社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ARC－172 (17)
2007－HPC－109 (17)
2007／3／1

コンピュータ資源活用のための ヒューリスティック手法を用いたNQSパラメータの最適化

伊藤 利佳
独立行政法人 理化学研究所 情報基盤センター

理研における PC クラスタ(RSCC)の利用効率向上をテーマとし，スケジューラ(NQS)のパラメータの最適化を行った．手法としては，NQS と同じふるまいをするシミュレータを構築し，数理計画法のひとつである Randomized Local Search を用い，過去の統計情報に基づいてパラメータ設定の最適値の探索を実施した．現行のパラメータ設定による結果と最適化による結果とを比較した結果，サンプリングしたある期間においては 93%から 96%の利用効率の向上が見られた．また，十分大きくしたパラメータ設定と提案するパラメータ設定を比較することによって，パラメータ設定における最適化の重要性を確認することができた．

Job Scheduling Optimization for NQS Parameter Based on Heuristic Method

RIKA ITO
RIKEN (Institute of Physical and Chemical Research)

We propose a way of system usage optimization by using Randomized Local Search for parameter setting of scheduler software, targeting to improve the RSCC system usage efficiency, which is a large PC cluster system equipped at RIKEN. We newly built a sort of simulator based on the specification of Fujitsu NQS actually used here, which enables us to virtually reproduce the system efficiency for the past with various configurations of NQS parameters. In our experiments, we compared the efficiency of current NQS parameter configuration with that of our optimized one. As the result, we saw 3% of improvement from 93% to 96%, as the system usage %.

1. はじめに

一般に，共同研究機関のスーパーコンピュータシステムは，不特定多数のユーザによる共用システムであるという性格上，資源配分の公平性の確保には慎重に取り組まなければならない．個々のユーザの要求は多種多様であり，多くのユーザは使いたい時に十分な資源を使いたいと考えているため，時にそれらは競合を発生するからである．一方，システムの運用管理の立場からは，システムの利用効率の向上に積極的に取り組む必要がある．効率の向上は，機関の研究活動の支援を底上げすることであり，とりもなおさず，研究に対するスーパーコンピュータシステムの貢献の最大化となる．

昨今，国内外を問わず，多くの研究機関において，パーソナルコンピュータ（PC）をネットワーク装置で接続した形態，いわゆる PC クラスタによるシステム構築が進んでいる[2]．PC クラスタの浸透の背景には，様々な理由があるが，主な推進力として，コモディティの利用による価格低下，Linux の出現によるアプリケーションの可搬性の容易化などがあげられる．これらによって，限られた予算の範囲内で求めうる最大性能が特段にスケールアップし，ユーザにとっては，アプリケーションプログラムの移行にかつてほどの労力を割かなくても良くなった．

独立行政法人理化学研究所（理研）においては，スーパーコンピュータシステムとして，約 2000CPU からなる「理研スーパーコンバインドクラスタ（RSCC）」が 2003 年 3 月から本格的に稼動した．かつて，数個から数 10 個の CPU でなりたっていたスーパーコンピュータシステムが，2000 個もの CPU からなるシステムへと変貌を遂げるに伴い，システム資源の利用最適化がより重要なテーマとなってきたことは言うまでもない．

多数の CPU をユーザに割り当てるのに，最も単純な割当方法は，一人のユーザもしくはグループにいくつかの数の CPU を割り当て，占有させる方法である．このような方法では，容易に想像できることであるが，あるユーザあるいはグループが，自分たちに割り当てられたシステム資源容量を超えるような作業をしたい場合，他のグループがその時にシステムを使っていない場合であっても，資源を確保することができないという問題が発生する．これを解決する技術が，ジョブスケジューリングという機能であり，PC クラスタにおいては，一般に，システム管理ソフトウエア群の 1 つの機能として提供され，スケジューラと呼ばれている．商用のスケジューラとしては，PBS Professional

(Portable Batch System*), Platform LSF (Load Sharing Facility**), NQS (Network Queuing System***) などがあり，これらの製品にはそれぞれの特長があるが，現在理研においてはNQSを利用している．

ジョブスケジューリングという機能は，ユーザやグループ単位でスタティックにハードウエア資源を割り当てる代わりに，ユーザのプログラム実行（ジョブ）単位でダイナミックにハードウエア資源の時間を割り当てる機能である．スケジューラはユーザから投入されたジョブをいったん受け付け，優先順位やユーザ毎に同時に実行できるジョブ数など，多数の設定値（パラメータ）をあらかじめ設定したルール（ポリシー）に基づき，ジョブを逐次に実行するというバッチ処理のシステムを構築する．割当可能なシステム資源に余分が無い場合，ユーザのジョブは待ち行列（キュー）に入ることになるが，ジョブスケジューラのポリシーの設定によっては，空き資源があるにもかかわらず，ジョブが実行されずにキューにはいったままになるという状況が発生する．これは PC クラスタの規模が大きくなればなるほどより大きなインパクトとなり，昨今のスーパーコンピュータシステムの CPU 数の爆発的増加傾向を考慮すると，今後ますます有用となるスタディであると考える[3] [8].

2. 研究の背景とアプローチ

ジョブスケジューラのパラメータ設定の最適化は，ユーザのジョブ投入の振る舞いに依存するため，運用されるシステム単位で異なり，また季節によっても異なるため，一概に「こう設定すればよい」というものは存在しない．また，現場の管理者の経験によってパラメータ設定されている場合が多いので，明確な論拠に基づいて最適化されているわけでもない．ジョブスケジューラの最適化における最大の困難さは，最適化する上で必須の条件であるユーザジョブの終了時刻，つまりジョブの走行時間という条件が事前にわからないため，最適解を数学的に求めることができないという点にある．このことは，過去の統計情報からその期間における最適解を求めることはできるが，将来に対する最適なパラメータを事前に求めることはできないということを意味している．従って，現場の管理者が経験に基づいてパラメータを設定することはやむを得ないことであるが，少なくとも，それらを定期的に検証し，検証に基づくパラメータ変更を実施することは有意義なことである．

本論文は，RSCC における利用統計情報をサンプルとした，数理計画的手法によるパラメータ最適化のスタディである．方法論としては，PC 上に NQS と同じふるまいをするシミュレータシステムを開発し，このシミュレータによって設定パラメータを比較検討し，数理計画的手法のひとつである Randomized Local Search を用いて最適値を求め，検証を行った．これによって，過去におけるある一定期間の利用状況が，いかなるパラメータを設定していれば最適化できたかが視覚化され，それを運用に生かす道筋を考察することによって，今後のスループット向上の可能性を探る．

3. RSCC のシステム概要と NQS

3.1 RSCC のシステム環境と利用状況

RSCC は，計算化学，流体力学，生体科学など様々な分野の研究に利用されており，ユーザが Web ブラウザからジョブをサブミットできるなど，研究者にとって利用しやすいシステムとなっている．システムの構成は，以下のとおりである．

- CPU： Intel Pentium Xeon 3.06GHz
- ノード数: 1024 ノード（2048CPU）
- 理論性能：12.4 TFLOPS
- メモリ：4GB/ノード，あるいは，2GB/ノード
- HDD：146GB/ノード
- オペレーティングシステム：Linux (Red Hat Version 8)
- NQS : Version 1.0，NQS-JM

RSCC 全体の 1024 ノードは，PC1～PC5 までの 5 つのクラスタに分割して運用されており，実験を行った PC1 には半分の 512 ノードが割り当てられている．

年間の運用スケジュールとしては，年に 4 回，定期的な保守があり，その際にはシステムを 1 日～4 日程度停止しており，また計画停電等，状況に応じてシステムを停止しているため，保守発生時には利用が落ちる傾向がある．

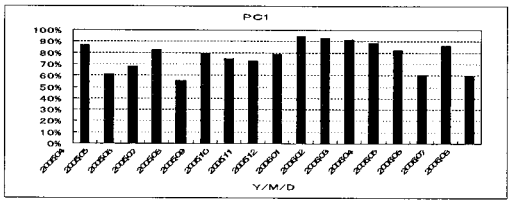

図 1　2005/04 – 2006/08 までの稼動状況
Fig. the system usage in PC1

図 1 は PC1 における稼働率を示したものである．2005 年度のシステム全体の稼働率は，年間を通して平均が約 78%となっている．このうち，2005 年 6 月および 8 月は 50～60%台にとどまり，2006 年 1～3 月は 90%以上と，季節によってばらつきがみられる．2006 年度に関しても 2005 年同様に 6，8 月は稼働率が落ちる傾向が見られる．

3.2 NQS の概要と RSCC における設定状況

3.2.1 NQS の概要

NQS は，アメリカ航空宇宙局（NASA）の航空力学数値シミュレーション計画の一環として開発されたジョブスケジューリングシステムを基に富士通株式会社が機能拡張して，Sun OS，VPP 環境で使用できるよう移植したものである．現在では，Solaris， Linux（Red hat 系）環境で動作しており，理研をはじめとし，さまざまな大学や研究機関などで広く利用されている．

---

* アルテアエンジニアリング株式会社
** プラットフォームコンピューティング株式会社
*** 富士通株式会社

3.2.2 NQSの主要なパラメータの意味

以下でNQSの主要パラメータについて解説する．

RUN LIMIT：ある時点において，同一キュー内で実行できる最大のジョブ数．空き資源がある場合でもこの本数を超えてジョブを流すことはできない．

PRIORITY：0-5までの6段階でそれぞれのキューの優先度を決定する．優先度の高いキューから処理される．

URUN：各ユーザが同時に実行できる最大ジョブ数．空き資源がある場合でもそのユーザはこの本数を超えてジョブを流すことはできない．

ULIMIT：各ユーザが空き資源がある場合に，ジョブを優先的に実行できる本数．また，他のユーザが待ち行列になければ，この本数を超えてジョブを流すことも可能である．

RUN LIMIT（COMPLEX）：キューに対して設定する．複数のキューにまたがって設定できる最大同時実行ジョブ数．例えばこの値を2つのキューに与えると，2つのキューで同時実行できるジョブの総和がこの値を越えることはできない．

NEXT RUN：同一ユーザのジョブが連続して実行されることを妨げるための制限．制限がある場合はU，ない場合はNとする．これがUになっている場合，あるユーザのジョブ終了時に，そのユーザのジョブが次のジョブとして待ち行列に存在していたとしても，ジョブは実行されずに次に持ち越される．

3.2.3 RSCCにおけるNQSの現在のパラメータ設定状況

キューの数：6

キューの種類：並列度によって，32CPU，64CPU，128CPUの3種類があり，それぞれが制限時間によって，ショート（10時間）とロング（72時間）の2種類に分かれているので，計6個のキューが設定されている．現在のNQSの設定は表1のとおりである．

ここでは便宜上，表中のRUN LIMIT 1はパラメータ解説のRUN LIMITを指し，RUN LIMIT 2はRUN LIMIT（COMPLEX）を指すものとする

表1 NQSの現在の設定
Table 1: The Current NQS parameter configuration

キュー名	S 032	L 032	S 064	L 064	S 128	L 128
RRIORITY	5	4	3	2	1	0
RUN LIMIT 1	12	12	12	12	8	8
URUN	6	6	3	3	3	3
ULIMIT	1	1	1	1	1	1
RUN LIMIT 2	16		16		8	
NEXT RUN	U	U	U	U	U	U

ショート：S 032，S 064，S 128
ロング：L 032，L 064，L 128

4. 最適化の手法

最適化とは一般に，ある状況において最善の決定をおこなうこと，あるいはいくつかの選択肢の中から最善のものを選ぶことを意味する．このような問題は与えられた制約条件のもとで，目的関数と呼ばれる最適性を測る評価指標が最小（大）となるような変数の値を見つけるという数学モデルとして定式化することができる[7].

資源制約の中で可能な限り多くのジョブを効率良く詰めるという観点で，スケジューラのパラメータ設定は最適化問題であり，各パラメータの値が整数でなければならないため，整数計画問題である．また，パラメータの組み合わせ的要素から「組み合わせ最適化問題」とも言える．この種類の問題は，いわゆるパラメータスタディであり，最適解を数理的に求めるためには全列挙法などの厳密解法もあるが，このようなアルゴリズムが必要とする計算量は，一般に整数変数個数の増加にともなって指数関数的に増加するため，計算には莫大な時間とシステム資源が必要になる[1].

一方，サンプルとして情報取得した期間の統計情報は，従来の設定でも既に高い利用率を有していたため，最適値は現行の設定パラメータの周辺に存在していることが予想され，発見的解法を用いることが有効であると判断された．そこで，本研究においては，発見的解法の1つであるRandomized Local Searchと呼ばれる近似解法を採用することとした．

4.1 Randomized Local Search

近似解法にはさまざまな手法があるが，そのひとつに局所探索法（Local Search）という手法がある[4][5]. この手法は「山登り法」とも呼ばれる．山頂の方向がわかっていない登山者は，山頂（最適値）を目指してある地点（初期解）から登り始め，暗い夜道を探索するが，その際には懐中電灯の照らせる範囲（近傍）内でのみ探索が可能である．懐中電灯の照らせる範囲で，現在の地点より少しでも高い地点があればそちらに移動し，そこで再度懐中電灯を照らし，少しでも高い位置を目指す．しかし，近傍にそれ以上高い地点がなければ現在の地点を最も高い地点と判断し，それ以上の登山を諦める．これをまとめると以下のとおりである．

＜Local Search のアルゴリズム＞

最大化問題の場合は以下のようになる．

＜最大化問題の場合＞

$\max\{f(x) \mid x \in F\}$を考える．

＜記号＞

f：目的関数と呼ばれる最適性を評価する評価指標

F：fの許容解

x^*が大域的最適解 $\Leftrightarrow \forall x \in F, f(x) \le f(x^*)$

N：許容解$x \in F$の近傍を$N(x)$と書く．

$x^* \in F$が局所的最適解 $\Leftrightarrow \forall x \in N(x^*), f(x) \le f(x^*)$

<アルゴリズム>

step1: 初期解 $x(\in F)$ を決定する.

step2: xが局所最適解ならば，終了

step3: xが局所最適解でなければ，x を，
$x' \in N(x)$ かつ，$f(x') > f(x)$を満たす
x' のどれかに更新し，*step1* に戻る.

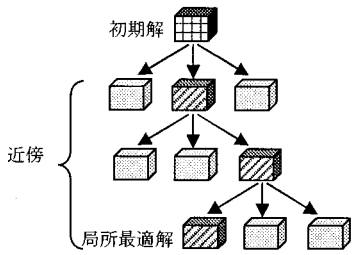

図 2　Local Search のアルゴリズム（近傍 3 個の例）
Fig. 2: The Local Search Algorithm

Randomized Local Search は Local Search を改良した近似解法のひとつである[6]. この手法は，Local Search における初期解を乱数によって複数個発生させることにより，局所から抜け出して，より大域的な解を見つけようという手法である.
この方法により得られる解が，必ずしも最適解であるとは限らないが，組み合わせ最適化問題においては，少ない計算コストでより良い解を得る手法として位置づけられ，工学の現場などで広く用いられている．すなわち，一度登山を諦めた登山者が新たな出発点（初期解）から再出発を行い山頂を目指すということを繰返した後に，たどり着いた地点の中でもっとも高い地点を山頂とすることになるので，より幅広い地域の探索が可能となる.

5. 数値実験

今回の数値実験は，最適なパラメータを求めて，その結果を比較検討するためのものであるが，実際の利用状況を実運用環境で再現することは不可能であるため，NQS と同じ振る舞いをするシミュレータを開発し，それを用いて Randomized Local Search に基づくパラメータスタディを実施するという手法を取った．この手法においては，初期解の発生個数と近傍の探索個数を与え，最適化度合いを評価することができるような指標（目的関数値）が定義できれば良い[6].

本研究においては，目的関数値として最大待ち時間のみを採用した．待ち時間とは，スケジューラがジョブを受け付けてから実際に実行されるまでの時間を指し，最大待ち時間とはある一定の期間内において，最も長い待ち時間とする．最大待ち時間のみを目的関数値として選択した理由は，複数のパラメータを同時に扱った場合，各パラメータの影響度が不明確になるという欠点があり，また，予備実験において，最大待ち時間を最小化することが付帯的にシステム利用効率（後述）も向上させたため，最大待ち時間のみを目的関数値とすることで十分であると判断した.
実験には前述の RSCC の PC1 (1024 CPU) を利用した.

5.1 最適化の評価基準

本研究の目的関数値は，最大待ち時間と設定したが，付帯的な評価基準としてシステム利用効率を百分率で定義する．実験のサンプルとして扱った利用統計情報は 2006 年 3 月 1 日～3 月 16 日の期間および 2006 年 6 月 20 日～7 月 6 日であるが，この期間における PC1 のシステム資源の総和を S とする．すなわち，

S = (1024 CPU)*日数（秒）である．期間中に実際に走行したジョブが要した CPU 数と日数（秒）の積の積分値を V とする.

この場合，V/S(%) は利用されたシステム資源を全体のシステム資源によって除した値となり，最適化による効率変化を判断する上で有用であると考えられるため，これを付帯的な評価基準として加える．以降においてはこの値をシステム利用効率と呼ぶことにする.

5.2 初期解および近傍の設定

前述の通り，実運用環境で設定されていたパラメータが比較的高いシステム利用効率を示していたため，シミュレータにかける最初の初期解は，実運用環境における設定パラメータの値とし，その近傍の探索から始めるものとした．2 個目以降の初期解は，乱数により決定する.

なお，今回の実験においては，1 回あたりの計算時間を短縮するため，初期解は 10 個以下，近傍は 8 個以下の探索としたが，これを繰り返し実施することによって，広範囲に探索を行った.

5.3 実験結果

今回の実験では，現運用で設定されていたパラメータによる結果（現運用）と，シミュレータを用いて求められた最適解に基づくパラメータによる結果（提案法）の比較に加え，さらに，パラメータを十分に大きくした場合の結果（無制約）との比較検討も行う．表中の項目「S 利用効率」はシステム利用効率を指し，「平均待ち時間（秒）」は，待ち時間の総和(秒) / 処理したジョブ数を示す.

5.3.1 現運用と提案法との比較検討

3 月期はサブミットされたジョブが非常に多く，恒常的にシステム利用効率が高く，多くのジョブがキューに並んだため，ジョブをサブミットしても非常に長く待たされるジョブが多い時期であった.

実運用の設定では，最大待ち時間が 806,728 秒であったが，提案する設定では 562,574 秒にまで短縮することができた．また，システム利用効率に関しては，92.9% から 96.2% に，約 3.3%向上し，さらに平均待ち時間が 21,756 秒短縮された.

表 2. 従来の設定と提案法による設定との比較
Table 2: The Comparison of Current and Proposal (March)

期間	項目	現運用	提案法	改善度
3月期	S 利用効率(%)	92.963	96.263	3.30
	最大待ち時間(秒)	806728	562574	244154
	平均待ち時間(秒)	91827	70071	21756
	処理したジョブ数	1123	1191	68

表 3 に提案法（3 月期）の結果に対するパラメータの設定を示す．

表 3 提案するパラメータの設定 (3 月期)
Table 3: Our proposal NQS parameter configuration (March)

キュー名	S 032	L 032	S 064	L 064	S 128	L 128
RRIORITY	5	4	3	2	1	0
RUN LIMIT 1	13	13	13	13	14	14
URUN	8	10	6	6	4	5
ULIMIT	2	1	2	2	1	1
RUN LIMIT 2	20		20		16	
NEXT RUN	U	U	U	U	U	U

一方，6-7 月期の最大待ち時間は，実運用による設定では，176,261 秒であるが，提案した設定では 113,793 秒 になり，62,468 秒短縮できた(表 4)．システム利用効率においては，約 0.53% にとどまったが，改善が見られた．平均待ち時間は 11,245 秒短縮される結果となった．

表 4. 従来の設定と提案法による設定との比較
Table 4: The Comparison of Current and Proposal (June-July)

期間	項目	現運用	提案法	改善度
6-7月期	S 利用効率(%)	76.489	77.024	0.535
	最大待ち時間(秒)	176261	113793	62468
	平均待ち時間(秒)	20720	9475	11245
	処理したジョブ数	1249	1249	0

表 5 に提案法による結果に対するパラメータの設定を示す．

表 5 提案するパラメータの設定 (6-7 月期)
Table 5: Our proposal NQS parameter configuration (June-July)

キュー名	S 032	L 032	S 064	L 064	S 128	L 128
RRIORITY	5	4	3	2	1	0
RUN LIMIT 1	17	17	15	15	14	14
URUN	10	10	8	8	7	7
ULIMIT	3	3	2	2	2	2
RUN LIMIT 2	24		22		18	
NEXT RUN	U	U	U	U	U	U

5.4 提案法と無制約との比較検討

全てのパラメータの値を十分大きくすると，スケジューリングに対して，事実上，制限は一切働いていないことになる．パラメータの値を十分大きくして制約をはずした（以降「無制約」と呼ぶ）場合，最大待ち時間は長くなるが，高いシステム利用効率が保たれるのではないかという仮説が考えられる．

この仮説を検証するため，数値実験をおこない，無制約の場合と提案法によるパラメータ設定とを比較検討する．ここでの無制約の設定を表 6 に示す．
Priority，Next Run の制約もあらかじめはずした．

表 6 無制約のパラメータの設定
Table 6: The NQS parameter configuration (no limit)

キュー名	S 032	L 032	S 064	L 064	S 128	L 128
RRIORITY	N	N	N	N	N	N
RUN LIMIT 1	100	100	100	100	100	100
URUN	100	100	100	100	100	100
ULIMIT	100	100	100	100	100	100
RUN LIMIT 2	100		100		100	
NEXT RUN	N	N	N	N	N	N

表 7 に結果を示す．3 月期に関しては，無制約は，提案法と比べて，平均待ち時間の値は良いが，最大待ち時間は非常に長く，システム利用効率は，僅かであるが低くなった．

6-7 月期においては，僅差ではあるが，システム利用効率，平均待ち時間において，無制約のほうがよい結果を得ているが，最大待ち時間においては提案法のほうがよい結果となった．

表 7: 提案法による設定と無制約との比較
Table 7: The Comparison of Our Proposal and no limit

期間	項目	提案法(A)	無制約(B)	(B)-(A)
3月期	S 利用効率(%)	96.263	96.14	-0.123
	最大待ち時間(秒)	562574	1024443	461869
	平均待ち時間(秒)	70071	58382	-11689
	処理したジョブ数	1191	1203	12
6-7月期	S 利用効率(%)	77.024	77.069	0.045
	最大待ち時間(秒)	113793	115642	1849
	平均待ち時間(秒)	9476	8655	-821
	処理したジョブ数	1249	1249	0

図 3- 6 は，システムのトータルの稼動 CPU 数を示したものである．図 3, 4 における横軸は 3 月 1 日を起点（0 秒）とした時刻 (t) で，縦軸は各時刻において使用されている総 CPU 数である．

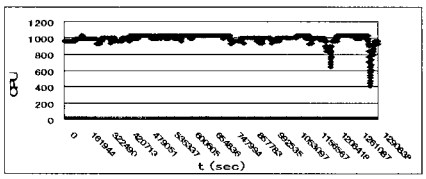

図 3　提案法による CPU 稼動状況（3 月期）
Fig. 3: The System Usage of Our Proposal (March)

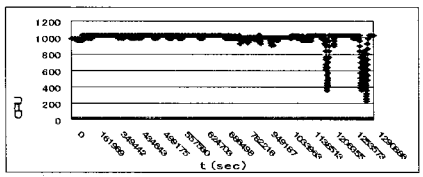

図 4　無制限による CPU 稼動状況（3 月期）
Fig 4: The System Usage of no limit (March)

3 月期はジョブが多いため恒常的に使用 CPU 数は高い水準で推移しているが，当該期間の後半で局所的な落ち込みが見られる．この時期は 3 月期の中でも，サブミットされたジョブが少なかった時期である．特に，

無制約においては，サブミットされたジョブが逐次処理されるため，局所的にジョブが減少したためであると考えられる．

また，3 月期において，無制約の最大待ち時間が非常に長くなった理由としては，使用 CPU が大きいジョブが長時間待たされたことがあげられる．例えば，128CPU を要求するような大きいジョブは，資源に余裕がなければ，空きができるまで待ち続けることになる．しかし，無制約の場合，同一キュー，同一ユーザのジョブが何本でも実行可能なため，当該ジョブが待っているという状況は考慮されないことになる．そのため，その時点で実行可能な小さいジョブが次々に実行されてしまい，十分まとまった空き資源ができるまで当該ジョブは実行されないという現象が起きる．このため，最大待ち時間が長くなったと考えられる．

図 5-6 は， 6-7 月期のシステムのトータルの稼動 CPU 数を示したものである．図5, 6における横軸は6月 20 日を起点（0 秒）とした時刻（t）である．

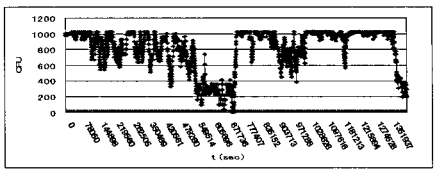

図 5 提案法による CPU 稼動状況（6-7 月期）
Fig 5: The System Usage of Our Proposal (June-July)

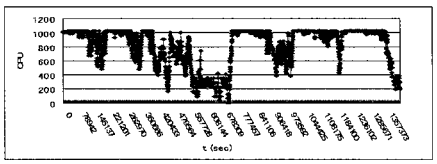

図 6 無制限による CPU 稼動状況（6-7 月期）
Fig 6: The System Usage of no limit (June-July)

6-7 月期はグラフとしては大きな相違は見られない．3 月に比べ，最大待ち時間が短縮されているのは，この期間においては，サブミットされているジョブがさほど多くなかったため，空き資源が比較的容易に確保できたためではないかと考えられる．

提案法と無制約の比較によって，時期や状況により異なる結果が得られることが確認できた．

6. 考察

現運用と提案法との比較に関しては，パラメータを最適化することによって，システム利用効率，最大待ち時間に関して，改善できる余地があることが判明した．現運用においては，3 月期においても 6-7 月期においても，最大時間待たされたジョブは，走行制限値侵害のため，資源に余裕があるにもかかわらずジョブが流れないという状況下で発生していた．提案法で最大待ち時間が短くなった要因は，走行制限値などのすべての制限において，現行の設定より緩和されたことによるものと考えられる．限られた実験期間の結果であるが，現設定より改善できたことによって，パラメータ設定の最適化が有効であると考えられる．特に 3 月期においては現設定においても，90%以上という高いシステム利用効率がさらに 3%高くなったことは成果であった．

また，提案法と無制約の比較に関しては，この実験より，時期やジョブの投入状況によって，必ずしも仮説どおりではないことが判明した．特に 6 月期のようにジョブが少ない時期は，無制約のようにパラメータを緩和しても最大待ち時間が大幅に延びることはないということがわかった．しかし，同時に，最適化という観点から見れば，3 月期のように，サブミットされるジョブが多く，混みあっている時期ほど，パラメータ設定の最適化が重要であることがわかった．

7. まとめ

NQS の最適なパラメータ設定を検討するために，シミュレータを構築し，それに基づき，数理計画法の一手法である Randomized Local Search を使って最適な設定を求めた．

今回得られた設定によって，実運用における設定は，最大待ち時間およびシステム利用効率の両方において，改善できることがわかった．また，無制約と比較することによって，パラメータ設定の最適化の重要性を確認することができた．

参考文献

[1] Colin R. Reeves, Modern Heuristic Techniques for Combinatorial Problems: Advanced Topics in Computer Science, Mcgraw Hill Book Co Ltd., 2000.

[2] Daisuke Takahashi, Mitsuhisa Sato and Taisuke Boku: Computation of High-Precision Mathematical Constants in a Combined Cluster and Grid Environment, Proc. 5th International Conference on Large-Scale Scientific Computations (LSSC'05), Lecture Notes in Computer Science, No. 3743, pp. 454--461, Springer-Verlag, (2006).

[3] Drror G. Feiterlson and Larry Rudolph, Parallel Job Scheduling: A Status Report, Proceedings of 10th Workshop on Job Scheduling Strategies for Parallel Processing, , pp.1-9, New York, 2004.

[4] M. Grotschel, L.Lovaz , and A. Schrijver, Geomatric Algorithms and Combinatorial Optimization, Springer, New York, NY, USA, 1988.

[5] E. Polak, Optimization -Algorithms and Consistent, Approximations, Appl. Math. , Sci., 124, Springer-Verlag, New York, 1997.

[6] 茨木俊秀，組み合わせ最適化問題をめぐる最近の話題，日本オペレーションズ・リサーチ学会，第30回シンポジウム論文集，pp.1-10.

[7]今野浩，鈴木久敏，整数計画法と組み合わせ最適化，日科技連, 1982.

[8]竹房あつ子，合田憲人，松岡聡，中田秀基，長嶋雲兵，グローバルコンピューティングのスケジューリングのための性能評価システム，Vol. 41, No.5, pp.1628-1637, 2000.