論　文

局所的人間関係における マルチエージェントシミュレーションの研究

宇津木　到・三上　達也

はじめに
Ⅰ．マルチエージェントシステムの概要
　1．人工知能の歴史
　2．マルチエージェントシステムの歴史
　3．マルチエージェント・シミュレーションの意義
Ⅱ．シミュレーション
　1．作成環境－JAVA
　2．シミュレータとその動作
　3．環境エージェントとアクターエージェント
　4．計算ユニット
Ⅲ．シミュレータでの実験と結果
　1．シミュレーション－1
　2．シミュレーション－2
　3．シミュレーション－3
おわりに

はじめに

1936年にアラン・チューリングは“*計算可能数について－決定問題への応用－*”という論文においてチューリング・マシンという中小万能計算機を提唱した。これは、ある特定のアルゴリズムに従って機械がデータ列を処理する仮想装置として示されたものであり、現在の一般的なコンピュータ源となったノイマン型計算機の考え方は、このチューリング・マシンに大きく影響を受けているものである。同じ頃、1943年にマッカロとピッツは発表した論文“*神経活動に内在する概念の論理的計算*”の中で、神経細胞のモデル化を行った。これは後にニューラルネットワークに発展していく。また、1948年のウィーナーは、著書“*人間機械論*”の中で、「サイバネティックス」の概念を示し、人工システムと生体システムは情報とフィードバックによる制御という観点で同じように考えることができると述べ、このフィードバックの概念はその後､コンピュータ・サイエンスだけでなく、多方面の研究に大きな影響を与えることとなった。

このような開発において、コンピュータとそのプログラミング技術も大きく発展してきた。人工知能研究用におけるプログラミング言語の例では、FORTRAN（1956年開発)、LISP（1962年LISP 1.5 発表)、ELIZA（1966年発表）などがあり、それらはそれぞれ人間の記述する言語を機械翻訳としてではなく、その言語を理解して人間の言語をコンピュータの知識に基づくシステムとしてモデル化することを目的とする自然言語理解、定理証明からのアプローチとしての自動プログラミングとして、人工知能の研究を促進させてきたのである。

このようなめざましい人工知能の研究の中において、なお人工知能というものは、当初その研究で目指したような人間の思考や言語といったものを模様やモデル化といった手段で実現をすることは未だ困難なことである。一方で、人間がどのように行動し、どのような問題が発生し、その問題をどのように解決するのが望ましいのかということは、主に経営学の立場から古く研究されており、現在では、上下の情報をスムーズに伝達するための望ましいモデルとして、トップダウンモデル・ボトムアップモデルなどが一般的に知られるようになり、これらモデルの組織体制におけるマネジメントの有効性や、さらに進んだ研究として組織の硬直化をどのようにすれば解消できるのか、どのように組織を活性化していくことが望ましいかといった問題に対しても盛んに研究が行われている。そこで本論文では、コンピュータを使用して、人間の行動をモデル化し、シミュレーションという手法を用いてその擬似的な空間の中で、それらを分析し、「エージェント」と呼ばれるモデル化された「彼ら」とシミュレーションの可能性を探るものである。

Ⅰ．マルチエージェントシステムの概要

1．人工知能の歴史

人工知能とは、Artificial Intelligenceの訳語であり、「AI」と略称で呼ばれることが多い。昨今では人工知能はインターネットやWWWと結びつくことにより、あたかも自然に振る舞うようにプログラミングされた人工知能“的”なマスコットと文字で会話をすることができるという試みがあったり、P2P技術を応用して、スクリーンセーバ上で泳ぐ魚に人工知能を持たせることによりその魚が突然姿を消したり、ネットワークで結びついている他人のコンピュータの画面に出現させる事ができるというソフトDALiWorld も実用化されている。その他にもお米がおいしく炊ける炊飯器などの家電やゲームなど、ここまで生活に浸透してきた人工知能であるが、それでは人工知能とはいったい何なのか。

1956年のアメリカで行われたダートマス会議（The Dartmouth Summer Research Project on Artificial Intelligence：人工知能に関するダートマスの夏期研究会）において、「知能に関する仮説をプログラムとして記述し、それをコンピュータによって実行し、その結果を評価することによって仮説の良し悪しの検証を行うもの」として人工知能は想定されていた。これはつまり、コンピュータ上で思考や判断などの人間の知能作業をシミュレーションという形で行うことであり、これが人工知能であったといえる

矢田（1986）によれば、人工知能とは「人間の知的機能を代行するコンピュータ・システム」と定義されている。つまり、人間の持っている知能である見る・聞く・考えるといった行為をコンピュータで実現できるのであれば、 人間の暮らしにとって非常に有益に働くのではないかということであり、最終的に社会の多くの人々において人工知能が目指すところであると彼は感じていたが、しかし、一方で人工知能という語に関して一つの明確なコンセンサスが得られていないのが現状である。人工知能の一つの側面としては、コンピュータを用いて、人間の知能に関する知見を得ることを目的としている。つまり、人間の知的活動というものはどのように行われているのか、そして、記憶・創造などの人間の活動は、どのようなプロセスを経て実行に移され、経験となるのか、仮に失敗したときにはその失敗を教訓として次の機会にはどのように失敗を活かし、失敗しないようにするのかという試行錯誤の問題などに焦点が置かれ、実際に脳波やPETなどを利用して実際の人間から得られたデータを基にして、あくまでも人間が日々行っている知的活動がどのように行われているのかということが研究の対象である。一方で、現在のコンピュータをより知的にすることを目的とし、これに人間の知的活動を応用するというアプローチも存在し、ここでは上記の人間の知的活動を参考にしながらもコンピュータに応用することが可能であることが必要であり、必ずしも実際に人間の知的活動のプロセスを正確に真似することはできないとしても、そのようなプロセスや人間の思考活動の見え方に近い擬似的なアルゴリズムを作り出し、現在のコンピュータに応用できればそれでいいという立場をとっている。そのアプローチの方法により、人工知能という言葉の認識は多岐にわたり、それぞれに微妙にぶれが生じているのである。

2．マルチエージェントシステムの歴史

マルチエージェントシステムとは、エージェントと呼ばれる処理や判断を行うための小さな集団を数多く配置し、総合的に高度な出力を行わせるコンピュータシステムのことである。エージェントをその集まりとしてシステムを捉えるという考え方はコンピュータが開発された時代にまで遡り、1959年に心理学者のSelfridgeによる「パンデモニアム（pandemoneam）」というモデルで説明したことに始まるとされる。これは、特徴分析を用いてパターンを認識する脳の働きを説明したもので、このモデルでは脳の多数のdemonと呼ばれる一種のエージェントが協調作業によってパターン認識を行っていることを応用している。この後にも1970年代後半のHEARSAY-IIのような音声理解システムがパンデモニウム・モデルに基づいて構築された。このシステムでは「黒板（blackboard）」という共有記憶空間を持ち、これを介して特定の知識を担当するエージェントが相互作用することにより、音声によって入力された文章を実時間で処理し、それに対応することを可能とした。一方で、このシステムではどのようにエージェントを分割すべきなのか、どのように知識というものを割り当てるべきかなど多くの曖昧な問題が存在したためにこのHEARSAYプロジェクトは1980年代に入り終息することとなった。

人工知能の創設者の一人であり、ダートマス会議にも

参加していたMarvin Minskyは、彼の著書“*The Society of Mind*”の中で、心の働きをエージェント活動の結果として説明しようとしている。彼は、この著書において「エージェントとは何なのか。」という問題を取り上げ、一人一人ではできないことも、どうしてマルチとなると可能になるのか、エージェントたちに統一性やパーソナリティを与えるものは一体何なのかという問いかけをし、マルチエージェントシステムを構築する上で非常に重要な問題を提示することとなった。

一方で、エージェントを基本とするコンピュータ・モデリングを最初に社会科学に対して適用しようとしたのはトーマス・シェーリングである。1969年からの彼の一連の論文である*Models of Segregation*の中で彼は、エージェントベースのモデリングや社会の複雑性を述べており、近隣構造を単純に空間分布させたモデルも提案した。このモデルにおけるエージェントは、少なくとも何割かの近隣エージェントが自分と同じ“色”になると満足するといった具合に区別され、実際にエージェントが実際に色を認識できなくとも極めて分居した近隣関係がもたらされることも発見した。

また、Brooksが1986年の“*A Robust Layered Control System for a Mobile Robot*”と1987年の“Inteigence without representatioの中でSAという概念を提唱した。SAとは、Subsumption Architectureの略称であり、これは、並列プログラムを持った機械の一種である。つまり、検知と行動というごく単純な組み合わせで成り立つエージェントを複数配置し、このエージェント群を並列的に動作させるというものである。この中でBrooksは、従来の人工知能もしくは知能ロボットの設計手法における欠点を指摘し、自然界における進化のアナロジーを応用し、その手法を実現するためのアーキテクチャとしてSubsumption Architecture（SA）を提唱したとされる。この論文で、与えられた環境モデルが滅茶苦茶で意味をなさないものであったり、最終的な出力がうまくいかなかったりなど、モジュールごとにしっかりとした入出力が行われたからといっても、統合してうまく動くとは限らないことが指摘され、また、一つの制御プログラムに複雑な状況判断をさせようとすると、状況事に多様な選択を迫られて、いわゆる組み合わせ爆発が発生し実際に行動させるためには莫大な処理を行わなければならなくなることを指摘した。一方で、SAの最大の利点は、全てのエージェントが同時に実行されていて、実際の動作には優先順位にもとづいて行われるために、状況を見定めて全体を制御するための複雑なプログラムを用意する必要が無いことも利点として挙げられている。従ってSAでは単純なエージェントを複数用意し、優先順位を指定さえすれば勝手に行動させることができるわけであると述べている。つまり、従来のAIやロボット研究における複雑な問題の解決では、その問題をいくつかの適切な層（レイヤー）に縦に分割して解決するのか良いとされてきたのであり、例えばそれを「認識」、「行動の計画」、「動作」などのレイヤーに分割して解決しようとすると、各レイヤーは正常に動作していても全体としてはしっかり動作する保証はないというのである。彼は、問題を縦に分割するのではなく、横に分割して解決する方法を試みた。彼は、非常に単純な、しかし全てのモジュールが揃ったロボットをまず作り、そしてそれを変更せずに新しいモジュールを追加できるようにすることを求めている。つまり、認識から動作までを行う完全なロボット（レイヤー１）を最初から作成し、その後でレイヤー２、レイヤー３…と付加していけばレイヤー１の段階ではこのシステムは全く知的でなくとも、後には堅牢さを保持しつつ複雑な振る舞いを行うことができるロボットが作れるであろうと述べている。実際にBrooksらが開発しているロボットは、実世界のオフィス空間で行動し、障害物をよけつつ、自分で目標地点を探索して到達しようとする、あたかも知的な活動を行っている。

BrooksのSAは、進化の過程において、あるモジュールが一から完全に再設計されることはありえず、古い設計モジュールは何らかの形で残り、そこに新しいモジュールが追加されるという進化の過程で起こっていることをロボットに応用したものであるが、この概念をマルチエージェントにおいてそれぞれのエージェントに内蔵すれば、エージェントそれぞれが知的に振る舞うことが予想され、さらにエージェントの集合であるシステムも、それが社会全体のように複雑に振る舞うのではないかと予想される。

3．マルチエージェント・シミュレーションの意義

組織の研究は、例えばその組織のマネジメントをどのようにするとより効率的に成長できるのか、より多くの利益が得られるのかなどを研究することを目的として発展してきたが、実際の人間がその研究の対象である以上、

人間をそのまま実験対象にすることは難しい。一方で、人間の内部構造としては心理学によって研究されているが、人間の判断や行動というものは、理論やモデルとして表せるものではなく、また自分をとりまく環境などのさまざまな状況により逐一変化するものである。これらの人間やその環境を含めた社会全体をマルチエージェントシステムとして捉え、その中で一つのエージェントを一人の人間として表現することにより、エージェント間のつながりや環境変化・エージェント同士の相互作用によって、エージェントがその社会でどのような変化を起こすかということが掴め、社会科学における様々な問題に応用可能ではないかと考えられる。本研究ではその前段階として、人間をエージェントとして当てはめ、社会の最小構成数であるエージェント数3としてモデル化することのより、コンピュータ上で閉鎖された社会として再現するための手法を模索することを目的とするものである。

Ⅱ．シミュレーション

1．作成環境－JAVA

次に、自作したJAVAによるシミュレータプログラムについての概要を記述する。シミュレータのプログラムを記述するにあたり、プログラミング言語にJAVAを選択したが、その理由は以下の3点である。

- JAVAがオブジェクト指向言語であり、コードの書き換えが比較的簡単であること。また、将来的にエージェントを増加させる際にそれまでと同様のエージェントをすぐに作成できること。つまり、それまでのエージェントの性質を引き継ぎながら、さらに新たな性質を付加したエージェントを作成することが用意である。
- JAVAは、プラットフォームやOSに依存せずに作動させることができ、プログラムコードを一つ作成すれば、そのコードでいずれの環境でも同様の結果を導くことができる。
- JAVAにはライブラリ（基本パッケージ）としてさまざまなパーツが予め用意されている。例えば乱数発生・画面表示／描画・データのファイルへの入出力などがそれぞれパッケージとして存在しており、これらを使用することで、効率的な開発が行えることが期待される。

2．シミュレータとその動作

このプログラムでは、3つのアクターエージェントが、環境エージェントと自分以外のアクターエージェントにより影響され、それぞれのアクターエージェントがx-y平面上を移動するということを行っている。このプログラムは、【図2】のようなデータの流れで設計されている。（なお、以下、アクターエージェントを単にエージェント、環境エージェントを環境と呼ぶ。）

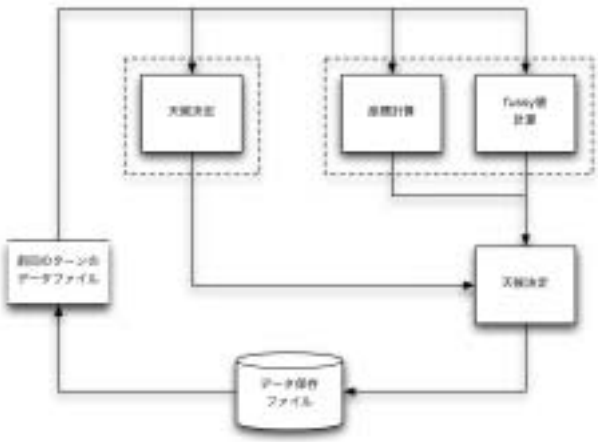

図1　本プログラムのDFD
（データフロー・ダイヤグラム）

先ずエージェントと環境を定義・初期化した後で、データを読み込む。データを読み込んだ時に、それぞれのエージェントにおける変数として現在の座標と他のエージェントに対する好き嫌いの値（fussy値と呼ぶ）を与える。次に環境が現在の天気を決定し、この値とエージェントに与えられた変数を基にして、それぞれのエージェントが次に動くべき座標の計算を行う。3つのエージェントがそれぞれ移動したことにより、各エージェントのfussy値も再計算が行われる。最後に、各エージェントの移動後の座標など、再計算された各変数の値をファイルに出力しターンは終了する。これを1ターンとする。これを繰り返すことにより、各アクターエージェントは、次々に座標を移動するように行動を起こすことになる。

3．環境エージェントとアクターエージェント

エージェントには、環境エージェントと、x-y座標上を動くアクターエージェントが存在する。それぞれのエージェントは、周囲のエージェントとの関係と、環境が生成する天候によりその行動を支配され、天候と自分以外のエージェントとの関係性によって現在の位置から移

動を行う。

環境は、エージェントの行動に制限を設ける目的で設定されるものであり、各ターンで 4 つの状況（「はれ」／「くもり」／「あめ」／「ゆき」）のいずれかの状況をとる。この時、エージェントには、wという別の変数が渡され、エージェントの行動を制限するものとなっている。また、天候はターンごとに変化する。

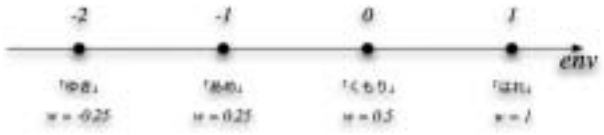

図 2　環境エージェントの変数と env と w

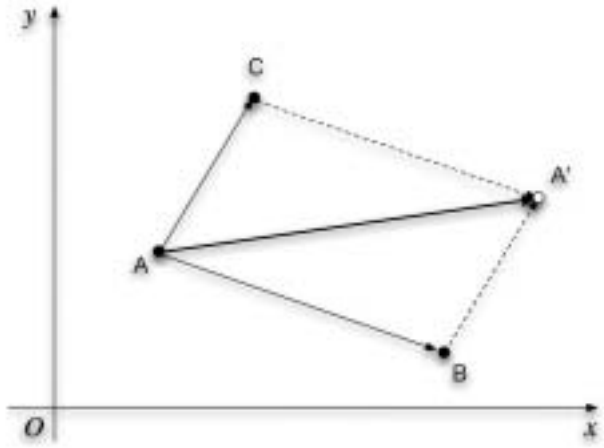

図 3　エージェント A の行動ベクトル

エージェント A は、自分と他のアクターエージェントの距離、また、自分が他のアクターエージェントを好きか嫌いかという入力情報を検知して出力計算し、次の座標に移動を行う。
その計算には $\overrightarrow{AA'} = w \cdot \{ f_{A \to B} \cdot \overrightarrow{AB} + f_{A \to C} \cdot \overrightarrow{AC} \}$ の式を利用する。（以下、「A」の「B」に対する fussy 値を $f_{A \to B}$ と記す）

アクターエージェントにはそれぞれ、自分の現在の座標（x, y）と、他のアクターエージェントに対する好き嫌いの値（fussy 値）が変数として用意されている。そして、ターンごとに自分と自分以外の二つのエージェントとのベクトル和・自分以外の二つのエージェントに対する fussy 値から自分が移動する座標が決定する。fussy 値は、- 1 ≤（fussy 値）≤ 1 の範囲で変化をする。fussy 値が- 1 に近づくほど、そのエージェントはターゲットのエージェントを嫌いということであり、ターゲットのエージェントからより遠くの距離をとるように移動し、逆に fussy 値が 1 に近づくほどターゲットのエージェントにより近づく方向に移動をすることになる。つまり、エージェント A（以下、エージェント A を単に「A」と呼ぶことにする）が行動する時に「A」における「B」と「C」の fussy 値が 1 に近い時、つまり、「A」が「B」と「C」をどちらも好きであるという状況においては、「A」は図 4 の方向に移動を行うことになり、同様にして、「B」と「C」の fussy 値に応じて「A」はそれぞれ矢印の方向に行動を起こすことになる。

同様にして、「A」においてと同様に「B」、「C」にもそれぞれ fussy 値が 2 つ設定され、その数値に応じて自分が動く方向ベクトルが決定することになる。

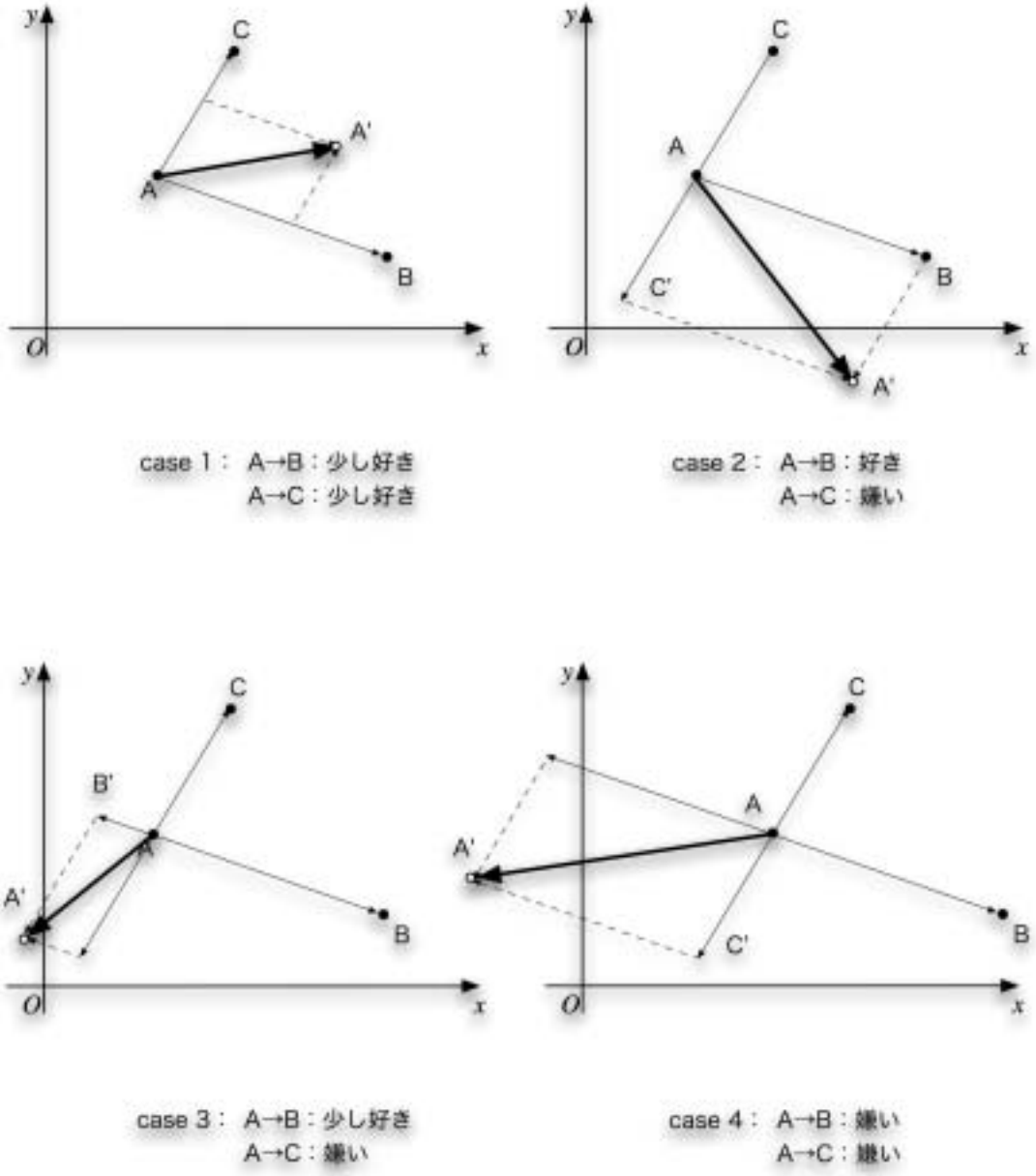

図4　「B」、「C」に対する様々なfussy値における「A」の移動ベクトル
fussy値は、一つのエージェントに対して2 つの変数が設定され、それぞれが-1から1の範囲でそれぞれ変動する。例えば「A」が「B」を好きということは、$f_{A\to B}$ が1に近いということであり、同様に「A」が「B」を嫌いということは、$f_{A\to B}$ が-1に近づくことを意味する。

4．計算ユニット

ここでは、内部でどのような演算を行っているかを述べる。

変数としては、エージェントそれぞれに対して、座標（x, y）と２つのfussy値（$f_{p\to q}$）が存在する。また環境エージェントには天候を記述する変数（env）が存在し、それらが初期データに書き込まれている。この初期データを読み込みエージェント「A」、「B」、「C」と環境が生成される。

環境における変数（env）は内部的に、$-2 \le (env) \le 1$ なる整数として設定され「はれ＝１」、「くもり＝０」、「あめ＝-１」、「ゆき＝-２」である。また、環境は、現在の天候から、最大でどちらかに１ずつしか変化しないようになっている。つまり、現在のターンが「はれ」であれば、次のターンの天候は「くもり」もしくは「はれ」のままの状態となり、「あめ」や「ゆき」の状態にはならないような仕様となっている。この天候変化によって予め設定されている変数wは、エージェントが移動する際に渡される。

次にアクターエージェントの行動であるが、環境が天候変化によって生成した変数wと他エージェントとの関係変数であるfussy値により移動先の座標を決定する。移動には、次の式を用いる。

「A」の移動先の座標を（x'_A, y'_A）としたとき、

$$\begin{pmatrix} x'_A \\ y'_A \end{pmatrix} = \begin{pmatrix} x_A \\ y_A \end{pmatrix} + w\left\{ f_{A\to B}\begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix} + f_{A\to C}\begin{pmatrix} x_C - x_A \\ y_C - y_A \end{pmatrix} \right\}$$

（ただし、移動前の「A」の座標を（x_A, y_A）、「B」、「C」の現在の座標をそれぞれ（x_B, y_B）、（x_C, y_C）とする。）ここで、wは、天候によりw = 0.25,0.25,0.5,1のいずれかの値をとるものであり、同様にして、$f_{A\to B}$と$f_{A\to C}$は、$-1 \le f_{A\to B}$、$f_{A\to C} \le 1$の値をとる。また、「A」の移動により、「B」と「C」の「A」に対するfussy値も変化する。例えば「A」が移動した際に「A」の移動後をA'としたときのA'とBの距離である$\overline{A'B}$は変化をするが、$\overline{A'B} = \sqrt{(x_B - x_{A'})^2 + (y_B - y_{A'})^2}$で表される。これをもとにして、「A」の移動により「B」、「C」の「A」に対するfussy値が以下のように変化するよう設計した。

$f_{B\to A} \ge 0$ の時

$$\begin{cases} \overline{AB} > \overline{A'B} \Rightarrow f'_{B\to A} = f_{B\to A} + \dfrac{1}{10}\cdot\Delta f \\ \overline{AB} < \overline{A'B} \Rightarrow f'_{B\to A} = f_{B\to A} - \dfrac{1}{10}\cdot\Delta f \\ \overline{AB} = \overline{A'B} \Rightarrow f'_{B\to A} = f_{B\to A} \end{cases}$$

$f_{B\to A} < 0$ の時

$$\begin{cases} \overline{AB} > \overline{A'B} \Rightarrow f'_{B\to A} = f_{B\to A} - \dfrac{1}{10}\cdot\Delta f \\ \overline{AB} < \overline{A'B} \Rightarrow f'_{B\to A} = f_{B\to A} + \dfrac{1}{10}\cdot\Delta f \\ \overline{AB} = \overline{A'B} \Rightarrow f'_{B\to A} = f_{B\to A} \end{cases}$$

となる。

ここでΔfは、$\dfrac{\overline{A'B}}{\overline{AB}}$

つまり $\Delta f = \dfrac{\sqrt{(x_B - x_{A'})^2 + (y_B - y_{A'})^2}}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}$

と設定した。

以上のようにして再計算された変数は、保存ファイルに書き込まれると同時に。次のターンで読み込まれることになる。以上が「A」の行動の流れであり、同様に「B」と「C」に関しても計算とファイルへの変数の保存が行われる。

Ⅲ．シミュレータでの実験と結果

1．シミュレーション-1

Ⅱで示したプログラムを利用して、以下の初期条件にてシミュレーションを実行した。

$$A(0,0)\ \ B(2,0)\ \ C(1,\sqrt{3})$$
$$f_{A\to B} = 0.82,\ f_{A\to C} = 0.5,\ f_{B\to A} = -0.82,$$
$$f_{B\to C} = -0.3,\ f_{C\to A} = 0.5,\ f_{C\to B} = -0.3$$
$$env = 0$$

先ず、それぞれのエージェントを、正三角形の頂点となるように配置した。これは、初期状態の距離をどれも一定にしたためである。さらに、fussy値に関しては、「A」は「B」を好きだが、逆に「B」は「A」を嫌っている。また、「A」と「C」はお互いに好き、「B」と「C」はお互いを嫌いであるという条件の下でシミュレーションを開始する。なお、初期状態の天候は「くもり」である。

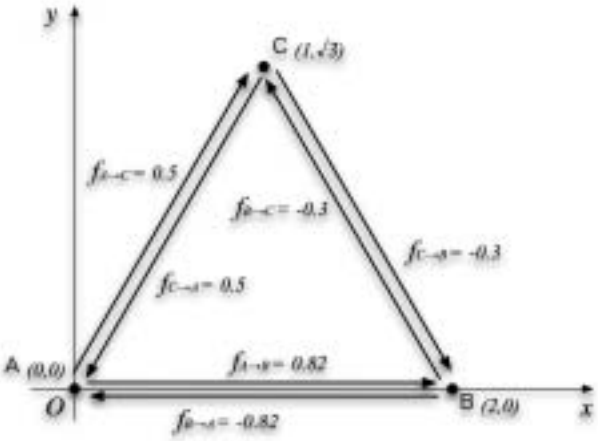

図5　エージェントの初期配置とfussy値

このうちそれぞれのエージェントの軌跡と、fussy値の変化を以下に示す。

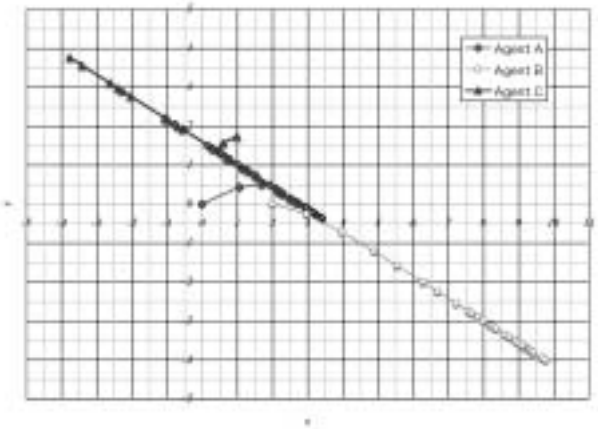

図6　エージェントの軌跡（0-30ターン）

図6は、開始から30ターンまでのエージェントの軌跡を示したものである。初期状態から、「B」は、他の二つのエージェントである「A」と「C」を嫌って、早い段階から他のエージェントからひたすら離れる方向に移動を開始する。一方で、「A」と「C」はお互いに好き同士である事から、早期からより近い位置に移動してくるが、「A」は「B」も好きなことから直接的に「C」に近づくというよりは、「C」にも近づこうとする一方で「B」にも近づくといった行動をとる。

また顕著なのは、全てのエージェントがある直線上に集約する方向で移動してくるということである。これは、初期状態を変えない状況で、天候だけがランダムに変化するという条件で何回かシミュレーションを行った結果、同様の結果が得られた。最終的には同一直線上にてそれぞれのアクターエージェントが移動を行うように行動するということが見いだされた。

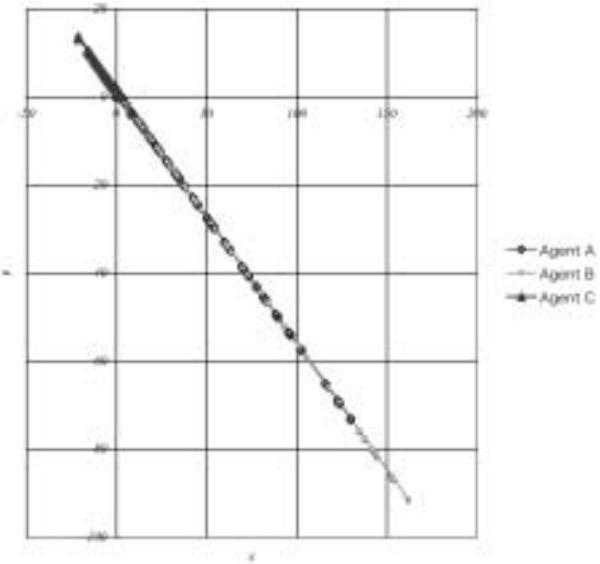

図7　エージェントの軌跡（0-100ターン）

引き続き、100ターンまで終了した時の結果を図7に示す。それぞれのエージェントは、同一ベクトル上にて移動をすることになり、同一直線上にて1ターンで移動する距離を拡大させながら3つのエージェントとも移動することとなった。興味深いことに、それぞれのエージェントの距離であるが、「B」においては「A」を「C」よりも、より嫌っているにもかかわらず、その距離を見てみると、$\overline{BC}$よりも$\overline{AB}$のほうが短いことが分かる。それには「C」が「A」および「B」に対してのfussy値と、それぞれの距離が影響していると思われる。「C」は「A」を好きであったが、距離が離れていくと$f_{C\to A}$の値が減少してくる。従って、だんだんと「A」との距離は開いていくことになり、同じベクトル上に嫌いな「C」も存在していることから「C」は「A」、「B」とは逆に移動することになる。また、離れすぎるとこれと逆

のことが起きることから、「C」は振幅を大きくしながら振り子のように移動し、お互いのfussy値がより小さいにもかかわらず距離的には近いという現象が起きたものと考えられる。

次に、fussy値の変化を図8に示すと、それぞれ対称的な数値の動きを示しているものがある。たとえば、「A」と「B」における$f_{A\to B}$と$f_{B\to A}$である。一方が増加すれば、他方は減少し、また一方が減少すれば他方が増加するという変数の変化をしていた。同様に$f_{A\to C}$と$f_{C\to A}$、$f_{B\to C}$と$f_{C\to A}$においては、互いに全く同じ数値をとりながらfussy値が変化をしている。これは、fussy値が各エージェントの距離にのみ依存する変数であることから、必然の結果となった。実際にシミュレーションを行う以前は、各アクターエージェントに設定された初期値からはさほど変化しないであろうと予想していたが、100ターンまで終了した時点で、顕著な動きを見せたのは、対となり変動する$f_{A\to B}$と$f_{B\to A}$、そして、$f_{A\to C}$と$f_{C\to A}$であった。$f_{A\to B}$と$f_{B\to A}$は、開始とともに急激に0に近づき、10ターンを過ぎると初めは好きでも嫌いに、初めは嫌いでも好きに変わってしまうという現象が起きる。さらに、$f_{A\to C}$と$f_{C\to A}$であるが、「A」と「C」のお互いがお互いを好きであったにもかかわらず、その勢いも急激に低下し、13ターンにはとうとうマイナスの値をとり、お互いを嫌っているという結果となった。10ターンから45ターンの間はどの数値もほぼ安定しているにもかかわらず、48ターンを過ぎると、$f_{A\to C}$と$f_{C\to A}$の数値が突然変化する。これに連動して$f_{A\to B}$と$f_{B\to A}$の数値が突然安定を崩して再び変化を始める。その一方で、$f_{A\to B}$と$f_{B\to A}$の数値以外はそれ以後も安定している事から見ると、「A」と「C」、「B」と「C」の関係は安定していたと考えられる。

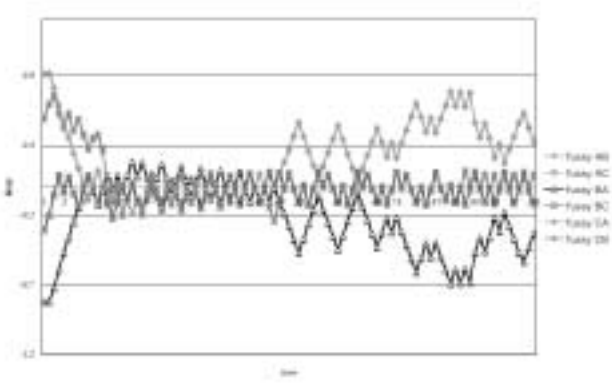

図8　fussy値の変化

さらに、この初期設定でシミュレータを実行し続けると、224ターンで定常状態とも考えられる状態となった。つまり、どの数値も変化しなくなったのである。これは、特定の座標に3点が集合してしまい、従って次のターンからは、アクターエージェントの座標、従ってアクターエージェント間の距離、そしてそれぞれのエージェントの持つfussy値というどの数値も変化しなくなったのである。この流れをみてみると、先ほどの100ターンから後「A」は、「B」とその同じベクトル上にいる「C」を追う形で第4象限上を移動するが、「B」と「C」が大きく第2象限上に戻るために「A」だけが「B」と「C」の後を追うような行動をとる。また、それぞれの距離は、$\overline{BC}$や$\overline{CA}$が最大となることはあっても$\overline{AB}$が最大となることはなかった。

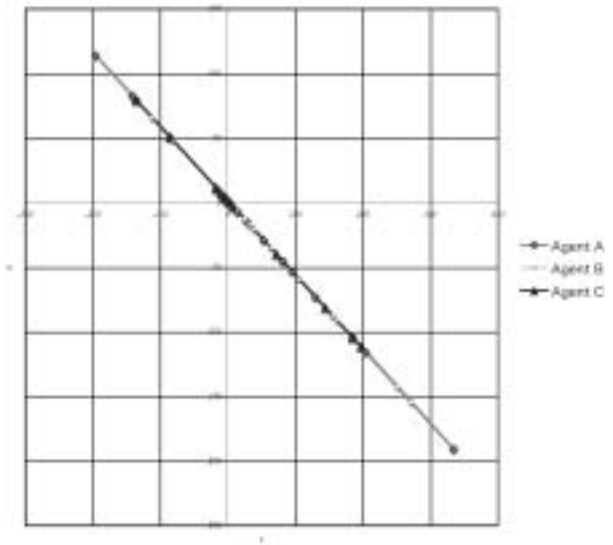

図9　「A」、「B」、「C」の座標の変化

この試行では、3点は最後には（-135, 79）付近で一点に集中し、以降は移動が起きなかった。

しかし一方で、$f_{A\to B}$は初期状態から概ね正数、即ち「A」は「B」がずっと好きだったのにもかかわらず、160ターンを境にして劇的な変化を始める。$f_{A\to B}$は160ターンから突然下降をはじめ、190ターンを迎えると一定して-1（つまり「とても嫌い」）となるのである。また、時期を同じくして、それまでは$f_{A\to B}$と対照的な動きをみせていた$f_{B\to A}$も含めて、その他全てのアクターエージェントにおけるfussy値は正数に向かい、最終的には全てが+1（つまり「最も好き」）となり3点が一点に集中するという事態を迎えるのである。このシミュレーションも初期状態を同じくして繰り返し実験を行ったが、最終的にアクターエージェントは、常にある一点で一致し、また、$f_{A\to B}=-1$となり、その他のfussy値はすべて+1で安定するという結果が得られた。

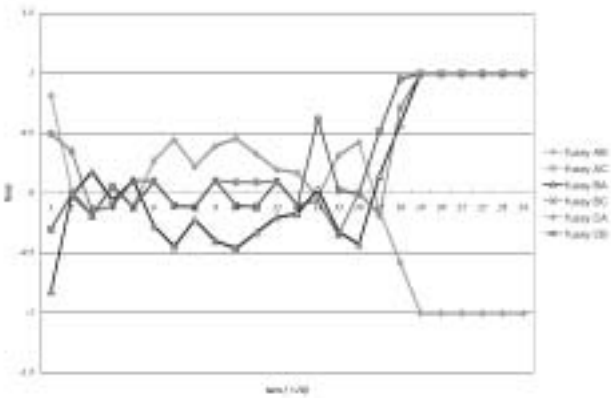

図10　それぞれのfussy値の変化

$f_{A\to B}$だけが負数（$f_{A\to B}=-1$）となり、その他のfussy値は最終的にすべて＋1で安定した。

今回作成したJAVAによるプログラムに関して、シミュレーション実行後に不備や問題が多く残る結果となった。例えば、実際にエージェントがその行動に関して支配されているものが天候とfussy値だけであり、それは実世界のそれとはほど遠いものとなってしまった。また、多くはfussy値の変化によりエージェントは次の自分の行動を決めることになるが、その行動距離によって、移動後の実際のfussy値の増減が大きく左右されることになり、従って次のターンでの行動に関して左右されるという結果になったまた、アクターエージェントの数もこのプログラムでは3つしか存在せず、アクターエージェントたちは、同一直線上に並ぶという結果になった。x-y平面上でシミュレーションを行ったにもかかわらず、アクターエージェントが他のアクターエージェントに対して、数ターンの処理を行った結果、3つのアクターエージェントが最も安定して移動できるということは解ったが、その平面を上手に利用することができなかった。

2．シミュレーション-2

シミュレーション-2では、シミュレーション-1での考察を基にコードの修正を行った。次のセクションではその詳細と結果を示す。プログラムコードに関する大きな変更はないが、修正コードでは、以下の改変を行った。

（1）偶然性による変化

シミュレーション-1における偶然性とは、変化天候によるものだけであった（天候により各エージェントの移動距離は変化するが、結果的に天候という偶然性は全てのエージェントに等しく作用してしまうために効果的ではなかった）のに対し、修正コードでは、移動のためにエージェントがデータを参照にして厳密に計算に従うのではなく、多少の柔軟性を持たせる仕様とした。これは、各アクターエージェントが同一直線上に並んでしまい、その後その直線上しか移動できないという問題を回避するためである。

さらに、各アクターエージェントにおけるfussy値にも同様に柔軟性を持たせる仕様とした。fussy値は距離だけに支配される変数であるために、例えば$f_{A\to B}$と$f_{B\to A}$は共通の数値である$\overline{AB}$のみに支配されることから、初期条件によってその推移は同一のものとなるか、全く対称的な動きになってしまうことになる。さまざまな柔軟性の度合いと各エージェントとのfussy値の推移に関するシミュレーションを繰り返した結果、以下の式を与えることとした。

「A」の移動により「B」、「C」の「A」に対するfussy値は、

$f_{B\to A}\geq 0$の時

$$\begin{cases}\overline{AB}>\overline{A'B}\Rightarrow f'_{B\to A}=f_{B\to A}+\dfrac{1}{10}\cdot\Delta f+\Delta b\\ \overline{AB}<\overline{A'B}\Rightarrow f'_{B\to A}=f_{B\to A}-\dfrac{1}{10}\cdot\Delta f+\Delta b\\ \overline{AB}=\overline{A'B}\Rightarrow f'_{B\to A}=f_{B\to A}+\Delta b\end{cases}$$

$f_{B\to A}<0$の時

$$\begin{cases}\overline{AB}>\overline{A'B}\Rightarrow f'_{B\to A}=f_{B\to A}-\dfrac{1}{10}\cdot\Delta f+\Delta b\\ \overline{AB}<\overline{A'B}\Rightarrow f'_{B\to A}=f_{B\to A}+\dfrac{1}{10}\cdot\Delta f+\Delta b\\ \overline{AB}=\overline{A'B}\Rightarrow f'_{B\to A}=f_{B\to A}+\Delta b\end{cases}$$

となる。

ここでΔfは、$\dfrac{\overline{A'B}}{\overline{AB}}$

つまり$\Delta f=\dfrac{\sqrt{(x_B-x_{A'})^2+(y_B-y_{A'})^2}}{\sqrt{(x_B-x_A)^2+(y_B-y_A)^2}}$

とする。

（2）シミュレーション-1では、「A」の移動先をA'とした時の$\overrightarrow{AA'}=w\cdot\{f_{A\to B}\cdot\overrightarrow{AB}+f_{A\to C}\cdot\overrightarrow{AC}\}$としてシミュレーションを行った結果、自分以外のアクターエージェントが大きく離れる、もしくは接近するという移動を行った際に、自分もその影響を受け、次のターンでは自分も大きな距離を移動することになる。当初は、このことは全くの想定外であったが、実際のシミュレーションの結果、このことが連続的になり、ターンが進むとその移動距離は急激に増大してしまうという結果となった。従って、各エージェントは自分以外のエージェントからは移動のベクトルに関しては影響を受けるが、その移動距離に関しては影響を受けすぎないという必要があり、仕様を変更した。仕様変更後による「A」の移動後の「A'（$x_{A'}, y_{A'}$）」は、

$$\begin{pmatrix}x_{A'}\\y_{A'}\end{pmatrix}=\begin{pmatrix}x_A\\y_A\end{pmatrix}+\frac{w}{\sqrt{(x_{A'}-x_A)^2+(y_{A'}-y_A)^2}}\left\{f_{A\to B}\begin{pmatrix}x_B-x_A\\y_B-y_A\end{pmatrix}+f_{A\to C}\begin{pmatrix}x_C-x_A\\y_C-y_A\end{pmatrix}\right\}+\begin{pmatrix}\Delta B_1\\\Delta B_2\end{pmatrix}$$

とする。

ただし、（1）を加味し、エージェントの移動に関しては柔軟性の要素であるΔB_1とΔB_2も含まれるために、各アクターエージェントの移動は単位ベクトルを基本としながらも、その移動距離は、厳密に$\left|\overrightarrow{AA'}\right|=1$とはならない。

この修正プログラムを使用して、シミュレーションを実施した。（1）、（2）以外のプログラムの変更は行われておらず、また、初期条件は、最初のシミュレーション実験と同様である。

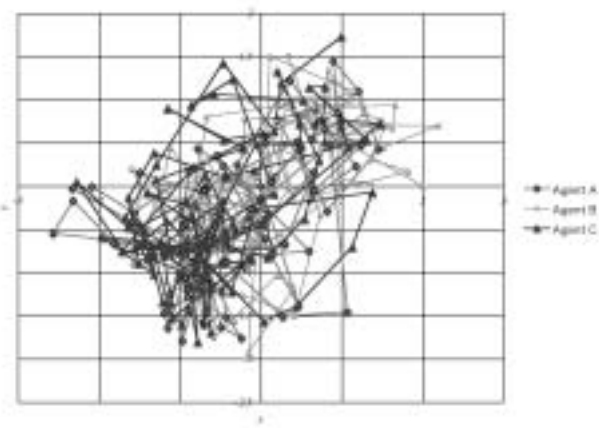

図11　シミュレーション-2の結果によるエージェントの座標

「A」、「B」、「C」とも同一直線上に並ぶことはなく、局所的な範囲で絶えず動き続けた。

それぞれのエージェントは、同一直線上に並ぶことはなく、それぞれが複雑に動き続ける結果となった。これは、主にエージェントの移動先の座標の指定にある程度の柔軟性を与えた結果であり、自然界でのアリなどの昆虫などの動きに繋がる動きではないかと推測される。また、各エージェントは、それぞれランダムに動いているわけではなく、自分が好きな別のエージェントが接近してきた時にはその座標付近にとどまるような動きを見せ、嫌いなエージェントが接近した時には大きく避けるという動きをとっていることが伺える。

次に各アクターエージェントのfussy値の推移のデータを提示する。

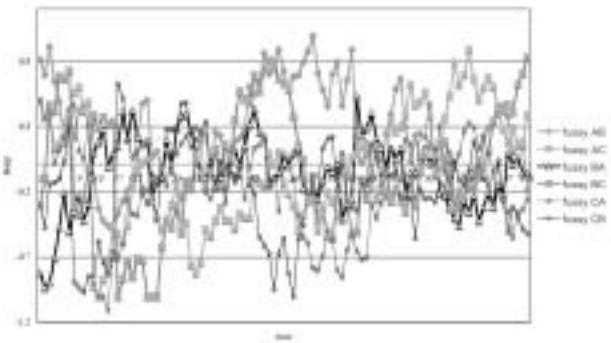

図12　修正プログラムによるfussy値の変化

fussy値にも乱数要素を採用したために前回のプログ

ラムと比べてそれぞれのfussy値の相関関係は対照的ではなくなり、それぞれのfussy値に「ぶれ」が見られるようになった。初期では、例えば$f_{A\to B}$と$f_{B\to A}$では、一方が低下した時に他方は上昇するといった相関関係が見られるが、それ以後にお互いに相関関係を示さなくなり、お互いが独立したような挙動をするようになった。同様に$f_{A\to C}$と$f_{C\to A}$においても、前回のプログラムでは完全に数値が一致して推移していたが、例えば、40ターン前後ではその数値は符号も含めて大きく異なる数値を示すこととなっている。これは、$f_{B\to C}$と$f_{C\to B}$でも同様の傾向を示し、この二つの変数も全く無関係でお互いに独立した変数のごとく挙動しているように観測された。このように、実はお互いに従属する関係の変数であるが、簡単な不確定要素を追加するだけで、一見すると複雑な挙動を示しているようなグラフとなった。

3．シミュレーション-3

先の2つのシミュレーションにおける3つのエージェントにおいては、あるエージェントは他のエージェントたちの状態を参照しながら次の行動を行っている。つまり、エージェントたちは、それぞれが他のエージェントの座標やエージェント間の距離をいうものを予め知っていることが前提となっている。例えばそれがずいぶん遠く離れてしまったエージェントでも、エージェントが次の行動を行うときには様々なパラメータが正確に反映されてしまうということである。

ここで人間の「視界」に似た近傍範囲の概念の導入を試みた。つまり、エージェントは、それぞれに自身からの近傍範囲を持ち、あるエージェント自身からの距離に応じて「視界」は弱まっていくと仮定する。また、近傍範囲からの他のエージェントの影響力Rをとし、自身からの距離をDとすると、$R = e^{-aD}$
という数式により視界を表現することにする。つまり、仮にあるエージェントとの距離が0（同一座標上に2点が存在する状態）であるならば、$R = 1$となり、そこから離れるにつれて影響力Rは0に近づいて減少していくこととなる。

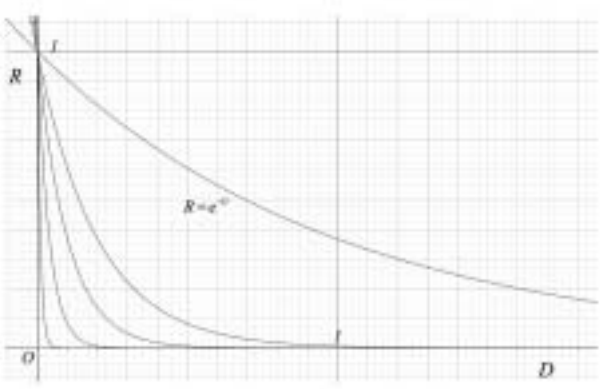

図13　$R = e^{-aD}$のグラフ

グラフはそれぞれ、$R = e^{-D}$、$R = e^{-5D}$、$R = e^{-10D}$、$R = e^{-25D}$、$R = e^{-50D}$ を示している。

この概念をエージェントに導入することにより、エージェント間が離れてしまった場合には、お互いを見つけることが困難になり、お互いの影響力が弱くなっていくことが予想され、実際のシミュレーションでは、3つのエージェントに$a = 25$を与えた際、以下のような結果が得られた。この試行における他の全ての初期設定は、シミュレーション-2と同様であり、全てのエージェントの近傍範囲において$a = 25$として行った。

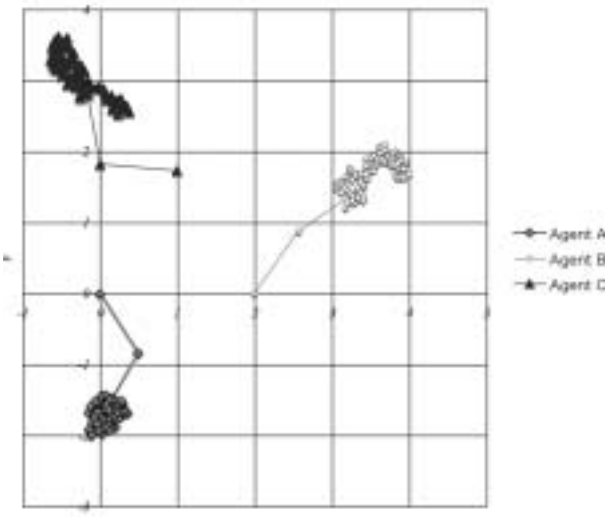

図14　a＝25におけるシミュレーション

この試行では、全てのエージェントの近傍範囲が非常に狭く設定されていることから、自身以外のエージェントの座標を確認することができず、エージェントは現在の座標周辺で停滞することとなる。一方で同様の初期値における別の試行において、たまたま自身の近傍範囲に

他のエージェントが接近してきたためにそこで影響を大きく受け、お互いが影響し合って行動をとるようになるという結果も得られた。

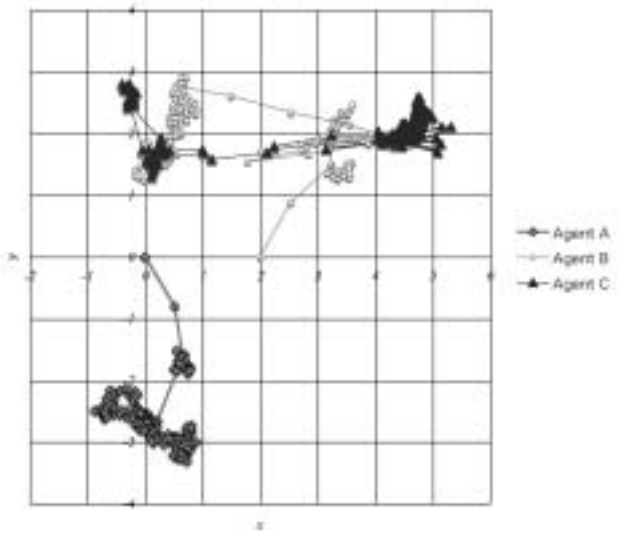

図15　a＝25におけるエージェントの影響し合う例

さらにαの値の変化により、エージェントの近傍範囲は変化をするが、例えば$\alpha=10$においては、初期の段階で「B」と「C」が影響し合い、「C」の後を「B」が追いかけるというような行動も確認できた。この試行における「B」と「C」のお互いのfussy値の変化であるが、ターンが進むと「B」と「C」はお互いを「好き」と認識して同調行動をとるようになったと推察される。

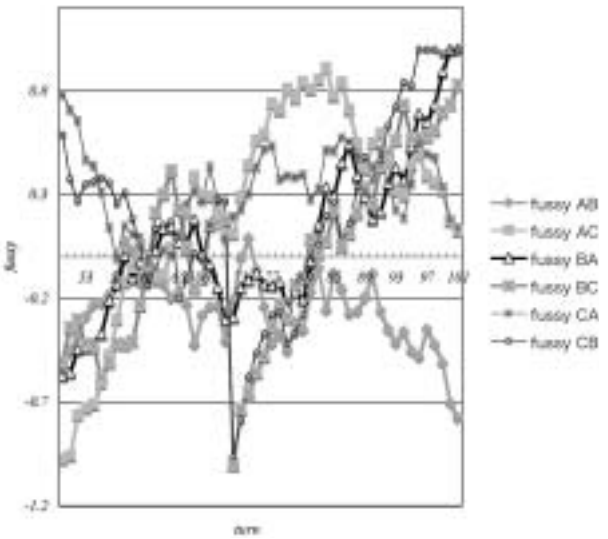

図16　a＝10におけるfussy値の変化（50ターン以降）

おわりに

この3つのシミュレーションにより、一定の入力によって複雑な挙動を行うように見えるエージェントを作ることができたのは大きな前進であった。しかし、想定する社会全体をコンピュータ上で再現するということのためには、 さらに多くの問題が露呈したことも事実である。先ず、人間の行動とひとくくりに述べても、人間がどのように行動しているのかという事に関して、実際の人間はその置かれている環境のなかで、様々なことを考え、様々な入力から出力を行っている。それは例えば、自分が仲間といる時と組織にいる時とは、同じ入力でも異なった出力をするであろうし、また同じ入力であっても、周囲の状況・時間によっても異なった出力を行う可能性も存在する。このような要素も入出力に関係していることから、人間の行動を完全にモデル化することはできなかった。従って、個々のアクターエージェントに対して膨大な入力を行い、入力された情報を取捨選択して、必要なデータだけを処理し、出力する機構が必要であるということが解った。しかし一方で、「好き」や「嫌い」という単純な判断と出力だけでも、時間を追うごとにそれぞれの数値は変化し、複雑に振る舞う可能性が見いだせたことは実験を行った有益な結果だったといえよう。

次に、現在のアクターエージェント数は、3という非常に特殊な空間における実験であったために、実際に組織として彼らを3人の動作させることができず、人間の行動といった複雑なシミュレーションを実行することは不可能であった。また、彼ら3つのエージェントは同じアルゴリズムで動作し、また、彼ら三者は対等な関係である前提での試行であったが、実際の社会においてそのようなことは存在せず、社会とは時間や場所の違いによって、様々な人間との結びつきも変化することから、彼らエージェントに個性をどのように与えるのか、またあるエージェントが別のエージェントの行動を管理するといった階層構造も、これからも研究で考えていかなければならない。

今後の改良点は上記2つの反省から、一つはアクターエージェントの数を増やすことにより、SAとマルチエージェントシステムの強みである中枢制御系を持たない意思決定モデルを実現するとともに、個々のエージェントにも複雑な入出力と制御・処理機構を持たせることに

よって、各エージェントにもより人間に近い判断処理を持たせ、それぞれに複雑な動きをさせることを目標とすることである。これには増大するエージェントの変数をどのように処理すべきかという技術的な問題や、エージェントそのものを賢くするために心理学などの人間の行動についてさらに知る必要がある。二つ目にはアクターエージェントどうしの関係性をどのように作っていくべきかということである。アクターエージェントがある関係性を持っているときに、それらをどのようなモデルで描写するのが適切かという事を深く考える必要があり、現在は２次元上で行われているエージェントの行動の軌跡などもそれは、３次元空間としてエージェントの行動を表せた方がいいのか、もしくはレイヤー構造にした方がいいのかなど新たな方法を模索する必要性があるであろう。

これらの反省点と改良点を考慮した上で、現在のプログラムの開発を推し進めることにより、これらマルチエージェントシステムによるシミュレーションが、社会科学における様々な問題を解くためのツールとして応用できるのではないかと考えられる。

参考文献

Brooks, Rodney A.,“Intelligence without representation”, *Artificial Intelligence*, Vol. 47, 1991, pp.139-159

Brooks, Rodney A.,“A Robust Layered Control System for a Mobile Robot”, *IEEE Journal of Robotics and Automation*, Vol. 2, No.1, 1986, pp.14-23.

Dreyfus, H. L・Dreyfus, S. E.・椋田直子訳『純粋人工知能批判』（アスキー海外ブックス）ASCII出版、1987

Epstein, Joshua M・Axtell, Rober・服部正太・木村香代子訳『人工社会-複雑系とマルチエージェント・シミュレーション-』共立出版、株式会社構造計画研究所発行、1999

Geneseles, M. R・Nicolson, N. J・古川康一訳『人工知能基礎論』オーム社, 1993

Greenblat, Cathy Stein・新井潔・兼田敏之訳『ゲーミング・シミュレーション作法』共立出版、1994

Selfridge, O.,“Pandemonium: A Paradigm for Learning” *Proceedings of Symposium on the Mechanization of Thought Processes*, 1959, pp.511-529.

有沢誠『ソフトウェア工学』（岩波コンピュータサイエンス）岩波書店、1988

市川真一編『思考』（認知心理学４）東京大学出版、1996

木村哲男・菅原研次『～エージェントの基礎と応用～エージェント指向コンピューティング』ソフト・リサーチ・センター、1995

沼岡千里・大沢英一・長尾確『マルチエージェントシステム』（分散強調メディアシリーズ１）共立出版、1998

馬場登・山田誠二『人工知能の基礎Fundamentals of Artificial Intelligenc』（情報系教科書シリーズ第15巻）昭晃堂、1999

廣瀬通孝・小木哲朗・田村善昭『シミュレーションの思想』東京大学出版会、2002

矢田光治『人工知能の話し』日刊工業新聞社、1986

山田誠二著『適応エージェント』（認知・科学モノグラフ⑧）共立出版株式会社、1997

レフ・セミョノヴィチ・ヴィゴツキー・柴田義松『思考と言語』明治図書、1962