Vol.2019-HPC-168 No.12
2019/3/6

組合せ最適化問題向けアクセラレータを用いたHPCシステムのジョブスケジューリング手法の検討

濱崎 龍馬[1,a)]　坂本 龍一[1]　近藤 正章[1,2]

概要：HPC システムでは，多数のユーザーからサブミットされたジョブに対して，計算資源やユーザ毎の公平性などの制約を満たすようにスケジューリングが行われ，処理されている．今後の HPC システムでは消費電力が考慮すべき最も重要な制約の一つになると考えられており，各ジョブの実行順序や計算資源だけではなく，電力資源も考えてスケジューリングを実行することが必要となる．このようなジョブスケジューリングの問題は，組合せ最適化問題の一つであるパッキング問題の一種であり，NP 困難な問題である．本稿では近年着目されているイジングモデルを基にした組合せ最適化問題処理向けのアクセラレータを用いてスケジューリング問題を解く手法を検討する．特に整数線形計画法と遺伝的アルゴリズムによるスケジューリング手法と比較しつつ，最適化の計算時間および精度の側面からアクセラレータを利用した場合について評価する．

1. はじめに

現代社会の多くの重要な場面において，高度かつ高速な情報処理が HPC システムによって実現されている．今後，諸分野の発展に伴って高性能計算の計算負荷や処理すべきデータ量が増加し，社会的な需要の増大と共にユーザー数がますます増加することを想定すると，HPC システムの性能をさらに向上させることは非常に重要な課題だと言える．

従来は，HPC センターのスケールを拡大することでより多くの計算ノードを利用することが可能となり，計算性能の向上が達成されてきた．しかし，近年は供給可能な電力資源がボトルネックとなっており，このようなスケールの拡大が難しくなりつつある．そのため，今後は計算資源よりも電力資源の有効利用が重要になると考えられている．

このような状況への解決策の一つとして，オーバープロビジョニングの研究が進められている．既存の HPC システムは一般的に，ピーク電力がシステム全体の電力制限を上回ることがないように設計されている．これは最悪の状況に備えた設計である．しかし，実際にシステムを稼働する際にこのような状況が起こることは滅多になく，大量の電力資源が有効利用されないという問題が生じる．これを解決するために，オーバープロビジョニングでは最大消費電力の合計値がシステム全体の電力制限を上回ることを許容し，ピーク電力が電力制約を上回ることがないように計算ノードへの適切な電力配分を行う．オーバープロビジョニングを導入した HPC システムにおいては，計算ノード数の増加によって既存のシステムよりも計算資源の柔軟な割り当てが可能である．そこで，ジョブの実行順序や電力資源の割り当て方を含めた適切なスケジューリングを実行することにより，構成要素の計算性能の向上に頼ることなく HPC システム全体の性能の向上が期待される．

こうした背景を踏まえ，本研究ではオーバープロビジョニングされた HPC システムを対象としてジョブスケジューリング問題を扱う．ジョブスケジューリング問題は組合せ最適化問題の一つであるパッキング問題の一種であり，NP 困難である．したがって，大規模なシステムの場合，組合せ最適化問題を単純に解く手法では現実的な時間内に適切なスケジューリングを行うことが困難になることが危惧される．そこで本研究では，近年着目されているイジングモデルを基にした組合せ最適化問題向けアクセラレータの一つである富士通株式会社の Digital Annealer を利用したジョブスケジューリング手法について検討する．

2. 関連研究

2.1 HPC システムにおけるジョブスケジューリング

既存のシステムにおけるジョブスケジューリング手法としては，FIFO（First In First Out）と Easy-Backfilling を併用して計算資源の割り当てを行う手法が広く用いられ

[1] 東京大学
The University of Tokyo
[2] 理化学研究所 計算科学研究センター
RIKEN Center for Computational Science
[a)] hamasaki@hal.ipc.i.u-tokyo.ac.jp

てきた．一方でオーバープロビジョニングを導入したスケジューリングでは，電力資源を有効に活用するため，ピーク時の電力消費量を制御する電力キャッピング技術によって計算ノード毎の供給電力を決定する．そのため，ジョブスケジューリングはさらに複雑であり，様々な評価基準を用いて適切な手法の研究が進められている．Sarood ら [11] は，NP 困難なジョブスケジューリング問題を各ジョブの状態遷移（開始・完了・異常終了）時における数理最適化問題として捉えるアプローチを考案し，小規模及び大規模なシステムでの実験において大幅な効率化を確認した．Cao ら [3] はユーザーからジョブを実行する際のパフォーマンスが要求される状況において，機械学習を用いた実行時間の予測を通して適応的な電力制御を行うことによりスループットを 17 %改善した．坂本ら [10] は拡張性に富んだリソース管理フレームワークを開発し，実システムでの検証実験の結果から，追加するノード数の指針を考案した．

2.2 量子アニーリング

スピングラスは非磁性体の金属結晶中に不純物として微量の磁性体が混合された合金のことであり，スピングラスの基底状態を探索する問題は物理学における組合せ最適化問題の代表例である．この問題を定式化する方法の一つに，イジング模型を用いたスピングラスの数理的なモデル化がある．イジング模型は物理系の多体相互作用を二体相互作用までで近似するモデルである [2]．ここでは，スピングラスの格子面を xy 平面としたとき，全体に z 方向の磁場が印加された状況を仮定する．スピングラスの基底状態においては各スピンは磁場に平行または反平行のいずれかの向きをとることが知られているため，解空間の各要素を表現するため，各スピンに対して $\pm z$ の向きを表す二値変数を導入する．格子点 i に対応するスピンを $s_i \in \{-1, 1\}$ とする．スピン同士が互いに及ぼし合う力の範囲は小さいと仮定した場合，相互作用によるハミルトニアンを最近接の格子の配置のみで決定する．スピン i，j 同士の相互作用を $J_{i,j}$，スピン i における磁気モーメントと印加される磁場の積を h_i とすると，ハミルトニアンは

$$H = -\sum_{\langle i,j \rangle} J_{i,j} s_i s_j - \sum_i h_i s_i \qquad (1)$$

となる．ただし，$\langle i, j \rangle$ は最近接格子間のスピン対全てについて和をとることを意味する．スピン i，j は $J_{i,j} > 0$ のとき等しい向きに，$J_{i,j} < 0$ のときは逆の向きに揃うことでエネルギー的に安定となる．また，スピン i は $h_i > 0$ のとき磁場と等しい向きに，$h_i < 0$ のとき磁場と逆の向きを取ることでエネルギー的に安定となる．

スピンの個数 N に対して，あり得るスピン配置は 2^N 通りと指数的に増加する．コンピュータ上で基底状態を見つけるためには，式 1 のハミルトニアンが最小となるようなアルゴリズムにしたがって物理系を時間発展させる必要がある． **図 1** に示すように，アルゴリズムとして最急降下法を用いる場合，各時刻においてエネルギー的に安定な方向へ遷移するため，初期状態の選び方によっては局所的に安定な状態に陥るという問題がある．一方で，金属の焼きなまし（アニーリング）に着想を得たシミュレーテッドアニーリングは，一定の確率でエネルギー障壁を超えることでこの問題を回避する．初期段階では高温によって様々なスピン配置を網羅的に遷移させ，時刻の経過とともに温度を低下させることで最終的な基底状態への収束を目指す．

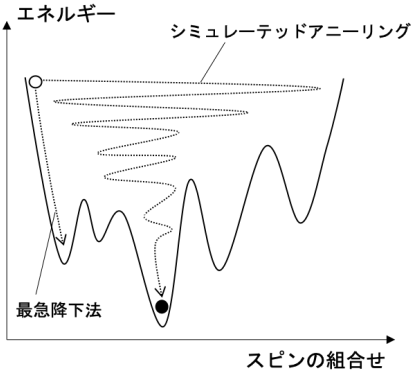

図 1: 状態が収束する様子の模式図.

量子アニーリングはシミュレーテッドアニーリングにおける古典的な熱揺らぎを量子力学的な揺らぎで代替した手法であり，量子力学的な重ね合わせによって複数の状態を同時に扱う．二つのアルゴリズムの性能を比較するため，イジング模型において磁場の印加がないモデル（Edwards-Anderson モデル）に関する理論的な考察を紹介する．シミュレーテッドアニーリングでは，スピンの個数を N 個，時刻を t とするとき，物理系のハミルトニアンは式 1 で磁場による項を除いた

$$H = -\sum_{\langle i,j \rangle} J_{i,j} s_i s_j \qquad (2)$$

となる．西森 [7] によると，物理系の温度 $T(t)$ を

$$T(t) \geq \frac{c}{\log(t+2)} \qquad (3)$$

を満たすように緩やかに時間変化させることで，長時間極限 $t \to \infty$ において物理系が確率 1 で基底状態に収束することが保証されている．ただし c は N に比例する定数である．一方，量子アニーリングでは，式 2 で各 s_i をスピン i の z 成分 σ_i^z へ置き換えることで，横磁場がないときのハミルトニアンが

$$H_0 = -\sum_{\langle i,j \rangle} J_{i,j} \sigma_i^z \sigma_j^z \qquad (4)$$

と得られ，スピン i の x 成分 σ_i^x を用いて横磁場によるハミルトニアンが

$$H_1 = -\sum_i \sigma_i^x \tag{5}$$

と得られる．横磁場の強度に相当する時間関数 $\Gamma(t)$ を用いると，全体のハミルトニアンは

$$H(t) = H_0 + \Gamma(t)H_1 \tag{6}$$

となる．$\Gamma(t)$ を非常に大きな値から 0 まで徐々に減少させることで，横磁場によって $\pm z$ の向きのスピンが重ね合わされた状態から，横磁場がないときの本来の物理系の基底状態へと連続的に遷移することが期待される．森田ら [5] によると，横磁場の強度を

$$\Gamma(t) \geq t^{\frac{c}{N}} \tag{7}$$

を満たすように緩やかに減衰させることで収束が保障される．式 3 と式 7 から，量子アニーリングの方がシミュレーテッドアニーリングよりも短時間での収束が保障される．

イジング模型において，全てのスピン間の相互作用を考慮した場合のハミルトニアンは

$$H = -\sum_{i<j} J_{i,j}\sigma_i\sigma_j - \sum_i h_i\sigma_i \tag{8}$$

となる．ただし，変数は $\sigma_i \in \{-1, 1\},\ 0 \leq i < N$ であるため，計算機システム上で扱うためには $\sigma_i = 2x_i - 1$ によって $x_i \in \{0, 1\},\ 0 \leq i < N$ へ変換する必要がある．このとき，ハミルトニアンは

$$H = -\sum_{i<j} \tilde{J}_{i,j}x_ix_j - \sum_i \tilde{h}_ix_i - c \tag{9}$$

となり，定数項を除いた形式を二次無制約二値最適化（Quadratic Unconstrained Binary Optimization, QUBO）形式と呼ぶ．組合せ最適化問題向けアクセラレータは QUBO 形式の基底状態を高速に探索することができる．そこで，我々が扱いたい組合せ最適化問題に二値変数を導入し，その最適解を QUBO 形式の基底状態に対応付けることにより，NP 困難な問題であっても高速な解の探索が期待できる．これをイジング模型へのマッピングと呼ぶ．適切なマッピングにより，量子アニーリングの原理を用いることでスピングラスの基底状態の探索に限らない様々な組合せ最適化問題を解くことができる．

2.3 組合せ最適化問題向けアクセラレータ

現在，量子アニーリングの原理をはじめとした量子力学的な手法を駆使することで組合せ最適化問題を高速に解くハードウェアの開発が盛んに行われている [4, 8, 13–15]．

富士通株式会社がトロント大学と共同開発した Digital Annealer もその一つで，量子アニーリングの原理をデジタル回路上で模倣した計算機アーキテクチャである．量子現象が実現されているわけではないが，デジタル回路上で量子現象と等価な時間発展を模倣している．デジタル回路を用いた基本最適化回路を並列的かつ階層的に動作させることで，データ移動の極小化や高密度な集積化が可能となった．基本最適化回路内では確率的な状態遷移の評価を並列に計算し，効率的に最適解の探索を行う技術が用いられている．さらに，遷移確率を適切に制御することで局所解に陥って膠着状態となる可能性を小さくし，最適解への到達し易さの向上を実現している [15]．2018 年 12 月 21 日にリリースされた Digital Annealer（第二世代）では精度とパラメータチューニングの大幅な改善がなされ，実用性がさらに高められた．本研究では Digital Annealer（第二世代）を用いて実験を行う．

3. イジング模型を用いたジョブスケジューリングの最適化

本研究では，ジョブスケジューリング問題を組合せ最適化問題として捉えた先行研究を元にイジング模型へのマッピングを行った．本章では対象とするジョブスケジューリング問題のモデルとイジング模型へのマッピングの手法について述べる．

3.1 電力資源を考慮したジョブスケジューリング問題

本来，ジョブスケジューリング問題は計算資源と電力資源の制約下においてシステム稼動時のターンアラウンドタイムを最小化するようなパッキング問題として解釈されるべきである．しかし，実際の HPC システムではオンライン処理が行われるため，どのような特性のジョブがいつサブミットされるかという情報があらかじめ与えられておらず，本来の最適化を行うことは困難である．そこで Sarood ら [11] は，ジョブスケジューリング問題をジョブのサブミット・実行完了というトリガイベントにおける

- 入力： 実行待ちまたは実行中のジョブの集合と，各ジョブに特定のリソースを割り当てた場合の実行時間
- 出力： リソースの割り当て
- 目的： HPC システムのスループットの最大化

の最適化問題として捉えた．つまり，各トリガイベントで瞬間的な最適化を反復することでシステム稼動時の全体においてもターンアラウンドタイムが小さくなることを期待している．この解釈により，**図 2** のような整数線形計画問題としての定式化が可能となる．ただし，各変数は**表 1** で定義されている．ジョブ j が計算ノード数 n，電力キャップ値 p で実行されるときに 1，それ以外の場合に 0 をとる二値変数 $x_{j,n,p}$ を導入することで，ジョブが実行中であるか否かということと，実行中の場合に計算ノード数と電力キャップ値がいくら割り当てられるかということが表現される．

ここでは

- 同一ジョブに割り当てられる計算ノードには同一の電

目的関数：HPC システムのスループット

$$\sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} w_j s_{j,n,p} x_{j,n,p} \tag{10}$$

制約条件：

- 実行待ちのジョブの制約
  実行待ちのジョブに対し，割り当てられる計算ノード数・電力キャップ値の組が存在しない，またはただ一つ存在する：

$$\sum_{n\in N_j}\sum_{p\in P_j} x_{j,n,p} \le 1,\ \ \forall j \in I \tag{11}$$

- 実行中のジョブの制約
  実行中のジョブに対し，割り当てられる計算ノード数・電力キャップ値の組が唯一つ存在する：

$$\sum_{n\in N_j}\sum_{p\in P_j} x_{j,n,p} = 1,\ \ \forall j \in \mathcal{I} \tag{12}$$

- 計算ノード資源の制約
  割り当てられる計算ノード数の合計値はシステム全体の計算ノード資源の制限下である：

$$\sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} n x_{j,n,p} \le \mathrm{N} \tag{13}$$

- 電力資源の制約
  割り当てられる電力キャップ値とベース電力の合計値は電力資源の制限下である：

$$\sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} n(p+W_{base}) x_{j,n,p} \le W_{max} \tag{14}$$

図 2: 整数線形計画法による電力制約下でのジョブスケジューリング問題の定式化（Sarood らの文献 [11] より引用.）

力キャップ値が供給される

- 計算機の冷却に要する電力は考慮しない
- 実行中のジョブの特性の変化は無視できる
- ジョブの実行は中断されない

ことが仮定されている．式 10 中の

$$s_{j,n,p} = \frac{t_{j,\min(N_j),\min(P_j)}}{t_{j,n,p}} \tag{15}$$

はジョブ j を計算ノード数 n，電力キャップ値 p で実行した時の実行時間 $t_{j,n,p}$ の短縮率を表す指標である．最小のリソース（計算ノード数 $\min(N_j)$，電力キャップ値 $\min(P_j)$）を与えられた場合に要する実行時間 $t_{j,\min(N_j),\min(P_j)}$ との比をとることで，実行時間の短縮率が表現されている．また，

$$w_j = (t^{rem}_{j,\min(N_j),\min(P_j)} + (t_{now} - t^a_j))^{\alpha}\ \ (\alpha \ge 0) \tag{16}$$

はジョブ同士の公平性とシステム全体のスループットとの間のトレードオフを反映する重み変数である．ジョブ j が最悪の場合に（最小のリソースを与えられた場合に）要する残りの実行時間 $t^{rem}_{j,\min(N_j),\min(P_j)}$ と，サブミットされてから現在時刻までの経過時間 $t_{now} - t^a_j$ を加えて α 乗することで，α が大きいほど公平性が重視され，小さいほどシステム全体のスループットが重視される．

表 1: 整数線形計画問題の変数（Sarood らの文献 [11] より引用および一部改変.）

記号	説明
N	計算ノードの総数
J	全てのジョブの集合
W_{max}	供給電力の総量
$\mathcal{I}$	実行中のジョブの集合
I	待ち状態のジョブの集合
$\mathcal{J}$	到着したが完了していないジョブの集合，
N_j	ジョブ j を実行可能な計算ノード数の集合
P_j	ジョブ j を実行可能な電力キャップ値の集合.
n_j	ジョブ j が現在実行されている計算ノード数
w_j	ジョブ同士の公平性を規定する重み因子
α	公平性とスループット間のトレードオフの指標
$x_{j,n,p}$	ジョブ j が計算ノード数 n，電力キャップ値 p で実行されるときに 1，それ以外の場合に 0 をとる二値変数
t_{now}	現在時刻
t^a_j	ジョブ j の到着時刻
W_{base}	CPU とメモリを除く全ての電力（ベース電力）
$t_{j,n,p}$	ジョブ j が計算ノード数 n，電力キャップ値 p で実行されるときの実行時間
$s_{j,n,p}$	ジョブ j が計算ノード数 n，電力キャップ値 p で実行されるときの実行速度の上昇率の指標
$t^{rem}_{j,n,p}$	ジョブ j が計算ノード数 n，電力キャップ値 p で実行されるときの残りの実行時間

3.2 イジング模型へのマッピング

以上の整数計画法による定式化を元に，ジョブスケジューリング問題のイジング模型へのマッピングを提案する．簡単のため，以下では $p \in P_j$，W_{base}，W_{max} は整数とする．

まず，四つの制約条件をもとにそれぞれのハミルトニアンを構成する．

- 実行待ちのジョブの制約条件（式 11）：

$$H_{waiting\ job} = \sum_{j\in I}\Big(\sum_{n\in N_j}\sum_{p\in P_j} x_{j,n,p}\Big(\sum_{n\in N_j}\sum_{p\in P_j} x_{j,n,p} - 1\Big)\Big) \tag{17}$$

  へ変換する．明らかに $H_{waiting\ job} \ge 0$ で，式 11 が満たされる場合に限り $H_{waiting\ job} = 0$ となる．つまり，このハミルトニアンは制約条件が満たされない場合に対するペナルティの役割を担う．
- 実行中のジョブの制約条件（式 12）：
  同様に

$$H_{running\ job} = \sum_{j\in\mathcal{I}}\Big(\sum_{n\in N_j}\sum_{p\in P_j} x_{j,n,p} - 1\Big)^2 \tag{18}$$

  へ変換する．
- 計算ノード資源の制約条件（式 13）：
  補助変数として

$$y_k = \begin{cases} 1 & (\sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} nx_{j,n,p} = k) \\ 0 & (\text{上記以外}) \end{cases} \quad (19)$$

なる二値変数を導入することで

$$H_{node} = (\sum_{k=0}^{\mathrm{N}} y_k - 1)^2 + (\sum_{k=0}^{\mathrm{N}} ky_k - \sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} nx_{j,n,p})^2 \quad (20)$$

へ変換する．第一項は HPC システム全体で使用する計算ノード数が 0 から N までのうちのただ一つの値のみを取ることを保証しており，式 19 を反映している．第二項は HPC システム全体で使用される計算ノード数が実行待ちまたは実行中のジョブに割り当てられる計算ノード数の総和となることを保証しており，式 13 を反映する．

- 電力資源の制約条件（式 14）:
  式 14 では，ジョブを割り当てられたノードのみについてベース電力 W_{base} が考慮されている．しかし，実際の HPC システムでは全ての計算ノードがベース電力を消費する場合が一般的であるため，一部修正して

$$\sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} npx_{j,n,p} \le W_{max} - \mathrm{N}W_{base} \quad (21)$$

とする．補助変数として

$$z_l = \begin{cases} 1 & (\sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} npx_{j,n,p} = l) \\ 0 & (\text{上記以外}) \end{cases} \quad (22)$$

を導入することで

$$H_{power} = (\sum_{l=0}^{W_{max}-\mathrm{N}W_{base}} z_l - 1)^2 + (\sum_{l=0}^{W_{max}-\mathrm{N}W_{base}} lz_l - \sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} npx_{j,n,p})^2 \quad (23)$$

へ変換する．

これらの部分的なハミルトニアンを足し合わせることで，全ての制約条件に対するハミルトニアンを

$$H_{const} = H_{waiting\ job} + H_{running\ job} + H_{node} + H_{power} \quad (24)$$

と定める．

次に，目的関数をハミルトニアンに変換する．

$$H_{obj} = -\sum_{j\in\mathcal{J}}\sum_{n\in N_j}\sum_{p\in P_j} w_j s_{j,n,p} x_{j,n,p} \quad (25)$$

と定める．符号を反転させることによって，整数線形計画法における目的関数の最大化問題を，ハミルトニアンの最小化問題へと変換している．

最後に，H_{const} と H_{obj} を足し合わせることで全体のハミルトニアンを

$$H = AH_{const} + BH_{obj} \quad (26)$$

と定める．ここで，A，B は正の定数とする．

ここで，制約条件を全て満たすという前提の下で目的関数を最小化する必要があることに注意する．言い換えれば，H_{obj} を小さくするために H_{const} による制約が満たされなくなるという状況を排除する必要があるということである．そのため，A を B に対して十分大きく定める必要がある．

4. 実験

4.1 データセット

本節では，現実的な状況を再現するためのデータセットの作成方法について述べる．ジョブが使用する計算ノード数とサブミットされるタイミングは，実際の並列ワークロードのログ [1] に従う．ここではアルゴンヌ国立研究所のスーパーコンピュータ Intrepid - Blue Gene/P [12] のログを用いる．このログから各ジョブに割り当てられた計算ノード数とサブミットされた時刻のみを抽出する．特にジョブが要求するノード数が 64, 512, 1,024, 2,048, 4,096 のジョブのみを抽出した．サブミットされた全ジョブのうち 90 % のジョブが 64, 512, 1,024, 2,048, 4,096 ノードのいずれかのノード数を要求しており，その他の計算ノード数を要求するジョブの個数は 10 % 未満であったため，その他のジョブは時系列に大きな影響は与えないものとして除外した．また，Intrepid のログの中で 3,000 番から 3,049 番までの計 50 ジョブが含まれる区間を抽出した．さらに，資源の利用率が異なる場合を想定し，時系列におけるジョブのサブミットされた時刻を $\gamma \in (0, 1]$ 倍することで，ジョブのサブミットされる頻度を高めた場合についても考える．

Intrepid のログにはサブミットされた各ジョブの電力特性の情報は含まれていないため，ジョブの電力特性として九州大学情報基盤研究開発センターのスーパーコンピュータ HA8000（2017 年 9 月末に運用停止）で NAS Parallel Benchmarks（NPB）[6] を実行した結果を用いた．評価に用いた NPB の種類を**表 2** に示す．それぞれのベンチマーク，クラス，ノード数に対し電力キャッピング値を，$40\,\mathrm{W}, 45\,\mathrm{W}, \cdots, 100\,\mathrm{W}, 105\,\mathrm{W}$ とした場合の実行時間，消費電力を用いる．

Intrepid と HA8000 ではシステム全体のノード台数が大きく異なる．そのため，Interpid で各ジョブが要求する計算ノード数を HA8000 で実行できる規模に変換した．**表 3** のように，Intrepid で要求する計算ノード台数を HA8000 で実行できるよう縮小した．

表 2: 評価に用いる NPB の種類

ベンチマーク	クラス	要求する計算ノード数
BT	D	16
	E	256
CG	D	16, 32, 64
	E	128, 256
EP	D	16
	E	128, 256
FT	D	16, 32, 64
	E	128, 256
LU	D	16, 32, 64, 128
	E	256
MG	D	16
SP	D	64
	E	256

表 3: 計算ノード数の対応

Intrepid	64	512	1,024	2,048	4,096
HA8000	16	32	64	128	256

計算ノード数を要求するジョブのベンチマークとクラスは該当する特性からランダムに選び，時系列上に配置した．表 2 に示したベンチマークは様々な特性を持っていることから，ランダムに選択することで現実的な状況に近いデータセットとなることが期待される．この時，全てのジョブ $j \in J$ について $|N_j| = 1$ であることに注意が必要である．ほとんどの HPC システムにおいては実行開始時に割り当てられた計算ノード数を実行中に変化させることは不可能であるため，これは現実的な仮定であると言える．

全てのジョブ $j \in J$ に対する電力キャップ値の集合 P_j は共通とし，$\min P_j = 40\,\mathrm{W}$，$\max P_j = 105\,\mathrm{W}$ となるように区間を $|P_j|$ 等分し，整数に丸める．例えば $|P_j| = 2$ の場合は $P_j = \{40\,\mathrm{W}, 105\,\mathrm{W}\}$，$|P_j| = 3$ の場合は $P_j = \{40\,\mathrm{W}, 73\,\mathrm{W}, 105\,\mathrm{W}\}$ となる．さらに，計算ノードが計算負荷に依らず定常的に消費するベース電力を $W_{base} = 30\,\mathrm{W}$ とする．

4.2 スケジューリング方式

提案手法の性能を評価するため，二つのスケジューリング手法と最適化の精度および時間の側面から比較した．

4.2.1 提案手法

提案手法の実験は Digital Annealer（第二世代）を用いて行った．Digital Annealer には Web API として QUBO 形式の最適解を求める qubo/solve というサービスが含まれる．これを利用し，ハミルトニアン中の QUBO 形式を qubo/solve API でリクエストとして送信し，受信したレスポンスに含まれる解のうち最もハミルトニアンが小さいものと，その解を得るのに要した時間の情報を抽出した．この操作を全ての制約条件を満たすような解が得られるまで反復した．シミュレーションを実行する PC と Digital Annealer サーバーとの間で発生する通信時間や，サーバーでの実行待ちの時間の影響を排除するため，ハードウェア上でのアニーリング処理に要した時間のみを最適化に要した時間とした．また，アニーリングを実行する際は各種温度パラメータのチューニングが必要となるが，扱う問題の種類や規模によって最適な値は異なるため，これらを網羅的に調べることは困難である．従って，提供されているソルバのうち温度パラメータのチューニングを自動的に行う機能を持つ FujitsuDA2PTSolver を利用し，指定可能な各種パラメータの値は全てデフォルト値で統一した．並列にアニーリングを行う個数であるレプリカ数については，指定可能な最大値である 128 に設定した．また，本実験では $A = 10000|\mathcal{J}| \max(w_j s_{j,n,p})$，$B = 1$ とした．

4.2.2 従来手法 1：整数線形計画法

整数線形計画法は第 3.1 節で概要を述べた Sarood ら [11] の手法に基づく．Python の線形計画法のモジュールである PuLP 1.6.9 [9] で定式化と解の探索を行なった．ソルバとしては，モジュールに同梱されているオープンソースの混合整数計画ソルバである CBC（COIN-OR Branch and Cut）を用いた．CBC は分枝限定法の一種である分枝切除法に従っており，分枝限定操作と切除平面操作によって緩和問題における目的関数の下界値または上界値を改善するという特長がある．実行結果には解が最適であるか，探索が完了したかどうか，実行可能であったかどうか等についての情報がステータスとして含まれており，正常にスケジューリングが実行されていることをトリガイベント毎に確認する．

4.2.3 従来手法 2：遺伝的アルゴリズム

遺伝的アルゴリズムは，自然界において生物が進化によって環境へ適応する原理を模倣したアルゴリズムであり，組合せ最適化問題の汎用的な近似解法として広く用いられている．

FIFO に従って実行中のジョブと実行待ちのジョブをそれぞれサブミットされた時刻でソートし，順に 5 ジョブずつスケジューリングした．これは単なる遺伝的アルゴリズムでは多数のジョブに対するスケジューリングの精度が低く，現実的な時間内に最適化を行うことが困難なためである．符号化では待ち状態および実行状態のジョブを遺伝子と一対一に対応させ，100 個体をまとめて 1 集団とした．各遺伝子には 0 または $p \in P_j$ の値が格納されており，0 は対応するジョブが実行されないことを意味し，$p \in P_j$ はジョブが電力キャップ値 p で実行されることを意味する．初期化においては，待ち状態または実行中のジョブの遺伝子を 50 % の確率で 0，50 % の確率で $p \in P_j$ からランダムに与えた．適応度は制約条件（式 11，12，13，14）を満たす場合は式 10 の符号を反転させた値とし，満たさない場合は適当なペナルティを加算した値とした．ここでは式 12

が満たされないようなジョブ毎に 100 を加算し，式 13 または式 14 が満たされない場合には超過したリソース（計算ノード数，電力キャップ値（W））の 100 倍の値を加算した．ただし，本実験では $|N_j| = 1$ であり，符号化の方法から式 11 は常に満たされることに注意する．選択は上位 20 % の適応度を持つ個体の抽出によって行い，20 組を親として交叉させて子を 20 個体生成した．こうして得られた 20 組の親と子を保存し，親の世代の残り 80 個体からランダムに 60 個体を保存することで合計 100 個体からなる次の世代を生成した．これはエリート主義と呼ばれる選択方法で，世代の更新による評価値の低下を防ぐという利点がある．交叉には各遺伝子の値を確率 0.5 で入れ替える方法である一様交叉を用いた．これは最適解に近いが最適解とは異なる評価値を持つ遺伝子が多数生成されるヒッチハイキングという現象に陥る可能性が低いという利点を持つ．そして，集団全体では 5 %，個体の染色体全体では 10 % の確率で遺伝子を突然変異させ，世代を更新した．以上の操作を第 100 世代まで反復し，適応度が最大の個体が制約を満たせばスケジューリングは終了とし，最適化するジョブの集合を更新した．制約を満たさなければ再び初期化に戻り，所望の個体が得られるまで反復した．全てのジョブ $j \in \mathcal{J}$ に対するリソース割り当てが決定した時，またはリソースが不足した時にスケジューリングを終了した．

4.3 結果

本実験では，計算ノードの総数を N =1,024 ，供給電力の総量を $W_{max} = 10{,}000$ とした．変化させるパラメータとしては

- 公平性の指標 α
- 電力キャップ値の集合の大きさ $|P_j|$
- ジョブの到着間隔をスケーリングする因子 γ

の三つが考えられるが，本研究ではジョブ同士の公平性よりもシステム全体のスループットを最大化することを重視し，$\alpha = 0$ とした．

はじめに，最適化の精度に対する $|P_j|$ の影響を調べた．整数線形計画法を用いたスケジューリングにおいて，$\gamma = 0.1$ の場合の電力利用率の時間平均は**図 3** に示す通りであった．ジョブに割り当てられず，ベース電力のみを消費する状態を遊休状態，ジョブに割り当てられることで電力キャップ値を含めて消費する状態を実行状態としている．これより，各状態の時間平均は $|P_j|$ に依らずほぼ一定となっていることがわかる．ただし，電力利用率は全てのジョブの実行に要した時間から最適化に要する時間を除いた時間における平均値である．電力利用率はスケジューリングの最適化の精度を反映した指標であるため，$|P_j| = 3$（$P_j = \{40\,\mathrm{W}, 73\,\mathrm{W}, 105\,\mathrm{W}\}$）の場合でも電力資源が有効に利用されていると言える．また，最適化に要した時間の合計値を表 4 に示す．解の探索領域が比較的小さい $|P_j| = 3$

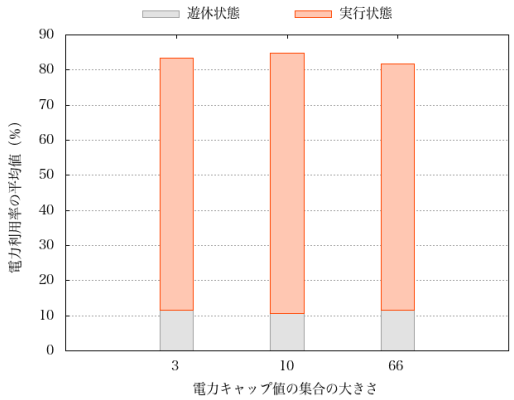

図 3: 整数線形計画法を用いたスケジューリングにおける電力キャップ値の集合の大きさと電力利用率の関係．

表 4: 整数線形計画法による最適化に要した時間．

電力キャップ値の集合の大きさ	3	10	66
最適化に要した時間の合計値（秒）	4.23	65.52	42.08

では短時間で最適化が完了した．したがって，以下では効率的なスケジューリングが期待される電力キャップ値の集合の大きさとして $|P_j| = 3$ を固定し，γ を 0.05 から 0.5 まで 0.05 ずつ 10 段階で変化させることで各スケジューリング手法の性能を比較した．ただし，図の凡例では整数線形計画法を ILP，遺伝的アルゴリズムを GA，Digital Annealer を DA と表記する．

まず，HPC システム全体のスループットとして，処理された単位時間あたりのジョブ数を**図 4** に示す．ただし，$\gamma = 0.05,\ 0.1,\ 0.15$ においては Digital Annealer によるシミュレーションが現実的な時間内に完了しなかったため，値を表示していない．また，γ の軸が反転していることに注意する．これは以降の図でも同様である．γ が大きい場合には各手法でのスループットに大きな差は生じなかったが，γ が小さい場合には整数線形計画法や遺伝的アルゴリズムと比較して Digital Annealer でのスループットが小さくなった．γ が小さくなるとジョブの到着頻度が高くなるため，最適解に近いスケジューリングを行うことができる場合にはリソースを有効活用しやすくなると考えられる．整数線形計画法と遺伝的アルゴリズムでは γ に対してスループットがほぼ単調減少しており，この傾向が認められる．一方で，スループットがほぼ一定であった Digital Annealer では

- 制約条件を満たす解が得られても，目的関数が小さくない
- 制約条件を満たす解を得ること自体が困難であり，最適化に要する時間が長い

ことのいずれか，あるいはその両方が起きていると考えられる．

そこで，**図 5** に，全てのジョブの実行に要した時間か

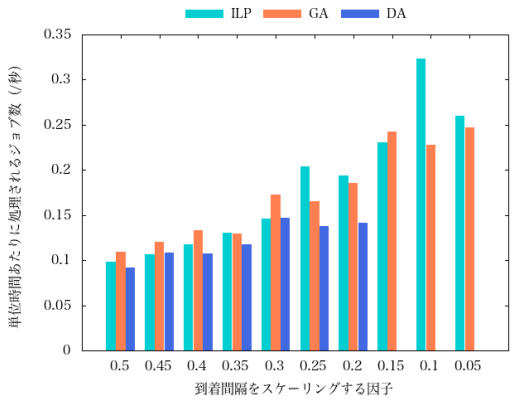

図 4: システムのスループット．

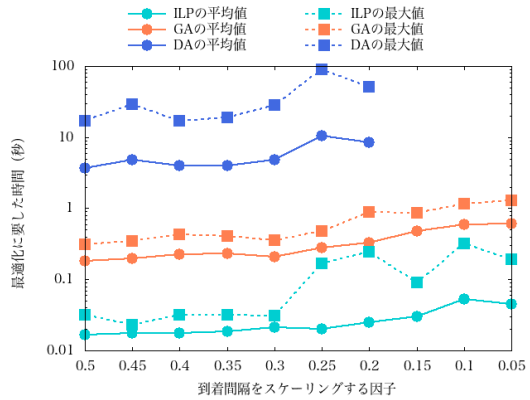

図 6: 最適化に要した時間の平均値および最大値．

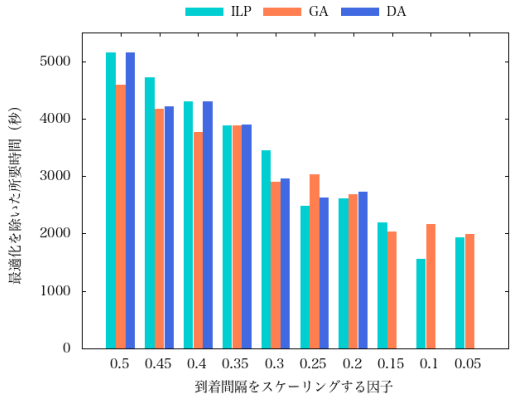

図 5: 最適化に要する時間を除いた所要時間．

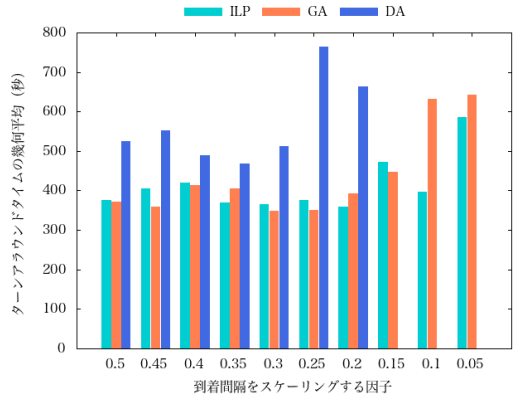

図 7: ジョブのターンアラウンドタイムの幾何平均．

ら最適化に要する時間を除いた時間を示す．三つの手法による所要時間に大きな差は生じていないことから，得られた解の最適解への近さに大きな差はない．したがって，Digital Annealer を用いたスケジューリングにおけるスループットを低下させ得る前述の原因のうち，前者には該当しないと考えられる．

一方で，最適化に要した時間の平均値および最大値は**図 6** のようであった．従来手法と比較して Digital Annealer における平均値・最大値は非常に長くなっていることから，前述の原因のうちの後者によってスループットが低下したと考えるのが自然であろう．Digital Annealer では，一度のアニーリングにおける最適化時間はほぼ変化しなかった．そのため，アニーリングの結果として得られた解が制約条件を満たさず，最適化を何度も反復することによって最適化に要する時間が長くなった．制約条件を満たす解を得ることが困難であるのは，γ が小さくなると一度のスケジューリングにおいて考慮すべきジョブ数が増加し，それに伴って扱うビット数が増加することで解の探索領域が大きくなるためである．

さらに，ジョブ毎のターンアラウンドタイムの幾何平均を**図 7** に示す．ターンアラウンドタイムはジョブがサブミットされてから実行が完了するまでの合計時間であり，通信時間などを無視すれば HPC システムのユーザーの待ち時間に相当する．ジョブ毎のターンアラウンドタイムの代表値として算術平均値ではなく幾何平均値を用いたのは，算術平均値では実行時間の長い大規模なジョブの影響が支配的になり，小規模なジョブの結果を正しく反映し難いという欠点を考慮したためである．整数線形計画法や遺伝的アルゴリズムと比較して Digital Annealer でのターンアラウンドタイムが大きくなっており，γ が大きく到着頻度が低い場合でも，大きな差が生じた点でシステムのスループットと異なる傾向を示した．これは，各ジョブのわずかな実行時間が変動した場合にも，データセット全体の所要時間に応じて定まるシステムのスループットには大きな影響を与えなかったことを示唆している．

Digital Annealer では現実的な時間内にシミュレーションを完了させることができなかった場合について調べるため，$\gamma = 0.05$ の場合に，アニーリングに初めて 100 回連続して失敗した時の QUBO 形式のエネルギーの分布を**図 8** に示す．ソルバの温度パラメータとして並列にアニーリングを行う回数であるレプリカ数を 128 に設定したため，一度のアニーリングでは解の候補が縮重度も含めて 128 通り得られる．アニーリングを 100 回反復することで，12,800 通りの解のエネルギー分布が得られた．図から，基底状態付近において局所的最適解が複数存在しており，大域的最適解に到達できていないことが読み取られる．このような

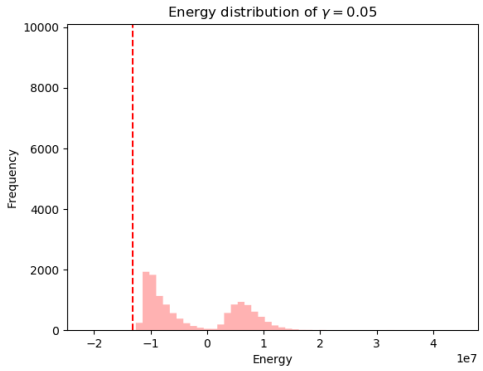

図 8: エネルギーの分布．点線は基底状態のエネルギー．

場合に対処するためには，本実験で指定しなかった温度パラメータなどを適切に指定し，局所解での膠着状態を回避する工夫が必要となる．しかし，システムの規模やジョブの特性，更には各トリガイベント毎に有効なパラメータは異なるため，これらを試行錯誤的に求めることは難しいと考えられる．

5. まとめ

本研究ではオーバープロビジョニングされた HPC システムを対象として扱い，将来的に最も重要な制約になると考えられている電力資源を考慮したジョブスケジューリング問題の組合せ最適化問題向けアクセラレータを用いた最適化について検討した．NP 困難なジョブスケジューリング問題をトリガイベントにおける整数計画問題として捉えた先行研究をもとにイジング模型へのマッピングを示し，量子アニーリングの原理を用いた組合せ最適化問題向けアクセラレータの一つである富士通株式会社の Digital Annealer（第二世代）を利用して提案手法の実装を行った．実際の HPC システムのログを元に作成したデータセットを用いたシミュレーションによる実験では，従来手法として位置付けた整数線形計画法と遺伝的アルゴリズムとの比較を通して評価を行った．評価の結果，提案手法はアニーリングにおける最適化の精度が主な原因で従来手法に計算時間の面で及ばなかったが，状況に応じた適切な温度パラメータチューニングの指針が得られれば局所解を回避することで性能の改善が見込まれることがわかった．また，得られた解の最適解への近さにおいては従来手法に匹敵する結果が得られた．

謝辞 本研究の一部は，JST CREST 課題番号 JPMJCR18K1（研究課題名「エッジでの高効率なデータ解析を実現するグラフ計算基盤」）の研究，および株式会社富士通研究所との共同研究によるものである．

参考文献

[1] Logs of Real Parallel Workloads from Production Systems. http://www.cs.huji.ac.il/labs/parallel/workload/logs.html, 2015.
[2] Stephen G. Brush. History of the Lenz-Ising model. *Reviews of Modern Physics*, Vol. 39, No. 4, pp. 883–893, 1967.
[3] Thang Cao, Yuan He, and Masaaki Kondo. Demand-Aware Power Management for Power-Constrained HPC Systems. *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, pp. 21–31, 2016.
[4] D-Wave Systems. The D-Wave 2000Q^{TM} Quantum Computer Technology Overview. https://www.dwavesys.com/sites/default/files/D-Wave2000QTechCollateral_1029F.pdf, 2018.
[5] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, Vol. 49, No. 12, pp. 1–47, 2008.
[6] NASA Advanced Supercomputing Division. NAS Parallel Benchmarks. https://www.nas.nasa.gov/publications/npb.html.
[7] Hidetoshi Nishimori. *Statistical Physics of Spin Glasses and Information Processing: An Introduction*. 2001.
[8] NTT 先端技術総合研究所. 光を使って難問を解く新しい量子計算原理を実現――量子ニューラルネットワークの開発. http://www.ntt.co.jp/journal/1701/files/jn20170155.pdf, 2017.
[9] Python Software Foundation. PuLP 1.6.9. https://pypi.org/project/PuLP/.
[10] Ryuichi Sakamoto, Thang Cao, Masaaki Kondo, Koji Inoue, Masatsugu Ueda, Tapasya Patki, Daniel Ellsworth, Barry Rountree, and Martin Schulz. Production Hardware Overprovisioning: Real-World Performance Optimization Using an Extensible Power-Aware Resource Management Framework. *Proceedings - 2017 IEEE 31st International Parallel and Distributed Processing Symposium, IPDPS 2017*, pp. 957–966, 2017.
[11] Osman Sarood, Akhil Langer, Abhishek Gupta, and Laxmikant Kale. Maximizing Throughput of Overprovisioned HPC Data Centers under a Strict Power Budget. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, Vol. 2015-Janua, No. January, pp. 807–818, 2014.
[12] TOP500.org. Intrepid - Blue Gene/P Solution. https://www.top500.org/system/176322.
[13] 革新的研究開発推進プログラム（ImPACT）．量子人工脳を量子ネットワークでつなぐ 高度知識社会基盤の実現. http://www.jst.go.jp/impact/program/12.html, 2018.
[14] 株式会社日立製作所. 組合せ最適化問題に特化したクラウド型計算サービスの無償提供を開始-アニーリングマシンの普及と技術者育成の加速をめざす-. https://www.hitachi.co.jp/New/cnews/month/2018/09/0919.html, 2018.
[15] 富士通株式会社．量子コンピュータを実用性で超える新アーキテクチャーを開発. http://pr.fujitsu.com/jp/news/2016/10/20-1.html.