PG Lab ディスカッションペーパーシリーズ
第8号　2006年10月

ISSN 1880-9146

Discussion Paper Series
No.8　October, 2006

タスク実行支援フレームワークに基づくグリッド環境への託児所配置問題の適用

蟻川　浩，村田 忠彦

文部科学省私立大学社会連携研究推進拠点
関西大学政策グリッドコンピューティング実験センター

Policy Grid Computing Laboratory,
Kansai University
Suita, Osaka 564-8680 Japan
URL : http://www.pglab.kansai-u.ac.jp/
e-mail : pglab@jm.kansai-u.ac.jp
tel. 06-6368-1228
fax. 06-6330-3304

関西大学政策グリッドコンピューティング実験センターからのお願い

本ディスカッションペーパーシリーズを転載，引用，参照されたい場合には、
ご面倒ですが、弊センター（pglab@jm.kansai-u.ac.jp）宛に
ご連絡いただきますようお願い申しあげます。

Attention from Policy Grid Computing Laboratory, Kansai University

Please reprint, cite or quote WITH consulting Kansai University Policy Grid Computing Laboratory (pglab@jm.kansai-u.ac.jp).

タスク実行支援フレームワークに基づくグリッド環境への託児所配置問題の適用

蟻川 浩[†] , 村田 忠彦[† ‡]

A Implementation Issues for the Grid-based Day Care Center Allocation Problem Solving Environment using the Task Control Support Framework

Hiroshi ARIKAWA[†] ,Tadahiko MURATA[† ‡]

概要

近年、グリッド環境におけるマルチエージェントシミュレーションが社会学、経済学、政治学などの社会科学の研究者に注目されているが、グリッド環境によるシミュレーションの実現方法については理解されていない。

我々は大規模計算向けマルチエージェントシミュレーションツールの開発を行っている。本稿では、タスク実行支援フレームワークに基づくグリッド環境による託児所配置問題解決環境の実装について述べる。本稿で提案する託児所配置問題解決環境により、最適解の探索にかかる時間を改善できることを示す。

Abstract

Social Scientists are being watched to do computation for Grid-based multi agent simulation in recent years. However, there are many points which are uncertain at present.

We have been developping multi agent simulation tools for the Grid that enable a large-scale computing. This paper describes a implementation issue for Grid-based day care center allocation problem solving environment using the task control support framework. By using proposed environment, social scientists could search the optimum result quickly.

キーワード: グリッド, 高スループット計算, タスク実行支援, 託児所配置問題, 継続的改善

Keywords: Grid, High Throughput Computing, Task Control Suport, Day Care Center Allocation Problem, Continual Improvement

[†] 関西大学 政策グリッドコンピューティング実験センター
[‡] 関西大学 総合情報学部
[†] Policy Grid Computing Laboratory, Kansai University
[‡] Faculty of Informatics, Kansai University

1. はじめに

マルチエージェントシミュレーションによる託児所配置問題では、適切な託児所の配置条件を導き出すために、パラメータスウィープによるアルゴリズムを採用している。パラメータスウィープによるアルゴリズムでは、想定される範囲でのパラメータリストを作成し、すべてのパラメータリストを実行する。このとき、パラメータの種類、パラメータに与える値の数が増えると、パラメータの組み合わせ数が指数的に増加する。

すべてのパラメータによるタスク実行が完了するまでに非常に長い時間を要するため、シミュレーション実施者の意思決定に基づいてシミュレーションを実施することが必要である。ところが、既存技術によるグリッド環境の構築方法では、以下に示す問題により、シミュレーション実施者の意思決定に基づくシミュレーションが実現できない。

1. フロントエンドではシミュレーションの進捗状況を確認する機能とパラメータ変更に対してシミュレーション結果を再利用できる機能が提供されていないこと。

2. バックエンドでは複数のタスク管理ミドルウェアが動作しており、タスク管理の一元化ができていないこと。

3. 各タスク管理ミドルウェアはタスク実行をするのみである。どのパラメータを実行したかという管理は、シミュレーション実施者が行わなければならない。

結果として、従来はシミュレーション実施者の知識と経験と関係なく、最適解の探索のために無駄と考えられるようなタスクの実行も行われている。

本稿では、シミュレーション実施者の意思決定によって指数的に増加するパラメータの組み合わせによることなく最適解を探索する方法として、対話的動作に基づいた託児所配置問題を解くためのグリッド環境の構築について提案するとともに、適用に際しての課題と解決策について述べる。

2. マルチエージェントシミュレーションと託児所配置問題

2.1 マルチエージェントシミュレーション

マルチエージェントシミュレーションは、従来人工知能分野で研究が行われてきた人間の知能モデルをシミュレーションに展開した研究分野である。コンピュータならびにネットワーク技術の発展を背景として、1990 年以降急速に発展している。一般的には、エージェントが相互影響を受けることで対象となる場の振舞いを定量的に示す方法と考えることができる。ここで指すエージェントとは、知覚、認識、記憶、学習、判断、行動といった人間がもっている情報処理機能をモデル化したものである。

マルチエージェントシミュレーションの適用範囲は広い。特に、社会科学分野では、エージェントの振舞いを人間の振舞いとして置き換えることにより、社会における発見や定式化を支援することができると考えられている。マルチエージェントシミュレーションの社会科学の応用として、文献 [1] では、サッカーシミュレーション、交通流・人流シミュレーション、金融市場シミュレーション、災害・防災シミュレーションの例を挙げている。また、既婚女性の労働供給に関する適用例 [2] や、国民年金納付問題への適用例 [3] などがある。

社会科学分野においてマルチエージェントシミュレーションが注目されている理由は、社会現象が実験により検証することが困難なことにある。自然科学分野では現象を実験室などで再現できるが、社会科学分野では人間の思考、動作、決断を対象としているため、実験室で現象を再現することができない。そのため、マルチエージェントシミュレーションに期待が寄せられている。

2.2 託児所配置問題

日本が直面している課題のひとつに少子化問題がある。厚生労働省が発表した平成 17 年 人口動態統計の年間推計[1] において、平成 16 年の合計特殊出生率[2] は 1.29 である。この数値は将来における日本の総人口減少を意味している。労働力の低下、経済活動の停滞、高齢化社会への変化をもたらし、様々な社会問題を引き起こす。

厚生労働省が発行した平成 17 年度版 労働経済白書 [4] において、女性の意識と就業促進に向けた課題がまとめられている。その中で、女性の就業行動は出産および育児によって大きく影響を与えていること、すなわち、就業と育児の両立が困難であることが示されている。また、財務省財務総合政策研究所が発行した「少子化の要因と少子化社会に関する研究会」報告書 [5] において、就業による出産抑制効果は育児休暇制度や保育サービスの充実により緩和されることが提言されている。これらの報告から、少子化対策のためには女性の就業行動を促進すること、具体的には、保育サービスをどう充実させるかが解決の糸口であるといえる。そこで考えられるのが、託児所設置による非就業者の就職行動促進に関する検討である。本稿では、託児所配置問題と呼ぶ。

託児所配置問題は、託児所の配置によって子供をもつ女性が就業にどのような影響をもたらすかを知ることである。具体的には、職場との位置関係によって女性が就業するか、性格の異なる住民間の相互作用によって就業するかどうかを検討することである。この問題を解決することは、地方公共団体においては住民サービスの提供につながる。また、日本政府においては少子化の解消とともに、税収向上も期待できる。

2.3 マルチエージェントシミュレーションに基づく託児所配置問題

託児所配置問題は、職場との位置関係によって女性が就業するか、性格の異なる住民間の相互作用によって就業するかどうかを検討することであると述べた。すなわち、ある地域の女性は、会社、託児所、近隣の女性からの影響を受ける。これらの相互影響により就業するかを判断する問題と置き換えることができる。このように、何らかの相互影響を受ける問題と解く場合には、マルチエージェントシミュレーションが得意とする。本稿では、マルチエージェントシミュレーションによる託児所配置問題として、村田らが提案する方法 [6] を取り上げる。

本稿で取り上げる託児所配置問題は、図 1 に示すように、非トーラス状の 2 次元空間内に職場、託児所、エージェントが存在する。

託児所配置問題においてエージェントは女性とする。各エージェントは就業行動を行うか否かを決定する。エージェントが就業するかを判断する基準は、効用差関数、エージェントの知

[1] http://www.mhlw.go.jp/toukei/saikin/hw/jinkou/suikei05/index.html

[2] その年次の 15 歳から 49 歳までの女子の年齢別出生率を合計したもの。一人の女子が仮にその年次の年齢別出生率で一生の間に生むとしたときの子どもの数に相当する。

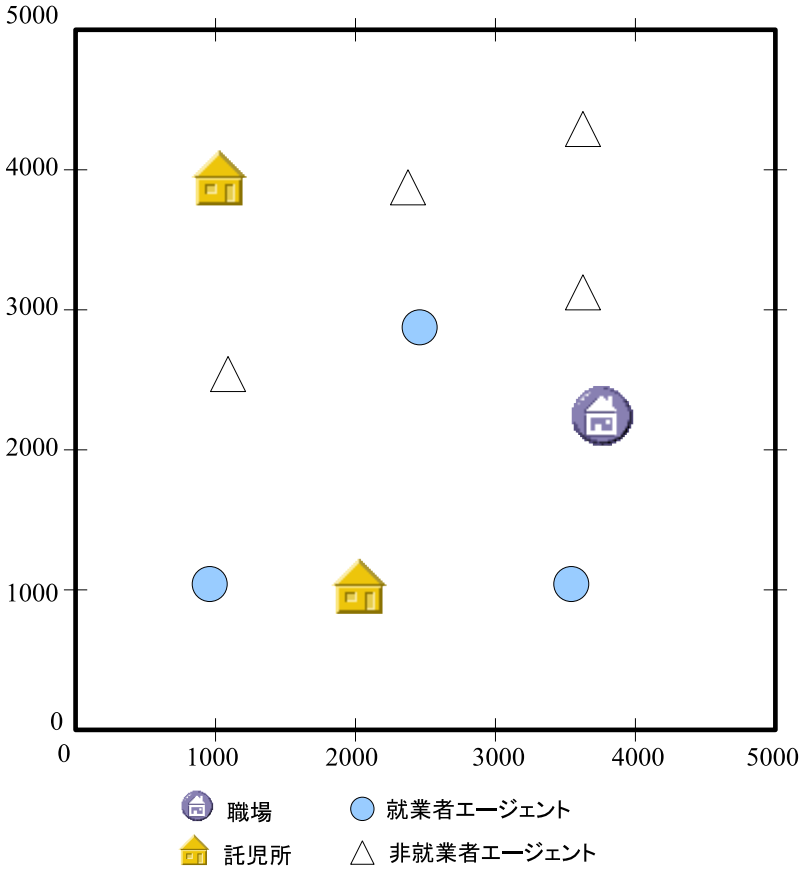

図 1　託児所とエージェントの位置関係

覚範囲、周囲のエージェントとの相互影響である。効用差関数とは、年齢、教育年数、配偶者の収入、子供の数を変数とした就業時の効用と非就業時の効用の差を示す関数である。エージェントの知覚範囲とは、エージェントが就業するか否かを判断する際に情報を収集する範囲である。知覚範囲に託児所がある場合は就業行動をとりやすいと仮定する。周囲のエージェントとの相互影響は、知覚範囲内に存在する就業中のエージェントが多いほど就業選択を行うと仮定する。本稿で取り扱う問題におけるエージェントは、周囲のエージェントが就業していなくても就業行動を率先して行うリーダエージェントと、周囲の行動に左右されるフォロワエージェントの 2 種類が存在するものとする。

2.4　託児所配置問題における対話的動作の必要性

マルチエージェントシミュレーションによる託児所配置問題は 2 つの特徴を有する。

- マルチエージェントシミュレーションは確率論的手法である。
  入力データの一部にある確率分布もしくは乱数を使うため、統計的に十分な数の計算を繰り返す。

- パラメータの個数が増えるとパラメータの組み合わせ数が指数的に増大する。
  託児所配置問題は託児所の適切な配置を検討することが目的である。託児所の適切な配

置をみつけるために、例えば、託児所の位置、職場の位置、エージェントの初期条件を変更する。その結果、シミュレーションを実行する数が指数的に増える。

村田らの実験では、各実験条件において場合分けをできるだけ少なくなるように、職場の位置、エージェントの人数比 (リーダとフォロアの比率)、託児所の位置、エージェントが託児所を選択するうえで許容できる距離がそれぞれ 4,9,7,3 通りとし、かつエージェントの分布中心は職場の位置が中央にある場合は 12 通り、それ以外は 8 通りとしている。これらの値から算出されるシミュレーションの実行数は 6804 通りになる。なお、6804 通りのシミュレーションを Pentium4 2.8GHz の計算機で 1 台で実行した実験を行っており、その結果、317 時間 54 分 46 秒かかったと報告している。

より詳細なシミュレーションを行う場合、例えば、職場の位置、託児所の位置を増やした場合の就業選択行動に関する効果を確認する場合、シミュレーションの実行数の増加が容易に想像できる。考えられる限りでのパラメータの組み合わせを作成し、パラメータの組み合わせすべてを総当たりで計算する従来の考え方では、パラメータの組み合わせ数が増えれば、シミュレーションに要する時間も増加する。

ここで、託児所配置問題に取り組む政策立案者の立場を考える。政策立案者はシミュレーションから得られる結果に政策的な意味をもつため、具体的な条件を設定したシミュレーションを希望する。また、最適な託児所配置の条件を知るために、特定のパラメータを与えてはタスク実行をするといった動作が行われる。

2.5 託児所配置問題解決システム構築における対話的動作実現の難しさ

託児所配置問題では、パラメータの組み合わせによって作成される解集合から評価値が最大となるパラメータの組み合わせを探索することが要求される。既存の技術を用いて託児所配置問題解決システムを構築する場合、以下の方法が考えられる。

- 想定されるパラメータの組み合わせすべてを試す方法 (自動的に実行する場合)

  長所 タスク実行が自動化可能であること。
  短所 解集合の数によって問題解決にかかる時間が増大すること。

- パラメータの組み合わせを一つずつ試す方法 (対話的に実行する場合)

  長所 シミュレーション実施者の結果を確認しながら解探索ができること。
  短所 問題解決にかかる時間が増大すること。シミュレーション実施回数が大幅に増えること。

シミュレーション実施者は、パラメータ群の集合からサンプルとして一部の解候補を取り出して実行し、その結果を確認したのち再度別の解候補を実行する、という繰り返し操作が望ましい。したがって、これらの中間的な方式がシミュレーション実施者にとって望ましいといえる。

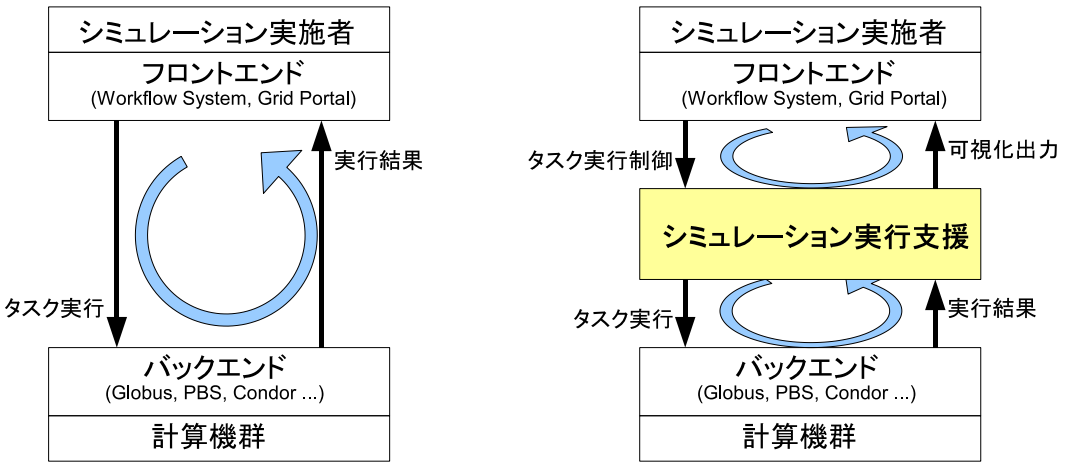

図 2 グリッド技術を用いた問題解決システムとシミュレーション実施者との関係 (左: 既存技術によるモデル, 右: 実施者の要求モデル)

3. タスク実行支援フレームワークに基づいた託児所配置問題解決システム

マルチエージェントシミュレーションによる託児所配置問題に求められる目標は、大量の解集合から託児所配置で最大の効果をえられる解を探索することにある。このことを踏まえ、タスク実行支援フレームワークに基づいたグリッド環境による託児所配置問題解決システムを構築する。

3.1 タスク実行支援フレームワーク

問題解決環境の構築における従来の考え方では、図 2 左側に示すように、グリッド基盤側はバックエンドが、シミュレーション実施者の操作はフロントエンドが担当する。このモデルでは、シミュレーション途中でのパラメータ変更と実行中のタスクへの反映において、実行中タスクの停止、パラメータ変更作業、タスクの再実行作業といった作業をシミュレーション実施者が操作しなければならない。シミュレーション実施者は大量タスクを含むシミュレーションの繰り返しにおいて操作上の負担を強いられる。

大量タスク実行のように、タスク間で通信が殆んど発生しないような場合は大量の計算機を利用することが望ましく、かつ試行錯誤に基づいた大量タスク実行によるシミュレーションを実施する場合には、シミュレーション実施者から見てタスクひとつひとつのパラメータの変更が容易でなければならない。ところが、このモデルでは特定のグリッドミドルウェアに依存した形で問題解決システムを構築することが前提である。

そこで、図 2 右側に示すようなモデルを実現するために、文献 [7] においてタスク実行支援フレームワークを提案した。タスク実行支援フレームワークの詳細については文献 [7] を参照していただくこととし、ここでは、タスク実行支援フレームワークの概要について述べる。

タスク実行支援フレームワークは、グリッドミドルウェアで構成されたグリッド環境を 1 つのモジュールと考える。図 3 に示すように、グリッド環境側を下位、シミュレーション実施者

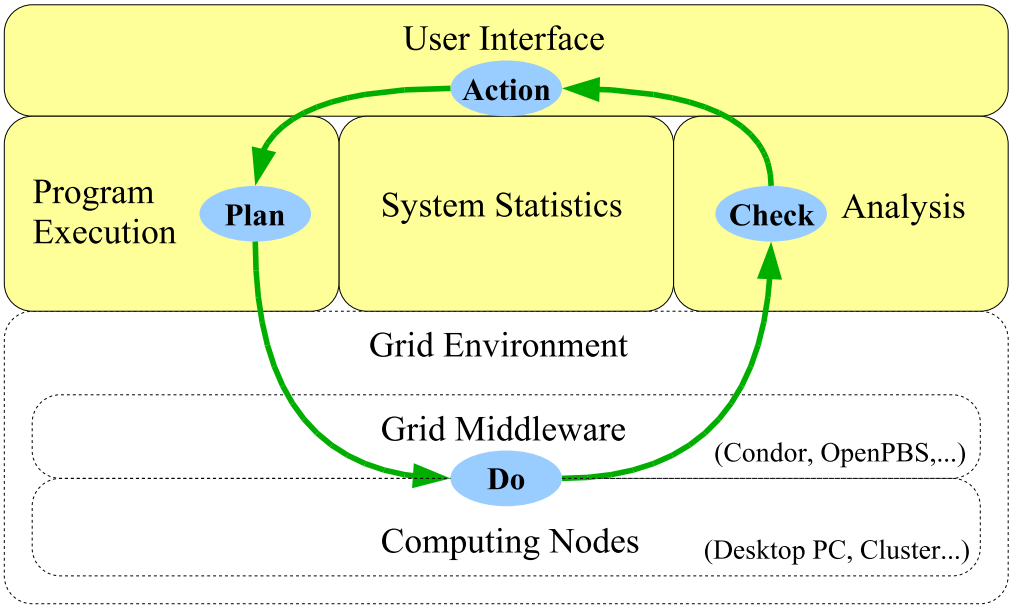

図 3　タスク実行支援フレームワークの全体像

側を上位とし、グリッド環境より上位に対して、プログラム実行モジュール、計算結果解析モジュール、システム状況監視モジュール、ユーザインターフェイスの 4 つのモジュールで構成する。

これらのモジュールは ISO9000 において継続的改善を実現するための具体的なモデルである PDCA サイクルに基づいて配置している。各モジュールの特徴について説明する。

- **ユーザインターフェイス**
  ユーザインターフェイスはシミュレーション実施者とグリッド環境との架け橋の役目をするモジュールである。PDCA サイクルにおける Action に相当する。ユーザインターフェイスでは、シミュレーション実施者の動作要素として、プログラムの選定、入力データの準備、入力パラメータの設定、プログラム実行順序の決定、結果確認、処理状況確認、を担当する。

- **プログラム実行モジュール**
  プログラム実行モジュールはシミュレーション実施者の操作に基づいてプログラムの実行および停止を行うモジュールである。PDCA サイクルにおける Plan に相当する。プログラム実行モジュールでは、シミュレーション実施者の動作要素として、シミュレーション実施者におけるプログラム実行順序の決定およびプログラム実行を担当する。

- **計算結果解析モジュール**
  計算結果解析モジュールはプログラム実行結果として出力されるデータを回収し、シミュレーション実施者が意思決定をするために必要な形式へとデータを変換するモジュールである。PDCA サイクルにおける Check に相当する。計算結果解析モジュールでは、シミュレーション実施者の動作要素として、計算結果の回収および可視化処理を担当する。

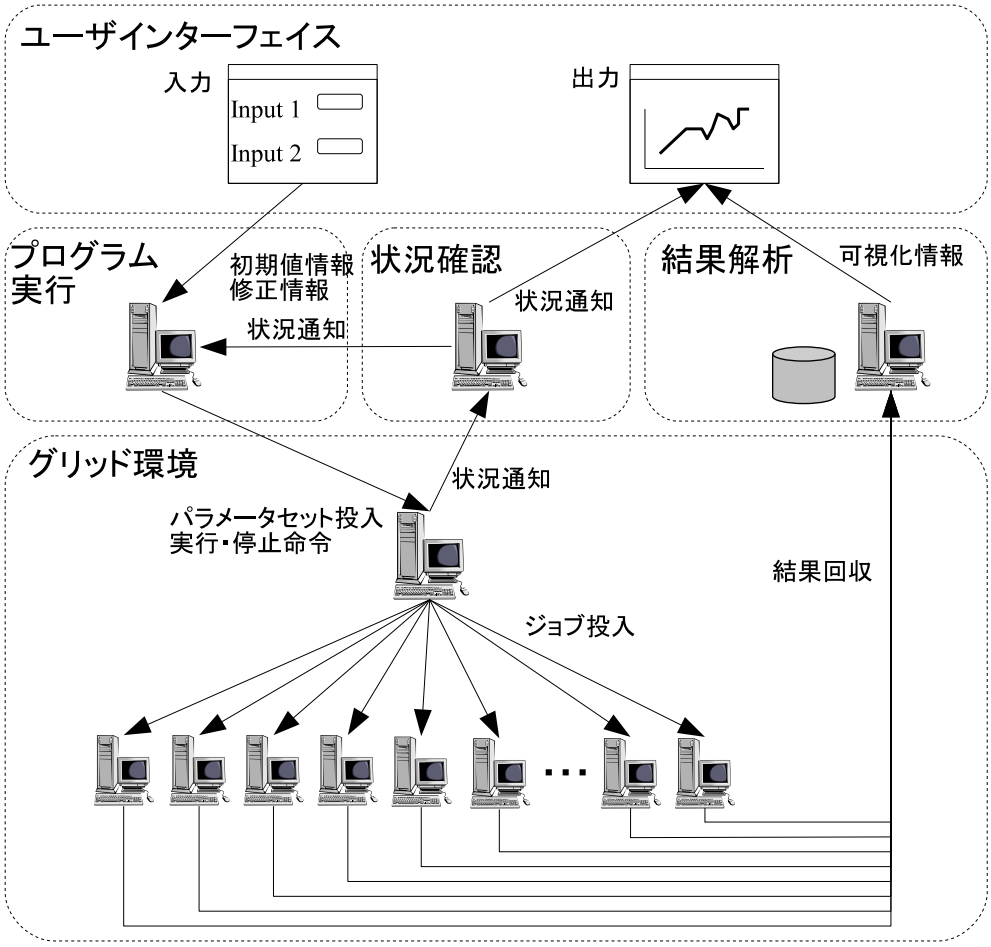

図 4 タスク実行支援フレームワークによるシステム構成

- **システム状況監視モジュール**
  システム状況監視モジュールでは、グリッド環境の負荷情報およびプログラム実行状況を把握するモジュールである。PDCA サイクルと直接の関わりはないが、シミュレーション実施者の動作や他のモジュールとの連携のために設けた。

タスク実行支援フレームワークに基づいて構築した問題解決環境は図 4 のように構成される。タスク実行支援フレームワークはフロントエンドとバックエンドとの間にタスク実行を支援する層を導入する形となり (図 2 右側)、問題解決システム側でタスク管理の一元化を実現するとともに、タスクと計算結果の関連付けを容易にする。この結果、以下に示すメリットが得られる。

- **バックエンド側のメリット**
  バックエンド側については、複数のタスク管理ミドルウェアの操作上の違いを吸収することが可能である。タスク実行について例えば、Globus Toolkit であれば globus-job-run だが、Condor では condor_submit が使われる。統一的なタスク実行コマンドに置き換えることで、シミュレーション実施者は 1 つの実行コマンドで操作することができるようになる。

- **フロントエンド側のメリット**
  フロントエンド側については、タスク管理の統一化が可能になる。従来、タスク実行と計算結果の関連付け、およびタスク実行とパラメータの関連付けはシミュレーション実施者が手動で行わなければならなかった。支援層を導入することでタスク実行と計算結果、タスク実行とパラメータの関係づけが容易になる。また、シミュレーション実施者の操作がタスク情報の修正だけになる。

3.2 タスク実行支援フレームワークの託児所配置問題への適用

本節では、マルチエージェントシミュレーションによる託児所配置問題をタスク実行支援フレームワークを用いたグリッド環境を構築するための課題と解決策について示す。

3.2.1 パラメータリストファイルの準備と大量タスク実行スクリプトの実装

託児所配置問題解決システムは、各パラメータの値の組み合わせによって作成される大量のパラメータの組から託児所配置で最大の効果を得る各パラメータの値を探す。いわゆるパラメータスウィープ型のシミュレーションである。

一般的に、パラメータスウィープ型のシミュレーションはタスク実行の際に用いられるプログラムが 1 種類である。そのため、シミュレーション実施者はパラメータリストを作成し、それに基づいてタスク実行を行うことでパラメータごとのタスク実行の負担を削減することができる。

本稿で取り上げる託児所配置問題では、図 5 に示すようなフォーマットでパラメータリストファイルを準備しているため、これらのパラメータをすべて実行するスクリプトとして、大量タスク実行スクリプトを実装した (付録参照)。

大量タスク実行スクリプトは、以下に示すように、大量タスク実行をするためのプログラム名とパラメータリストを引数する。

```
submitter.pl <プログラム> <パラメータリストファイル>
```

大量タスク実行スクリプトによって出力される例を図 6 に示す。図 6 はタスク管理ミドルウェアが Condor の場合である。Condor の場合、タスク管理サーバと計算ノードとの双方で必要なデータを転送できる機能をもっているため、計算結果を回収する枠組みを設ける必要はない。その代わり、`should_transfer_files`、`when_to_transfer_output`、`transfer_input_files`、`transfer_output_files` の 4 種類の設定語を使って、双方で転送するデータを記述する (図 6 の例では、各タスクの実行が完了すると実行結果が戻ってくる設定になっている)。

3.2.2 タスク優先実行制御機能によるパラメータの優先実行

パラメータリストファイルに基づいた大量タスク実行の場合、記述されている順に実行される。タスクの実行順序が固定化されてしまうため、シミュレーション結果の出力順はリストの記述順に影響を受ける。ゆえに大量タスク実行においては次のような問題が生じる。

```
0 1000 1000 1000 -1 -1 2 2 1000 1000 1000 4000 4000 4000 100 0.7 →
output_0-1000_1000_1000_-1_-1_2_2_1000_1000_1000_4000_4000_4000_100_0.7.dat
0 1000 1000 1000 -1 -1 2 2 1000 1000 1000 4000 4000 1000 100 0.7 →
output_0-1000_1000_1000_-1_-1_2_2_1000_1000_1000_4000_4000_1000_100_0.7.dat
0 1000 1000 1000 -1 -1 2 2 1500 4000 1000 4000 4000 1000 100 0.7 →
output_0-1000_1000_1000_-1_-1_2_2_1500_4000_1000_4000_4000_1000_100_0.7.dat
0 1000 1000 1000 -1 -1 2 2 2500 2500 1000 4000 4000 1000 100 0.7 →
output_0-1000_1000_1000_-1_-1_2_2_2500_2500_1000_4000_4000_1000_100_0.7.dat
```

(以下続く。紙面の都合上、→は行の続きを意味する。)

図 5 パラメータリスト例

```
UNIVERSE = vanilla
Executable = calculateVersion4
Log=calcv4-62.log
Output = calcv4-62.out
Error = calcv4-62.err

Arguments = 250 750 -1 -1 2000 2500 2 2 4000 4000 1000 4000 4000 \\
1000   100  0.7  output_250-750_-1_-1_2000_2500_2_2_4000_4000_1000_\\
4000_4000_1000_100_0.7.dat

should_transfer_files = YES
when_to_transfer_output = ON_EXIT

transfer_input_files = AGE.txt, C18.txt, ED.txt, HI.txt, parameter.txt, \\
WORK.txt
transfer_output_files = output_250-750_-1_-1_2000_2500_2_2_4000_4000_\\
1000_4000_4000_1000_100_0.7.dat

Queue
```

図 6 Condor におけるジョブ投入用スクリプト作成例

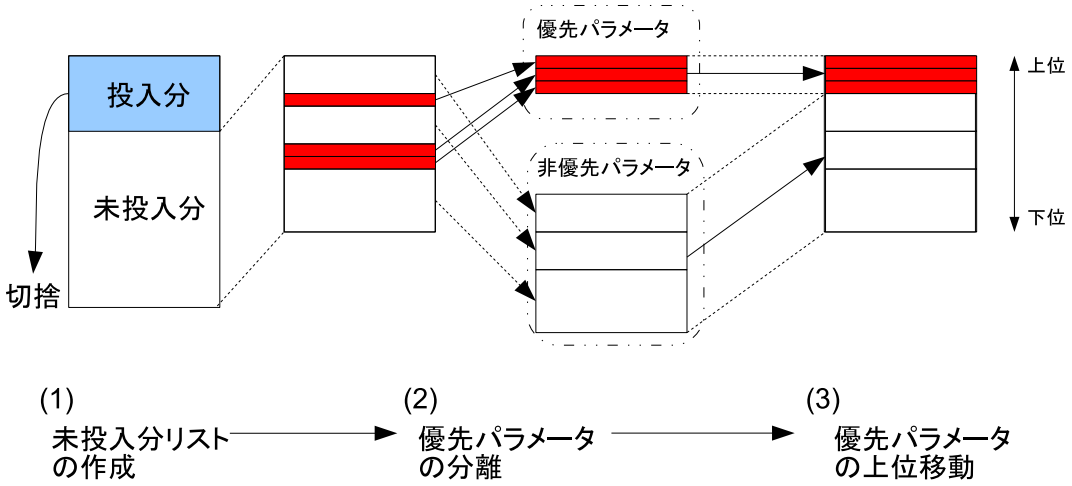

図 7　パラメータリストの更新手順

- 高スループット計算のうち、データ並列計算やパラメータスウィープ型計算のように全探索が必要な計算の場合、パラメータリストファイルがシーケンシャルに定義されていると、他のパラメータの計算結果を見ることができない。

- シミュレーション実施者の動作として、特にシミュレーションの予備実験の段階ではパラメータを変更して複数のシミュレーションを実施し、様子を見ることが頻繁に行われる。このとき、大量タスク実行の放棄とパラメータ変更による再実行の繰り返しが頻繁に行う。シミュレーション実施者自身がパラメータの実行状況を管理せねばならない。重複したパラメータでタスク実行してしまうこともあり得るため、計算機資源を無駄に利用することにつながる。

そこで、既に得られた計算結果を利用したシミュレーション実施者の意思決定に基づくシミュレーション実施を行うために、プログラム実行モジュールにて提供されているタスク優先実行制御機能を用いた。

タスク優先実行制御機能は、パラメータリストファイルに記述されている順序を変更し、ある特定の値をもつパラメータの組を優先して実行するものである。タスク優先実行制御機能の動作の流れについて、図 7 を用いて説明する。また、具体例として、図 8、図 9、図 10 を用いる。それぞれ、初期パラメータリスト、優先実行するパラメータの情報、タスク優先実行制御機能によって作成された優先タスクリストである。

1. まず、シミュレーション実施者は随時出力されるシミュレーション結果を確認し、優先実行させたいパラメータの値を決定してもらう。優先実行させたいパラメータの値が決定したら、その値を操作インターフェイスに入力する。入力された値は操作インターフェイスによって図 9 に示すデータを出力する。優先実行させたいパラメータを示すデータは CSV の書式で、第 1 番目がパラメータの位置を、第 2 番目がその値を示す。アスタリスクで記述されているパラメータの位置はどの値でも構わないことを意味する。図 9 の例では、7 番目と 8 番目の引数が 4000 の値をもつリストを優先する意味である。

```
 1: 1000 1000 -1   -1   2500 3000 2500 2500
 2: 3000 2500 -1   -1   2500 2000 2500 2500
 3: 4000 1000 4000 1000 2500 2000 4000 4000
 4: -1   -1   1000 1000 4000 2500 2500 2500
 5: 1000 1000 -1   -1   1500 4000 2500 2500
 6: -1   -1   4000 4000 1500 4000 4000 1000
 7: 4000 4000 -1   -1   2500 4000 2500 2500
 8: 1000 1000 -1   -1   4000 1000 2500 2500
 9: 4000 4000 -1   -1   2500 4000 4000 1000
10: 1500 4000 1500 4000 2000 2500 4000 4000
11: 4000 4000 4000 4000 2500 4000 4000 4000
12: 2500 2500 -1   -1   1000 1000 4000 4000
13: -1   -1   3000 2500 2500 3000 4000 1000
14: 1500 4000 -1   -1   1000 1000 2500 2500
15: -1   -1   2500 2500 1000 1000 4000 1000
16: -1   -1   3000 2500 2500 4000 4000 1000
17: -1   -1   4000 1000 1000 1000 2000 4000
18: -1   -1   2000 2500 4000 2500 2500 2500
19: 4000 1000 -1   -1   3000 2500 2000 4000
20: 4000 1000 -1   -1   1000 1000 2000 4000
```

図 8 優先タスクリスト作成前 (上位 20 行, 左端は行番号)

```
param01, *
param02, *
param03, *
param04, *
param05, *
param06, *
param07, 4000
param08, 4000
```

図 9 優先実行するパラメータの情報

```
 1: 1500 4000 1500 4000 2000 2500 4000 4000
 2: 4000 4000 4000 4000 2500 4000 4000 4000
 3: 2500 2500 -1   -1   1000 1000 4000 4000
 4: 2000 2500 -1   -1   2500 2000 4000 4000
 5: 1000 1000 -1   -1   3000 2500 4000 4000
 6: 1000 1000 1000 1000 2500 4000 4000 4000
 7: 2500 2500 2500 2500 4000 4000 4000 4000
 8: 1500 4000 -1   -1   3000 2500 4000 4000
 9: 1500 4000 1500 4000 2500 2000 4000 4000
10: 4000 1000 -1   -1   2500 1000 4000 4000
11: 2000 2500 -1   -1   3000 2500 4000 4000
12: 2000 2500 -1   -1   2500 2500 4000 4000
13: 2000 2500 2000 2500 4000 1000 4000 4000
14: 3000 2500 3000 2500 1500 4000 4000 4000
15: 2000 2500 -1   -1   4000 1000 4000 4000
16: 2500 2500 2500 2500 2500 2000 4000 4000
17: 3000 2500 -1   -1   2000 2500 4000 4000
18: 1500 4000 1500 4000 1000 1000 4000 4000
19: 3000 2500 -1   -1   2500 4000 4000 4000
20: 2500 2500 2500 2500 1000 1000 4000 4000
```

図 10 優先タスクリスト作成後 (上位 20 行, 左端は行番号)

2. プログラム実行モジュールは、優先実行させたいパラメータの値を受けとると、次の操作を行う。

   (a) 実行し終えたパラメータリストを除いたパラメータリストを作成する (図 7-(1))。これを差分パラメータリストと呼ぶ。この操作は、同一パラメータリストの重複実行を避けるために行う。付録に掲載する大量タスク実行用スクリプトでは、submitdat.bat というファイルに実行し終えたパラメータリストが記述される。このファイルに書かれているパラメータをパラメータリストから抜き出す。

   (b) 差分パラメータリストから、優先実行させたいパラメータの値をもつパラメータリストとそうでないパラメータリストと分離する (図 7-(2))。前者を優先パラメータリスト、後者を非優先パラメータリストと呼ぶ。差分パラメータリストの先頭から順に、優先実行させたいパラメータの値を持っているか比較する。比較した結果、優先パラメータリストであれば、優先パラメータリストファイルとして一時的に出力する。また、非優先パラメータリストであれば、非優先パラメータリストとして一時的に出力する。

   (c) 優先パラメータリストファイルに非優先パラメータリストファイルの内容を追記する (図 7-(3))。この結果得られたものを修正パラメータリストと呼ぶ。修正パラメータリストの例を図 10 に示す。7 番目と 8 番目の引数が 4000 になっている。また、初期パラメータリストの 10 行目、11 行目、12 行目が修正パラメータリストの 1 行目、2 行目、3 行目になっている。

3. プログラム実行モジュールで作成した修正パラメータリストファイルを用いて、タスク実行を再開する。タスク実行支援モジュールでは、大量タスク実行スクリプトによるタスク実行を行うので、現在実行中の大量タスク実行スクリプトを放棄する。そして、修正パラメータリストファイルを大量タスク実行スクリプトの第 2 引数に設定し、タスク実行を再開する。

3.2.3 ユーザインターフェイス

シミュレーション実施者が優先タスク実行を可能にするために必要なユーザインターフェイスはパラメータ変更用操作インターフェイスと結果確認用インターフェイスの 2 種類である。

- パラメータ変更用操作インターフェイスは、図 11 に示すように、Web ブラウザをインターフェイスとして採用する。パラメータ変更用操作インターフェイスでは、初期パラメータリストの入力、優先実行させるパラメータ値の入力が可能である。

- 結果出力インターフェイスは、パラメータ変更用操作インターフェイスと同様、図 12 に示すように、Web ブラウザをインターフェイスとして採用する。託児所配置問題解決システムでは、パラメータごとに結果が出力される。シミュレーション実施者が結果を確認する場合、パラメータごとに結果を確認する場合と複数の結果を比較しながら確認する場合が考えられる。そこで、結果確認用インターフェイスでは、タスク実行が完了したパラメータリストの一覧から、シミュレーション実施者が興味をもつパラメータリストを選択する (図 12 左側)。そして、選択されたパラメータに対してグラフ出力するよう

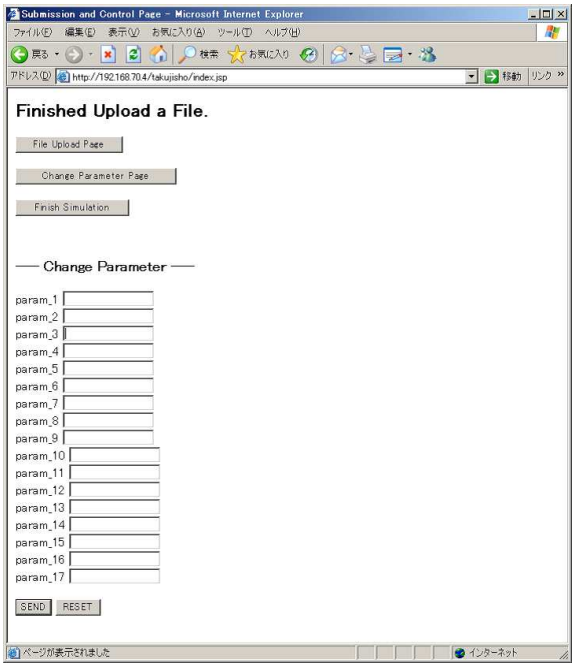

図 11 パラメータ変更操作ページ

に施した (図 12 右側)。これにより、パラメータの違いによるシミュレーション結果の違いを比較できる。

3.2.4 シミュレーション実施者が得られる効果

託児所配置問題解決システムをタスク実行支援フレームワークに適用することによって、シミュレーション実施者は以下の効果が得られる。

- シミュレーション実施者の操作により実施者の希望に基づくパラメータの組を優先的に実行できる。これにより、パラメータの組をひとつずつ入力しながら結果を得るのと同じ効果が得られる。
- パラメータの組み合わせすべてを実行する必要がなくなる。これにより、最適解探索おける解候補の早期発見が可能になる。また、託児所配置問題では様々な条件下でのシミュレーションを短期間で行うことが可能になる。

3.3 対話的動作を考慮することによるシミュレーション結果の改善効果

シミュレーション実施者の意図的な動作が継続的に行われることで、託児所配置問題の結果にどのような改善効果をもたらすかについて述べる。

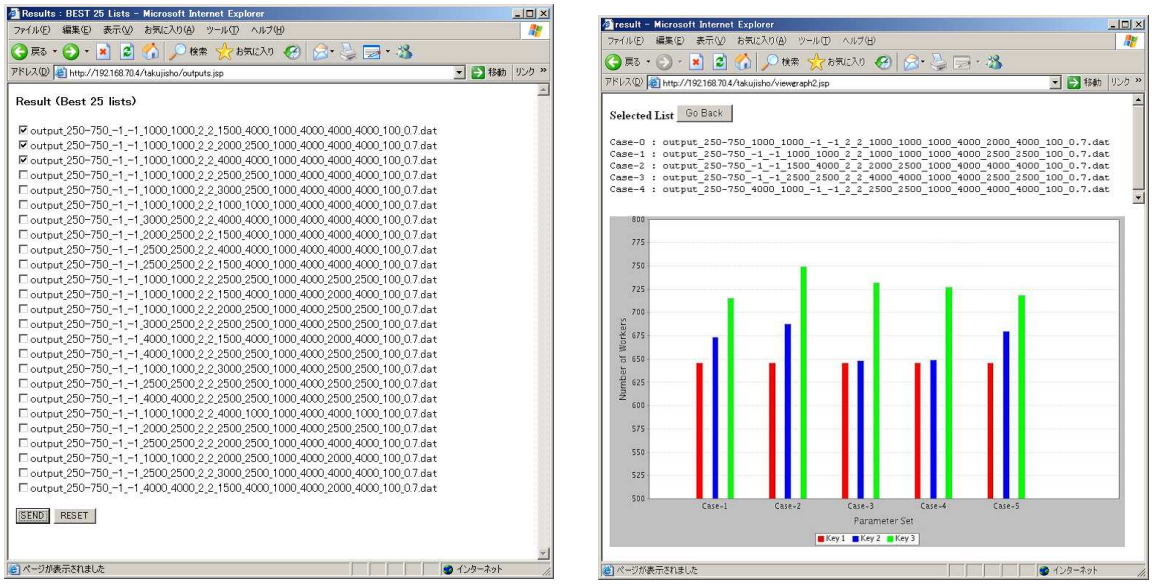

図 12 出力結果確認ページ (左:選択画面, 右:グラフ描画)

3.3.1 計算機環境

託児所配置問題解決システムで使用した計算機の性能を示す。本稿においては PC クラスタを利用した。PC クラスタは Intel 社 3.0EGHz Pentium 4 プロセッサを搭載した PC を 10 台で構成したホモジニアスクラスタである。すべての PC は 1Gbps の Ethernet ポートを有しており、プロセッサ間の通信速度は 1Gbps である。

PC クラスタでのジョブ管理は Condor(Version 6.6.7) を使用した。Condor では実行するプログラムを各計算ノードにコピーする機能を有しているので、プロトタイプシステムではこの機能を利用して、各パラメータの計算結果を回収する。

PC クラスタを構成する計算機のうち 1 台は、プログラム実行モジュール、状況確認モジュール、結果解析モジュールを担当する。このノードをサーバノードと呼ぶ。残り 9 台はすべて計算ノードとした。なお、サーバノードは計算ノードの作業も行うため、シミュレーション実施者からは 10 台の計算ノードが登録されているように見える。

サーバノードではパラメータ入力および変更、シミュレーション結果の出力を可能にするために、Java Server Pages(JSP) を利用した。プロトタイプシステムでは Tomcat(Version 4.1.30) を使用した。シミュレーション実施者は Web ブラウザを利用してパラメータの入力と変更、シミュレーション結果の確認を行う。

3.3.2 シミュレーション実施者によるタスク実行操作の比較

シミュレーション実施者がシミュレーション結果を逐次確認した後、パラメータリストの順序変更によるシミュレーションの再実行が必要な場合の動作を比較する。託児所配置問題はパラメータスウィープ型のシミュレーションであるから、大量のタスク実行が発生する。シミュレーションの再実行が必要と判断し、問題解決システムにシミュレーションの再実行を反映させるには、

表 1 シミュレーション実施者によるタスク実行制御動作の比較

動作項目	既存ミドルウェアのみ	フレームワーク適用後
1. 投入されたすべてのタスク放棄	実施者による操作	フレームワークが吸収
2. パラメータリストから実行完了したパラメータを分離	実施者による操作	フレームワークが吸収
3. パラメータの特徴抽出	実施者が確認	実施者が確認
4. 未実行パラメータリストの作成	実施者が作成	フレームワークが吸収
5. 新たなパラメータリストによる再実行命令	実施者による操作	フレームワークが吸収

1. タスク管理システムに投入されているタスクをすべて放棄して、シミュレーションの実施を中断する
2. 実行が完了したパラメータのタスクと実行されていないパラメータのタスクを整理する
3. 実行完了したタスクの結果から、評価値が最大になると思われるパラメータの特徴を抽出する
4. 特徴抽出結果から優先実行されるパラメータを上位にした未実行パラメータリストを作成する
5. 新しいパラメータリストに基づいて、シミュレーションの実行を再開する

の手順が必要になる。

表 1 はシミュレーション実施者がパラメータリストの変更を行う際に生じる動作を既存グリッドミドルウェアのみを利用した場合と提案するフレームワークを適用した場合での比較した結果である。タスク実行支援フレームワークを適用することで、シミュレーション実施者は 5 つの動作から、3. パラメータの特徴抽出に軽減できる。

タスク実行支援フレームワークを適用した場合の動作の流れは以下のようになる。

1. シミュレーション実施者の作業
   (a) シミュレーション結果を確認する。
   図 12 に示した出力結果確認ページを用いて、評価値が最大となるパラメータを確認する。出力結果確認ページでは、複数のパラメータリストを選択して評価値を比較するグラフが出力できる。また、パラメータ一覧を出力する際に、評価値の高い順にパラメータを並び替えて出力することで、評価値を比較するためのグラフを確認する手間が減る。
   (b) パラメータの特徴を抽出する。
   出力結果確認ページで評価値を確認した結果、評価値が高くなると思われる引数の位置や値を確認する。
   (c) 特徴抽出した結果をタスク実行支援フレームワークに指示する。
   託児所配置問題解決システムではタスクフローの修正はないため、パラメータ変更

操作のみである。図 11 に示したパラメータ変更操作ページを使う。パラメータ変更操作ページはタスクの引数位置と関連づけているので、該当するパラメータの位置に優先したい値を入力する。入力し終えたら、送信ボタンを押す。

2. タスク実行支援フレームワークによる作業

   (a) 投入されているすべてのタスクを放棄する

   (b) パラメータリストから実行完了したパラメータを分離し、優先実行するパラメータを上位にした未実行パラメータリストを作成する。

   (c) 新しいパラメータリストに基づいてシミュレーションの実行を再開する。

この結果から、タスク実行支援フレームワークを適用することで、シミュレーション実施者には下線部の作業が増えるものの、解探索時間の短縮に必要な作業が軽減できる。また、パラメータリストを修正せずすべてのパラメータを実行する場合と比較した場合、結果確認とパラメータ修正依頼の動作に抑えることができるため有効であるといえる。

3.3.3 シミュレーション実施者の操作による解探索結果および解探索時間の変化

パラメータスウィープ型のシミュレーションは、一般的に各パラメータの値の集合から、すべての組み合わせを作成し、組み合わせに基づいてシミュレーションを行う。すべての組み合わせによるシミュレーション結果が出力されたのち、目的を満足する各パラメータの値を探しだす。

ところで、パラメータの組み合わせの多くは目的を満足するか判断しにくい組み合わせである。パラメータを与えてシミュレーションを行うことではじめて目的を満足するか否かが判別できる。従来は、すべての組み合わせをシミュレーションし終えてから、良い結果だけを選別する。目的を満足するパラメータの組み合わせを確実にみつけだすことができるが、シミュレーションに要する時間も長時間必要になる。

本論文で提案する託児所配置問題解決システムでは、シミュレーション実施者の知識と経験による操作を必要とするが、すべての組み合わせのシミュレーションを行わなくとも目的を満足するパラメータの組み合わせをみつけることができるようになる。そこで、すべての組み合わせのシミュレーションを行わなくとも目的を満足するパラメータの組み合わせをみつけだせるか確認するための実験を行った。実験における条件は以下の通りである。

- 託児所配置シミュレーションの環境条件
  本実験で変更可能なパラメータは職場位置、エージェントの種類、エージェントの中心分布、託児所設置位置とする。各パラメータの数はそれぞれ 4 通り、2 通り、7 通り、7 通りとする。これら以外の条件および具体的な値は村田らの報告と同様とする。変更可能なパラメータから作られるパラメータの組み合わせは 394 通りとなる。

- 初期パラメータリストファイルの条件
  本実験で用いたパラメータリストは一様乱数によってパラメータの順序を変更する。パラメータの順序を変更するために使った乱数は Matsumoto らによって開発された Mersenne Twister と呼ばれる疑似乱数生成アルゴリズム [8] である。

表 2 被験者による結果の改善効果 (ソートなし)

	上位 5 位内	上位 10 位内	タスク数	変更回数
被験者 1	×	×	193	7
被験者 2	○	○	173	10
被験者 2	○	○	296	4

表 3 被験者による結果の改善効果 (ソートあり)

	上位 5 位内	上位 10 位内	タスク数	変更回数
被験者 1	○	○	202	8
被験者 2	○	○	172	8
被験者 3	○	○	283	9

- 託児所配置問題解決システムを操作する被験者の振る舞い
  被験者は常に問題解決システムの前に座ったまま実験するのではなく、被験者の都合の良いタイミングでシミュレーション結果の確認をしてもらうようにした。シミュレーション結果を確認する時点で、希望する結果と異なると判断した場合、パラメータ変更用ユーザインターフェイスを用いてタスクの優先実行をしてもらった。そして、託児所配置の評価値が最大になるパラメータの組み合わせが、以後現れないと思うまで被験者の意思決定に基づくパラメータ変更の操作を行ってもらう。

  なお、パラメータ変更の操作はすべてのパラメータの組み合わせが終わるまで可能とする。

- 結果出力の提示方法
  本実験では、被験者に対して 2 種類のシミュレーション結果を提示する。ひとつは、シミュレーション結果の良い順に並べたもの、もうひとつはシミュレーション結果が出力された順に並べたものである。差をつけることで、シミュレーション実施者の結果出力確認時の負担が及ぼす影響を確認する。

シミュレーション実施者の操作による結果の改善効果を表 2、表 3 に示す。表 2 は被験者に対して計算結果をそのまま提示した場合 (ソートなし) の結果、表 3 は被験者に対して上位 25 位までの計算結果を提示した場合 (ソートあり) の結果である。この実験では、3 人の被験者に 2 種類の実験をしてもらった。

本稿では 10 台のホモジニアス PC クラスタを用いているため、最適解決定までのタスク数を解探索時間とみなした。最大タスク数は 394 個であるから、それよりも小さい値であれば解探索時間が短縮できたとみなせる。また、マルチエージェントシミュレーションプログラムは内部で乱数を用いているため、パラメータリスト 1 つあたり 100 回の演算を行う。1 つのパラメータリストから出力される評価値は 100 回の演算結果の平均値とする。100 回の演算結果の平均値を評価値として採用しているため、シミュレーション全体としては一定の傾向を出力す

る。試行実験を繰り返すことで、小さな範囲での順位変動があり得る。そこで、被験者が最適解と選択したパラメータの組み合わせが上位 5 位内もしくは上位 10 位内に収まっているかを確認した。

実験の結果、最適解を選択するまでのパラメータ変更回数、最適解を選択するまでのタスク数に大きな差は見られなかった。但し、ソートした計算結果を被験者に提示することで改善の兆しが得られた。被験者 1 においては上位 5 位内、上位 10 位内の結果を出力できるようになった。また、被験者 2 は変更回数が低減した。逆に、被験者 3 はソートしない結果を提示した場合のほうが変更回数が少なかった。被験者 3 はソートしない場合を提示された場合は最適解の選択作業に時間を費やしたが、ソートする場合を提示されると選択作業にかかる負担が軽減されるため、より多くのパラメータ変更を行う意欲が起きたと考えることができる。

最適解を選択するまでの解探索時間については大幅に改善されたといえる。ソートした計算結果を被験者に提示した場合、被験者 1 については 48.8%、被験者 2 については 56.4%、被験者 3 については 27.8%、にあたるタスクの実行を削減できた。

3.3.4 組み合わせ数増加によるシミュレーション実施者の継続的改善の効果

託児所配置問題解決システムに対話的動作を考慮することで、パラメータに与える値の種類が増加しても、必要な計算結果を優先的に確認することが可能になる。

マルチエージェントシミュレーションでは、評価値を最大にするパラメータの値を探すために、各パラメータに対して想定される値の集合を準備することになる。シミュレーションプログラムを実行する回数、言い替えると組み合わせの数は、パラメータの数および各パラメータとして与える値の集合から指数的に増加する。

従来、マルチエージェントシミュレーションでは、指数的に増加する組み合わせの数を少しでも減らすために、マルチエージェントシミュレーションの定性的特徴と 2 次元空間の対称性といった問題特有の特徴を考慮してパラメータリストを作成していた。そのため、評価値を最大にする各パラメータの値の範囲を特定するにとどまっていた。

ここで、政策立案のような実務的な観点で使うことを想定する。実務的な観点では、パラメータの値は定性的なものではなく、定量的なものが要求される。従来は、予備実験として定性的なシミュレーションを行い、ある程度の絞り込みを行ってから具体的な値の特定を行うことになる。すなわち、同じシミュレーションを 2 回実施することになる。

これに対し、提案する託児所配置問題解決システムでは、単純に各パラメータに与える値の数を増やしても、シミュレーション実施者の継続的な改善によって、パラメータ値の特定を 1 回の実施でよい。そこで、パラメータの組み合わせの数を増やし、かつシミュレーション実施時間を制限した状況下で、目的を満足する各パラメータの値をみつけだせるかの実験を行った。この実験では、シミュレーション実施者の継続的な改善による解探索の効果を確認する。

実験における条件は先の実験と同様の条件とするが、異なる点について示す。託児所配置シミュレーションの環境条件について、パラメータリストの数を増やすために、託児所設置位置を 7 通りから 12 通りに増やす。この変更によって作られるパラメータの組み合わせは 672 通りになる。

託児所配置問題解決システムを操作する被験者について、被験者の都合の良いタイミングでシミュレーション結果の確認をしてもらう。本実験では、シミュレーション実施時間を制限し

表 4 パラメータ数増加による結果の改善効果

	上位 5 位内	上位 10 位内	タスク数	変更回数
被験者 1	○	○	374	14
被験者 2	×	×	160	13

た状況を想定した。そこで、被験者が意思決定しながら操作できる時間は 3 時間以内とする。シミュレーション開始からこれは、672 通りの半数にあたる 386 通りの組み合わせが完了する時間が 2.8 時間になることから決定した。

被験者 2 名の操作による結果の改善効果を表 4 に示す。先ほどの実験同様、被験者に対して上位 25 位までの計算結果を提示した。被験者 1 は制限時間直前までパラメータの変更を行い、最適解の可能性が高いと思われるパラメータの組み合わせを実行した結果を反映している。一方、被験者 2 は短時間での解決を試みたため、被験者 2 が選択した最適解は上位 10 位内にも含まれていない。パラメータの組み合わせ数を増やし、かつ被験者の操作可能な時間に制限がある場合、被験者の判断ミスにより、必ずしも良い結果が得られるとは限らない。

4. 政策立案の現場へ適用するための課題

提案する託児所配置問題解決システムは、将来的には政策を立案する現場での活用を目指している。本稿では託児所配置問題解決システムの一例を示したが、政策立案の現場に適用するには課題が残されている。システム構築の観点から、政策立案の現場で用いるための課題について述べる。

4.1 パラメータリスト作成における課題

本稿では、初期パラメータリストファイルを作成し、初期パラメータリストファイルの記述順にタスク実行が行われる。政策立案者が、政策上の条件をもとに定義されるパラメータの値を初期パラメータリストファイルに記述する必要がある。また、実験において、処理されたパラメータの組み合わせを一意に確認する仕組みを提供しなかったため、被験者の判断ミスが発生した。

対策としては、グラフィカルな操作インターフェイスを提供する方法が考えられる。グラフィカルな操作インターフェイスを介して、初期パラメータリストファイルの作成、優先実行するタスクの選択操作、計算結果を見ながら不必要と思われるパラメータリストの排除操作をできるようにする。

4.2 複数のグリッド環境を利用するための課題

政策立案者はマルチエージェントシミュレーションの定性的特徴を理解しているとは限らない。そのため、初期パラメータリストは政策立案上のあらゆる場合を想定して作成される。作

成されたパラメータリストは大量のタスク実行を伴うことから、大量タスク実行に見合った計算機能力が必要になる。本稿ではタスク管理システムとして Condor を採用した PC クラスタに適用した。より短い時間でのシミュレーションを実現するには、必要に応じて複数の組織にある計算機を利用することが望まれる。

対策としては、パラメータ変更部において複数のタスク管理システムに対応したタスク投入モジュールを実装することが挙げられる。現状では、様々なタスク管理システムが用いられているが、操作体系そのものは Condor の場合と変わらない。したがって、Condor 以外のジョブ管理システムに対応した改良を行う。

5. まとめ

本稿では、マルチエージェントシミュレーションによる託児所配置問題について、タスク実行支援フレームワークに基づいたグリッド環境への適用について述べた。

本稿で示した託児所配置問題解決システムでは、随時出力されるタスクの結果を確認しながら、最適解をもつパラメータリストを優先的に実行することが可能になる。対話的動作による解探索時間と最適解探索の効果について実験を行い、適切な計算結果の提示方法を採用し、かつ被験者の対話的動作を可能にすることで、最適解の探索が短期間でできるようになることがわかった。

最後に、システム構築で得られた内容を踏まえ、政策立案の現場へ適用するための課題について示した。政策立案の現場に適用するためには、グラフィカルなユーザインターフェイスによるパラメータリスト作成支援ツールと複数のタスク管理システムに対応したタスク投入モジュールの実装が必要であることを示した。

付録: 大量タスク実行用スクリプト

大量タスク実行用スクリプトでは、各パラメータのタスク実行および放棄を容易にするために、一定値以上のタスクを投入しない工夫を施した。ここでは、ローカルのタスク管理システムとして Condor を用いた場合の大量タスク実行用スクリプトを示す。

```
#!/usr/bin/perl

$progname=$ARGV[0];     # プログラム名
$paramIniFile=$ARGV[1]; # パラメータリストファイル名
$sleeptime = 20;         # サンプリング時間
$max_lines = 12;         # Condor にタスクを投入する際の最大値
$submitcounterFile = "/var/log/submitcounter.dat";
$allCountFile="/var/log/suballcount.ini";
$startlines = 0;

open(OIn, $allCountFile) or die;
@readdataCount = <OIn>;
close(OIn);
@lines_count = split(/\n/,$readdataCount[0]);
```

```
$allCount=$lines_count[0];

open(OIn, $paramIniFile) or die;
@paramlistdata = <OIn>;
close(OIn);
open(OIn, $submitcounterFile) or die;
@countdata = <OIn>;
close(OIn);
@lines_count = split(/\n/,$countdata[0]);
$i=$lines_count[0];


while($#paramlistdata >= $i) {
    # タスク実行状況の確認
    $str = &checktask;
    @lines_tmp = split(/\n/,$str);
    $lines = 0;
    $lines = $#lines_tmp;

    # タスクバッファに空きが生じたら新たなタスクを投入する
    if($lines < $max_lines) {
        $j=$i+1;
        $tmp_char = $readdata[$i];
        @get_filename = split(/\s/,$tmp_char);

        # タスク実行スクリプトの生成およびタスク実行
        if($startlines < $j) {
            # パラメータを付与したタスク実行
            &tasksubmit;

            # 実行したタスクのパラメータを出力する
            open(FOut, ">>./submitdat.bat");
            print FOut "$paramlistdata[$i]";
            close(FOut);

            $i++;
            open(OOut, ">./submitcounter.dat") or die;
            print OOut "$i";
            close(OOut);

            $allCount++;
            open(OOut, ">$allCountFile") or die;
            print OOut "$allCount";
            close(OOut);
            sleep $sleeptime;
        }
        else {
            print "already submit a job [$i]\n";
            $i++;
            open(OOut, ">./submitcounter.dat") or die;
            print OOut "$i";
            close(OOut);
        }
    }
    else {
        sleep $sleeptime;
    }
}
```

```
# パラメータを付与したタスク実行のサブルーチン (Condor の場合)
sub tasksubmit{
    open(FOut, ">./calcv4-$allCount.cmd");
    print FOut "UNIVERSE = vanilla\n";
    print FOut "Executable = $progname\n";
    print FOut "Log=calcv4-$i.log\n";
    print FOut "Output = calcv4-$i.out\n";
    print FOut "Error = calcv4-$i.err\n";
    print FOut "\n";
    print FOut "Arguments = $paramlistdata[$i]\n";
    print FOut "\n";
    print FOut "should_transfer_files = YES \n";
    print FOut "when_to_transfer_output = ON_EXIT\n";
    print FOut "\n";
    print FOut "transfer_input_files = AGE.txt, C18.txt, ED.txt,
                                        HI.txt, parameter.txt, WORK.txt \n";
    print FOut "transfer_output_files = $get_filename[16]\n";
    print FOut "\n";
    print FOut "Queue\n";
    print FOut "\n";
    close(FOut);

    $submit_command = "/usr/local/condor/bin/condor_submit
                                                    ./calcv4-$allCount.cmd\n";
    $system_message = system($submit_command);
}

# ローカルのタスク管理システムに投入されているタスクの数を確認する
sub checktask {
    $taskstr = `/usr/local/condor/bin/condor_q | /bin/grep calcv`;
    return $taskstr;
}
```

参考文献

[1] 車谷浩一. マルチエージェント社会シミュレーション展望. システム/制御/情報, Vol. 46, No. 9, pp. 518–523, 2002.

[2] 鵜飼康東, 村田忠彦, 北埜裕子. 既婚女性の労働供給における政策グリッドコンピューティング実験. 関西大学経済論集, Vol. 55, No. 3, pp. 421–443, 12 2005.

[3] Noriyuki Tanida and Masatoshi Murakami. Agent Based Modelling for Pension Systems(1) - Fundermental Design. *The Economic Review of Kansai University*, Vol. 54, No. 2, pp. 22–36, 2004.

[4] 平成 17 年度版 労働経済の分析 (労働経済白書). 厚生労働省.

[5] 「少子化の要因と少子化社会に関する研究会」報告書. 財務省財務総合政策研究所, 2005.

[6] Tadahiko Murata, Hiroko Kitano, Tomoharu Nakajima, and Hisao Ishibuchi. Application of a Multi-Agent Model with Pioneers and Followers to a Day Care Center Allocation Problems. In *International Conference on Intelligent Technologies 2003*, pp. 179–186, 2003.

[7] 蟻川浩. 高スループット計算を指向したグリッド環境構築におけるタスク実行支援フレームワークに関する研究. 奈良先端科学技術大学院大学 情報科学研究科 博士論文, 2006.

[8] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transaction on Modeling and Computer Simulations*, Vol. 8, No. 1, pp. 3–30, January 1998.