プログラム自動生成手法によるジレンマゲームにおける行動戦略の進化と分析

今福 啓　獨協大学経済学部
email: k03082@dokkyo.ac.jp

1. はじめに

人や組織などの行動主体から構成される実社会では、それぞれの主体が行動を繰り返し、さまざまな相手と相互作用することで成り立っている。また、実社会におけるさまざまな状況を簡略化した囚人のジレンマ問題のように、合理的に行動するにもかかわらず社会全体では非合理的な結果となってしまうジレンマ的な問題が存在している。

このようなジレンマを回避して、より効率的な社会を作り出すにはどうすれば良いのかについて扱ったものとして、コンピュータ内部で実社会での行動主体を模倣し、シミュレーション結果から、どのような要素が合理的な結果につながるのかを分析する研究がある。これらの研究では、周囲の他者の行動を模倣して自らの行動ルールを決定し、実行される行動がどのように変化していくのかを分析している。その結果、行動主体をつなぐネットワーク構造が協調に影響を与えること（塚本 2009）や、行動の模倣を限定的にすることが有効であること（脇山 2008）、そしてネットワーク構造と行動の模倣を同時に行うことが有効である（谷本 2008）といった結果が得られている。

しかし先行研究の多くでは、過去に実行した行動と次の行動を対応づけるテーブルを作成し、テーブルの内容を他者との相互作用を通じて得られた結果をもとに変更するといった、単純な手順が用いられてきた。また、結果の多くは主体ごとに行動ルールが協調あるいは裏切りとなったまま安定している。人の考えや行動は社会状況に応じて変化することから、決まった行動を安定的に繰り返す結果は不自然とも考えられ、得られた知見が人の行動決定過程に沿ったものとなっているかは明らかではない。行動が時々刻々と変化するシミュレーション結果も得られているが、そのような結果が得られることはまれであり、特定の行動を繰り返す結果に至ることがほとんどであると結論づけられている（星野 1998）。

本研究では、ジレンマ的な社会状況において、合理的に行動する主体の行動ルールをコンピュータプログラムにより表現することで、どのような過程を経て行動が変化していくのかを分析する。プログラム形式の行動ルールは、プログラムの自動生成手法により周囲との相互作用により進化するため、社会状況に応じてどのように変化していくのかという過程を、プログラムが変化する過程を通じて分析できる利点がある。

シミュレーションでは、個々の行動主体を接続するネットワーク構造を正確に模倣することが実社会のシミュレーションでは重要とされる。そこで、本研究では実社会と同様の構造を持つ「複雑ネットワーク」とよばれる手法により主体間の接続を構築し、パラメータの変更が行動に与える影響についても考察を加える。

以下では、2章で提案モデルを構築する際に用いるプログラム自動生成手法、3章では実社会でみられるネットワーク構造を模倣した複雑ネットワークについて、4章ではシミュレーションで扱うモデル設定および問題設定と、シミュレーション結果および考察を行う。

2. プログラム自動生成手法

プログラム自動生成手法とは、目的のプログラムを人が作成する代わりに、コンピュータ内部でプログラムを進化させることで構築する手法である。具体的な手法として、本研究で用いるGraph Structured Program Evolution（GRAPE）（Shirakawa2007）や、遺伝的プログラミング（GP）（Koza1992）、Grammatical Evolution（GE）（O'Neill2003）、遺伝的ネットワークプログラミング（GNP）（片桐2002）がある。

GRAPEは、処理内容の置かれたノードを複数用意し、命令で使用する複数の数値を格納するデータセットと合わせて、１つのプログラムを表す個体とする。そして、個体を複数内部に配置した構造となっている。図1にノードの例を示す。１つのノードは整数を一次元状に並べた内容で構成されており、それを決められたルールにもとづいて、演算や条件分岐などの命令や、出力および使用するデータセット番号に変換する。また、各ノードには次に処理するノードの番号が記述されており、1つのノードの処理が終了すると指定したノードに移動し、移動先のノードを同様に処理する。出力するデータセットの番号が記入されているノードに達した際には、指定されたデータセットの値を出力し、プログラムが終了する。個体の内容をプログラムに変換した例を図2に示す。図2に見られるとおり、木構造を持ちループ構造のプログラムを作成することが困難なGPとは異なり、GRAPEではループが存在することから、同じ命令を繰り返し実行することが可能となっている。

図1のノード内の値は、ノード番号、命令の種類、移動先のノード番号（2つ）、使用するデータセット番号（3つ）の順に並んでいる。これらの要素のうち、移動先のノード番号とデータセット番号の一部は、命令の種類によっては使用されない。そのためGRAPEでは、GEやGNPのようにノードを可変長として扱う手法と異なり、固定長という単純な構造ながら、他の手法と同様に命令に応じて可変長として処理できる利点がある。本研究で使用したノードとデータセットの具体的な内容は、4章で述べる。

GRAPEでは、初期状態としてすべてのノードにランダムな整数値を代入し、データセットには設計者の指定する初期値を格納する。そして後述する手法で個体を進化させ、正解に近い値を出力する個体には高い適合度を与える。最終的に目的となる出力値が得られた場合、正しいプログラムが作成できたとする。例えばx^2を計算するプログラムを自動生成する場合、初期値としてデータセットに1, 2, 3, 4, 5を順に入力した際の結果が、それぞれを2乗にした1, 4, 9, 16, 25となったとき、正しいプログラムが構築されたとする。

プログラムを進化させる際には、複数の個体の中から適合度の高いものを選択し、選択した個体のノードに対して交叉、突然変異と呼ばれる操作を加えて一部を変更し、新たな

個体を作成する。そして、あらかじめ用意した個体と同じ数の新たな個体を作成した後、個体を入れ替えて世代交代させることでプログラムを進化させる。

GRAPE では、個体の多様性が効率的に目的のプログラムを得るために重要であるとして、選択および世代交代の方法に MGG（Minimal Generation Gap）（佐藤 1997）とよばれる手法を採用している。しかし、MGG は、適合度の低い個体の選択を許すことと、親となる個体が次世代に残ることから、大域的最適解への収束の遅れと局所最適解からの脱却の遅れが指摘されている（小林 2009）。また、GRAPE ではグラフの有効な構造を保持した交叉が出来ないため、交叉の効果が限定的であることも示されている（石堂 2009-1）。

そこで本研究では、JGG（Just Generation Gap）（小林 2009）とよばれる世代交代モデルを用いてプログラムを進化させる。また交叉は行わず、突然変異のみでプログラムを更新する。JGG では、選択した個体をもとに交叉、突然変異により新たな個体を複数個作製し、その中で最も良い適合度を持つ個体を次世代に残すため、MGG での問題点を解決できるとされる。そして、交叉を行わずに突然変異により個体の一部を変更するにとどめ、目的となるプログラムへの収束速度を向上させる。

本論文で用いる JGG による新しい個体の生成方法は、以下のとおりである。

1. GRAPE の全個体から、一部を重複しないように n_p 個選択する。
2. 1.で選択した個体から 1 つをランダムに選ぶ（選択）。
3. ノードの値を、確率的にランダムな値に変更する（突然変異）。
4. 2～3.の手順により、n_c 個（n_c>n_p）の新たな個体を作製する。
5. 4.で作成した個体を適合度の高い順番に並べ替え、上位 n_p 番目までの個体を 1.で選択した個体と入れ替える。

また、従来提案されている GRAPE では、出力のノードに至らないまま一部の命令を繰り返し実行し続ける、無限ループになることがある。無限ループとなると問題の解を出力しないため、本研究では無限ループが生じた場合、その個体を破棄して新たな個体を作成することで、プログラム生成の効率化を図る。

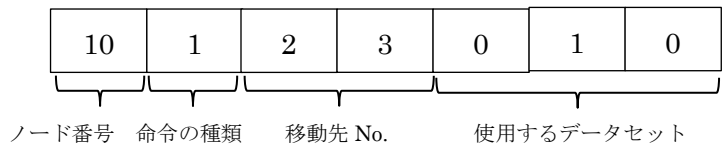

図 1　GRAPE のノード

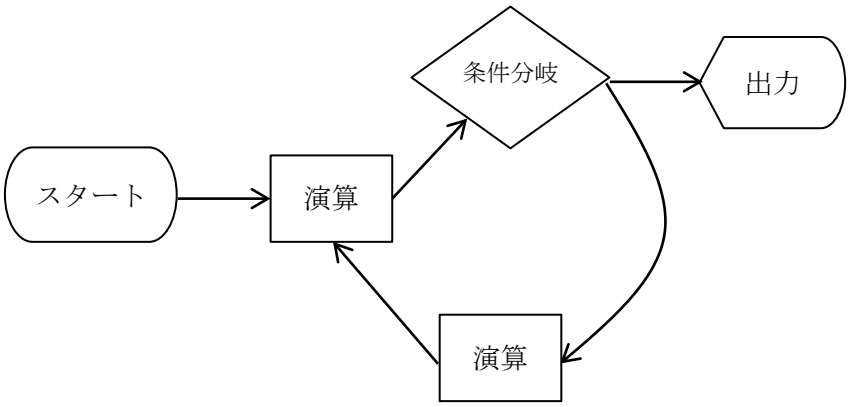

図 2　プログラム構造の例

3.　行動主体間のネットワーク構造

人と人のつながりのような実社会においてみられるネットワークを分析し、コンピュータ内部に実社会と同様のネットワーク構造を作り出す方法を提案する分野に「複雑ネットワーク」がある（増田 2005）（今野 2008)。ネットワークを構築する代表的な方法には BA モデルや頂点非活性化モデルがあるが、それぞれの手法だけでは実社会のネットワークの持つ特徴である、小さい頂点間距離と大きいクラスター係数を持つネットワークを表現できない。その問題を解決した手法として「SW 頂点非活性化モデル」が提案されている。このモデルは、ネットワーク構築の際に、BA モデルと頂点非活性化モデルを確率的に使い分けることで、小さい頂点間距離と大きいクラスター係数を再現することが可能である。

本研究では、この手法を用いて実社会の行動主体間のネットワークをモデル化してシミュレーションを行う。ネットワークモデルを構築する際、いくつかのパラメータ設定が必要となるが、本研究ではその違いがシミュレーション結果にどのような違いを生じるのかについて比較を行う。

4.　問題設定とシミュレーション

提案モデルを用いたシミュレーションを行った。シミュレーションでは、GRAPE にしたがって行動を決定する行動主体間を、3 章で述べたネットワークで接続して繰り返し囚人のジレンマゲームを行い、ネットワーク構造が行動ルールに与える影響について考察する。

4.1 GRAPE のパラメータ

1 つのプログラムを表す個体は、図 1 の構造を持つノード 50 個と、Boolean 型のデータセット 12 個から構成される。各ノードには、並ぶ順番に応じてノード番号を 1 から 50 まで割り当てる。また、データセットにも並ぶ順番に応じて 0〜11 の番号を割り当てる。

ノードで使用する命令の種類は表 1 のとおりである。AND, OR は、ノード中の最初の 2 つのデータセット番号で指定する値を用いて論理演算を行い、3 つ目のデータセット番号に結果を代入する。そして次に処理するノードとして、移動先 No.の 1 つ目で指定するノードに移動する。NOT も同様に、最初のデータセットの値を用いて論理演算を行い、2 つ目で示すデータセット番号に結果を代入した後、移動先 No.の最初で指定されたノードに移動する。SWAP は、最初の 2 つのデータセット番号の値を入れ替え、移動先 No.の 1 つ目のノードに移動する。IF は、最初の 2 つのデータセットの値を比較し、等しければ最初の移動先 No.に、等しくなければ 2 つ目の移動先 No.のノードに移動する。OUTPUT は、1 つ目で指定するデータセット番号の値を出力する。

各ノードの初期値は、命令の種類は 0〜5 のランダムな値、移動先 No.は 0〜49 のランダムな値、使用するデータセット番号は 0〜11 のランダムな値とした。また、データセットの初期値は全て False とした。False および True がどのような行動に対応するのかは、4.3 節で述べる。データセットにどのような値を記憶するのかは、4.3 節で述べる。

表 1　使用する命令

命令の種類	命令の内容	使用するノードの値
0	AND	移動先 1 つ データセット 3 つ
1	OR	移動先 1 つ データセット 3 つ
2	NOT	移動先 1 つ データセット 2 つ
3	SWAP	移動先 1 つ データセット 2 つ
4	IF	移動先 2 つ データセット 2 つ
5	OUTPUT	データセット 1 つ

4.2 ネットワークのパラメータ

SW 頂点非活性化モデルを構築する際、最初に一部のノード間をすべて接続した完全グラフを作成する。そして、接続を持たないノードから 1 つ以上の接続を持つ決められた数のノードに、SW モデルか頂点非活性化モデルを確率的に選択して接続することを全ノードに対して処理することでネットワークを構築する。よって、ネットワーク構築に必要なパラメータは「初期に完全グラフを構築するノード数 n_1」「他のノードへの接続リンク数 n_2」「SW モデルを選択する確率 p」となる。

シミュレーションでは、パラメータ$(n_1, n_2, p)=(4, 3, 0.5)$と$(n_1, n_2, p)=(10, 5, 0.5)$とした場合のシミュレーションを行った。後者の接続数のほうが前者より多くなるが、その違いが結果に与える影響について考察を加える。

4.3 行動主体の行動とゲーム

行動主体は、接続のある他のすべての行動主体と決められた回数繰り返しゲームを行い利得を獲得する。そして、利得を最大化することを目的として、GRAPE で決定される行動ルールを進化させる。

各主体は、ネットワークで接続されている相手のすべての各々と「囚人のジレンマゲーム」または「タカハトゲーム」を 30 回行う。ゲームにおいて、各主体は同時に協調および裏切り、あるいはタカ派、ハト派の 2 通りに限定された行動のいずれかを同時に提示し、その行動に応じて囚人のジレンマゲームの場合は表 2、タカハトゲームの場合は表 3 の利得を獲得する。囚人のジレンマゲームでは、最大利得を得るためには互いに協調することが必要であるにもかかわらず合理的な行動は裏切りを実行することであり、タカハトゲームでは 3 分の 1 の確率でタカ派の行動を実行することが合理的となる。シミュレーションでは、利得の最大化を目的とする行動主体がどのように行動ルールを進化させるのかを分析する。

シミュレーションでは裏切りまたはハト派の行動を False、協調またはタカ派の行動を True に対応させてデータセットに記憶し、GRAPE の個体から得られるプログラムをもとに False または True を行動として出力する。データセットには、自己と相手の 2 回前までに実行した行動をデータセットに記憶されている。データセットの最初の 3 つ（No.0~2）に自己の 1 ステップ前の行動、次の 3 つ（No.3~5）には相手の 1 ステップ前の行動、その次の 3 つ（No.6~8）には自己の 2 ステップ前の行動、最後の 3 つ（No.9~11）には相手の 2 ステップ前の行動を記憶させた。ただし、主体が記憶する値は、その主体が直前にゲームを行った相手の行動を記憶したものであるため、これからゲームを行う相手が直前に実行した行動を記憶したものとは異なっている。

表 2　囚人のジレンマゲームでの利得

行動 （自分，相手）	利得 （自分，相手）
（協調，協調）	(3, 3)
（協調，裏切り）	(0, 5)
（裏切り，協調）	(5, 0)
（裏切り，裏切り）	(1, 1)

表 3　タカハトゲームでの利得

行動 （自分，相手）	利得 （自分，相手）
（タカ派，タカ派）	(-2, -2)
（タカ派，ハト派）	(2, 0)
（ハト派，タカ派）	(0, 2)
（ハト派，ハト派）	(1, 1)

4.4 行動主体の行動と進化

シミュレーションでは総数 100 の行動主体を用意し、それぞれの行動主体間を 4.2 節で述べたネットワークにより接続した。シミュレーションの 1 ステップにおける行動主体の行動は、以下の手順により行われる。3 節で述べた JGG にもとづき、最初に全行動主体からランダムに n_p=10 の主体を選択する。選択された主体は、それぞれ新たな個体を n_c=100 作成し、自らの新たな行動ルールの候補とする。そして、その中から現在の個体と置き換えるものを選択して更新する。

選択した 10 の行動主体のうちの 1 つを更新する手順は次のとおりである。まず、自らの個体と、接続する他の主体のすべての個体からランダムに GRAPE の個体を 1 つ選ぶ。そして、その個体が持つノードの整数値を 2%の確率で他の数値に変更（突然変異）する。ただし、突然変異は 90%の確率で実行し、10%の確率で元の個体をそのまま残す。このような新たな個体を 100 個作成し、自らの新たな戦略の候補とする。これらの個体は、個々に接続を持つ他のすべてと 4.3 節のいずれかの 1 対 1 のゲームを 30 回繰り返し行い、獲得した利得の合計が最も高い個体を現在の個体と置き換える。

以上をシミュレーションの 1 ステップとして 10000 回繰り返し、得られた結果を分析する。

4.5 結果と考察

4.2 節で述べたネットワークのパラメータを(n_1, n_2, p)=(4, 3, 0.5)とし、囚人のジレンマゲームを複数回対戦した際の平均利得と最大利得を図 3 に、協調率を図 4 に示す。図 3 では実線が平均利得、＋の記号が最大利得を表している。最大利得とは、すべての行動主体が接続のある他の主体と 30 回の対戦を行って獲得した利得の平均値のうち、最も大きな値を表す。結果は平均利得が 2.5 以下と低く、最大利得も 3 から離れ、裏切りが実行されている状況が多い。また協調率はステップが進むにつれて下降し、高い値とはなっていない。パラメータを(10, 5, 0.5)としたときの平均利得および最大利得の結果を図 5 に、協調率を図 6 に示す。図 5 では、平均利得が 3 に近づく状況が生じている。また図 6 では協調率が 1 に近づく状況がみられる。

パラメータを (4, 3, 0.5)から(10, 5, 0.5)に変更することにより、接続リンク数が 3 から 5 へと若干増える程度ではあるが、相互に行動ルールを参照する際に与え合う影響が大きくなり、より高い利得を得るために協調する割合が増えたと考えられる。

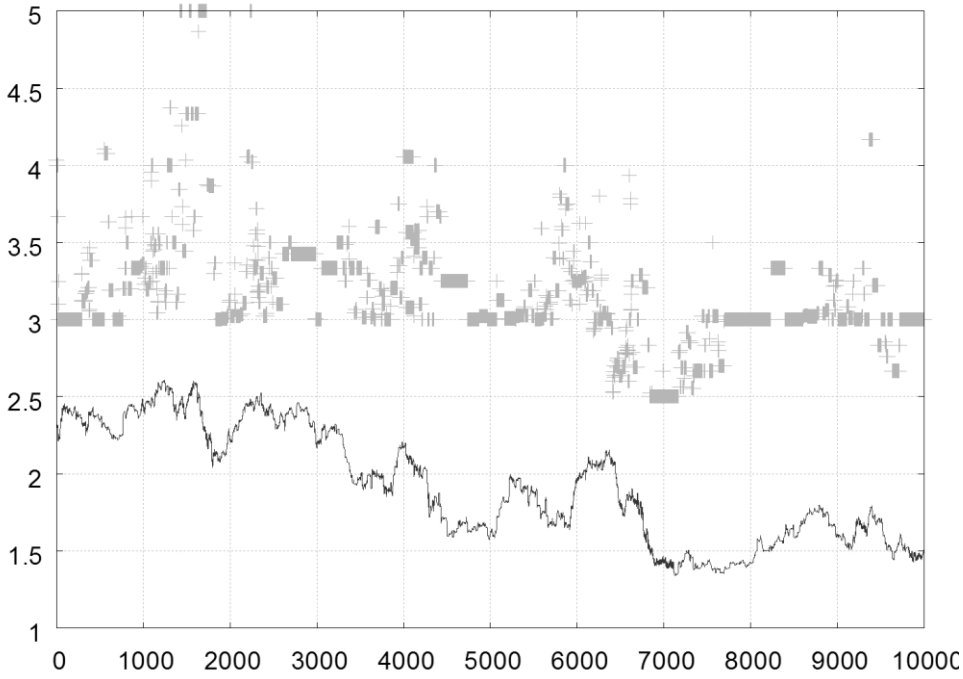

図 3　(n_1, n_2, p)=(4, 3, 0.5)における囚人のジレンマゲームの平均利得（実線）と最大利得（＋）

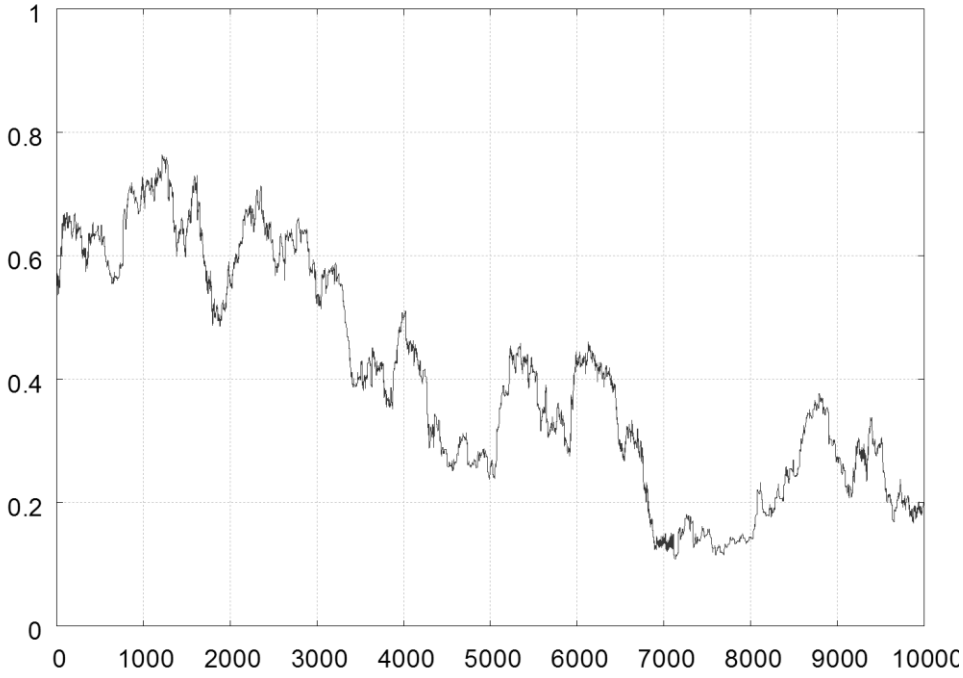

図 4　(n_1, n_2, p)=(4, 3, 0.5)における囚人のジレンマゲームの協調率

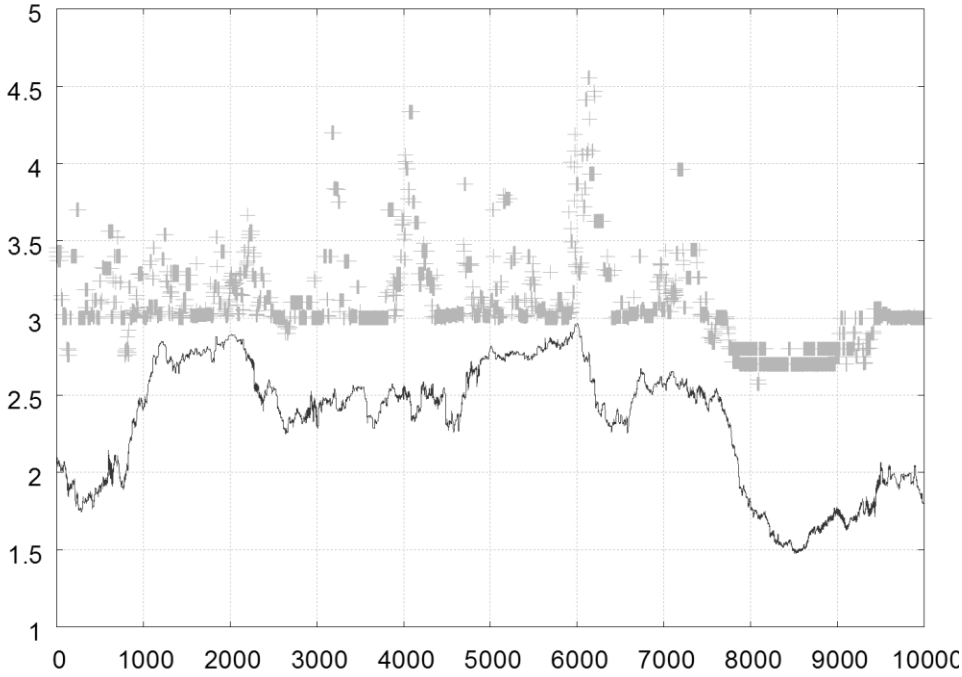

図 5　(n_1, n_2, p)=(10, 5, 0.5)における囚人のジレンマゲームの平均利得（実線）と最大利得（＋）

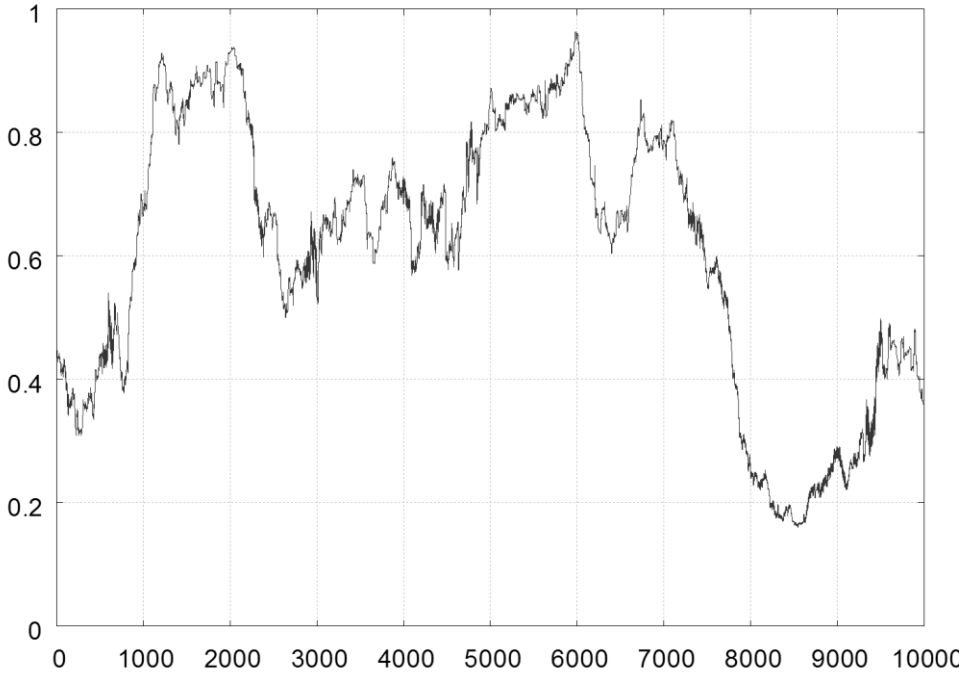

図 6　(n_1, n_2, p)=(10, 5, 0.5)における囚人のジレンマゲームの協調率

行動主体がどのように行動ルールを変更したのかを示す。図 7, 8 は、それぞれ GRAPE により構築された、パラメータを(n_1, n_2, p)=(4, 3, 0.5)としたときの行動主体 No.0 と 1 の行動ルールである。図 7 が 3636 ステップ、図 8 が 3805 ステップのルールである。図中の$に続く