Vol.2020-HPC-176 No.2
2020/9/25

投機を用いた擬似焼きなまし法の分散並列処理手法

鈴木 基己[1,a)]　鷹野 芙美代[1,b)]　井手口 裕太[1,c)]　荒木 拓也[1,d)]

概要：本稿では，擬似焼きなまし法の分散並列処理手法を提案し，イジングモデルにより定式化された大規模な組合せ最適化問題に適用した結果を報告する．本手法では，疑似焼きなまし法の特徴である状態更新の逐次性を維持しつつ，ボトルネックとなる評価関数の差分計算を投機的に実行することで，分散処理に伴う通信処理を隠蔽し，効率的な分散並列処理を実現している．5 万変数から 20 万変数までのイジングモデルに本手法を適用し，本手法が弱スケーラブルであることを確認した．また，10 万変数の問題では従来の SA に対して同程度の求解精度となった結果を報告する．

Parallel distributed processing of simulated annealing via speculative approach

Abstract: We report a methodology to parallelize simulated annealing (SA) on a distributed system. We focus on the Ising model as huge combinatorial optimization problems to be solved. The novelty of the present method lies in a speculative approach where communications associated with distributed processing get hidden. The dynamics of search does not change from the original SA, but the present method provides more effiective ways to parallelize SA. In this report, 50,000-spin to 200,000-spin Ising model are solved, and as a result we report the present method has a weak scalability. For 100,000-spin Ising model, it turns out that an approximate solution of the present method is the same precision as the original SA.

1. はじめに

現代社会における生産性向上の課題の多くは，様々な選択肢の組合せの中から最適なものを求める組合せ最適化問題に帰着される．例えば金融ポートフォリオ計画，工場生産計画，人員シフト計画などの社会課題が組合せ最適化問題とみなすことができる．一般に，問題の規模に対して組合せの数は指数的に，あるいはそれ以上に増大するため，巨大な問題を扱うには高速化の工夫が重要である．

多様な組合せ最適化を汎用的に扱うために、問題をイジングモデルという形式に変換して解く手法が知られている [1]．様々な最適化手法がイジングモデルをベースに考案されており，ソルバの汎用性を担保するためにはイジングモデルを扱うことが重要である [2], [3], [4], [5].

また，組合せ最適化問題の近似的解法として擬似焼きなまし法 (SA) [6] を用いて解く手法がよく用いられている．この手法では問題規模が増大するにつれてメモリ使用量も増大し，巨大な問題の処理には分散メモリシステムを用いた並列化が必要である．一方で，SA の動作原理として探索時の状態更新を逐次に行う必要があるため，メモリ分散下では状態を更新するたびに通信が発生し，データ通信のレイテンシ及び待ち時間がボトルネックとなり効率的な並列化は難しいという課題がある．実際，SA をアルゴリズムとハードウェアの両面から高速化する複数のアプローチ [7] が提案されているものの，並列化による高速化はメモリ共有を前提としており，依然としてメモリ分散下では状態更新ごとに通信が必要であり並列数に対して性能がスケールしない [2].

本稿では、SA の動作の一部を投機実行することで通信の回数を削減し，また通信の待ち時間を隠蔽する並列化手法を提案する．結果として、本手法によりデータ通信のボトルネックが解消され，メモリ分散下でも並列数に対して性能が十分にスケールする結果を得た．

本稿は以下のような構成である．2 章では SA の動作原理について説明する．3 章では提案手法の動作原理を説明

[1] 日本電気株式会社
NEC Corporation
[a)] suzumoto@nec.com
[b)] f-takano@nec.com
[c)] ideguchi@nec.com
[d)] takuya_araki@nec.com

する．4 章では提案手法の評価方法，5 章では提案手法を評価した環境，6 章では提案手法の評価結果について記述する．最後に 7 章にて本稿をまとめを述べる．

2. 擬似焼きなまし法

SA の動作についてイジングモデルに適用する場合を例として説明する．

2.1 イジングモデル

$+1$ または -1 のいずれかの値を取る N 要素の変数 (スピン)s と，2 つのスピン間の相互作用を表す $N \times N$ 行列 J，1 体の相互作用 h によって系のエネルギー H が

$$H = \sum_{i<j} J_{ij} s_i s_j + \sum_i h_i s_i \tag{1}$$

と書けるような磁性体のモデルをイジングモデルと呼ぶ．以下本稿では 1 体の相互作用は扱わず，$h = 0$ として第 2 項を無視する．

上向きのスピンを $+1$，下向きのスピンを -1 に対応させた可視化表現も良く用いられる (図 1)．行列 J は各スピンの結合 J_{ij} が正なら，スピン i と j は反対向きのときにエネルギーを減少させ，J_{ij} が負なら同方向の時にエネルギーを減少させるものとして理解できる．

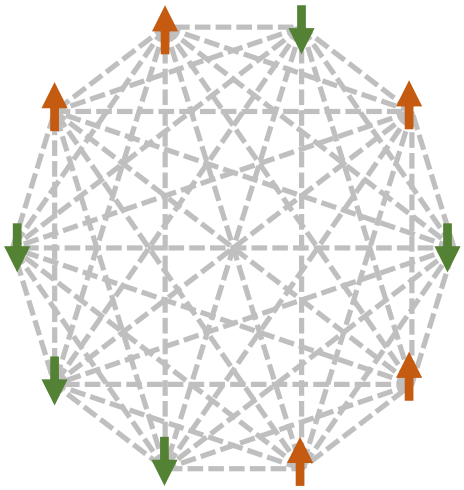

図 1 イジングモデルの表現．上向きの矢印を $+1$ 下向きの矢印を -1 のスピンに対応させている．スピンの間にひかれている線は結合 J_{ij} を表す．

組合せ最適化問題としては，イジングモデルのエネルギーを最小にするようなスピンの向きの組合せを求める問題として利用される．すなわち，行列 J が与えられて

$$\arg\min_s \sum_{i<j} J_{ij} s_i s_j \tag{2}$$

を求める問題がイジングモデルによって定式化された組合せ最適化問題である．

2.2 擬似焼きなまし法の動作

SA は組合せ最適化問題を近似的に解く手法である．以下でイジングモデル型の組合せ最適化問題に適用した場合の SA の動作を説明する．擬似焼きなまし法はマルコフ連鎖モンテカルロ法に基づいており，下記はメトロポリス法による状態更新の動作である [8], [9].

SA では，現在の状態から次の状態へ更新するときに，エネルギーの値が改善する (小さくなる) 場合には状態の更新を必ず受理し，エネルギーの値が悪化する (大きくなる) 場合には状態の更新を確率的に受理する．この確率を制御するパラメータとして温度 T が導入され，探索の初期には温度を高く設定し，エネルギーの値が悪化するような更新も積極的に受理する．探索が進むにつれて温度を低くしていき，徐々にエネルギーの悪化を許さないようにすると，いずれ状態の更新が停止し，その時点において高い確率で最適値に近い値が得られている．

SA の 1 ステップの動作を Algorithm 1 に示す．J は問題として入力される行列，β は逆温度 ($\beta = 1/T$) である．

Algorithm 1 擬似焼きなまし法 (1 ステップ)

1: スピン番号 i をランダムに選択
2: $\Delta H_i = -2 s_i \sum_{j \neq i} J_{ij} s_j$ を計算
3: 確率 $\min\{\exp(-\beta \Delta H_i), 1\}$ でスピン更新 $s_i \leftarrow -s_i$ を実行
4: 1 に戻る

Algorithm 1 の 2 行目ではスピン s_i が $-s_i$ に更新されたときの H の差分を計算している．上のステップを繰り返しながら，逆温度 β を徐々に上げていく．逆温度 β が十分に高くなるとスピンの更新が実行される確率が下がっていき，スピンの更新が行われなくなる．その時点で計算を終了し，そこまでに探索したスピン配位 s のうち H が最小となるものを出力する．または，あらかじめ繰り返しの回数を定めて，β を徐々に上げながら探索を進めていく．

問題ごとに β のスケジュールや繰り返しの数をチューニングする必要があるが，十分にゆっくりと β を上昇させる場合に確率 1 で最適解が発見されることが保証されている [10]．実際には現実的な時間で最適解を発見することは難しいが，β をうまくスケジュールすることで現実的な時間内に最適解に近い近似解を得ることができる．現実的な課題においては最適性が重要でないことが多く，近似解で十分な場合に組合せ最適化問題を解く強力な手法として SA が広く用いられている．

一方で，問題が巨大な場合にはシステムが提供するメモリ容量に行列 J が乗りきらず，メモリ分散しなければならない状況では SA の動作原理である逐次性が問題となる．Algorithm 1 の 2 行目の差分計算は 3 行目のスピン更新の結果を待たなければならず，1 から 3 行目を素朴に並列化することはできない．メモリ分散下では特に通信の頻度が重大な問題で，各プロセスでスピンを並列に更新しようと

すると，各スピンの更新ごとに通信が必要となり，通信時間がボトルネックとなってしまう．実際、Algorithm 1 のような最も素朴な SA よりも洗練された実装 [7] が考案されているものの，並列化のための工夫ではメモリ共有下での高速化に工夫がなされるのみで，スピン更新の並列化は行われておらず，メモリ分散下では状態を更新するたびに通信が発生し並列化効率は著しく悪化する．

上記のようなメモリ分散下での並列化での通信によるボトルネックの課題を解決すべく次章以下でメモリ分散下でも効率よく並列動作する SA を提案する．

3. 提案手法

3.1 投機処理

前節での議論のように，エネルギーの差分 ΔH_i の値は逐次的に更新されるスピン s に依存しているため，素朴にはスピンの更新を並列化することはできない．そこで，エネルギー差分 ΔH_i 計算とスピンの状態更新の処理を分離し，差分計算を状態更新に先立って投機実行する．すなわち p 個の差分計算 ΔH_i を先に実行し，次に投機計算したエネルギー差分 ΔH_i を過去のスピン更新に応じて修正しながら逐次にスピンの更新を行う．これにより，投機計算したエネルギー差分 ΔH_i を修正するための演算量が増加するが，これは高々 $O(p)$ の処理である．したがってスピン数 N に対して投機の個数 p が十分に小さい範囲では追加した演算量の影響なく，効率的に並列化することができる．この動作を Algorithm 2 に示す．6 行目の V はスピン更新が行われたスピンのインデックスを保存するメモリである．

Algorithm 2 投機 SA(p ステップ)

```
1: p_0 ← 1
2: V ← φ
3: for i = 0 to p − 1 do
4:     k ← (i + p_0) mod N
5:     ΔH_k = −2s_k Σ_{j≠k} J_kj s_j を計算
6: end for
7: for i = 0 to p − 1 do
8:     for v ∈ V do
9:         ΔH_i− = 4.0 J_iv s_i s_v
10:    end for
11:    k ← (i + p_0) mod N
12:    確率 min(exp(−βΔH_k), 1) でスピンの更新 s_k ← −s_k を
       実行
13:    if スピン更新が実行された？ then
14:        V ← V ∪ {k}
15:    end if
16: end for
17: p_0 ← p_0 + p mod N
18: 3 へ戻る
```

Algorithm 2 は本質的には Algorithm 1 を p ステップ行っているに過ぎないが，p 個分の差分計算 (4 行目) がスピン更新 (12 行目) よりも先に実行されるため，その修正 (9 行目) が必要となる．

さて，ここで Algorithm 2 全体の分散処理を考える．すなわち，Algorithm 2 を D 個のプロセスが実行し，合計で Dp 個のスピン更新を試行するような場合の動作を考える．行列 J のみが $O(N^2)$ のメモリ使用量で他は全て高々 $O(N)$ であることから，行列 J のデータのみ各プロセスに分散させればよい．スピン s については，全てのプロセスで全スピンのデータ領域を確保し，各プロセスが更新したスピンのインデックスを他のプロセスに転送することで，全プロセスでスピン s の一貫性を保つ．

これにより，Algorithm 1 で分散処理するには 1 ステップごとにスピン更新の有無を通信しなければ s の一貫性が担保できない一方で，Algorithm 2 の分散処理では p ステップに 1 回の通信回数に削減することができる．次節でその動作の詳細を説明する．

3.2 分散処理

提案手法における行列 J の各プロセスへの割り当てを図 2 に示す．

区間 $[1, N]$ を $[1 = a_1, b_1], [b_1 + 1 = a_2, b_2], \cdots, [a_D, b_D = N]$ の D 個の区間に分割し，d $(1 \leq d \leq D)$ 番目のプロセスには $J_{a_d,1}$ から $J_{b_d,N}$ までの横長の矩形部を割り当てる．ここで，イジングモデルの定義 (1) に出現しない J の下三角部は J を対称行列 $J_{ji} = J_{ij}$ とすることで定義する．また，以下では簡単のため対角要素は $J_{ii} = 0$ とする．

Algorithm 2 の分散処理における d 番目のプロセス $(1 \leq d \leq D)$ の動作を Algorithm 3 に示す．Algorithm 3 の 8 行目で他プロセスが更新したスピンの番号を受信し，22 プロセスで送信する．Algorithm 3 で使用される通信はすべて 1 対 1 通信であり，図 3 のようにパイプライン状に各プロセス間の通信をずらして処理することで，通信時間

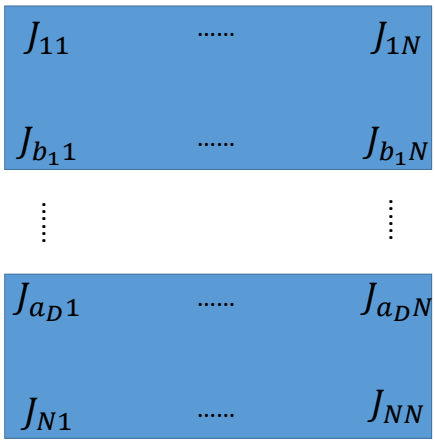

図 2 行列 J の分散方式．全体が $N \times N$ の正方行列となっており，横方向に分割して上から順に各プロセスのメモリに割り当てる．

Algorithm 3 メモリ分散版 SA (p ステップ, d 番目のプロセス)

1: $p_0 \leftarrow 1$
2: num $\leftarrow 0$
3: $V \leftarrow \phi$
4: **for** $i = 0$ **to** $p-1$ **do**
5: $\quad k = (i + p_0 \bmod (b_d - a_d)) + a_d$
6: $\quad \Delta H_k = -2s_k \sum_{j \neq k} J_{kj} s_j$ を計算
7: **end for**
8: V を $d-1 \bmod D$ から受信
9: V のうち最初の num 個を削除
10: num $\leftarrow 0$
11: **for** $i = 0$ **to** $p-1$ **do**
12: $\quad k = (i + p_0 \bmod (b_d - a_d)) + a_d$
13: $\quad$ **for** $v \in V$ **do**
14: $\quad\quad \Delta H_k -= 4.0 J_{kv} s_k s_v$
15: $\quad$ **end for**
16: $\quad$ 確率 $\min(\exp(-\beta \Delta H_k), 1)$ でスピンの更新 $s_k \leftarrow -s_k$ を実行
17: $\quad$ **if** スピン更新が実行された？ **then**
18: $\quad\quad V \leftarrow V \cup k$
19: $\quad\quad$ ++num
20: $\quad$ **end if**
21: **end for**
22: V を $d+1 \bmod D$ へ送信
23: $p_0 \leftarrow p_0 + p \bmod (b_d - a_d)$
24: 4 に戻る

を隠蔽することができる．また，SA の動作中で最も演算回数の多い 6 行目の差分計算が並列に動作し，Algorithm 1 に対し全体の処理時間を短縮することができる．

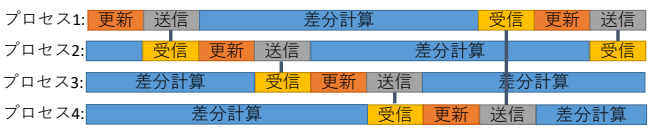

図 3 4 並列での Algorithm 3 の動作概念図．左から右に時間が進んでいる．受信と送信を結ぶ線は通信路を表す．差分計算は並列化されているが，更新部分は逐次に行われていることに注意．これにより SA の本来の動作が保証されたうえで計算を並列化することができる．

理想的には図 3 のように差分計算部分とそれ以外の部分が完全にオーバーラップする．ここで，図 3 における各ブロックの演算量のオーダーは，表 1 のようになっている．

表 1 図 3 の各ブロックの演算オーダー

差分計算	$O(Np)$
送受信	$O(pD)$
更新	$O(p^2 D)$

図 3 から，受信待ちの時間なく並列に実行するには差分計算部の時間と，送受信及び更新の時間の D 倍が釣り合う必要がある．すなわち差分計算を投機実行する個数 p とスピン数 N，並列数 D の間には $p^2 D^2 \sim Np$ の関係が必要である．これがバランスしない状況では通信の受信待ちが発生し並列化効率が悪くなる．すなわち与えられた問題サイズ N と処理に使用する並列数 D によって，Algorithm 3 で使われるパラメータ p を調整する必要がある．

表 1 では通信のレイテンシが考慮されていないが，p が小さい領域では通信の回数が多くなり，レイテンシが全体の処理時間に占める割合が増大する，この効果も p のチューニングに影響を及ぼす．

4. 評価方法

提案手法を $(-1, 1)$ からの一様分布から sampling した Random Matrix を対象として評価する．

最初に本手法特有のチューニングパラメータ p についての挙動を調べる．次に，スケーリングについてスピン数 N が 5 万, 10 万, 20 万 の 3 問題を扱い，並列数に対する計算時間の変化を示す．最後に精度についての指標として，通常の SA との比較を行う．

5. 評価環境

SA のカーネル演算はベクトル同士の内積であり，これは演算量とメモリアクセス回数が同じオーダーとなるためメモリネック処理であるといえる．したがって高い性能を引き出すためにはメモリ帯域性能の良い環境が必要となる．そのため，本手法の評価には高いメモリ帯域幅を持つ SX-Aurora TSUBASA を用いた．SX-Aurora TSUBASA は x86 CPU として Intel Xeon を，ベクトルプロセッサを内蔵したアクセラレータとしてベクトルエンジン (VE) を搭載しており，1VE あたりの最大メモリ帯域幅は 1.22 TB/s である．装置の諸元を表 2 に示す．

表 2 性能評価環境

	SX-Aurora TSUBASA (A311)	
	Xeon Gold 6226 (Cascade Lake)	Vector Engine (VE) Type 10BE
コア仕様		
動作周波数	2.7 GHz	1.408 GHz
理論演算性能 *	224.0 GFLOPS	608 GFLOPS
平均メモリ帯域	10.6 GB/s	153.6 GB/s
プロセッサ仕様		
コア数	12 /socket	8 /card
理論演算性能 *	1.76 TFLOPS	4.86 TFLOPS
最大メモリ帯域	0.12 TB/s	1.35 TB/s
キャッシュ容量	19.25 MB	16.00 MB
メモリ容量	192GB**	48GB

* 単精度　** ホストメモリ

今回用いた環境では，PCIe 3.0 によって 8 枚の VE が接続されたシステムを用いて評価を行った．したがって 8 並列より大きな並列数での計算の場合 1 枚の VE 内の通信と PCI を通じた VE 間の通信が混在することになる．また，

通信に用いたソフトウェアは NEC MPI 2.9.0 である．

6. 性能評価

6.1 p の選び方

3.2 での p は通信頻度を制御するパラメータで，最適な p を選ぶことで全体の処理時間を軽減することができる．10 万スピンの問題に対して全スピンの更新を試行するプログラムを，4VE/4 並列の環境で p を変えて実行し、処理時間を計測した．時間の計測は一つのプロセスに対して行い，主となる 4 つの処理，投機的な差分の計算部分，通信部分 (受信/送信)，差分を投機したことによる修正部分についての積み上げ棒グラフを図 4 に示す。

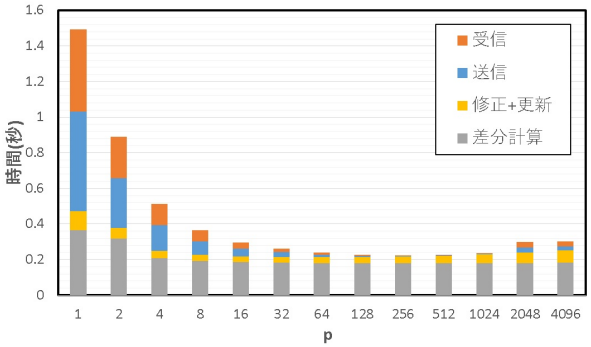

図 4 p のチューニング図．計算は 4VE 4 並列で 10 万スピンの問題に対して行われている．横軸は p 縦軸はそれぞれの p に対する各処理の時間を表す．

図 4 から，p の小さい領域と大きい領域で全体の計算時間が伸びていることがわかる．p が小さい領域では全体の処理時間に対して通信が占める割合が高くなり，p が大きい領域では修正計算が長くなることで，図 3 で現れる差分計算と通信/修正 (更新) 部分とのバランスがとれなくなり，並列化効率が悪化する．一方で最適となる p の領域は 128 から 1024 と広く，p に対する全体の計算時間の感度は高くない．そのため実用上は厳密に最適な p を定める必要はなく，p について数回本手法を試すことで十分効率的な p を得ることができる．

6.2 スケーリング

5 万スピン，10 万スピン，20 万スピンの問題に対し提案手法を適用した際の並列数に対するスケーリングを図 5,6,7 にそれぞれ示す．それぞれの処理は β を 0.0 から 0.05 までリニアに変化させ，$p = 200$ に固定して，全スピンの更新を 200 回試行している．

それぞれの問題サイズで並列数が大きくないときに台数効果が得られていることがわかる．5 万スピン問題では 8 並列，10 万スピンでは 16 並列，20 万スピンでは 32 並列程度まで全体の処理時間がスケールしている．問題サイズが大きくなるにつれて並列数を増やすことができ，さらに巨大な問題に対しても処理時間を短縮できることが期待される．

問題サイズに対して並列数が大きい場合には図 3 のように差分計算とそれ以外の部分がバランスせずに，通信の待ち時間が発生し，通信時間が長くなっている．

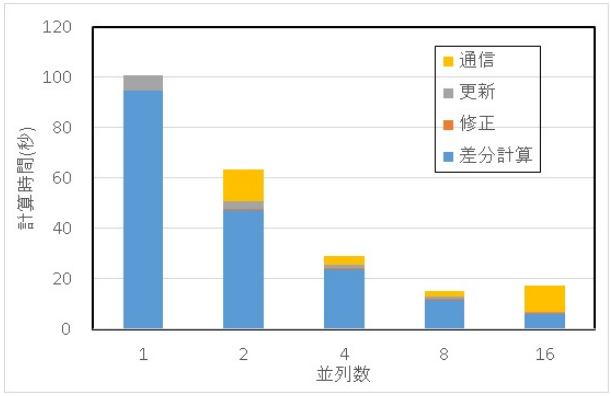

図 5 5 万スピンの問題に対する提案手法のスケーリング．横軸は並列数，縦軸は各処理の経過時間を表す．1 プロセスでは通信が起きないため通信時間が 0 となることに留意されたい。

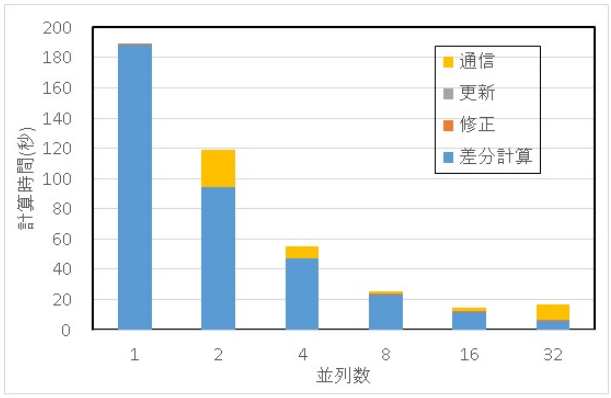

図 6 10 万スピンの問題に対する提案手法のスケーリング

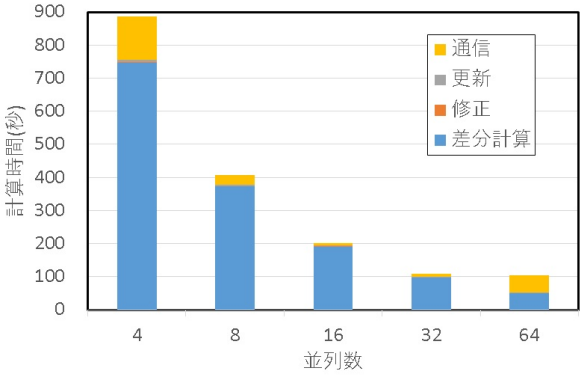

図 7 20 万スピンの問題に対する提案手法のスケーリング．1 並列及び 2 並列のデータが存在しないのは，20 万スピンの問題ではメモリ消費量が 160 GB で，VE につき 48 GB メモリのため最低でも 4VE が必要となるから．

6.3 解精度評価

素朴な SA との比較により本手法の解の精度を検討する．比較に用いた問題は 10 万スピン問題で，素朴 SA では 1VE の逐次実行，提案手法では 4VE 4 並列で動作させている．なお，素朴な SA は Algorithm 1 の動作を素直に逐次実行した動作である。

素朴 SA と提案手法について，計算時間とそれまでに得られた最良解のエネルギーをプロットし比較した結果を図 8 に示す．

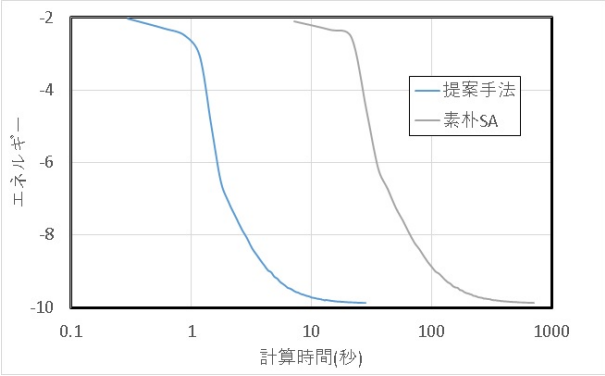

図 8 素朴 SA と提案手法の比較．横軸は計算時間，縦軸はエネルギーで $\times 10^6$ が省略されている．

どちらの手法によっても同程度のエネルギー値で解が収束し，本手法が素朴 SA からアルゴリズムを変更したことによる解精度への影響は本稿での評価範囲では確認されなかった．解精度への影響を含めても本手法は従来 SA に対して 10 倍強高速となる結果となった．

並列数の 4 倍以上に高速となった一つの理由として，提案手法では p 個の差分計算をまとめて行っているため，差分計算を逐次で行う素朴な SA に対して VE のベクトル演算の効率が上がったことが考えられる．

7. まとめ

本稿ではメモリ分散下でも並列数に対して性能がスケールする SA を提案した．本稿での提案手法を 5 万,10 万,20 万スピンの問題に適用したところ，それぞれ 8,16,32 並列まで性能がスケールすることを確認した．これらの結果から，問題サイズが巨大になるにつれ適用可能な並列数もスケールしており，本手法が弱スケーラブルであることが示唆される．

また，通信がボトルネックとなる従来の SA と比較すると，10 万スピンの問題に対して同程度の精度で解が得られた．

参考文献

[1] Lucas, A.: Ising formulations of many NP problems, *Frontiers in Physics*, Vol. 2, p. 5 (online), DOI: 10.3389/fphy.2014.00005 (2014).
[2] Goto, H., Tatsumura, K. and Dixon, A. R.: Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems, *Science Advances*, Vol. 5, No. 4 (online), DOI: 10.1126/sciadv.aav2372 (2019).
[3] Okuyama, T., Sonobe, T., Kawarabayashi, K.-i. and Yamaoka, M.: Binary optimization by momentum annealing, *Phys. Rev. E*, Vol. 100, p. 012111 (online), DOI: 10.1103/PhysRevE.100.012111 (2019).
[4] Johnson, M. W., Amin, M. H. S., Gildert, S., Lanting, T., Hamze, F., Dickson, N., Harris, R., Berkley, A. J., Johansson, J., Bunyk, P., Chapple, E. M., Enderud, C., Hilton, J. P., Karimi, K., Ladizinsky, E., Ladizinsky, N., Oh, T., Perminov, I., Rich, C., Thom, M. C., Tolkacheva, E., Truncik, C. J. S., Uchaikin, S., Wang, J., Wilson, B. and Rose, G.: Quantum annealing with manufactured spins, *Nature*, Vol. 473 (online), DOI: 10.1038/nature10012 (2011).
[5] Durkin, G. A.: Quantum speedup at zero temperature via coherent catalysis, *Phys. Rev. A*, Vol. 99, p. 032315 (online), DOI: 10.1103/PhysRevA.99.032315 (2019).
[6] Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P.: Optimization by Simulated Annealing, *Science*, Vol. 220, No. 4598, pp. 671–680 (online), DOI: 10.1126/science.220.4598.671 (1983).
[7] Isakov, S., Zintchenko, I., Rnnow, T. and Troyer, M.: Optimised simulated annealing for Ising spin glasses, *Computer Physics Communications*, Vol. 192, pp. 265 – 271 (online), DOI: https://doi.org/10.1016/j.cpc.2015.02.015 (2015).
[8] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E.: Equation of State Calculations by Fast Computing Machines, *The Journal of Chemical Physics*, Vol. 21, No. 6, pp. 1087–1092 (online), DOI: 10.1063/1.1699114 (1953).
[9] Hastings, W. K.: Monte Carlo sampling methods using Markov chains and their applications, *Biometrika*, Vol. 57, No. 1, pp. 97–109 (online), DOI: 10.1093/biomet/57.1.97 (1970).
[10] Birkhoff, G. D.: Proof of the Ergodic Theorem, *Proceedings of the National Academy of Sciences*, Vol. 17, No. 12, pp. 656–660 (online), DOI: 10.1073/pnas.17.2.656 (1931).