2007－ICS－147 (16)
2007／3／15

動的な負荷分散問題へのエージェントベースアプローチ

○大宮健太（公立はこだて未来大学）
鈴木恵二（公立はこだて未来大学，CREST, JST）

本研究では、動的な負荷分散問題に対して社会的ジレンマ問題に対するエージェントベースアプローチを適用し、その有効性の検証を行う。本稿では、動的負荷分散問題としてグリッドコンピューティングにおけるタスクスケジューリング問題を対象とし、メタエージェントによる階層化と状況に応じた役割選択機能の導入を行った。即ち、本研究では、動的にシステム構成が変化するスケジューリングシステムを提案し、動的な負荷分散問題への有効性の検証を行う。今回は、その前段階として、状況に応じた役割選択によるシステム構成の決定について詳細を記す。

An Agent-based Approach for Dynamic Load Balancing

○Kenta Oomiya (Future University-Hakodate)
Keiji Suzuki (Future University-Hakodate, CREST)

Abstract

The purpose of this paper is to examine the effectiveness of our proposed scheduling system for dynamic load balancing problems. The proposed scheduling system is applied to scheduling problems in a grid computing environment. Our approach is one of meta-agent-based approaches. In this approach, agents can dynamically decide constructs of the scheduling system by deciding each of agent's roles. This approach can introduce robustness into scheduling systems for dynamic load balancing problems. This paper explains the basic of this autonomous role selection and its effectiveness.

1. はじめに

近年、テーマパーク問題[1]やグリッドコンピューティングにおける負荷分散問題などに代表される、動的かつ大規模な環境下での共有資源分配に関する問題に注目が集まっている。スケジューリングや負荷分散問題などに対しては従来、ゲーム理論を背景としたアプローチがなされてきたが、ゲーム理論の求めるシンプルさと実問題が持つ複雑かつ巨大さがミスマッチを起こしており、新たなアプローチが求められている。新たなアプローチとして、マルチエージェントアプローチが提案されているが、本研究では更に一歩進めた「社会性エージェント群」によるアプローチを用いる。社会性エージェントとは、従来のマルチエージェントが基本的に同じ立場、同じ役割、同じ能力でエージェントを扱っているのに対して、実社会と同じように「組織化」を行うエージェント、すなわち、役割と目的の異なるエージェント群を扱う。さらにこの役割と目的を問題設定者があらかじめ設定するのではなく、シミュレーションの中で、自律的に創発させる技術を組み込んでいる点に新規性があると考えられる。

社会性エージェントアプローチに関する先行研究としては、メタエージェントによる課税戦略[3][4][5]をゲーム理論的な問題に適用している研究が挙げられる。これらの研究では、共有資源問題をN人社会的ジレンマ問題として定式化し、メタエージェントの課税によってゲームの利得構造を変更する事で問題の解決を図っている。また、我々の先行研究[5]では、メタエージェントによる課税戦略に、メタエージェント化機能による拡張を行う事で、動的な役割変化を行うエージェント群によるアプローチの提案と有効性の検証を行っている。

本研究では、グリッド環境におけるコンピュータリソースの集合を共有資源として捉え、共有資源問題に有効な社会性エージェントアプローチの適用を試みる。

2. グリッド環境における負荷分散問題

グリッドコンピューティングとは、地理的、組織的に離れた複数の計算資源をネットワークで接続・共有し、ユーザがそれらを仮想的な巨大コンピュータのように利用できるようにするためのインフラストラクチャである[6]。図 1 に、グリッドコンピューティング環境の例を示す。この例では、複数のプロセッサを持つコンピュータが、ネットワークによって接続されている。グリッドコンピューティングでは、複数のユーザが入力したタスク群を複数のコンピュータに対して配分し、処理を行う。この時に、それぞれのタスクの大きさ、締め切り、タスクの数などのタスクに関する情報と、プロセッサの処理能力などのコンピュータリソースに関する情報を配慮しながら、タスクの負荷分散問題を解く必要がある。この環境で問題となる点は、グリッド環境における計算資源の処理能力が動的に変化する点である。静的なスケジューリングアルゴリズムの場合、スケジューリング時からタスクが実際に処理されるまではタイムラグが大きく、その間に計算資

源の処理能力の変化が発生する事が考えられる。タスクを割り当てた計算資源に対する負荷がスケジューリング時よりも大きくなった場合、スケジューリングと実際の実行時間の間に大幅な遅れが発生してしまう。上記のような問題が発生する理由としては、スケジューリング開始前に行われる静的な情報のみを使ってスケジューリングしている事が考えられる。この問題に対応する手段として様々な方法が考えられているが、本研究では、段階的にスケジューリングを行う手法を採用する。理由としては、各段階を役割として設定する事で、スムーズに社会的エージェントアプローチの適用が可能であると考えられるからである。

3. 提案手法

本研究では、動的かつ大規模な共有資源問題に対して社会性エージェントベースアプローチを適用し、その有効性の検討を行う。アプローチの適用対象として、グリッドコンピューティングにおけるタスクスケジューリング問題を採用している。ここでは、その適用対象の詳細について述べる。図 1は、本研究で想定しているグリッド環境の一例である。

この例では、大学が3つ有り、それぞれの大学に複数の CPU を持ったクラスタマシンが一台以上ある状況である。提案スケジューリングシステムでは、クラスタマシンをエージェントとして扱い、それぞれのエージェントが自身の役割を選択する。役割としては、タスクの実行を行うローカルスケジューラと、それらにタスクを割り振るグローバルスケジューラのどちらかである。提案システムでは、これら2種類のスケジューラを組み合わせることで負荷分散問題の効率化を図る。最終的な目標は、この手法に社会的エージェントアプローチを適用することである。即ち、エージェントが状況に応じて、どちらのスケジューラとして働くかを選択する事で、動的な階層構造を形成し、ロバストなシステムを構築できるような手法を提案することである。

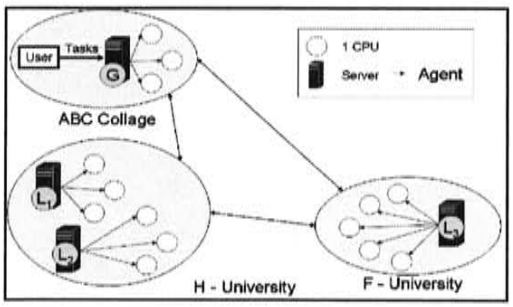


図 1 グリッド環境の一例

3.1 共有資源問題との対応

ここでは、グリッドコンピューティングにおけるタスクスケジューリングを共有資源問題として扱う場合、その対応関係について論ずる。

グリッドコンピューティング環境における共有資源は、ネットワークによって接続されているコンピュータリソース（CPU などの計算資源、ハードディスクなどの記憶資源、ネットワークの大域幅など）を指す。エージェントはユーザから指定されたタスクを自身のコンピュータリソースを用いて実行する事になるが、これらのエージェントの目的は出来るだけ自身のコンピュータリソースを使わない状態でいる事である。即ち、エージェントの個人的合理性は出来るだけタスクを実行しない事であり、システム全体の合理性は出来るだけ早くタスクを完了することである。従って、エージェントの協調行動はタスクを実行する事を示し、裏切り行動はタスクを実行しない事を示している。多くのエージェントがタスクを実行しないという裏切り行為を選択した場合、タスクが実行されないという悲劇的状況が発生する事になる。

3.2 提案システムについて

図 2に提案システムの概要図を示す。システムの構成要素は2種類で、タスクとエージェントである。但しエージェントはグローバルスケジューラ、ローカルスケジューラのどちらかの役割を選択する。グローバルスケジューラは、タスクをどのローカルスケジューラに分散させるべきかのスケジューリングを行う。ローカルスケジューラは、与えられたタスクをどの様な順番で処理するべきかというスケジューリングを行う。即ち、提案手法は、二段階でスケジューリングを行う手法の一種である。

Cao らは論文[7]で、グリッドコンピューティングにおける負荷分散問題のモデルを以下の様に定義している。本研究では、その定義を基に拡張を行っていく。

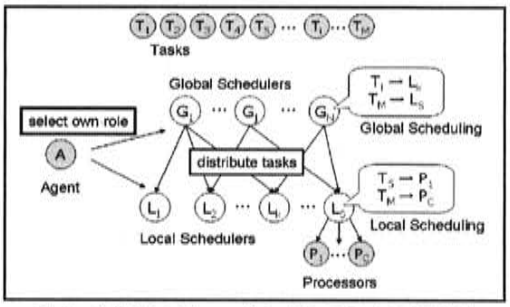


図 2 提案スケジューリングシステムの概要図

3.2.1 タスク

処理対象となるタスクを以下の様に定義する。

$$T = \{T_k | 0 \le k < M\} \quad (1)$$

$$\sigma = \{\sigma_l | 0 \le l < M\} \quad (2)$$

$$\delta = \{\delta_d | 0 \le d < M\} \quad (3)$$

各タスク T_k について、そのタスクの仕事量に関するデータ σ と、締め切り δ がある。締め切りは、システム全体での時間軸における時間を示す。

3.2.2 リソース

リソースを以下の様に定義する。

$$R = \{P_a | 0 \le a < P\} \quad (4)$$

$$\rho = \{\rho_b | 0 \le b < P\} \quad (5)$$

リソース R はそれぞれ複数個のプロセッサ{P_a}とそれらの処理能力に関する情報 ρ を持つとする。

3.2.3 タスク実行時間の見積り

グリッドコンピューティングのシミュレーションを行う場合、あるパフォーマンスを持つプロセッサがある仕事量を持ったタスクを実行する為にどれだけの時間が必要なのかを見積もる必要がある。タスク完了時間の見積りに関して多くの研究があるが、ここでは、Cao らの論文[7]の実験結果の中の一部を利用した見積り関数を設定する。この関数は、7種類のタスクに関して、あるプロセッサの処理能力[MHz]に対するタスクの実行時間[s]を返す。

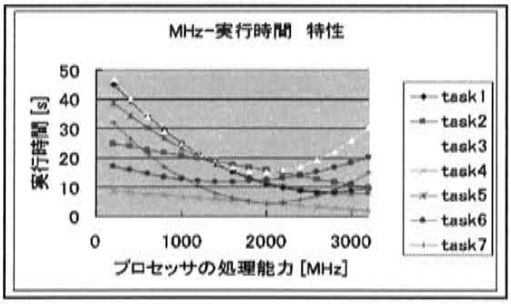


図 3 タスクの実行時間見積り表

3.2.4 ローカルスケジューラ

ローカルスケジューラの定義を以下に示す。

$$L = \{L_m | 0 \le m < S\} \quad (6)$$

$$LS = \{LS_n | 0 \le n < S\} \quad (7)$$

各ローカルスケジューラ L_m は、リソース R_j を持ち、自身の持つリソースの各プロセッサに対するタスク割り当てのスケジューリングを行う。そのスケジュールを LS_n とする。このスケジューラの目的は、自身のコンピュータリソースを出来るだけ節約する事である。即ち、タスクの最大完了時間 makespan を最小化する事ができるスケジュールを得る事が目的であると言える。

$$makespan_m = \max_{0 \le n < S}(\eta_n) \quad (8)$$

$$\eta_n = \sum_{l=1}^{|T|} t(\rho_n, \sigma_l) \quad (9)$$

最大完了時間 $makespan_m$ はあるローカルスケジューラ m において、割り当てられたタスク T を全て実行する為に必要とされる時間である。あるプロセッサ n における割り当てられたタスクの完了時間 η_n は、各タスクについて、各タスクの実行時間 t の合計値である。

本稿では、ローカルスケジューラのスケジューリングアルゴリズムとして、Cao らの研究と同様のものと同様の遺伝子表現を用いた遺伝的アルゴリズムを適用する。図 4 はローカルスケジューラのタスクスケジュールを図に示した遺伝子型で表現している。ローカルスケジューラにおけるスケジュールの遺伝子表現は、タスクの実行順番を表す Ordering part とプロセッサとタスクの対応関係を表す Mapping Part の二つから成り立っている。Ordering part の遺伝子座はタスクの番号を示し、Mapping Part の遺伝子座はそのタスクをどのプロセッサに割り当てるかを 2 進値の数列で表現している。図に示した例を挙げると、タスクは 1、3、2、・・・の順番で実行され、タスク 1 はプロセッサ 1 と 3 に分割して割り当てられ、タスク 3 はプロセッサ 2 に対して割り当てられるということを示している。図の右側のグラフはスケジュールの表現型を示している。割り当てられたタスクは、プロセッサのアイドル状態を短くするように実行される。これらの遺伝子の評価関数としては、式(8)に示すタスクの最大完了時間を用いている。

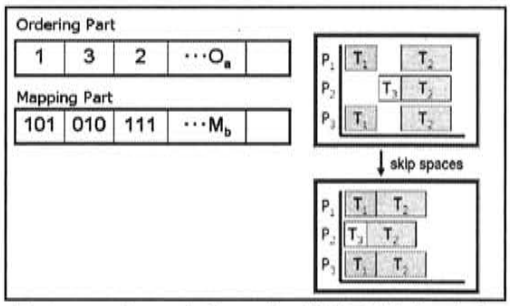


図 4 ローカルスケジューラの遺伝子型とその表現

3.2.5 グローバルスケジューラ

グローバルスケジューラを以下の様に定義する。

$$G = \{G_i | 0 \le i < N\} \quad (10)$$

$$GS = \{GS_j | 0 \le j < N\} \quad (11)$$

グローバルスケジューラ G_i は、それぞれグローバルスケジュール GS_j を持つ。即ち、処理候補のタスク群からどのタスクをどのローカルスケジューラに割り当てるべきか、というスケジュールを持つ。このエージェントの目的は、自身の持つスケジュールの効率を最大化する事である。即ち、各ローカルスケジューラに割り当てたタスクの最大完了時間を最小にする事がこのエージェントの目的である。ここでは、各最大完了時間の平均値をグローバルスケジューラの効用とした。即ち、以下の式を遺伝子の評価関数として用いている。式における M'は、ローカルスケジューラの数を表す。

$$eRevenue(G_i) = \frac{1}{\frac{\sum_m makespan_m}{M'}} \quad (12)$$

また、スケジューリングアルゴリズムとして、遺伝的アルゴリズムを適用する。図 5 はグローバルスケジューラのスケジュールの遺伝子型とその表現型の例を示

している。遺伝子座のインデックスはタスクの番号を示し、遺伝子座の値は割り当てるプロセッサの番号を示している。即ち、下記の例では、タスク T_1 をローカルスケジューラ L_1 に、タスク T_2 をローカルスケジューラ L_2 に、タスク T_3 をローカルスケジューラ L_1 に、というように割り当てが行われている。

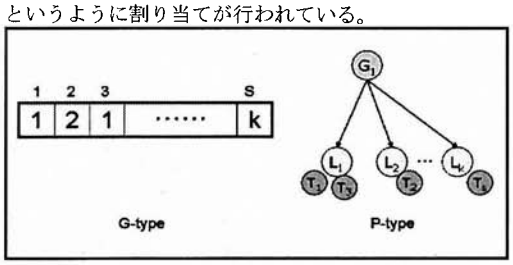


図 5　グローバルスケジューラの遺伝子表現

3.2.6　エージェントの役割選択の詳細

エージェントの役割選択は、全ての役割の組み合わせの中から、システムの期待利得が最大になる組み合わせを選択する事で行われる。即ち、N体のエージェントがそれぞれ選択できる役割が NR 個ある場合、$(NR)^N$ 通りの状況が存在する。それぞれの状況について評価を行い、最も評価の高い状況の時の役割をそれぞれのエージェントが選択する。

本稿では、簡略化の為に、複数のエージェントがグローバルスケジューラになる事はないと仮定している。また、全てのローカルスケジューラの最大完了時間の平均値の逆数である。ここで用いられている値には、仮想的にスケジューリングを実行した結果を用いている。

以下の図では、エージェント3体が役割選択を行う例を示している。この例では、全てのエージェントがローカルスケジューラの役割を選択した状況と、一体のエージェントがグローバルスケジューラになった3通りの状況の評価を行い、最も評価の高い状況の時の役割を選択する。例では、エージェント2がグローバルスケジューラになった状況が最も評価が高く、その役割を用いて、スケジューリングが実行されたことを示している。

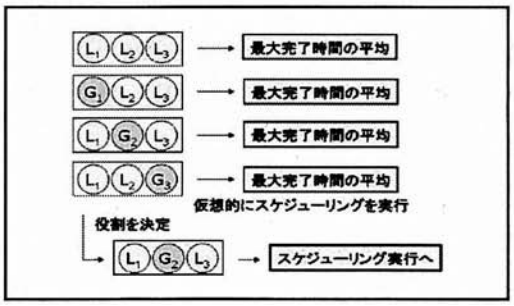


図 6　N=3 の場合の役割選択の例

3.2.7　シミュレーションプロセス概要

以下に、提案手法を用いたスケジューリングシステムの動作例を示す。この例では、3 体のエージェントによるスケジューリングシステムがあり、そのシステムに対して、タスク群が時間間隔をあけて 2 回入力された場合を示している。システムは、タスク群が入力されると、まず、役割選択を行いスケジューリングシステムの構成の決定を行う。その後、グローバルスケジューラによるスケジューリングを経て、各ローカルエージェントが受け取ったタスクが処理される。

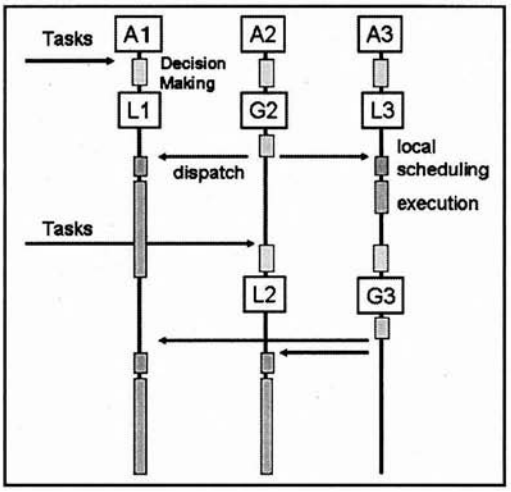


図 7　提案システムの動作の一例

4.　シミュレーション実験と結果

今回のシミュレーションでは、同じタスク（図 3 中の task1）３０個を、t=0 の時にシステムに入力し、その際の処理時間やタスク分配状況について出力を行う。システムは能力の異なる 4 体のエージェントから構成されている。４体のエージェントの能力の詳細については、以下の表に示す。これらのエージェントがどの様に役割を選択し、スケジューリングを行うかを観察・分析し、その有効性の検討を行う。

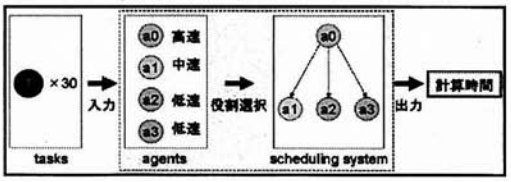


図 8　実験の流れ

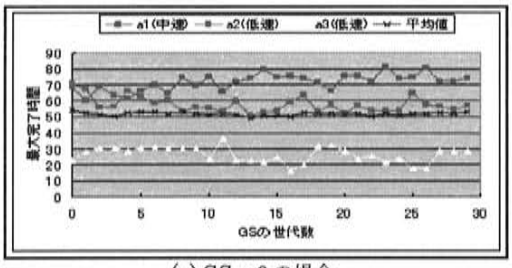
表 1　スケジューラの設定

	a0	a1	a2	a3
プロセッサ数	5	5	5	5
処理速度	高速	中速	低速	低速

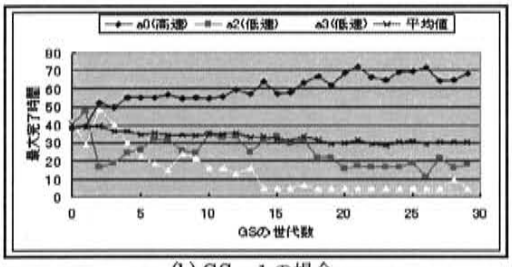
4.1 シミュレーション結果

本稿では、実験結果として、グローバルスケジューラにおけるＧＡの各世代における、ローカルスケジューラの性能の変化を示す。図 9 は、役割選択の時に、仮想的に実行されるスケジューリングの結果である。横軸はグローバルスケジューラのスケジューリングにおけるＧＡの世代を表し、縦軸は各ローカルスケジューラの makespan の値と、それらの平均値を示している。これらの結果には、最も速いローカルスケジューラの makespan が増大する傾向にあり、その他のローカルスケジューラの makespan が減少する傾向がある。また、平均値が漸減している事から、システム全体としてのスケジューリングの効率性としては上昇している事が推測できる。

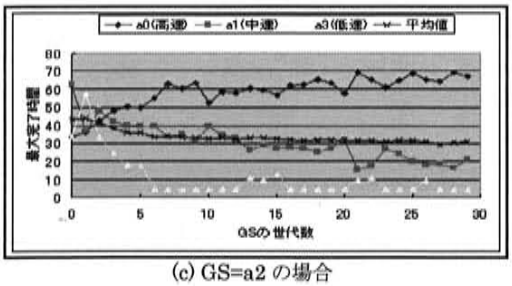
各状況を比較して見ると、グローバルスケジューラが a0 である状況以外の方が最大完了時間の平均値が短く、約 30 秒に収束していることがわかる。この違いは、タスクの処理に参加しているスケジューラの性能の違いによるものだと推測される。エージェントの役割選択では、各状況における最大完了時間の平均値が最も短い状況を選択する。



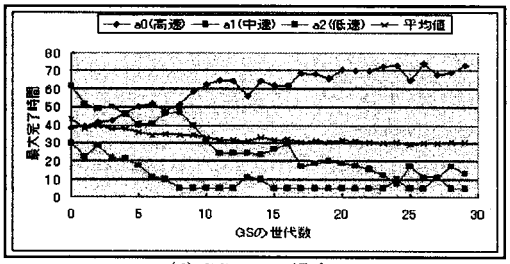
(a) GS=a0 の場合



(b) GS=a1 の場合



(c) GS=a2 の場合



(d) GS=a3 の場合

図 9　ＧＳにおけるＧＡの各世代でのＬＳの最大完了時間の変化

4.2 役割選択の詳細

ここでは、システムを構成するエージェントのパフォーマンスを変更した場合、どのように役割選択の結果が変化するかを示す。今回は、最も高速なエージェントの処理能力を、4/4、3/4、2/4、1/4 と減少させた場合の結果を以下の図と表に示す。4/4=1 における４点は、図 9(a)～(d)における最終世代での最大完了時間の平均値に相当する。その値を、ａ０の処理能力の値を減少させた状態で測定した結果と役割選択の結果を比較して見ると、どの状態においても最大完了時間が最も短い状態を選択していることがわかる。この実験では、エージェントの処理能力が動的に変化する状況を擬似的に再現しており、状況の変化によって適切なシステム構成を採用できていることを示す事ができたと考えられる。表 2 では、色違いの欄によってその状況でグローバルスケジューラになったエージェントを表現している。

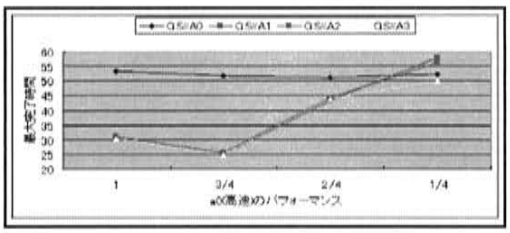


図 10　状況による最大完了時間の平均値の推移

表 2　最大完了時間の平均値と役割選択の結果

A0（高速）のパフォーマンス	GS=A0	GS=A1	GS=A2	GS=A3
4/4	53.28	30.51	31.13	30.46
3/4	51.81	25.04	25.38	24.99
2/4	51.28	43.58	44.26	43.92
1/4	52.20	57.97	56.48	50.19

4.3 タスクの分配

最大完了時間の変化に対するタスクの分配状況の変化を以下の図に示す。この結果は、低速のエージェント a2 が GS になったときにものである。この結果を見てみると、速いスケジューラに対しては多くのタスクを、遅いスケジューラに対しては少ないタスクを割り当てるように進化している事がわかる。この事から、グローバルスケジューラが、処理能力の高いローカルスケジューラに積極的にタスクを分配するアルゴリズムを獲得した事がわかる。

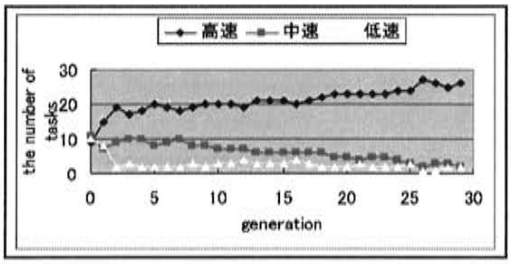


図 11 GSにおけるGAの各世代でのタスク分配の変化（GS=a2）の場合

4.4 まとめ

以上の実験によって、スケジューリングシステム自体の性能と、役割選択によって適切なスケジューリング構成が選択されている事が確認できた。

5. おわりに

本研究では、動的な共有資源問題に対して社会性エージェントアプローチを適用し、その有効性の検証を行った。スケジューリングシステムの性能と、役割選択が状況に応じて適切に行われている事を確認できた。今後は、動的な状況におけるシミュレーションを行う事、システムを構成するエージェントが動的に変化する場合におけるロバスト性の検証などを行っていく。

参考文献

[1]川村 秀憲, 車谷 浩一, 大内 東, "テーマパーク問題の提案と調整アルゴリズムの検討 ～ユビキタス環境における群ユーザ支援の実現へ向けて～", 情報処理学会研究会報告, Vol. 2003, No. 39, 2003-UBI-1, pp. 57-64 (2003)

[2]Junwei Cao, Daniel P. Spooner, etc, "Agent-Based Grid Load-Balancing using Performance-Driven Task Scheduling", International Parallel and Distributed Processing, p.49b

[3]山下 倫央, 鈴木 恵二, 大内 東, "Iterated Multiple Lake Game における社会的ジレンマに対するプレイヤー群の挙動に関する考察", 計測自動制御学会論文集, Vol. 36, No. 2, pp. 195-203 (2000)

[4]Keiji Suzuki, "Dynamics of Autonomous Changing Roles in social Dilemma Games", The IJCIA-03 Workshop on Multiagent for Mass User Support, pp. 19-25, August 2003

[5]Kenta Oomiya, Keiji Miyanishi, Keiji Suzuki, "Evolutionary Effects of Meta-agent Approach in the Tragedy of the Commons", Artificial Computational Economics & Social Simulation 2005, CD-ROM にて掲載, 函館, 2005 年 12 月 17-18

[6]日本アイ・ビー・エムシステムズ・エンジニアリング株式会社, "グリッドコンピューティングとは何か Globus Toolkit ではじめるグリッドの基礎", ソフトバンクパブリッシング, 2004 年 5 月

[7]Junwei Cao, Daniel P. Spooner, Stephen A. Jarvis, Subhash Saini and Graham R. Nudd, "Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling", In Proceedings of 17th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2003), Nice, France, April 2003.