

さまざまなイジング計算機による 組合せ最適化問題の解法と比較

深田 佳佑¹ パリジ マチュー^{1,2} 富田 憲範² 戸川 望¹

概要: 組合せ最適化問題とは、与えられた制約条件を満たしつつ目的関数を最大化または最小化する変数の組合せを探索する問題である。組合せ最適化問題を効率的に解く計算機としてイジング計算機が研究されている。イジング計算機はイジングモデルを用いることで効率よく準最適解を得ることが可能となる。近年、ハードウェアならびにソフトウェアによって実装された、さまざまなイジング計算機が発表されている。本稿では、典型的な組合せ最適化問題として二次割当問題ならびに巡回セールスマン問題、現実的な組合せ最適化問題としてスロット配置問題を取り上げる。イジング計算機として、ハードウェア実装されたイジング計算機、ソフトウェアとハードウェアによって構成されたイジング計算機ソフトウェアシステム、そしてソフトウェア実装されたシミュレーテッドアニーリングベースのイジング計算機シミュレータを用いる。さまざまな種類のイジング計算機で、組合せ最適化問題を解法し比較評価した結果を報告する。

1. はじめに

組合せ最適化問題とは、与えられた制約を満たした上で目的関数を最大化または最小化する変数の組合せを探索する問題である [1-5]。組合せ最適化問題は変数の増加とともに計算量も指数的に増加するため、従来のノイマン型コンピュータでは最適解を得ることが困難な場合がある。そこで、組合せ最適化問題を効率的に解く新しい計算機として、非ノイマン型コンピュータであるイジング計算機が研究されている。イジング計算機とは、統計力学のモデルであるイジングモデルをハードウェア実装した計算機であり、効率的に準最適解を得ることができる。イジング計算機では、組合せ最適化問題を解くためにイジングモデル又はイジングモデルと等価な Quadratic Unconstrained Binary Optimization (QUBO) モデルに組合せ最適化問題をマッピングする。現在では様々な組合せ最適化問題がイジングモデル又は QUBO モデルにマッピングされ、イジング計算機によって計算されている [6-10]。

近年、ハードウェアならびにソフトウェアによって実装された、様々なイジング計算機が発表されている。ハードウェアのみで実装されたイジング計算機をハードウェアイジング計算機、ハードウェアとソフトウェアの二つによって実装されたイジング計算機をイジング計算機ソフトウェアシステムと呼ぶ。ハードウェアイジング計算機やイジ

ング計算機ソフトウェアシステムには数多くの種類があり、様々な組合せ最適化問題を効率的に解くことができる。ハードウェアイジング計算機には、D-Wave の量子アニーリングマシン [11]、コヒーレントイジングマシン [12]、第 2 世代デジタルアニーラ [13] など様々な種類が存在する。イジング計算機ソフトウェアシステムとしては、第 3 世代デジタルアニーラ [14, 15] がある。

本稿では、実社会に応用されやすい組合せ最適化問題として、二次割当問題、巡回セールスマン問題、スロット配置問題の三つを取り上げる。それら三つの組合せ最適化問題を QUBO モデルに落とし込み、ハードウェアイジング計算機、イジング計算機ソフトウェアシステム、ソフトウェア実装されたシミュレーテッドアニーリングベースのイジング計算機シミュレータの三つを用いて解法し、比較評価した結果を報告する。

本稿の貢献は以下の三つである。

1. QUBO モデルにマッピングした二次割当問題を用いて、様々なイジング計算機を評価した。ソフトウェア実装されたイジング計算機シミュレータでは、これまで知られている最良のコストと比較して平均で 222% 大きい解が得られた。ハードウェアイジング計算機では、最良のコストと比較して平均で 7.03% 大きい解が得られた。イジング計算機ソフトウェアシステムでは、全てのインスタンスで最良のコストと同じ解を得ることができた。
2. QUBO モデルにマッピングした巡回セールスマン問題

¹ 早稲田大学大学院基幹理工学研究科情報理工・情報通信専攻

² 富士通 (株) 富士通研究所

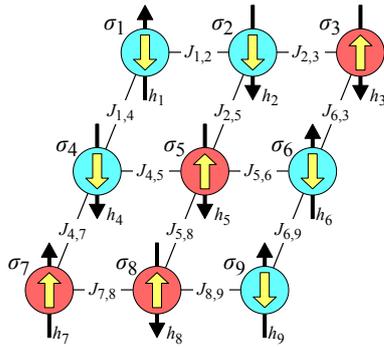


図 1: イジングモデルの例.

を用いて、様々なイジング計算機を評価した。ソフトウェア実装されたイジング計算機シミュレータでは、これまで知られている最良のコストと比較して平均で 362% 大きい解が得られた。ハードウェアイジング計算機では、最良のコストと比較して平均で 141% 大きい解が得られた。イジング計算機ソフトウェアシステムでは、最良のコストと比較して平均で 11.6% 大きい解が得られた。

3. QUBO モデルにマッピングしたスロット配置問題を用いて、様々なイジング計算機を評価した。ハードウェアイジング計算機では、ソフトウェア実装されたイジング計算機シミュレータと比較して重み付き総配線長が平均で 19.6% 小さい解が得られた。イジング計算機ソフトウェアシステムでは、ソフトウェア実装されたイジング計算機シミュレータと比較して重み付き総配線長が平均で 26.2% 小さい解が得られた。

本稿は以下のように構成される。2 章では、イジングモデルおよび QUBO モデルについて説明する。また、様々なイジング計算機として、ハードウェアイジング計算機、イジング計算機ソフトウェアシステムについて紹介する。3 章では、二次割当問題を QUBO モデルにマッピングし、三種類のイジング計算機で評価した結果を報告する。4 章では、巡回セールスマン問題を QUBO モデルにマッピングし、三種類のイジング計算機で評価した結果を報告する。5 章では、スロット配置問題を QUBO モデルにマッピングし、三種類のイジング計算機で評価した結果を報告する。6 章では、本稿を総括する。

2. イジングモデル, QUBO モデルとさまざまなイジング計算機

2.1 イジングモデルと QUBO モデル

イジングモデルとは、物理学の分野の一つである統計力学における基礎モデルである [16], [17]。スピンと呼ばれるミクロな要素が複数集まり、スピン間の相互作用と各々のスピンに働く磁場によって、システム全体でどのような振る舞いを示すかを調べるためのモデルである。イジングモデルは、スピンの配置される頂点集合 V 、及び頂点間の辺

集合 E で構成される無向グラフ $G = (V, E)$ 上で定義される。二つの頂点 i, j が接続されているとき、 $(i, j) \in E$ は頂点 i, j 間の接続辺を表す。頂点 i に配置されるスピンを σ_i とする。 σ_i は ± 1 のいずれかを取り、 $+1$ を上向きのスピン、 -1 を下向きのスピンとする。二つのスピン σ_i, σ_j 間に作用する相互作用係数を $J_{i,j} \in \mathbb{R}$ (\mathbb{R} は実数集合)、スピン σ_i に作用する外部磁場係数を $h_i \in \mathbb{R}$ と定義すると、イジングモデルのエネルギー関数 \mathcal{H} は、次の式 (1) のように表せる。

$$\mathcal{H} = - \sum_{(i,j) \in E} J_{i,j} \sigma_i \sigma_j - \sum_{i \in V} h_i \sigma_i \quad (1)$$

\mathcal{H} は値が小さいほど安定な状態であり、値が最小の状態を基底状態と呼ぶ。イジングモデルの例を図 1 に示す。

イジングモデルでは、スピン σ_i の値が ± 1 の 2 値をとっていたが、Quadratic Unconstrained Binary Optimization (QUBO) モデルと呼ばれるモデルでは、スピンの値が 0 または 1 をとるバイナリ変数 x_i が用いられている。QUBO モデルのエネルギー関数 \mathcal{H}' は、次の式 (2) のように表せる。

$$\mathcal{H} = - \sum_{(i,j) \in E} J'_{i,j} x_i x_j - \sum_{i \in V} h'_i x_i - \text{const} \quad (2)$$

$J'_{i,j}$ は二つのバイナリ変数 x_i, x_j 間に作用する相互作用係数、 h'_i はバイナリ変数 x_i に作用する外部磁場係数、 const は定数である。バイナリ変数 x_i は、式 (3) によってスピン σ_i から変換できる。したがって、イジングモデルと QUBO モデルは等価な関係である。

$$x_i = \frac{\sigma_i + 1}{2} \quad (3)$$

2.2 さまざまなイジング計算機

近年、ハードウェアならびにソフトウェアによって実装されたさまざまなイジング計算機が発表されている。ハードウェアのみで実装されたイジング計算機をハードウェアイジング計算機、ハードウェアとソフトウェアの二つによって実装されたイジング計算機をイジング計算機ソフトウェアシステムと呼ぶ。ハードウェアイジング計算機やイジング計算機ソフトウェアシステムには、数多くの種類がある。

2.2.1 ハードウェアイジング計算機

ハードウェアイジング計算機として、D-Wave 量子アニーリングマシン [11]、NTT のコヒーレントイジングマシン [12]、東芝のシミュレーテッド分岐マシン [18]、富士通の第 2 世代デジタルアニーラ [13]、日立製作所の CMOS アニーリングマシン [19]、Fixstars の Amplify Annealing エンジン [20] など様々なマシンがある。D-Wave 量子アニーリングマシン [11] は、専用の集積回路で構成され、求解アルゴリズムとして量子アニーリングという実際の量

子効果を用いている。扱えるスピン数は最大で 5,000 スピンであるが、スピン間の結合がキメラグラフまたはペガサスグラフという疎結合の状態であるため、全結合の組合せ最適化問題には不向きという特徴がある。コヒーレントイジングマシン [12] は、縮退光パラメトリック発振器をベースとし、求解アルゴリズムとして縮退光パラメトリック発振器の光の振幅を測定する方法を用いている。スピン間の結合は全結合であり、最大で 100,000 スピンを扱うことができる。シミュレーテッド分岐マシン [18] は GPU と Field-Programmable Gate Array (FPGA) で構成され、求解アルゴリズムとしてシミュレーテッド分岐アルゴリズムを用いている。シミュレーテッド分岐アルゴリズムとは、量子分岐現象という量子力学の現象を古典計算機に落とし込んだ古典分岐マシンにおける運動方程式を並列に計算するアルゴリズムである。スピン間の結合は全結合であり、並列計算により最大で 1,000,000 スピンを扱うことができる。第 2 世代デジタルアニーラ [13] とは、専用プロセッサで構成され、求解アルゴリズムとしてマルコフ連鎖モンテカルロ法に基づく確率的探索を用いている。スピン間の結合は全結合であり、最大で 8,192 スピンを扱うことができる。CMOS アニーリングマシン [19] には二つある。一つは専用の集積回路および FPGA で構成されるもので、求解アルゴリズムとしてシミュレーテッドアニーリングを用いている。扱えるスピン数は最大で 144,000 スピンであるが、スピン間の結合がキングスグラフという疎結合の状態であるため、全結合の組合せ最適化問題には不向きという特徴がある。もう一つの CMOS アニーリングマシンは GPU で構成され、求解アルゴリズムとしてモメンタムアニーリングを用いている。スピン間の結合は全結合であり、最大で 100,000 スピンを扱うことができる。Amplify Annealing エンジン [20] は、GPU で構成され、求解アルゴリズムとして GPU によるアニーリングを用いている。スピン間の結合は全結合であり、最大で 65,536 スピンを扱うことができる。

2.2.2 イジング計算機ソフトウェアシステム

イジング計算機ソフトウェアシステムとして、第 3 世代デジタルアニーラ [14,15] がある。第 3 世代デジタルアニーラとは、ソフトウェアとハードウェアの二つでハイブリッド構成される新しいマシンで、最大で 100,000 スピンを扱うことができる。求解アルゴリズムは、マルコフ連鎖モンテカルロ法を実行しつつ、適当なタイミングで隣接温度の探索結果を入れ替えるレプリカ交換法を用いている。本マシンには QUBO モデルのコスト関数と制約項を別々に入力でき、ハードウェアとソフトウェアが協調することにより、探索中に制約項の違反状態を分析することができる。

3. 二次割当問題

本章では、現実用いられる組合せ最適化問題の一つと

して二次割当問題を取り上げ、二次割当問題の定式化および QUBO モデルマッピングについて説明する。また、ハードウェアイジング計算機、イジング計算機ソフトウェアシステム、ソフトウェア実装されたシミュレーテッドアニーリングベースのイジング計算機シミュレータの三つのイジング計算機で二次割当問題を求解した結果を示す。

3.1 二次割当問題の定式化

二次割当問題では、複数の工場と、工場と同数の地区が与えられ、任意の工場間には物流量が、任意の地区間には距離が定義されているとき、各工場間の物流量と工場が割り当てられた各地区間の距離の積の総和が最小となるような、工場の地区への割り当てを求める問題である。二次割当問題は、病院の病室配置 [21] や油圧タービンにおける羽の最適配置 [22] などに応用できる。

N 個の工場の集合を $F = \{f_1, f_2, \dots, f_N\}$ 、工場を割り当てるための N 個の地区の集合を $L = \{l_1, l_2, \dots, l_N\}$ とする。任意の工場間には物流量が定義されており、工場 f_i, f_j ($1 \leq i, j \leq N$) 間の物流量を $w(f_i, f_j)$ とする。また、任意の地区間には距離が定義されており、地区 l_k, l_l ($1 \leq k, l \leq N$) 間の距離を $d(l_k, l_l)$ とする。工場 f_i が割り当てられた地区を $l(f_i)$ 、工場 f_j が割り当てられた地区を $l(f_j)$ とすると、二つの工場 f_i, f_j 間の距離は $d(l(f_i), l(f_j))$ となる。したがって、各工場間の物流量と工場が割り当てられた各地区間の距離の積の総和 C は、次の式 (4) のように表せる。

$$C = \sum_{i=1}^{N-1} \sum_{j=i+1}^N w(f_i, f_j) d(l(f_i), l(f_j)) \quad (4)$$

なお、各工場はただ一つの地区にのみ必ず配置するものとし、これを工場配置制約と呼ぶ。また、各地区にはただ一つの工場が割り当てられるものとし、これを地区内工場制約と呼ぶ。工場配置制約と地区内工場制約を合わせて二次割当制約と呼ぶ。以上のもと、二次割当問題を次のように定義する。

定義 1. N 個の工場があり、各工場間には物流量が定義されている。また、工場を割り当てるための N 個の地区があり、各地区間には距離が定義されている。各工場間の物流量と工場が割り当てられた各地区間の距離の積の総和を C とする。二次割当問題とは、工場配置制約と地区内工場制約で構成される二次割当制約を満たした上で、 C を最小化するような工場の地区への割り当てを求める問題である。

3.2 QUBO モデルマッピング

本節では、3.1 節で定義した二次割当問題を QUBO モデルにマッピングする手法について説明する。まず工場の数を N 、 i 番目の工場を f_i ($1 \leq i \leq N$)、 k 番目の地区を l_k ($1 \leq k \leq N$) とし、バイナリ変数 $x_{i,k}$ を次の式 (5) のよ

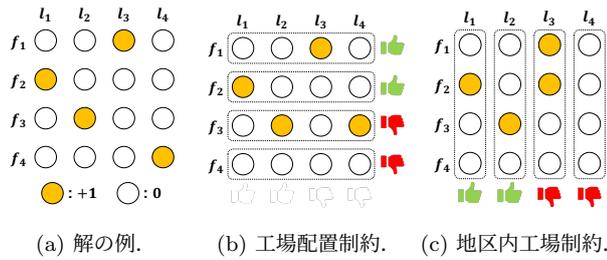


図 2: 工場数 $N = 4$ における解の例と制約条件.

うに定義する.

$$x_{i,k} = \begin{cases} 1 & (\text{工場 } f_i \text{ を地区 } l_k \text{ に配置するとき}) \\ 0 & (\text{上記以外}) \end{cases} \quad (5)$$

工場数 $N = 4$ の場合におけるバイナリ変数 $x_{i,k}$ を用いた二次割当問題の解の例を図 2(a) に示す. オレンジ色の丸はバイナリ変数 $x_{i,k}$ が 1, 白色の丸はバイナリ変数 $x_{i,k}$ が 0 であることを表す.

3.2.1 コスト関数

コスト関数は, 各工場間の物流量と工場が配置された各地区間の距離の積の総和であり, 3.1 節で定義した式 (4) によって与えられる. バイナリ変数 $x_{i,k}, x_{j,l}$, 二つの工場 f_i, f_j 間の物流量 $w(f_i, f_j)$, 二つの地区 l_k, l_l 間の距離 $d(l_k, l_l)$ を用いて, コスト関数 \mathcal{H}_A は次の式 (6) のように表せる.

$$\mathcal{H}_A = \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \sum_{l=1}^N w(f_i, f_j) d(l_k, l_l) x_{i,k} x_{j,l} \quad (6)$$

\mathcal{H}_A の最小値を h_a とする. h_a は問題の入力に依存する.

3.2.2 工場配置制約

工場配置制約とは, i 番目の工場 f_i はただ一つの地区にのみ配置するという制約である. 例を図 2(b) に示す. 赤色のマークは本制約に違反していること, 緑色のマークは本制約を満たしていることを意味する. 本制約を i 番目の工場 f_i について定式化すると, 次の式 (7) のように表せる.

$$\sum_{k=1}^N x_{i,k} = 1 \quad (1 \leq i \leq N) \quad (7)$$

全ての工場 f_i が式 (7) を満たすときに最小値をとるようなエネルギー関数 \mathcal{H}_B を導入すると, 次の式 (8) のように表せる.

$$\mathcal{H}_B = \sum_{i=1}^N \left(\sum_{k=1}^N x_{i,k} - 1 \right)^2 \quad (8)$$

\mathcal{H}_B の最小値は 0 をとる.

3.2.3 地区内工場制約

地区内工場制約とは, k 番目の地区 l_k にはただ一つの工場が配置されているという制約である. 例を図 2(c) に示す. 赤色のマークは本制約に違反していること, 緑色のマークは本制約を満たしていることを意味する. 本制約を

k 番目の地区 l_k について定式化すると, 次の式 (9) のように表せる.

$$\sum_{i=1}^N x_{i,k} = 1 \quad (1 \leq k \leq N) \quad (9)$$

全ての地区 l_k が式 (9) を満たすときに最小値をとるようなエネルギー関数 \mathcal{H}_C を導入すると, 次の式 (10) のように表せる.

$$\mathcal{H}_C = \sum_{k=1}^N \left(\sum_{i=1}^N x_{i,k} - 1 \right)^2 \quad (10)$$

\mathcal{H}_C の最小値は 0 をとる.

3.2.4 QUBO モデルのエネルギー関数

本項で紹介した $\mathcal{H}_A, \mathcal{H}_B, \mathcal{H}_C$ の三つのエネルギー関数を重み付き和で表すと, 最終的なエネルギー関数 \mathcal{H} は次の式 (11) のように表せる.

$$\mathcal{H} = \mathcal{H}_A + \alpha(\mathcal{H}_B + \mathcal{H}_C) \quad (11)$$

$\alpha (> 0)$ はハイパーパラメータである. エネルギー関数 \mathcal{H} は最小値 h_a を取り, このとき基底解となる. 基底解が得られたときのスピニングが最適解となる.

3.3 イジング計算機による解法

本節では, QUBO モデルにマッピングした二次割当問題を, ソフトウェア実装されたイジング計算機シミュレータ, ハードウェアイジング計算機, イジング計算機ソフトウェアシステムの三種類で実験した結果を示し考察する.

3.3.1 実験環境

インスタンスは QAPLIB [23] を使用した. イジング計算機には様々な種類があるが, 本実験ではソフトウェア実装されたイジング計算機シミュレータ, ハードウェアイジング計算機, イジング計算機ソフトウェアシステムを用いた.

ソフトウェア実装されたイジング計算機シミュレータとして, シミュレーテッドアニーリングベースの Simulated Annealing Sampler [24] (以降, SA と呼ぶ) を用いた. SA を実行した環境は, OS は CentOS Linux 7.7.1908, CPU は Intel Xeon Gold 6148 CPU プロセッサである. SA の探索開始温度を 10,000 度, 探索終了温度を 10 度, イテレーション数を 2,750,000 回, 一度に出力する解の個数を 1 個とした. また, 全てのインスタンスで式 (28) における α の値を 10,000 とした. 本実験の SA では, 一つの二次割当問題に対して 10 回実行した結果を取得する.

ハードウェアイジング計算機 (以降, HIC と呼ぶ) として, 第 2 世代デジタルアニーラ [13] を用いた. HIC では Parallel Tempering モードを使用し, イテレーション数を 1,000,000 回, 一度に出力する解の個数を 128 個とした. また, 全てのインスタンスで式 (28) における α の値を 10,000 とした. 本実験での HIC では, 一つの二次割当問題に対して 1 回実行した結果を取得する.

表 1: 三種類のイジング計算機による二次割当問題の実験結果.

instance	#variables	QAPLIB のコスト	コストの最小値			実行時間 [s]			制約充足率 [%]		
			SA	HIC	ICSS	SA	HIC	ICSS	SA	HIC	ICSS
esc16a	256	68*	96 (+41.2%)	68 (±0%)	68 (±0%)	6.94	11.6	10	100	100	100
esc16b	256	292*	302 (+3.12%)	292 (±0%)	292 (±0%)	6.88	11.1	10	100	100	100
esc16c	256	160*	218 (+36.3%)	160 (±0%)	160 (±0%)	6.84	11.3	10	100	100	100
esc16d	256	16*	36 (+125%)	16 (±0%)	16 (±0%)	6.84	11.4	10	100	100	100
esc16e	256	28*	50 (+78.6%)	28 (±0%)	28 (±0%)	6.89	11.9	10	100	100	100
esc16g	256	26*	50 (+92.3%)	26 (±0%)	26 (±0%)	6.83	11.3	10	100	100	100
esc16h	256	996*	1238 (+24.3%)	996 (±0%)	996 (±0%)	6.87	11.3	10	100	100	100
esc16i	256	14*	36 (+157%)	14 (±0%)	14 (±0%)	6.82	11.2	10	100	100	100
esc16j	256	8*	18 (+125%)	8 (±0%)	8 (±0%)	6.91	11.5	10	100	100	100
esc32a	1024	130	410 (+215%)	198 (+34.3%)	130 (±0%)	31.7	12.5	10	100	94.5	100
esc32b	1024	168	408 (+143%)	284 (+40.8%)	168 (±0%)	31.6	12.3	10	100	94.5	100
esc32c	1024	642	912 (+42.1%)	694 (+7.49%)	642 (±0%)	31.1	12.4	10	100	94.5	100
esc32d	1024	200	328 (+64%)	234 (+14.5%)	200 (±0%)	31.1	12.3	10	100	95.3	100
esc32e	1024	2*	24 (+1100%)	2 (±0%)	2 (±0%)	31.2	12.5	10	100	95.3	100
esc32f	1024	2*	24 (+1100%)	2 (±0%)	2 (±0%)	31.1	12.5	10	100	95.3	100
esc32g	1024	6*	26 (+333%)	6 (±0%)	6 (±0%)	31.1	12.2	10	100	93.8	100
esc32h	1024	438	650 (+48.4%)	500 (+12.4%)	438 (±0%)	31.2	12.5	10	100	95.3	100
esc64a	4096	116	232 (+100%)	140 (+17.1%)	116 (±0%)	180	78.9	10	100	68.0	100
esc128	16384	64	308 (+381%)	-	64 (±0%)	1483	-	10	100	-	100

イジング計算機ソフトウェアシステム (以降, ICSS と呼ぶ) として, 第 3 世代デジタルアニーラ [14, 15] を利用した. ICSS では, 実行時間を 10 秒, 一度に出力する解の個数を 10 個, `gs_level` というパラメータの値を 0 とした. `gs_level` とはコスト関数の探索に関するパラメータで, 0 から 100 までの整数値をとる. 値が小さいほど制約を満たす範囲内でコスト関数を探索する. 図 2(b) や図 2(c) に示されるように, 一つの成分が 1 で残りの成分が全て 0 であるような 1-hot 制約を含む問題の場合には, `gs_level` を 0 とすることが推奨されているため [25], 本実験では, `gs_level` の値を 0 とした. また本実験で扱う ICSS では, 制約項におけるハイパーパラメータの値を自動で探索するため, 式 (28) における α の値を 1 とした. 本実験の ICSS では, 一つの二次割当問題に対して 1 回実行した結果を取得する.

3.3.2 実験結果

表 1 に実験結果を示す. SA では各問題に対して, 乱数のシード値を変更して 10 回実行し, 結果を取得した. 10 回の実行結果から得られた 10 個の解の中で, コストが最小のものを項目「コストの最小値」に, 10 回の実行時間の平均値を項目「実行時間 [s]」に, 10 個の解のうち二次割当制約を満たしていた解の割合を項目「制約充足率 [%]」に示す. HIC では各問題に対して 1 回実行し, 結果を取得した. 1 回の実行で得られた 128 個の解の中で, コストが最小のものを項目「コストの最小値」に, 1 回の実行時間を項目「実行時間 [s]」に, 128 個の解のうち二次割当制約を満たしていた解の割合を項目「制約充足率 [%]」に示す. ICSS では, 各問題に対して 1 回実行し, 結果を取得した.

1 回の実行で得られた 10 個の解の中で, コストが最小のものを項目「コストの最小値」に, 1 回の実行時間を項目「実行時間 [s]」に, 10 個の解のうち二次割当制約を満たしていた解の割合を項目「制約充足率 [%]」に示す. 表中の括弧の中の数値は QAPLIB の各インスタンスの最良コストと比較した時の相対誤差を, ハイフンはイジング計算機が扱えるスピンの上限を超えたために計算できなかったことを意味する.

コストの評価

SA のコストは, QAPLIB の各インスタンスの最良コスト ([23] によって与えられる) と比較して平均で 222% 大きい解が得られた. SA では, スピン数にかかわらず式 (11) のハイパーパラメータ α の値を 10,000 に設定したため, スピン数に応じた α を適切に設定することで, 各インスタンスの最良コストとの相対誤差を縮めることができると考えられる. HIC のコストは, 各インスタンスの最良コストと比較して平均で 7.03% 大きい解が得られた. HIC では SA と同様に, スピン数にかかわらず α の値を 10,000 に設定したため, α を適切に設定することで各インスタンスの最良コストとの相対誤差を更に縮めることができると考えられる. ICSS のコストは, 全てのインスタンスで最良コストと同じコストを取得することができた.

実行時間の評価

SA の実行時間は, スピン数が大きくなるにつれて実行時間も増大する傾向にあるが, スピン数の小さいインスタンスでは HIC や ICSS よりも高速に解を得ることができた. HIC の実行時間は, スピン数が大きくなるほど実行時

間も増大する傾向にあるが、増大の割合は SA より HIC の方が小さい結果となった。ICSS の実行時間は、全てのインスタンスであらかじめ設定した 10 秒で解を取得することができた。

制約充足率の評価

SA の制約充足率は、全てのインスタンスで 100% となった。HIC の制約充足率は、スパイン数が大きくなるにつれて制約充足率も減少する傾向にある。ICSS の制約充足率は SA の場合と同様に、全てのインスタンスで 100% となった。

4. 巡回セールスマン問題

本章では、現実に用いられる組合せ最適化問題の一つとして巡回セールスマン問題を取り上げ、巡回セールスマン問題の定式化および QUBO モデルマッピングについて説明する。また、ハードウェアイジング計算機、イジング計算機ソフトウェアシステム、ソフトウェア実装されたシミュレーテッドアニーリングベースのイジング計算機シミュレータの三つのイジング計算機で巡回セールスマン問題を求解した結果を示す。

4.1 巡回セールスマン問題の定式化

巡回セールスマン問題では、複数の都市が与えられ、任意の二都市間には距離が定義されているとき、全ての都市を一度だけ訪問して出発地点に戻ってくるような巡回路の中で、総移動距離が最小となるような都市の訪問順序を求める問題である。巡回セールスマン問題は、タンパク質の構造推定 [26] や商品配送問題 [27] に応用ができる。

N 個の都市の集合を $C' = \{c_1, c_2, \dots, c_N\}$ と定義する。任意の二都市間には距離が与えられており、都市 c_i, c_j ($1 \leq i, j \leq N$) 間の距離を $d(c_i, c_j)$ とする。また、 i 番目の都市 c_i 、 j 番目の都市 c_j に関するバイナリ変数を $v_{i,j}$ とする。 $v_{i,j}$ は都市 c_i の次に都市 c_j を訪問する時に 1、そうでないときに 0 を取るとすると、全ての都市を一度だけ訪問して出発地点に戻ってくるような巡回路の総移動距離 C は、次の式 (12) のように表せる。

$$C = \sum_{i=1}^N \sum_{j=1}^N d(c_i, c_j) v_{i,j} \quad (12)$$

なお、一度に訪れることのできる都市はただ一つのみとし、これを同時訪問制約と呼ぶ。また、任意の都市にはただ一度しか訪問しないものとし、これを訪問回数制約と呼ぶ。同時訪問制約と訪問回数制約を合わせて巡回セールスマン制約と呼ぶ。以上のもと、巡回セールスマン問題を次のように定義する。

定義 2. N 個の都市があり、各都市間には距離が定義されている。全ての都市を一度だけ訪問して出発地点に戻ってくるような巡回路の総移動距離を C とする。巡回セール

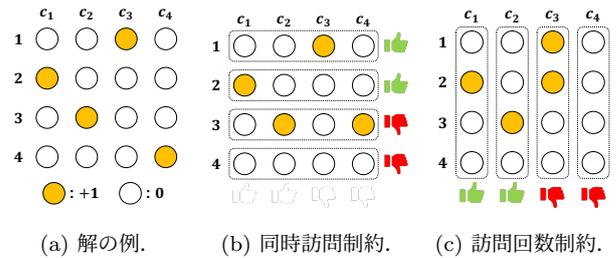


図 3: 都市数 $N = 4$ における解の例と制約条件。

マン問題とは、同時訪問制約と訪問回数制約で構成される巡回セールスマン制約を満たした上で、 C を最小化するような巡回路を求める問題である。

4.2 QUBO モデルマッピング

本節では、4.1 節で定式化した巡回セールスマン問題を QUBO モデルにマッピングする手法について説明する。まず都市数を N 、 i 番目の都市を c_i ($1 \leq i \leq N$)、時刻を t ($1 \leq t \leq N$) とし、バイナリ変数 $x_{t,i}$ を次のように定義する。

$$x_{t,i} = \begin{cases} 1 & \text{(時刻 } t \text{ に都市 } c_i \text{ を訪問する)} \\ 0 & \text{(上記以外)} \end{cases} \quad (13)$$

都市数 $N = 4$ の場合における巡回セールスマン問題の解の例を図 3 に示す。オレンジ色の丸はバイナリ変数 $x_{t,i}$ が 1、白色の丸はバイナリ変数 $x_{t,i}$ が 0 であることを表す。

4.2.1 コスト関数

コスト関数は、ある地点から出発して全ての都市を一度だけ訪問し、元の出発地点に戻ってくるような巡回路の総移動距離であり、4.1 節で定義した式 (12) によって与えられる。二つの都市 c_i, c_j 間の距離 $d(c_i, c_j)$ 、バイナリ変数 $x_{t,i}, x_{t+1,j}$ を用いて、コスト関数 \mathcal{H}_A は次の式 (14) のように表せる。

$$\mathcal{H}_A = \sum_{t=1}^N \sum_{i=1}^N \sum_{j=1}^N d(c_i, c_j) x_{t,i} x_{t+1,j} \quad (14)$$

ただし、 $x_{N+1,j} = x_{1,j}$ とする。 \mathcal{H}_A の最小値を h_a とする。 h_a は問題の入力に依存する。

4.2.2 同時訪問制約

同時訪問制約とは、時刻 t に訪問する都市はただ一つのみであるという制約である。例を図 3(b) に示す。赤色のマークは本制約に違反していること、緑色のマークは本制約を満たしていることを意味する。本制約のある時刻 t について定式化すると、次の式 (15) のように表せる。

$$\sum_{i=1}^N x_{t,i} = 1 \quad (1 \leq t \leq N) \quad (15)$$

全ての時刻 t が式 (15) を満たすときに、最小値をとるようなエネルギー関数 \mathcal{H}_B を導入すると、次の式 (16) のように

表 2: 三種類のイジング計算機による巡回セールスマン問題の実験結果.

instance	#variables	TSPLIB のコスト	コストの最小値			実行時間 [s]			制約充足率 [%]		
			SA	HIC	ICSS	SA	HIC	ICSS	SA	HIC	ICSS
berlin52	2704	7542	24949 (+231%)	16495 (+119%)	8057 (+6.83%)	103	77.3	10	100	79.7	100
eil51	2601	426	1555 (+265%)	948 (+123%)	428 (+0.469%)	99.1	74.7	10	100	78.1	100
eil76	5776	538	2273 (+322%)	1384 (+157%)	578 (+7.43%)	292	102	10	100	60.9	100
eil101	10201	629	3152 (401%)	-	686 (+9.06%)	680	-	10	100	-	100
kroA100	10000	21282	97893 (360%)	-	26170 (23.0%)	660	-	10	100	-	100
kroC100	10000	20749	100695 (+385%)	-	24576 (+18.4%)	668	-	10	100	-	100
kroD100	10000	21294	94105 (+342%)	-	24669 (+15.8%)	661	-	10	100	-	100
lin105	11025	14379	82627 (+475%)	-	17123 (+19.1%)	765	-	10	100	-	100
pr76	5776	108159	N/A	251949 (+133%)	112963 (+4.44%)	N/A	99.1	10	N/A	40.6	100
rd100	10000	7910	45157 (+471%)	-	9314 (+17.7%)	644	-	10	100	-	100
st70	4900	675	3199 (+374%)	1856 (+175%)	713 (+5.63%)	231	97.9	10	100	64.1	100

表せる.

$$\mathcal{H}_B = \sum_{t=1}^N \left(\sum_{i=1}^N x_{t,i} - 1 \right)^2 \quad (16)$$

\mathcal{H}_B の最小値は 0 をとる.

4.2.3 訪問回数制約

訪問回数制約とは, i 番目の都市 c_i はただ一度しか訪問しないという制約である. 例を図 3(c) に示す. 赤色のマークは本制約に違反していること, 緑色のマークは本制約を満たしていることを意味する. 本制約を i 番目の都市 c_i について定式化すると, 次の式 (17) のように表せる.

$$\sum_{t=1}^N x_{t,i} = 1 \quad (1 \leq i \leq N) \quad (17)$$

任意の都市 c_i が式 (17) を満たすときに, 最小値をとるようなエネルギー関数 \mathcal{H}_C を導入すると, 次の式 (18) のように表せる.

$$\mathcal{H}_C = \sum_{i=1}^N \left(\sum_{t=1}^N x_{t,i} - 1 \right)^2 \quad (18)$$

\mathcal{H}_C の最小値は 0 をとる.

4.2.4 QUBO モデルのエネルギー関数

4.2.1 節から 4.2.3 節にかけて紹介した三つのエネルギー関数 \mathcal{H}_A , \mathcal{H}_B , \mathcal{H}_C を重み付き和で表すと, 最終的なエネルギー関数 \mathcal{H} は次の式 (19) のように表せる.

$$\mathcal{H} = \mathcal{H}_A + \alpha(\mathcal{H}_B + \mathcal{H}_C) \quad (19)$$

$\alpha (> 0)$ はハイパーパラメータである. エネルギー関数 \mathcal{H} は最小値 h_a を取り, このとき基底解となる. 基底解が得られたときのスピニングが最適解となる.

4.3 イジング計算機による解法

本節では, QUBO モデルにマッピングした巡回セールスマン問題を, ソフトウェア実装されたイジング計算機シミュレータ, ハードウェアイジング計算機, イジング計算機ソフトウェアシステムの三種類で実験した結果を示し考察する.

4.3.1 実験環境

インスタンスは TSPLIB [28] を使用した. 本実験ではソフトウェア実装されたイジング計算機シミュレータ, ハードウェアイジング計算機, イジング計算機ソフトウェアシステムを用いた.

ソフトウェア実装されたイジング計算機シミュレータとして, 3.3.1 節と同様に SA を用いた. SA を実行した環境は, OS は CentOS Linux 7.7.1908, CPU は Intel Xeon Gold 6148 CPU プロセッサである. SA の探索開始温度を 10,000 度, 探索終了温度を 10 度, イテレーション数を 2,750,000 回, 一度に出力する解の個数を 1 個とした. また, 全てのインスタンスで式 (28) における α の値を 10,000 とした. 本実験の SA では, 一つの巡回セールスマン問題に対して 10 回実行した結果を取得する.

ハードウェアイジング計算機 (HIC) として, 3.3.1 節と同様に [13] を用いた. HIC では Parallel Tempering モードを使用し, イテレーション数を 1,000,000 回, 一度に出力する解の個数を 128 個とした. また, 全てのインスタンスで式 (28) における α の値を 10,000 とした. 本実験での HIC では, 一つの巡回セールスマン問題に対して 1 回実行した結果を取得する.

イジング計算機ソフトウェアシステム (ICSS) として, 3.3.1 節と同様に [14,15] を用いた. ICSS では, 実行時間を 10 秒, 一度に出力する解の個数を 10 個, gs_level の値を 0 とした. また本実験で扱う ICSS では, 制約項におけるハイパーパラメータの値を自動で探索するため, 式 (28) における α の値を 1 とした. 本実験の ICSS では, 一つの巡回セールスマン問題に対して 1 回実行した結果を取得する.

4.3.2 実験結果

表 2 に実験結果を示す. SA では各問題に対して, 乱数のシード値を変更して 10 回実行し, 結果を取得した. 10 回の実行結果から得られた 10 個の解の中で, コストが最小のものを項目「コストの最小値」に, 10 回の実行時間の平均値を項目「実行時間 [s]」に, 10 個の解のうち巡回セールスマン制約を満たしていた解の割合を項目「制約充足率

[%]」に示す。HIC では各問題に対して 1 回実行し、結果を取得した。1 回の実行で得られた 128 個の解の中で、コストが最小のものを項目「コストの最小値」に、1 回の実行時間を項目「実行時間 [s]」に、128 個の解のうち巡回セールスマン制約を満たしていた解の割合を項目「制約充足率 [%]」に示す。ICSS では、各問題に対して 1 回実行し、結果を取得した。1 回の実行で得られた 10 個の解の中で、コストが最小のものを項目「コストの最小値」に、1 回の実行時間を項目「実行時間 [s]」に、10 個の解のうち巡回セールスマン制約を満たしていた解の割合を項目「制約充足率 [%]」に示す。表中の括弧の中の数値は TSPLIB の各インスタンスの最良コストと比較した時の相対誤差を、ハイフンはイジング計算機が扱えるスピン数の上限を超えたために計算できなかったことを、N/A は実験したにもかかわらず巡回セールスマン制約を満たした解が一つも得られなかったことを意味する。

コストの評価

SA のコストは、TSPLIB の各インスタンスの最良コスト ([28] によって与えられる) と比較して平均で 362% 大きい解が得られた。SA では、スピン数にかかわらず式 (19) の α の値を 10,000 に設定したため、スピン数に応じた α を適切に設定することで、各インスタンスの最良コストとの相対誤差を縮めることができると考えられる。HIC のコストは、各インスタンスの最良コストと比較して平均で 141% 大きい解が得られた。HIC では SA と同様に、スピン数にかかわらず α の値を 10,000 に設定したため、 α を適切に設定することで各インスタンスの最良コストとの相対誤差を縮めることができると考えられる。ICSS のコストは、各インスタンスの最良コストと比較して平均で 11.6% 大きい解が得られ、SA や HIC の場合より各インスタンスの最良コストに非常に近い値を得ることができた。

実行時間の評価

SA の実行時間は、スピン数が大きくなるにつれて実行時間も増大する傾向にあった。HIC の実行時間は、スピン数が大きくなるほど実行時間も増大する傾向にあるが、増大の割合は SA より HIC の方が小さい結果となった。ICSS の実行時間は、全てのインスタンスであらかじめ設定した 10 秒で解を取得することができた。

制約充足率の評価

SA の制約充足率は、pr76 を除く全てのインスタンスで 100% となった。pr76 では、最良コストが 108,159 と非常に大きい値であるため、式 (19) の α の値を 10,000 から更に大きくすることで、制約を満たす解が得られると考えられる。HIC の制約充足率は、スピン数が大きくなるにつれて減少する傾向にある。ICSS の制約充足率は、全てのインスタンスで 100% となった。

5. スロット配置問題

本章では、現実用いられる組合せ最適化問題の一つとしてスロット配置問題を取り上げ、スロット配置問題の定式化および QUBO モデルマッピングについて説明する。また、ハードウェアイジング計算機、イジング計算機ソフトウェアシステム、ソフトウェア実装されたシミュレーテッドアニーリングベースのイジング計算機シミュレータの三つのイジング計算機でスロット配置問題を求解した結果を示す。

5.1 スロット配置問題の定式化

スロット配置問題とは、複数の格子状のスロットと、一定の配線数で接続された複数の部品がそれぞれ与えられたとき、各部品間の配線数と各部品が配置されたスロット間の距離の積の総和が最小となるような部品の配置を求める問題である。スロット配置問題は、集積回路の最適配置などに応用できる [29], [30]。

m 個の部品集合を $M = \{c_1, c_2, \dots, c_m\}$ 、 p 行 q 列のスロット集合を $S = \{s_1, s_2, \dots, s_t\}$ と定義する。ただし、 $t = p \times q$ とする。スロット s_a が a_1 行 a_2 列に、スロット s_b が b_1 行 b_2 列 ($1 \leq a_1, b_1 \leq p, 1 \leq a_2, b_2 \leq q$) にあるとすると、二つのスロット s_a, s_b 間のマンハッタン距離 $l(s_a, s_b)$ は次の式 (20) のように表せる。

$$l(s_a, s_b) = |a_1 - b_1| + |a_2 - b_2| \quad (20)$$

また、各部品間には部品と部品を接続する非負整数の配線数が与えられ、二つの部品 c_i, c_j ($1 \leq i, j \leq m$) 間の配線数を $w(c_i, c_j)$ とする。なお、 $w(c_i, c_j) = w(c_j, c_i)$ 、 $w(c_i, c_i) = 0$ である。部品 c_i が置かれたスロットを $s(c_i)$ 、部品 c_j が置かれたスロットを $s(c_j)$ 、部品 c_i, c_j 間の重み付き配線長を $w(c_i, c_j) \times l(s(c_i), s(c_j))$ と定義すると、全ての部品間の重み付き配線長の総和 L は次の式 (21) のように表せる。

$$L = \sum_{i=1}^{m-1} \sum_{j=i+1}^m w(c_i, c_j) l(s(c_i), s(c_j)) \quad (21)$$

なお、任意の部品はただ一つのスロットに必ず存在するものとし、これを部品重複禁止制約と呼ぶ。また、任意のスロットには高々一つの部品を配置するものとし、これをスロット重複禁止制約と呼ぶ。部品重複禁止制約とスロット重複禁止制約を合わせてスロット配置制約と呼ぶ。以上のもと、スロット配置問題を次のように定義する。

定義 3. m 個の部品と p 行 q 列のスロットがあり、各部品間には部品と部品を接続する非負整数の配線数が与えられている。 L を重み付き総配線長とする。スロット配置問題とは、部品重複禁止制約とスロット重複禁止制約で構成されるスロット配置制約を満たした上で、 L を最小化するような部品の配置を求める問題である。

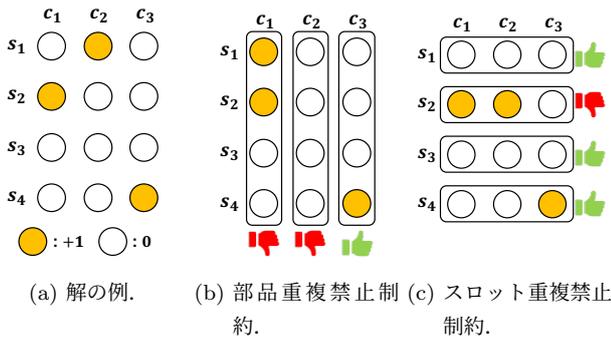


図 4: スロット数 $t = 4$, 部品数 $m = 3$ における解の例と制約条件.

5.2 QUBO モデルマッピング

本節では、スロット配置問題を QUBO モデルにマッピングする手法について説明する。部品数を m , スロット数を t , i 番目の部品を c_i ($1 \leq i \leq m$), a 番目のスロットを s_a ($1 \leq a \leq t$) とし、バイナリ変数 $x_{a,i}$ を次のように定義する。

$$x_{a,i} = \begin{cases} 1 & (\text{スロット } s_a \text{ に部品 } c_i \text{ を配置}) \\ 0 & (\text{上記以外}) \end{cases} \quad (22)$$

スロット数 $t = 4$, 部品数 $m = 3$ の場合におけるスロット配置問題の解の例を図 4(a) に示す。オレンジ色の丸はバイナリ変数 $x_{a,i}$ が 1, 白色の丸はバイナリ変数 $x_{a,i}$ が 0 であることを表す。

5.2.1 コスト関数

スロット配置問題のコスト関数は重み付き総配線長であり, 5.1 節で定義した式 (21) によって与えられる。バイナリ変数 $x_{a,i}$, $x_{b,j}$, 二つの部品 c_i , c_j 間の配線数 $w(c_i, c_j)$, 二つのスロット s_a , s_b 間のマンハッタン距離 $l(s_a, s_b)$ を用いて, コスト関数 \mathcal{H}_A は次の式 (23) のように表せる。

$$\mathcal{H}_A = \frac{1}{2} \sum_{a=1}^t \sum_{i=1}^m \sum_{b=1}^t \sum_{j=1}^m w(c_i, c_j) l(s_a, s_b) x_{a,i} x_{b,j} \quad (23)$$

\mathcal{H}_A の最小値を h_a とする。 h_a は問題の入力に依存する。

5.2.2 部品重複禁止制約

部品重複禁止制約とは, i 番目の部品 c_i はただ一つのスロットにのみ存在するという制約である。例を図 4(b) に示す。赤色のマークは本制約に違反していること, 緑色のマークは本制約を満たしていることを意味する。本制約を i 番目の部品 c_i について定式化すると, 次の式 (24) のように表せる。

$$\sum_{a=1}^t x_{a,i} = 1 \quad (1 \leq i \leq m) \quad (24)$$

全ての部品 c_i が式 (24) を満たすときに最小値をとるようなエネルギー関数 \mathcal{H}_B を導入すると, 次の式 (25) のように表せる。

$$\mathcal{H}_B = \sum_{i=1}^m \left(\sum_{a=1}^t x_{a,i} - 1 \right)^2 \quad (25)$$

\mathcal{H}_B の最小値は 0 をとる。

5.2.3 スロット重複禁止制約

スロット重複禁止制約とは, a 番目のスロット s_a には高々一つの部品が配置されているという制約である。例を図 4(c) に示す。赤色のマークは本制約に違反していること, 緑色のマークは本制約を満たしていることを意味する。本制約を a 番目のスロット s_a について定式化すると, 次の式 (26) のように表せる。

$$\sum_{i=1}^m x_{a,i} = 1 \text{ or } \sum_{i=1}^m x_{a,i} = 0 \quad (1 \leq a \leq t) \quad (26)$$

全てのスロット s_a が式 (26) を満たすときに最小値をとるようなエネルギー関数 \mathcal{H}_C を導入すると, 次の式 (27) のように表せる。

$$\mathcal{H}_C = \sum_{a=1}^t \left(\sum_{i=1}^m x_{a,i} - \frac{1}{2} \right)^2 - \frac{t}{4} \quad (27)$$

\mathcal{H}_C の最小値は 0 をとる。

5.2.4 QUBO モデルのエネルギー関数

5.2.1 節から 5.2.3 節にかけて紹介した三つのエネルギー関数 \mathcal{H}_A , \mathcal{H}_B , \mathcal{H}_C を重み付き和で表すと, 最終的なエネルギー関数 \mathcal{H} は次の式 (28) のように表せる。

$$\mathcal{H} = \mathcal{H}_A + \alpha(\mathcal{H}_B + \mathcal{H}_C) \quad (28)$$

$\alpha (> 0)$ はハイパーパラメータである。エネルギー関数 \mathcal{H} は最小値 h_a を取り, このとき基底解となる。基底解が得られたときのスピニングが最適解となる。

5.3 イジング計算機による解法

本節では, QUBO モデルにマッピングしたスロット配置問題を, ソフトウェア実装されたイジング計算機シミュレータ, ハードウェアイジング計算機, イジング計算機ソフトウェアシステムの三種類で求解した結果を示す。

5.3.1 実験環境

部品数を m とし, 任意の部品 c_i , c_j ($1 \leq i, j \leq m$) 間には $w(c_i, c_j) \in [0, 10]$ の範囲で配線数を持つように設定する。 p 行 p 列 ($4 \leq p \leq 8$) のスロットに対し, 部品数 $m = \lceil t/2 \rceil$, $\lceil 3t/4 \rceil$, t ($t = p \times p$) とする。各インスタンスで 1 つの問題をランダムに生成し, 各イジング計算機に入力した。本実験ではソフトウェア実装されたイジング計算機シミュレータ, ハードウェアイジング計算機, イジング計算機ソフトウェアシステムを用いた。

ソフトウェア実装されたイジング計算機シミュレータとして, 3.3.1 節と同様に SA を用いた。 SA を実行した環境は, OS は CentOS Linux 7.7.1908, CPU は Intel Xeon Gold 6148 CPU プロセッサである。 SA の探索開始温度

表 3: 三種類のイジング計算機によるスロット配置問題の実験結果.

t	m	#variables	重み付き総配線長の最小値			実行時間 [s]			制約充足率 [%]		
			SA	HIC	ICSS	SA	HIC	ICSS	SA	HIC	ICSS
4 × 4	8	128	300	172 (-42.6%)	174 (-41.9%)	4.08	9.79	60	100	100	100
4 × 4	12	192	667	470 (-29.6%)	446 (-33.2%)	5.79	10.7	60	100	100	100
4 × 4	16	256	1483	1287 (-13.2%)	1235 (-16.7%)	7.22	10.8	60	100	100	100
5 × 5	12	300	812	514 (-36.7%)	446 (-45.1%)	10	11.4	60	100	100	100
5 × 5	18	450	2449	1906 (-22.2%)	1719 (-29.8%)	14.7	12	60	100	100	100
5 × 5	25	625	4525	4113 (-9.11%)	3857 (-14.8%)	19.4	11.8	60	100	99.2	100
6 × 6	18	648	2879	1993 (-30.8%)	1721 (-40.2%)	23.6	12	60	100	100	100
6 × 6	27	972	6326	5358 (-15.3%)	4977 (-21.3%)	35	12.6	60	100	96.1	100
6 × 6	36	1296	10610	9980 (-5.94%)	9207 (-13.4%)	46.6	23.8	60	100	93.8	100
7 × 7	24	1176	5608	4108 (-26.7%)	3471 (-38.1%)	47.9	23.6	60	100	96.1	100
7 × 7	36	1764	12281	10551 (-14.1%)	9469 (-22.9%)	76	26.1	60	100	87.5	100
7 × 7	49	2401	24822	23804 (-4.1%)	22164 (-10.7%)	103	76	60	100	80.5	100
8 × 8	32	2048	11760	8324 (-29.2%)	7459 (-36.6%)	96.8	27.1	60	100	84.4	100
8 × 8	48	3072	26030	22854 (-12.2%)	20904 (-19.7%)	155	78.3	60	100	81.3	100
8 × 8	64	4096	48675	47247 (-2.93%)	44223 (-9.15%)	217	78	60	100	72.7	100

を 10,000 度, 探索終了温度を 10 度, イテレーション数を 2,750,000 回, 一度に出力する解の個数を 1 個とした. また, 全てのインスタンスで式 (28) における α の値を 10,000 とした. 本実験の SA では, 一つのスロット配置問題に対して 10 回実行した結果を取得する.

ハードウェアイジング計算機 (HIC) として, 3.3.1 節と同様に [13] を用いた. HIC では Parallel Tempering モードを使用し, イテレーション数を 1,000,000 回, 一度に出力する解の個数を 128 個とした. また, 全てのインスタンスで式 (28) における α の値を 10,000 とした. 本実験の HIC では, 一つのスロット配置問題に対して 1 回実行した結果を取得する.

イジング計算機ソフトウェアシステム (ICSS) として, 3.3.1 節と同様に [14,15] を用いた. ICSS では, 実行時間を 60 秒, 一度に出力する解の個数を 10 個, gs_level の値を 0 とした. また本実験で扱う ICSS では, 制約項におけるハイパーパラメータの値を自動で探索するため, 式 (28) における α の値を 1 とした. 本実験の ICSS では, 一つのスロット配置問題に対して 1 回実行した結果を取得する.

5.3.2 実験結果

表 3 に実験結果を示す. SA では各問題に対して, 乱数のシード値を変更して 10 回実行し, 結果を取得した. 10 回の実行結果から得られた 10 個の解の中で, 重み付き総配線長が最小のものを項目「重み付き総配線長の最小値」に, 10 回の実行時間の平均値を項目「実行時間 [s]」に, 10 個の解のうちスロット配置制約を満たしていた解の割合を項目「制約充足率 [%]」に示す. HIC では各問題に対して 1 回実行し, 結果を取得した. 1 回の実行で得られた 128 個の解の中で, 重み付き総配線長が最小のものを項目「重み付き総配線長の最小値」に, 1 回の実行時間を項目「実行時間 [s]」に, 128 個の解のうちスロット配置制約を満た

していた解の割合を項目「制約充足率 [%]」に示す. ICSS では, 各問題に対して 1 回実行し, 結果を取得した. 1 回の実行で得られた 10 個の解の中で, 重み付き総配線長が最小のものを項目「重み付き総配線長の最小値」に, 1 回の実行時間を項目「実行時間 [s]」に, 10 個の解のうちスロット配置制約を満たしていた解の割合を項目「制約充足率 [%]」に示す. 表中の括弧の中の数値は, SA の重み付き総配線長と比較した時の相対誤差を意味する.

重み付き総配線長の評価

HIC の重み付き総配線長は, SA と比較して重み付き総配線長が平均で 19.6% 小さい値が得られた. ICSS の重み付き総配線長は, SA と比較して重み付き総配線長が平均で 26.2% 小さい値が得られ, HIC よりも大きい減少幅となった.

実行時間の評価

SA の実行時間は, スピン数が大きくなるにつれて実行時間も増大した. HIC の実行時間は, スピン数が大きくなるほど実行時間も増大したが, 増大の割合は SA より HIC の方が小さい結果となった. ICSS の実行時間は, 全てのインスタンスであらかじめ設定した 60 秒で解を取得することができた.

制約充足率の評価

SA の制約充足率は, 全てのインスタンスで 100% となった. HIC の制約充足率は, HIC ではスピン数が大きくなるにつれて減少する傾向にある. ICSS の制約充足率は SA と同様に, 全てのインスタンスで 100% となった.

6. おわりに

本稿では, 現実に応用される組合せ最適化問題として二次割当問題, 巡回セールスマン問題, スロット配置問題を取り上げ, それらの問題をハードウェアイジング計算機,

イジング計算機ソフトウェアシステム, ソフトウェア実装されたシミュレーテッドアニーリングベースのイジング計算機シミュレータの三つの計算機で解法した結果を示した.

解の精度という観点では, 今回使用したイジング計算機ソフトウェアシステムが最も優れている. 実行時間の観点では, イジング計算機ソフトウェアシステムは設定した時間で解を取得できるという特徴がある一方で, スピン数の小さな問題に対してはソフトウェアによって実現したシミュレーテッドアニーリングの方が高速に解を取得できるという結果も得られた. 制約充足率という観点では, 今回使用したハードウェアイジング計算機よりも, イジング計算機ソフトウェアシステムおよびシミュレーテッドアニーリングの方が優れているということが分かった. 2.2節で紹介したように, 現在, 多くのイジング計算機が提案されているため, 今後, さらに多くのイジング計算機ならびに組合せ最適化問題について, 本稿と同様な評価を行う予定である.

参考文献

- [1] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial intelligence review*, vol. 13, no. 2, pp. 129–170, 1999.
- [2] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica: journal of the Econometric Society*, pp. 53–76, 1957.
- [3] P. Festa, P. M. Pardalos, M. G. Resende, and C. C. Ribeiro, "Randomized heuristics for the max-cut problem," *Optimization methods and software*, vol. 17, no. 6, pp. 1033–1058, 2002.
- [4] T. R. Jensen and B. Toft, *Graph coloring problems*. John Wiley & Sons, 2011.
- [5] E.-G. Talbi and P. Bessiere, "A parallel genetic algorithm for the graph partitioning problem," in *Proceedings of the 5th International Conference on Supercomputing*, pp. 312–320, 1991.
- [6] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, pp. 1–15, 2014.
- [7] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, "Evidence for quantum annealing with more than one hundred qubits," *Nature physics*, vol. 10, no. 3, pp. 218–224, 2014.
- [8] D. Oku, K. Terada, M. Hayashi, M. Yamaoka, S. Tanaka, and N. Togawa, "A fully-connected ising model embedding method and its evaluation for cmos annealing machines," *IEICE Transactions on Information and Systems*, vol. 102, no. 9, pp. 1696–1706, 2019.
- [9] A. Marandi, Z. Wang, K. Takata, R. L. Byer, and Y. Yamamoto, "Network of time-multiplexed optical parametric oscillators as a coherent Ising machine," *Nature Photonics*, vol. 8, no. 12, pp. 937–942, 2014.
- [10] S. Tanaka, R. Tamura, and B. K. Chakrabarti, *Quantum spin glasses, annealing and computation*. Cambridge University Press, 2017.
- [11] P. I. Bunyk, E. M. Hoskinson, M. W. Johnson, E. Tolka-
cheva, F. Altomare, A. J. Berkley, R. Harris, J. P. Hilton, T. Lanting, A. J. Przybysz *et al.*, "Architectural considerations in the design of a superconducting quantum annealing processor," *IEEE Transactions on Applied Superconductivity*, vol. 24, no. 4, pp. 1–10, 2014.
- [12] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara *et al.*, "A fully programmable 100-spin coherent ising machine with all-to-all connections," *Science*, vol. 354, no. 6312, pp. 614–617, 2016.
- [13] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 667–672. IEEE, 2020.
- [14] "Third generation digital annealer white paper," <https://www.fujitsu.com/jp/documents/about/research/techintro/3rd-g-da.pdf>.
- [15] "Digital annealer homepage," <https://www.fujitsu.com/jp/digitalannealer/>.
- [16] E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Zeitschrift fur Physik*, vol. 31, no. 1, pp. 253–258, Feb. 1925.
- [17] H. Nishimori, *Statistical physics of spin glasses and information processing: an introduction*. Clarendon Press, 2001, no. 111.
- [18] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Science advances*, vol. 5, no. 4, p. eaav2372, 2019.
- [19] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 303–309, 2015.
- [20] "Fixstars Amplify," <https://amplify.fixstars.com/en/>.
- [21] A. N. Elshafei, "Hospital layout as a quadratic assignment problem," *Journal of the Operational Research Society*, vol. 28, no. 1, pp. 167–179, 1977.
- [22] G. Laporte and H. Mercure, "Balancing hydraulic turbine runners: A quadratic assignment problem," *European Journal of Operational Research*, vol. 35, no. 3, pp. 378–381, 1988.
- [23] R. E. Burkard, S. E. Karisch, and F. Rendl, "Qaplib—a quadratic assignment problem library," *Journal of Global optimization*, vol. 10, no. 4, pp. 391–403, 1997.
- [24] "Simulated annealing sampler," <https://docs.ocean.dwavesys.com/projects/real/en/latest/reference/sampler.html>.
- [25] "Digital annealer api reference (qubo api v3)," <https://portal.aispf.global.fujitsu.com/apidoc/da/jp/api-ref/da-qubo-v3-en.html>, Accessed: 2022-01-23.
- [26] O. Johnson and J. Liu, "A traveling salesman approach for predicting protein functions," *Source code for biology and medicine*, vol. 1, no. 1, pp. 1–7, 2006.
- [27] G. Mosheiov, "The travelling salesman problem with pick-up and delivery," *European Journal of Operational Research*, vol. 79, no. 2, pp. 299–310, 1994.
- [28] G. Reinelt, "Tsp-lib—a traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.
- [29] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, "An application of combinatorial optimization to statistical physics and circuit layout design," *Operations Re-*

search, vol. 36, no. 3, pp. 493–513, 1988.

- [30] S. Held, B. Korte, D. Rautenbach, and J. Vygen, “Combinatorial optimization in vlsi design,” in *Combinatorial Optimization*. IOS Press, 2011, pp. 33–96.