マルチスケールシミュレーション 課題1

- トイレへのkMC法の適用
- 強化学習による迷路の脱出アルゴリズム

2012/06/08

機械工学専攻　37-126207　高本 聡

便器のヒエラルキー

トイレへのKMC法の適用

背景

- トイレは人類にとって欠かせない存在である

サブラタ遺跡(B.C.500~)のトイレ
出典: 日本トイレ協会

- 今回は特に，公衆トイレに着目する

目的

- 周囲のユーザーにより選択が変化する例は日常生活に頻繁に登場する
  - （例）電車の乗る/座る位置，映画の席，授業の席…
- ユーザーはすべての便器を均一に扱っているわけではない
  - 男子トイレは特に用がなければ離れて使う

- 各トイレの汚れ具合に差は出るだろうか?
  - モンテカルロ法によりシミュレーションを行う

方法

- 以下のモデルでkMC法を使用

- 便器がn個
- ユーザーはポアソン到着（平均間隔λ）で来る
- ユーザーは以下のルールで便器を選定
  - (1) 両隣が開いてる便器があればそれを使用
  - (2) なければ，それ以外の便器をしぶしぶ使用
  - (3) それもなければ，待ち行列に追加

- トイレの使用時間はガウス分布に従う

（起こりうるイベント：到着，使用開始，使用終了）

パラメータ決定

- シミュレーション時間は260000秒≒72時間
- トイレの数n，到着頻度λを変動させて観察する

- 着目する指標は「全体の時間の中での各便器使用率」

- 使用時間のガウス分布はどう決定するか?
  - 高本の1週間分の計測データを使用
  - 平均20秒，標準偏差3秒程度
  - 今回は両側に+3秒して26±3として扱った

  - なお，無次元化を行うためシミュレーションでは1.0±0.12となる

実装

- 言語：C++

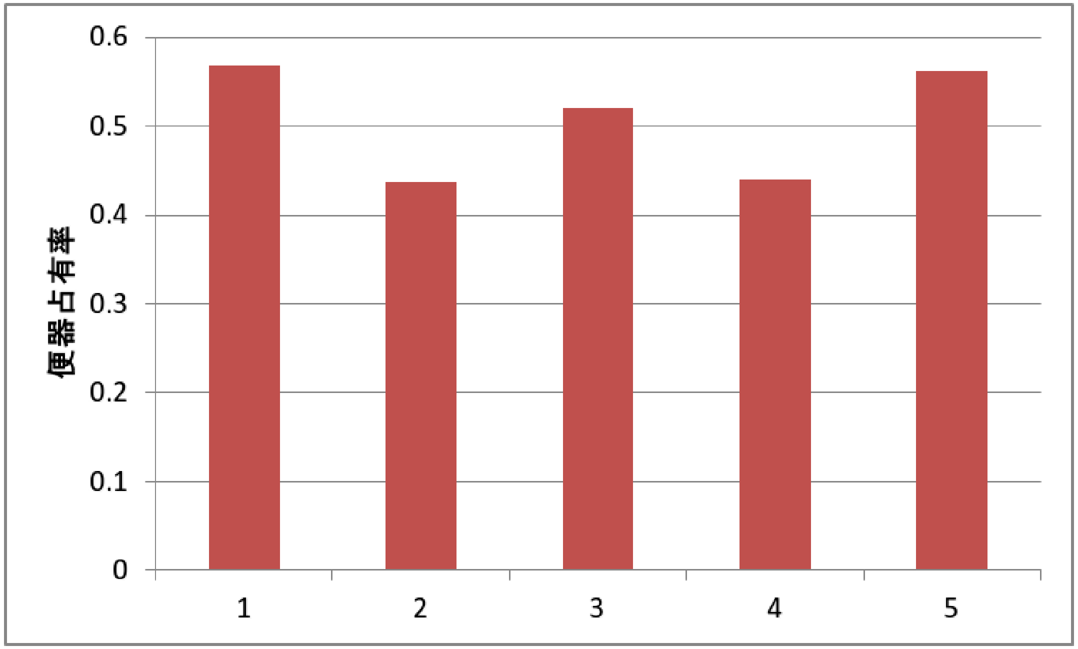
- 乱数：メルセンヌ・ツイスタ
（新規格C++11で搭載されたmt19937を使用）
ガウス分布作成にはC++11のnormal_distribution<>を，
ポアソン到着は自前の関数を用意

- kMC法：順序付きキューにイベントをpush/popする

- 特別な機能：
トイレ待ち行列が伸びすぎるとプログラム全体がクラッシュ（←我慢の限界をシミュレート）

実験結果

- （例）n=5, λ=2.5の場合：

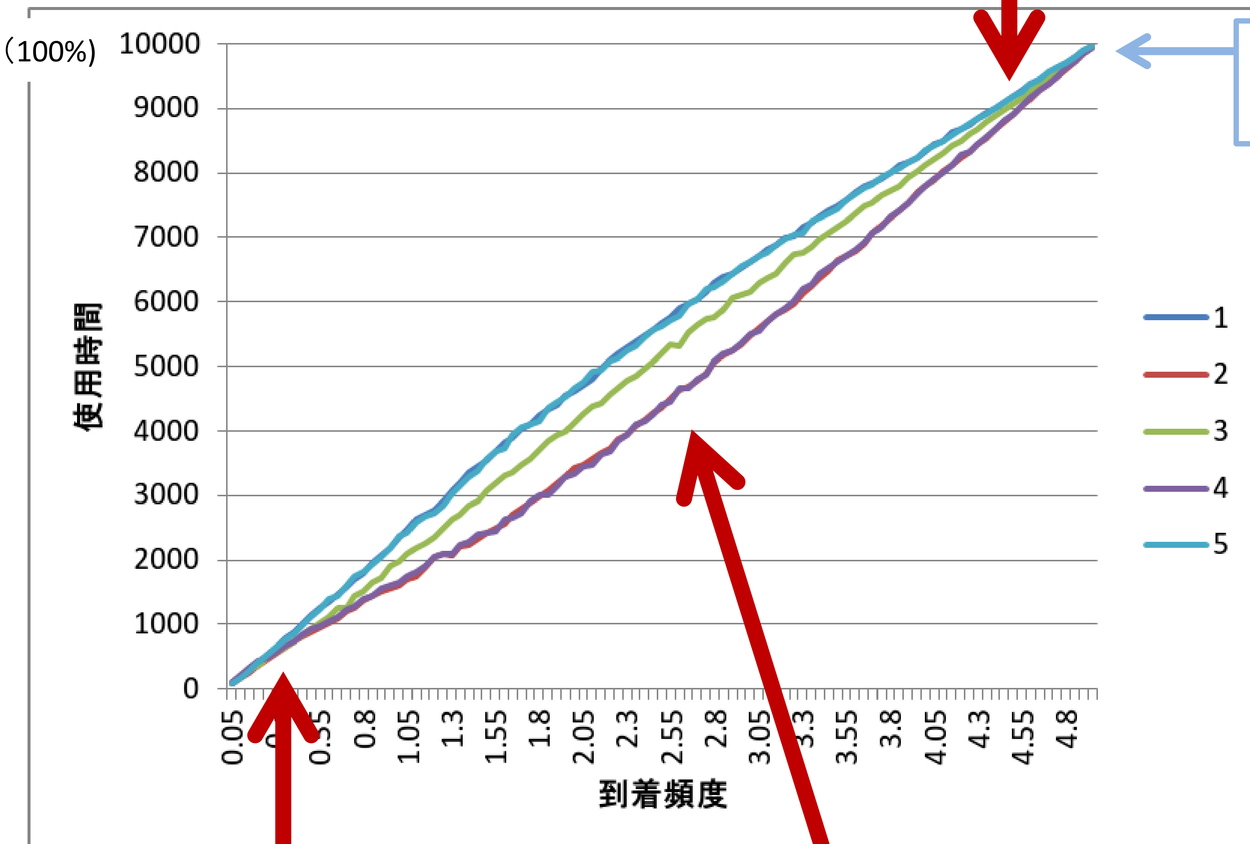


- 1, 3, 5番のトイレが使われやすい

実験結果

- n=5, λを変化させた場合：

人が詰まっている



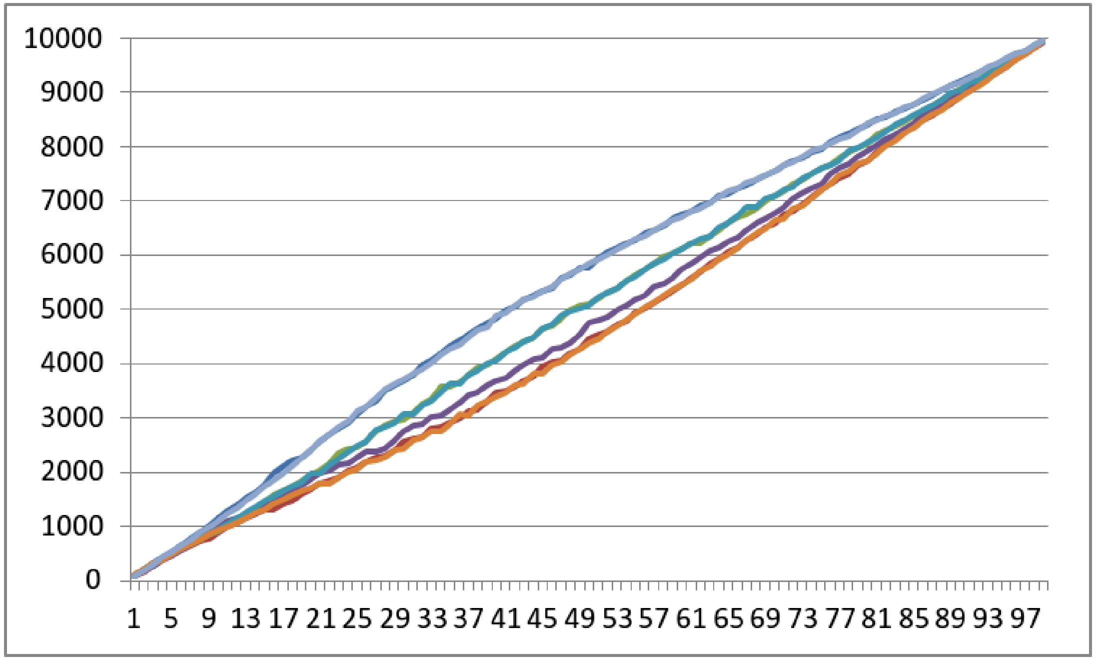
行列が際限なく増加
(=クラッシュ)

まばらで差がつかない

ほどほどに詰まると
使用率に差が開く

実験結果

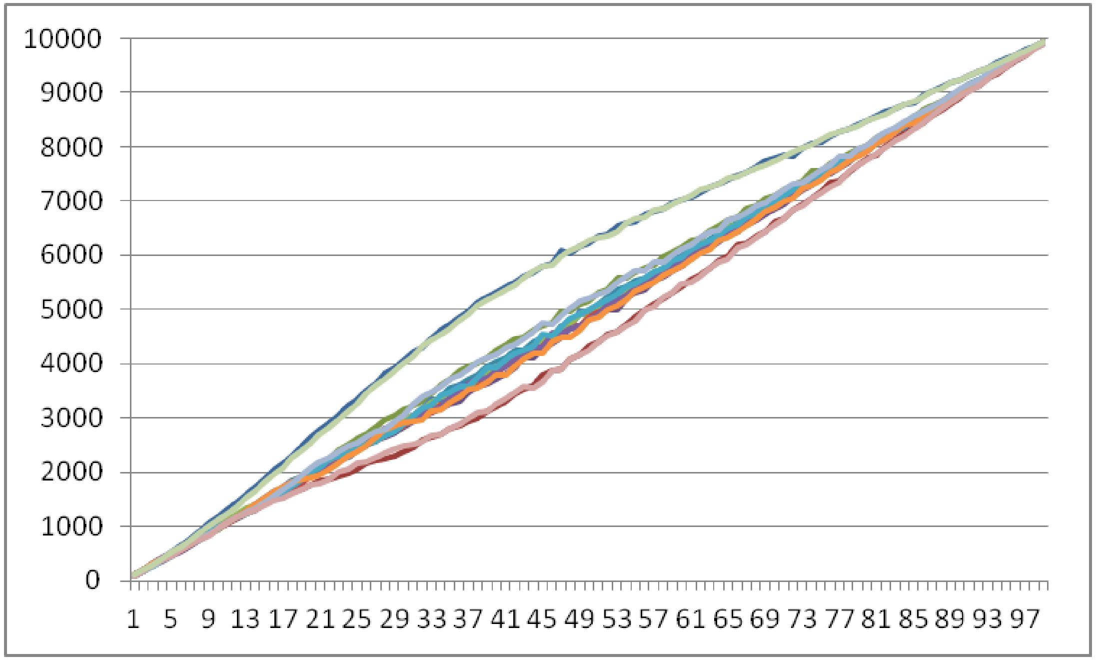
- nが増加すると



n=7

実験結果

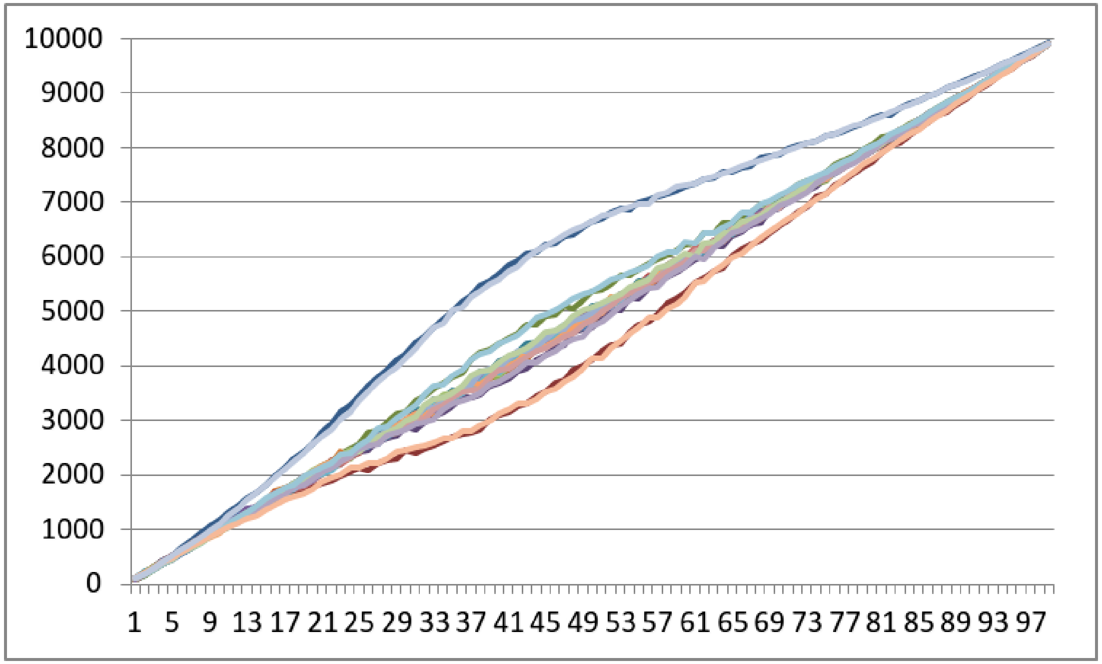
- nが増加すると



n=15

実験結果

- nが増加すると

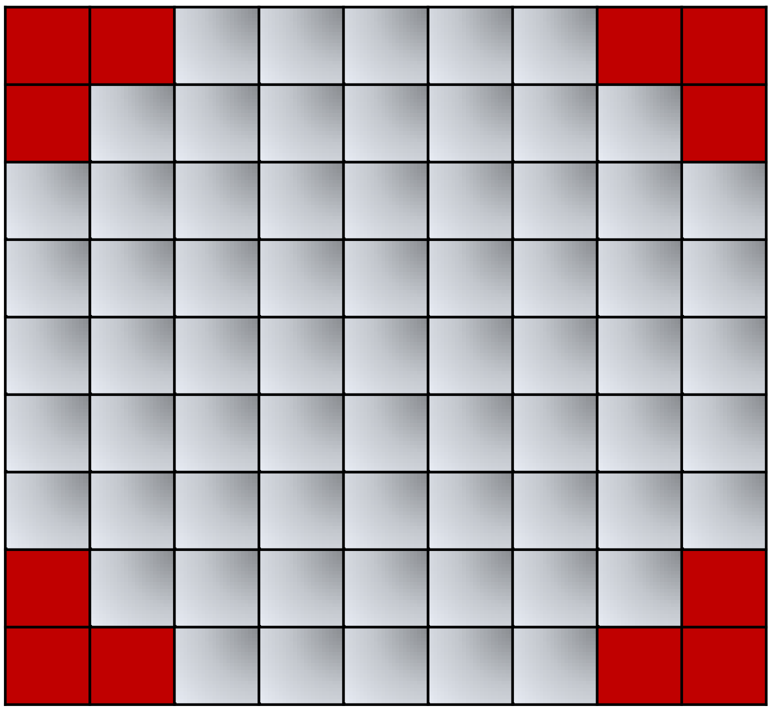


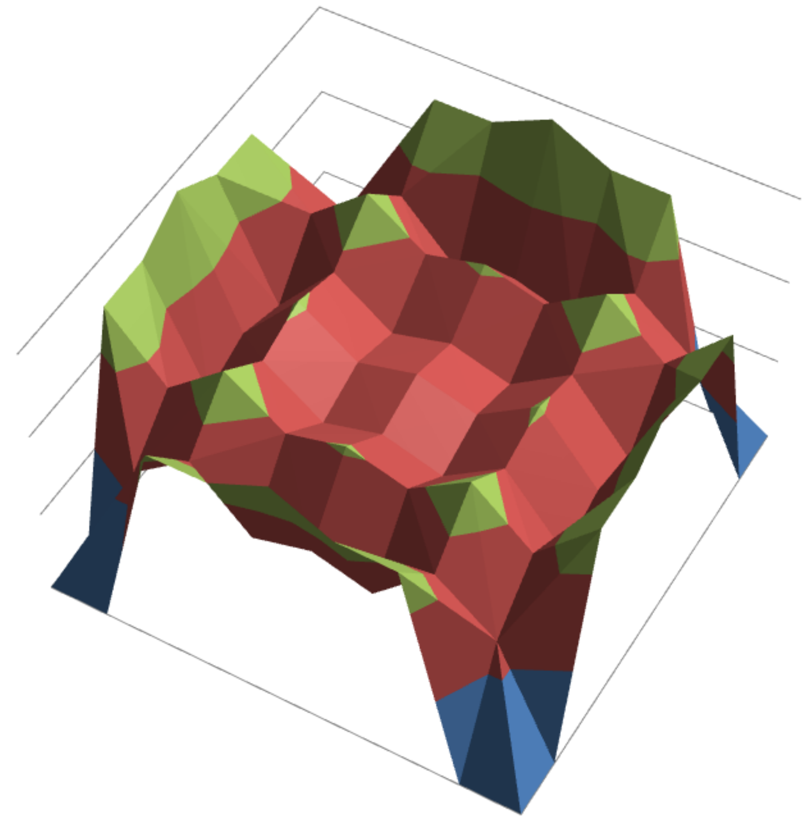
n=31

- 端の使用率は減らない (むしろ増えている?)
→境界で特別なふるまいをすることがわかる

参考：バトルシッププロジェクト

- 船の存在確率のモンテカルロ法





- 船は隣り合って置くことができない
- 境界に高い存在確率がある
  （今回のシミュレーションとの類似性）

結論と今後の課題

- 結論と考察
  ユーザーが便器の場所に関して志向を持たなくても，「離れて使う」という条件から，特に境界で使用率に偏りが起きることがわかった

  公衆トイレは端がよごれやすいと考えられる

- 今後の課題
  実験結果との定量的な比較
  到着率の時間変化
  トイレが1列でない場合
  連れション実装

モンテカルロ法と人工知能

強化学習による迷路の脱出アルゴリズム

強化学習とは

- 報酬のみが与えられる状態で，最も報酬が大きくなるような行動を獲得する学習手法
  - 関連分野
    - 教師あり学習
      正解となるやり方がいくつか与えられたとき，一般の場合に正解を探す学習手法（例：回帰，クラス分類）
    - 教師なし学習
      正解がなくデータのみが与えられたとき，そのデータの性質を調べる学習手法（例：クラスタリング，主成分分析）

- 具体例
  振り子の振り上げ/倒立制御の学習
  ブラック・ジャック方策学習
  携帯電話の基地局最適配置問題

強化学習とモンテカルロ法

- モンテカルロ法を使って強化学習を行うことができる

- 環境内で様々な行動を起こし，その報酬の平均をとることで各行動の利益を求める手法

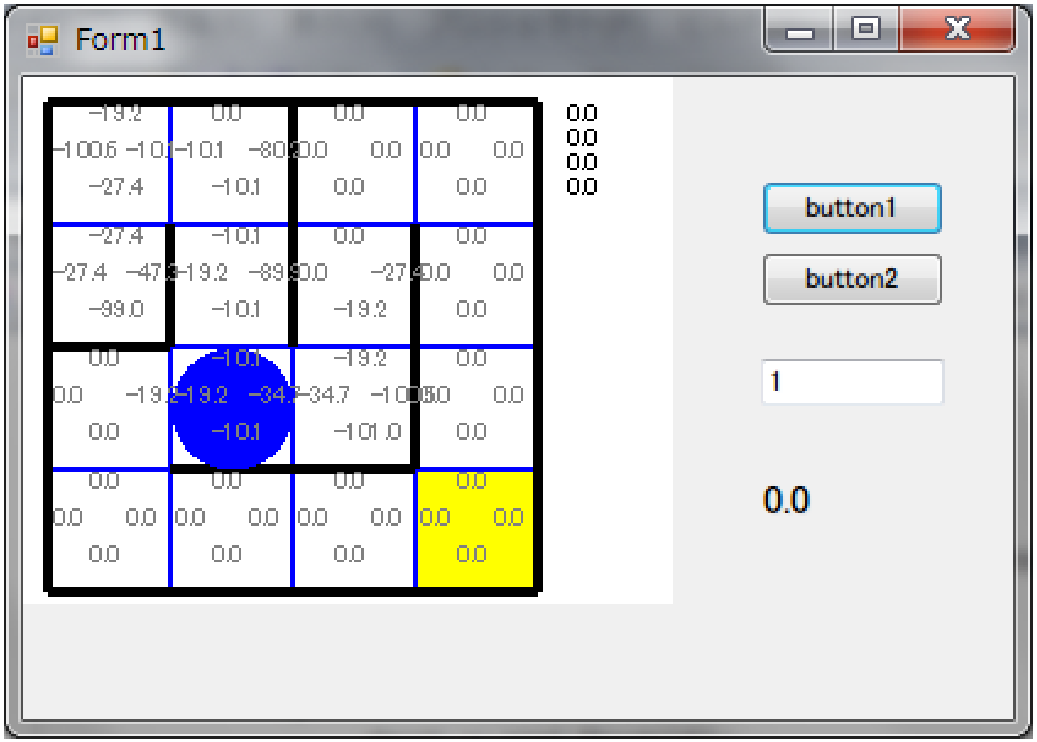
  以下のBellman方程式を解くことに相当：

$$V^{\pi}(s)=\sum_{a}\pi(s,a)\sum_{s'}P_{ss'}^{a}\left[R_{ss'}^{a}+\gamma V^{\pi}(s')\right]$$

- モンテカルロ法により各行動の報酬を推定
- 各行動の利益がわかれば最適な行動もわかる

テーマ

- 迷路の脱出アルゴリズム



青丸がエージェント，黒線が壁（ボタンはデバッグ用）

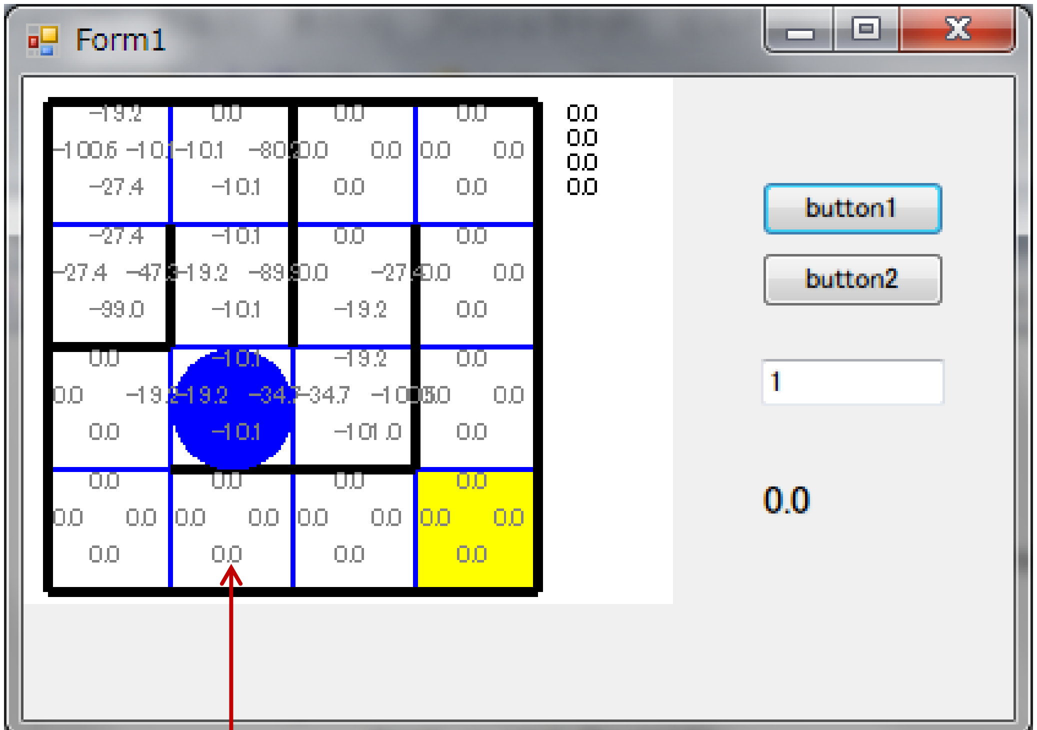
エージェントは上下左右の中から行動を選ぶ

全体のマップはエージェントに与えられないが，ゴール（右下）に着いたときのみゴールということがわかる

ゴールすると適当な場所に再配置される

テーマ

- 迷路の脱出アルゴリズム



報酬として行動毎に-1, ゴールに0の報酬を与える

エージェントは報酬を最大化する
→最短でゴールに着こうとする

この数字は行動に対する報酬の推定値
（学習開始直後なのでまだ0）

Exploration vs. Exploitation

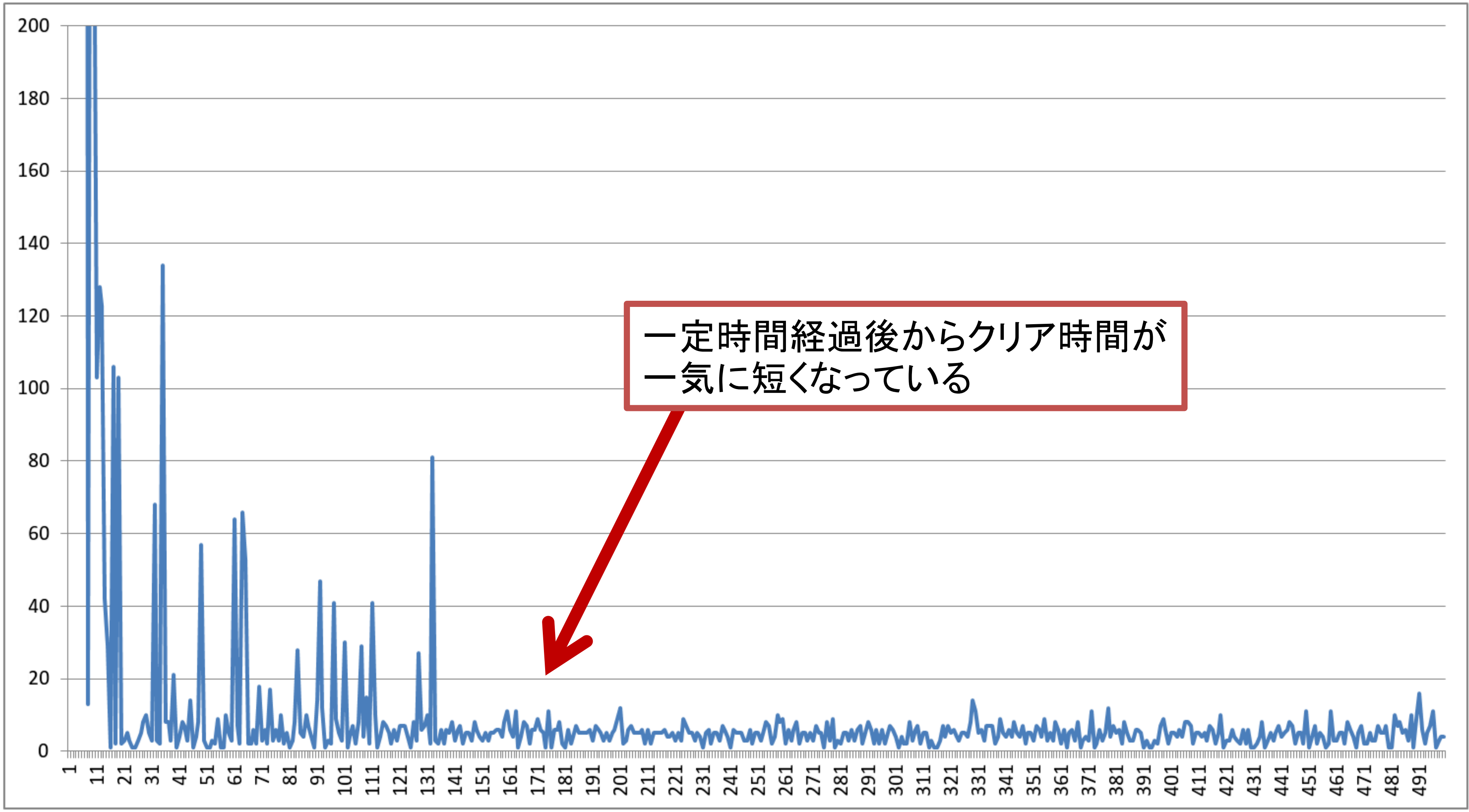
- エージェントは活発に動き，最も報酬が高くなるような行動を探索する

- ただし，報酬のみを考えると局所最適解に陥ってしまう可能性がある

- そのため一定の確率で探索を行うようにする必要がある
  – 今回はエージェントが理由もなくふらふら歩くことがある

結果

- 学習回数とクリアまでにかかったステップ数



結論

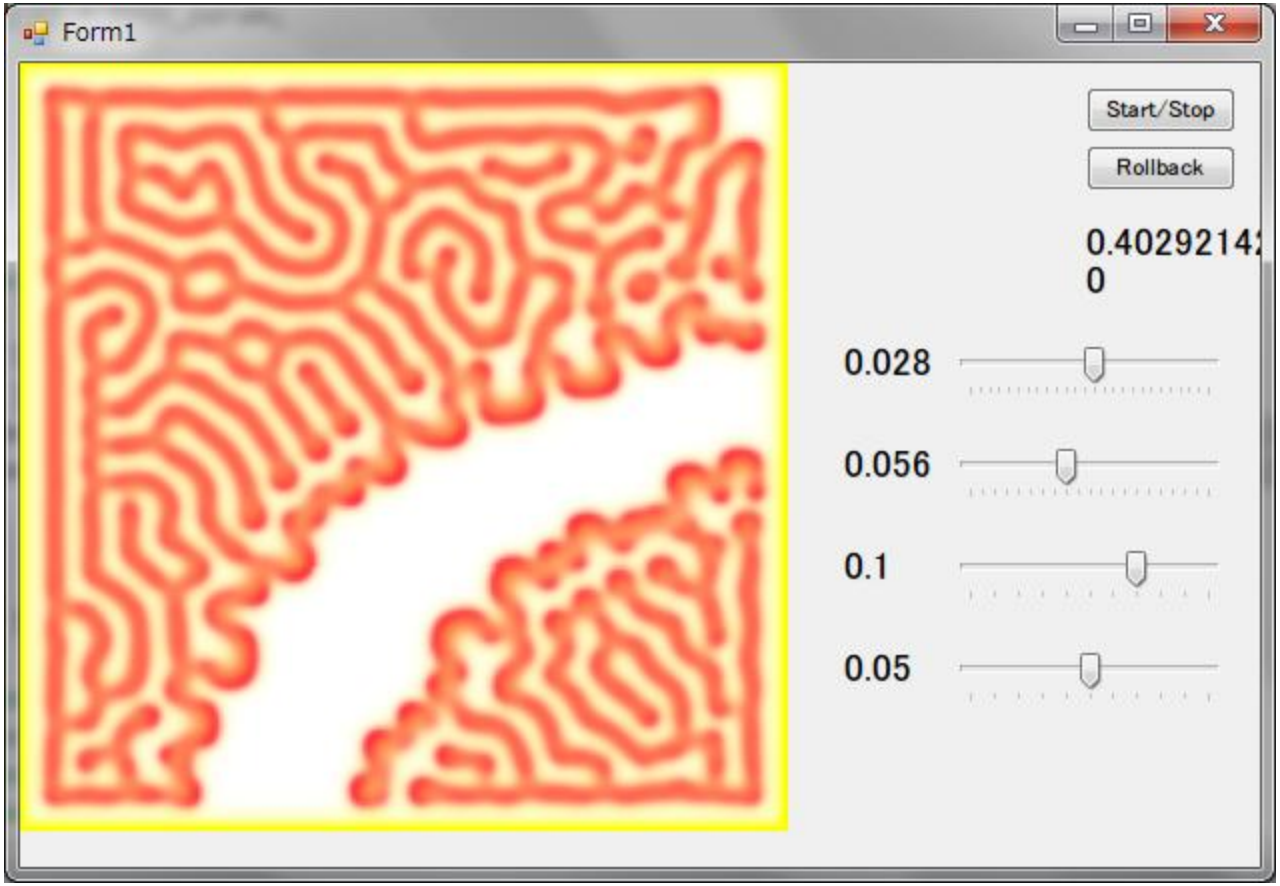
- 木構造探索などを行わなくても，経験をこなせば機械でも迷路を解くことができるようになる

- モンテカルロ法は方策の判断，決定に対しても有用な手法であることがわかった

おまけ

- 反応拡散系シミュレータ
  - 「パラメータをいじってマウスで遊べます」

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1-u),$$
$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F+k)v.$$



•実装
C++（差分方程式）
C#（描画まわり）

•計算
オイラー法