A Framework for Industrial Cluster Detection

Tomoya Mori*

March 2014

Abstract

This manual describes the usage of C++ and Python programs developed for identifying industrial clusters and for studying their spatial patterns in Mori and Smith [1].

*Institute of Economic Research, Kyoto University. Email: mori@kier.kyoto-u.ac.jp.

1 Introduction

This manual describes the usage of C++ and Python programs developed for identifying industrial clusters and for studying their spatial patterns in Mori and Smith [1].[1] In Section 2, the required data and their structures are described. In Sections 3 through 6, the usage of the programs are described. All the source codes as well as the data can be downloaded from the website specified in Section 7.

2 Data

This section describes the basic data to be prepared prior to the cluster detection. All data are to be saved in comma-separated-values (CSV) format.

List of all industries

Denote by $I \equiv \{i_1, i_2, \ldots, i_{\overline{I}}\}$ the set of industries, where $\overline{I}\,(\equiv |I|)$ be the cardinality of I. The list of industries is provided by $(\overline{I} \times 2)$-dimensional data as in Table 1, where the first column lists the enumerated integer-valued identities (IDs) of industries (from 0 to $\overline{I} - 1$), and the second column lists the string-valued IDs of industries, where the latter is typically taken from industrial classifications such as Standard Industrial Classification (SIC) codes.

0	i_1
1	i_2
$\vdots$	$\vdots$
$\overline{I} - 1$	$i_{\overline{I}}$

Table 1: List of all industries

[1] See also Mori and Smith [2] for an application.

Establishment Location Data

It is assumed that establishment counts are available for a given set of basic regions (e.g., municipalities in Japan, counties in the US, and unions in Germany). Denote by $R \equiv \{r_1, r_2, \ldots, r_{\overline{R}}\}$ the set of basic regions, where $\overline{R} \equiv |R|$. Then, the location patterns of each industry is expected to be given by a CSV matrix as in Table 2. Here, the first column lists integer-valued region IDs,[2] and the first row lists the string-valued industry IDs (strings). Each (j, k)-th element is the number of establishments of industry i_k in region r_j.

Region/Industry	i_1	i_2	$\cdots$	$i_{\overline{I}}$
r_1	n_{11}	n_{12}	$\cdots$	$n_{1\overline{I}}$
r_2	n_{21}	n_{22}	$\cdots$	$n_{2\overline{I}}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$r_{\overline{R}}$	$n_{\overline{R}\,1}$	$n_{\overline{R}\,2}$	$\cdots$	$n_{\overline{R}\,\overline{I}}$

Table 2: Establishment locations

Areal Size Data

The double-valued areal size, a_r, in each individual basic region, $r \in R$, is expected to be given by two column data as in Table 3. More precisely, the relevant area is the census-defined *developable area* for the cluster detection in Section 3, while it is the *total area* for identifying essential containment in Section 5.

REGION	AREA
r_1	a_1
$\vdots$	$\vdots$
$r_{\overline{R}}$	$a_{\overline{R}}$

Table 3: Areal size data

[2] These IDs are typically the Census-defined jurisdiction IDs.

Adjacency Relations of the Basic Regions

Adjacent neighbors of each basic region should be identified prior to cluster detection,[3] and given by a CSV matrix as in Table 4, where in each row, j, the second element is a basic region which is adjacent to the first element so that region $r_j \in R$ has N_j adjacent neighbors.[4]

REGION	NEIGHBOR
r_1	r_{11}
$\vdots$	$\vdots$
r_1	r_{1N_1}
r_2	r_{21}
$\vdots$	$\vdots$
r_2	r_{2N_2}
$\vdots$	$\vdots$
$r_{\overline{R}}$	$r_{\overline{R}1}$
$\vdots$	$\vdots$
$r_{\overline{R}}$	$r_{\overline{R}N_{\overline{R}}}$

Table 4: Adjacent neighbors list

Boundary Regions

The set of *boundary regions* ($\subset R$) needs to be defined for the cluster detection. It is used in order to solidify clusters in Section 3 and agglomerations in Section 6 (see Mori and Smith [1, §3]). Typical boundaries include the basic regions along the national borders and sea coasts.[5] The list of boundary basic regions is given by a CSV column of basic region IDs as in Table 5.

[3] GIS softwares (ArcGIS Ver.10.2 for our case) should be able to identify adjacency relations among the basic regions. But, some manual modifications may be necessary, since some basic regions may better be considered as adjacent, for instance, when they are separated by narrow rivers and inland seas, but connected bridges.

[4] Here, only formal requirement for the set of adjacent regions of a given basic region r is that it does not include region r itself.

[5] The coast of a large lake could also be boundaries when multiple basic regions face this lake. An example is Biwa Lake in Japan as indicated in Figure 1.

BOUNDARY
b_1
⋮

Table 5: Boundary regions

Figure 1 depicts the boundary regions of Japan.

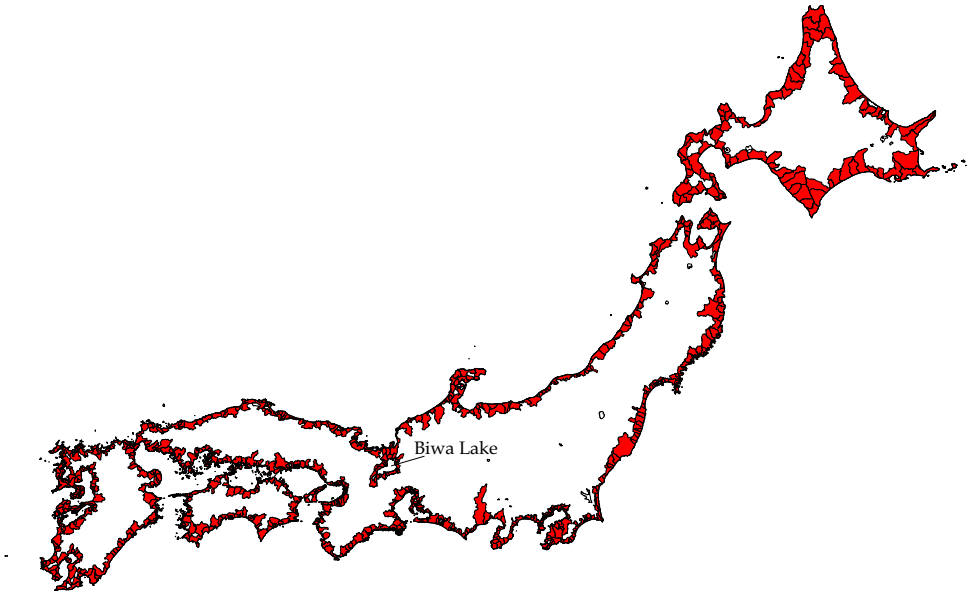


Figure 1: Boundary regions

Shortest-Path Regional Sequences and Shortest-Path Distanes

In the present cluster detection framework, all the basic regions are considered to form a regional network, where the set of vertices consist of the set of all basic regions, and the set of edges consist of the shortest-route paths between all pairs of adjacent basic regions (see Table 4), where the shortest-route distance between a given pair of basic regions is the travel distance along the shortest road-path connecting these two

regions.[6]

A Python program, *ShortestPathLocalMoveMP.py*, identifies the shortest path sequence of basic regions between each pair of basic regions, from (*i*) the shortest-route distance between each pair of basic regions and (*ii*) the set of neighbor basic regions of each basic region. Data (*i*) is assumed to be given by a three-column CSV data as in Table 6 where the distance values, d_{rs}, between the basic regions r and s represents the shortest travel distance between these two regions.[7] Data (*ii*) is the same data described in Table 4.

ORIGIN	DESTINATION	DISTANCE
r_1	r_2	d_{12}
$\vdots$	$\vdots$	$\vdots$

Table 6: Shortest-route / path distances

The shortest-path sequence of basic regions between each given pair of basic regions generated by *ShortestPathLocalMoveMP.py* is stored in a CSV matrix as in Table 7. Each element l_{ij} is the first basic region to reach from region r_i along the shortest path from basic region r_i to basic region r_j, where $l_{ii} \equiv r_i$. The shortest-path sequences of basic regions connecting any given pair of connected basic regions can be identified from this matrix. Here, a given pair of regions are *connected* if there exists a sequence of adjacent regions in R between these two regions. If regions $r_i$and $r_j$are not connected, then it is indicated by $\ell_{ij} = -1$.

[6]Only formal requirement for the shortest-route paths is that the inter-regional distances define a metric in the basic-region network. Travel distances are more desirable than simpler distances such as the Great-circle distances, since they take into account the basic topographical influences on the practical distances.

[7]Table 6 is expected to include only the origin-destination pairs, (r, s), of regions such that $r < s$.

REGION	r_1	$\cdots$	$r_{\overline{R}}$
r_1	l_{11}	$\cdots$	$l_{1\overline{R}}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$r_{\overline{R}}$	$l_{\overline{R}1}$	$\cdots$	$l_{\overline{R}\overline{R}}$

Table 7: Shortest-path sequences between basic regions

To illustrate how it works, a three-region example is shown in Table 8. The shortest path from region 1 to region 3 is given by $(1, \ell_{13} = 2, \ell_{23} = 3)$, and that from region 3 to region 1 is given by the reversed sequence of regions. Any other pair of regions can be reached directly from one another.

REGION	1	2	3
1	1	2	2
2	1	2	3
3	2	2	3

Table 8: An example of the shortest-path sequences

The corresponding *shortest-path distance* between any given pair of basic regions is stored as a three-column data in a CSV format as in Table 6, where DISTANCE value here indicates the shortest-path distance instead of shortest-route distance. As noted in Mori and Smith [1, §3.1], shortest-path distance is at least as large as the shortest-route distance. For instance for the three-region example shown in Table 8, if the shortest-path distance between regions i and j is denoted by $d^*_{ij} (\equiv d^*_{ji})$, then $d^*_{12} = d_{12}$, $d^*_{23} = d_{23}$ and $d^*_{13} = d_{12} + d_{13} (\geq d_{13})$. If two regions are not connected, then the shortest-path distance between these two regions is not specified, and is treated as infinity.

Figure 2 depicts the shortest path sequence of basic regions from Aomori-shi to Sata-cho, Kimotsuki-gun in Japan.

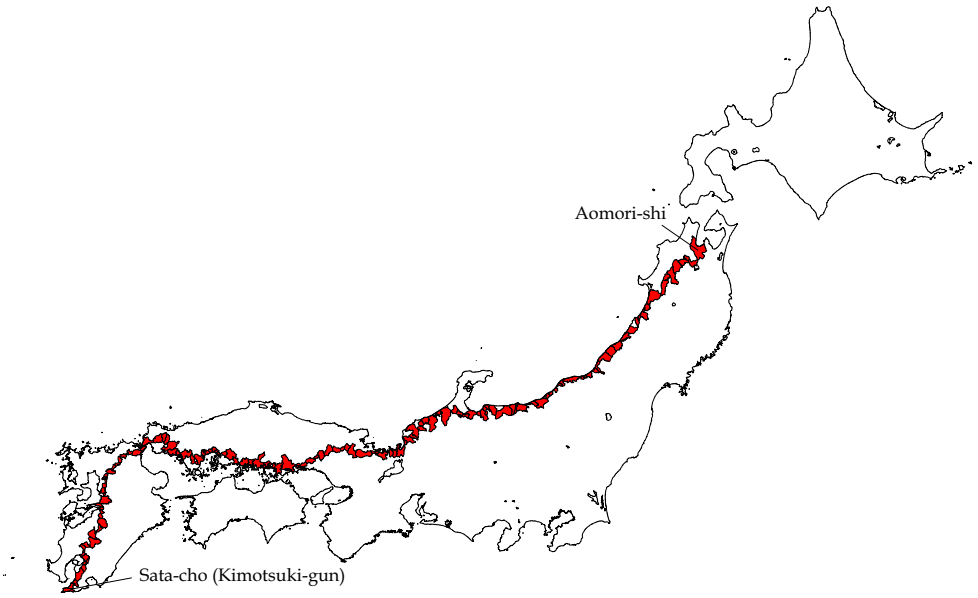


Figure 2: Shortest path from Aomori-shi to Sata-cho (Kimotsuki-gun)

3 Cluster Detection Framework

3.1 Inputs and Outputs

The list of input parameters are supposed to be written in "input_ detectclusters.txt". There are 15 lines in this file as shown in Table 9.[8]

The first and second lines in Table 9 specify the range, (i, j), of industries for which the cluster detection is to be conducted, i.e., the cluster schemes for the i-th through $(j-1)$-th industries are identified. The values for i and j should be chosen from the first column in Table 1.

The third, and the fifth through tenth lines specify the filenames whose contents have already been described above (the corresponding tables are indicated in the parentheses). The fourth line specifies the maximum allowable distance for cluster expansion which is named as δ in Mori and Smith [1, §4.2.2]. The eleventh line specifies the output

[8] In each line of the input file named "input_ ... .txt" for any given C++ program below, the part to the right of "//" is taken as comments, and is ignored by the program.

directory to which all the output files are expected to be written.

Line number	Input	Data type	Description
1	First industry	integer	$0 \leq i < \overline{I}$
2	Last industry		$i < j \leq \overline{I}$
3	Industry list (Table 1)	string	File name
4	Cluster expansion range	double	$[0, \infty)$
5	Establishment counts (Table 2)	string	File name
6	Areal sizes (Table 3)		
7	Adjacency relations (Table 4)		
8	Boundaries (Table 5)		
9	Inter-regional distances (Table 6)		
10	Shortest-path sequences (Table 7)		
11	Output directory		Directory path
12	Event-log flag	bool (0/1)	True$\equiv 1$
13	Summary-log flag		
14	Progress-log flag		
15	Temporary cluster-log flag		

Table 9: Inputs for the cluster detection

The twelfth through fifteenth lines specify the types of output files to be generated. Each input takes a 0/1 value, where value "1" means that the corresponding log file is to be written to the folder specified in the eleventh line, while value "0" means that this log output is suppressed. The *event log* (12th line) records each instance of the formation of a new cluster and expansion of an existing cluster during the process of cluster detection described in Mori and Smith [1, §4]. An example of the event log for the case of an SIC three-digit industry, "ophthalmic goods, including frames" of Japan (in 2001), is shown in Table 10 below.[9] The first column in the log indicates the event, either "formation" or "expansion", which took place; the second column indicates the cluster at which this event took place; the third column indicates the consolidated basic region or cluster if the event were "ex-

[9]The examples of cluster schemes in this article may be slightly different from those presented in Mori and Smith [1] since the set of boundary basic regions have been slightly updated. But, the differences are very minor.

pansion" of an existing cluster.[10,11] The third and forth columns show the log-likelihood and BIC values (see Mori and Smith [1, eq. (2.13)]), respectively, after the event took place.

EVENT	CLUSTER	CONSOLIDATED	LIKELIHOOD	BIC
FORMATION	18207	–	4538.439	4534.92
FORMATION	18381	–	4954.233	4947.195
FORMATION	18201	–	5227.980	5217.423
FORMATION	18421	–	5441.396	5427.320
FORMATION	18203	–	5650.183	5632.589
FORMATION	27116	–	5802.202	5781.088
FORMATION	27202	–	5890.609	5865.977
EXPANSION	18201	18426	5959.944	5935.312
FORMATION	27126	–	5993.229	5965.077
FORMATION	13106	–	6023.563	5991.892
EXPANSION	13106	13116	6064.826	6033.155
EXPANSION	13106	11234	6096.827	6065.156
⋮	⋮	⋮		

Table 10: Event log

The *summary log* in (13th line in Table 9) contains the information shown in Table 11. The identified clusters are ordered in terms of their BIC contributions shown in the BIC_INC column (whose share in the total BIC value for the cluster scheme is shown in the INC_SHARE column), where the formal definition of the BIC contributions of individual clusters is given in Mori and Smith [1, §4.4]. The CENTER column indicates the cluster IDs, COUNT and COUNT_SHARE columns indicate the establishment counts in the corresponding cluster and their shares in the total counts of establishments in all the clusters of this industry, respectively, and the AREA_SHARE column indicates the areal

[10]The ID for each cluster is the ID of the first basic region which was included in this cluster.

[11]The basic regions or clusters indicated in the column, CONSOLIDATED, is not only the region/cluster to be added to the cluster in the column, CLUSTER, but the all the basic regions in the d-convex solid of this CONSOLIDATED region/cluster and the original CLUSTER will replace the original cluster.

share of individual clusters in the national area.[12]

ORDER	CENTER	COUNT	COUNT_SHARE	AREA_SHARE	BIC_INC	INC_SHARE
1	18207	688	0.6040386	0.00044936	4534.92	0.7085581
2	18381	67	0.05882353	0.00011927	412.2748	0.06441583
⋮	⋮	⋮	⋮	⋮	⋮	⋮
44	8449	1	0.00087796	0.00050586	0.8882262	0.00013878

Table 11: Summary log

The *progress log* in (14th line in Table 9) contains the detailed record of each change made in the cluster scheme as in Table 12 for the "ophthalmic goods" industry. The first log indicates the location and the corresponding BIC value of the first (single-region) cluster formed. Thereafter, each log indicates the locations and the corresponding BIC values for the best new cluster formation, or the best expansion of an existing cluster. As for the expansion, both the expanding cluster and the candidate expansion are indicated. For example, in log 14, the best expansion is of cluster 27126 toward region 27212 located at 6.394km from the cluster.[13] The last line in each log is the chosen action (formation or expansion) together with the number of clusters after this change is made. In this particular example, the cluster detection is completed at the 91st modification, where it was found that neither new cluster

[12]It is possible that BIC contributions of a few clusters become negative, as this cluster detection (based on the forward search) only guarantees the local optimum. During the cluster detection procedure, the BIC increment is always positive when each clusters are formed or expanded. But, it does not imply that the BIC contributions by all the resuting clusters are positive, when BIC contributions by clusters are recomputed after the cluster scheme has been identified. Such obvious sub-optimality arises typically for ubiquitous industries (e.g., "bakery and confectionary products" and "cement and its products"). In the case of Japan, there are seven industries (out of 163) in the SIC three-digit manufacturing industries for which there are clusters with negative BIC contributions. In such a case, at least the clusters with negative BIC contributions should be removed from the further analyses based on the identified clusters.

[13]This log does not distinguish whether expansion is toward a residual region or toward an existing cluster. In this particular example "27212" is a residual region, but in general it could belong to an existing cluster. In that case, the expansion means the consolidation of this entire cluster to cluster, 27126.

formation nor new expansion of an existing cluster would not improve BIC for the cluster scheme.[14]

Log : 1
First cluster at 18207
Current value : 4534.92
Log : 2
Current value : 4534.92
Candidate formation at 18381(4947.19)
Candidate expansion at : 18207 to 18381 with dist 10.0441 (4910.33)
Formation : Num clusters = 2
⋮
Log : 14
Current value : 6093.54
Candidate formation at 23202(6110.4)
Candidate expansion at : 27126 to 27212 with dist 6.39345 (6110.44)
Expansion : Num clusters = 9
⋮
Log : 91
Current value : 6400.21
Candidate formation at -1(-1)
Candidate expansion at : 9402 to 9322 with dist 31.9367 (6399.37)
Cluster system has been found ...

Table 12: Progress log

The composition of each cluster is written to a file when either formation or expansion of a cluster takes place if the flag in (15th line in Table 9) is "1", and also when the cluster detection is completed. If this flag is "0", then the result is written to a file only when the cluster detection is completed. There are two outputs, one of which lists the "core" part of each cluster (i.e., those member basic regions of each cluster containing establishments of the industry), and the other of which

[14]The location, "-1", of the potential formation of a new cluster means that there is no possible new cluster, i.e., all the basic regions with establishments of this industry have been included in the existing clusters.

lists the entire d-convex solidified set of the core part of each cluster.[15] Both outputs are two-column CSV data. The case of the latter output is shown in Table 13, where the elements in the first column, C_i, indicates the cluster ID, while the elements in the second column, $r_{ij} \in R$, indicates the member basic regions in cluster C_i, where $\overline{C_i}$ represents the number of basic regions in C_i. The former output is in the same format, where only difference is that the second column lists the core basic regions of the cluster specified in the first column.

CLUSTER	MEMBER
C_1	r_{11}
$\vdots$	$\vdots$
C_1	$r_{1\overline{C}_1}$
C_2	r_{21}
$\vdots$	$\vdots$
C_2	$r_{2\overline{C}_2}$
$\vdots$	$\vdots$
$C_{k_\mathbf{C}}$	$r_{k_\mathbf{C}1}$
$\vdots$	$\vdots$
$C_{k_\mathbf{C}}$	$r_{k_\mathbf{C}\overline{C}_{k_\mathbf{C}}}$

Table 13: Cluster coverages

The cluster detection for this particular industry took about four minutes on Macbook Pro (Late 2013, 2.6GHz Intel Core i7, 16GB RAM, 1600MHz DDR3). Generally, the computation takes longer for more ubiquitous industries. For instance, the cluster detection for another SIC three-digit industry, "bakeries and confectionary products", took three days.[16]

[15] Each cluster detected is a slightly conservative version of the one defined in Mori and Smith [1, §4], in the sense that each cluster consists of the d-convex solid of only the core part rather than that of the entire cluster. The present approach is computationally less demanding without requiring any critical conceptual compromise.

[16] The computation time varies widely depending on the environment. For instance, Mac Pro (Late 2013, 12 Core Xeon E5) finished identifying "bakery" clusters in a few hours.

Figures 3 and 4 show the spatial coverage of clusters of "livestock products" and "ophthalmic goods including frames" industries in Japan in 2001, respectively. The red area indicates the core part of the clusters in which establishments are located, while the pink area are the vacant regions which have been included in the d-convex solid of the core parts.

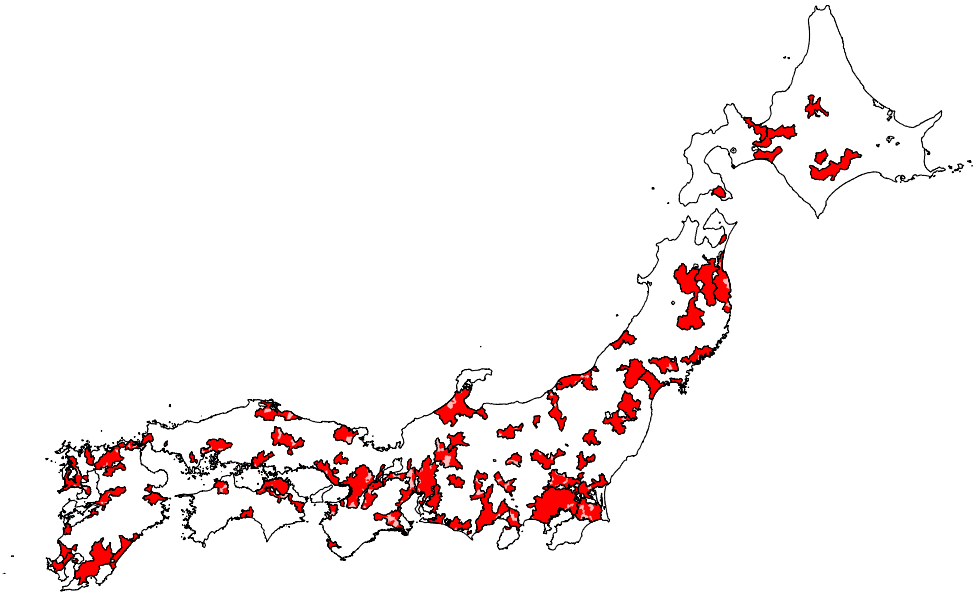


Figure 3: Clusters of "livestock products"

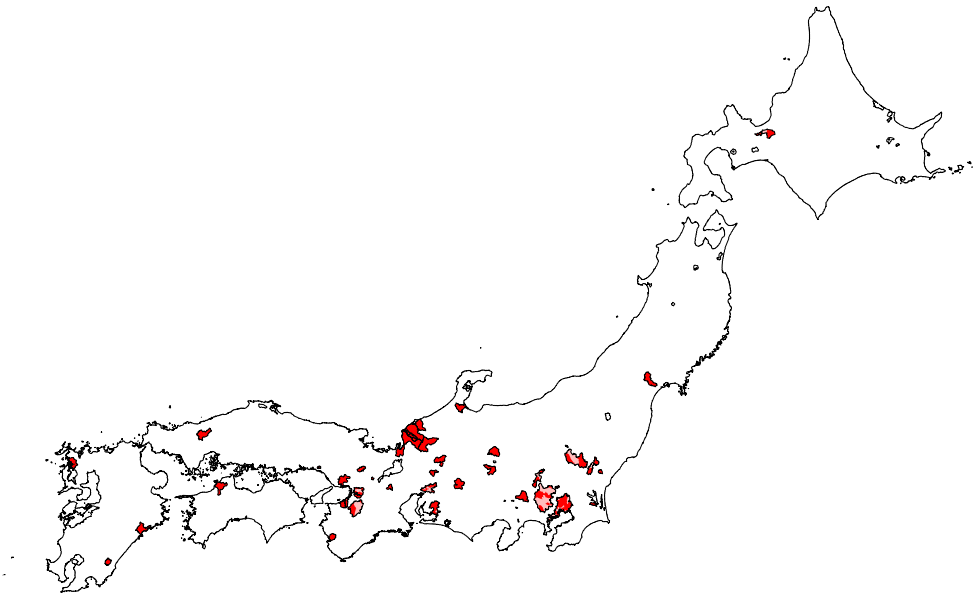


Figure 4: Clusters of “ophthalmic goods including frames”

3.2 Compilation

There are eleven common components used in the C++ cluster detection program. These are listed in Table 14 together with their dependencies. Our compilation is conducted in Apple’s Xcode Ver. 5.1 under Mac OS X ver. 10.9.2 with the compiler for C++, Apple LLVM ver. 5.0, with options, “-Ofast” (i.e., fast, aggressive optimization) and “-static” (i.e., prohibit dynamic link). The compilation is also verified using GNU C++ (g++, ver.4.7.3),[17] under Ubuntu Linux ver. 13.04, where the makefile is shown in Table 15.[18,19]

[17]GNU g++ is a free C++ implementations for Unix/Linux operating systems provided by the GNU project (http://www.gnu.org).

[18]The “makefile” is the name of the text file with the contents shown in Table 15. This makefile assumes all the relevant files are in the same folder.

[19]It is known that the command line compilation using GNU g++ does not work under Mac OS X due to linker problems. The use of Xcode is recommend under Mac OS X.

Class	Dependencies
ClusterBuilder	ClusterSystem Cluster Region Distance
ClusterSystem	Cluster Region Distance
Cluster	Region Distance
Region	Distance
Distance ShortestPathData IndustryLocationData	Csv1
Csv1	NamedObject
NamedObject	IOAdapter StringManipulator
CantOpenFile NoFile	

Table 14: Class dependencies

```
CC = g++
EXEFILE = main
TARGET =IdentifyClusters
OBJ = $(EXEFILE).o Region.o IndustryLocationData.o Distance.o ShortestPathData.o NoFile.o
    CantOpenFile.o Cluster.o ClusterSystem.o ClusterBuilder.o Csv1.o StringManipulator.o IOAdapter.o
$(TARGET) : $(OBJ)$(CC) $(OBJ) -o $(TARGET) -O3 -w -static
$(EXEFILE).o : $(EXEFILE).cpp Region.h IndustryLocationData.h $(CC) -c $< -o $@ -O3 -w -static

Region.o : Region.cpp Region.h Distance.h
$(CC) -c $< -o $@ -O3 -w -static
IndustryLocationData.o : IndustryLocationData.cpp IndustryLocationData.h
$(CC) -c $< -o $@ -O3 -w -static

Distance.o : Distance.cpp Distance.h
$(CC) -c $< -o $@ -O3 -w -static

ShortestPathData.o : ShortestPathData.cpp ShortestPathData.h
$(CC) -c $< -o $@ -O3 -w -static
Cluster.o : Cluster.cpp Cluster.h Region.h Distance.h
$(CC) -c $< -o $@ -w -static

ClusterSystem.o : ClusterSystem.cpp ClusterSystem.h Cluster.h Region.h Distance.h
$(CC) -c $< -o $@ -O3 -w -static

ClusterBuilder.o : ClusterBuilder.cpp ClusterBuilder.h Cluster.h Region.h Distance.h
$(CC) -c $< -o $@ -O3 -w -static

StringManipulator.o : StringManipulator.cpp StringManipulator.h
$(CC) -c $< -o $@ -O3 -w -static

Csv1.o : Csv1.cpp Csv1.h NamedObjectVector.h StringManipulator.h IOAdapter.h
$(CC) -c $< -o $@ -O3 -w -static

IOAdapter.o : IOAdapter.cpp IOAdapter.h
$(CC) -c $< -o $@ -O3 -w -static

NoFile.o : NoFile.cpp NoFile.h
$(CC) -c $< -o $@ -O3 -w -static

CantOpenFile.o : CantOpenFile.cpp CantOpenFile.h
$(CC) -c $< -o $@ -O3 -w -static
```

Table 15: Makefile for cluster detection

4 Spurious Clusters

This section presents C++ programs generating spurious clusters for each industry. If the actual establishment locations of a given indus-

try exhibit a significant clustering tendency, then the BIC value of the corresponding cluster scheme should be significantly larger than those of the spurious clusters identified from randomized establishment locations of this industry. Below, Section 4.1 presents the program which generates the set of randomized locations for each industry, and Section 4.2 presents the program which identifies the cluster schemes from these random location patterns.

4.1 Randomization of Establishment Locations

GenerateRandomLocations generates samples of randomized location patterns of establishments, where in each sample, all the establishments in a given industry are randomly distributed among the basic regions according to a given reference probability distribution over the set of basic regions. The reference distribution adopted in Mori and Smith [1] is the distribution of developable area, i.e., the uniform probability distribution over the developable area. But, any other reasonable reference distribution can be substituted.

The input parameters should be listed in the text file named "input_generate_ randomlocation.txt" containing eight lines shown in Table 16.

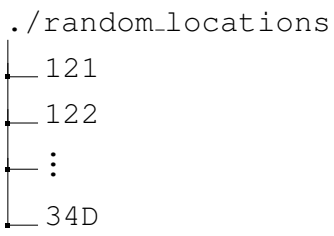
Line number	Input	Data type	Description
1	First industry, i	integer	$0 \leq i < \overline{I}$
2	Last industry, j		$i < j \leq \overline{I}$
3	Industry list (Table 1)	string	File name
4	First random sample, k	integer	$k \geq 1$
5	Last random sample, ℓ		$\ell \geq k$
6	Establishment counts (Table 2)	string	File name
7	Areal sizes (Table 3)		
8	Output directory		Directory path
9	Output filename (common part)		

Table 16: input_generate_randomlocation.txt

The program generates from k-th through ℓ-th random location sam-

ples for the i-th through j-th industries. All samples for a given industry i is written in the industry-specific folder specified at the eighth line, e.g., "./random_locations/".

Before running this program, the subdirectories for individual industries should be constructed under the output directory, where the the name for each subdirectory should be the industry ID. For instance, in the case of the Japanese SIC three-digit manufacturing industries in 2001, these are 121, 122, ..., 34D. Thus, the output directory structure should look as below:



The results for each k-th sample for this industry is then written to a file named "random_locations_ind_i_k.csv" in the directory, "./ random_locations/i/". It is a two-column CSV data as in Table 17 where the first column lists the all the basic regions in R, and the second column lists the number of establishment counts in each region specified in the first column. The sum of the second column, $\sum_{k=1}^{\overline{R}} n_k$, equals the total number of establishments in the given industry i.

r_1,	n_1
r_2,	n_2
$\vdots$	$\vdots$
$\overline{R}$,	$n_{\overline{R}}$

Table 17: Randomized location patterns

4.2 Construction of Spurious Clusters

GenerateSpuriousClusters identifies the best cluster scheme for each random location pattern generated by *GenerateRandomLocations* above.

Before running this program, the output directory should be constructed just like the case of random location patterns in Section 4.1 above. Namely, if the results for all industries were to be stored in a folder named "./spurious_clusters/", then under this folder, construct subfolders for individual industries, $i_1, i_2, \ldots, i_{\overline{I}} \in I$, i.e., "./spurious_clusters/i_1/" ... "./spurious_clusters/$i_{\overline{I}}$/".

The input parameters should be listed in the text file named "input_generate_ spuriousclusters.txt" containing the following fourteen lines as in Table 18.

Line number	Input	Data type	Description
1	First industry, i	integer	$0 \leq i < \overline{I}$
2	Last industry, j		$i < j \leq \overline{I}$
3	Industry list (Table 1)	string	File name
4	First random sample, k	integer	$k \geq 1$
5	Last random sample, ℓ		$\ell \geq k$
6	Cluster expansion range	double	$[0, \infty)$
7	Areal sizes (Table 3)	string	Filename
8	Adjacency relations (Table 4)		
9	Boundaries (Table 5)		
10	Inter-regional distances (Table 6)		
11	Shortest-path sequences (Table 7)		
12	Input directory		Directory path
13	Output directory		
14	Input filename		Filename

Table 18: input_generate_spuriousclusters.txt

The "spurious" cluster schemes for the k-th through ℓ-th random location samples of the i-th through j-th industries are identified. The input directory specified at the twelfth line should coincide with the output folder in (the eighth line of Table 16) in Section 4.1, and the input filename specified at the last line in Table 18 should coincide with the output filename in (the last line of Table 16) in Section 4.1. The identified cluster schemes are saved in the output directory specified at the thirteenth line of Table 18.

4.3 Spuriousness Test

SpuriousnessTest.py computes p-value under the null hypothesis that the identified cluster scheme is spurious, i.e., the BIC value of the actual cluster scheme and those of the spurious cluster schemes come from the same statistical population. The p-value is computed as the share of spurious cluster schemes with larger BIC values than that of the actual cluster scheme. This program also generates a file containing a column of BIC values for spurious cluster schemes identified by GenerateSpuriousClusters in Section 4.2 for each industry. In the case of 163 Japanese three-digit manufacturing in 2001, the spuriousness of cluster schemes is not rejected for eight industries ("coke", and seven munitions industries).[20]

The inputs for this computation are the *event_log* files from *IdentifyClusters* in Section 3 and those from *GenerateSpuriousClusters* in Section 4.2 together with the industry list file described in Table 1.

5 Essential Containment

IdentifyEssentialContainment identifies the d-convex solid of the *essential clusters* (i.e., the most significant clusters in terms of the BIC contributions accounting for at least a given share of the total BIC values of the cluster scheme) in each connected subset of the basic regions.[21] If the set of the basic regions, R, consists of multiple disconnected regions, then the d-convex solidification of essential clusters is conducted within each connected subset of basic regions. For instance, our Japanese data consists of two disconnected subsets of basic regions, i.e., the subsets of basic regions inside and outside Hokkaido.

[20] In Mori and Smith [1], the spuriousness is not rejected also for "tobacco manufactures" industry at 0.05 level. But, under our newly generated spurious cluster schemes, the spuriousness has been rejected for this industry. But except for these nine industries, p-values are virtually zero.

[21] The definition of essential clusters here is simpler than the one introduced by Mori and Smith [1]. But, the program can be easily modified to incorporate alternative conditions, including the original definition by Mori and Smith [1].

The input parameters should be listed in the text file named "input_identify_ essentialcontainment.txt" containing the fourteen lines as in Table 19.

Line number	Input	Data type	Description
1	First industry, i	integer	$0 \leq i < \overline{I}$
2	Last industry, j		$i < j \leq \overline{I}$
3	Industry list (Table 1)	string	File name
4	Input directory (summary of cluster detection)	string	Directory
5	Input directory (cluster coverage)		
6	Output directory		
7	Cumulative BIC share	double	$[0, \infty)$
8	Establishment counts (Table 2)	string	File name
9	Areal sizes (Table 3)		
10	Adjacency relations (Table 4)		
11	Boundaries (Table 5)		
12	Inter-regional distances (Table 6)		
13	Shortest-path sequences (Table 7)		

Table 19: input_identify_essentialcontainment.txt

Many inputs (the first through third and eighth through thirteenth lines) are common to the inputs for the cluster detection listed in Table 9. The inputs in the fourth and fifth lines should coincide with the locations of the outputs from the cluster detection.

The double-valued cumulative BIC share, γ, for the essential clusters is specified at the seventh line. There are two output file for each industry from the present computation, "essential_containment_ cumulative_BIC_share_[$\gamma \times 100$]_ind_[industry ID].csv" and "summary_ essential_containment_cumulative_BIC_share_[$\gamma \times 100$]_ind_[industry ID].csv", to be stored in the directory indicated in the sixth line of Table 19. The former contains a two-column CSV data similar to Table 13. The second column in this output data lists the basic regions which belong to the essential containment of the clusters for the given industry. The basic regions in the second column with the common first-column value belong to the same connected portion of essential containment. Thus, the number of distinctive values in the first column coincides with the number of disconnected portions in the essential containment. In the

case of Japan, since our data consists of two disconnected subsets of basic regions (i.e., inside and outside Hokkaido), the maximum number of disconnected portions in the essential containment is two.[22]

The latter contains summary description of essential containment: the total number of clusters, the number of essential clusters, share of essential clusters (in terms of the number of clusters), the share of establishment counts in essential clusters, BIC share of essential clusters, areal share of essential clusters, the global extent of essential clusters (i.e., areal share of essential containment), and local density of essential clusters (i.e., areal share of essential clusters within the essential containment).

For example, the essential containment for $\gamma = 0.88$ of the cluster scheme of "livestock products" and that of "ophthalmic goods including frames" are shown as the yellow areas in Figures 5 and 6, respectively, where the clusters are also shown in the former.[23] The "livestock products" is a typical ubiquitous industry, and hence, the clusters are distributed widely across the country. The essential containment for this industry covers a large area (64.5%) within the country. On the other hand, "ophthalmic goods" industry is highly concentrated in Sabae region as indicated in the figure, and hence this region itself is the essential containment, covering only 3.5% of the national area.

[22]The values in the first column are only to distinguish disconnected portions of essential containment. Their specific value does not carry any other information.

[23]The correlation between the global extent and local density of essential containment across industries become minimum at γ =0.88. See Mori and Smith [1, §5.2] for the interpretation of these two measures of spatial concentration.

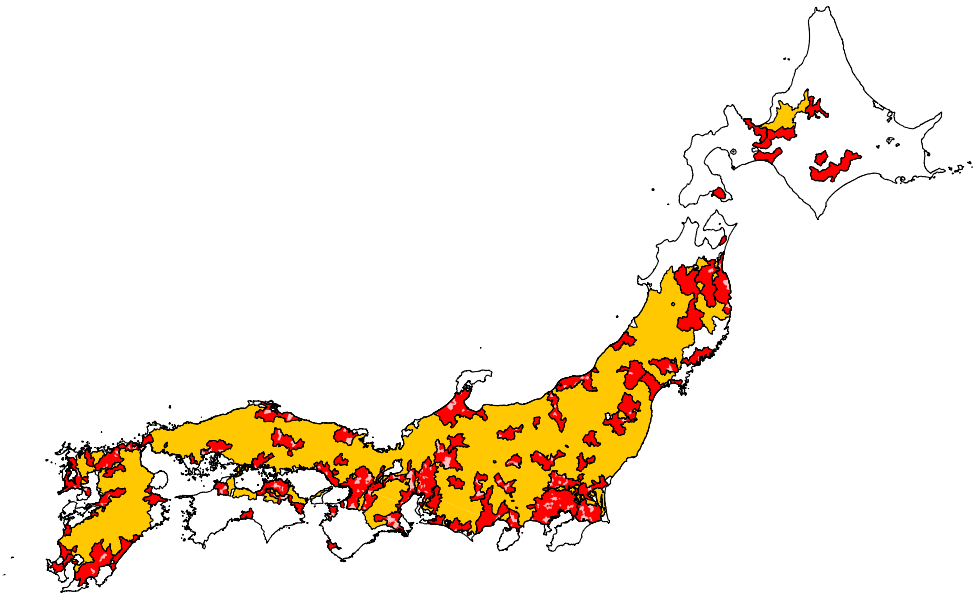


Figure 5: Essential containment of "livestock products" for $\gamma = 0.88$

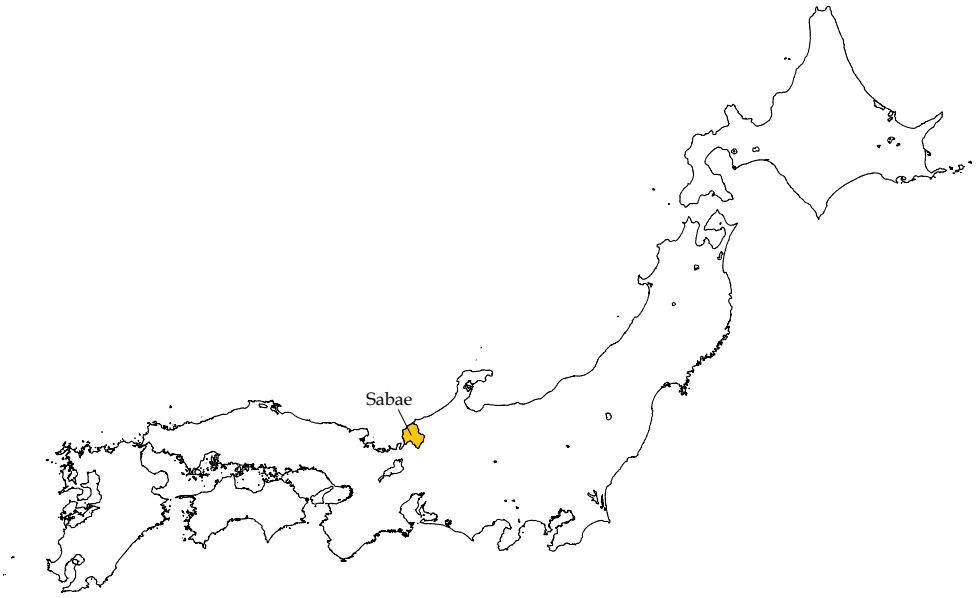


Figure 6: Essential containment of "ophthalmic good including frames" for $\gamma = 0.88$

6 Agglomerations

The python program, *FindAgglomerations.py*, identifies the agglomerations which is the refinement of clusters defined in Mori and Smith [1, §5.4]. Essentially, each *agglomeration* is defined by the single-peaked set of contiguous clusters where the peak is identified in terms of establishment density (or density of any other variables of interest, e.g., employment).[24] Establishment and employment distributions across clusters should be easily computed from the output of *IdentifyClusters* (Section 3) together with the establishment location patterns (Table 2).[25] These inputs for identifying agglomerations are specified at the beginning of *FindAgglomerations.py*.[26] The clusters of negative BIC contributions (those clusters with negative values for BIC_INC column in the summary log, Table 11) are excluded from the agglomerations.

The maps of agglomerations for "livestock products" and "ophthalmic goods including frames" are shown in Figures 7 and 8, respectively. In each figure, the agglomerations are distinguished by color, where a larger (smaller) concentration of establishments is indicated by the color closer to red (yellow).

[24] Moreover, each agglomeration is *filled*, i.e., there is no interior complement of the agglomeration in R.

[25] A Python program, *EstabEmpSizeOfClusters.py*, can be used for this task.

[26] The first inputs, *targets*, is a list of industry IDs for which agglomerations are to be identified. If it is empty, then agglomerations will be identified for all industries.

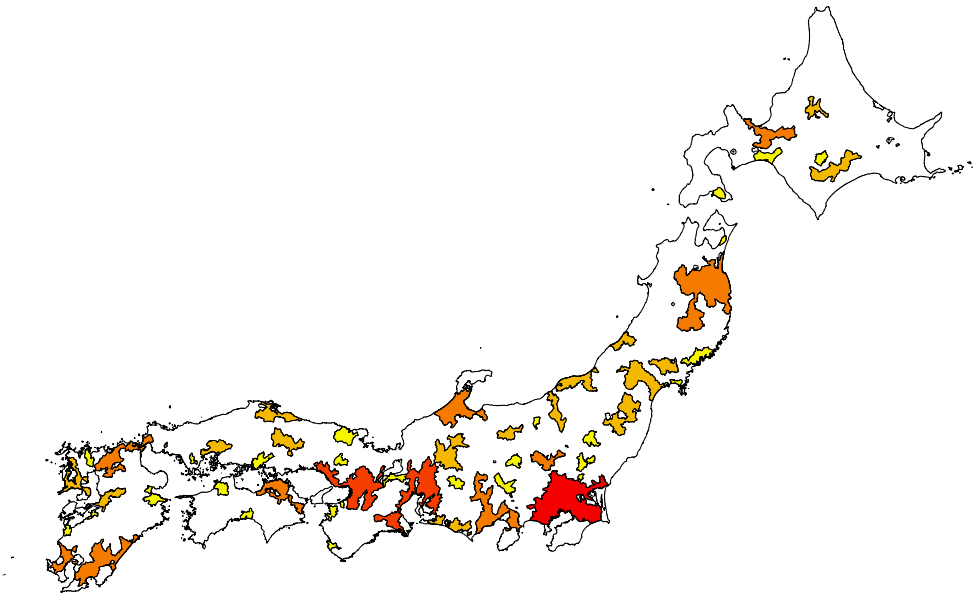


Figure 7: Agglomerations of “livestock products”

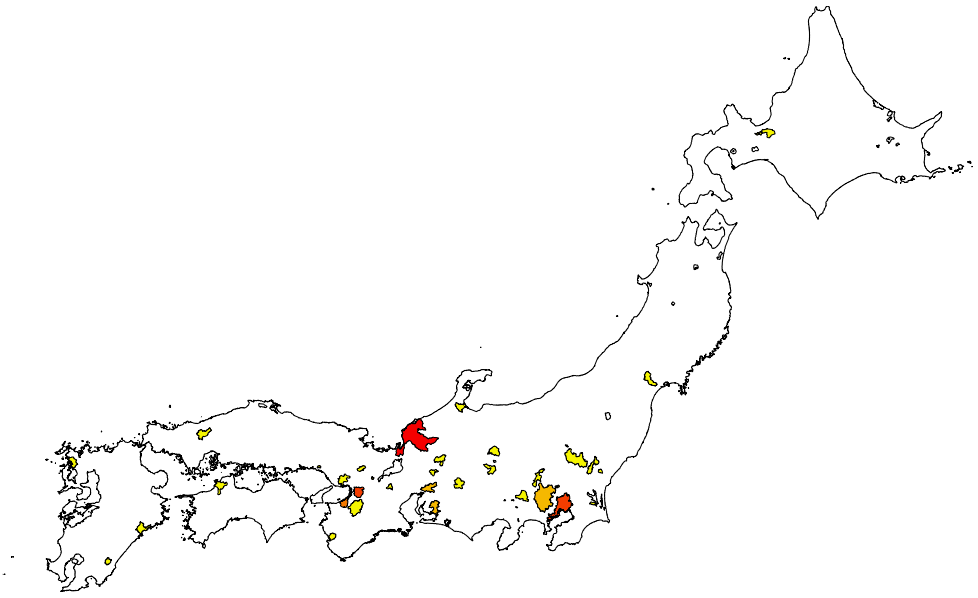


Figure 8: Agglomerations of “ophthalmic goods including frames”

7 Programs and Data

All the source codes of the C++ and Python programs, relevant input and output data, as well as the shape file of Japanese municipalities in 2001 are available from the website: http://www. mori.kier.kyoto-u.ac.jp/data/ cluster_detection.html.[27] In particular, by "joining" the outputs to the shape file of Japanese municipalities, the maps of clusters, agglomerations, and essential containments can be generated.

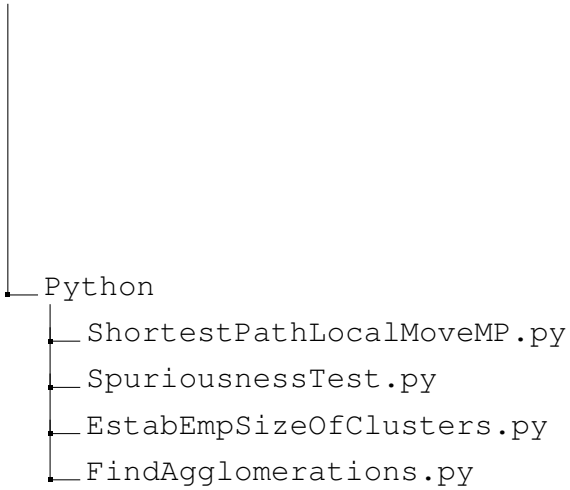
The "Program.zip" file contains all the C++ and Python programs. Under the `C++` folder, all the relevant C++ source codes are stored. The `Classes` folder contains the common source codes used across C++ programs. In each of the other subfolders, e.g., `IdentifyClusters`, contains the specific `main.cpp` codes to be compiled with the relevant C++ files in `Classes`.[28] An example makefile, *makefile*, in *IdentifyClusters* folder can be used to compile any other C++ programs by specifying the appropriate main.cpp file.[29] All the input files, e.g., input_detectclusters.txt, for C++ programs are stored directly under `C++` folder. All the Python programs are stored in the `Python` folder.

```
/Programs
└── C++
    ├── Classes
    ├── IdentifyClusters
    ├── IdentifyEssentialContainment
    ├── GenerateRandomLocations
    └── GenerateSpuriousClusters
```

[27] The sets of random location patterns and spurious cluster schemes are not uploaded on the web (due to their large size). The shape file is tested on ESRI's ArcGIS ver.10.2.

[28] Thus, in Apple's Xcode, in order to compile executable programs, the references to the relevant files in Classes should be explicitly added. In the case of the command-line terminals in Linux operating systems, the directory path to these relevant files should be explicitly specified in the makefile.

[29] TARGET variable in makefile is going to be the name of the executable file. Thus, depending on the program to be compiled, an appropriate name should be substituted.



References

[1] Mori, Tomoya and Tony E. Smith. 2014. "A probabilistic modeling approach to the detection of industrial agglomerations." *Journal of Economic Geography*, forthcoming.

[2] Mori, Tomoya and Tony E. Smith. 2011. "An industrial agglomeration approach to central place and city size regularities." *Journal of Regional Science* 51(4): 694-731.