日本オペレーションズ・リサーチ学会
2022年 春季研究発表会

イジング最適化を用いた二次割当問題の高速求解

富士通株式会社　*神田　浩一　KANDA Kouichi
DXR 研究所　田村　泰孝　TAMURA Hirotaka
トロント大学　Mohammad Bagherbeik, Parastoo Ashtari
Seyed Farzad Mousavi, Ali Sheikholeslami

1. はじめに

近年、半導体の微細化による指数関数的なコンピュータ性能向上トレンドが鈍化する中、イジング最適化専用ハードウェアを用いて、組合せ最適化問題を高速に解く取組みが盛んになっている[1,2]. 特にデジタルアニーラ[2](以下 DA)では、全結合した 8k ビット変数に対応し、MaxCut 等の問題で既存のソルバよりも良い求解性能が得られている. 本稿では[2]のアルゴリズムを拡張し、マルチコア CPU 上で二次割当問題の求解を行った結果について報告する.

2. 定式化

2.1 DA のイジング最適化アルゴリズム

目的関数(又はエネルギ)$E(\boldsymbol{x})$はバイナリ変数$\boldsymbol{x} = (x_1, \dots, x_N)$を用いて以下のイジング形式で表せる. 重み行列$\boldsymbol{W}$は$N$次正方対称行列、$\boldsymbol{b}$は$N$次元ベクトルである.

$$E(\boldsymbol{x}) = -\,{}^t\boldsymbol{x}\boldsymbol{W}\boldsymbol{x} - {}^t\boldsymbol{b}\boldsymbol{x} \tag{1a}$$

式(1a)は展開して以下のようにも書ける.

$$E(\boldsymbol{x}) = -\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}x_i x_j - \sum_{i=1}^{N} b_i x_i \tag{1b}$$

DA の 1 動作ステップでは 1 ビット状態遷移の試行を行い、このステップを繰返しながら、$E(\boldsymbol{x})$がより低くなる状態を探索する. 遷移$x_i \to x_i + \Delta x_i$に対するエネルギ変化ΔE_iは式(2)で、遷移受入れ確率$P(\Delta E_i)$はメトロポリス基準に従って式(3)で求める.

$$\Delta E_i = -h_i \cdot \Delta x_i \ , \quad h_i = \sum_j W_{ij}x_j + b_i \tag{2}$$

$$P(\Delta E_i) = min\left[1,\, exp\left(-\frac{\Delta E_i}{T}\right)\right] \tag{3}$$

ここでTは温度である. 高温では$\Delta E_i > 0$の遷移受入れ確率も高くなる. Simulated Annealing 法のように温度を徐々に下げる場合、低温での局所解脱出が難しいため、DA では温度の異なる複数レプリカを並列実行し、時々レプリカ間で状態を入替える交換モンテカルロ法[3]を用いている. x_iの遷移を行った後は、次の試行に備えて$\boldsymbol{W}$の 1 行を読出し、式(4)に従ってh_jを更新する.

$$h_j \to h_j + W_{ij}\Delta x_i \qquad \left(1 \le {}^\forall j \le N\right) \tag{4}$$

2.2 二次割当問題 (Quadratic Assignment Problem)

QAP は施設や部品の最適配置問題として様々な場面に現れる. 工場配置の例では、各工場間のフローfと各拠点間の距離dが各々$n \times n$行列$\boldsymbol{F}, \boldsymbol{D}$で与えられ、工場$i, j$を拠点$\pi(i), \pi(j)$に配置した時のコストは$f_{i,j} \cdot d_{\pi(i),\pi(j)}$で定義される. 求めたいのは全工場を配置した時の総コスト$E(\pi)$を最小化する順列$(\pi(1), \dots, \pi(n))$である.

$$E(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{i,j}\, d_{\pi(i),\pi(j)} \tag{5}$$

順列πはn^2個の変数$\boldsymbol{x} = \left(x_{i,j}\right)$で表現できる. ただし$x_{i,j}$は工場$i$が拠点$j$に配置された時に限り 1 をとり、それ以外は 0 と定義する. 以下は順列(4,3,1,2)の例である.

$$\boldsymbol{x} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{6}$$

$E(\pi)$は式(7)のように$x_{i,j}$の 2 次式で書直せる. 従って QAP はイジング最適化の枠組みで解くことができる. $\boldsymbol{F}, \boldsymbol{D}$が対称行列の時、$n^2 \times n^2$行列$\boldsymbol{W}$は式(8)で求まる.

$$E(\boldsymbol{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{i,j}\, d_{k,l}\, x_{i,k}\, x_{j,l} \tag{7}$$

$$\boldsymbol{W} = 2\begin{pmatrix} \boldsymbol{0} & f_{12}\boldsymbol{D} & \cdots & f_{1n}\boldsymbol{D} \\ f_{12}\boldsymbol{D} & \boldsymbol{0} & \cdots & f_{2n}\boldsymbol{D} \\ \vdots & \vdots & \ddots & \vdots \\ f_{1n}\boldsymbol{D} & f_{2n}\boldsymbol{D} & \cdots & \boldsymbol{0} \end{pmatrix} \tag{8}$$

2.3 従来手法の課題

式(6)の例からも明らかなように、$\boldsymbol{x}$が順列を表すには$n \times n$行列表示した時に、どの行と列も 1 の数は 1 つであるという 1-hot 制約が課される. しかし DA の 1 ステップは 1 ビット遷移であり、遷移過程で順列として意味をなさない状態も取りうるため、式(9)の制約項$P(\boldsymbol{x})$に適当な係数を掛けたものを$E(\boldsymbol{x})$に加算する必要があった.

$$P(\boldsymbol{x}) = \sum_{p=1}^{n}\left(\sum_{q=1}^{n} x_{p,q} - 1\right)^2 + \sum_{q=1}^{n}\left(\sum_{p=1}^{n} x_{p,q} - 1\right)^2 \tag{9}$$

この求解手法は二つの点で不利である．一つ目は探索空間の広さである．順列の種類は高々$n!$個だが、探索対象の空間は、それよりはるかに多い2^{n^2}個も状態がある．二つ目は、制約を満たす状態間には最低でも3つの制約違反状態があり、そこでは$P(\boldsymbol{x})$によってエネルギが増加するため状態遷移が滞る点である．その結果、従来のイジング最適化では大規模なQAPの求解は困難であった．

3. 提案手法

今回、大規模QAPの高速な求解を可能にした三つの要素技術について以下に説明する．

3.1 4ビット遷移試行

2.3節で述べた課題は、制約を満たす状態のみを探索することで解決できる．そのために1ステップでは順列中の2要素を入替える試行を行う．これは工場i,jに関して(工場，拠点)の組合せを(i,k)と(j,l)から、(i,l)と(j,k)に変更することに相当し、4ビット遷移が必要になる．実際は現在の値が0である変数$x_{i,l}$を選ぶと、一緒に反転すべき他の3変数$x_{i,k},x_{j,k},x_{j,l}$は自動的に決まる．

$$\begin{pmatrix} \ddots & \vdots & & \vdots & \\ \cdots & x_{i,k} & \cdots & x_{i,l} & \cdots \\ & \vdots & \ddots & \vdots & \\ \cdots & x_{j,k} & \cdots & x_{j,l} & \cdots \\ & \vdots & & \vdots & \ddots \end{pmatrix} \Bigg| \begin{pmatrix} \ddots & \vdots & & \vdots & \\ \cdots & 1 & \cdots & 0 & \cdots \\ & \vdots & \ddots & \vdots & \\ \cdots & 0 & \cdots & 1 & \cdots \\ & \vdots & & \vdots & \ddots \end{pmatrix} \Rightarrow \begin{pmatrix} \ddots & \vdots & & \vdots & \\ \cdots & 0 & \cdots & 1 & \cdots \\ & \vdots & \ddots & \vdots & \\ \cdots & 1 & \cdots & 0 & \cdots \\ & \vdots & & \vdots & \ddots \end{pmatrix}$$

ΔEの計算とhの更新は式(10)(11)で計算できる．ただし、添え字ikは$n(i-1)+k$を意味する．hの更新は1ビット遷移に対する式(4)の素直な拡張になっている．

$$\Delta E_{il} = h_{ik} + h_{jl} - h_{jk} - h_{il} - \left(W_{ik,jl} + W_{il,jk}\right) \quad (10)$$

$$h_m \rightarrow h_m + W_{m,il} + W_{m,jk} - \left(W_{m,ik} + W_{m,jl}\right) \quad (11)$$

3.2 重み行列の生成

$\boldsymbol{W}$の要素数はn^4で増えるため、大規模な問題では演算器から遠いメモリに配置された$\boldsymbol{W}$の読出し遅延で求解速度が落ちる．そこで$\boldsymbol{W}$全体の配列を持つ代わりに、$\boldsymbol{F},\boldsymbol{D}$を配列として保持し、$\Delta E$の計算と$h$の更新の際に必要な重み成分$W_{ij}$だけを生成することにした．$\boldsymbol{W}$1行分の生成には、式(8)からも分かるように、$\boldsymbol{F},\boldsymbol{D}$から1行ずつ読み出し、値の積を取るだけで良い．

3.3 実行時間の均等化

高温レプリカでは頻繁に状態更新が発生し、hの更新に伴う重み生成分1ステップの実行時間が長くなる．高温レプリカと低温レプリカをペアにして1つのスレッドに割り当てることで、同一ステップ数に対する実行時間がスレッド間で均等になるようにし、レプリカ交換までの間にアイドルなスレッドが発生しないようにした．

4. 性能評価

ベンチマークにはQAPLIB[4]のインスタンスを用いた．プログラムはC++で実装し、CentOS 8.1上のgccでコンパイルした．AMD Threadripper 3990X CPU(64コア,128GB DDR4)上で、OpenMP APIによるスレッド並列計算を行った．100回の計算を行い、既知最低エネルギに到達した時間を平均したものを求解時間として下表に纏めた．既存ソルバに対して、1スレッドで約10倍、64スレッドで約100倍の求解時間短縮を確認できた．

問題	求解時間[秒]			文献	問題	求解時間[秒]			文献
	従来	本手法				従来	本手法		
		1	64				1	64	
sko90	92	8.02	0.49	[5]	tai50b	0.2	1.94	0.11	[6]
sko100a	69	9.6	0.56	[5]	tai60b	0.4	4.06	0.22	[6]
sko100b	45	6.77	0.57	[5]	tai80b	5.5	9.33	0.59	[6]
sko100d	37	9.58	0.84	[5]	tai100b	10.1	6.77	0.48	[6]
sko100e	47	6.14	0.61	[5]	tho40	0.4	1.11	0.25	[5]
sko100f	57	10.8	0.65	[5]	wil100	97	15.5	0.77	[5]

5. まとめ

本稿ではDAのイジング最適化アルゴリズムを拡張し、100拠点規模のQAPを高速に解けることを示した．本手法を用いて巡回セールスマン問題や線形順序問題などの順列最適化問題も求解可能であり、イジング最適化の適用領域拡大が期待できる．

参考文献

[1] Takemoto T. et al. "A 2 ×30k-Spin Multichip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems," ISSCC 2019, pp. 52-53.

[2] Matsubara S. et al. "Digital Annealer for High-Speed Solving of Combinatorial Optimization Problems and Its Applications,", ASPDAC 2020, pp. 667-672.

[3] Hukushima K. and Nemoto K. "Exchange Monte Carlo method and application to spin glass simulations," J. Phys. Soc. Jpn. Vol.65, pp.1604-1608, 1996.

[4] Burkard R.E. et al, "QAPLIP-a quadratic assignment problem library," J. Global Optim. 10(4), 391-403, 1997.

[5] 6. Munera, D. et al, "Hybridization as cooperative parallelism for the quadratic assignment problem", HM 2016. LNCS, vol. 9668, pp. 47-61. Springer.

[6] Tsutsui, S et al, "ACO on multiple GPUs with CUDA for faster solution of QAPs", PPSN 2012. LNCS, vol. 7492, pp. 174-184. Springer.

[7] Mohammad B. et al. "A Permutational Boltzmann Machine with Parallel Tempering for Solving Combinatorial Optimization Problems," PPSN 2020, pp 317-331.