F-003

マルチエージェントシミュレーション環境 GPGCloud における 実行順序生成機構の開発
The Sequence Generator for the GPGCloud Multiagent Simulation Environment

岩崎 裕太郎† 浅井 勇貴† 八槇 博史‡
Yutaro Iwasaki Yuki Asai Hirofumi Yamaki

1. はじめに

現実における社会現象は、マルチエージェントシミュレーションの特徴と類似性が非常に高い。このことから、政治・経済学分野において、マルチエージェントシミュレーションを用いた研究は非常に有用な研究手法となっている。しかし、政治・経済学分野の研究者がマルチエージェントシミュレーションを用いた研究を行おうにも、シミュレーションモデル（以下「モデル」）の開発や、実行結果の分析には情報工学分野における専門的な知識を必要とするため、その敷居は高いものであった。そこで、政治・経済学に特化した GPGSiM[1][2]などのシミュレータが開発された。GPGSiM などのシミュレータの開発により、モデル開発の点ではシミュレーション研究に参入しやすい環境となった[3]。

しかし、政治・経済学におけるシミュレーション研究では、GPGSiM 開発の際に、その知見から浮上したモデルの再利用性向上[4]や、モデルの特性を決定する外生変数であるパラメータが多くなる傾向から、パラメータ空間の次元数が大きくなり、したがって大量にシミュレーションを実行することが重要となった 。この問題を解決するために、GPGSiM により開発されたモデルの実行を、クラウド上の仮想マシン（以下「VM」）を用いて並行実行する環境を提供するシミュレーション支援環境 GPGCloud[5]の開発が進んでいる。GPGCloud において、利用者はモデルの選択、計算すべきパラメータの範囲を指定するだけで、入力されたデータに従って大量のシミュレーションを並行実行することが可能である。

GPGCloud を用いた並行実行の特徴は、指定したパラメータ空間全てのシミュレーションが完了せずとも、実行が完了したタスクの結果から、順次確認を行うこと可能であるという点である。しかし、実行順序を制御しない場合、パラメータ空間内から機械的順序で実行されるため、結果の蓄積もその順序で行われる。そのため、必要とするタスクが後回しにされるなど、研究効率の向上に余地を残すものであった。

本稿では、利用者からのパラメータの優先順位の指定方法と、優先順序に基づいたシミュレーション実行順序の生成方式の提案と GPGCloud への実装について述べる。

2. GPGCloud

2.1 政治・経済シミュレーション

現実における社会現象は、個々の主体がそれぞれ別々の行動をとり、その相互作用などによって形成されている。このことは，マルチエージェントシミュレーションにおける、エージェントを複数用意し、相互に干渉させた場合にどのような現象がおきるのかという点と対応付けが非常にしやすい。このことから、政治・経済学分野において、マルチエージェントシミュレーションを用いた研究は非常に有用な研究手法となっている。しかし、従来の Swarm[6]、SOARS[7]、artisoc[8]に代表されるマルチエージェントシミュレータはシミュレーションとしての機能をライブラリとして提供するだけで，モデル作成には情報工学分野の専門的知識を必要とするものであった。そのため、政治・経済学分野の専門家は参入しづらい環境となっていた。そこで、情報工学分野の専門的な知識は必要とせず、Java の入門書レベルの知識だけでもシミュレーション研究が行えるよう、GPGSiM などの政治・経済学分野に特化したシミュレータが開発された。

政治・経済学分野のシミュレーション研究においては、モデルの再利用性の向上が重要となる。既存モデルを再利用し、新たな機能を実装したりする際に、既存モデルの開発者がそのことを見越して拡張性を予め用意しておくことは困難である。そのために、モデルの再利用性、拡張性を向上させるために、モデルを中心とした開発者同士のコミュニティの必要性があげられる。

また、上述のようにこれらの分野のシミュレーションはパラメータの個数が多く、可能なパラメータの値の組（以下「パラメータセット」と呼ぶ）は膨大な数となる。これらのパラメータセット一つ一つについてシミュレーションを実行する必要があり、したがって研究の遂行上、大量のシミュレーションを実行することとなる。そのためには大規模な計算環境の利用が望ましいが、当該分野においては未だそのようなシステムの効率的な活用方法が確立されていない。これらの要求に応えるために、筆者らは GPGCloud システムの開発を進めてきた。

GPGCloud では、Web インタフェースからモデルの登録・公開やシミュレーションの実行・結果の参照などといった操作が提供されている。GPGCloud のユーザーは、GPGSiM で作成したモデルを公開することで、他の研究者とモデルを共有し、様々な意見交換を行うことができ、将来的に必要とされる拡張性を追加する機会を増やすことができる。登録されているモデルの実行にはクラウド上の VM を用いた並行実行機能を用いて行うことができる。そのため、GPGCloud の利用者は自身が高性能な計算機を持たずとも、GPGCloud を通して大量のシミュレーションを実行し、Web インタフェースを通して結果を確認することができる。

†名古屋大学大学院　情報科学研究科
‡名古屋大学　情報基盤センター

2.2 構成

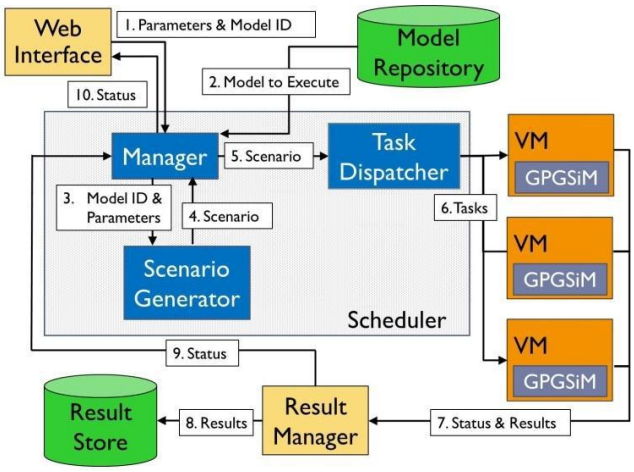

図 1: GPGCloud の設計

GPGCloud における並行実行機能の設計[9]を図 1 に示す。利用者は Web Interface を用いて GPGCloud にシミュレーションの実行要求を行う（1）。実行要求には、利用者が選択したモデルと、そのモデルで調べたいパラメータの型と定義域が含まれる。Manager は、利用者の要求したモデルを Model Repository からダウンロードする（2）。その後、Scenario Generator に シナリオの作成を要求する（3）。シナリオはタスクの集合であり、タスクは実行モデルと 1 回のシミュレーション実行に必要なパラメータの値との組である。Manager は、Scenario Generator からシナリオを受け取り（4）、Task Dispatcher に渡す（5）。Task Dispatcher は、Manager より受け取ったシナリオに含まれるタスクを各 VM に割り当てる（6）。VM は、タスク中のパラメータを用いて GPGSiM で実行し、実行状態および実行結果を Result Manager に通知する（7）。シミュレーションが正常に終了した場合、Result Manager は、タスクの実行結果を統合してシナリオの実行結果を作成し Result Store に格納し（8）、同時に Manager に状態を通知する（9）。Manager は、Result Manager から受け取った状態通知を Web Interface を通じて利用者に結果と格納場所を通知する（10）。Manager からのシミュレーション結果通知は Web Interface が結果履歴として保持しつづける。利用者は Web Interface から参照することでいつでも結果を受け取ることができる。

2.3 実行順序制御

GPGCloud による並行実行機能を用いることで大規模計算環境を持たなくとも、巨大なパラメータ空間に対するシミュレーションをクラウド上の VM で実行することができる。そのため、探索すべきパラメータセットが巨大になる傾向にある政治・経済学分野のシミュレーションも、高性能な計算機を持つことなく実行することが可能となった。

しかし、パラメータ空間が巨大になると実行結果も巨大となる。

また、探索すべきパラメータ空間の中には、優先的に検討を行いたいパラメータセットや、モデルの性質上優先度が高くないパラメータセットが存在する。そのため、優先的に検討を行いたいパラメータセットを膨大な実行結果の中から抽出するには手間がかかる。また、最優先で検討を行いたいパラメータセットが存在した際に、実行順序を制御しない場合、実行が後回しにされ、全てのパラメータセットの実行が完了してからでないと結果の確認が行えないケースも考えられる。

これらの問題を解決するために、実行順序の制御を提案した。実行順序を制御することで、結果の蓄積を実行者が優先的に検討を行いたいパラメータセットから行うことができる。そのため、利用者が実行結果を順繰りに確認していった際に、実行結果の確認を利用者が望む順番におこなうことができる。また、Result Store に格納される実行結果は Web Interface を通して逐次確認が可能なため、実行順序制御を行った場合、優先的に検討したいパラメータセットから実行され、Result Store に格納されるため、全てのパラメータセットの実行が完了するのを待つことなく、検討を開始することができるため、研究効率が向上する。

3. 実行順序生成機構

3.1 制御基準の指定方式

実行順序生成機構において重要となるのは制御基準の決定方法である。

実行順序制御において、同じモデルに対しても優先的に検討を行いたいパラメータセットは実行者によって異なる。そのため、どの利用者にも適用可能な順序制御を事前に定義することは困難であり、利用者自身が制御基準の指定が可能であることが必要となる。

また、パラメータ間で密接に関係しているパラメータも存在するため、制御基準の指定の際には、複数のパラメータ間の条件が指定できなくてはならない。そのため、全ての利用者に適用可能な一般的な制御基準のみで要求を満たすことは難しい。実行者が複雑なパラメータセット間の条件などを自由に指定できることが望ましい。

実行順序生成機構では実行者が比較関数を用いることで制御基準の指定を行うアプローチを本研究ではとる。

定義 1　比較関数

2 つのパラメータベクトル xi,xj(i≠j)が与えられた時、以下の値をとる関数

$$f(x_i, x_j) = \begin{cases} 1 & (x_i \succ x_j) \\ 0 & (x_i \sim x_j) \\ -1 & (x_i \prec x_j) \end{cases}$$

比較関数は定義 1 で表わされ、パラメータセットのパラメータ値を格納したリストx_i, x_jの 2 つを引数として与えた際に、$1,0,-1$を返すことで、どちらのパラメータセットが優先的かを表現する。返り値が1の時にはx_iを優先的に、-1のときはx_jを優先的に、0のときはどちらのパラメータセットから実行してもよいことを表わしている。

比較関数を用いて全てのパラメータセットの組み合わせに適応させ、関係付けを行った場合、各パラメータセット間の関係は半順序となる。利用者の求める優先順序で実行を行うためには、半順序関係を崩さないようにパラメータセットをソートする必要がある。

3.2 トポロジカルソート

半順序関係のパラメータセットをソートするアルゴリズムにはトポロジカルソートを採用した。トポロジカルソートは Program Evaluation and Review Technique[10]と呼ばれるプロジェクト管理手法のスケジューリングのために研究が開始された、無閉路有向グラフにおいて、半順序関係を崩さないようにノードを直列に並べるソート法である。

```
U = [ ] #訪問済みノードリスト
L = [ ] #ソート結果が入る
def visit(u):
    ノード u を訪問済みにする。
    for v in ノード u から出ている逆向き辺の全てのノード:
        if v が未訪問:
            visit(v)
    #全ての優先ノードの探索が完了した後、自身をリストに追加
    ノードを L に追加する

def tsort():
    #全てのノードに対して visit()を行うためのループを追加
    for n in 全てのノード
        visit(n)
```

図 2: トポロジカルソートのアルゴリズム

トポロジカルソートにはいくつかのアルゴリズムが存在する。実行順序生成機構においては、トポロジカルソートの手法の中から、深さ優先探索をベースとしたものを採用した。アルゴリズムを図 2 に示す。

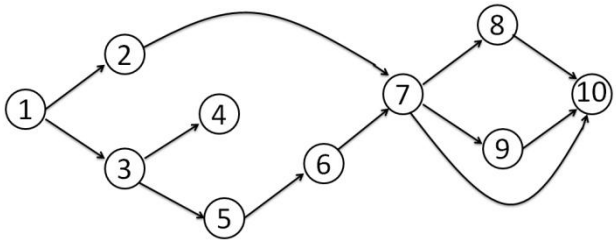

図 3: 無閉路有向グラフの例

図 3 は無閉路有向グラフの例で、ノード間の優先順序が辺で表わされている。図 2 のトポロジカルソートのアルゴリズムでは、あるノードを基準とした際に、そのノードよりも優先的なノードの方向、逆向き辺の方向へ探索を行っていく。基準としたノードよりも優先的なノード全ての探索が完了した後に自身を結果リストに追加するため、訪問済みノードリストを共有することで、既に探索が済んでいるノードに到達した場合、そのノードよりも優先的なノードは既に結果リストに格納されているため、そこで探索を止め、他のルートを探索するアルゴリズムとなっている。

トポロジカルソートを用いて図 3 の無閉路有向グラフをソートした場合、ソート結果の一例は次のようになる。

1 > 2 > 3 > 5 > 6 > 7 > 8 > 9 > 10 > 4

トポロジカルソートのアルゴリズムにおいて、逆向き辺の探索が基準としているノードから優先ノードを探索する行為である。そのため、逆向き辺の判定を比較関数に置き換えることにより、実行者が望む優先順序でソートすることができる。

4. GPGCloud 上での実装

4.1 実行順序生成機構の構成

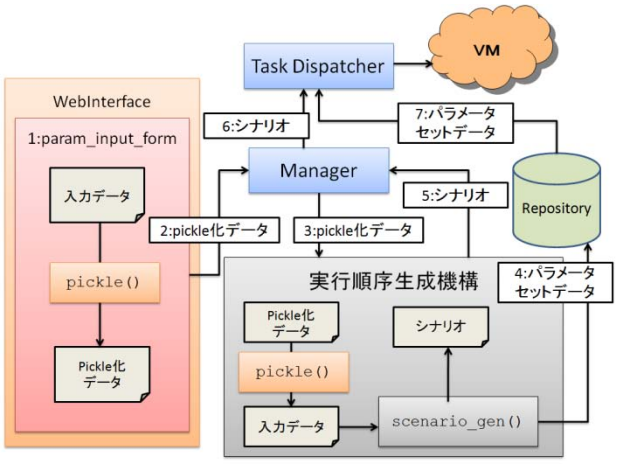

図 4: 実行順序生成機構の構成

実行順序生成機構は GPGCloud における ScenarioGenerator に相当する部分に実装を行った。実行順序を制御しない場合、タスクの実行順序はパラメータ空間から機械的に決定される。クラウド上の VM ではシナリオに記述されたタスクの順番で実行されるため、シナリオを生成する際に、タスクの順序を実行順序生成機構で制御してから出力する。WebInterface にて実行に必要な情報の入力を行ってから、クラウド上の VM へタスクの実行がされるまでのデータの流れを図 4 に示す。

利用者は Web インタフェースから実行したいモデルを選択し、パラメータ入力フォーム（param_input_form）を呼び出す。パラメータ入力フォームでは実行するモデルのパラメータ空間、繰り返し回数、乱数の初期値、比較関数の入力欄が表示されるので、各種情報を入力し、実行する（1）。入力されたデータは Python 言語の pickle モジュールによりシリアル化され Manager へと送られる（2）。pickle は Python 言語の標準的なシリアル化のモジュールである。pickle を用いてシリアル化をしたデータを通信することで、受信側はオブジェクトの構造を保ったままデータを送受信を行うことができる。Manager は WebInterface と ScenarioGenerator（実行順序生成機構）、TaskDispatcher の統括を行っており、WebInterface から送られてきたデータは実行順序生成機構へそのまま送られる（3）。実行順序生成機構では Manager を通して送られたデータを pickle モジュールにより逆シリアル化を行った後、入力データの比較関数、パラメータ空間などの入力データから実行者が望む順序にパラメータセットをソートし、その順序でパラメータセットの値を記したファイルを出力する（4）。次に（4）において保存した場所へのポインタ情報などを記したシナリオを生成し、Manager へ送信する（5）。シナリオは TaskDispatcher が VM を用いて実行する際に必要な情報が記された XML ファイルである。シナリオには実行者のユーザーID、モデル名、繰り返し回数、乱数の初期値、パラメータセットの値が記されたファイルが保存された場所へのポインタが記されている（5）。シナリオは Manager を通して TaskDispatcher へ送られる（6）。TaskDispatcher ではシナリオに記述されたポインタをもと

に、パラメータセットの値が記述されたファイルを読み込み、クラウド上の VM を用いて実行を行う（7）。

4.1.1 パラメータ入力フォーム

パラメータ名	型	最小値	最大値	刻み幅
N_RATE_GROWTH	REAL	0.1	0.1	0
G_RATE_DISCOUNT	REAL	1.0	1.0	0
G_ROUND_NUM	INTEGER	100000	100000	0
N_POPULATION_NUM	INTEGER	1000	1000	0
G_POINT_DD	REAL	1.0	1.0	0
G_POINT_DC	REAL	5.0	5.0	0
G_IDLING_TIME	INTEGER	0	0	0
MEM_BIT_L_INI	INTEGER	1	1	0
M_MUTATION_RATIO	REAL	0.0010	0.0010	0
N_LIMIT_FRACTION	REAL	0.0010	0.0010	0
P_RATE_NOISE	REAL	0.01	0.01	0
M_RATE_POINT_MUTATION	REAL	2.0E-5	2.0E-5	0
MEM_BIT_L_MIN	INTEGER	1	1	0
AAA_OBS_INTERVAL	INTEGER	10	10	0
M_RATE_GENE_DUPLICATION	REAL	1.0E-5	1.0E-5	0
M_RATE_SPLIT	REAL	1.0E-5	1.0E-5	0
MEM_BIT_L_MAX	INTEGER	5	5	0
G_POINT_CD	REAL	0.0	0.0	0
G_POINT_CC	REAL	3.0	3.0	0
G_SW_MEMORIZED_RESULTS	ENUM	☑ ON ☐ OFF		
SW_MUTATION	ENUM	☑ ON ☐ OFF		
SW_DUMP	ENUM	☑ ON ☐ OFF		
SW_ENDANGERED	ENUM	☑ ON ☐ OFF		
SW_CONSIDERING_INIACTS	ENUM	☐ ON ☑ OFF		
SW_RANDOM_FRACTION	ENUM	☐ ON ☑ OFF		

図 5: パラメータ入力フォーム

WebInterface は GPGCloud において、モデルの登録、実行、編集などを行う。設計はコンテンツ管理システムである Plone を用いて開発が行った。Plone は動的に Web ページを作成するためのツールである。Plone の特徴として、Web ページの動的生成や、スクリプトに Python 言語が使用可能である。そのため、自由度の高い、複雑な Web ページを動的に生成することができる。

利用者が WebInterface を用いてモデルの実行を行う場合は、パラメータ入力フォームから実行する範囲を指定する必要がある。パラメータ入力フォームはモデルの各種パラメータの実行範囲を指定するためのページである。その画面は図 5 である。パラメータ入力フォームはそれぞれのモデルのモデルパラメータファイルから動的に生成される。モデルパラメータファイルは GPGSiM において、モデルを開発、実行した際に作成される、パラメータ情報が記述された XML 形式のファイルである。パラメータ情報にはパラメータの名前、型、選択できる値の一覧などが含まれている。モデルの実行者が WebInteface から実行したいモデルを選択すると、モデルパラメータファイルを読み込み、パースを行い、必要なパラメータの入力フォームを含んだパラメータ入力フォームを表示させる。実行者はパラメータ入力フォーム上から、最大値、最小値、刻み幅の入力、真偽の設定などを行う。また、パラメータ入力フォームには比較関数入力フォームもあるので、優先的に検討を行いたい範囲がある実行者は比較関数入力フォームに記述することで、実行順序の制御を行うことができる。

利用者が指定したモデルのパラメータ入力フォームへの入力を行い実行をすると、入力されたデータはシリアル化が行われたのち、Manager を通して実行順序生成機構へと送られる。シリアル化には Python の標準モジュールである `pickle` を用いて行った。

4.1.2 比較関数の記述方式

```
def compare(param_a、param_b):
     if param_a[0] > param_b[0] :
          return 1
     if param_a[0] < param_b[0] :
          return -1
     elif:
          return 0
```

図 6: 第 1 項目を降順とする比較関数

実行順序生成機構で用いる比較関数は２つのパラメータセットを引数として取った場合に、1,0,−1を返すことで優先順序付けを行う。図 6 は降順の比較関数の例である。`compare(param_a, param_b)` が比較関数となっており、パラメータセット `param_a`、`param_b` を引数にとり、それぞれの第 1 パラメータを比較し、大きいほうを優先して実行、もし数値が同じならばどちらでもよいということを表わしている。返り値は1の場合は第 1 引数を優先的に実行、−1の場合は第２引数を優先的に実行、0の場合はどちらでもよいということを示している。図 6 の例ではパラメータセット `param_a` と `param_b` で、それぞれのパラメータの第 1 項目を比較し、第 1 項目が大きいほうのパラメータセットを優先的に実行することを表わしている。`param_a[0]`と `param_b[0]`の値が同じ場合は0を返し、どちらを先に実行してもよいことを表わしている。

比較関数の記述方式は Python による記述か、予め用意された指定の命令を用いることで指定する。記述方式による制限は次の通りである。

(1)関数の宣言方法は固定
(2)返り値としては1,0,−1のみを使用
(3)import 文は使用不可

比較関数として予め用意されている命令には昇順、降順といった単調なものが用意されている。図 6 の比較関数は降順を表わしているので、比較関数入力フォームに図 6 を入力しなくとも、`dec(0)` を入力するだけで同じ実行順序の指定を行うことができる。比較関数の記述において、Python 言語として構文が間違っている命令が記述された場合、実行順序の制御は行われず、実行結果の蓄積も機械的順序で行われる。

4.1.3 実行順序生成機構

実行順序生成機構の役割は、パラメータ入力フォームにおいて、入力されたパラメータ空間からパラメータセット一覧を生成し、パラメータセットをノード比較関数を隣接辺の判定基準としてパラメータセットをトポロジカルソートし、ソートした順序でファイルを出力した後に、TaskDispatcher が VM にパラメータセットを渡す際に、どの順序でパラメータセットを渡すのかを記したシナリオの作成である。

(1) 受け取ったデータの逆シリアル化とノードの作成

Manager を通して送られるデータは `pickle` によりシリアル化されているため、逆シリアル化を行う。逆シリアル化を行うと、そのデータの構造は次のようになっている。

[繰り返し回数, 乱数の初期値, [[パラメータ情報(名、最小値、最大値、刻み幅、…)],...],"比較関数"]

まずはパラメータ空間内の各パラメータセットのノードを作成する必要があるため、パラメータ情報の最小値、最大値、刻み幅などからノードを作成する。

(2) トポロジカルソートを用いたソート

次に(1)で作成したノードに対してトポロジカルソートによりソートを行う。トポロジカルソートは半順序関係である無閉路有向グラフに対するソートであるため、作成したノードと比較関数による関係を用いてソートを行う。

トポロジカルソートにおける隣接辺の判定に比較関数を用いることで、比較関数を変えることで制御基準を変化させ、その順序にトポロジカルソートが可能となる。深さ優先探索を用いたトポロジカルソートでは、基準としたノードから逆向き辺へ探索を進めていく。そのため、比較関数の出力結果が−1となるほうへ探索を進めていくアルゴリズムである。そのため、トポロジカルソートのアルゴリズムにおける隣接辺の判定の箇所を compare(S[n],S[i])=-1 とする。訪問済みリストを共有し、全てのノードに対し、まだ訪問済みでない場合は、逆向き辺が存在する場合はその方向へ探索を進め、逆向き辺が存在しなくなるか、既に訪問済みのノードへの辺しか存在しなくなるまで探索を行っていくことにより、全てのノードに対して実行者が指定した比較基準でソートできるアルゴリズムになっている。

実行順序生成機構ではソートが完了した後に、各パラメータセットの値が記述されたファイルを 1 パラメータセット 1 ファイルで出力を行う。これは後述するシナリオに保存場所が記述され、TaskDispatcher はシナリオからパラメータセットの値が記述されたファイルの位置を読み込むことにより、パラメータセットの値を知るようになっている。

(3) シナリオの作成

(2)においてパラメータ空間の各ノードのソートが終わった後に、TaskDispatcher で用いるシナリオを作成する。シナリオはパラメータセットの値が記述されたファイルへのポインタ、繰り返し回数、乱数の初期値、モデル情報などが含まれた XML ファイルである。シナリオには実行者のユーザーID、モデル名、繰り返し回数、乱数の初期値、パラメータセットの値が記されたファイルが保存された場所へのポインタが記されている。

4.2 実行例

4.2.1 囚人のジレンマの適用による実行順序制御

以下では、実行順序生成機構の動作例としてエラー付き指導者ゲーム[11]のモデルを用いる。エラー付き指導者ゲームはプレイヤーが利得行列と呼ばれる自身の利得が表現された行列に従って行動することで、プレイヤーの行動が相互にどのように影響し合い、プレイヤーがどのように行動するかを観測する戦略ゲームである。

エラー付き指導者ゲームのモデルにおいて、様々な利得設定にて実行を行う際に、囚人のジレンマの条件にマッチしたパラメータセットから優先的に確認を行うとして比較関数を設定し、実行を行った。

4.2.2 エラー付き指導者ゲーム

表 1 はエラー付き指導者モデルのパラメータの一部である。今回の実験では指導者ゲームにおいて様々な利得を設定してどのような挙動を取るかを検証する際に、特に囚人のジレンマモデルを表わすパラメータセットから優先的に実行を行いたいという設定で実行を行った。そのため、指導者ゲームのモデルにおいて、利得行列の条件を囚人のジレンマゲームの利得条件に設定を合わせた。

表 1: エラー付指導者ゲームの主要パラメータ

パラメータ名	パラメータの内容
GPOINT_CC(&CD&DD&CC)	利得行列の値
MEM_BIT_L_INI(MIN,MAX)	記憶長の初期値（最小、最大）
M_MUTATION_RATIO	突然変異率
N_POPULATION_NUM	個体数
P_RATE_NOISE	ノイズの発生率

囚人のジレンマゲームは二人のプレイヤーが個人の利得を優先して行動した結果、二人のプレイヤーの総合利得が最低となってしまう事象である。囚人のジレンマゲームにおいては、自身が積極、相手が消極的行動を取った際に自身の利得が最大になり、その逆の場合に自身の利得が最低となる。そのため、利得は式(1)を満たす必要がある。

$$DC > CC > DD > CD \quad (1)$$

D は積極、C は消極を表わしている。利得における DC は自身が積極的行動をとり、相手が消極的行動をとった場合の自身の利得を表わしている。

そこで、比較関数において、式(1)を満たす条件を指定した。図 7 は式（1）の条件を満たした、囚人のジレンマ状況を優先する比較関数である。

```
#[4]  :G_POINT_DD
#[5]  :G_POINT_DC
#[17]:G_POINT_CD
#[18]:G_POINT_CC

def compare(a,b):
    if (a[5] > a[18] > a[4] > a[17]):
        if (b[5] > b[18] > b[4] > b[17]):
            return 0
        else:
            return 1
    elif not (a[5] > a[18] > a[4] > a[17]):
        if (b[5] > b[18] > b[4] > b[17]):
            return -1
        else:
            return 0
```

図 7: 囚人のジレンマ状況を優先する比較関数

パラメータセットにおける第4,5,17,18項目がそれぞれ利得行列の DD，DC、CD、CC である。そのため、式(1)を満たすための条件としてパラメータセットの値のうち、項目 $5 > 18 > 4 > 17$の順に利得が大きくなる条件を満たすパラメータセットを優先的に実行する指定を行う。図 7 では、パラメータセットを表わすリストを引数にとり、パラメータセット内の利得を表わすパラメータ値が含まれる項目同士を比較し、どちらかのみが式(1)を満たす場合にはそちらを優先的に実行、そうでない場合はどちらでもよいという指定の比較関数である。パラメータ空間の大きさは各利得の最小を0、最大を5、刻み幅を1として実行を行った。実際に実行順序を制御せずに実行した場合と、実行順序を制御して実行した場合の結果はそれぞれ表 2, 3 となった。

表 2: 実行順序制御を行わない場合の実行順序

タスク No	DC	CC	DD	CD
1	0	0	0	0
2	0	0	1	0
3	0	0	2	0
4	0	0	3	0
5	0	0	4	0
6	0	0	5	0
7	1	0	0	0

表 3: 実行順序制御を行った場合の実行順序

タスク No	DC	CC	DD	CD
1	3	2	1	0
2	4	2	1	0
3	5	2	1	0
4	4	3	1	0
5	4	3	2	0
6	5	3	1	0
7	5	3	2	0

表 2 は実行順序を制御せずに実行した場合の結果であり、利得の大きさがパラメータ空間から機械的に選択され実行されるため、囚人のジレンマゲームとして成立していないパラメータセットから実行されてしまうことがわかる。表 3 は図 7 の比較関数を用いて実行順序制御を行い実行した結果で、各利得の大小関係が式（1）を満たし、囚人のジレンマゲームとして成立していることが確認できる。

5. 考察

本研究で実装した実行順序生成機構を用いることで、研究者は膨大なパラメータ空間から優先的に実行するパラメータセットを選択することができるようになった。これにより、現状分析機能が不十分な GPGCloud において、研究効率をあげることに一定の成果を出すことができた。

しかし、GPGCloud にはいまだに問題点が数多く残されている。GPGCloud には並行実行機能の提供のほかにも、モデル開発環境 GPGSiM における課題点であった再利用性の向上も目的としている。

今回実行順序生成機構を作成する際に、変更を加えた WebInterface では、パラメータ入力フォームにおける初期入力値をモデルパラメータファイルから取得することができる。そのため、モデルの登録者が GPGSiM において、どのようなシミュレーションを行ったかの再現は可能となっている。しかし、GPGCloud における並行実行の再現については機能として実現ができていない。そのため、GPGCloud において実行したパラメータ空間においても再現可能であることが必要だと考えられる。また、パラメータ入力フォームはモデルパラメータファイルに含まれる情報から表示を行う。そのため、パラメータ入力フォームでは、パラメータの名前、型といった情報しか表示することができない。そのため、モデルの利用者がパラメータの意味する内容を理解しづらいといった問題点もある。再利用性向上のためにはこれらの問題点の解決が必要である。

実行順序の制御方式については先行研究として山田の動的制御によるシミュレーション最適化がある[12]。動的制御では、予め実行者が制御基準を与える必要はない。実行するパラメータを決定変数として目的関数に与え、その結果が最大、もしくは最小となるよう、実行するパラメータセットを制御していくという手法である。今回実装した実行順序生成機構ではモデルの実行者が制御基準を決定し、静的制御による実行順序制御により、研究効率の向上を図った。制御基準を決定することにより、実行順序生成機構は研究効率を向上させることができる。しかし、シミュレーションの結果から得られる数値から最適化を行う場合など、予め制御基準を指定することができない場合には動的制御が必要となる。しかし、現在の実行順序生成機構においては、静的制御によるシミュレーション最適化しか行うことができない。そこで、静的制御だけでなく、動的制御によるシミュレーション最適化も組み合わせることで、様々な制御基準に対応することができると考えられる。

参考文献

[1] 齋藤宗香，山口裕，八槇博史，秋山英三，瀬島誠，吉田和男，「国際政治・経済研究のためのシミュレーション環境 GPGSiM の開発」,エージェント合同シンポジウム(JAWS2008), 2008.

[2] Muneyoshi Saito, Yutaka Yamaguchi, Hirofumi Yamaki, Eizo Akiyama, Makoto Sejima, and Kazuo Yoshida. “GPGSiM:a new simulation environment for international politics and economics," *Summe Computer Simulation Conference 2009*, pp.283－290, 2009.

[3] 齋藤宗香, 山口裕, 「マルチエージェントシミュレーション環境 GPGSiM」, 吉田和男, 井堀利宏, 瀬島誠編著『地球秩序のシミュレーション分析』第 6 章, 日本評論社, pp.95－112, 2009.

[4] 八槇博史, 「政治・経済モデルの再利用性向上に向けた考察」, 吉田和男,井堀利宏, 瀬島誠編著『地球秩序のシミュレーション分析』第 5 章, 日本評論社, pp.75－94,2009.

[5] Yoshiki Kato, Hirofumi Yamaki, and Yuki Asai, “GPGCloud : Model sharing and execution," *PRIMA 2009*, pp.616－623, 2009.

[6] Nelson Minar, Roger Burkhart, Chris Langton, and Manor Askenazi, “The swarm simulation system: A toolkit for building multi-agent simulations," *In Working Paper 96-06-042*,Santa Fe Institute, 1996.

[7] 田沼英樹, 出口弘, 「エージェントベース社会シミュレーション言語 SOARS の開発」, 電子情報通信学会論文誌, No.9, pp.2415－2422, 2007.

[8] 山影進, 『人工社会構築指南－ artisoc によるマルチエージェント・シミュレーション入門』,書籍工房早山, 2007.

[9] 浅井勇貴, 八槇博史, 「エージェントシミュレーション支援環境 GPGCloud における並行実行機能の設計」, 平成 22 年電気関係学会東海連合大会, 2010.

[10] M. P. Jarnagin, “Automatic machine methods of testing PERT networks for consistency,"*Technical Memorandum*, No. K-24/60, 1960.

[11] 秋山英三, 吉田和男, 「エラー付き指導者ゲームにおける戦略の進化. Joint Agent Workshops and Symposium 2008, 2008.

[12] 山田周平, 「マルチエージェントシミュレーションのためのパラメータ探索支援機能の開発」, 名古屋大学工学部情報工学コース卒業研究報告, 2009.