KIGAS Vol. 22, No. 6, pp 136~143, 2018
(Journal of the Korean Institute of Gas)
https://doi.org/10.7842/kigas.2018.22.6.136

사고로 인한 유해화학물질 누출확산의 대응을 위한 Cellular Automata기반의 시뮬레이션 시스템

폴 신* · 김창완 · 곽동호 · 윤인섭** · †김태옥
*한국외국인학교, **서울대학교 EDRC, 명지대학교 화학공학과
(2018년 11월 14일 접수, 2018년 12월 21일 수정, 2018년 12월 22일 채택)

Cellular Automata Simulation System for Emergency Response to the Dispersion of Accidental Chemical Releases

Insup Paul Shin* · Chang Won Kim · Dongho Kwak · En Sup Yoon · †Tae-Ok Kim**
**Korea International School, Seongnam, Gyeonggi-do 13543, Korea*
***Engineering Development Research Center, Seoul National University, Seoul 08826, Korea*
Department of Chemical Engineering, Myongji University, Yongin, Gyeonggi-do 17058, Korea
(Received November 14, 2018; Revised December 21, 2018; Accepted December 22, 2018)

요 약

Cellular automata는 천체물리, 사회현상, 화재 확산 및 피난 등 많은 분야의 시뮬레이션에 활용되고 있다. 본 연구는 빈번히 발생하고 있는 화학사고에 대비한, 위험성평가 및 비상대응계획 작성시 요구되는 화학물질 확산 시뮬레이션을 위한 보급용 모델을 cellular automata를 기반으로 개발하였다. 상세한 플랜트 안전설계용과는 달리, 실시간 사고대응을 위해선 빠른 계산과 더불어 피해영역 분포의 불확실성을 줄이기 위한 반복 계산이 요구된다. EPA ALOHA, 화학물질안전원 KORA 등이 있지만, 지속적인 모델과 코드의 보완이 가능하고, 중소기업용의 무료 S/W개발에 본 연구의 차별성이 있다. 계산시간이 많이 요구되는 full-scale CFD에 비해 상대적인 정확도의 손실은 감수하고, 특히 일반 사용자의 편리성을 도모하였다. 기상청 기상정보 연계를 비롯해, Python open-source 라이브러리들을 활용해, 기능 확장 및 지속적인 update가 가능하며, 사용자는 해당 플랜트의 지형도와 사용 물질의 입력만으로 쉽게 결과를 얻을 수 있다. Full-scale CFD 시뮬레이션과 대비해 정확도를 확인하였으며, 빠른 계산을 위해 GPU를 활용하는 open source software로 배포될 예정이다.

Abstract - Cellular automata have been applied to simulations in many fields such as astrophysics, social phenomena, fire spread, and evacuation. Using cellular automata, this study develops a model for consequence analysis of the dispersion of hazardous chemicals, which is required for risk assessments of and emergency responses for frequent chemical accidents. Unlike in cases of detailed plant safety design, real-time accident responses require fast and iterative calculations to reduce the uncertainty of the distribution of damage within the affected area. EPA ALOHA and KORA of National Institute of Chemical Safety have been popular choices for these analyses. However, this study proposes an initiative to supplement the model and code continuously and is different in its development of free software, specialized for small and medium enterprises. Compared to the full-scale computational fluid dynamics (CFD), which requires large amounts of computation time, the relative accuracy loss is compromised, and the convenience of the general user is improved. Using Python open-source libraries as well as meteorological information linkage, it is made possible to expand and update the functions continuously. Users can easily obtain the results by simply inputting the layout of the plant and the materials used. Accuracy is verified against full-scale CFD simulations, and it will be distributed as open source software, supporting GPU-accelerated computing for fast computation.

Key words : cellular automata, gas dispersion, chemical accident, open source simulation software

†Corresponding author:kimto@mju.ac.kr

1. Introduction

Accidental chemical releases often lead to serious accidents due to leakage of toxic, combustible and explosive materials through various paths such as tanks, pipes, cracks, pumps, valves, and flange leaks. In addition to the damage caused by toxic chemicals, spilled chemicals may form explosive vapors and explode.

The dispersion model explains the phenomenon in which gaseous chemicals are transferred from the accident site to the factory or other nearby areas. After spillage, the toxic material moves away due to wind in the form of plume or puff (see Fig. 1). Since toxic substances disperse and rapidly mix with air, their concentrations decrease as wind blows from the source of the leak. Several parameters affect the atmospheric dispersion of toxic substances: wind direction/speed, atmospheric stability, site conditions (building, water, trees, etc.), height of leakage point, and buoyancy and momentum of leaking material. Many models exist to represent these dispersion phenomena numerically, also expressed in various empirical formulas.

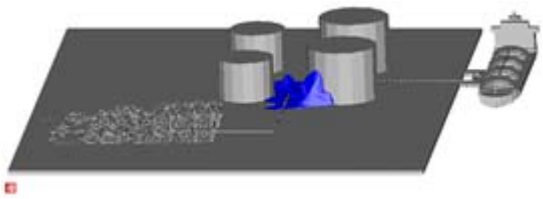

Fig. 1. An example of an LNG spill and vapor dispersion [2].

2. Theory

2.1. Consequence analysis software

Unlike in cases of detailed plant safety design, real-time accident responses require fast and repetitive calculations to predict the affected area quickly as well as to reduce the uncertainty in

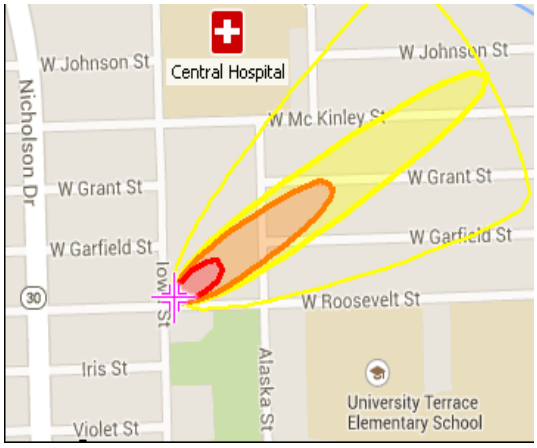

Fig. 2. An ALOHA threat zone estimate displayed on a MARPLOT map [7].

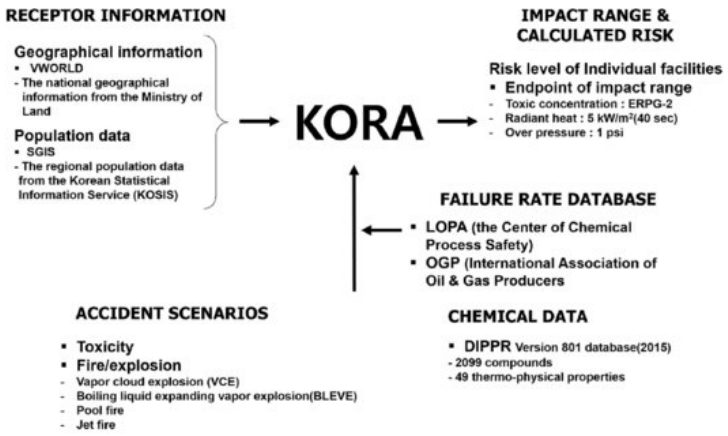

Fig. 3. KORA framework of National Institute of Chemical Safety [5].

the distribution of damage within the affected area. (In a real-time response, on-site information of the accident naturally contains guesses/estimates, and simulations should handle inherent uncertainties.) EPA ALOHA [7] and KORA of National Institute of Chemical Safety [5] are popular choices in off-line analyses (see Figs. 2 and 3). Sensor-enhanced, real-time estimation of the dispersion is another notable approach (see Fig. 4).

2.2. Cellular automata

Cellular automata (see Fig. 5) have been applied to simulations in many fields such as astrophysics, social phenomena, fire spread, and evacuation. Lattice gas cellular automata are

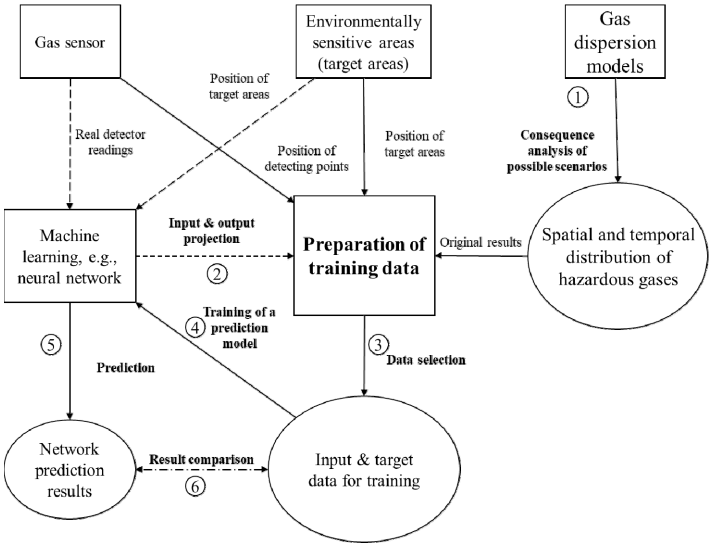

Fig. 4. Estimation of real-time gas dispersion, integrated with surrogate prediction models (adapted from [10])

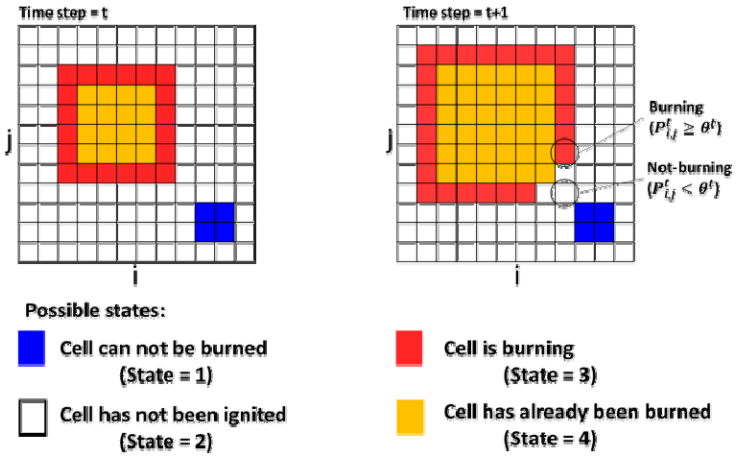

Fig. 5. Basic principle of cellular automata simulation (adapted from [13])

used to simulate fluid flows, deriving the macroscopic Navier-Stokes equations.

Lattice Boltzmann methods (LBM) is a sort of simplified molecular dynamics model in which space, time, and particle velocities are all discrete. LBM, originated from the lattice gas automata method, is actively used for fluid simulation [12].

Based on cellular automata [6,12,13], this study develops a model for off-line as well as real-time consequence analyses of the dispersion of hazardous chemicals. It implements main functionalities for a public domain software package, necessary for mandatory risk assessments of and emergency responses for probable dangerous accidents in chemical plants.

2.3. Agent-based modeling

Agent-based modeling (ABM) is a computational methodology that enables one to model complex systems [11]. It has some overlaps with cellular automata. If identification of characteristics of emerging behavior during dispersion is main part of investigation, ABM could be a good choice. NetLogo would be used to study variance of domino effects depending on the layout configuration of storage tanks of hazardous materials. Fig. 6 shows an agent-based model investigating propagation of a forest fire, simulated on the NetLogo environment.

2.4. Traditional CFD-based solution

Computational fluid dynamic models are also based on cells; control volumes are used in case of the finite volume method [9]. Cases solving turbulent unsteady flows inside a complex 3D geometry of a chemical plant are suitable for CFD simulations. However, our system is instead focused on the long-range dispersion simulation for emergency response, and CFD solutions would be integrated later when we consider short-term, inside-plant dispersions.

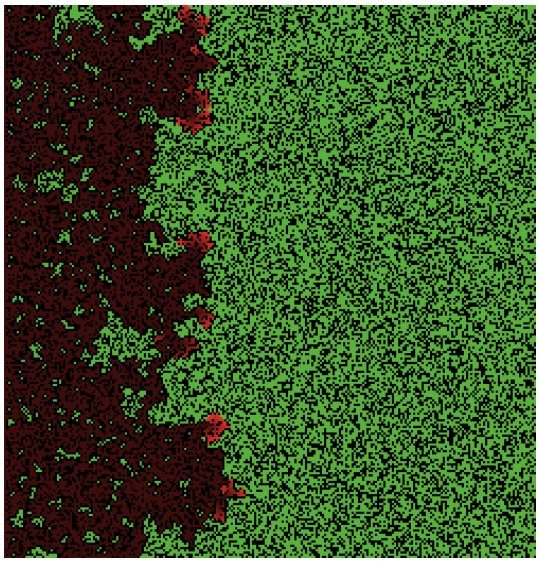

Fig. 6. An agent-based, forest fire simulation on NetLogo [11]

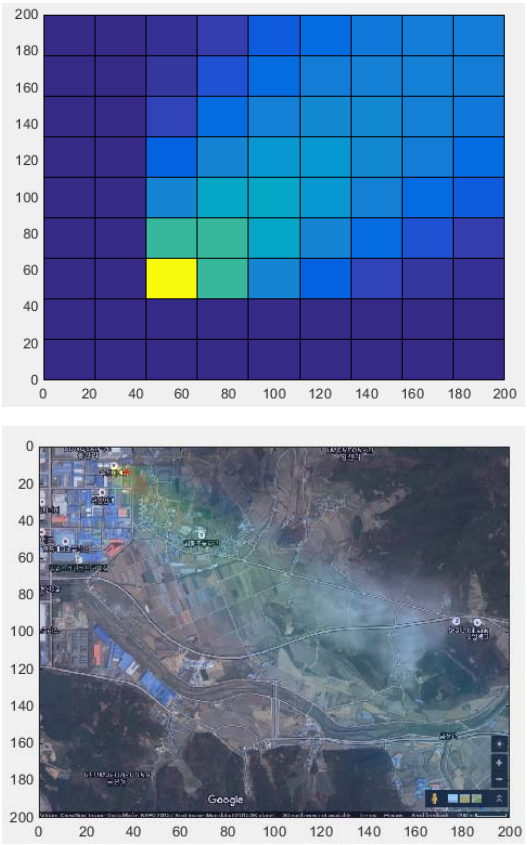

Fig. 7. Proposed dispersion simulation, based on cellular automata implemented in Python

2.5. Proposed approach

This study made it possible to supplement the model and code continuously and is different in its development of free software specialized for small and medium enterprises. Compared to the full-scale CFD, which requires large amounts of development and computation time and is hard to use in real-time responses, the loss in relative accuracy would be compromised, and the convenience of the general user is enhanced with easier interfaces and intuitive understanding of the models.

3. Results

3.1. Implementation and open source software distribution

Python is a widely used, general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation [8]. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently [3].

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the seventh most popular and the number one most wanted technology of the year. Quite a number of leading software development companies use Python [8].

Among various Python libraries (shown in Fig. 8), the following libraries were selected and used for open-source development:

- NumPy: base N-dimensional array package
- Matplotlib: data visualization
- JSON: JavaScript Object Notation encoder and decoder
- Urllib: URL handling modules
- SciPy: fundamental library for scientific computing

Python open-source libraries and meteorological information linkage made it possible to expand and update the functions continuously, and users can easily obtain the results by simply inputting the layout of the plant and the hazardous materials registered for use at the plant site. The developed system (see Figs. 7, 9 and 11) could be distributed as open source software, supporting GPU-accelerated computing for faster computation. Fig. 9 shows the flowchart of the system execution integrated with real-time meteorological information.

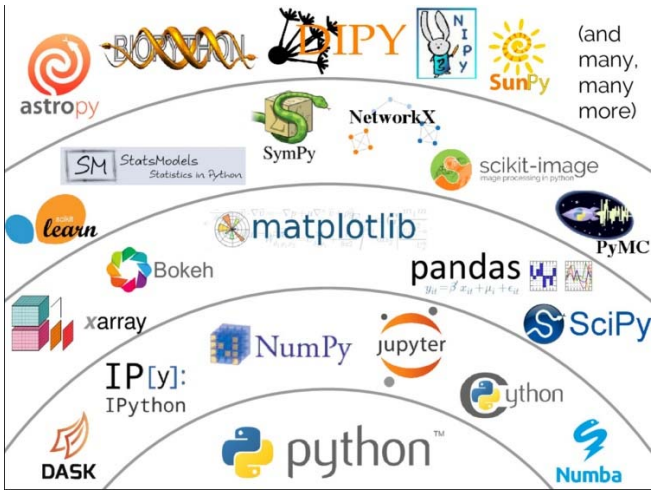

Fig. 8. Popular scientific Python libraries [1]

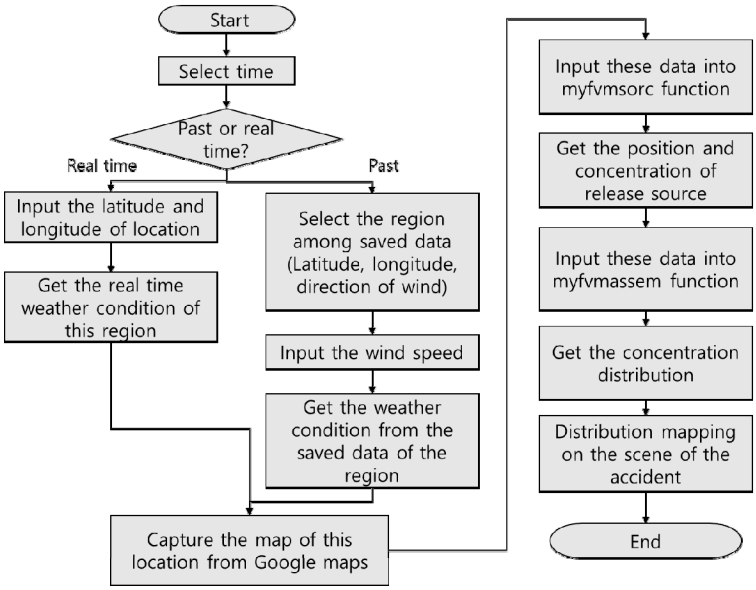

Fig. 9. Flowchart of the implemented system

Fig. 10. Comparison of a simulation result against Gumi chemical leak (2012)

3.2. Verification of the simulation results

Accuracy of the simulations was verified against full-scale CFD simulations, including a real accident case of Gumi, Korea (2012). Red, green and blue lines in Fig. 10 represent the concentration contour obtained by the extensive simulations, with the support of Korean EPA [4]. Simulation results of the proposed system, in Fig. 7, correspond well against it.

3.3. Comparison of cellular automata against computational fluid dynamics

Dispersion of gas release is a well-known problem of fluid dynamics. However, traditional CFD-based implementation and execution of the code requires huge amount of computation. Even the code is hard to maintain, requiring a group of experts during the revision process. Equations of CFD are straightforward if you are familiar with differential equations; but the mechanisms that cause these dynamics are not readily apparent from the equations.

Cellular automata or agent-based representations would employ simple algorithmic models [7]. These instructions are also explicitly stated in an easy to read and understand language. This is especially important in cases of complex dispersion simulations such as dispersion in complicated and/or dense geometry or convertsion of a part of the leaked gas during dispersion by rain or mitigative actions. Time is explicit in solving of CFD, but sometimes, arbitrary time step in cellular automata simulations needs to be converted into corresponding time in an actual physical domain.

```
import numpy as np

def myfvmsorc(sourcec, sourced, num_row, num_col, sourcep, cel_cen):

    # empty arrays initialized
    Source = np.array([])
    Source_x =np.array([])
    Source_y =np.array([])

    o = 0
    num_of_cells = num_row*num_col

    # repeat for each source cell
    for l in range(1,len(sourced)+1):
        g = 0   # no of source cells initialized
        for k in range(1,num_of_cells+1):
            # xnum, ynum: indices of the corresponding cell (by rows and columns)
            xnum = np.remainder(k,num_col)
            if np.remainder(k,num_col) == 0:
                xnum = num_col      # the last column
            ynum = np.ceil(k/num_col)

            # distance between cells
            d = np.sqrt((cel_cen[k-1,0]-sourcep[0])**2 + (cel_cen[k-1,1]-sourcep[1])**2)
            if d < sourced[l-1]/2:  # inside the effective distance
                # no of components inside the scope of the source is increased
                g = g+1

                flg = 0     # assume no repetition
                cmax = sourcec[l-1]     # concentration is compared
                # repeat for the current source data
                for m in range(1,len(Source_x)+1):
                    # when belongs to source cell, update using higher value
                    if Source_x[m-1] == xnum and Source_y[m-1] == ynum:
                        flg = m     # index of the repeated source
                        o = o+1     # no of the repeated
                        # higher value is saved after comparison against the old
                        Source[m-1] = np.maximum(cmax,Source[m-1])

                if flg == 0:    # in case of no repetition, source component is increased
                    Source_x = np.append(Source_x, xnum)
                    Source_y = np.append(Source_y, ynum)
                    Source = np.append(Source, sourcec[l-1])

    # Using the indices of row and column, obtain the corresponding serial no of the control
    Source_po = num_col*(Source_y-1) + Source_x

    return Source_po, Source
```

Fig. 11. Part of the open source OSSCA implmented in Python

4. Conclusions

Using cellular automata, this study developed a simulation model for real-time emergency responses as well as off-line consequence analyses of the dispersion of hazardous chemicals. Even though the National Institute of Chemical Safety distributes KORA as free software, implementation as open source software that is fully shared among members of academic communities and industrial experts has been in much need of timely and continued updates of the system, including new results and advances in gas dispersion simulation and computation.

Compared to the full-scale CFD, which requires large amounts of computation time and is often restricted to off-line uses, the relative accuracy loss was compromised, and the convenience of the general user was improved. Python open-source libraries and meteorological information linkage are supported, making it possible to expand and update the functions continuously. Users easily obtain the results by simply inputting the layout of the plant and suspected candidate materials of high risk. Accuracy of the implemented system was verified against

full-scale CFD simulations and real cases of accidents, and it will be available as open source software (OSSCA as the package name), with enhanced support for GPU-accelerated computing.

Acknowledgments

This research was supported by a grant [17IFIP-B087592-04] from Haptic-based Plant Operator Safety Training R&D Program funded by the Ministry of Land, Infrastructure and Transport of Korean Government.

REFERENCES

[1] Bowne-Anderson, H., *The Case for Python in Scientific Computing*, https://www.datacamp.com/community/blog/python-scientific-computing-case (2017)

[2] Gexcon, *Safety Studies: LNG Spill & Vapor Dispersion,* https://www.gexconus.com/safety-studies/lng-spill-vapor-dispersion.html

[3] Guttag, J. V., *Introduction to Computation and Programming Using Python*, 2nd Ed., MIT Press (2016)

[4] Kim, J. H. et al., Comparison Study for Impact Range of Prediction Models Through Case Study about Gumi Hydrogen Fluoride Accident. *Korean Chem. Eng. Res*., **55**(1), 48-53 (2017)

[5] National Institute of Chemical Safety, *Korea Off-site Risk Assessment Supporting Tool (KORA) User's Guide*, Ver. 3 (2018)

[6] Newman, D., *A Program to Model Gas Diffusion using Cellular Automata*, B.S. Thesis, University of Leeds (2006)

[7] NOAA, *ALOHA,* https://response.restoration.noaa.gov/oil-and-chemical-spills/chemical-spills/response-tools/aloha.html

[8] Python Software Foundation, https://www.python.org/

[9] Versteeg, H. and Malalasekera, W., *An Introduction to Computational Fluid Dynamics: The finite volume method*, 2nd Ed., Pearson (2007)

[10] Wang, B. et al., The real-time estimation of hazardous gas dispersion by the integration of gas detectors, neural network and gas dispersion models. *Journal of Hazardous Materials*, **300**, 433-442 (2015)

[11] Wilensky, U. and Rand, W., *An Introduction to Agent-Based Modeling*, MIT Press (2015)

[12] Wolf-Gladrow, D. A., *Lattice-Gas Cellular Automata and Lattice Boltzmann Models*, Springer (2000)

[13] Zheng, Z. et al., Forest fire spread simulating model using cellular automaton with extreme learning machine. *Ecological Modelling,* **348**, 33-43 (2017)