Python を用いたｾﾙｵｰﾄﾏﾄﾝによる避難時交差点ｼﾐｭﾚｰｼｮﾝ

２年３組　中村　祐太　２年３組　木下　太陽　２年３組　岩田　耀介
２年４組　酒井　　宝　２年４組　中西　琉惺
指導者　田中　善久

１　課題設定の理由

「地震大国日本」と称され、南海トラフ地震が発生する可能性が高まる中において、中規模の地震が頻発している昨今、脅威となる地震やそれに伴う津波への被害を最大限に減じる対策について、今一度深く考え直さなければならないと思い、本研究テーマを設定した。

本研究の先行研究である「シミュレーションを用いた避難経路の最適化」（中村ら、2021）を参考にして、先行研究で用いた Excel によるセルオートマトンではなく、新たにプログラミング言語 Python によるセルオートマトンの手法を習得するよう取り組み、避難時交差点における人流のシミュレーションを行うよう決めた。特に、先行研究ではうまくシミュレーションできなかった、人流の合流については大きな課題であると捉え、速やかな避難行動のシミュレーションにつながるよう目指した。

２　仮説

Python によるセルオートマトンの手法を用いることによって、交差点での人流の合流におけるシミュレーションを、Excel を用いた場合にはうまくできなったところを、円滑に実現できる。

そして、交差点での人流の合流という課題を解決することによって、実際の避難行動に近いシミュレーションを実現することができ、多くの避難者がその交差点を渡り切るのに要する時間を推定できる。将来的には、最適避難経路等を導き出すことができる。

３　研究の方法

⑴ Python を用いて一次元セルオートマトンによる避難シミュレーションのプログラム（図１）を作る。

ア　セルオートマトンとは格子状のセルと単純な規則による計算モデルのことである。

イ　本研究では、「１」を人がいる状態、「０」を誰もいない状態として捉える。

ウ　規則は右を進行方向として、「１」の１セル前方に「０」があるとき、時間が経過すると「１」が１セル進み、「１」の１セル前方に「１」があるとき、時間が経過しても、後方の「１」はその場に留まっている。

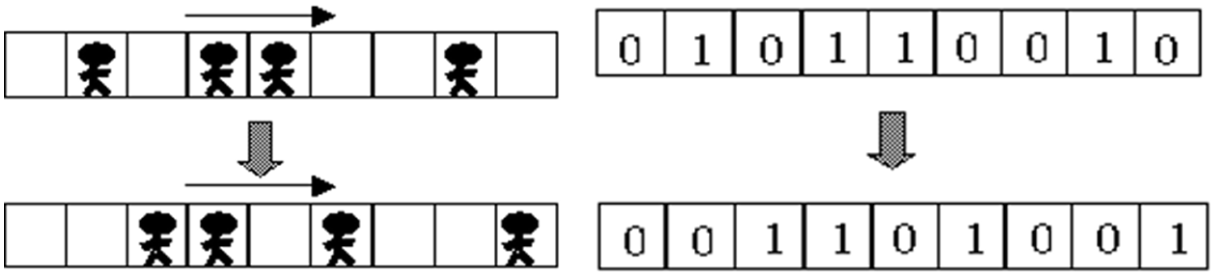


図１　セルオートマトンの一例

⑵ 二次元セルオートマトンへの応用

二次元セルオートマトンとは、一次元の（直線的な）セルオートマトンから二次元の（平面的な）セルオートマトンに置き換えたものを指す。二次元では、一次元に比べて、プログラミングにおけ

る条件の設定を細かくしなければならないが、根本的な考え方は一次元と変わらない。

図２はライフゲームの一例を示したもので、ライフゲームとは、二次元セルオートマトンを利用した有名なプログラミングである。**表１**にはライフゲームのルールをまとめる。

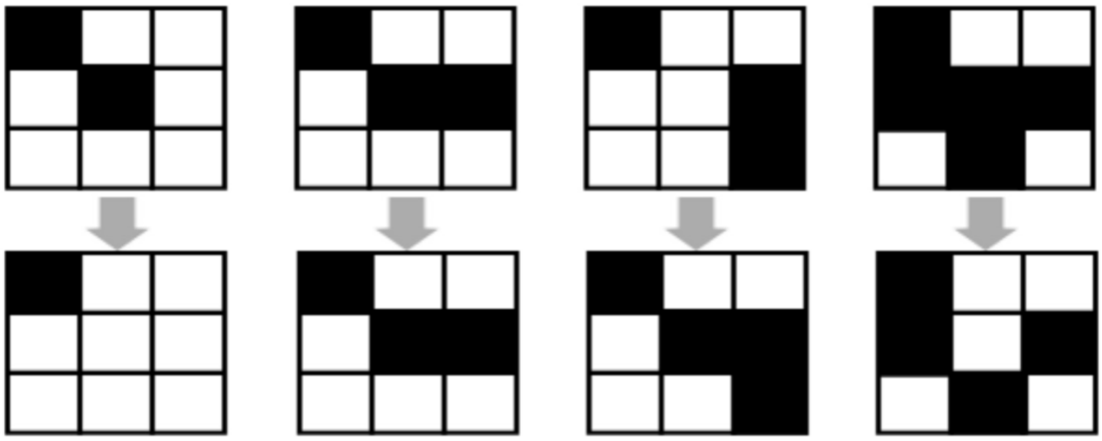


図２　ライフゲームの一例

表１　ライフゲームの３つのルール

ルール１	周りの８セルのうち３つのセルが黒ならばそのセルは黒になる
ルール２	周りの８セルのうち２つのセルが黒ならばそのセルは現状を維持する
ルール３	それ以外はそのセルは白となる

このようなルールは細菌の繁殖をモデル化するときに応用される。このルールを二次元の避難行動に則したルールに書き換えることによってそれぞれのセルが変化していく。人流の合流におけるルールに書き換えることには特に注意を払う。

４　結果と考察

(1) 一次元セルオートマトン（図３：プログラム実行、図４：プログラムコード）

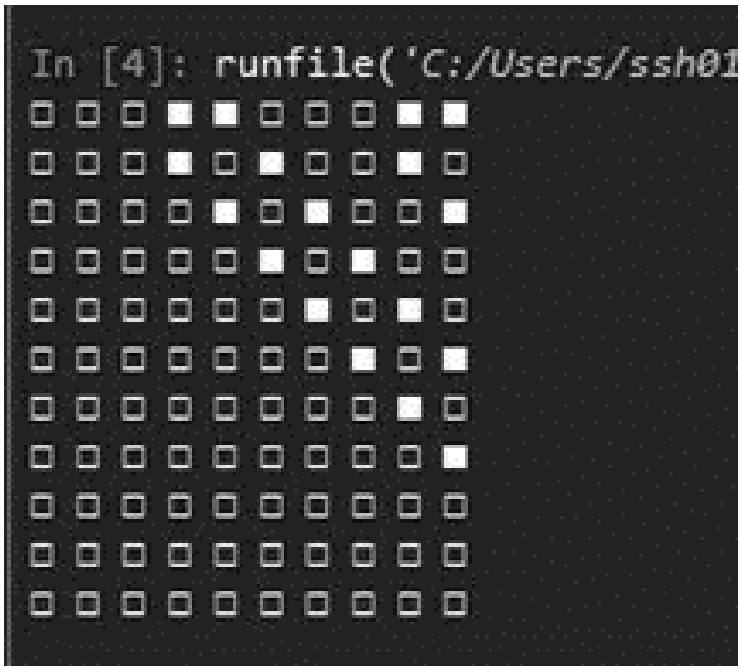


図３　プログラム実行

```
import random
import copy

MAX = 10
SIM_TIME = 10
P = 0.2

# 状態 0 は車なしの空きスペース
# 状態 1 は車両を表す
trafic = [0 for i in range(MAX)]


def list2string(l):
    string = ""
    for state in l:
        if state == 0:
            s = "□ "
        elif state == 1:
            s = "■ "
        string += s
    return string


def setRandom(l, p):
    for i, state in enumerate(l):
        if p < random.random():
            l[i] = 0
        else:
            l[i] = 1


def nextTime(l):
    temp = copy.deepcopy(l)
    for i, state in enumerate(l):
        if state == 0:
            # 左端なら
            if i == 0:
                temp[i] = 0
            # それ以外のとき後方チェック
            elif l[i-1] == 0:
                temp[i] = 0
            else:
                temp[i] = 1

        elif state == 1:
            # 右端なら
            if i == len(l)-1:
                temp[i] = 0
            # それ以外のとき前方チェック
            elif l[i+1] == 0:
                temp[i] = 0
            else:
                temp[i] = 1
    return temp

if __name__ == "__main__":
    # 実行部分
    setRandom(trafic, P)
    print(list2string(trafic))

    for i in range(SIM_TIME):
        # 情報を更新
        trafic = nextTime(trafic)
        # 情報を文字にして描画
        print(list2string(trafic))
```

図４　プログラムコード

【補足】避難者の密度のランダム設定と人流のシミュレーション

図５、**図６**に示すように、いくつかの例を用いて説明する。初期条件として、全てのセルの数 10 に対して、人の密度を 1.0 や 0.2 と設定するプログラムを作成することができた。密度 0.2 の場合、10 セル中２セルに人をランダムに配置させて、避難シミュレーションを開始させた。

図５　密度の設定と避難シミュレーション（密度 1.0 の場合）

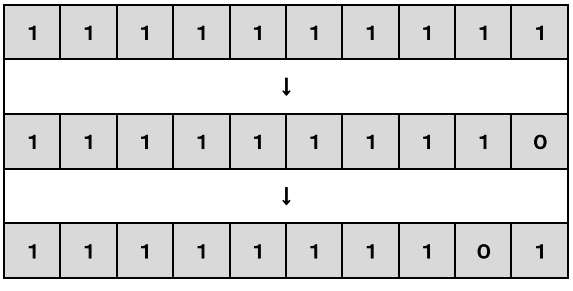
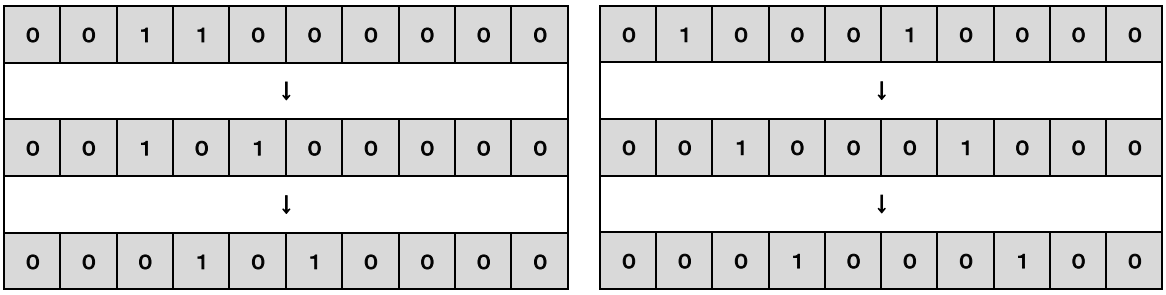


図６　密度の設定と避難シミュレーション（密度 0.2 の場合）



⑵ 二次元セルオートマトンへの応用

ライフゲームのルールにならって、二次元の避難行動に関するルールを、プログラムとして完成させることはできなかった。

⑶ 結果と考察

一次元のセルオートマトンのプログラムは簡単に作ることができたが、次の難所は、一次元から二次元セルオートマトンへの応用のプログラムを作成させることが次の難所である。その次の難所は、避難時交差点での人流の合流における二次元セルオートマトンのプログラムである。Python の動作環境を調整するための苦労や、セルオートマトンという概念をしっかり理解するための時間に多くを要し、研究が途中までしか進まなかったが、進むべき研究の方向性は定まったと考える。

５　まとめと今後の課題

・一次元のセルオートマトンまではシミュレーションを実行できたが、二次元のセルオートマトンに発展させ、また、避難時交差点における人流の合流における避難シミュレーションを完成させるまでには至らなかった。

・今後、本校正門前の交差点において、実際の避難行動に近いシミュレーションを実現させ、多くの避難者がその交差点を渡り切るのに要する時間を推定できる。将来的には、最適避難経路等を導き出すことができる。

謝辞

この研究に取り組むにあたり、愛媛大学大学院理工学研究科の土屋卓也教授をはじめ、御指導・御助言をいただいた先生方に、お礼申し上げます。ありがとうございました。

参考文献

・中村ら(2020)「シミュレーションを用いた避難経路の最適化」
令和２年度ＳＳＨ生徒課題研究論文集

・幸田ら(2018)「地震避難シミュレーションから本校の課題を考察する－１次元セルオートマトンを活用してー」社会共創コンテスト 2018　社会共創コンテスト 201 研究・探究部門出品（グランプリ）
https://www.cri.ehime-u.ac.jp/cri_k5m4gn7/wp-content/uploads/2018/06/b0fb44b76cd4a1b653ade57a84fb6c5e.pdf

・機械系エンジニアの備忘録【python】【tkinter】【part2】canvas を使って図形を動かす
https://www.stjun.com/entry/2019/10/29/201602

・森巧尚(2021)「Python１年生」株式会社翔泳社

・三谷純(2021)「Python ゼロからはじめるプログラミング」株式会社翔泳社

・ライフゲーム　https://tomari.org/main/java/lifegame.html

・待ち行列理論 その４　https://qiita.com/ogata-k/items/01657a6628e92a641257