Korean J. Chem. Eng., **26**(6), 1620-1629 (2009)
DOI: 10.1007/s11814-009-0280-x

Speed-up of the disaggregation of emission inventories and increased resolution of disaggregated maps using landuse data

Jong Ho Kim*, Byoung Kyu Kwak*, Chee Burm Shin, Hyeon-Soo Park***, Kyunghee Choi****, Sang Mok Lee****, and Jongheop Yi*,†**

*School of Chemical and Biological Engineering, Seoul National University, San 56-1, Sillim-9-dong, Gwanak-gu, Seoul 151-742, Korea
**Department of Chemical Engineering, Ajou University, Wonchen-dong, Yeongtong-gu, Suwon, Gyeonggi-do 433-749, Korea
***TO21, Lotte Tower 402-Ho, Sindaebang 2-dong, Dongjak-gu, Seoul 156-711, Korea
****National Institute of Environmental Research, Environment Research Complex, Gyeongseo-dong, Seo-gu, Incheon 404-170, Korea
(*Received 20 September 2008 • accepted 12 May 2009*)

Abstract−This study describes the full disaggregation process of emission inventory maps in support of environmental modeling studies in a geographical information system. Using a heuristic approach, appropriate algorithms were found to accelerate the computational disaggregation speed. The algorithms were based on scan-conversion algorithms employed in the field of computer graphics. Various algorithms were analyzed in terms of supporting emission inventories with different shapes, such as points, polylines and polygons. The algorithms were implemented by using Visual Basic, thereby enabling the efficiencies of the algorithms to be analyzed and compared with each other. For the disaggregation of polygon types, with the aim of increasing the resolution of an inventory map, we suggest the advanced polygon-disaggregation method with land use data. An air dispersion simulation was performed in order to compare the accuracy of the emission input data generated by existing disaggregation methods and the advanced method proposed in the present study.

Key words: Emissions, Disaggregation, Scan-conversion, Dispersion, GIS, Landuse

INTRODUCTION

Pollutant concentrations predicted by environmental quality models are highly sensitive to the spatial accuracy of the emission input data [1-3]. An integrated system of emission inventory and GIS has recently been proposed to integrate and analyze spatial information that contains emission trends, and to enhance the accuracy of the model input data [4-8].

An emission database in a GIS-based emission inventory consists mainly of points (e.g., stacks and incinerators), polylines (roads and trails), and polygons (administrative districts and industrial complexes) shaped in a vector format (hereafter 'vector'). This vector is suitable as the input data for some environmental quality models such as ISC3 [9]; however, several environmental quality models (e.g., the Urban Airshed Model (UAM) [10]) require emission data in raster format (hereafter 'raster'). Thus, the vectors used in emission inventories cannot be applied to environmental quality models. To solve this problem, the vector-formatted emission inventories must be resolved on a raster grid-cell network.

Several research groups have attempted to disaggregate emission inventories from vector to raster. For polyline-type emissions, Tuia et al. [11] suggested the simplified emission estimation model (SEEM) to disaggregate area-type hot traffic emissions to a road network at a grid cell size of 1 km; the authors demonstrated the spatial accuracy of SEEM [12]. Niemeier and Zheng [13] investigated the impact of grid resolution (from 1 to 16 km^2) on vehicle emission inventories. For polygon data, Dai and Rocke [14] suggested a spatial allocation method based on an area-weighted clipping method, and studied 150 square cells (cell size, 16 km^2). Dalvi et al. [15] developed a GIS environment based on the interpolation of large emissions (G-SMILE) for statistical models, using a $1^o \times 1^o$ grid cell size. From the GIS environment, it takes a relatively long time to disaggregate the emission inventory, as it commonly contains several hundred thousand points or vertices. In addition, higher grid resolution leads to increased computing time in geometrical progression [13], yet previous studies have not dealt with the computing speed of the disaggregation process. In the present study, we analyzed the computing speed required to enable the disaggregation of a large emission-inventory database with fine-scale grids in the range of several tens of meters to several hundred meters.

The disaggregation problem is similar to the scan-conversion problem encountered in the field of computer graphics. Many studies have sought to speed-up the scan-conversion process [16-27]; however, to the best of our knowledge, scan-conversion algorithms have yet to be applied to the assessment of emissions data. In this work, we applied various scan-conversion algorithms to disaggregate an emission inventory in a GIS environment, and analyzed the effectiveness of the algorithms by implementing them in Visual Basic 6.0.

Artificially created toxic chemicals are not emitted from natural areas of the earth surface such as surface water, vegetation, soil,

†To whom correspondence should be addressed.
E-mail: jyi@snu.ac.kr

and oceans; however, polygon-type vector emission sources (e.g., a national administrative district) contain such natural sites within individual polygons. A general disaggregation method only determines whether a grid point is enclosed by the edges of the polygon. This means that disaggregated emission data (as disaggregated by a general method) have limited spatial accuracy. To increase the spatial accuracy, we propose an advanced land use-supported disaggregation method for polygon-type emission sources.

This paper describes the practical application of disaggregation methods to an emission inventory formatted from the GIS environment for toluene emissions (0.01 km^2 grid cell size) in the area of Seoul, South Korea. Using the CALPUFF [28] model, we simulated the air dispersion of toluene using disaggregated emission maps (both general disaggregation maps and advanced maps compiled using the method proposed in this study). The dispersions of the calculated and observed concentrations were compared to validate the spatial accuracy of the disaggregation methods. The results of the comparison demonstrate that the landuse-based advanced method provides superior spatial accuracy compared with the general method.

DISAGGREGATION PROCESS

1. GIS Environment

This study considers three disaggregation processes, point-, polyline-, and polygon-emission sources, constructed in shapefile format (SHP) as an emission inventory database [29]. The SHP file format is a vector GIS file format developed by Environmental System Research Institute (ESRI), in the United States of America. This

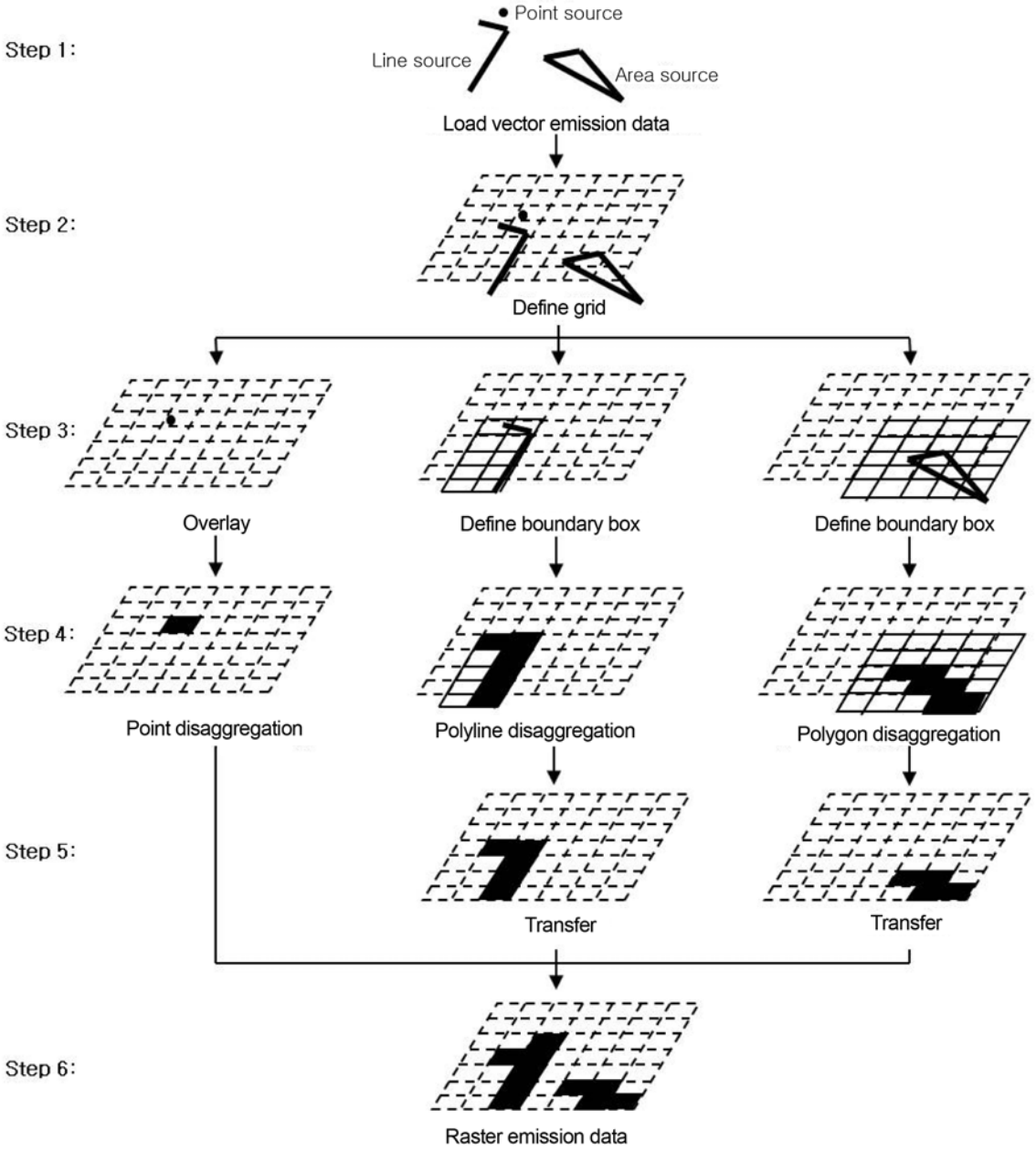


Fig. 1. Flow chart of the disaggregation process.

file format supports point, multi-point, polyline, polygon, and multi-patch sources. The SHP file consists of a main file (*.SHP), an index file (*.SHX), and a database (DB) table (*.DBF). The main file contains various spatial shapes of emission sources and their geographical coordinates. In the index file, each record contains the offset of the corresponding main file as recorded from the beginning of the main file. The DB table contains attributes such as emission identification, emission parameters, and emission rates. There exists a one-to-one relationship between geometry and attribute in these files, based on the record number.

2. Disaggregation of Emission Data

Fig. 1 shows the disaggregation process employed in allocating the vector emission data to the raster grid-cell network. The process involves the following six steps.

• Step 1: Load vector data

The process loads the GIS-based emission vector data to the computer memory. The vector data from the emission sources are read from an SHP file and the emission rate from a DBF table.

• Step 2: Define the grid

The boundary of the target domain and the cell size in the grid are then determined. The initial target domain and cell size are selected by the user, and the boundaries of the domain are enlarged slightly to avoid problems related to finite arithmetic:

$$\begin{aligned} x_{g\,min}&=x_{min} & x_{g\,min}&=x_{min} \\ y_{g\,min}&=y_{min} & y_{g\,min}&=y_{min} \\ x_{g\,min}&=x_{max}+e_x & x_{g\,max}&=x_{max}+e_x \\ y_{g\,min}&=y_{max}+e_x & y_{g\,max}&=y_{max}+e_y \end{aligned} \quad (1)$$

where x_{min}, y_{min}, x_{max}, and y_{max} are the boundary coordinates of the domain initially determined by the user; $x_{g\,min}$, $y_{g\,min}$, $x_{g\,max}$ and $y_{g\,max}$ are the coordinates of a enlarged rectangle; and e_x and e_y are error terms between the user-defined rectangle and the enlarged rectangle. When the user inputs the cell size (g), the number of cells is then determined as follows:

$$x_{ncell}=\frac{|x_{g\,max}-x_{g\,min}|}{g}+1 \qquad y_{ncell}=\frac{|y_{g\,max}-y_{g\,min}|}{g}+1 \quad (2)$$

where x_{ncell} and y_{ncell} are the number of cells; an exact square is used as the cell shape to simplify the process.

• Step 3: Classify the types of shapes and define the minimum boundary box

Different shapes require different disaggregation algorithms and routines. Therefore, in this step the loaded vector data are first classified as different types of shapes: points, polylines, or polygons.

To enable rapid progress in cases of polyline and polygon shapes, this system creates a minimum boundary box, which is defined by finding the minimum and maximum coordinate values of the vector data segment. This box reduces the computing time by avoiding calculation of the entire domain boundary. The SHP file contains the minimum boundary box. These data also must be matched to the grid; therefore, the minimum boundary box is also enlarged a little. The minimum boundary box does not have to be defined for point data.

• Step 4: Disaggregation

Disaggregation of vector emission data is performed by using the disaggregation algorithms in the minimum boundary box. Here, the emission rate in each cell is also calculated by allocating the emission rate of the vector segments. The algorithms are explained further in Section 3 below.

• Step 5: Transfer

The raster data, which are calculated in the minimum boundary box, are transferred into the grid cell. If the rasterized data (in the minimum boundary box) exceed the domain boundary, the exceeding data are eliminated.

• Step 6: Combine the raster data

The raster data (calculated from points, polylines, and polygons) are combined in a single grid boundary.

DISAGGREGATION ALGORITHM

The generation of raster images from point, polyline, and polygon vector segments on a digital plotter such as a monitor or printer is a fundamental task in the field of computer graphics. In the proposed system, the disaggregation process for an emission source type uses a similar principle to that employed in computer graphics to describe the topological relationship between the spatial representation and the emission rate. The graphics algorithm can be applied to convert the emission data format, despite some differences in factors such as the data characteristics and operating conditions. In this section, several disaggregation algorithms are used to convert the emission data format, and the effectiveness of each algorithm is assessed and compared.

1. Development and Test Environment

To analyze the effectiveness of the disaggregation algorithms, they were implemented with Microsoft Visual Basic 6.0 (VB6), with tests performed using an Intel Pentium 4 1.7 GHz Mobile Processor. Table 1 lists the execution times for 10 million repetitions of some basic operations. Traditionally, there is a set of general rules regarding computational speeds for different tasks. Addition and subtraction are generally faster than multiplication, which in turn is faster than division. Integer calculations are faster than floating-point calculations. In contrast, with the VB6 compiler, the four arithmetic operations have similar calculation times, and VB6 is unable to adopt any bit operations (e.g., shift operation for calculating additions). This means that the effectiveness of the different algorithms depends on the compiler; in fact, the characteristics of the compiler lead to contrasting results between theoretical and practical effectiveness, as shown in the following section.

2. Point Data

Point data are simply allocated in a grid according to the following equations:

$$x_{grid}=\frac{|x-x_{g\,min}|}{g} \qquad y_{grid}=\frac{|y-y_{g\,min}|}{g} \qquad E_{grid}=\frac{E_{vector}}{g^2} \quad (3)$$

Table 1. CPU time (s) required for 10 million repetitions of basic operations using Visual Basic 6.0

Operation \ Variable type	Long (4 bite integer)	Double (8 bite float-point)
Add (+)	0.312	0.328
Subtract (−)	0.265	0.281
Multiply (*)	0.265	0.281
Divide (/)	0.265	0.281

where x and y are the point coordinates of the vector data, x_{grid} and y_{grid} are the coordinates of a point source matched with the grid, E_{vector} is the emission rate (kg/yr) of the vector data, and E_{grid} is the emission rate (kg/yr·m^2) of the contained grid. In point disaggregation, additional calculations of the emission rate allocation and boundary box are not required. In the case of Korea, there exist about 3,000 point sources for toxic chemical emission inventories. The computing time required for 3,000 points sources is about 0.14 μs, posing no problem for the disaggregation process.

3. Line Data

Existing algorithms for line disaggregation can be broadly classified into three groups: incremental algorithms, packing algorithms, and algorithms that employ the symmetry principle. The first incremental algorithm was devised by Bresenham [17]. This well-known algorithm uses only integer arithmetic, which removes real numbers and division operations in the computation to ensure the prompt generation of line segments. Pitteway [24] proposed a formulation, known as the midpoint technique, which differed from Bresenham's incremental algorithm and was subsequently adapted by Aken [16] and Kappel [22], among others.

Many subsequent researchers have proposed methods for packing the calculation steps to reduce the number of incremental steps. Wu and Rokne [30] developed the double-step method, which was then expanded to the multi-step technique [19]. The main idea behind the multi-step technique is to choose one pattern (a set of points) among different possible patterns. The run-length algorithm has also been proven in several studies [20,31]. In this algorithm, the

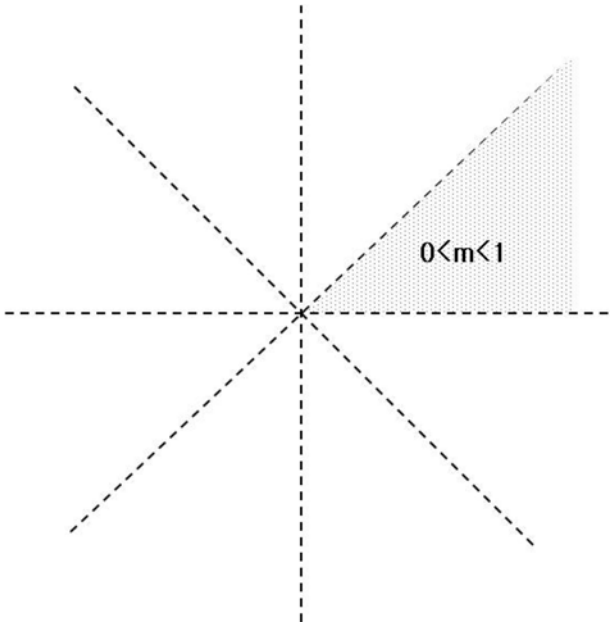


Fig. 3. Details of different line-disaggregation algorithms.

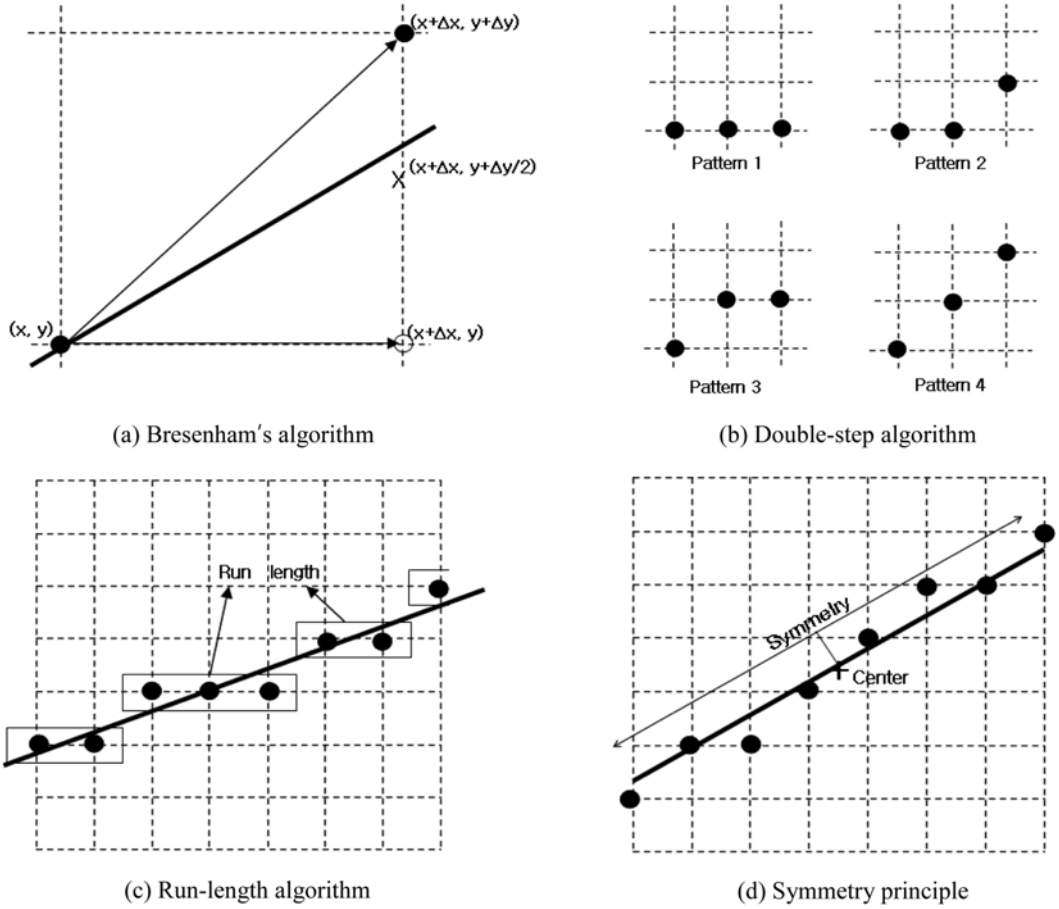


Fig. 2. Coordinate system employed in our explanation of disaggregation.

incremental step is reduced by determining the run-length.

Gardner [18] proposed a new approach to speed up the line scan-conversion process, based on the symmetry property of a line: the scan-converted line is symmetric about the center point of the line. The symmetry principle enables the calculation of only half of the line.

In performing line disaggregation, we tested three algorithms: Bresenham's algorithm, the double-step method, and the run-length algorithm employing the symmetry principle. We discuss the line disaggregation problem in the first octant (slopes between 0 and 1) of the Cartesian coordinate system, as shown in Fig. 2. The algorithm is easily extended to other octants. The disaggregation algorithms for lines are described below.

• Bresenham's algorithm

Fig. 3(a) shows the principle behind Bresenham's algorithm. We assume a starting point (x, y). In the first octant, there are two basic optional moves in the raster grid field: the diagonal move (x+dx, y+dy) and the horizontal move (x+dx, y). The optional moves can be approximated by the relative positions of the mid-point (x+dx, y+dy/2) and the line (bold line in Fig. 3(a)). The relative position is determined by testing the sign of a function known as a discriminator.

• Double-step method

The basic idea of the algorithm in the double-step method is to consider two steps at a time (rather than one). If we have a drawn pixel (x, y), the next two pixels can be selected from four possible fixed patterns, as shown in Fig. 3(b). In Bresenham's algorithm, the sign of a discriminator predicts the pixel to be chosen at any step. In contrast, the double-step method calculates the discriminator for every two steps. Theoretically, calculations can be performed at almost twice the speed of that possible using Bresenham's algorithm.

• Run-length algorithm

A run is defined as a contiguous set of pixels with the same X or Y coordinate. Consider a line from (x1, y1) to (x2, y2) on the raster grid field, where x2>y2. If we consider lines with slopes from 1/2 to 1/3, the run-length must be 2 or 3, as shown in Fig. 3(c). The line can be viewed as the series of the run. The calculation speed of this algorithm depends on the slope of the line.

• Symmetry principle

Fig. 3(d) describes the symmetry principle of a line. The raster pixel related to the vector line has a symmetric pixel pattern based on the center of the line. This principle can be applied to the disaggregation in combination with other algorithms. Theoretically, this algorithm can almost double the calculation speed of the original disaggregation algorithm.

After using the above algorithms to find the cell crossed by the line, the emission rate in each cell is allocated as follows:

$$E_{grid} = \frac{E_{vector}}{g^2 n_c} \quad (4)$$

where n_c is the number of grid cells crossed by the line.

The graph in Fig. 4 shows that the CPU (central processing unit) time required to implement line-disaggregation algorithms depends on the slope of the line. The test was performed with 100,000 line segments and 100×100 grid cells. Bresenham's algorithm was faster than the step-packing methods (the double-step and run-length algorithms). Theoretically, the double-step algorithm and run-length algorithm should be faster than Bresenham's incremental algorithm. When implemented in Visual Basic 6.0, however, the performances of the different algorithms are different from those predicted theoretically due to compiler properties. If bit operations were to be adopted in this implementation using a C and C++ compiler, the packing algorithms would be faster than the incremental algorithm. The symmetry algorithm applied to the line-disaggregation algorithms showed a slightly faster CPU time than that recorded by the algorithms that did not employ the symmetry principle. Theoretically, the symmetry algorithm would be calculated twice as fast as the original algorithm, although this is also dependent on the properties of the compiler.

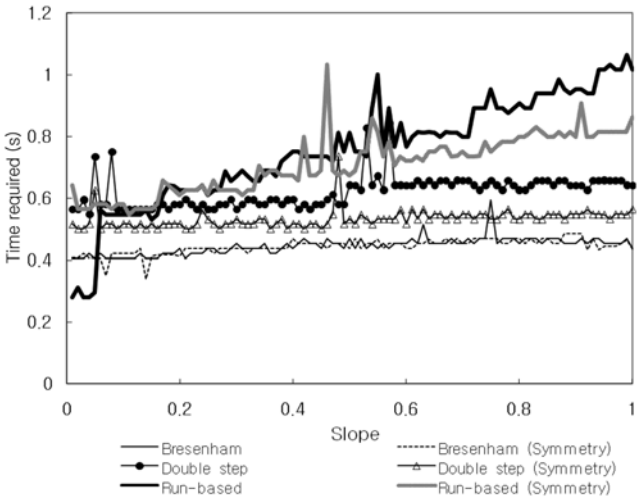


Fig. 4. CPU times required for implementing different line-disaggregation algorithms.

In the case of 2.5 million line segments with a grid cell size of 0.01 km^2, similar to the GIS road network in Korea, the result is obtained in just 1 second by using any of the considered algorithms. We transplanted the simple Bresenham's symmetry algorithm to our system.

4. Polygon Data

Many scientists have proposed polygon disaggregation algorithms, including a sign-of-offset method, a sum-of-area method, a grid-based algorithm [21], a ray-crossing algorithm [23], a sum-of-angles method [25], a swath method [26], a wedge method [27] and a winding number rule [32]. Most of the above algorithms run effectively if the polygons under consideration are convex; however, some are not applicable to the disaggregation of a concave polygon, and some are numerically unstable (e.g., the sum-of-angles method is strongly affected by rounding errors).

Concave polygons with a vertex defined in float coordinates are commonly observed in the GIS environment. In this context, the algorithms listed below were tested to solve the emission disaggregation problem. In this article, O notation is used to indicate the degree of complexity of the algorithm, and n is the number of polygon edges. O(n) indicates that the CPU time of the algorithm is proportional to n.

• Ray-crossing Algorithm

The ray-crossing algorithm is the most commonly used algorithm. Fig. 5(a) shows the principle of this algorithm. Imagine an

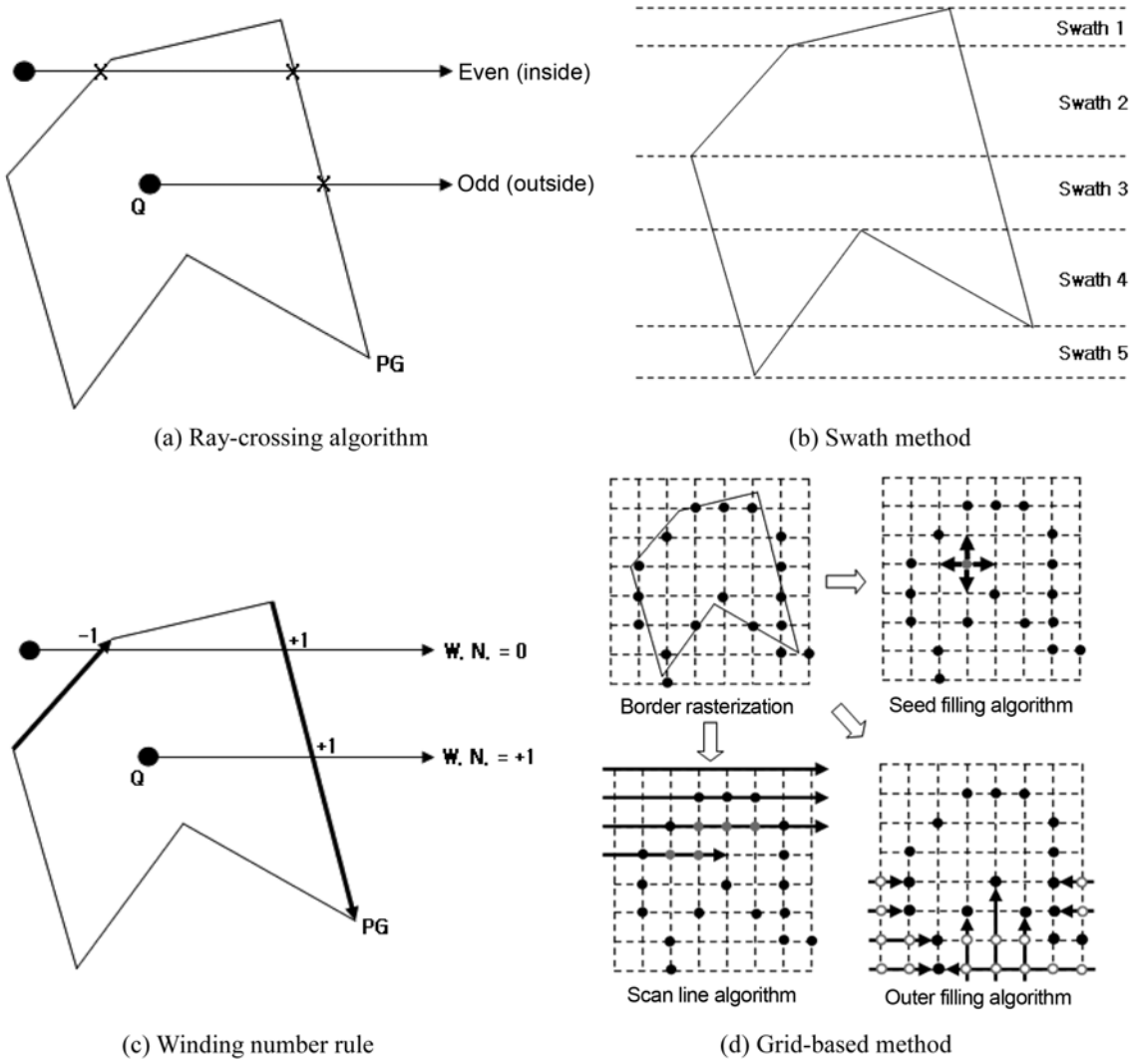


Fig. 5. Details of different polygon-disaggregation algorithms.

infinite line starting at point Q, then count the number of intersections made by the line and the edges of the polygon. If the number of intersections on either side of point Q is an odd number, point Q is inside the polygon PG. This algorithm successfully solves all kinds of polygons, in O(n) time. For implementation, some special situations must be considered (e.g., point Q lying on one of the polygon edges, or the ray intersecting the polygon at a node).

• Swath method

The swath algorithm uses the ray-crossing algorithm, but several pre-processing procedures are required. The polygon is first divided into swaths, as shown in Fig. 5(b). This procedure is completed in O(nlogn) time. Each swath has at least two edges, with a maximum of n−1 edges. Using a binary search, the swath is located for the considered point in time O(logn); the ray-crossing method is then applied to the polygon edges located in the swath. This algorithm is useful when more points must be matched against the same polygon.

• Winding number rule

The winding number accurately determines if a point is inside a non-simple closed polygon. It is performed by computing how many times the polygon winds around the point. A point is outside only when the polygon does not wind around the point at all (i.e., the winding number is zero). This algorithm requires O(n) time.

• Grid-based method

The grid-based method is suitable for a raster-based situation. The polygon is represented by a group of cells; thus, the algorithm requires pre-processing for a vector-based situation. The polygon border is first rasterized by the line-disaggregation algorithm (black circles in Fig. 5(d)). This is completed in time O(n). The inner or outer cells are then determined by various algorithms such as a seed-fill algorithm, a scan-line algorithm, or a border-start algorithm; these algorithms are explained in Fig. 5(d). The seed-fill algorithm (also known as a 'flood-fill algorithm') plants a seed, and progressively more seeds are recursively planted around the first seed. Each new seed is responsible for finding the inner cell. The core function of the scan-line algorithm is detection of the spans of the scan-line lying within the polygon, based on the following process. The algorithm identifies the intersections of the scan-line with all the edges of the polygon, sorts the intersections, and fills in all the pixels between pairs of intersections. The border-start algorithm operates by mov-

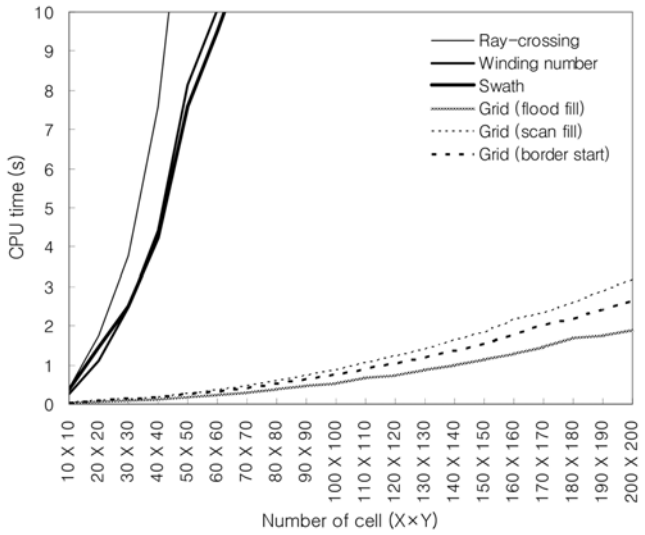


Fig. 6. CPU times required for implementing different polygon-disaggregation algorithms.

ing from the borders of the minimum boundary box in the left, right, down, or up directions, and then finds the outer cells until it encounters the edge of the polygon.

Fig. 6 shows that the CPU time required for polygon-disaggregation algorithms depends on the number of grid cells. The time required was determined by using a square shape with 1000 repetitions. The grid-based algorithm was faster than the vector-based algorithm, in agreement with Zalik and Kolingerova's [27] result. Among the grid-based algorithms, the seed-fill algorithm was the fastest, but requires many stack memories because it is a recursive process. This algorithm did not operate when we created many grid cells. The scan fill algorithm was also fast; however, the correct edge intersection count is heavily dependent on the topological distribution of the polygon vertices. Situations that require additional analysis and computation include concavities, self-crossing edges, and edges parallel to the scan direction. The CPU time for the grid scan-fill algorithm in Fig. 6 was obtained for an implementation that did not check a complex topological distribution. This algorithm is too cumbersome for additional analysis. The border-start algorithm was also fast, and successfully converted the vector data to raster data. We used the grid-based border start algorithm for polygon disaggregation.

In the case of a polygon, the emission rate in each cell is allocated as follows:

$$E_{grid} = \frac{E_{vector}}{g^2 n_e} \tag{5}$$

where n_e is the number of grid cells enclosed by the polygon.

The administrative district (polygon) of GIS data for South Korea consists of 300 segments, 2,580,000 vertices, and an average boundary box size of 26×26 km. To disaggregate this polygon data to a 100×100 m raster map using the ray-crossing algorithm or grid-based method requires about 350 h or about 1 min, respectively. The ray-crossing algorithm is clearly unsuitable for such a GIS environment. Although the grid-based polygon-disaggregation method requires only 1 min, it represents a bottleneck in the disaggregation process compared with the disaggregation of points (requiring 0.14

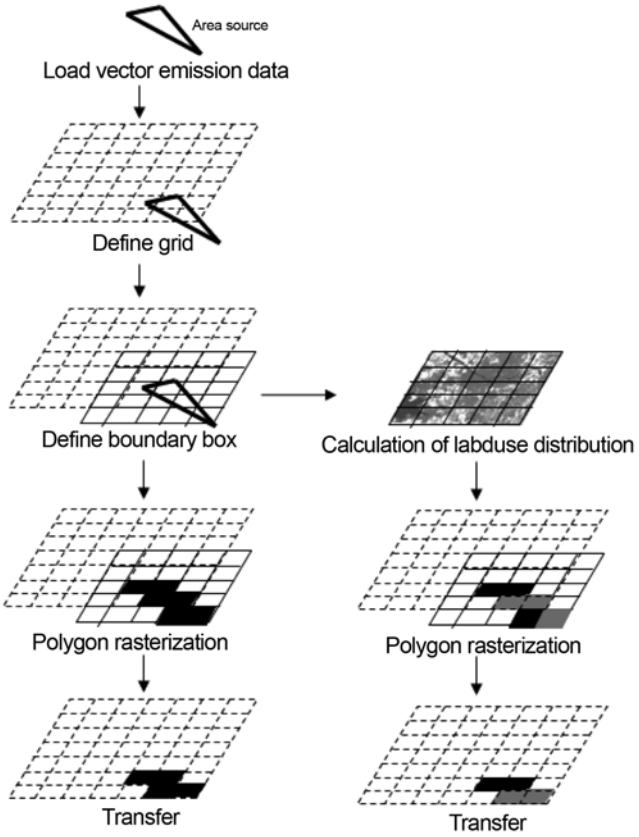


Fig. 7. Flow chart of the polygon disaggregation process.

Table 2. Categories employed in landuse maps

Category 1	Category 2
Human living area	House
	Industry
	Business
	Leisure
	Transport
	Public
Agricultural area	Paddy
	Patch
	Vinyl house
	Orchard
	Cultivation etc.
Forest	Broadleaf
	Needle-leaf
	Complex forest
Grassy place	Grassland
	Golf links
	Grassland etc.
Marsh place	Inland marshy
	Coast marshy
Soil	Mining
	Bare ground
Water	Surface water
	Sea

μs) and polylines (1 s).

LANDUSE-SUPPORTED DISAGGREGATION METHOD

1. Methodology

To increase spatial accuracy, we propose an advanced polygon-disaggregation method (landuse-supported method). We assumed that among the different landuse categories, toxic chemicals are only released in human living areas. In our case, the landuse map was acquired from the Environmental Geographic Information System (http://egis.me.go.kr) managed by the Ministry of Environment, Korea. The landuse map consists of 23 categories, with a cell size of 30×30 m. The categories are listed in Table 2, with those categories containing human living areas emphasized by bold type. The advanced method starts from the general polygon-disaggregation method (grid-based method) introduced in Section 2 and Fig. 1, to which one process is added. Following definition of the boundary box, the distribution ratio of the target landuse is calculated on each grid with reference to the landuse map. The disaggregation process is then performed with a spatial allocation of the emission rate obtained by the following improved form of Eq. (5):

$$E_{grid} = \frac{\mu E_{vector}}{g^2 n} \tag{6}$$

where μ is the proportion of human living areas among the landuse categories.

2. Disaggregated Map

To compare the disaggregated maps with and without consideration of landuse, we tested the two methods in terms of toluene emissions around Seoul, the largest city in Korea. Fig. 8 shows the study area, as expressed in the Universal Transverse Mercator (UTM) coordinate system. The study area extends from 289 to 344 km on the x coordinate and 4,125 to 4,176 km on the y coordinate of the 52 N UTM. This site is the major source of toluene emissions in East Asia, and has a population of about 24 million (about half the popula-

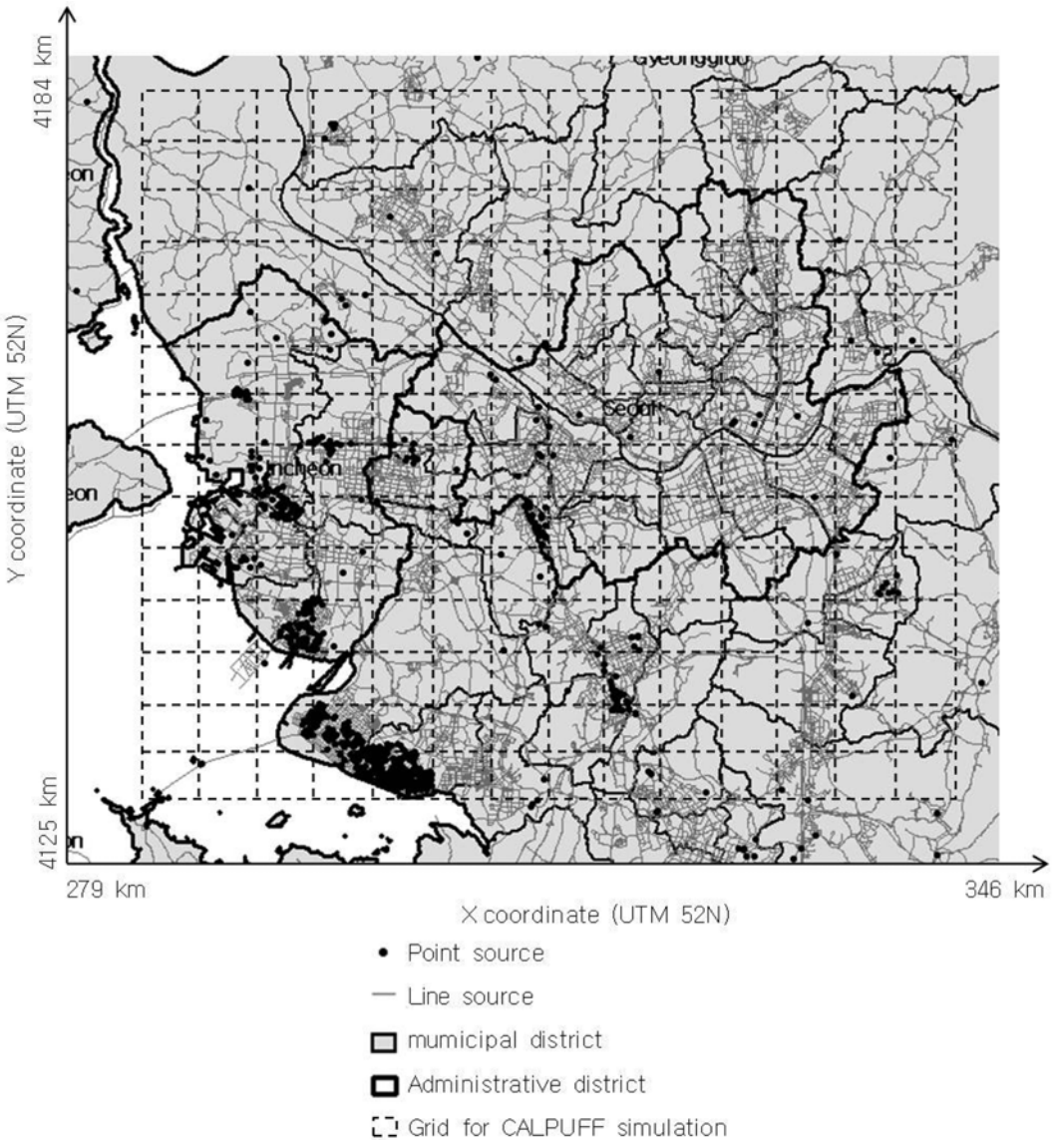
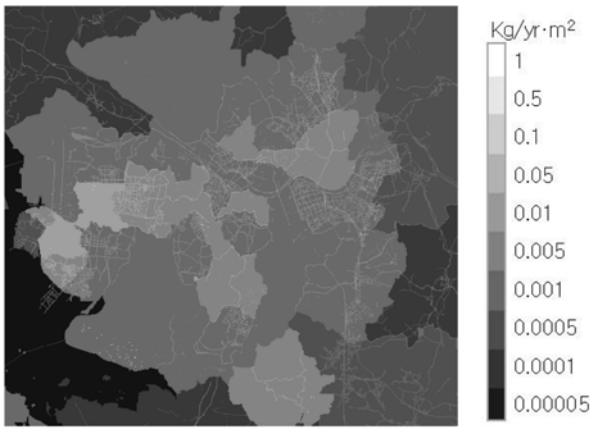
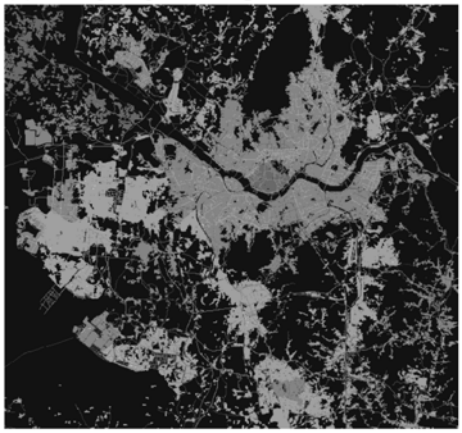


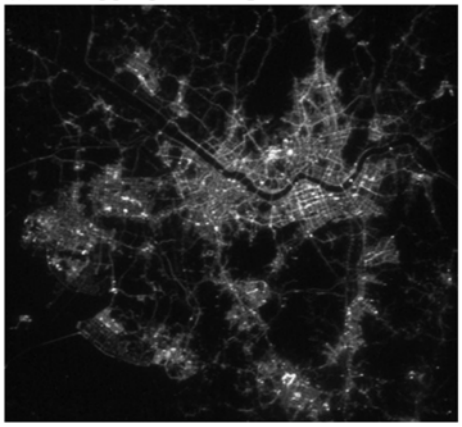
Fig. 8. Study area considered for disaggregation tests using a CALPUFF simulation.



(a) Disagreegated map without landuse



(b) Disaggregated map with landuse



(c) Modified Satellite image in capital region (The origin of picture: The Gateway to Astronaut Photography of Earth (http://eol.jsc.nasa.gov/))

Fig. 9. Disaggregated maps of emission data (as calculated using the general (a) and land use-supported (b) disaggregation models) and nighttime satellite image (c) of the Seoul region.

tion of South Korea). The emission rate of toluene was obtained from previous publications [33-35].

Fig. 9(a) shows the map that is spatially allocated using the general disaggregation method without landuse data; Fig. 9(b) shows the map allocated using the landuse-supported disaggregation process. A grid cell size of 100×100 m (0.01 km^2) was chosen for this test. Fig. 9(c) is a modified nighttime satellite image (modified to a grayscale image) of the study area obtained from a NASA Web site (The Gateway to Astronaut Photography of Earth; http://eol.jsc.nasa.gov/). The emission pattern in Fig. 9(b) is much more similar to the satellite image than is Fig. 9(a). This result indirectly demonstrates that the landuse-supported disaggregation method has greater spatial accuracy than the general method.

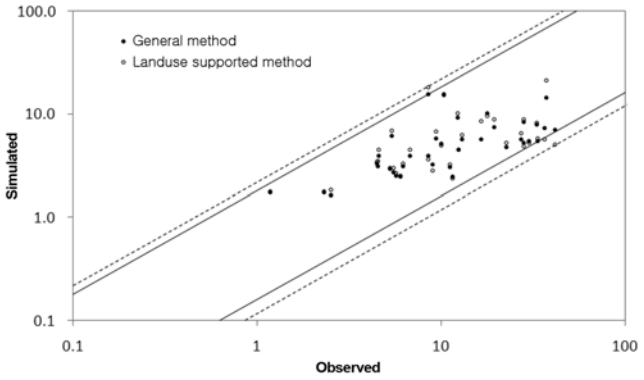


Fig. 10. Results of CALPUFF modeling, comparing observed data with calculated results of the general and landuse-supported disaggregation methods.

2. Simulation using CALPUFF

We performed a CALPUFF simulation to demonstrate the differences between the general and landuse-supported disaggregation methods. Emission input data for the CALPUFF model are 14×14 grid cell emissions (see Fig. 8). The disaggregated emission map (100×100 m grid cell size) was rearranged to 14×14 grid cells to produce the input data. The simulation time is the year 2004. To compare the calculated and observed results, observation data were collected from 20 sites during 2004. Observations were made seasonally, with air sampled at 3-h intervals at each site.

The spatial accuracy of emission data was validated by comparing the model calculations with monitoring data. Fig. 10 shows a site- and time-matched point-to-point comparison of the observed and calculated concentrations (for both the general and landuse-supported disaggregation maps). The calculated results derived from both methods are in good overall agreement with the monitoring data, although slightly underestimated, probably because of background concentrations. The values calculated using the landuse-supported disaggregation method lie closer to the center of the graph than the values obtained by the general disaggregation method. The simulated results obtained with the landuse-supported disaggregation method fall within a 1.8-fold difference of corresponding observed values in the upper direction (simulated/observed) and within a 6.1-fold difference in the lower direction (solid lines in Fig. 10). The results obtained by the general method fall within 2.2-fold differences in the upper direction and within 8.2-fold difference in the lower direction (dashed lines in Fig. 10). These findings suggest that the advanced disaggregation method yields a more accurate result than the general method.

CONCLUSIONS

We introduced a disaggregation process and related methods to

disaggregate and allocate the emission rate from a vector-format emission inventory to a grid network in a GIS environment. To reduce the CPU time required for the disaggregation process, we applied various scan-conversion algorithms (as commonly employed in the field of computer graphics) and tested the effectiveness of each algorithm. The algorithms were implemented with Visual Basic 6.0. The most suitable algorithms identified from the test data were the point allocation algorithm for point disaggregation, Bresenham's symmetry algorithm for line disaggregation, and the grid-based border start algorithm for polygon disaggregation. A practical analysis of an emission inventory in Korea revealed that the polygon-disaggregation step is a bottleneck in the process rather than point or polyline disaggregation. To overcome the problem of a limitation on spatial resolution due to polygon disaggregation, we proposed the landuse-based advanced disaggregation method. A comparison study of the general and landuse-based disaggregation methods was undertaken by CALPUFF modeling. From the modeled and observed results, considerable differences were observed between the two methods, with the advanced method providing superior spatial accuracy. The analysis of CPU time and the proposed disaggregation method could help in the pre-processing of emission data and could be applied to various environmental quality models.

ACKNOWLEDGMENT

We gratefully acknowledge financial support provided by the ECO-Technopia-21 Project of the Ministry of Environment, Korea.

REFERENCES

1. R. Berkowicz, M. Winther and M. Ketzel, *Environ. Modell. Softw.*, **21**, 454 (2006).
2. A. Jeaecker-Voirol and P. Pelt, *Environ. Modell. Softw.*, **15**, 575 (2000).
3. F. Monforti and A. Pederzoli, *Environ. Modell. Softw.*, **20**, 505 (2005).
4. J. M. Armstrong and A. M. Khan, *Comput. Environ. Urban*, **28**, 421 (2004).
5. J. Hao, H. Tian and Y. Lu, *Environ. Sci. Technol.*, **36**, 552 (2002).
6. P. Symonidis, I. Ziomas and A. Proyou, *Environ. Modell. Softw.*, **19**, 413 (2004).
7. D. Q. Tong, N. Z. Muller, D. L. Mauzerall and R. O. Mendelsohn, *Environ. Sci. Technol.*, **40**, 1395 (2006).
8. W. Winiwarter and G. Schimak, *Environ. Modell. Softw.*, **20**, 1469 (2005).
9. Lakes Environmental Software, Inc., User's guide for the Industrial Source Complex (ISC3) dispersion models, http://support.lakes-environmental.com/Models/ISC3Docs/isc3v1.pdf (2006).
10. T. W. Tesche, R. M. Morris, G. Tonnesen, D. McNally, J. Boylan and P. Brewer, *Atmos. Environ.*, **40**, 4906 (2006).
11. D. Tuia, M. Osses de Eicker, R. Zah, M. Osses, E. Zarate and A. Clappier, *Atmos. Environ.*, **41**, 3658 (2007).
12. M. Osses de Eicker, R. Zah, R. Trivino and H. Hurni, *Atmos. Environ.*, **42**, 1491 (2008).
13. D. A. Niemeier and Y. Zheng, *Environ. Sci. Technol.*, **38**, 2133 (2004).
14. J. Dai and D. M. Rocke, *Environ. Modell. Softw.*, **15**, 293 (2000).
15. M. Dalvi, G. Bieg, U. Patil, A. Kaginalkar, C. Sharma and A. P. Mitra, *Atmos. Environ.*, **40**, 2995 (2006).
16. V. Aken JR., *IEEE Comput. Graph.*, **4**, 24 (1984).
17. J. E. Bresenham, *IBM Syst. J.*, **4**, 25 (1965).
18. P. L. Gardner, *IBM Technol. Disclosure B.*, **18**, 1595 (1975).
19. G. Gill, *IEEE Comput. Graph.*, **14**, 66 (1994).
20. A. Haque, M. S. Rahman, M. Bakht and M. Kaykobad, *Comput. Graph.*, **30**, 207 (2006).
21. C.-W. Huang and T.-Y. Shih, *Comput. Geosci.*, **23**, 109 (1997).
22. M. R. Kappel, *An ellipse-drawing algorithm for raster displays. In: Fundamental algorithms for computer graphics*, Springer: Berlin (1985).
23. U. Manber, *Introduction to algorithms; a creative approach*, Addison-Wesley Publishing Co., Reading, Massachusetts (1989).
24. M. L. V. Pitteway, *Comput. J.*, **10**, 282 (1967).
25. F. P. Preparata and M. I. Shamos, *Computational geometry: An introduction*, 2nd ed. Springer, New York (1985).
26. B. Zalik and G. J. Clapworthy, *Comput. and Graph.*, **23**, 353 (1999).
27. B. Zalik and I. Kolingerova, *Comput. Geosci.*, **27**, 1135 (2001).
28. TRC Companies, Inc., A user's guide for the CALPUFF dispersion model, http://www.src.com/verio/download/CALPUFF_UsersGuide.pdf (2006).
29. Environmental Systems Research Institute, Inc., ESRI Shapefile Technical Description, printed in the United States of America, http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf (1998).
30. X. Wu and J. G. Rokne, *Comput. Graph. Image Anal.*, **37**, 331 (1987).
31. P. Stephenson and B. Litow, *Comput. Graph.*, **25**, 681 (2001).
32. F. Xia, *Pattern Recogn.*, **36**, 1383 (2003).
33. H. Park, S. Chah, E. Choi, H. Kim and J. Yi, *Atmos. Environ.*, **36**, 4851 (2002).
34. M.-S. Kim, J. H. Kim, H.-S. Park, Y. S. Sun, H.-S. Kim, K. H. Choi and J. Yi, *Korean J. Chem. Eng.*, **23**, 919 (2006).
35. M.-S. Kim, J. H. Kim, H.-S. Park, Y. S. Sun, H.-S. Kim, K. H. Choi and J. Yi, *Korean J. Chem. Eng.*, **24**, 763 (2007).