グリッドタスクスケジューリングアルゴリズム R3Q の 不均一な中粒度タスクへの適用*

大磯 正嗣^{*1}, 松村 嘉之^{*2}, 保田 俊行^{*1}, 大倉 和博^{*1}

Application of R3Q for Medium-Grained Task with Nonuniform Computational Granularity

Masashi OISO^{*1}, Yoshiyuki MATSUMURA, Toshiyuki YASUDA and Kazuhiro OHKURA

^{*1} Graduate School of Engineering, Hiroshima University
1-4-1, Kagamiyama, Higahishirosima, Hiroshima, 739-8527, Japan

Evolutionary Computation (EC) has been recognized to be effective to various optimization problems. However, EC needs a large computational resource to calculate all the solution candidates for all the generations. To provide a required computational resource, grid computing can be a good approach for EC in many applications. In this paper, a computational method of how to implement Evolution Strategies (ES), which is a kind of EC, in a grid computing environment is discussed. In particular, we focus on the latency caused by distributing ES tasks using task scheduling algorithms. Conventionally, list scheduling with round-robin order replication (RR) is adopted to reduce waiting time due to synchronization for grid computing. However, in many cases of EC, RR does not show good performance due to the fact that many real world problems, e. g., the problems that belong to Evolutionary Robotics (ER), are medium-grained tasks, where RR does not work well. Therefore, in this paper, extended RR, round-robin replication remote work queue (R3Q) method is adopted to reduce both the communication overhead and waiting time due to synchronization. Our results show that R3Q can reduce the delay time, which is the sum of synchronous waiting time and communication time, compared to the other two methods. In addition, it is also shown that R3Q is effective to the problems in which the computation time of each task can be variable.

Key Words : Grid Computing, Evolutionary Computation, Task Scheduling, Evolution Strategies, Evolutionary Robotics

1. は じ め に

進化計算は、確率的な手法でしか実質解けないようなクラスの工学的最適化問題に対しても、実行可能な時間内に有用な結果を精度よく探索し、その工学的適用範囲は多岐にわたっている。しかし、適用する工学的問題が複雑で大規模になるほど進化計算は解の探索に膨大な時間を要し、この問題点は最適化の際の解候補の評価に要する時間が長くなるほどより顕著になる。この状況に対し、1 台の高性能なスーパーコンピュータによって高速な処理を行う方法から、近年では、進化計算手法が持つ本来の並列・分散性に注目して、複数の計算機をネットワークで接続したクラウドコンピューティングの様な分散処理を行う方法が用いられるに至っている。特に、科学技術計算では、並列・分散性を活かした安価な GPU カードを計算機に搭載して、画像処理だけでなく、最適化問題の様な汎用計算へ使用されるに至り、マルチコア CPU など、並列・分散処理はあらゆる段階で実施されている。

一般的に、並列・分散計算機システムにおいてクラスタは均質な性能の計算機で構成されるシステムであり、高速な新しい計算機を導入するためにはシステム全体を新たに構成し直す必要がある。これに対し、グリッド⁽¹⁾は地理的に離れた複数機関が所有する場合などの様に、非均質な性能の計算機を扱う前提であるため、新しい計算機を容易に追加することが出来る。特に本稿では、オフィスや家庭などに存在する遊休資源が増加していることを鑑

* 原稿受付 2011 年 9 月 22 日

^{*1} 正員, 広島大学大学院工学研究科 (〒739-8527 広島県東広島市鏡山 1-4-1)

^{*2} 信州大学繊維学部 (〒386-0018 長野県上田市常田 3-15-1)

E-mail: kohkura@hiroshima-u.ac.jp

み、その資源をも有効に取り込んで利用しようとするデスクトップ・グリッド⁽²⁾⁽³⁾に注目する。デスクトップ・グリッドは各計算機資源の性能や OS やネットワーク構成といったスペックやプラットフォームが異なるヘテロな環境であり、かつ、他ユーザの存在やネットワーク回線の混雑具合の影響により環境が動的に変化する特徴を持つ。このようなデスクトップ・グリッドに進化計算で一般的に用いられている各世代で全個体を用いた選択を行う離散世代モデルを実装するには、世代交代を行うために計算機の同期処理が必要となる。そのため、世代交代の際に低速計算機がボトルネックとなることがある。さらに、進化計算で取り扱う問題が進化ロボティクス問題や蛋白質立体構造解析問題などの場合、広域ネットワーク上の通信遅延時間より大きい計算時間を要するが、通信遅延時間を無視出来ない程度の計算量のタスク (中粒度タスク) となる。

そこで、進化計算の離散世代モデルをグリッドに実装するにあたり、島モデル型にすることにより通信回数を減らす⁽⁴⁾⁽⁵⁾⁽⁶⁾、あるいは、集団の一部で世代交代を行う連続世代モデルを非同期型に改良する⁽⁷⁾⁽⁸⁾⁽⁹⁾といった方法が研究されている。しかし、島モデル型で並列数が島数分になる場合、離散世代モデルの集団を並列にすることよりは高い並列性が得られない。また、連続世代モデルを用いる場合、世代交代モデルの違いにより進化ダイナミクスが変化するため、no-free-lunch theorem⁽¹⁰⁾で指摘されているように、問題によっては解の探索に悪影響が出る場合がある。そこで、本研究では、離散世代モデルの進化計算を実装する場合に、個体評価をグリッドに効率よく並列・分散計算させることをグリッドタスクスケジューリングの立場から対処を考える。特に、デスクトップ・グリッドにおいてタスクスケジューリングを行う場合、一般に環境が大きく変動するため、状況に応じて動的にスケジューリングを行うことが求められる。また、他ユーザによる計算資源の使用状況や、ネットワーク回線の通信速度等の予測精度を向上させるのは困難が伴う。そこで本稿では、非予測型動的スケジューリング Round-robin Replication Remote Work Queue (R3Q)⁽¹¹⁾に着目する。

これまで、R3Q の評価は一定の計算粒度を持つタスク⁽¹¹⁾を用いて行われてきた。一方、工学的実問題のタスクの計算粒度は不均一であったり、例えば、正規分布に従わない様な不規則な分布を持ったりといったような特徴を持つこともある。そのため、工学的実問題の特性の一部側面を皮切りに、さらには複数側面への対応への拡張を鑑みて、計算粒度の様々な特性に対するグリッドスケジューリングの有効性を検証して検証結果を積み上げていくことが肝要である。そこで、本稿では、特に、工学的実問題として我々が着目している進化ロボティクス (Evolutionary Robotics: ER)⁽¹²⁾問題の取り扱いを念頭に、その標準的なテストケースとして、評価関数と LU 分解の行列変化による計算粒度一様分布問題によって、不均一な計算粒度を持つ問題に対するグリッドスケジューリングの性能を定量評価する。さらに、実際の ER 問題におけるグリッドスケジューリングの性能を実例に基づいて検証する。ここで、LU 分解は問題サイズを変更することで計算量を容易に調節できるため、タスク毎に一定の範囲で一様に分布するように異なる問題サイズ (行列サイズ) を与えることによって計算粒度をある範囲で一様に分布させることも可能となる。また、ER 問題は進化計算による一定回数の評価を必要とし、進化によって獲得したコントローラ毎に計算量が変化するため、各評価において正規分布とは異なる計算時間を持っている (図 1)。

本稿は以下の通り構成される。第 2 章では、グリッドスケジューリングアルゴリズムの概要、および本稿で扱う非予測型動的スケジューリングアルゴリズムの詳細について述べる。第 3 章では、ベンチマークおよび実問題を用いた計算機実験の結果・考察を示す。最後に、第 4 章で結論を述べる。

2. グリッドタスクスケジューリングアルゴリズム

2.1 非予測型動的スケジューリング

一般に、タスクスケジューリングアルゴリズムには静的、動的スケジューリングの 2 通りの方法がある⁽¹³⁾。静的スケジューリングは、スケジューリングに関する決定を実行前に行うタスクスケジューリングである。一方、動的スケジューリングでは、一部もしくは全てのスケジューリングをタスク実行中に決定する。グリッド環境内では、グリッド用に割かれるリソースの割合に応じて処理速度が連続的に変動するため動的スケジューリングがより適している。

動的スケジューリングアルゴリズムは予測型と非予測型に分類される。予測型スケジューリングは計算資源の予測情報を用い、入手可能な計算機資源の情報が正確な場合に効果的なスケジューリングを行うことが知られているが、予測誤差が大きくなると性能が著しく悪化する。また、ER 問題等の実問題では、問題の終了条件等の影

Fig. 1 A typical example of the distributions of calculation time for individuals of the MRS problems dealt in this paper.

響によりタスク毎に計算量が大きく変化し、実際に計算処理を行うまで計算時間を知ることが出来ない。そのため、各タスクの計算時間を予測し、スケジューリングを行うことは事実上不可能である。以上の理由から、デスクトップ・グリッドにおけるタスクスケジューリングでは、非予測型動的スケジューリングアルゴリズムが適していると考えられる。

非予測型動的アルゴリズムの代表として、Work Queue (WQ)⁽¹⁴⁾, list scheduling with Round-robin order Replication (RR)⁽¹⁶⁾がある。WQは均質な並列計算機のために開発された古典的なアルゴリズムである。WQはアイドル状態のプロセッサにタスクを割り当て、計算が終わった計算機から結果を受け取り、新しいタスクをそこに割り当てる。WQはタスクに優先順位を任意に設け、処理を中断することが出来る。WQは、処理の速いプロセッサには、遅いプロセッサよりも多くのタスクを割り当てるという考え方に基づく。

一般に、WQは低速機がボトルネックとなり同期待ち時間を生じるが、RRはWQを改良しタスクの複製を導入することで同期待ち時間の問題に対処している。例えば、図2のように複製されたタスク ($\bar{V}_{10}$) が先に割り当てられたオリジナルタスク (V_{10}) より早く終了する可能性がある。よって、高速なプロセッサ (図2の M_1) が使用されている場合にRRはWQの同期待ち時間を短縮することが出来る。RRはグリッドスケジューリングとして用いられるものの中で性能保証を持つ最初の近似アルゴリズムとして提案されている。

2.2 グリッドタスクスケジューリングアルゴリズム RR

グリッド、または他の並列計算機におけるタスクスケジューリング問題において、最も一般的な評価指標はメイクスパンである。しかしながら、グリッドでは非最適なスケジューリングのメイクスパンは最適なスケジューリングを行ったものより非常に大きくなる。これは、グリッド環境から得られる計算機資源が常に変化するためである。例として、最小実行時間 (理想的並列実行時間の理論値 OPT) を持つ最適アルゴリズムを考える。グリッドの計算量が時間 OPTの前に減少し、その状態が続いた場合、メイクスパンはOPTからかけ離れたものとなる。そのため、スケジューリングの評価にメイクスパンを用いる場合、一般に近似アルゴリズムは存在しない。このため、新しい評価基準としてTPCCが提案された⁽¹⁶⁾。RRはタスク粒度一定の粗粒度パラメータスウィープ型アプリケーションのスケジューリングにおいて最小のTotal Processor Cycle Consumption (TPCC) を持つ、 $(1 + [m(\log_e(m-1)+1)]/n)$ -近似アルゴリズムである。ここで、 n はタスク数、 m はプロセッサ数である。TPCCは粗粒度パラメータスウィープ型アプリケーションで消費される総計算量を表す。

RRでは、未処理のタスク数が計算機の台数より少なくなると、タスクリング (Ring of Tasks) を用いて実行中のタスクを管理する。オリジナルタスクの処理を終え、アイドル状態になるプロセッサが生じた場合、タスクリングからラウンドロビン順にタスクを選び、そのタスクの複製をプロセッサへ送信する。また、複製されたタスクが終了した場合、同じタスクを実行中の計算機とタスクリングからタスクを取り除く。そして、アイドル状態となった計算機へ、タスクリングからラウンドロビン順に選ばれたタスクの複製が送信される。なお、スケジューリングアルゴリズムRRの詳細は⁽¹⁵⁾⁽¹⁶⁾を参照されたい。

Fig. 2 Comparison of WQ and RR by task flow.

2.3 グリッドタスクスケジューリングアルゴリズム R3Q

グリッドに ER 問題等の中粒度タスクが実装される場合, RR を含む動的スケジューリングアルゴリズムでは, 通信遅延時間を無視出来ない. これは, タスクの実行時間に対する通信時間の割合が比較的大きいため, スレイブが頻繁にアイドル状態となることに起因する (図 3 (a)). この遅延を隠蔽するため, スレイブのプロセッサへ実行タスクの他に次のタスクを待機させるスライド転送方式⁽⁸⁾が提案されている (図 3 (b)). しかし, 離散世代モデルの場合, 各プロセッサに対して複数のタスクを割り当てることによって低速計算機のボトルネックがより顕著となり, 同期待ち時間が増大することが予想される. そこで本稿では, 通信遅延時間を隠蔽しつつ, RR を適用することにより同期待ち時間へ対処する R3Q⁽¹¹⁾ (図 4 参照) に着目する.

R3Q ではスライド転送方式と同様に複数のタスクをプロセッサへ割り当てるが, RR におけるタスクの強制終了を簡略化するため, プロセッサ毎にタスク待機用のキューを設ける (図 4 の Slave 参照). この方法を Remote Work Queue (RWQ)⁽¹¹⁾と呼び, マスタ内のスレッド数はプロセッサ数と等しい. これにより, マスタからスレイブの実行タスクを管理でき, 強制終了も容易に行える.

R3Q のアルゴリズムは以下の通りである.

1. 全てのタスクをマスタのキュー Q に置く.
2. スレイブを $k \in \{1, 2, \dots, M\}$ とし, スレイブ k はそのプロセッサ数 (r) に等しい実行スレッドを開始する (本稿では, シングルプロセッサ ($r = 1$) を扱う).
3. マスタは $2r$ 個のタスクをスレイブ k に転送する. スレイブ k はタスクをキュー $Q_{k,r}$ へ挿入する.
4. スレイブの実行スレッドはキュー $Q_{k,r}$ の先頭のタスクを実行し, 結果がマスタへ返信される.
5. マスタは Q の先頭のタスクをスレイブ k のキュー $Q_{k,r}$ へ溜める.
6. マスタの Q が空になるまでステップ (4) (5) の過程を繰り返す.

R3Q では Q が空になった時, $2M$ 個 ($r = 1$ の場合) のタスクが実行中である. ここで, R3Q は RR と同様にタスクリングがこの $2M$ 個のタスクを保持・管理する. R3Q は, 以下の過程を $2M$ 個のタスクが全て完了するまで繰り返す. あるタスク v のタスクインスタンスがプロセッサ p で完了した場合, R3Q はその結果を受け取り, タスクリングと全てのスレイブのキュー $Q_{k,r}$ から v を取り除く. そして, プロセッサ p および v を取り除いたプロセッサへ, タスクリングからラウンドロビン順にタスクを 1 つ選び, 複製する.

3. 計 算 機 実 験

本稿では, あらゆるグローバルなデスクトップ・グリッドを想定したシミュレータを構築することは実質困難であるため, 実在の計算機環境を構築し, 実例による実証の積み上げ方式による検証を行っていく観点に立脚する. そのため, 標準的な計算粒度一様分布問題による定性評価を行い, その後に, 工学的実問題の例題を用いた計算

Fig. 3 CPU utilization with WQ, Slide Transfer and R3Q.

Fig. 4 R3Q Grid System using RR along with remote work queue technique.

機実験を行って、これら両方に対するスケジューリングアルゴリズム R3Q による同期待ち時間の減少効果、および通信時間の隠蔽効果について評価し、実証例を積み上げていく。

3.1 実験設定

本実験では (30,200) - FES⁽¹⁷⁾ で 100 世代進化を行う。すなわち、1 試行の間にスケジューラは複製タスクを含めて 20,000 回以上の通信を行う。この間、総実行時間、計算時間、同期待ち時間、通信時間を計測し、R3Q が RR, WQ に対して同期待ち時間を削減し、また、通信遅延時間を隠蔽できるかを検証する。本稿では、各試行の開始時刻から終了時刻までの時間を総実行時間とし、スレイブ上でタスクの処理を開始した時刻から、そのタスクの結果を発信するまでの時間を 1 タスクの計算時間とする。WQ では、マスタのキュー Q が空になってから全てのタスクの結果が得られるまでの時間を同期待ち時間とする。R3Q と RR では、結果の得られていないタスクの数が $M-1$ 個となってから、全てのタスクの結果が得られるまでの時間を同期待ち時間とする。通信時間は、各スレイブがアイドル状態となった時間の平均値とする。なお、実験は独立に 10 回試行し、その平均時間により評価を行う。

表 1 に本実験で用いる計算機の性能と台数、図 5 に本実験環境で用いるネットワークを示す。表 1 の計算機 M は信州大学内にあるマスタ、計算機 L1-L4 は信州大学内にあるスレイブ (Local Slave) である。計算機 WR はグローバル IP を持つ広島大学のルータである。計算機 W1-W7 は広島大学内のスレイブ (WAN Slave) であり、グローバル IP を持たず、マスタ M と直接通信することが出来ない。これらの計算機はルータ WR を経由してタスクの受け渡しを行う。広島大学 (Hiroshima) の計算機は全てと信州大学 (Shinshu) のルータ LR につながる一部は 1Gbps のスイッチングハブであり、信州大学内のネットワークはほとんど 100Mbps のスイッチングハブを介して接続されており、両大学間はインターネットにより接続されている。なお、本実験では、スケジューリングアルゴリズムの定量的比較検証を容易にするため、両大学内の計算機およびネットワークは他ユーザの影響を排除した占有使用状態とする。このことにより、本実験環境はインターネット通信時間以外の外乱を極力抑えているため、計算実行時間は実験環境での最大性能をおおよそ示していることになる。

Table 1 Specifications of computers.

No.	CPU	memory (MB)	Num
M	Xeon 2.4 GHz	512	-
LR	Xeon 2.4 GHz	512	-
L1	Xeon 2.4 GHz	512	1
L2	Xeon 3.0 GHz	512	2
L3	CeleronD 3.2 GHz	512	1
L4	Athlon 2.0 GHz	1024	1
WR	Xeon 2.4 GHz	512	-
W1	CeleronD 3.2 GHz	512	2
W2	PentiumD 2.8 GHz	512	1
W3	PentiumD 3.0 GHz	512	1
W4	Athlon MP 1800+	512	3
W5	Athlon XP 2700+	512	1
W6	Athlon 64 3500+	512	2
W7	Opteron 2.0 GHz	512	10
Total		-	25

Fig. 5 The networks of computers in the global desktop grid environment.

本稿では、通信プロトコルとして SOAP (simple object access protocol) を用いる。マスタは JAVA のマルチスレッド構造を用いてスレッドを生成し、GLUE SOAP を用いてスレイブヘデータを転送する。スレイブはマスタからのメッセージを Tomcat サーバと Axis から受け取り、タスクを実行する。スレイブは実行結果を Axis SOAP によって返信する。

3.2 実験 I: LU 分解の行列変化による計算粒度一様分布問題

LU 分解の計算処理は、対象とする行列の次元数を変化させることによって計算量を任意に調整できる。本稿では、この特徴を利用して、グリッドタスクスケジューリングのタスクの計算粒度が不均一に分布する問題に対する基本的な特性を明らかにするため、行列の大きさを一様に変化させることによってタスクの計算粒度を一様に変化させる。これにより、タスクの計算粒度が一様分布する様な問題に対するスケジューリングアルゴリズムの定量評価を行う。

3.2.1 実験設定

予備実験よりマスタ - WAN Slave 間の通信時間が 150 (ms) 程度であったことから、中粒度タスクの定義を基に、本実験で使用するタスクの計算時間の範囲を 200-500 (ms), 200-1000 (ms), 500-1000 (ms), 200-1500 (ms), 500-1500

Table 2 The execution time of WQ, RR and R3Q for each computational granularity of LU decomposition.

Granularity (ms)	Model	Synchronization		Communication		Calculation		Total	
		time (sec)	stddev	time (sec)	stddev	time (sec)	stddev	time (sec)	stddev
200-500	R3Q	67.5	0.7	110.4	4.6	1732.2	7.0	1910.1	8.2
	RR	54.3	1.3	321.2	4.0	1866.3	3.9	2241.8	4.2
	WQ	66.7	1.6	322.8	2.8	1898.6	5.6	2288.1	7.1
200-1000	R3Q	121.5	4.3	63.8	3.6	2740.7	10.1	2926.0	14.4
	RR	97.8	1.5	338.1	8.6	2955.7	11.3	3391.6	18.8
	WQ	118.9	4.5	329.9	18.6	3022.3	9.4	3482.2	20.5
500-1000	R3Q	132.9	3.1	29.7	1.8	3799.2	4.8	3961.8	5.7
	RR	110.0	2.4	347.7	2.6	4057.9	4.4	4515.6	5.9
	WQ	136.3	2.6	349.8	2.6	4172.1	3.6	4658.1	8.0
200-1500	R3Q	177.6	3.3	46.8	1.4	3516.9	13.1	3741.4	13.0
	RR	136.4	3.8	336.4	4.1	3751.9	12.4	4224.8	11.8
	WQ	170.8	5.5	335.4	3.8	3860.3	13.0	4265.6	15.3
500-1500	R3Q	182.4	7.5	23.9	1.6	4812.1	4.3	5018.8	8.8
	RR	143.2	3.4	356.6	4.8	5067.5	5.5	5567.3	8.1
	WQ	177.3	5.8	355.6	4.1	5203.3	4.8	5736.2	10.5
1000-1500	R3Q	219.9	4.6	17.6	1.7	6406.4	5.1	6644.0	8.2
	RR	176.9	6.4	369.9	7.3	6646.7	8.8	7193.5	16.6
	WQ	222.4	4.6	368.7	6.4	6837.8	8.2	7428.9	16.3

(ms), 1000-1500 (ms) の 6 通りとする。本実験では、各タスクで処理する行列の次元数を一定範囲の正規乱数により与える。この次元数の範囲は、マスタ M の計算機における計算時間を基準として、上記の計算時間の範囲を満たすように定める。以上より、次元数の範囲を 342-473, 342-603, 473-603, 342-686, 473-603, 603-686 とする。これに従い、各計算粒度の範囲において、一回の試行で必要となる 20,000 個のタスクを生成する。また、3・3 章における ER 問題との比較のため、各タスクにおいて転送されるデータ量は本稿で取り扱う ER 問題と同量の約 2KB とする。

3・2・2 実験結果

表 2 に LU 分解のタスクの各計算粒度における R3Q, RR, WQ の実行時間を示す。最も基本的なスケジューリングアルゴリズムである WQ に対して、RR は 2.0~3.6%, R3Q は 7.3~18.6% 実行時間を短縮しており、計算粒度が正規乱数によって変化するベンチマーク問題において R3Q スケジューリングによる実行時間短縮効果が示された。

さらに 2 章に挙げたスケジューリングの有効性を検証するため、計算グリッドを用いた並列処理によって生じる遅延である同期待ち時間および通信時間の和を遅延時間と定義し、図 6 に LU 分解のタスクの各計算粒度における R3Q, RR, WQ の遅延時間を示す。遅延時間では、RR は WQ に対して 2.9%~7.5%, R3Q は WQ に対して 54.3%~66.6% 減少した。特に通信時間は、R3Q は WQ, RR に対して 65.6%~95.2% と大幅に減少しており、RWQ の効果によってそのほとんどが隠蔽されている。また、これにより WAN Slave でのタスク実行回数が他の 2 手法に比べて増加しており、計算機資源がより有効に利用されている。

一方、同期待ち時間に着目すると、RR は WQ に対して同期待ち時間を 17.7%~20.5% 減少しているのに対し、R3Q では WQ に対して同期待ち時間が 2% 程度増減しておりほぼ同等である。R3Q は、タスクを複数保持するため、単一のタスクを扱う他の 2 モデルと比較すると同期待ち時間の影響を受けやすいが、RR のタスク複製を導入することで、その影響を抑えることに成功している。また、RR と R3Q 共に、タスクの計算粒度の範囲によらず同程度のスケジューリング効率を示しており、不均一かつ不規則な計算粒度によるスケジューリングへの影響は限定的であると考えられる。以上より、本稿で対象としている ER 問題のような中粒度タスクにおいて、ある範囲で正規乱数に従って粒度が変化するベンチマーク問題では、R3Q が最も有効なアルゴリズムであると言える。

Fig. 6 The delay time (both the synchronization and communication time) of WQ, RR and R3Q for each computational granularity of LU decomposition.

3.3 実験 II: ER 問題

タスクの計算粒度が不均一な工学的実問題の一例として、ER 問題の一つである進化型 Multi Robot System 協調荷押し問題 (以下、MRS 問題) を適用し、これに対するスケジューリングアルゴリズム R3Q による同期待ち時間の減少および通信時間の隠蔽効果について検証する。

3.3.1 実験設定

サセックス・アプローチ⁽¹²⁾に基づき、図 7 に示すようにシミュレータを構築した。ロボットはフィールドを動き回り、荷をゴールまで押し運ぶことを目的とする。フィールドには 3 個の荷があり、小型の荷は 2 台以上、中型の荷は 3 台以上、大型の荷は 5 台以上のロボットで同じ方向に押すことによって動かすことが出来る。この問題の 1 回の試行は最大 2000 ステップであり、全ての荷がゴールに運ばれた場合は直ちに終了される。10 台のロボットによる MRS 問題の解の一例を図 9 に示す。

ロボットは全方位カメラと 8 個の IR センサ、2 個のモータホイールを持ち、図 8 のリカレントニューラルネットワーク (RNN) によって制御される。この RNN は 8 個の IR センサ情報 (8 入力)、最も近いロボットの位置 (3 入力)、最も近い荷の位置 (3 入力)、ロボットの絶対方向 (2 入力)、計 16 のセンサ入力、4 個の隠れノード、左右のモータ出力 (2 出力) を持ち、この RNN は全体で 132 箇所のシナプス結合を持つ。なお、この問題では全てのロボットが同一仕様の RNN を各々持つこととする。

本実験において各スレイブで処理するタスクは個体の適応度評価である。従って、FES の各世代において、新たに生成される 200 個体に対応して同数のタスクが生成される。1 タスクの間に MRS 問題による評価が 1 回行われる。実行するステップ数、およびオブジェクト同士の接触の多寡等により計算量が異なるため、1 回のシミュレーションによる計算時間は不均一かつ不規則に分布する。第 1 章の図 1 は、計算機 L2 (表 1) より得られた MRS 問

Fig. 7 Cooperative package-pushing MRS problem environment.

Fig. 8 Recurrent neural networks.

Fig. 9 Cooperative package-pushing behavior.

題の計算時間分布の一例を示している．図 1 のある 1 試行における 1 タスクの計算時間の分布は，平均値は 0.73 秒，標準偏差は 0.14 であり，異なる試行で異なる平均と標準偏差を持っている．

3.3.2 実験結果

表 3 に MRS 問題における R3Q, RR, WQ の実行時間を示す．LU 分解と同様に WQ と比較すると，RR は 1.0%，R3Q は 8.2% 実行時間を短縮しており，実験 I と異なる計算粒度を持つ MRS 問題においてもスケジューリングの有効性が示された．また遅延時間（図 10）の比較においても，RR は WQ に対して 3.6%，R3Q は WQ に対して 71.9% 減少した．LU 分解においてタスクの計算時間が比較的近い計算粒度 500-1500 (ms) の結果と比較すると，計算粒度の分布の違いから同期待ち時間が短縮しているが，負の影響は見られなかった．このため，MAS 問題における計算粒度の分布による R3Q のスケジューリングへの影響は LU 分解と同様に限定的であり，R3Q は少なくとも

Table 3 The execution time of WQ, RR and R3Q for the MRS problem.

Model	Synchronization		Communication		Calculation		Total	
	time (sec)	stddev	time (sec)	stddev	time (sec)	stddev	time (sec)	stddev
R3Q	127.7	9.0	7.3	1.7	5944.5	91.7	6079.5	95.4
RR	115.8	19.0	347.4	5.8	6091.2	83.7	6554.4	95.8
WQ	128.7	28.7	351.6	3.7	6139.7	107.8	6620.0	119.5

Fig. 10 The delay time of WQ, RR and R3Q for the MRS problem.

も ER 問題において同様に機能するものと考えられる。

4. 結 論

本稿では、進化計算の膨大な計算コストに対してデスクトップ・グリッドによる並列分散処理を適用し、グリッド環境における進化計算向けスケジューリングとして非予測型動的スケジューリングアルゴリズムに着目した。特に、中粒度タスクにおいて有効な結果を示す R3Q に着目し、様々な特性の不均一な計算粒度を持つ問題に対するグリッドスケジューリングの定量評価の実証を積み上げていく方針の基、その第一歩として、タスクの計算粒度が一様に分布する問題を取り扱い、基本的な定量評価を実施した。さらに、実際の ER 問題におけるグリッドスケジューリングの性能を一つの実例に基づいて実証した。

計算機実験では、タスクに LU 分解を用いて行列の大きさを一様に分布させ、計算粒度が一様に分布するようにした基本的な問題を採用し、さらに実例に基づく実証として、ER 問題の一例である進化型 MRS 協調荷押し問題を採用し、広島大学および信州大学の計算機で構成されるデスクトップ・グリッド環境において、WQ, RR との比較検証を行った。その結果、R3Q は、WQ に対して実行時間を 7.3%~18.6%、遅延時間を 54.3%~66.6%減少した。これにより、少なくともタスクの計算粒度が一様に分布する問題や、実際の ER 問題の様に不均一に分布するクラスの問題に対して頑健であり、工学的実問題の中粒度タスクにおけるスケジューリングでは、比較に用いた 2 手法より有効であることの一例を示せた。今後、今回の実証で示しきれない範囲のあらゆる不均一な特性を持つ工学的実問題での実証が積み上げられることが期待される。

文 献

- (1) Foster, I., and Kesselman, C., *The grid: blueprint for a new computing infrastructure*, (1998), Morgan Kaufmann Publishers Inc.
- (2) Brevik, J., Nurmi, D., and Wolski, R., "Automatic Methods for Predicting Machine Availability in Desktop Grid and Peer-to-peer Systems", *Fourth IEEE International Symposium on Cluster computing and the Grid (CCGrid'04)*, (2004), pp.190-199.
- (3) Kacsuk, P., Podhorszki, N., and Kiss, T., "Scalable Desktop Grid System", *High performance computing for computational science (VECPAR'06)*, (2006), pp.1-13.

- (4) Castillo, P.A., Arenas, M.G., Castellano, J.G., Merelo, J.J., Rivas, V.M., and Romero, G., "Optimisation of Multilayer Perceptrons Using a Distributed Evolutionary Algorithm with SOAP", *The 7th International Conference on Parallel Problem Solving from Nature (PPSNVII), Lecture Notes in Computer Science*, No.2439 (2002), pp.676-685.
 - (5) Tanimura, Y., Hiroyasu, T., Miki, M., and Aoi, K., "The System for Evolutionary Computing on the Computational Grid", *IASTED 14th International Conference on Parallel and Distributed Computing and Systems*, (2002), pp.39-44.
 - (6) Jing, T., Lim, M.H., and Ong, Y.S., "A Parallel Hybrid GA for Combinatorial Optimization", *Congress on Evolutionary Computing (CEC2003)*, (2003), pp.1895-1902.
 - (7) 染谷博司, "グリッド環境に適した遺伝的アルゴリズムによる最適化", *統計数理*, Vol.52, No.2 (2004), pp.381-391.
 - (8) 小野功, 水口尚亮, 中島直敏, 小野典彦, 中田秀基, 松岡聡, 関口智嗣, 楯真一, "Ninf-1/Ninf-G を用いた NMR 蛋白質立体構造決定のための遺伝的アルゴリズムのグリッド化", *先進的計算基盤システムシンポジウム (SACSIS 2005)*, Vol.5 (2005), pp.143-151.
 - (9) 小野功, "グリッド向け遺伝的アルゴリズムによる大規模最適化", *計測自動制御学会論文集*, Vol.47, No.6 (2008), pp.473-479.
 - (10) Wolpert, D. H. and Macready, W. G., "No Free Lunch Theorems for Optimization", *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1 (1997), pp.67-82.
 - (11) 松浦芳樹, 大倉和博, 松村嘉之, 藤本典幸, 萩原兼一, "R3Q による進化型計算の中粒度 Grid スケジューリング", *情報処理学会論文誌: コンピューティングシステム*, Vol.47, No.SIG 12(ACS 15) (2006), pp.240-249.
 - (12) Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N., "Evolutionary robotics: the Sussex approach", *Robotics and Autonomous Systems*, Vol.20, No.2 (1997), pp.205-224.
 - (13) Dong, F. and Akl, S.G., "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", *Technical Report 2006-504*, (2006), School of Computing, Queen's University.
 - (14) Graham, R. L., "Bounds for Certain Multiprocessing Anomalies", *Bell System Technical Journal*, Vol.45 (1966), pp.1563-1581.
 - (15) Fujimoto, N. and Hagihara, K., "A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks", *SAINT 2004 Workshop on High Performance Grid Computing and Networking*, (2004), pp.674-680.
 - (16) Fujimoto, N. and Hagihara, K., "Near-Optimal Dynamic Task Scheduling of Independent Coarse-Grained Tasks onto a Computational Grid", *Proc. of The 32nd Annual International Conference on Parallel Processing (ICPP-03)*, (2003), pp.391-398.
 - (17) Yao, X. and Liu, Y., "Fast Evolution Strategies", *Control and Cybernetics*, Vol.26, No.3 (1997), pp.467-496.
-