearest neighbor subsets, which are called *Primary Nearest Neighbor* and *Secondary (Oblique) Nearest Neighbor.*

A. Primary Nearest Neighbor Cells (1*N*)

1*N* structure is capture neighborhood cell $s\left(x_j, y_j\right)$ by horizontal-vertical neighbor relationship with $s\left(x_i, y_i\right)$. The relationship shown by

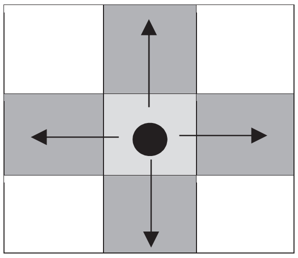


Figure 2　Neighbor Cell Construction of 1*N*

Figure 2 below.

The list of 1*N* Cells denotes as $s_1\left(x_j, y_j\right)$, which is consists of

$$1N_i = \begin{cases} s_1\left(x_j^+, y_i\right) = \left(x_i + 1, y_i\right) \\ s_1\left(x_i, y_j^+\right) = \left(x_i, y_i + 1\right) \\ s_1\left(x_j^-, y_i\right) = \left(x_i - 1, y_i\right) \\ s_1\left(x_i, y_j^-\right) = \left(x_i, y_i - 1\right) \end{cases} \tag{10}$$

B. Secondary (Oblique) Nearest Neighbors (2*N*)

On the other hand, 2*N* system capture diagonal neighbor relationship with $s\left(x_i, y_i\right)$,

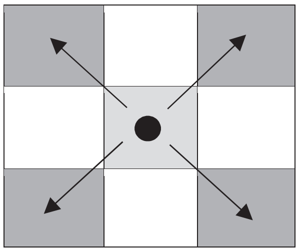


Figure 3　Neighbor Cell Construction of 2*N*

which is constructed on **Figure 3**.

The list of 2*N* Cells denotes as $s_2\left(x_j, y_j\right)$, which is consists of,

$$2N_i = \begin{cases} s_2\left(x_j^+, y_j^+\right) = \left(x_i + 1, y_i + 1\right) \\ s_2\left(x_j^-, y_j^+\right) = \left(x_i - 1, y_i + 1\right) \\ s_2\left(x_j^-, y_j^-\right) = \left(x_i - 1, y_i - 1\right) \\ s_2\left(x_j^+, y_j^-\right) = \left(x_i + 1, y_i - 1\right) \end{cases} \tag{11}$$

Based on those two sub-categories of N_i and substitute (10) and (11) into (9), then we will have identity that N_i is

$$N_i = 1N_i + 2N_i \tag{12}$$

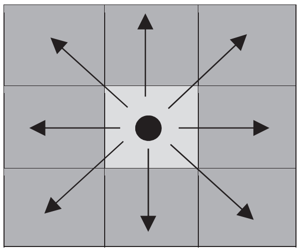


Figure 4 Neighbor Cell Construction of N_i

From those neighbors list then we can make a weighted list configuration of neighbor properties. In this research, we use binary methods in order to construct the weighted list of neighbor cell. The binary methods created as follows,

$$s_N\left(x_j, y_j\right) = \begin{cases} 1, s\left(x_j, y_j\right) \in N_i \\ 0, s\left(x_j, y_j\right) \notin N_i \end{cases} \tag{13}$$

If we are using primary style of equation (10), we will have,

$$s_1\left(x_j, y_j\right) = \begin{cases} 1, s\left(x_j, y_j\right) \in 1^{\text{st}}\text{-N}_i \\ 0, s\left(x_j, y_j\right) \notin 1^{\text{st}}\text{-N}_i \end{cases} \tag{14}$$

condition. Moreover, if we using secondary style on equation (11), then we have

$$s_2\left(x_j, y_j\right) = \begin{cases} 1, s\left(x_j, y_j\right) \in 2^{\text{nd}}\text{-N}_i \\ 0, s\left(x_j, y_j\right) \notin 2^{\text{nd}}\text{-N}_i \end{cases} \tag{15}$$

Therefore, given any **D** image matrix of any map $\mathcal{M}$, then the weighted neighbor relation matrix **W** become,

$$\mathbf{W}_{Z\times Z} = \mathbf{WP}_{Z\times Z} + \mathbf{WS}_{Z\times Z} \tag{16}$$

where, Z is the last cell formed by an (x, y) formation on $\mathcal{M}$ map. $\mathbf{WP}_{Z\times Z}$, called as *primary neighbor* relation matrix, is **W** neighborhood matrix for N_i. On the other hand, $\mathbf{WS}_{Z\times Z}$, called as *secondary neighbor* relation matrix, is **W** neighborhood matrix for $2N_i$. Besides, Z value on equation (16) is also called as a "datasize" of matrix $\mathbf{D}_{r\times c}$ hence each $Z \equiv r \times c$ row is represented all $s(x_i, y_i)$ on $\mathbf{D}_{\mathcal{M}}$.

3. Construction Methods of W Matrix

In this section, we will discuss several methods, used in order to construct spatial neighbor matrix $\mathbf{W}_{Z\times Z}$. However, in this research, we focused on how we construct this matrix by using R language program. By using R language program there are several methods that can be used, such as: 1) direct arrow reading (DAR); 2) inner-outer neighbor matrix method (ION); 3) 'cell2nb' and 'poly2nb' function on *spdep* R package; and 4) Kronecker product (KP). The first two methods are also called the *naïve* methods, where the neighbor searching process is directly from each cell on the image matrix.

3.1 Direct Arrow Reading Method (DAR)

Direct arrow reading (DAR) is a naïve and simplistic method by literally apply arrow direction during the neighbor searching process, which is illustrated in **Figure 2**, **Figure 3**, and **Figure 4**. Based on equation (13), (14), and (15) above, we can create a spatial weighted neighbor matrix, which provides distance-based nearest neighbor. Those process can be shown on **Figure 5** below. Assume **D** matrix has 3×3 dimension, defined as *Y*- row and *X*-column matrix. By using $\mathbf{D}_{3\times3}$ image matrix, then we will have $\mathbf{W}_{(3\cdot3)\times(3\cdot3)}$.

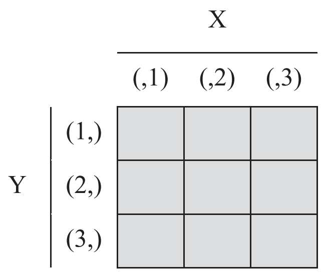


Figure 5　Construction Weighted Neighbor Matrix from $\mathbf{D}_{3\times3}$ Grid Cell

By the neighbor searching process on **Figure 4** then we will have neighbor relation matrix of $\mathbf{D}_{3\times3}$ on following **Table 1** and **Table 2**.

Based on those two construction we can construct 2*N* by subtract Total Neighbor Matrix with 1*N* matrix, which is defined by equation (12).

Script that used in this method of construction is defined by **Script III.A** in **Appendix III**. Each block on **Script III.A** represents each arrow direction to identify the neighbor cell of each cell. Generally, DAR process can be illustrated in **Figure 6** above.

Table 1　Construction $\mathbf{W}_{9\times9}$ Spatial Weighted Neighborhood Matrix

s_N (1,1)	0	1	0	1	1	0	0	0	0
s_N (1,2)	1	0	1	1	1	1	0	0	0
s_N (1,3)	0	1	0	0	1	1	0	0	0
s_N (2,1)	1	1	0	0	1	0	1	1	0
s_N (2,2)	1	1	1	1	0	1	1	1	1
s_N (2,3)	0	1	1	0	1	0	0	1	1
s_N (3,1)	0	0	0	1	1	0	0	1	0
s_N (3,2)	0	0	0	1	1	1	1	0	1
s_N (3,3)	0	0	0	0	1	1	0	1	0

Table 2　Construction $\mathbf{W}_{9\times9}$ Weighted Neighborhood Matrix for $1N_i$

s_1 (1,1)	0	1	0	1	0	0	0	0	0
s_1 (1,2)	1	0	1	0	1	0	0	0	0
s_1 (1,3)	0	1	0	0	0	1	0	0	0
s_1 (2,1)	1	0	0	0	1	0	1	0	0
s_1 (2,2)	0	1	0	1	0	1	0	1	0
s_1 (2,3)	0	0	1	0	1	0	0	0	1
s_1 (3,1)	0	0	0	1	0	0	0	1	0
s_1 (3,2)	0	0	0	0	1	0	1	0	1
s_1 (3,3)	0	0	0	0	0	1	0	1	0

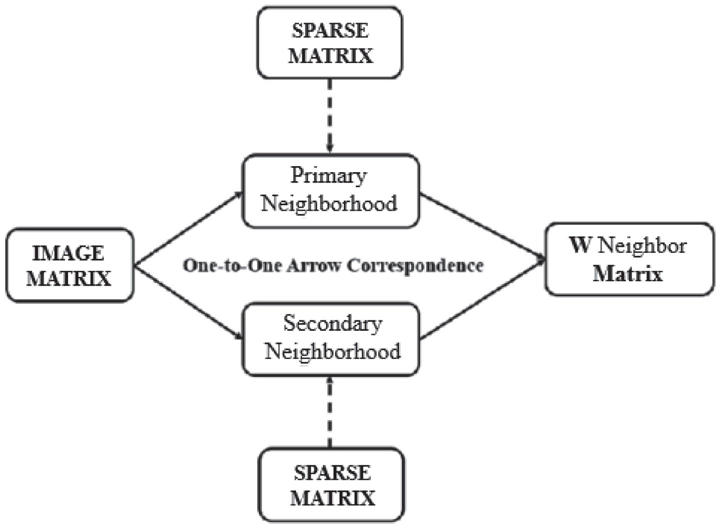


Figure 6　Direct Arrow Reading Process

3.2 Inner-Outer Neighbor Method (ION)

The second method to construct the **W** matrix is Inner-Outer Neighbor (ION) method. The searching process is to calculate the neighborhood from the inner matrix and after that, we calculate the outer matrix area neighborhood later. This method. is used in order to optimize DAR time complexity, which is aimed to produce more efficient method than DAR method (further analysis regarding this efficiency will explain further and shown in **Table 3** and **Table 4** comparison).

In this ION method, we divide **D** image matrix into two subsets of the cell, which are inner cells and outer cells. Inner cell subset of **D** are defined,

$$Inner_{r'\times c'} = \mathbf{D}_{\tilde{\mathrm{R}}\times\tilde{\mathrm{C}}} \tag{17}$$

where, $\tilde{\mathbf{R}} = (\max[r]-1)-(\min[r]+1)$ and $\tilde{\mathbf{C}} = (\max[c]-1)-(\min[c]+1)$. On the other hand, $Outer_{r'\times c'}$ consists of the first and last section of *r* row and *c* column of matrix **D**. The construction process is illustrated as follows,

For example, let assume we have $\mathbf{D}_{4\times4}$, then the outer matrix of that matrix is $\mathbf{D}_{3'\times3'} \subset \mathbf{D}_{4\times4}$ as illustrated as follows, ION method calculated separately hence inner matrix will always have all 8-Neighbor constructed at equation (9). On the other hand, every cell of $Outer_{r'\times c'}$ will have less than 8-Neighbor due to the restricted condition of equation (7) and shown in **Figure 8** configuration.

The Construction by using Inner-Outer matrix is generated by using **Script III.B** in **Appendix III**.

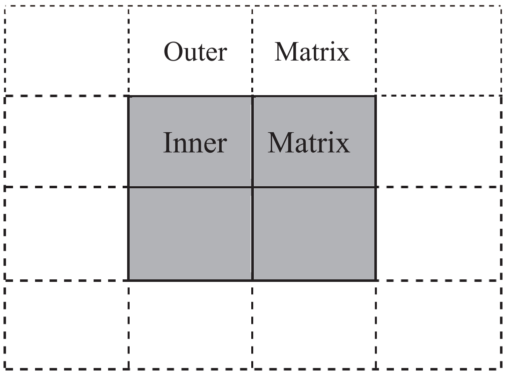


Figure 8 Inner-Outer Matrix Configuration of $\mathbf{D}_{4\times4}$

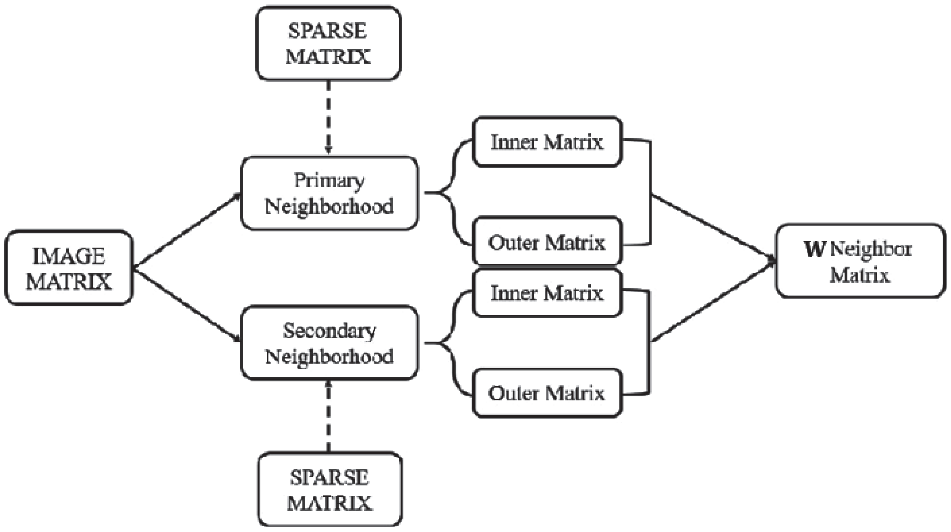


Figure 7 Inner-outer Neighbor Method Process

3.3 Kronecker Product (KP)

By definition, Kronecker product is an outer product representation of two matrices in form of block matrix. Kronecker product denoted by ⊗ symbol. For example, given an A and B matrix as follows,

$$\boldsymbol{A}_{ji} = \begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix} \text{ and}$$

$$\boldsymbol{B}_{ji} = \begin{bmatrix} b_{11} & \cdots & b_{m1} \\ \vdots & & \vdots \\ b_{1n} & \cdots & b_{mn} \end{bmatrix}$$

Thus, we will have $A \otimes \mathrm{B}$ relation as follows,

$$\left\{\boldsymbol{A} \otimes \mathbf{B}\right\}_{m\times n} =$$

$$\begin{bmatrix} a_{11}\begin{bmatrix} b_{11} & \cdots & b_{m1} \\ \vdots & & \vdots \\ b_{1n} & \cdots & b_{mn} \end{bmatrix} & \cdots & a_{m1}\begin{bmatrix} b_{11} & \cdots & b_{m1} \\ \vdots & & \vdots \\ b_{1n} & \cdots & b_{mn} \end{bmatrix} \\ \vdots & & \vdots \\ a_{1n}\begin{bmatrix} b_{11} & \cdots & b_{m1} \\ \vdots & & \vdots \\ b_{1n} & \cdots & b_{mn} \end{bmatrix} & \cdots & a_{mn}\begin{bmatrix} b_{11} & \cdots & b_{m1} \\ \vdots & & \vdots \\ b_{1n} & \cdots & b_{mn} \end{bmatrix} \end{bmatrix}_{(ji)\times(ji)}$$

By using this relationship, we can directly construct a spatial neighbor matrix, **W**. Relation of those two concepts shown by Tanaka and Nishii (2009). However, Tanaka and Nishii (2009) only shown us how to construct **W** which only consists of primary neighbor relationship. Therefore, in this paper, we try to construct primary and secondary neighbor relation by using a Kronecker product.

First, we need to adopt Kronecker delta of tensor into matrix component to construct a *shift matrix*.

Definition 3.1. (Wedderburn, 1934)

Kronecker delta δ_{ji} of matrix **A** means that,

$$\mathrm{a}_{ji} := \begin{cases} 1 \text{ if } j = i \\ 0 \text{ if } j \neq i \end{cases} \text{ such that } \mathrm{a}_{ji} \text{ element of } \mathbf{A}$$

Therefore, by **Definition 3.1** we can define $\mathbf{I}_{ii} = \delta_{ji}$. On the other hand, if we have a Kronecker delta $\delta_{j+1,i}$ and $\delta_{j,i+1}$ of matrix **A**, then it means,

$$\mathrm{a}_{ji} := \begin{cases} 1 \text{ if } j+1 = i \\ 0 \text{ if } j+1 \neq i \end{cases} \tag{18}$$

such that a_{ji} element of **A**

and,

$$\mathrm{a}_{ji} := \begin{cases} 1 \text{ if } j = i+1 \\ 0 \text{ if } j \neq i+1 \end{cases} \tag{19}$$

such that a_{ji} element of **A**

respectively.

Under condition on equation (18), the value of **A** similar to shifted the value of $\mathbf{I}_{ii}$ one column to the left. Thus, we called this matrix as *left-shifted* matrix, denoted as $\mathbf{L}_{ji}$. On the other hand, if we apply condition (19) into matrix **A** the value of a_{ji} will be the same if we shifted matrix $\mathbf{I}_{ii}$ one row upward. Therefore, we called this matrix as an *upper-shifted* matrix, denoted as $\mathbf{U}_{ji}$. Assume we have $\mathbf{I}_{i}$ as follows,

$$\mathbf{I}_i = \begin{bmatrix} 1 & 0 & 0 & & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & & \cdots & 1 \end{bmatrix}_{(r\times r) \text{ or } (c\times c)}$$

Thus, we will have $\mathbf{U}_{ji}$ and $\mathbf{L}_{ji}$ respectively as,

$$\mathbf{U}_{ji} = \begin{bmatrix} 0 & 1 & 0 & 0 & & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & & 0 \\ & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{(r\times r)\text{ or }(c\times c)} \qquad (20)$$

$$\mathbf{L}_{ji} = \begin{bmatrix} 0 & 0 & 0 & & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}_{(r\times r)\text{ or }(c\times c)} \qquad (21)$$

The special consequences of equation (20) and (21) structure, then we will have conditions that $\mathbf{U}_{ji} = (\mathbf{L}_{ji})'$ and $\mathbf{L}_{ji} = (\mathbf{U}_{ji})'$. Moreover, assume we have s-times shifted matrix of an $\mathbf{I}_n$ identity matrix, then we can denote the shifted process matrix as follows,

$\mathbf{L}_{n,s}$ for s left-shifted $\mathbf{I}_n$, and
$\mathbf{U}_{n,s}$ for s up-shifted $\mathbf{I}_n$ (22)

Afterwards, we can apply the *shift matrix* into the construction of **W** matrix. Suppose we have the image matrix $\mathbf{D}_{r\times c}$, where r and c are each row and column of **D** respectively, Tanaka and Nishii (2009) define that,

$$\mathbf{W}_{\text{RC}\times\text{RC}} = (\mathbf{L}_\text{R} + \mathbf{U}_\text{R}) \otimes \mathbf{I}_\text{C} + \mathbf{I}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C}) \qquad (23)$$

However, in order to capture primary and secondary relation of nearest neighborhood (1N and 2N) we need to expand equation (23) into following equation,

$$\begin{aligned}\mathbf{W}_{\text{RC}\times\text{RC}} &= (\mathbf{L}_\text{R} + \mathbf{U}_\text{R}) \otimes \mathbf{I}_\text{C} + \mathbf{I}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C}) \\ &\quad + \mathbf{U}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C}) \\ &\quad + \mathbf{L}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C}) \end{aligned} \qquad (24)$$

where,

$$\begin{aligned}\mathbf{WP}_{Z\times Z} &= (\mathbf{L}_\text{R} + \mathbf{U}_\text{R}) \otimes \mathbf{I}_\text{C} + \mathbf{I}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C}) \\ \mathbf{WS}_{Z\times Z} &= \mathbf{U}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C}) + \mathbf{L}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C})\end{aligned}$$

Equation (24) above can be breakdown into the neighbor searching process by applying associative property to Kronecker product. By properties of neighbor, searching is illustrated by arrow direction as shown in **Figure 9**.

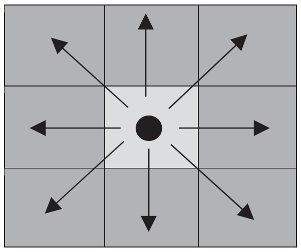


Figure 9 Neighbor Searching from Central Point

By applying associative property on equation (24) we will have,

$$\begin{aligned}\mathbf{W}_{\text{RC}\times\text{RC}} &= (\mathbf{L}_\text{R} \otimes \mathbf{I}_\text{C}) + (\mathbf{U}_\text{R} \otimes \mathbf{I}_\text{C}) + (\mathbf{I}_\text{R} \otimes \mathbf{L}_\text{C}) \\ &\quad + (\mathbf{I}_\text{R} \otimes \mathbf{U}_\text{C}) + (\mathbf{U}_\text{R} \otimes \mathbf{L}_\text{C}) \\ &\quad + (\mathbf{U}_\text{R} \otimes \mathbf{U}_\text{C}) + (\mathbf{L}_\text{R} \otimes \mathbf{L}_\text{C}) \\ &\quad + (\mathbf{L}_\text{R} \otimes \mathbf{U}_\text{C})\end{aligned}$$

After that, we can rearrange them into following equation,

$$\begin{aligned}\mathbf{W}_{\text{RC}\times\text{RC}} &= (\mathbf{L}_\text{R} + \mathbf{U}_\text{R}) \otimes (\mathbf{I}_\text{C} + \mathbf{L}_\text{C} + \mathbf{U}_\text{C}) \\ &\quad + \mathbf{I}_\text{R} \otimes (\mathbf{L}_\text{C} + \mathbf{U}_\text{C})\end{aligned} \qquad (25)$$

Consider that equation (25) represent each 1-grid (nearest) neighborhood, N_j, from each cell in image matrix. Let us want to search *k*-grid neighborhood of each cell, then by applying identity on equation (22) for *s* shifted matrix[2], we will have,

$$\mathbf{W}(k) = \left(\sum_{s=1}^{k} \mathbf{L}_{R,s} + \mathbf{U}_{R,s} \otimes \left\{ \mathbf{I}_C + \sum_{s=1}^{k} \mathbf{L}_{C,s} + \mathbf{U}_{C,s} \right\} \right) + \left(\mathbf{I}_R \otimes \sum_{s=1}^{k} \mathbf{L}_{C,s} + \mathbf{U}_{C,s} \right) \tag{26}$$

where,

$$\mathbf{WP}(k) = \left(\sum_{s=1}^{k} \mathbf{L}_{R,s} + \mathbf{U}_{R,s} \otimes \mathbf{I}_C \right) + \left(\mathbf{I}_R \otimes \sum_{s=1}^{k} \mathbf{L}_{C,s} + \mathbf{U}_{C,s} \right)$$

$$\mathbf{WS}(k) = \left(\sum_{s=1}^{k} \mathbf{L}_{R,s} + \mathbf{U}_{R,s} \right) \otimes \left(\sum_{s=1}^{k} \mathbf{L}_{C,s} + \mathbf{U}_{C,s} \right)$$

In case of Kronecker product construction, we used two different construction methods: 1) By using *Shift Matrix* function of *matrixcalc* package; and 2) By using our generic function to construct Shift Matrix. Both methods are built by using Sparse matrix construction. In order to use sparse matrix, we need to transform the matrix-base class into ‘CsparseMatrix’ class on *Matrix* package. ‘CsparseMatrix’ class is the virtual class of all sparse matrices coded in sorted compressed column-oriented form (Bates and Maechler, 2017)

3.3.1 Shift Matrix by *matrixcalc* Package

There are four types of shift function that provided by *matrixcalc* package, which each of one of them represents the direction of shift. The function explained as follows (Novometsky, 2015),

$$\left.\begin{array}{l}\text{shift.up}\\ \text{shift.down}\end{array}\right\} (\mathbf{A},\ \text{rows}=\ s_1,\ \text{fill}=\textit{fill})$$

$$\left.\begin{array}{l}\text{shift.right}\\ \text{shift.left}\end{array}\right\} (\mathbf{A},\ \text{cols}=\ s_2,\ \text{fill}=\textit{fill})$$

where, $\mathbf{A}$ is a matrix, s_1 is the number of row-direction of shift process, s_2 is the number of column-direction of shift process, and *fill* is fill value which as the default value of zero. However, hence $\mathbf{A}_{ji}$ matrix that shifted on our analysis is $\mathbf{I}_{ii}$, and suppose we have $\mathbf{D}_{ji}$ is a ‘shift.down’ matrix and $\mathbf{R}_{ji}$ is a ‘shift.right’ matrix, then If and only if $\mathbf{A}_{ji}$ is $\alpha \cdot \mathbf{I}_i$, then $\mathbf{U}_{ji} = \mathbf{R}_{ji}$ and $\mathbf{D}_{ji} = \mathbf{L}_{ji}$.

By using R program and shift matrix function inside of *matrixcalc* package, then we can write the algorithm as follows,

```
library (Matrix);
library (matrixcalc)

ic ← diag(1, ncol = c, nrow = c);
ir ← diag(1, ncol = r, nrow = r);
uc ← shift.up(ic); lc ← shift.left(ic);
ur ← shift.up(ir); lr ← shift.left(ir)

# Block 1–Block 8 #
ic.s ← as(ic, "CsparseMatrix"); ir.s ← as(ir, "CsparseMatrix")
uc ← as(uc, "CsparseMatrix"); lc ← as(rc, "CsparseMatrix")
ur ← as(ur, "CsparseMatrix"); lr ← as(rr, "CsparseMatrix")
```

```
ac ← uc+lc; ar ← ur+lr

# Block 9 #
W ← kronecker (ar,ir.s) + kronecker(ic.s, ar) + kronecker(ur, ac) + kronecker(lr, ac)
```

Note: r and c is each length of row and column of $\mathbf{D}_{r\times c}$ within simulation.

Script 1 Kronecker Product Method using *matrixcalc*

3.3.2 Generic Function for Shift Matrix

In the second approach, instead of using a package we try to build our generic function by using R Core function. The function constructed by following **Script 2**.

```
## Constructing the Function ##
left ← function(dims){
   require(Matrix)
   B ←as(diag(1, dims–1, dims–1), "CsparseMatrix")
   D ←as(matrix(rep(0, dims), nrow = dims, ncol = 1), "CsparseMatrix")
   E ←cbind(rbind(t(D[1:dims–1]), B), D)
   return(E)
}

up <– function(dims){
   require(Matrix)
   B ←as(diag(1, dims–1, dims–1), "CsparseMatrix")
   D ←as(matrix(rep(0, dims), nrow = dims, ncol = 1), "CsparseMatrix")
   E ←cbind(D, rbind(B, t(D[1:dims–1])))
   return(E)
}

## Elapsed Time Calculation ##
ic ←as(diag(1, ncol = c, nrow = c), "CsparseMatrix")
ir ← as(diag(1, ncol = r, nrow = r), "CsparseMatrix")
dim.c ←ncol(ic);
dim.r <– ncol(ir)

# Block 1–Block 6 #
ac ←left(dim.c)+up(dim.c);
ar ← left(dim.r)+up(dim.r);
uc ←up(dim.c); lc ← left(dim.c);
ur ←up(dim.r); lr ← left(dim.r);

# Block 7 #
W ← kronecker((lr+ur), ic)+ kronecker(ir, (lc+uc))+
       kronecker(ur, (lc+uc)) + kronecker(lr, (lc+uc))
```

Note: r and c is each length of row and column of $\mathbf{D}_{r\times c}$ within simulation.
dims is referenced to the value of $r\times c$

Script 2 Kronecker Product Method using Generic Function for Shift Matrix

In order to derive complete analysis for our Kronecker product attempts, we need to compare both methods with the other methods for construction of **W** matrix (The result shown in **Section IV**). As general, we can illustrate execution process for Kronecker product methods as follows,

3.4 Polygon and Cell Based on *spdep* R Package

The *spdep* R package is a widely used library to analyze spatial research. One of two functions, which are available within this package is also widely used by the researcher to construct spatial neighbor matrix. The functions are ‘poly2nb’ and ‘cell2nb’. As general, the process of execution of those two functions is shown in **Figure 11a** and **Figure 11b**.

Based on **Figure 11a** and **Figure 11b**, the ‘cell2nb’ and ‘poly2nb’ function are used to generate a neighbor list class for each cell within an image matrix. In order to convert that list into a spatial neighbor matrix, **W**, then we need one or two more functions on *spdep*. As provides in general steps to construct **W** matrix from a list of neighbors, we need a ‘nb2mat’ functionality

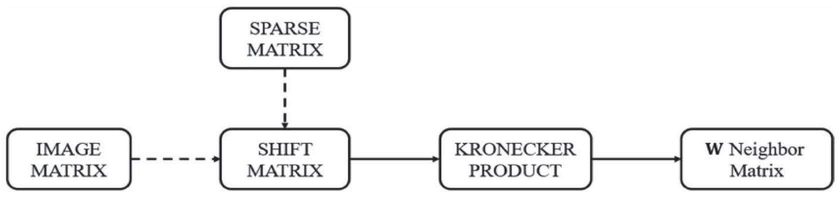


Figure 10 Kronecker Product Method by using Shift Matrix

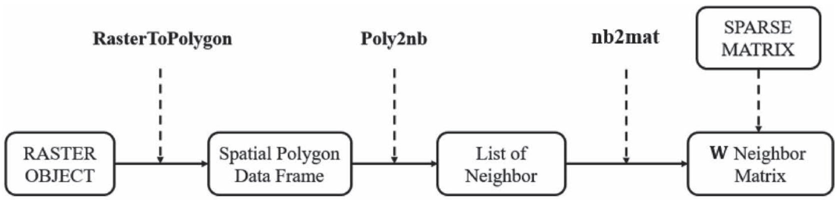


Figure 11a Illustration of ‘poly2nb’ Process

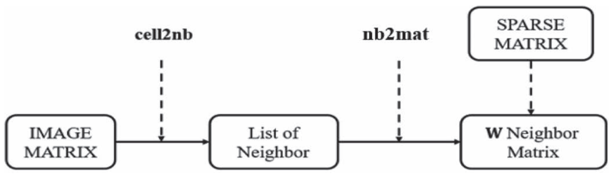


Figure 11b Illustration of ‘cell2nb’ Process

```
library(raster); library(sp); library(spdep)

# Block 1–Block 3 #
poly.sim ← rasterToPolygons(raster.sim)
nb.sim ← poly2nb(poly.sim, queen = T)
mat.nb.sim ← as(nb2mat(nb.sim, style = "B"), "CsparseMatrix")
```

Script 3 Poly2nb Method

```
library(raster); library(sp); library(spdep)

# Block 1 and Block 2 #
nb.cell ← cell2nb(nrow = row, ncol = col, type = "queen")
nb.mat.cell ←as(nb2mat(nb.cell, style = "B"), "CsparseMatrix")
```

Script 4 Cell2nb Method

within the package.

Unfortunately, the 'nb2mat' functionality does not provide the *sparse* option. In order to create a **W** sparse matrix for the matrix output, then we need to define that the matrix output from 'nb2mat' function is constructed on 'CsparseMatrix' form. In this case, we combine the *spdep* package with *Matrix* package.

4. Computational Comparison Between Methods

In this section, we compare all methods—Direct Arrow Reading (DAR), Inner-Outer Neighbor (ION), Kronecker Product (KP), and *spdep* R package—regarding their application using R language program. The measurements are based on their time complexity of the algorithm. Time complexity, denoted by $\mathcal{O}$, is one of computational complexity. Let T is an algorithm system, and let n be an input string for T, thus

Definition 4.1 The *running time*, $\text{time}_T(n)$, represent the number of units of time taken by an algorithm on any input size n (Aho and Ullman, 1995). On the other hand, $\text{space}_T(n)$ is the maximum number, over all inputs of length n, of all memory to which the system writes (Gács and Lovász, 1999).

The function of $\text{time}_T(n)$ and $\text{space}_T(n)$ hold by the assumption that $\text{time}_T(n) \geq n$ and $\text{space}_T(n) \geq 1$ (Knuth, 1973). Hence this condition time complexity function, $\text{time}_T(n)$ write as $\mathcal{O}(n)$. Therefore, algorithm efficiency can be measure under $\mathcal{O}(n)$ value. The algorithm that derives smaller $\mathcal{O}(n)$ value is more time efficient than the other. Based on **Script III.A**, **Script III.B**, and **Script 2** construction, then we can compare the time complexity of those two algorithms on **Table 3**. However, as shown in **Table 3** we cannot compare *spdep* method to the others method. The Kronecker product process as shown in **Table 3** in Block 3 of KP script, we refer the time complexity structure from Temperton (1983).
Based on **Table 3**, time complexity differences can be calculated as follows,

$$\begin{aligned}\Delta\mathcal{O}\big(ION(n) - DAR(n)\big) \\ &= 2n^2 + 11n - \left(8n^2 + n\right) \\ &= -6n^2 + 12n\end{aligned}$$

The negative sign of time differences indicates that ION method (theoretically) has running time faster than DAR method to construct neighbor relation matrix. In order to strengthen the evaluation, then we also calculate the space complexity between them.

On the other hand, if we compare the $\mathcal{O}(KP)$ with DAR and ION method, we will have,

$$\begin{aligned}\Delta\mathcal{O}\big(KP(n) - DAR(n)\big) \\ &= 6n + 4 \cdot n\log n - \left(8n^2 + n\right) \\ &= -8n^2 + 5n + 4 \cdot n\log n\end{aligned}$$

Table 3　Time Complexity Comparison between ION, DAR, and KP Methods

Method		$\mathcal{O}(\boldsymbol{ION})$	**Method**		$\mathcal{O}(\boldsymbol{DAR})$	**Method**		$\mathcal{O}(\boldsymbol{KP})$
	Block 1	$(n-2)^2$		Block 1	$n(n-1)$		Block 1	$4n$
	Block 2	$(n-2)$		Block 2	$n(n-1)$		Block 2	$4n$
	Block 3	$(n-2)$		Block 3	$n(n-1)$		Block 3	$2n$
	Block 4	$(n-2)$		Block 4	$n(n-1)$	KP	Block 4	$2n$
	Block 5	$(n-2)$	DAR	Block 5	$(n-1)^2$		Block 5	$2n$
	Block 6	n		Block 6	$(n-1)^2$		Block 6	$2n$
ION	Block 7	$(n-2)^2$		Block 7	$(n-1)^2$		Block 7*	$4 \cdot n \log n$
	Block 8	$(n-2)$		Block 8	$(n-1)^2$	TOTAL		$6n + 4 \cdot n \log n$*
	Block 9	$(n-2)$		Block 9	n			
	Block 10	$(n-2)$	TOTAL		$8n^2 + n$			
	Block 11	$(n-2)$						
	Block 12	n						
	Block 13	n						
TOTAL		$2n^2 + 11n$						

Note: *) The time complexity is referred to Cooley and Turkey (1965).

and

$$
\begin{aligned}
&\Delta\mathcal{O}\big(KP(n) - ION(n)\big)\\
&\quad = 6n + 4 \cdot n \log n - \left(2n^2 + 11n\right)\\
&\quad = -2n^2 - 5n + 4 \cdot n \log n
\end{aligned}
$$

Both of those results shown us that,

Claim. Kronecker product method will derive faster execution process compared to ION and DAR methods.

In order to confirm this **Claim**, later we compute the actual elapsed time among all methods (including ‘poly2nb’, ‘cell2nb’ and *matrixcalc* methods). However, before we proceed to our actual computational comparison, we also theoretically compared the space complexity among methods. In this comparison we also cannot compare ‘poly2nb’, ‘cell2nb’, and *matrixcalc* to DAR and ION methods, because we do not have enough immediate information regarding the algorithm source for each function within the packages.

As shown in **Table 4** above, we also cannot gather enough information for algorithm source of Kronecker functionality inside R program. Even though we are looking for inside Cooley and Turkey (1965), and Temperton (1983), we cannot immediately obtain the measurement of space complexity for FFT based algorithm process. However, based on **Table 4** we can conclude that there are no significant differences in space between DAR and ION method. The Δ value between those methods only $9n$, where DAR has less memory space.

In this paper, we also did the actual computational comparison among the existing methods. Our computations are executed by each block within each method’s Script and calculated by using R Core utility function called ‘system.time’ for elapsed time and ‘object.size’. ‘system.time’ function calculated how long CPU times need to return each R expression execute (R v.3.4.0

Table 4 Space Complexity Comparison between ION, DAR, and KP Methods

Method		$S(\boldsymbol{ION})$	Method		$S(\boldsymbol{DAR})$	Method		$S(\boldsymbol{KP})$
	Block 1	$4(n-2)^2$		Block 1	$n(n-1)$		Block 1	$4n^2$
	Block 2	$3\cdot(n-2)$		Block 2	$n(n-1)$		Block 2	$4n^2$
	Block 3	$3\cdot(n-2)$		Block 3	$n(n-1)$		Block 3	$2n$
	Block 4	$3\cdot(n-2)$		Block 4	$n(n-1)$	KP	Block 4	$2n$
	Block 5	$3\cdot(n-2)$	DAR	Block 5	$(n-1)^2$		Block 5	$2n$
	Block 6	n		Block 6	$(n-1)^2$		Block 6	$2n$
ION	Block 7	$4\cdot(n-2)^2$		Block 7	$(n-1)^2$		Block 7	*unknown*
	Block 8	$2\cdot(n-2)$		Block 8	$(n-1)^2$	TOTAL		*unknown*
	Block 9	$2\cdot(n-2)$		Block 9	n			
	Block 10	$2\cdot(n-2)$						
	Block 11	$2\cdot(n-2)$	TOTAL		$8n^2+n$			
	Block 12	n						
	Block 13	n						
TOTAL		$8n^2+10n$						

Documentation, 2017). System.time describes as follows,

$$system.time\left(expr,\ gcFirst = \text{TRUE}\right)$$

where, *expr* is a valid R expression to be timed and *gcFirst* is a logical expression that allows garbage collection to be performed immediately. All computational simulations are conducted by using following computational environment on **Table 5**.

Table 5 Computational Environments for Simulations

Computational Environment		
Processor		Intel core i7-6700K, 4 GHz, 8 MB
Memory		DDR4-2133 64 GB (16 GB x 4 slots)
R version	Main Body	3.4.0 (64-bit)
	Library (*Matrix*)	1.2-8
	Library (*sp*)	1.2-4
	Library (*raster*)	2.5-8
	Library (*matrixcalc*)	1.0-3

The simulation procedure is conducted by comparing both method in term of time elapse for its algorithm executed for every block on **Script 1-Script 4**, **Script III.A**, and **Script III.B**. The comparison evaluated based on time complexity of algorithms, which is represented by elapsed time for each blocks of algorithm executed. These evaluations are simulated into several series of spatial lattice matrices $\mathbf{D}_{r\times c}$ as follows,

$$\mathbf{D}_{r\times c} = \mathbf{D}_{3\times 3};\mathbf{D}_{3(10s)\times 3(10s)} \text{ for } s = \left\{1,2,\ldots,11\right\}$$

where each matrix dimension will derive $\mathbf{W}_{Z\times Z} = \mathbf{W}_{(r\cdot c)\times(r\cdot c)}$ given an *s*-series of **D** matrix of simulation. Matrix **D** series for simulation begin $\mathbf{D}_0$ with dimension *r*=3 and *c*=3 for construction **W** matrix from **Figure 5**. This simulation repeated 4 times for each **D** matrix and then calculate the average for those repetitions.

However, in this research, we also build the matrix simulation for $\mathbf{W}_{Z\times Z}$ in form of *Sparse*

Matrix using R language. Sparse matrix used in order to avoid overflow. Overflow is occurred when the memory space that is needed to construct any object is surpassed memory limit in the system. For example, if we input an integer value as an element value of a matrix, then we will have 208 bytes memory object, where an integer object itself consume 48 bytes memory[3) in R language. On the other hand, sparse matrix object generally built to under construction that zero entries need not to be represented, which has impact to decrease amounts of memory (Knuth, 1973: 299) by using Boolean or logical value, instead of integer value. This condition will derive an advantage of sparse matrix object.

We illustrate the advantages of using sparse matrix method instead of the regular matrix by comparing space requirement to build the adjacency matrix, $\mathbf{W}_{Z\times Z}$, for every spatial lattice matrices, $\mathbf{D}_{r\times c}$, on equation (21). In R language program we calculate space requirement by using 'object.size' function. This function provides an estimate of each memory that is being used to store an R object (R Core, 2017). The results shown in **Table 6** below.

In order to build Sparse Matrix by R language, we use 'sparseMatrix' function on *Matrix* package, especially build as 'dgCMatrix' class. dgCMatrix class is a class of sparse numeric matrices in the compressed, sparse, column-oriented format (Maechler and Bates, 2006). The function we use as follows,

$$\begin{aligned} WP.t &\leftarrow as\big(sparseMatrix\big(i,j,index1 \\ &\qquad = \text{FALSE}\big),\text{"dgCMatrix"}\big) \\ WP &\leftarrow WP.t\big[1:Z,1:Z\big] \end{aligned} \tag{27}$$

WP.t is the base sparse matrix which is built by the function. However, hence the sparse matrix is prepared before we conduct neighbor searching process, then we prepare all-zero values for each element on HP. "Index1" option valued as FALSE give us 0-based index factors for the matrix and row-column index counting are also start from 0. This condition implies that HP.t dimension will become $\mathbf{W}_{(Z+1)\times(Z+1)}$. Therefore, we define WP dimension on equation

Table 6 Object Size Comparison of Regular Matrix and Sparse Matrix in R

Simulation Model	Spatial Lattice Matrix ($\mathbf{D}_{r\times c}$)	Datasize (Z)	Object Size	
			Regular Method	Sparse Matrix
1	$\mathbf{D}_{3\times3}$	9	848 Bytes	1432 Bytes
2	$\mathbf{D}_{30\times30}$	900	6.48 MB	1544 Bytes
3	$\mathbf{D}_{60\times60}$	3600	103.68 MB	1664 Bytes
4	$\mathbf{D}_{90\times90}$	8100	524.88 MB	1784 Bytes
5	$\mathbf{D}_{120\times120}$	14400	1658.88 MB	1904 Bytes
6	$\mathbf{D}_{150\times150}$	22500	4.05 GB	2024 Bytes
7	$\mathbf{D}_{180\times180}$	32400	8.398 GB	2144 Bytes
8	$\mathbf{D}_{210\times210}$	44100	15.559 GB	2264 Bytes
9	$\mathbf{D}_{240\times240}$	57600	"Integer Overflow"	2384 Bytes
10	$\mathbf{D}_{270\times270}$	72900	"Integer Overflow"	2504 Bytes
11	$\mathbf{D}_{300\times300}$	90000	"Integer Overflow"	2624 Bytes
12	$\mathbf{D}_{330\times330}$	108900	"Integer Overflow"	2744 Bytes

(27) as $\mathbf{W}_{Z\times Z}$.

The regular matrix method in this computation is applied to DAR method as a benchmark. Therefore, as shown in **Table 6**, by using sparse matrix methods we can conserve a lot amount of memory and compatible with a basic method to analysis huge dimension spatial lattice matrix. By using R Core program only, regular matrix methods cannot handle $\mathbf{D}_{240\times240}$. Based on this condition, then we run the next simulations by using sparse matrix method to construct the neighbor relation matrix for all methods.

As shown in **Table 7**, we have an actual object size comparison for all methods in bytes. Based on its, we have ION and DAR methods, which are a naïve approach has two least space allocated on the objects created during the algorithm executed. On the other hand, 'poly2nb' and 'cell2nb' (*spdep* packages) consume the most memory during the **W** matrix construction. Even though the Kronecker product methods do not provide the best result, but both of them stored relatively small memory size compared to both *spdep* approaches. Especially, our generic function for shift matrix is stored almost similar memory size to the best approach, DAR method.

The reason why the two *spdep* methods (especially 'poly2nb' method) are consumed a lot of space because during the construction process, which is neighborhood search process (explained in **Figure 11a**) is not based on matrix sparse class, but in form of list class. Furthermore, in the original function of 'poly2nb' in *spdep* function, there are not methods to convert list class on neighbor list class (the result of 'poly2nb' function) into **W** sparse matrix class. However, there is the tricky part to convert the list into **W** sparse matrix class, by using a code 'as' and 'CsparseMatrix'.

The original steps to construct spatial neighbor matrix based on *spdep* guidance as follows,

$$\text{polygon} \xrightarrow{\text{'poly2nb'}} \text{neighbor list} \xrightarrow{\text{'nb2mat'}} \mathbf{W}\,\text{matrix}$$

Based on those original steps, then we will have matrix class instead of sparse matrix class object. Therefore, we combine *spdep* with Matrix package as follows,

Table 7 Actual Memory Size Comparison between All Methods (in MB)

Image Matrix	Generic Function	Matrixcalc	cell2nb	poly2nb	ION	DAR
1	0.011	0.015	0.006	0.036	0.003	0.002
2	0.123	0.156	0.312	2.940	0.088	0.087
3	0.466	0.587	1.247	11.748	0.353	0.353
4	1.040	1.306	2.809	26.432	0.799	0.799
5	1.845	2.313	4.997	46.991	1.425	1.424
6	2.879	3.608	7.812	73.425	2.230	2.230
7	4.144	5.1904	11.252	105.734	3.216	3.216
8	5.640	7.061	15.320	143.918	4.382	4.381
9	7.365	9.219	20.013	187.978	5.727	5.727
10	9.322	11.667	25.333	237.913	7.253	7.253
11	11.508	14.402	31.280	293.723	8.959	8.958
12	13.925	17.424	37.853	355.408	10.845	10.844

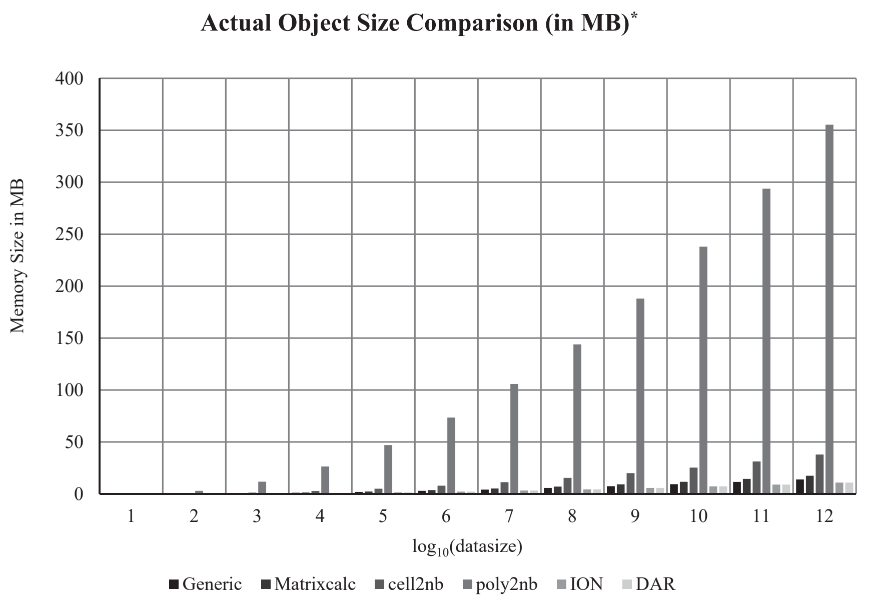
$$\begin{tikzcd} \text{polygon} \arrow[r, "\text{‘poly2nb’}"] & \text{neighbor list} \arrow[r, shift left=3, "\text{‘nb2mat’}"] \arrow[r, phantom, "+"] \arrow[r, shift right=3, "\textit{as “CsparseMatrix”}"'] & \mathbf{W}\text{ matrix} \end{tikzcd}$$

However, even though we applied this steps to our analysis, we found that ‘poly2nb’ method requires space during construction of **W** matrix most. Our results in **Table 7** are provided in sparse matrix construction, rather than the original one. **Figure 12** provides graphics comparison of memory size between all methods.

On the other hand, **Figure 13** provides a graphical comparison for top 4 least memory requirement methods. As we found that in **Table 7** and **Figure 12** that *spdep* methods require the most memory, compared to the other methods. The differences between those two methods to the others are too big. Therefore, we separate them in **Figure 13** to get a clearer picture of comparison to the other methods.

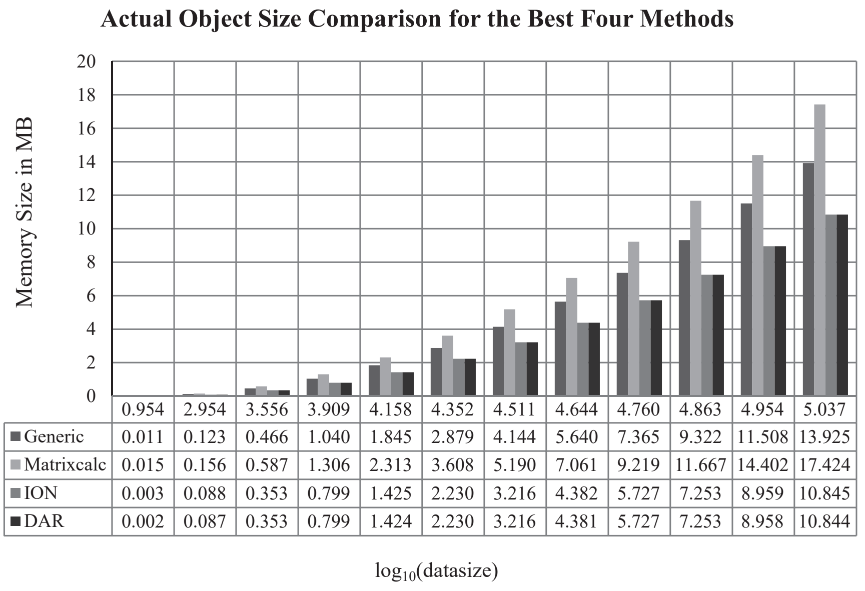
The second efficiency we measured is time efficiency. In this research, we simulate all methods using 12 type of size of image matrix, then measured it by using ‘system.time’ functionality in R program. The efficiency is measured in real-time regarding time lapsed during each method’s algorithm executed.

Elapsed time to execute each block or segment calculated by using R Core utility function called ‘system.time’. Each segment and blocks timed as separated expressions, except “Segment 6” and “Segment 12”, which are separated by (;) sign on each segment. All actual elapsed time record for both scripts are compiled together in **Table 8** below.



Note: *) Each simulation model results are represented on each partition vertical grid, which are ordered by legend order

Figure 12 Actual Memory Size Comparison of All Methods



	0.954	2.954	3.556	3.909	4.158	4.352	4.511	4.644	4.760	4.863	4.954	5.037
■ Generic	0.011	0.123	0.466	1.040	1.845	2.879	4.144	5.640	7.365	9.322	11.508	13.925
■ Matrixcalc	0.015	0.156	0.587	1.306	2.313	3.608	5.190	7.061	9.219	11.667	14.402	17.424
■ ION	0.003	0.088	0.353	0.799	1.425	2.230	3.216	4.382	5.727	7.253	8.959	10.845
■ DAR	0.002	0.087	0.353	0.799	1.424	2.230	3.216	4.381	5.727	7.253	8.958	10.844

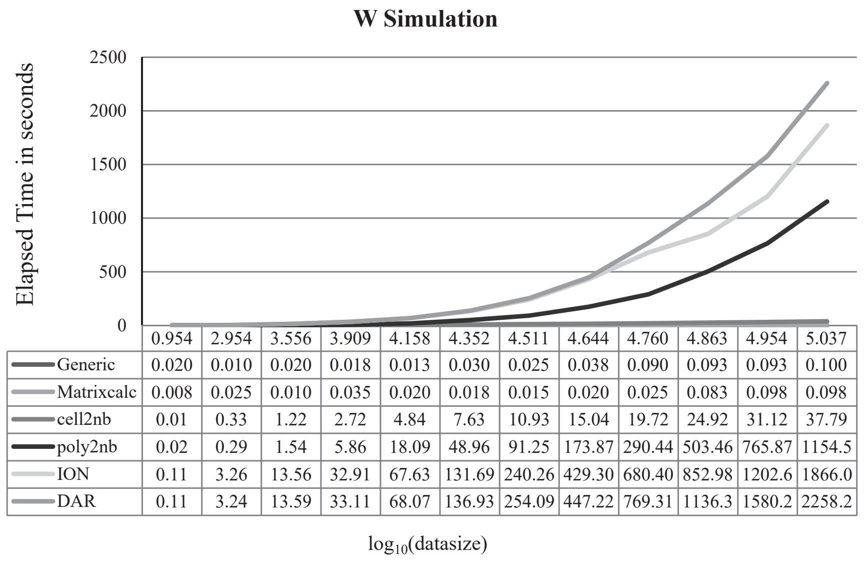
Figure 13 Actual Memory Size Comparison of Top Four Least Memory Methods

Table 8 Total Elapsed Time Comparison for All Methods

Spatial Lattice Matrix ($\mathbf{D}_{r \times c}$)	Datasize (Z)	Elapsed Time in seconds					
		KP1*	KP2**	cell2nb	poly2nb	ION	DAR
$\mathbf{D}_{3\times3}$	9	0.020	0.008	0.01	0.02	0.108	0.103
$\mathbf{D}_{30\times30}$	900	0.010	0.025	0.33	0.29	3.255	3.23
$\mathbf{D}_{60\times60}$	3600	0.020	0.010	1.22	1.54	13.560	13.58
$\mathbf{D}_{90\times90}$	8100	0.018	0.035	2.72	5.86	32.910	33.02
$\mathbf{D}_{120\times120}$	14400	0.013	0.020	4.84	18.09	67.628	67.55
$\mathbf{D}_{150\times150}$	22500	0.030	0.018	7.63	48.96	131.688	143.58
$\mathbf{D}_{180\times180}$	32400	0.025	0.015	10.93	91.25	240.263	253.87
$\mathbf{D}_{210\times210}$	44100	0.038	0.020	15.04	173.87	429.298	438.08
$\mathbf{D}_{240\times240}$	57600	0.090	0.025	19.72	290.44	680.398	757.51
$\mathbf{D}_{270\times270}$	72900	0.093	0.083	24.92	503.46	852.975	1143.28
$\mathbf{D}_{300\times300}$	90000	0.093	0.098	31.12	765.87	1202.630	1725.88
$\mathbf{D}_{330\times330}$	108900	0.100	0.098	37.79	1154.57	1866.040	2055.15

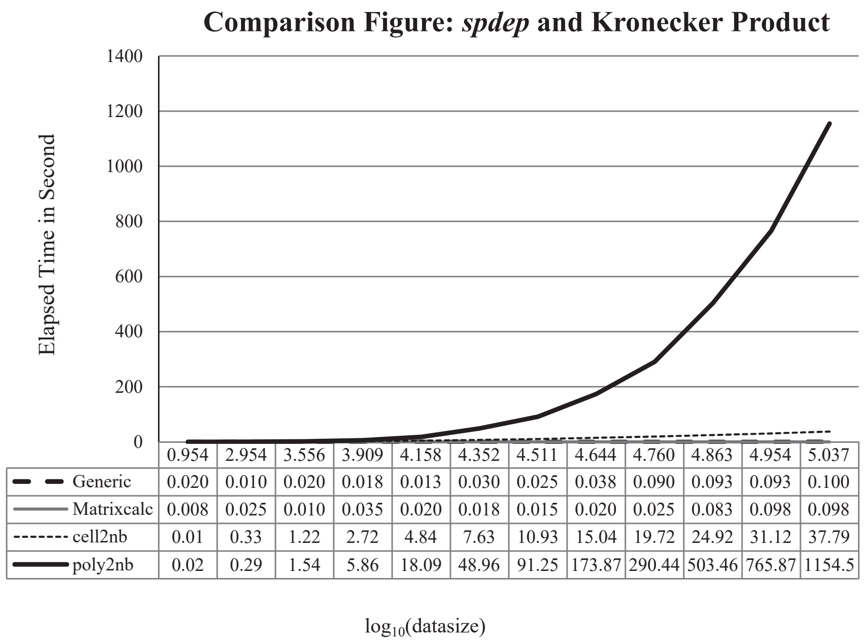
Note: *) **KP1** method represents Generic Function for shift matrix.
) **KP2 method represents *matrixcalc* function for shift matrix.

Furthermore, the result in **Table 8**, **Figure 14** and **Figure 15** indicates that those methods we mention before are not the optimal methods for construction of **W** matrix. We found that both methods of Kronecker product by applying shift matrix provide the best result. This conclusion is derived from that both methods produce the minimum elapsed time and relatively less memory space (our generic function to generate shift matrix derive not many differences with the least



	0.954	2.954	3.556	3.909	4.158	4.352	4.511	4.644	4.760	4.863	4.954	5.037
Generic	0.020	0.010	0.020	0.018	0.013	0.030	0.025	0.038	0.090	0.093	0.093	0.100
Matrixcalc	0.008	0.025	0.010	0.035	0.020	0.018	0.015	0.020	0.025	0.083	0.098	0.098
cell2nb	0.01	0.33	1.22	2.72	4.84	7.63	10.93	15.04	19.72	24.92	31.12	37.79
poly2nb	0.02	0.29	1.54	5.86	18.09	48.96	91.25	173.87	290.44	503.46	765.87	1154.5
ION	0.11	3.26	13.56	32.91	67.63	131.69	240.26	429.30	680.40	852.98	1202.6	1866.0
DAR	0.11	3.24	13.59	33.11	68.07	136.93	254.09	447.22	769.31	1136.3	1580.2	2258.2

Figure 14 Elapsed Time Comparison between *spdep* Packages and Kronecker Product



	0.954	2.954	3.556	3.909	4.158	4.352	4.511	4.644	4.760	4.863	4.954	5.037
Generic	0.020	0.010	0.020	0.018	0.013	0.030	0.025	0.038	0.090	0.093	0.093	0.100
Matrixcalc	0.008	0.025	0.010	0.035	0.020	0.018	0.015	0.020	0.025	0.083	0.098	0.098
cell2nb	0.01	0.33	1.22	2.72	4.84	7.63	10.93	15.04	19.72	24.92	31.12	37.79
poly2nb	0.02	0.29	1.54	5.86	18.09	48.96	91.25	173.87	290.44	503.46	765.87	1154.5

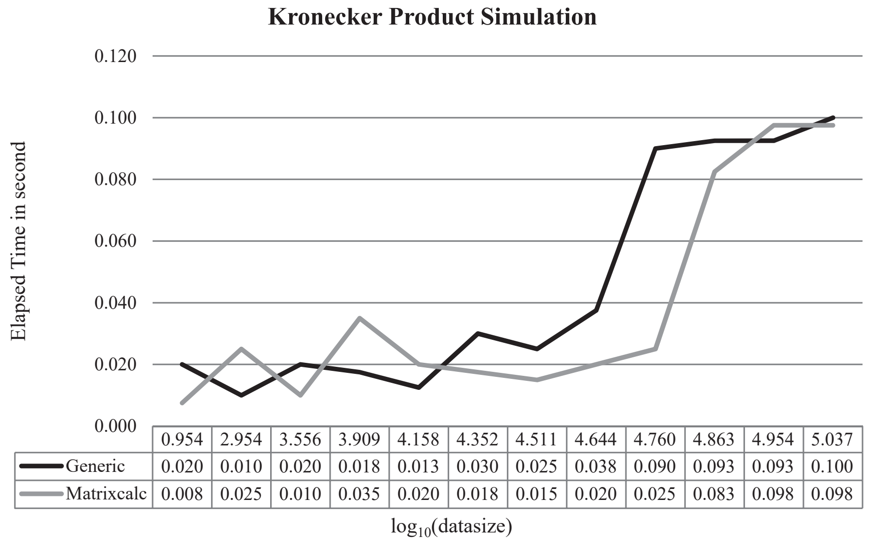
Figure 15 Elapsed Time Comparison of Top Four Fastest Algorithm

space method, DAR).

Based on those result in **Table 8** and **Figure 14** we can confirm that our actual elapsed time measurement is supporting our theoretical **Claim** beforehand. The Kronecker product method derived the faster elapsed time to construct **W** matrix.

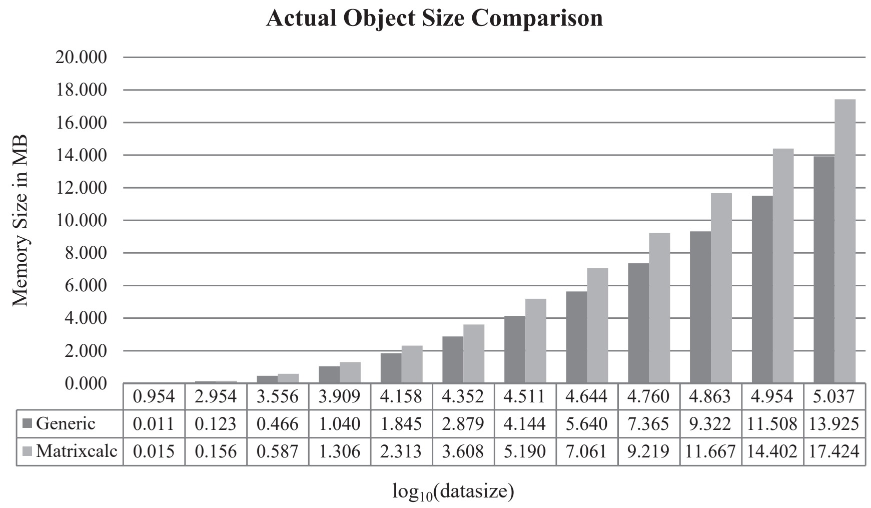
Our next highlight in this section is the elapsed time comparison. The Kronecker product can generate more efficient elapsed time compares to both *spdep* package methods. As we discussed earlier, that in R language program the only practical methods, which is can be found in many handbooks of application of spatial analysis, to construct **W** matrix is using *spdep* packages. However, in this paper we provide better methods that can be used as different approaches.

Therefore, the last procedure to determine the best methods to construct **W** matrix is compared both methods of Kronecker product. Based on **Figure 16a** and **Figure 16b** we can compare the actual object size and elapsed time for each method for Kronecker Product construction of **W** matrix. Basically, both of them consumed quite similar elapsed time and memory consumption. However, by using our generic function to construct shift matrix we can have less memory object compared to *matrixcalc* functionality (**Figure 16b**). By those results we can conclude that Kronecker product based on our generic function for shift matrix derived the best result among the other methods.



	0.954	2.954	3.556	3.909	4.158	4.352	4.511	4.644	4.760	4.863	4.954	5.037
━ Generic	0.020	0.010	0.020	0.018	0.013	0.030	0.025	0.038	0.090	0.093	0.093	0.100
━ Matrixcalc	0.008	0.025	0.010	0.035	0.020	0.018	0.015	0.020	0.025	0.083	0.098	0.098

Figure 16a Elapsed Time Comparison of Kronecker Product Simulation



	0.954	2.954	3.556	3.909	4.158	4.352	4.511	4.644	4.760	4.863	4.954	5.037
■ Generic	0.011	0.123	0.466	1.040	1.845	2.879	4.144	5.640	7.365	9.322	11.508	13.925
■ Matrixcalc	0.015	0.156	0.587	1.306	2.313	3.608	5.190	7.061	9.219	11.667	14.402	17.424

Figure 16b Elapsed Time Comparison of Kronecker Product Simulation

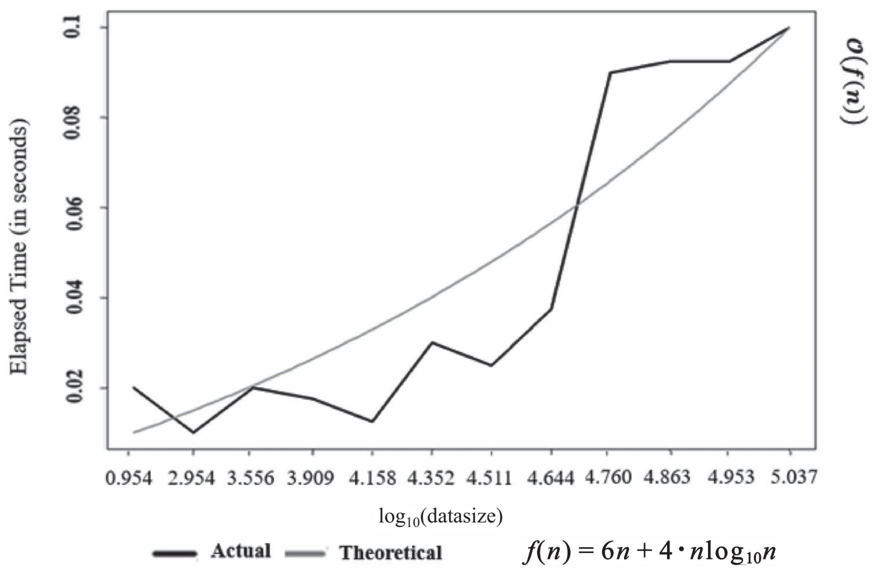


Figure 17 Theoretical and Actual Simulation Overlay of Kronecker's Time Complexity

For a detailed comparison of the best method, Kronecker Product, into our theoretical time complexity in **Table 3**, we provide comparison figures in **Figure 17**. Theoretical approximation of time complexity, $\mathcal{O}(f(n))$, shows similar trend for Kronecker product's methods. As shown in **Figure 17**, the figure shows increasing trend for larger lattice structure on **D**, which also showed on the theoretical curve. Unfortunately, we cannot provide overlay comparison for space complexity, hence—as shown in **Table 4** (Block 7 of Kronecker product method)—we cannot gather immediate information of space complexity for Kronecker product process.

5. Concluding Remarks

Very few precedent researches were found for the development of the spatial **W** matrix so far. Furthermore, we also found the fact that there is no precedent research, which provides a technical and application for the formulation of spatial **W** matrix for large lattice structure data. Under this condition, we need the optimal method that is efficient in term of space memory and elapsed time consumption. This paper thoroughly compared the methods in term of computer CPU and memory resources. We found that Kronecker product method by using shift matrix structure gave us the best result compared to the other methods.

Acknowledgment

The authors would like to thank Prof. Koichi Maekawa of Hiroshima University of Economics and Prof. Ryuei Nishii at Kyushu University for their continuous supports during the writing process of this paper. This work was also supported by JSPS KAKENHI Grant Number JP17K400064.

Notes

1) Data source United States Geological Survey (USGS).
2) If we are using term on equation (22) we also can replace any $\mathbf{I}_n$ into zero-shifted matrix, $\mathbf{L}_{n,0}$ or $\mathbf{U}_{n,0}$.
3) The value is defined by computing one integer value space in R by using 'object.size' function of R. On the other hand, we need 208 bytes to store one integer value on 1×1 dimension regular matrix. Even though, 1×1 dimension sparse matrix need 1632 bytes to store 1 integer value, but bigger the dimension, than we will have less memory space for sparse matrix object (confirmed by **Table 7**)

References

Aho, Alfred V. and Jeffrey D. Ullman, 1995. Foundations of Computer Science: C Edition, 786 pp.

Aldstald, Jared and Arthur Getis, 2006. "Using AMOEBA to create a spatial weights matrix and identify spatial clusters," *Geographical Analysis*, Vol. 28 (4), pp. 327–343.

Anselin, Luc and Raymond J. G. M. Florax, 1995. *New Directions in Spatial Econometrics*. Springer, 420 pp.

Anselin, Luc and Serge Rey, 2010. "Properties of Tests for Spatial Dependence in Linear Regression Models," *Geographical Analysis*, Vol. 23 (2), pp. 112–131.

Arbia, Giuseppe, 2014. *A Primer for Spatial Econometrics*. Palgrave Texts in Econometrics. Springer, 230 pp.

Bates, Douglas and Martin Maechler. 2017. "Package *Matrix*," R Documentation.

Bivand, Roger S. *et al*., 2008. *Applied Spatial Data Analysis with R (Use R!)*. Springer, 405 pp.

Cooley, James W. and John W. Turkey, 1965. "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, Vol. 19, No. 90, pp. 297–301.

Cressie, Noel A. C., 1991. *Statistics for Spatial Data*. John Willey & Sons, Inc. New York, 928 pp.

Cressie, Noel A. C. and Christopher K. Wikle, 2011. *Statistics for Spatio-Temporal Data*. John Willey & Sons, Inc. New Jersey, 628 pp.

Gács, Peter and László Lovász, 1999. "*Complexity of Algorithms*," Lecture Notes, Spring 1999, 242 pp.

Getis, Arthur and Jared Aldstaldt, 2010. "Constructing the Spatial Weights Matrix Using a Local Statistic," *Geographical Analysis*, Vol. 36 (2), pp. 90–104.

Keleijan, Harry and Gianfranco Piras, 2017. *Spatial Econometrics*. Academic Press, 458 pp.

Knuth, Donald E., 1973. *Fundamental Algorithm: Second Edition*. Addison-Wesley Publishing Company, 672 pp.

Maechler, Martin and Douglas Bates, 2006. "2nd Introduction to the Matrix Package," R Core Development Team. Accessed on: https://stat.ethz.ch/R-manual/R-devel/library/Matrix/doc/Intro2Matrix.pdf

Novomestky, Frederick, 2015. "Package *matrixcalc*," R Documentation.

Tanaka, Shojiro and Ryuei Nishii, 2009. "Nonlinear Regression Models to Identify Functional Forms of Deforestation in East Asia," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 47 (8), pp. 2617–2626.

Temperton, Clive, 1983. "Self-sorting mixed-radix fast fourier transforms," *Journal of Computational Physics*, 52 (1), pp. 1–23.

Wedderburn, Joseph, 1934. *Lectures on Matrices*. American Mathematical Society: Colloquium Publication, Vol. 17. Rhode Island, 205 pp.

APPENDIX I.A. LIST OF PRECEDENT RESEARCHES

Table I.A Table of Precedent Researches

Source: Google Scholar
Keyword: allintitle: ("spatial * matrix")
Hit: 180 (without patents and citations)
Retrieval Date: 16–20 August 2017

No.	Title	Author(s)	Journal	Date of Publication
1	Constructing the Spatial Weights Matrix Using a Local Statistic	Getis, Arthur and Jared Aldstadt	*Perspectives on spatial data analysis*	2010
2	Using AMOEBA to Create a Spatial Weights Matrix and Identify Spatial Clusters	Aldstald and Getis	*Geographical Analysis*	2006
3	Spatial weights matrix	Zhou and Lin	*Encyclopedia of GIS*	2008
4	Matrix fitting approach to direction of arrival estimation with imperfect spatial coherence of wavefronts	Gershman *et al*	*IEEE Transactions on Signal Processing*	1997
5	Measuring the neighboring and environmental effects on residential property value: Using spatial weighting matrix	Hui *et al*	*Building and Environment*	2007
6	Minimal realization of a spatial stiffness matrix with simple springs connected in parallel	Roberts, R.G	*IEEE Transactions on Robotics and Automation*	1999
7	Direction of arrival estimation using the parameterized spatial correlation matrix	Dmochowski, Jacek *et al*	*IEEE Transactions on Audio, Speech, and Language Processing*	2007
8	Estimating a spatial autoregressive model with an endogenous spatial weight matrix	Xi Qu and Lung-fei Lee	*Journal of Econometrics*	2015
9	A low complexity algorithm to simulate the spatial covariance matrix for clustered MIMO channel models	Vorenza, A. *et al*	*Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*	2004
10	Estimating direct-to-reverberant energy ratio using D/R spatial correlation matrix model	Hioka, Yusuke *et al*	*IEEE Transactions on Audio, Speech, and Language Processing*	2011
11	Robust uplink to downlink spatial covariance matrix transformation for downlink beamforming	Chalise, B.K *et al*	*2004 IEEE International Conference on Communications*	2004
12	Estimation of spatial weights matrix in a spatial error model, with an application to diffusion in housing demand	Bhattarcharjee, Arnab and Chris Jensen-Butler	n.a	n.a
13	Nonparametric Estimation of the Spatial Connectivity Matrix Using Spatial Panel Data	Beenstock and Felsenstein	*Geographical Analysis*	2012
14	Estimation of Spatial Weights Matrix, with an Application to Diffusion in Housing Demand	Bhattarcharjee, Arnab and Chris Jensen-Butler	*Centre for Research into Industry, Enterprise, Finance and the Firm, (CRIEFF Discussion Papers)*	2006
15	Estimation of the spatial weights matrix under structural constraints	Bhattarcharjee, Arnab and Chris Jensen-Butler	*Regional Science and Urban Economics*	2013

16	Computation of the information matrix for models with spatial interaction on a lattice	Smirnov, Oleg A.	*Journal of Computational and Graphical Statistics*	2004
17	High-spatial resolution matrix-assisted laser desorption ionization imaging analysis of glucosylceramide in spleen sections from a mouse model of Gaucher disease	Snel and Fuller	*Analytical Chemistry*	2010
18	The spatial autocorrelation matrix	Chessel, D.	*Vegetation dynamics in grasslans, healthlands and mediterranean ligneous formations*	1981
19	Minimal realization of an arbitrary spatial stiffness matrix with a parallel connection of simple and complex springs	Roberts, R.G	*IEEE Transactions on Robotics and Automation*	2000
20	Spatial regression models in criminology: Modeling social processes in the spatial weights matrix	Tita and Radil	*Handbook of quantitative criminology*	2010
21	Model boosting for spatial weighting matrix selection in spatial lag models	Kostov, Phillip	*Environment and Planning B: Urban Analytics and City Science*	2010
22	Direction of arrival estimation using eigenanalysis of the parameterized spatial correlation matrix	Dmochowski, Jacek *et al*	*2007 IEEE International Conference on Acoustics, Speech and Signal Processing*	2007
23	The spatial stiffness matrix from simple stretched springs	Sellig, J.M.	*Robotics and Automation, 2000. Proceedings*	2000
24	On the Four Types of Weight Functions for Spatial Contiguity Matrix	Yanguang Chen	*Letters in Spatial and Resource Sciences*	2012
25	A Dynamic Spatial Weight Matrix and Localised STARIMA for Network Modelling	Tao Cheng *et al*	*Geographical Analysis*	2014
26	Comparative analysis of spatial covariance matrix estimation methods in OFDM communication systems	Maltsev, Alexander *et al*	*2006 IEEE International Symposium on Signal Processing and Information Technology*	2006
27	Automatic selection of a spatial weight matrix in spatial econometrics: Application to a spatial hedonic approach	Seya, Hajime *et al*	*Regional Science and Urban Economics*	2013
28	Spatial Nonstationarity and Spurious Regression: The Case with Row-Normalized Spatial Weights Matrix	Lung-fei Lee and Jihai Yu	*Spatial Economic Analysis*	2009
29	Analyzing the Effect of Spatial Weighted Matrix On Spatial Autocorrelation——Taking Hunan's Income Gap between Urban and Rural Areas as A Case	Hongliang, Wang *et al*	*Journal of South China Normal University (Natural Science Edition)*	2010
30	Bounds on eigenvalues of a spatial correlation matrix	Choi and Love	*IEEE Communications Letters*	2014
31	Structure of the spatial stiffness matrix	Sellig and Ding	*Structure of the spatial stiffness matrix. International Journal of Robotics and Automation*	2002
32	On the normal form of a spatial stiffness matrix	Roberts, R.G	*Proceedings 2002 IEEE International Conference on Robotics and Automation*	2002

33	Measuring quantum states: Experimental setup for measuring the spatial density matrix	Tegmark, M.	*Physical Review A*	1996
34	Bayesians in Space: Using Bayesian Methods to Inform Choice of Spatial Weights Matrix in Hedonic Property Analyses	Mueller, Julie M. and John B. Loomis	*The Review of Regional Studies*	2010
35	Note on the normal form of a spatial stiffness matrix	Roberts, R.G	*IEEE Transactions on Robotics and Automation*	2001
36	Two-step lasso estimation of the spatial weights matrix	Ahrens, Achim and Arnab Bhattacharjee	*Econometrics*	2015
37	Detection and estimation of block structure in spatial weight matrix	Lam, Clifford and Pedro C.L. Souza	*Econometrics Review*	2016
38	Temporal convergence of phase spatial covariance matrix measurements in tomographic adaptive optics	Martin, O. *et al*	*Adaptive Optics Systems III. Proceedings of the SPIE*	2012
39	A matrix exponential spatial specification	LeSage and Pace	*Journal of Econometrics*	2007
40	DOA Estimation in Impulsive Noise Environments Using Fractional Lower Order Spatial-Temporal Matrix [J]	Jin and Zhong	*Acta Aeronautica Et Astronautica Sinica*	2006
41	Analysis and application on the specification methods of the spatial weight matrix	Liu and Wang	*Geo-information Science*	2002
42	Uplink to downlink spatial covariance matrix transformation concepts for downlink beamforming	Chalise, B.K *et al*	*Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology*	2003
43	A Proposal for an Alternative Spatial Weight Matrix under Consideration of the Distribution of Economic Activity	Perret, Jens K.	*Schumpeter Discussion Paper*	2011
44	On the eigenstructure of the signal-only tempo-spatial covariance matrix of broadband sources using a circular array	Messer and Rockah	*IEEE Transactions on Acoustics, Speech, and Signal Processing*	1990
45	The generalization of narrowband localization methods to broadband environments via parametrization of the spatial correlation matrix	Dmochowski, Jacek *et al*	*Signal Processing Conference, 2007 15th European*	2007
46	Computation of the information matrix for models of spatial interaction	Smirnov, Oleg A.	n.a	2003
47	Enhanced spatial covariance matrix estimation for asynchronous inter-cell interference mitigation in MIMO-OFDMA system	Jung Su Han *et al*	*VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*	2009
48	Modelling a large, sparse spatial interaction matrix using data relating to a subset of possible flows	Bailey and Munford	*European journal of operational research*	1994
49	QML Estimation of the Spatial Weight Matrix in the MR-SAR Model	Benjanuvatra, Saruta	*VII World Conference of the Spatial Econometrics Association*	2013
50	The constitution and realization of spatial weight matrix based on ArcObjects [J]	Pan Hai-Yan *et al*	*Science of Surveying and Mapping*	2007

51	Choosing the right spatial weighting matrix in a quantile regression model	Kostov, Phillip	*ISRN Economics*	2013
52	Spatial Weighting Matrix Selection in Spatial Lag Econometric Model	Kostov, Phillip	*Econometrics*	2013
53	Spatial neighborhood matrix computation: inverse distance weighted versus binary contiguity	Negreiros, J	n.a	2009
54	Spatial heteroskedasticity and autocorrelation consistent estimation of covariance matrix	Kim and Sun	*Journal of Econometrics*	2011
55	Optimization of spatial filter matrix [J]	Zhou Hai *et al*	*Optics and Precision Engineering*	2007
56	On the spatial density matrix for the centre of mass of a one-dimensional perfect gas	Carazza, B.	*Foundations of Physics Letters*	1997
57	Quantitative Submonolayer Spatial Mapping of Arg-Gly-Asp-Containing Peptide Organomercaptan Gradients on Gold with Matrix-Assisted Laser Desorption/Ionization Mass Spectrometry	Qian Wang *et al*	*Analytical Chemistry*	2004
58	New spatial weight matrix and its application in China's regional foreign trade [J]	Zhang Jia-Wei *et al*	*Systems Engineering-Theory & Practice*	2009
59	Simulation of the spatial covariance matrix	Forenza, A. *et al*	*Simulation*	2003
60	Stata Implementation of the non-parametric spatial heteroskedasticity and autocorrelation consistent covariance matrix estimator	Jeanty, P.W.	*Stata conference San Diego*	2012
61	A robust algorithm based on spatial differencing matrix for source number detection and DOA estimation in multipath environment	Liu *et al*	*Physics Procedia*	2012
62	Investigation of basic imaging properties in digital radiography. 12. Effect of matrix configuration on spatial resolution	Fujita and Giger	*Medical Physics*	1988
63	Efficient multichannel nonnegative matrix factorization exploiting rank-1 spatial model	Kitamura, Daichi *et al*	*2015 IEEE International Conference on Acoustics, Speech and Signal Processing*	2015
64	One% Step Regularized Spatial Weight Matrix and Fixed Effects Estimation with Instrumental Variables	Lam and Souza	n.a	2015
65	Iterative receiver with enhanced spatial covariance matrix estimation in asynchronous interference environment for 3GPP LTE MIMO-OFDMA system	Jun-Hee Jang *et al*	*IEICE TRANSACTIONS on Communications*	2009
66	Model uncertainty in matrix exponential spatial growth regression models	Pribauer and Fischer	*Geographical Analysis*	2015
67	Model selection using J-test for the spatial autoregressive model vs. the matrix exponential spatial model	Han and Lee	*Regional Science and Urban Economics*	2013
68	Using matrix exponentials to explore spatial structure in regression relationships	LeSage and Pace	*researchgate*	2002

69	Momentum density and spatial form of correlated density matrix in model two-electron atoms with harmonic confinement	Akbari, A. *et al*	*Physical Review A*	2007
70	The spatial transfer matrix of curved box-girder bridge	Yan and Li	*Journal of Harbin Engineering University*	2014
71	A novel array calibration method based on spatial correlation matrix for HFSWR	Xiagou *et al*	*IEEE 10th International Conference On Signal Processing Proceedings*	2010
72	Synthesizing a positive definite spatial stiffness matrix with a hybrid connection of simple compliances	Roberts, R.G and Shirey	*Proceedings World Automation Congress, 2004*	2004
73	Large sample properties of the matrix exponential spatial specification with an application to FDI	Debarsy, N.	*Journal of Econometrics*	2015
74	Image classification using spatial relationship matrix based on color spatio-histogram	Woosaeng, Kim *et al*	*Proceedings. 2003 International Conference on Multimedia and Expo, 2003. ICME '03.*	2003
75	Determination of the reverberation chamber stirrer uncorrelated positions by means of the spatial and frequency correlation matrix	Gradoni, G. *et al*	*2013 International Symposium on Electromagnetic Compatibility (EMC EUROPE)*	2013
76	Spatial correlation matrix selection using Bayesian model averaging to characterize inter-tree competition in loblolly pine trees	Boone and Bullock	*Journal of Applied Statistics*	2008
77	Taking off some hoods: Estimating spatial models with a non-arbitary W matrix	Fernandez-Vazquez, Esteban *et al*	*Spatial Econometrics Conference*	2007
78	A closer look at the spatial exponential matrix specification	Rodrigues, E *et al*	*Spatial Statistics*	2014
79	Robust spatial time-frequency distribution matrix estimation with application to direction-of-arrival estimation	Syariff, W. *et al*	*Signal Processing*	2011
80	Interference suppression with iterative channel and spatial covariance matrix estimation for LTE downlink	Fan, J *et al*	*Digital Signal Processing*	2014
81	Political interaction in the senate: estimating a political "spatial" weights matrix and an application to lobbying behavior	Chupp, B. A.	*Public Choice*	2014
82	Spatial weighting matrix selection in spatial lag econometric model	Kostov, Phillip	*Econometrics*	2013
83	A Combined Matrix Analysis on The Spatial Dislocation of Landscape Resources, Nameplate Scenery and Finance Achievement in China	Wang Mei-Hong *et al*	*Human Geography*	2009
84	Multichannel audio separation by direction of arrival based spatial covariance model and non-negative matrix factorization	Nikunen and Virtanen	*2014 IEEE International Conference on Acoustics, Speech and Signal Processingv (ICASSP)*	2014

85	Noise robust direction of arrival estimation for speech source with weighted bispectrum spatial correlation matrix	Wei Xue *et al*	*IEEE Journal of Selected Topics in Signal Processing*	2015
86	Multibody mass matrix sensitivity analysis using spatial operators	Jain and Rodriguez	*International Journal for Multiscale Computational Engineering*	2003
87	Evaluating effects of low quality habitats on regional population growth in Peromyscus leucopus: insights from field-parameterized spatial matrix models	Grear and Burns	*Landscape Ecology*	2007
88	A new spatial interpolation algorithm to reduce the matrix fill time in the method of moments analysis of planar microstrip structures	Ogucu, G.O.	*IEEE Transactions on Antennas and Propagation*	2007
89	Estimation of a weights matrix for determining spatial effects	Lima and Macedo	n.a	1999
90	Robustness of the affine equivariant scatter estimator based on the spatial rank covariance matrix	Kai Yu *et al*	*Communications in Statistics - Theory and Methods*	2015
91	Selecting the Most Adequate Spatial Weighting Matrix:A Study on Criteria	Gomez, Marcus Herrera *et al*	*EconPapers*	2012
92	MUSIC algorithm of spatial prefiltering for vector hydrophones in an array matrix [J]	He Yiyinga *et al*	*Journal of Huazhong University of Science and Technology (Natural Science Edition)*	2011
93	Correlative Analysis of Regional Economy Based on Transportation Network Spatial Weight Matrix——Taking Gansu Province as An Example [J]	Ma Qing-Yuan *et al*	*Areal Research and Development*	2007
94	Analysis on Forestland Change by Using Spatial Transition Matrix Model [J]	Zheng Chung-Yang *et al*	*Forest Resources Management*	2012
95	The construction of spatial weight matrix based on discrete points through building a searching area for each point	Wenii Yu *et al*	*2010 International Conference on Computer Application and System Modeling*	2010
96	An Improved Multichannel Spatial Correlation Matrix Based ARMA Model for Short-Term Wind Forecasting	Filik, T.	n.a	n.a
97	Alternative neighbourhood specifications of the spatial weight matrix; effects on spatial autocorrelation index and multivariate analysis of health data	Bertazzon and Elikan	n.a	2009
98	Estimation of sound source orientation using eigenspace of spatial correlation matrix	Kenta Niwa *et al*	*2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*	2010
99	Uplink to downlink spatial covariance matrix transformation methods for downlink beamforming of a UMTS-FDD system	Chalise, B.K *et al*	*Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*	2004
100	Shape of turbulent lump in a circular pipe flow determined by spatial-dependence matrix	Kohei Ogawa *et al*	*Chemical Engineering Communications*	1988

101	Regularizing the covariance matrix using spatial information	Tax, D.M.J	n.a	2004
102	Rotation invariant texture feature based on spatial dependence matrix for timber defect detection	Hashim, Ummi R. *et al*	*Intelligent Systems Design and Applications (ISDA), 2013 13th International Conference on*	2013
103	Nonparametric Estimation of the Spatial Connectivity Matrix by the Method of Moments Using Spatial Panel Data	Beenstock *et al*	n.a	2009
104	Daily activity prediction based on spatial-temporal matrix for ongoing videos	Hsin-Lin Yang *et al*	*Society of Instrument and Control Engineers of Japan (SICE), 2015 54th Annual Conference of the*	2015
105	Blind Suppression of Nonstationary Diffuse Acoustic Noise Based on Spatial Covariance Matrix Decomposition	Nobutaka Ito *et al*	*Journal of Signal Processing Systems*	2015
106	Matrix Powers and Marginal Effects in Spatial Autoregressive Models	Kripfganz, Sebastiaan	*SSSRN*	2015
107	Quantization and feedback of spatial covariance matrix for massive MIMO systems with cascaded precoding	Yinsheng Liu *et al*	*IEEE Transactions on Communications*	2017
108	Bayesian Estimation of A Spatial Autoregressive Model with An Unobserved Endogenous Spatial Weight Matrix and Unobserved Factors$	Xiaoyi Han	*Manuscript, The Ohio State University*	2013
109	Spatial, Temporal, and Matrix Variability of Clostridium botulinum Type E Toxin Gene Distribution at Great Lakes Beaches	Wijesinghe, Rasanthi U.	*Applied and Environmental Microbiology*	2015
110	Discrete choice modeling with interdependencies: A spatial binary Probit model with endogenous weight matrix	Zhou Y.	n.a	2015
111	Testing for a structural break in the weight matrix of the spatial error or spatial lag model	Angulo, Ana *et al*	*Spatial Economic Analysis*	2017
112	Institutions and growth: Testing the spatial effect using weight matrix based on the institutional distance concept	Ahmad and Hall	*mpra.ub.uni-muenchen.de*	2012
113	Handloom Silk Defect Recognition and Categorization using Gray Level Weight Matrix & Multi Resolution Combined Statistical and Spatial Frequency Method	Sabeenian, R.S *et al*	*researchgate*	n.a
114	Spatial wavelet approach to local matrix crack detection in composite beams with ply level material uncertainty	Sarangapani, G. *et al*	*Applied Composite Materials*	2013
115	The Improbable Nature of the Implied Correlation Matrix from Spatial Regression Models	Sen, Monalisa and Anil K. Bera	*Regional Statistics*	2014
116	A matrix exponential spatial specification approach to panel data models	Figueiredo and Da Silva	*Empirical Economics*	2015
117	Interaction matrix selection in spatial econometrics with an application to growth theory	Debarsy and Ertur	n.a	2016

118	A New Selection Method of Spatial Weight Matrix	Ren Yinghua and You Wanhai	*Statistical Research*	2012
119	Comparison of Uniform and Kernel Gaussian Weight Matrix in Generalized Spatial Panel Data Model	Purwaningsih, Tuti *et al*	*Open Journal of Statistics*	2015
120	Estimating the impact of air quality with a spatial hedonic: Geostatistical versus weight matrix approaches	Tandon, S, *et al*	n.a	2007
121	Analyzing the impact of different spatial weight matrix for the western coast of the Taiwan straits	Chen and Huang	*Journal of Shangqiu Normal University*	2016
122	Model for Effect of Spatial Weighted Matrix on Spatial Autocorrelation	Zhilang Wang *et al*	*Jurnal of Applied Mathematics and Statistics*	2013
123	A novel soft spatial weights matrix method based on soft sets	Wang and Xiao	*International Journal of Applied Decision Sciences*	2016
124	An Estimation Method of Sound Source Orientation Using Eigenspace Variation of Spatial Correlation Matrix	Kenta Niwa *et al*	*IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*	2013
125	Spatial Lag Model Estimation with Sparse Adjustment for Spatial Weight Matrix	Lam, Clifford and Pedro C.L. Souza	n.a	n.a
126	Extracting geographical networks from online network resource: Building a spatial neighbourhood matrix of local municipalities using free online encyclopaedia information	Shikano and Meissner	*Annual meeting des Arbeitskreises Handlungs- und Entscheidungstheorie der Deutschen Vereinigung der Politischen Wissenschaft*	2013
127	A bearing fault diagnosis technique based on singular values of EEMD spatial condition matrix and Gath-Geva clustering	Kun Yu *et al*	*Applied Acoustics*	2017
128	Relative Spatial Distance Matrix: a Novel and Invariant Data Structure for Representation and Retrieval of Exact Match Symbolic Images	Punitha, P. *et al*	*Proceedings of the International Conference on Cognition and Recognition*	2005
129	The Influence of the Structure of Spatial Weight Matrix on Regression Analysis in the Presence of Spatial Autocorrelation	Tsutsumi, Morito *et al*	*土木計画学研究・論文集*	2000
130	Dynamic Large Spatial Covariance Matrix Estimation in Application to Semiparametric Model Construction via Variable Clustering: the SCE approach	Song, Song	*arXiv*	2011
131	Data-aided SIMO channel estimation with unknown noise spatial covariance matrix	Xin Meng *et al*	*IEICE Communications Express*	2014
132	A Note on Eigenstructure of a Spatial Design Matrix In R	Kim and Tarazaga	*Communications for Statistical Applications and Methods*	2005
133	Spatial Prominence and Spatial Weights Matrix in Geospatial Analysis	Changping Zhang	*Progress in Geospatial Analysis*	2012
134	Two-Stage Procedure Of Building A Spatial Weight Matrix With The Consideration Of Economic Distance	Pietrzak, M.B.	*Oeconomia Copernicana*	2010

135	The best-approximate realization of a spatial stiffness matrix with simple springs connected in parallel	Jue Yu *et al*	*Mechanism and Machine Theory*	2016
136	Iterative Receiver with Enhanced Spatial Covariance Matrix Estimation in Asynchronous Interference Environment for 3GPP LTE MIMO-OFDMA System	Jang, Jun-Hee *et al*	*PAPER-Wireless Communication Technologies*	2009
137	Spatial stiffness matrix corrected and dynamic analysis of short-leg shear wall	Guo Zeying and Zhougang	*Sichuan Building Science*	2007
138	Spatial Relationships in Rural Land Markets with Emphasis on a Flexible Weights Matrix	Soto, Patricia *et al*	*Paper session of the American Agricultural Economics Association Long Beach, Florida*	2002
139	Transmit waveform synthesis for MIMO radar using spatial-temporal decomposition of correlation matrix	Tao Yang *et al*	*Radar Conference, 2014 IEEE*	2014
140	Robust spatial covariance matrix transformation techniques for downlink beamforming	Chalise, B.K *et al*	*2004 International Zurich Seminar on Communications*	2004
141	Rotation Invariant Texture Feature Based on Spatial Dependence Matrix for Timber Defect Detection	Muda and Raba'ah	n.a	2013
142	Computational architecture for linear n-processor array implementations of composite rigid-body spatial inertial matrix algorithms	Howell, P.D.	*Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation, 1992. Power Electronics and Motion Control*	1992
143	On Misspecification of Spatial Weight Matrix for Small Area Estimation in Longitudinal Analysis	Zadlo, Tomasz	*Comparative Economic Research*	2012
144	Hyperspectral image super-resolution extending: An effective fusion based method without knowing the spatial transformation matrix	Yong Li *et al*	*2017 IEEE International Conference on Multimedia and Expo (ICME)*	2017
145	Evaluating Estimation of Direct-to-Reverberation Energy Ratio using D/R Spatial Correlation Matrix Model	Yusuke, Hioka *et al*	*Proceedings of 20th International Congress on Acoustics*	2010
146	Spatial weights matrix construction and economic space gravitational effects analysis-Empirical testing based on European debt crisis	Li Li *et al*	*Systems Engineering-Theory & Practice*	2015
147	Estimating Non-stationary Spatial Covariance Matrix using Multi-resolution Knots	Nandy, Siddharta *et al*	n.a	n.a
148	¿Cuál matriz de pesos espaciales?. Un enfoque sobre selección de modelos [Which spatial weighting matrix? An approach for model selection]	Herrera Gomez, Marcos *et al*	*mpra.ub.uni-muenchen.de*	2011
149	Model uncertainty in matrix exponential spatial growth regression models	Fischer and Piribauer	*ePubWU Institutional Repository*	2013

150	Enhanced Spatial Covariance Matrix Estimation for Asynchronous Inter-Cell Interference Mitigation in MIMO-OFDMA System	Moon, Jong-Gun *et al*	*The Journal of Korean Institute of Communications and Information Sciences*	2009
151	The normal form of a positive semi-definite spatial stiffness matrix	Roberts, R.G.	*2002 Proceedings of the 5th Biannual World Automation Congress*	2002
152	Objective Assessment and Design Improvement of a Staring, Sparse Transducer Array by the Spatial Crosstalk Matrix for 3D Photoacoustic Tomography	Wong P. *et al*	*PloS One*	2015
153	A Grid Based Approach to Spatial Weighting Matrix Specification	Rahal, Charles	*SSRN Papers*	2017
154	Spatial dependence matrix feature and redundancy elimination algorithm using AdaBoost for object detection	Wen Jia *et al*	*Optical Engineering*	2011
155	Eigenstructures of Spatial Design Matrices	Gorsich, David J. *et al*	*Journal of Multivariate Analysis*	2002
156	Dynamic nearest neighbours for generating spatial weight matrix	Mawarni, Mutiara and Imam Machdi	*2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*	2017
157	Imaging through a scattering medium based on spatial transmission matrix	Bin Zhuang *et al*	*Proceedings Volume 10416, Optical Coherence Imaging Techniques and Imaging in Scattering Media II*	2017
158	Combined asymmetric spatial weights matrix with application to housing prices	Haiyong Zhang and Xinyu Wang	*Journal of Applied Statistics*	2017
159	Spatial elastic-plastic rigidity matrix of T short shearing wall	Wang Ning and Xu Xiao-xu	*Shanxi Architecture*	2006
160	Studies on the Spatial Homogeneity and Flexural Strength of Several sic Fiber-Reinforced Cvi-Sic Matrix Composites	Araki, H. *et al*	*Materials for Advanced Energy Systems and Fission & Fusion Engineering*	2003
161	Spatial Dependence in Financial Data: Importance of the Weights Matrix	Bera, Anil K.	*researchgate*	2016
162	Dependence of spatial effects on the level of regional aggregation, weights matrix, and estimation method	Demidova, Olga *et al*	*EconSTOR*	2015
163	Lecture 6: Matrix Exponential Spatial models	LeSage, James P.	*researchgate*	2004
164	Effects of Mixer Intermodulation Distortion on the Spatial Cross Correlation Matrix of Received Signals in Wireless Communication Systems	Dadashzadeh, G. *et al*	*researchgate*	n.a
165	A method of estimation of turbulent diffusion in a circular pipe based on spatial-dependence matrix	Ogawa and Yoshikawa	*Chemical Engineering Communications*	1990
166	Spatial Lag Model with Time-lagged Effects and Spatial Weight Matrix Estimation	Lam, Clifford and Cheng Qian	n.a	n.a

167	Measuring quantum states: an experimental setup for measuring the spatial density matrix Measuring quantum states	Tegmark, M.	n.a	1996
168	Inferring the contiguity matrix for spatial autoregressive analysis with applications to house price prediction	Sarkar and Chawla	*arXiv*	2016
169	A structured matrix approach for spatial-domain approximation of 2-D IIR filters	Shaw and Pokala	*IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*	1997
170	Visualizing the spatial localization of active matrix metalloproteinases (MMPs) using MALDI imaging MS	Muruganantham, Sasirekha	*ProQuest (Book)*	2011
171	Is a matrix exponential specification suitable for the modeling of spatial correlation structures?	Strauß, M.E *et al*	*Spatial Statistics*	2017
172	Division of Weight Matrix Based on Pair-Distances and Resulting Nonlinear Spatial Dependency in Spatial Autoregressive Models	Yokoi, Takahisa	n.a	2012
173	A Study for Gender Classification Based on Gait via Incorporating Spatial and Temporal Feature Matrix	Wang and Yu	*Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on*	2013
174	Supplement to “Irregular N2SLS and LASSO estimation of the matrix exponential spatial specification model”	Fei Jin and Lung-fei Lee	*econ.shufe.edu.cn*	2017
175	The W Matrix in Network and Spatial Econometrics: Issues Relating to Specification and Estimation	Corrado and Fingleton	*CEIS Working Paper No. 369*	2016
176	Spatial Weights Matrix and its Application	Changping Zhang	*Journal of Regional Development Studies*	2012
177	The Best Spatial Weight Matrix Order of Radius for GSTARIMA	Mubarak, Fadhlul *et al*	*International Journal of Engineering and Management Research*	2017
178	Enhanced Spatial Covariance Matrix Estimation for Asynchronous Inter-Cell Interference Mitigation in MIMO-OFDMA System	Moon, Jong-Gun *et al*	*The Journal of The Korean Institute of Communication Sciences*	2009
179	Modified local getis statistic on AMOEBA weights matrix for spatial panel model and its performance	Jajang	*repository.ipb.ac.id*	2014
180	Which spatial weighting matrix? An approach for model selection	Gomez, Marcos Herrera *et al*	*mpra.ub.uni-muenchen.de*	2011

APPENDIX I.B. LIST OF RELATED PRECEDENT RESEARCHES OF SPATIAL NEIGHBOR MATRIX

Table I.B Table of Related Precedent Researches of Spatial Neighbor Matrix

No.	Title	Author(s)	Date of Publication	Summary
1	Constructing the Spatial Weights Matrix Using a Local Statistic	Getis, Arthur and Jared Aldstadt	2010	Constructing spatial weights matrix based on Gi * local statistic
2	Using AMOEBA to Create a Spatial Weights Matrix and Identify Spatial Clusters	Aldstald and Getis	2006	The creation of a spatial weights matrix by a procedure called AMOEBA, A Multidirectional Optimum Ecotope-Based Algorithm, is dependent on the use of a local spatial autocorrelation statistic. The result are: 1) a vector that identifies those spatial units that are related and unrelated to contiguous spatial units; and 2) a matrix of weights whose values are a function of the relationship of the ith spatial unit with all other nearby spatial units for which there is a spatial association.
3	Estimating a spatial autoregressive model with an endogenous spatial weight matrix	Xi Qu and Lung-fei Lee	2015	In this paper, authors are constructed the W matrix based on endogenous approach.
4	Nonparametric Estimation of the Spatial Connectivity Matrix Using Spatial Panel Data	Beenstock and Felsenstein	2012	The authors use the moments from the covariance matrix for spatial panel data to estimate the parameters of the spatial autoregression model, including the spatial connectivity matrix W.
5	A Dynamic Spatial Weight Matrix and Localised STARIMA for Network Modelling	Tao Cheng *et al*	2014	In this paper, authors are aims to describe autocorrelation in network data with a dynamic spatial weight matrix and a localized STARIMA (LSTARIMA) model that captures the heterogeneity and nonstationarity.
6	Estimation of the spatial weights matrix under structural constraints	Bhattarcharjee, Arnab and Chris Jensen-Butler	2013	The authors propose new mehodology to estimate spatial weights matrix under spatial error model based an assumption that symmetrical spatial weights with extensions to other important spatial models
7	On the Four Types of Weight Functions for Spatial Contiguity Matrix	Yanguang Chen	2012	The aim of this study is at how to select a proper weight function to construct a spatial contiguity matrix for spatial analysis. The scopes of application of different weight functions are defined in terms of the characters of their spatial autocorrelation function (ACFs) and partial autocorrelation function (PACFs).
8	Automatic selection of a spatial weight matrix in spatial econometrics: Application to a spatial hedonic approach	Seya, Hajime *et al*	2013	

9	Spatial Nonstationarity and Spurious Regression: The Case with Row-Normalized Spatial Weights Matrix	Lung-fei Lee and Jihai Yu	2009	This paper investigates the spurious regression in the spatial setting where the regressant and regressors may be generated from possible nonstationary spatial autoregressive processes. Under the near unit root specification with a row-normalized spatial weights matrix, it is shown that the possible spurious regression phenomena in the spatial setting are relatively weaker than those in the nonstationary time series scenario.
10	Bayesians in Space: Using Bayesian Methods to Inform Choice of Spatial Weights Matrix in Hedonic Property Analyses	Mueller, Julie M. and John B. Loomis	2010	This paper found that improper choice of spatial weight matrix triggered 5% differences on the analysis of Bayessian approach a spatial hedonic model. The model estimates the impact of repeated wildfire on house prices in Southern California
11	Two-step lasso estimation of the spatial weights matrix	Ahrens, Achim and Arnab Bhattacharjee	2015	This study considers a two-step estimation strategy for estimating the n(n-1) interaction effects in a spatial autoregressive panel model where the spatial dimension is potentially large. The identifying assumption is approximate sparsity of the spatial weights matrix.
12	Detection and estimation of block structure in spatial weight matrix	Lam, Clifford and Pedro C.L. Souza	2016	The authors propose a method that captures group affiliation or, equivalently, estimates the block structure of a neighboring matrix embedded in a Spatial Econometric model.
13	A Proposal for an Alternative Spatial Weight Matrix under Consideration of the Distribution of Economic Activity	Perret, Jens K.	2011	The author proposes a new measuring concept where takes into account the regional economic or demographic structures and constructs distances between regions accordingly.
14	QML Estimation of the Spatial Weight Matrix in the MR-SAR Model	Benjanuvatra, Saruta	2013	In this paper, we introduce a sub-model for spatial weights and estimate a variable weight matrix for the mixed regressive, spatial autoregressive (MR-SAR) model by maximum Gaussian likelihood.
15	Spatial Weighting Matrix Selection in Spatial Lag Econometric Model	Kostov, Phillip	2013	This paper investigates the choice of spatial weighting matrix in a spatial lag model framework. This article expands the latter transformation approach on Kostov (2010) into a two-step selection procedure. The proposed approach aims at reducing the arbitrariness in the selection of spatial weighting matrix in spatial econometrics.
16	Selecting the Most Adequate Spatial Weighting Matrix: A Study on Criteria	Gomez, Marcus Herrera *et al*	2012	In this paper, authors revise the literature looking for criteria to select the proper spatial weight matrix. Also, a new nonparametric procedure is introduced. Their proposal is based on a measure of the information, conditional entropy. We compare these alternatives by means of a Monte Carlo experiment.

17	The construction of spatial weight matrix based on discrete points through building a searching area for each point	Wenii Yu *et al*	2010	In this paper, authors present an algorithm to construct a spatial weight matrix by employing the threshold method to measure the spatial connectivity among discrete spatial points. This method can save searching costs by building a searching area for each point when we set values for each element of the weight matrix, and achieve optimization in the process of constructing a spatial weight matrix
18	The Improbable Nature of the Implied Correlation Matrix from Spatial Regression Models	Sen, Monalisa and Anil K. Bera	2014	This paper suggests a way that the weight matrix can capture the underlying dependence structure of the observations. the possibility of constructing the weight matrix (or the overall spatial dependence in the data) that is consistent with the underlying correlation structure of the dependent variable is explored.
19	A New Selection Method of Spatial Weight Matrix	Ren Yinghua and You Wanhai	2012	The paper applies a component-wise boosting algorithm to deal with the selection issue of a spatial weight matrix in spatial lag models.
20	Comparison of Uniform and Kernel Gaussian Weight Matrix in Generalized Spatial Panel Data Model	Purwaningsih, Tuti *et al*	2015	In this paper, the authors try to compare the differences between uniform and Kernel Gaussian weight matrix. The construction of spatial weight matrix is based on R progamming languange.
21	Model for Effect of Spatial Weighted Matrix on Spatial Autocorrelation	Zhilang Wang *et al*	2013	The authors constructed their W matrix based on three approaches: 1) adjacency relationship, 2) distance relationship, and 3) comprehensive factor relationship.
22	Testing for a structural break in the weight matrix of the spatial error or spatial lag model	Angulo, Ana *et al*	2017	Authors studied the W matrix in the two generic spatial econometric models, allowing the friction-of-distance parameter to be freely estimated.
23	Spatial Lag Model Estimation with Sparse Adjustment for Spatial Weight Matrix	Lam, Clifford and Pedro C.L. Souza	n.a	The authors propose a linear combination of spatial weight matrix specifications, added with a potentially sparse adjustment matrix in order to choose a good spatial weight matrix for modelling.
24	The Influence of the Structure of Spatial Weight Matrix on Regression Analysis in the Presence of Spatial Autocorrelation	Tsutsumi, Morito *et al*	2000	
25	Spatial Prominence and Spatial Weights Matrix in Geospatial Analysis	Changping Zhang	2012	The author analyzed the impact of regional division analysis into small area units, such as a prominent areal unit, which is obtained by Markov chains method from a W matrix.
26	Dynamic nearest neighbours for generating spatial weight matrix	Mawarni, Mutiara and Imam Machdi	2017	Authors propose Dynamic Nearest Neighbours algorithm instead of commonly used nearest neighbour algorithm. In their evaluation, they found DNN algorithm outperforms other techniques of Rook, Queen, and k-Nearest Neighbours since it can be applied to both contiguous and sparse regions and produce two-way relations.

27	A Grid Based Approach to Spatial Weighting Matrix Specification	Rahal, Charles	2017	The author used an exogenously set of weighting matrix to analyze its sensitive specification on spatial econometric model.
28	Combined asymmetric spatial weights matrix with application to housing prices	Haiyong Zhang and Xinyu Wang	2017	A combined asymmetric spatial weights matrix is proposed by authors to capture the unequal spatial dependence and estimated by using non-nested hypothesis test.
29	Spatial Lag Model with Time-lagged Effects and Spatial Weight Matrix Estimation	Lam, Clifford and Cheng Qian	n.a	This paper considers a spatial lag model with different spatial weight matrices for different time lagged spatial effects
30	Division of Weight Matrix Based on Pair-Distances and Resulting Nonlinear Spatial Dependency in Spatial Autoregressive Models	Yokoi, Takahisa	2012	In this paper, author purposes to discuss realistic models by dividing spatial terms. The resulting nonlinear dependency function is represented through the coefficients of a pair of spatial weight matrices.
31	The W Matrix in Network and Spatial Econometrics: Issues Relating to SpeciÖcation and Estimation	Corrado, Luisa and Bernard Fingleton	2016	The authors analyzed the impact of misspecifying of W matrix on spatial econometric model. W matrix are used as part of model specification and used in estimation model.
32	Spatial Weights Matrix and its Application	Changping Zhang	2012	In this paper, author used several W matrix construction concept, such as: 1) binary weight, 2) distance decay-weight, 3) generalized weight, and 4) k-order neighbors weight.

APPENDIX II. Proof of Kronecker Product Representation of Arrow Direction.

Suppose we have spatial image matrix,

(A.1) $$\mathbf{D}_{r\times c} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}; \text{ in case } r = c = 3$$

Thus, we will have $\mathbf{I}_i$, $\mathbf{L}_{ji}$, and $\mathbf{U}_{ji}$ as follow

(A.2) $$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{L}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{U}_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Afterwards, we can substitute equation (A.1) into equation (7), then we will have

(A.3) $$\left(\mathbf{L}_{\mathrm{C}} \otimes \mathbf{I}_{\mathrm{R}}\right) = \begin{bmatrix} 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ 1\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}$$

(A.4) $$\left(\mathbf{U}_{\mathrm{C}} \otimes \mathbf{I}_{\mathrm{R}}\right) = \begin{bmatrix} 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 1\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & 0\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{bmatrix}$$

(A.5) $$\left(\mathbf{I}_{\mathrm{C}} \otimes \mathbf{L}_{\mathrm{R}}\right) = \begin{bmatrix} 1\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\ 0\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & 1\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\ 0\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & 1\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

$$\text{(A.6)}\qquad \left(\mathbf{I}_{\mathrm{C}}\otimes\mathbf{U}_{\mathrm{R}}\right)=\begin{bmatrix} 1\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}\\ 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 1\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}\\ 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 1\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}\end{bmatrix}$$

$$\text{(A.7)}\qquad \left(\mathbf{U}_{\mathrm{C}}\otimes\mathbf{L}_{\mathrm{R}}\right)=\begin{bmatrix} 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 1\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}\\ 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 1\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}\\ 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}\end{bmatrix}$$

$$\text{(A.8)}\qquad \left(\mathbf{U}_{\mathrm{C}}\otimes\mathbf{U}_{\mathrm{R}}\right)=\begin{bmatrix} 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 1\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}\\ 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 1\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}\\ 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix} & 0\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}\end{bmatrix}$$

$$\text{(A.9)}\qquad \left(\mathbf{L}_{\mathrm{C}}\otimes\mathbf{L}_{\mathrm{R}}\right)=\begin{bmatrix} 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}\\ 1\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}\\ 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 1\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix} & 0\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}\end{bmatrix}$$

(A.10)
$$\left(\mathbf{L}_C \otimes \mathbf{U}_R\right) = \begin{bmatrix} 0\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ 1\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ 0\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & 1\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} & 0\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

Thus, we can have **W** all matrix from equation (A.3)–(A.10),

(A.11)
$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & & & \\ 1 & 0 & 1 & 1 & 1 & 1 & & & \\ 0 & 1 & 0 & 0 & 1 & 1 & & & \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ & & & 1 & 1 & 0 & 0 & 1 & 0 \\ & & & 1 & 1 & 1 & 1 & 0 & 1 \\ & & & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

APPENDIX III. DAR and ION Scripts in R Language

```
# Constructing Primary Neighborhood Cells #

# Block 1 (Left Neighbor Cell) #
for (j in 1:(ymx)) {
  for (i in 2:(xmx)) {
    WP.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)−1]←1
  }
}

# Block 2 (Right Neighbor Cell) #
for (j in 1:(ymx)) {
for (i in 1:(xmx−1)) {
WP.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)+1]←1
  }
}

# Block 3 (Lower Neighbor Cell) #
for (j in 1:(ymx−1)) {
  for (i in 1:(xmx)) {
    WP.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)+xmx]←1
  }
}

# Block 4 (Upper Neighbor Cell) #
for (j in 2:(ymx)) {
  for (i in 1:(xmx)) {
    WP.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)−xmx]←1
  }
}

# Constructing Secondary Neighborhood Cells #
# Block 5 (Upper-Right Diagonal Neighbor Cell) #
for (j in 2:(ymx)) {
  for (i in 1:(xmx−1)) {
    WS.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)−(xmx−1)]←1
  }
}

# Block 6 (Upper-Left Diagonal Neighbor Cell) #
for (j in 2:(ymx)) {
  for (i in 2:(xmx)) {
    WS.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)−(xmx+1)]←1
  }
}

# Block 7 (Lower-Left Diagonal Neighbor Cell) #
for (j in 1:(ymx−1)) {
  for (i in 2:(xmx)) {
    WS.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)+(xmx−1)]←1
  }
}

# Block 8 (Lower-Right Diagonal Neighbor Cell) #
for (j in 1:(ymx−1)) {
  for (i in 1:(xmx−1)) {
    WS.G[(j*xmx)−(xmx−i),(j*xmx)−(xmx−i)+(xmx+1)]←1
  }
}
```

```
# Cumulative Neighborhood Cells #

# Block 9 #
W_AGGREGATE.G ←WP.G + WS.G
```

Script III.A. Direct Arrow Reading Script

```
# Constructing Primary Neighborhood Cells #
# Block 1 (Inner Matrix) #
for (j in 2:(ymx–1)) {
  for (i in 2:(xmx–1)) {
    WP[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)–1]←1;
    WP[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)+1]←1;
    WP[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)–xmx]←1;
    WP[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)+xmx]←1
  }
}

# Block 2 (Upper side of Outer Matrix) #
for (ku in 2:(xmx–1)) {
  WP[ku,ku–1]←1; WP[ku,ku+1]←1; WP[ku,xmx+ku]←1
}

# Block 3 (Left side of Outer Matrix) #
for (kl in seq((xmx+1),((ymx–2)*xmx)+1,xmx)) {
  WP[kl,kl–xmx]←1; WP[kl,kl+1]←1; WP[kl,kl+xmx]←1
}

# Block 4 (Right side of Outer Matrix) #
for (kr in seq((xmx+xmx),((ymx*xmx)–xmx),xmx)) {
  WP[kr,kr–1]←1; WP[kr,kr–xmx]←1; WP[kr,kr+xmx]←1
}

# Block 5 (Bottom side of Outer Matrix) #
for (kb in (((ymx–1)*xmx)+2):((ymx*xmx)–1)) {
  WP[kb,kb–1]←1; WP[kb,kb+1]←1; WP[kb,kb–xmx]←1
}

# Block 6 (Corner Area of Outer Matrix) #
WP[1,2] ← 1; WP[1,(xmx+1)] ←1;
WP[((ymx–1)*xmx)+1,((ymx–1)*xmx)+2]← 1;
WP[((ymx–1)*xmx)+1,((ymx–1)*xmx)–(xmx+1)] ← 1;
WP[xmx,xmx–1] ← 1; HP[xmx,xmx+xmx] ← 1;
WP[(ymx*xmx),(ymx*xmx)–1]←1;
WP[(ymx*xmx),(ymx*xmx)–xmx] ← 1

# Constructing Secondary Neighborhood Cells #

# Block 7 (Inner Matrix) #
for (j in 2:(ymx–1)) {
  for (i in 2:(xmx–1)) {
    WS[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)–(xmx–1)]←1;
    WS[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)–(xmx+1)]←1;
    WS[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)+(xmx–1)]←1;
    WS[(j*xmx)–(xmx–i),(j*xmx)–(xmx–i)+(xmx+1)]←1
  }
}
```

```
# Block 8 (Upper side of Outer Matrix) #
for (ku in 2:(xmx−1)) {
  WS[ku,ku+(xmx−1)]←1; WS[ku,ku+(xmx+1)]←1
}

# Block 9 (Left side of Outer Matrix) #
for (kl in seq((xmx+1),((ymx−2)*xmx)+1,xmx)) {
  WS[kl,kl−(xmx−1)]←1; WS[kl,kl+(xmx+1)]←1
}

# Block 10 (Right side of Outer Matrix) #
for (kr in seq((xmx+xmx),((ymx*xmx)−xmx),xmx)) {
  WS[kr,kr−(xmx+1)]←1; WS[kr,kr+(xmx−1)]←1
}

# Block 11 (Bottom side of Outer Matrix) #
for (kb in (((ymx−1)*xmx)+2):((ymx*xmx)−1)) {
  WS[kb,kb−(xmx+1)]←1; WS[kb,kb−(xmx−1)]←1
}

# Block 12 (Corner area of Outer Matrix) #
WS[1,xmx+2] ← 1;
WS[((ymx−1)*xmx)+1,((ymx−2)*xmx)+2] ←1;
WS[xmx,xmx+(xmx−1)] ← 1;
WS[(ymx*xmx),(ymx*xmx)−(xmx+1)] ←1

# Cumulative Neighborhood Cells #
# Block 13 #
W_AGGREGATE ←WP + WS
```

Script III.B. Inner–Outer Neighbor Script

APPENDIX IV. AVERAGE ELAPSED TIME PER BLOCK (in Seconds) IN DETAIL

Inner-Outer Neighbor Matrix (ION) Method													
Block	Average Elapsed Time in seconds for Datasize (Z_s)												
	$\mathcal{O}(ION)$	$D_{3\times3}$	$D_{30\times30}$	$D_{60\times60}$	$D_{90\times90}$	$D_{120\times120}$	$D_{150\times150}$	$D_{180\times180}$	$D_{210\times210}$	$D_{240\times240}$	$D_{270\times270}$	$D_{300\times300}$	$D_{330\times330}$
1	$(n-2)^2$	0.0150	1.4800	6.4375	15.9075	32.9100	63.7775	115.9275	204.8450	320.5675	402.5925	576.2350	920.2600
2	$(n-2)$	0.0100	0.0450	0.0900	0.1575	0.2650	0.5075	0.8400	1.4875	2.2400	2.3075	3.1575	4.6775
3	$(n-2)$	0.0100	0.0425	0.1000	0.1600	0.2725	0.4950	0.8450	1.4675	2.1675	2.2575	2.9850	4.5250
4	$(n-2)$	0.0075	0.0375	0.0975	0.1575	0.2625	0.4775	0.8300	1.4900	2.1675	2.1750	3.0150	4.5625
5	$(n-2)$	0.0050	0.0325	0.0925	0.1575	0.2775	0.4850	0.7850	1.4250	2.0950	2.1175	2.8025	4.3500
6	$8n$	0.0000	0.0075	0.0000	0.0000	0.0000	0.0150	0.0075	0.0150	0.0225	0.0000	0.0150	0.0225
7	$(n-2)^2$	0.0200	1.4875	6.4925	15.9225	32.9125	64.5650	118.7475	214.6450	345.7400	435.1425	605.3950	915.6050
8	$(n-2)$	0.0050	0.0325	0.0625	0.1100	0.1900	0.3425	0.5875	1.0025	1.4050	1.6575	2.3400	3.1450
9	$(n-2)$	0.0025	0.0250	0.0625	0.1025	0.1775	0.3250	0.5575	0.9750	1.3375	1.5825	2.2675	2.9750
10	$(n-2)$	0.0150	0.0350	0.0575	0.1050	0.1700	0.3500	0.5675	0.9750	1.3425	1.5700	2.2350	3.0150
11	$(n-2)$	0.0075	0.0250	0.0625	0.1125	0.1750	0.3375	0.5550	0.9450	1.2850	1.5100	2.1400	2.8250
12	$4n$	0.0075	0.0050	0.0050	0.0075	0.0050	0.0050	0.0000	0.0050	0.0075	0.0125	0.0075	0.0125
13	n	0.0025	0.0000	0.0000	0.0050	0.0075	0.0050	0.0125	0.0150	0.0200	0.0500	0.0325	0.0375
TOTAL		**0.1075**	**3.2250**	**13.5600**	**32.9100**	**67.6275**	**131.6875**	**240.2625**	**429.2975**	**680.3975**	**852.9750**	**1202.6250**	**1866.0400**

Direct Arrow Reading (DAR) Method													
Block	Average Elapsed Time in seconds for Datasize (Z_s)												
	$\mathcal{O}(DAR)$	$D_{3\times3}$	$D_{30\times30}$	$D_{60\times60}$	$D_{90\times90}$	$D_{120\times120}$	$D_{150\times150}$	$D_{180\times180}$	$D_{210\times210}$	$D_{240\times240}$	$D_{270\times270}$	$D_{300\times300}$	$D_{330\times330}$
1	$n(n-1)$	0.0250	0.4125	1.6750	3.8550	6.9250	11.7150	17.5850	26.7550	38.0125	54.7600	72.1875	89.0625
2	$n(n-1)$	0.0125	0.3975	1.6800	4.0200	7.6650	15.3125	25.3225	43.5350	66.5600	105.6250	143.4425	172.7150
3	$n(n-1)$	0.0150	0.4250	1.7275	4.2575	9.0075	20.1625	34.9625	60.4900	104.6400	189.9300	258.5250	296.3150
4	$n(n-1)$	0.0100	0.4100	1.7625	4.5150	10.3175	26.8075	46.8600	84.5575	180.5500	238.5625	455.5750	476.6250
5	$(n-1)^2$	0.0050	0.3900	1.6150	3.7500	6.8500	11.6975	17.7725	26.8775	37.6175	51.5475	73.6900	88.5625
6	$(n-1)^2$	0.0075	0.3950	1.6550	3.9800	7.6350	15.6050	26.0650	42.7375	65.5475	94.0650	146.7200	172.0225
7	$(n-1)^2$	0.0075	0.4000	1.7450	4.1825	8.9525	18.8100	36.3700	62.7675	102.6375	152.5575	215.6500	280.7825
8	$(n-1)^2$	0.0050	0.4000	1.7175	4.4500	10.1975	23.4550	48.9225	90.3425	161.9300	256.1650	360.0225	479.0075
9	n	0.0150	0.0000	0.0050	0.0000	0.0000	0.0100	0.0050	0.0175	0.0175	0.0625	0.0675	0.0575
TOTAL		**0.1025**	**3.2300**	**13.5825**	**33.0150**	**67.5500**	**143.5750**	**253.8650**	**438.0800**	**757.5125**	**1143.2750**	**1725.8800**	**2055.1500**

Generic Function for Shift Matrix of Kronecker Product Method													
Block	Average Elapsed Time in seconds for Datasize (Z_s)												
	$\mathcal{O}(KP1)$	$D_{3\times3}$	$D_{30\times30}$	$D_{60\times60}$	$D_{90\times90}$	$D_{120\times120}$	$D_{150\times150}$	$D_{180\times180}$	$D_{210\times210}$	$D_{240\times240}$	$D_{270\times270}$	$D_{300\times300}$	$D_{330\times330}$
1	n	0.0000	0.0050	0.0050	0.0150	0.0025	0.0100	0.0000	0.0000	0.0125	0.0050	0.0150	0.0150
2	n	0.0175	0.0000	0.0050	0.0000	0.0000	0.0050	0.0025	0.0000	0.0000	0.0000	0.0000	0.0000
3	n	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050	0.0000	0.0000	0.0000	0.0000	0.0000
4	n	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	n	0.0000	0.0000	0.0000	0.0025	0.0000	0.0000	0.0000	0.0050	0.0000	0.0000	0.0025	0.0025
6	n	0.0000	0.0000	0.0025	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050	0.0050	0.0000	0.0000
7	$4 \cdot n \log n$	0.0025	0.0050	0.0075	0.0000	0.0100	0.0150	0.0175	0.0325	0.0725	0.0825	0.0750	0.0825
TOTAL		**0.0200**	**0.0100**	**0.0200**	**0.0175**	**0.0125**	**0.0300**	**0.0250**	**0.0375**	**0.0900**	**0.0925**	**0.0925**	**0.1000**

matrixcalc **Function for Shift Matrix of Kronecker Product Method**													
Block	Average Elapsed Time in seconds for Datasize (Z_s)												
	$\mathcal{O}(KP2)$	$D_{3\times3}$	$D_{30\times30}$	$D_{60\times60}$	$D_{90\times90}$	$D_{120\times120}$	$D_{150\times150}$	$D_{180\times180}$	$D_{210\times210}$	$D_{240\times240}$	$D_{270\times270}$	$D_{300\times300}$	$D_{330\times330}$
1	*unknown*	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0075	0.0000	0.0000
2	*unknown*	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	*unknown*	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050	0.0000	0.0000	0.0000	0.0000
4	*unknown*	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	n	0.0000	0.0000	0.0000	0.0000	0.0000	0.0025	0.0000	0.0000	0.0000	0.0000	0.0025	0.0000
6	n	0.0000	0.0000	0.0000	0.0050	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050
7	n	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
8	n	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
9	n	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
10	n	0.0000	0.0025	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0025	0.0000	0.0000	0.0000
11	n	0.0000	0.0075	0.0000	0.0150	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0050
12	n	0.0025	0.0000	0.0000	0.0000	0.0075	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	$4 \cdot n \log n$	0.0050	0.0150	0.0100	0.0150	0.0125	0.0150	0.0150	0.0150	0.0225	0.0775	0.0950	0.0875
TOTAL		**0.0075**	**0.0250**	**0.0100**	**0.0350**	**0.0200**	**0.0175**	**0.0150**	**0.0200**	**0.0250**	**0.0850**	**0.0975**	**0.0975**

Cell2nb of *spdep* Package Method													
Block	Average Elapsed Time in seconds for Datasize (Z_s)												
	$\mathcal{O}(cell2nb)$	$D_{3\times3}$	$D_{30\times30}$	$D_{60\times60}$	$D_{90\times90}$	$D_{120\times120}$	$D_{150\times150}$	$D_{180\times180}$	$D_{210\times210}$	$D_{240\times240}$	$D_{270\times270}$	$D_{300\times300}$	$D_{330\times330}$
1	*unknown*	0.0100	0.3075	1.1925	2.6600	4.7525	7.4700	10.7350	14.7700	19.3475	24.4725	30.4075	37.0925
2	*unknown*	0.0050	0.0225	0.0275	0.0550	0.0850	0.1650	0.1900	0.2650	0.3775	0.4450	0.7075	0.6925
TOTAL		**0.0150**	**0.3300**	**1.2200**	**2.7150**	**4.8375**	**7.6350**	**10.9250**	**15.0350**	**19.7250**	**24.9175**	**31.1150**	**37.7850**

Poly2nb of *spdep* Package Method													
Block	Average Elapsed Time in seconds for Datasize (Z_s)												
	$\mathcal{O}(poly2nb)$	$D_{3\times3}$	$D_{30\times30}$	$D_{60\times60}$	$D_{90\times90}$	$D_{120\times120}$	$D_{150\times150}$	$D_{180\times180}$	$D_{210\times210}$	$D_{240\times240}$	$D_{270\times270}$	$D_{300\times300}$	$D_{330\times330}$
1	*unknown*	0.0000	0.0700	0.2450	0.5775	1.0850	1.7925	2.9475	3.8800	5.0950	6.4100	8.3750	9.2025
2	*unknown*	0.0100	0.1925	1.2800	5.2375	16.9200	47.0400	88.1175	169.7350	285.0100	496.6250	756.9625	1144.7250
3	*unknown*	0.0100	0.0250	0.0175	0.0425	0.0825	0.1275	0.1850	0.2550	0.3375	0.4225	0.5350	0.6450
TOTAL		**0.0200**	**0.2875**	**1.5425**	**5.8575**	**18.0875**	**48.9600**	**91.2500**	**173.8700**	**290.4425**	**503.4575**	**765.8725**	**1154.5730**