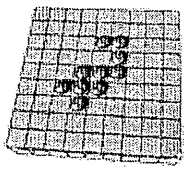
Cellular Automata

遠藤 聡志　赤嶺 有平
Satoshi Endo　Yuhei Akamine
琉球大学工学部 情報工学科　琉球大学大学院 理工学研究科

CA法の基本原理



同じ大きさの均一なセルの敷き詰められた空間を想定

各セルはk種類の状態をとる

隣接するセルの時間tにおける状態値に応じて時間t+1のセルの状態値が決定する.

近傍則：近傍のセルの状態値の和が3以上なら次の状態は1

近傍の状態値に対して次の状態値を決定する規則を近傍則と呼ぶ

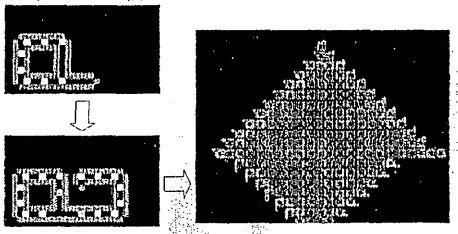
近傍則を設計することで様々な現象に対応できる

セルオートマトンの変遷

- Self-reproducing Automata
  - von Neumannの自動自己修復・複製装置
- Pattern Formation
  - Game of Life
  - S.Wolframの4つのクラス
- Multi-Agent?
  - LGA(Lattice Gas Automata)
  - 交通流，人の流れ，粒状体シミュレーションetc.

C.Langtonの自己複製ループ

- 8状態5近傍2次元CA



初期状態と同じ形状のパターンが自己複製する

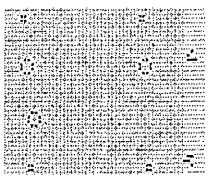
シミュレーション実行

H.Conwayのライフゲーム

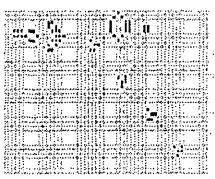
- 各セルは生と死の 2 種類の状態をとる
- 次のステップのセルの状態は 隣接する8つのセルの状態から以下のような規則によって決まる
  - 隣接する8つのセルでで生きてるセルの合計を *sum* とする
  - 2 <= *sum* <= 3 のとき、生きてるセルは生きつづける
  - *sum* >= 4 || *sum* <= 1 のとき、セルは死ぬ
  - *sum* == 3のとき、死んでるセルは生き返る
- 周囲にほどよい数がいるとき生存できて、過疎や過密のとき死滅し、最適な条件のとき生まれるという、動物やバクテリアの環境を単純化したモデル

ライフゲームシミュレーション

・2次元のダイナミクス



ブリンカー

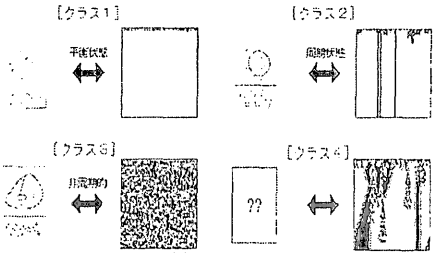


グライダー

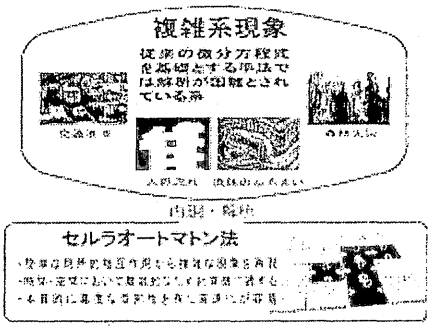
シミュレーション実行

S.Wolframの4つのクラス

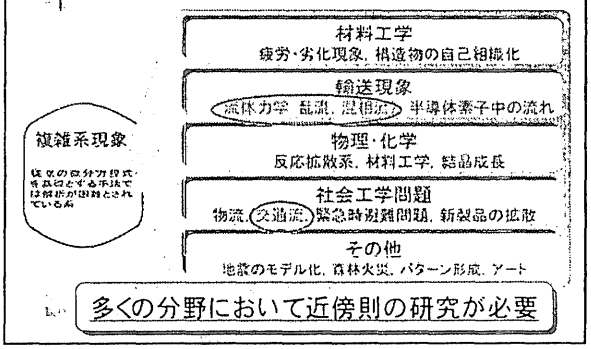
- 1次元CAのダイナミクス



複雑系におけるCA法



CA法の利用が期待される分野



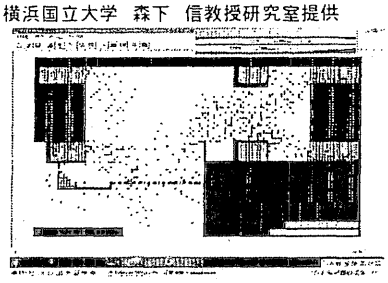
Cellular Automata 2001（横浜）

パターン形成関連	7件
計算機科学	3件
Logic・Algorithms	6件
最適設計	4件
粒状体	6件
流体解析	6件
人の流れ	4件
交通流	5件

マルチエージェント系

Multi-Agent Application

JR 田町駅構内の人の流れ
横浜国立大学 森下 信教授研究室提供



Rule 184

前方セルの状態	遷移前の状態	前方セルの状態	遷移後の状態
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$10111000_{(bin)}$
$184_{(dec)}$

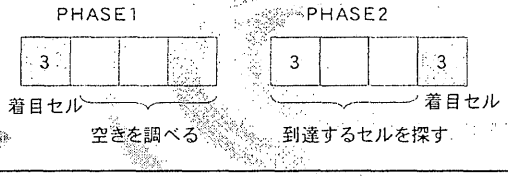
状態'1'を「エージェントが存在する」と考え，前方にエージェントが存在しないときにエージェントが前進することを表現する遷移規則

K. Nagelらの多速度交通モデル

- 近傍則
  - Vmaxに達していないときは，Vmaxまで1時間ステップで1加速
  - 前に車があれば減速
  - 停止状態でない車は確率Pで減速
- 実際の都市のシミュレーションが行われ，実用性，再現性が確認されている
- 以下，Vmax=3，P=0.1として試作

車の移動の記述

- 移動は直接的には記述できない
  - 以下の2段階で記述する
    - PHASE1:ぶつからないように速度を決定
    - PHASE2:着目セルに到達する後方のセルの値をコピー



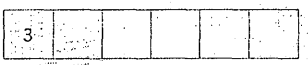
近傍の状態値の参照

- 後方3近傍，前方5近傍参照する
  - Vmax=3なので，前後3近傍必要
  - 減速する余裕を持たせるためにさらに2近傍
- 状態値変数の宣言(stateキーワード)
- 近傍のセル状態値の参照(セル参照演算子)


```
state n;
b3 = [-3,0];b2 = [-2,0];b1 = [-1,0];c  = [0,0];
f1 = [ 1,0];f2 = [ 2,0];f3 = [ 3,0];f4 = [4,0];f5 = [5,0];
```

PHASE1:速度決定

- 前方の空きを調べて加速する
  - timeキーワードを用いてPHASE分けを行う
  - timeはシミュレーション開始からのステップ数を返す

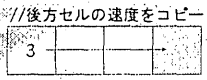



```
if(time % 2 == 0)
{
    n = 4 : when(f1==0&&f2==0&&f3==0&&f4==0&&f5==0&&(c>2))
        3 : when(f1==0&&f2==0&&f3==0&&(c>1))
        2 : when(f1==0&&(c!=0))   //速度1
        1 : when(c!=0)            //速度0
        0 : otherwise;            //空の状態
```

PHASE2:状態値の移動

- 着目セルに到達する後方のセルの状態値をコピー
- 着目セルの速度が1以上(n≧2)なら空に遷移

```
else
{
    if(c == 0 ) {
       n = 4 : when(b3 == 4) //後方セルの速度をコピー
           3 : when(b2 == 3)
           2 : when(b1 == 2)
           0 : otherwise;
    }else {
       n = 1 : when(c == 1) //停止ならそのまま停止
           0 : otherwise;   //走行中なら空に遷移
    }
}
```



減速率Pの導入

- パラメータ型変数による確率Pの調整
- randキーワードによる速度の揺らぎの導入

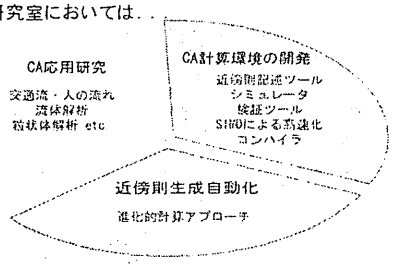
```
param P;
if(time % 2 == 0)
{
   sp=  4 : when(f1==0&&f2==0&&f3==0&&f4==0&&f5==0&&(c>2))
        3 : when(f1==0&&f2==0&&f3==0&&(c>1))
        2 : when(f1==0&&(c!=0))
        1 : when(c!=0)
        0 : otherwise;

   if(sp >1 && rand % 100 < P)
       sp = sp - 1;
   n = sp;
```

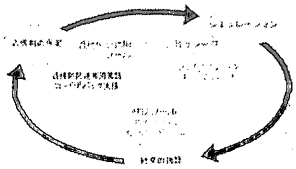
シミュレーション実行

CA研究のアプローチ

当研究室においては．．



CA用計算環境の開発



近傍則記述ツール
- 近傍則記述専用言語の設計
- コード中の理論的誤り指摘機能

シミュレータ
- 設計用インタプリタ型シミュレータ
- シミュレータの高速化手法

検証ツール
- 状態値可視化ツール

全ての機能を統合した総合計算環境

近傍則記述専用言語

ライフゲームのコーディング例

```
state s;
sum =
 [-1,-1]+[0,-1]+[ 1,-1]+
 [-1, 0]+[1, 0]+[-1, 1]+
 [ 0, 1]+[1, 1];
this = [0,0];
s =1     : when(sum == 3)
   this.s: when(sum == 2)
   0     : otherwise ;
```

- 近傍則を必要最小限のコードで記述できる
- 各セルを原点とする相対座標系を用いることで，CAの定義に違反する記述を排除する
- C言語と似た仕様とすることで，必要とする知識を最小限とする
- n次元のCA空間に対応する

近傍則記述中の誤り検出

- 近傍則の記述において意味的に間違いと推測される個所を指摘

検出可能な誤り
1. 結果が不変な条件文
2. セルの状態値に影響をあたえない変数
3. 2回以上変更される状態値

```
if( a > 0 && b < 0 ) {
   if( c > 3 && a < 0) {
      some statements;
   }
}
```

```
state s;
a = [0,1];  b = [0,-1];
if(b == 1)
   s = [1,0];
```

```
if( a > 0 )
   cell = 0;
if( a > -2)
   cell = 1;
```

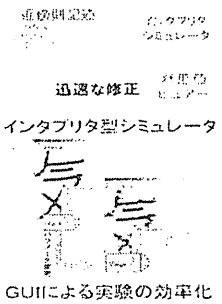
インタプリタの必要性

- 数ステップの視覚的な検証ですむモデルの存在（道路交通モデル，粒状体など）
  - インタプリタは，小規模のシミュレーションを効率的に実行できる
- 実行時エラーチェック
  - システムによるエラーチェックを強化できる
- 実行時の大幅な近傍則の変更
  - シミュレーションを止めることなく近傍則を変更できる

モデルの設計段階ではインタプリタが有利

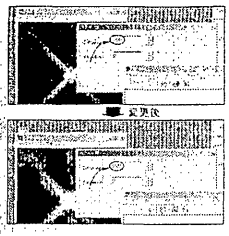
インタプリタ型CAシミュレータ

- 設計用としてインタプリタ型シミュレータを開発
- 特徴
  - レスポンスの向上
  - 実行時エラーチェック
  - GUIによる実験の効率化
    - パラメータ調整



近傍則記述デバッグ用GUI

- パラメータのGUIによる動的な変更
  - 近傍則記述中の定数変数をGUIのスライダーを用いて実行時に変更可能
  - paramキーワードで宣言
- 特定のセルの状態値の監視及び変更
  - 特定のセルの状態値をモニタ
  - シミュレーション中の状態値の強制的な変更

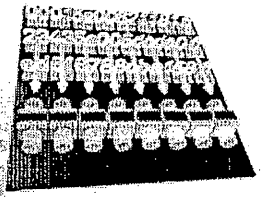


CAシミュレータの高速化

- インタプリタは実行速度が遅い
  - 大規模なシミュレーションのために高速なシミュレータが必要
- コンパイラ型シミュレータが必要(開発中)
  - MMXテクノロジを用いた高速化手法を提案
- ユーザは状況に応じて使用するシミュレータを選択ようにする
  - 設計段階ではインタプリタ型
  - 応用段階ではコンパイラ型

MMXテクノロジとCA

- MMXテクノロジとは?
  - パーソナルコンピュータ用のCPUに搭載されたSIMD型演算命令セット
- CAはSIMD型の並列演算に適している
  - CAは全てのセルを同期的にかつ同様の手続きで更新するため



MMXテクノロジのイメージ図

64ビット長のデータの並びを8つの8ビットデータの集合として並列に処理する

MMXテクノロジはCAの処理に適している

提案手法

ライフゲームをMMXテクノロジを用いて高速化

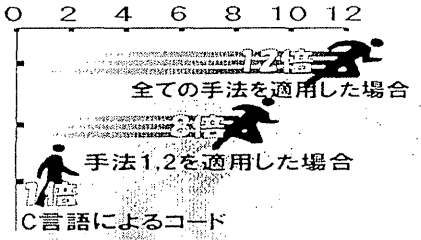
手法1) 近傍の状態値を読み出して加算する処理を並列化

手法2) 局所近傍則を適用して新しい状態を決定する処理を並列化

手法3) 上記の二つの処理をオーバーラップしてさらに並列化

これらの手法は全て同時に適用できる

提案手法による ライフゲームシミュレータの性能



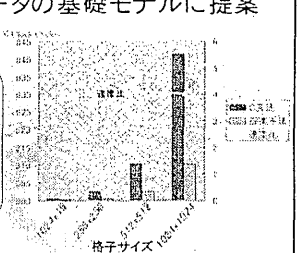
提案手法を用いたライフゲームシミュレータとC言語版の実行速度比較

提案手法では最大12倍に高速化した

複雑なモデルへの適用例

- 道路交通シミュレータの基礎モデルに提案手法を適用

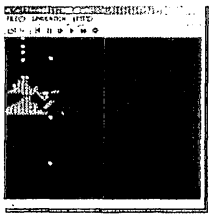
適用したモデルの近傍則
1セルが車1台分の空間を表現する
上下各1車線
速度は停車か前進のみ
進行方向のセルが空か青信号なら前進
進行方向のセルが車か赤信号なら停車

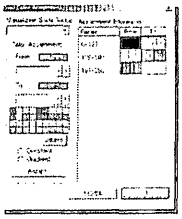


複雑なモデルにおいて4倍程度高速化

状態値の視覚化ツール

- CA法においては視覚的な検証が一般的
- GUIによる色の割り当て
  - 各セルの状態値に色を割り当てる





CAルール自動設計問題

CAルール設計を自動化

ルール設計が難しい問題 ⇒ 密度分類タスク

Gacs,P.（1978）
密度分類タスクを解くheuristicなルールを設計
（GKLルール）

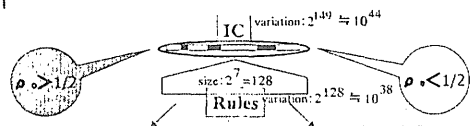
進化計算を用いたルール自動設計

EvCA

Mitchell,M.（1993）
Simple-GAを用いてルールを自動設計
⇒ 達成率がGKLより低い

Juille,H.（1998）
共進化計算を用いてルールを自動設計
⇒ タスクに関する先見知識を用いた拡張が行われた

密度分類タスク



タスクの特徴

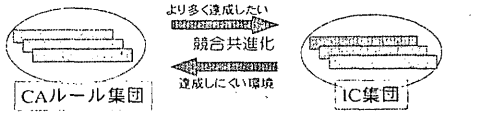
ルールに適応させる環境(IC)が膨大である.

ルールの探索範囲が膨大である.
(GAで探索可能な範囲: $2^{60} \fallingdotseq 10^{18}$)

複数のサブタスク(ρ <0.5⇒all 0s、ρ >0.5⇒all 1s)の集まりである.

共進化計算を用いた自動設計

CAルール集団とIC集団の共進化



$F(Rule_i) = \sum_j coverd(Rules_i, IC_j)$

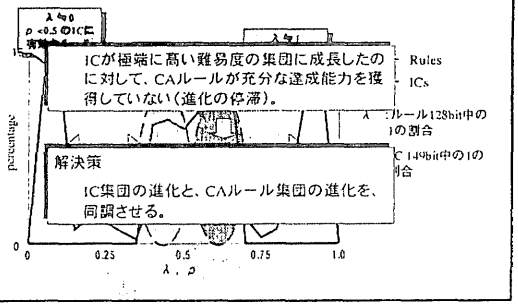
より多くタスク達成するルールが生成される

$F(IC_j) = \sum_i coverd(Rules_i, IC_j)$

より難易度の高いICが生成される

競合共進化による実験結果

収束時の各集団分布(対出力平均値・状態平均値)



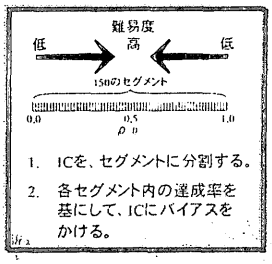
ICが極端に高い難易度の集団に成長したのに対して、CAルールが充分な達成能力を獲得していない(進化の停滞).

解決策

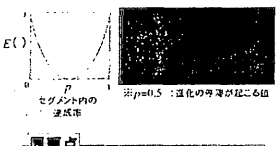
IC集団の進化と、CAルール集団の進化を、同調させる。

Juilleの提案手法

目的:
ICの難易度を、ルールの進化に同調させる



1. ICを、セグメントに分割する。
2. 各セグメント内の達成率を基にして、ICにバイアスをかける。



問題点

セグメントの定義に関して、難易度に関する先見知識を用いている。
バイアスの関数$E(\ )$がheuristicな定義によるものである。

問題解決の方策

問題点

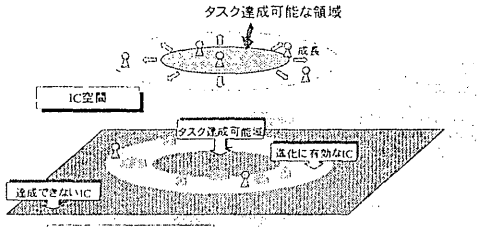
セグメントの定義に関して，難易度に関する先見知識を用いている．
バイアスの関数$E(\ )$がheuristicな定義によるものである．

あらゆる現象を対象とするCAルール設計では，
問題領域に関する知識を必要としない処理が必要

問題解決のポイント

ルール集団の精度に合わせて，学習用のICを選択
進化の停滞を起こさないICを検出・生成する
ルールのIC達成状況を利用

進化に有効なICの生成



進化に有効なICの生成

1. 達成率の低いICに有効に働くCAルールの部分集団(種)を検出
2. 種の個体を生じさせるICの評価・生成

種分類を用いた進化計算

ルールのIC達成状況を基に，進化の停滞が起こらないICを自動的に検出

種分類 Step

ルールを，達成したICを元に種を実現する
ルールのIC達成状況を符号化(表現型)
表現型を用いてルールをグループ化

CAルール表現型の符号化

種分類

進化 Step

複数の種の機能を統合したルールを生成する
競合共進化を用いる
進化の停滞を招くICを取り除き，
複数の種で学習が見込めるICに
高い評価値を与える

CAルール群

初期状態(IC_0)群

階層的クラスター分析

最短距離法

サンプルは，距離が一番近いサンプルを含んでいるクラスターに割り当てられる。

最長距離法

サンプルは，各クラスターにおける最長距離が短い方に割り当てられる。

重心法

サンプルをクラスターの中心で比較し，一番近いクラスターに割り当てる。

※ サンプルは，各Ruleの，ICに対するタスク達成状況のbit列を表している．

適応度計算式

Ruleの評価式

$$f(Rule_i) = \sum_{IC_j} Weight\_IC_j \times covered(Rule_i, IC_j)$$

$$Weight\_IC_j = \frac{1}{\sum_{Rule_i} covered(Rule_i, IC_j)}$$

ルールの評価:
最も達成率が高いルールを評価

ICの評価式

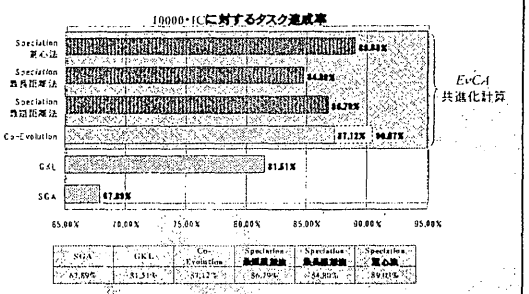
$$f(IC_j) = \sum_{Rule_i} Weight\_Rule_i' \times covered(Rule_i, IC_j)$$

$$Weight\_Rule_i' = \frac{1}{\sum_{IC_j} covered(Rule_i, IC_j)}$$

ICの評価:
所属するクラスタでの達成率
×
他のクラスタでの達成率

※いずれのクラスタでも学習が見込めるICが選択されることが期待できる

ルールの性能比較



SGA	GSL	Co-Evolution	Speciation 最短距離法	Speciation 最長距離法	Speciation 重心法
47.89%	81.61%	87.12%	86.79%	84.88%	89.01%

C4.5を適用したCAルール分析

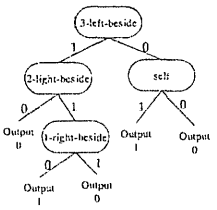
目的:要素の集合体であるCAルールを、論理的に解析する

CAルール

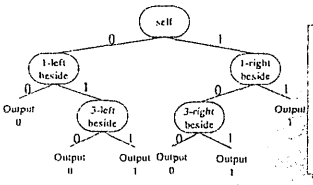
Input	Output
0000000	0
0000001	1
⋮	⋮
1111111	1

概念学習システム

決定木



GKLルールの木構造



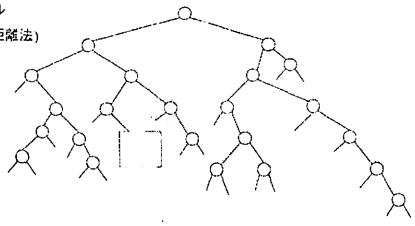
If $s_i(t) = 0$, then $s_i(t+1) =$ *majority* $[s_i(t), s_{i-1}(t), s_{i-3}(t)]$

If $s_i(t) = 1$, then $s_i(t+1) =$ *majority* $[s_i(t), s_{i+1}(t), s_{i+3}(t)]$

C4.5により取り出された木構造は，完全に元の論理式と同一ルールは，高圧縮率の決定木が作成される

自動設計ルールの木構造

自動設計ルール
(極分類・最短距離法)



・自動設計ルールは，heuristicルールに比べて決定木のサイズが大きい
自動設計ルールは，一部のルール要素がC4.5によりノイズとして取り扱われ，決定木に収納できない

各設計手法の決定木比較

C4.5で生成した決定木

	GKL	Speciation I	Speciation II	Speciation III
決定木サイズ	11	33	29	23
圧縮率(%)	8.6	25.8	22.7	17.2
エラー率	0	10.2	18.0	10.2
タスク達成率	81.51	86.79	84.80	89.03

に設計したCAルールは，論理的に理解しやすいが，その性能に限界がある．
自動設計したCAルールは，論理的に理解することが難しいが，高い達成率を記録する．

高精度のCAモデル作成には、自動設計が有利

まとめ

- 複雑系工学シミュレーションの一手法
- マルチエージェントシステムの基礎モデル?
  - 設計手順の単純性
  - 並列計算向きの性質から大規模計算への応用が可能
- 研究課題
  - 近傍則開発の計算環境
  - 近傍則の自動生成
  - 応用的事例研究