卒業論文　　2011年度（平成23年度）

任意の物理的量子ビット群に対する効率的な汎用量子変数配置アルゴリズム

慶應義塾大学 環境情報学部

氏名：石崎 佳織

担当教員

慶應義塾大学 環境情報学部

村井 純

徳田 英幸

楠本 博之

中村 修

高汐 一紀

Rodney D. Van Meter III

植原 啓介

三次 仁

中澤 仁

武田 圭史

平成24 年3 月26 日

卒業論文要旨 - 2011 年度 (平成 23 年度)

任意の物理的量子ビット群に対する効率的な汎用量子変数配置アルゴリズム

量子効果や熱リーク問題などの要因により，2020 年代にはノイマン型コンピュータの速度向上は行き詰ると考えられている．打開策の一つとして，量子力学に基づいた計算理論により構成される量子コンピュータが提案されている．量子計算は超並列計算の実現を目指しているが，演算を行う量子プロセッサ上の物理的な制約により，量子計算一つあたりに必要となる量子演算が膨大な数になってしまう可能性がある．

本研究では，量子計算を量子プロセッサ上で効率的に演算するために，量子計算が必要とする最小の演算ステップ数で計算可能となるよう量子変数を配置するアルゴリズムを構築した．本アルゴリズムでは，量子計算中の変数の依存関係や時間経過に対応したグラフを作成し，量子プロセッサ上に配置された物理量子ビットの位置関係からに対応したグラフに埋め込むことで問題を解決した．また，現在考案されている量子プロセッサには様々なモデルが存在するため，あらゆるモデルの量子プロセッサに対応可能な汎用アルゴリズムを作成した．本研究は量子コンパイラの必須要素であり，量子アルゴリズムなど実用的かつ大規模なプログラムを量子コンピュータ上で実現させるための道筋となる．

キーワード

1. 量子コンピュータ, 2. 量子プロセッサ, 3. グラフ理論, 4. マッピングアルゴリズム

慶應義塾大学 環境情報学部

石崎 佳織

An algorithm for optimizing movement of quantum variables on arbitrary physical qubit structures

Improvement in digital computers will stagnate soon, because of thermodynamic problems and the difficulty of managing the quantum effects. As one possible solution, quantum computers based on quantum mechanics have been proposed. Quantum arithmetic possibly requires a large number of quantum calculation because of the physical restriction on the quantum processor. This thesis presents an algorithm for mapping the quantum variables onto the quantum processor to execute quantum arithmetic on the quantum processor efficiently. This algorithm uses two graphs - the variable dependency graph and the physical layout graph. This work is an indispensable function of a quantum compiler, and serves as a route for realizing practical large-scale programs on a quantum computer.

Keywords :
1. Quantum Computer, 2. Quantum Processor, 3.Graph Theorem, 4. Mapping Algorithm

Keio University, faculty of Environmental Information

Kaori Ishizaki

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　序論

量子コンピュータは1980年代に，デビット・ドイチュらにより提案された．量子コンピュータでは，量子力学に基づき0と1の任意の割合・位相を持つ重ね合わせ状態で記述される量子ビットを用いて並列計算を行う．しかし，量子コンピュータ実現のための様々な技術的課題は，当時の微細加工技術の水準では解決不能とされており，量子コンピュータの研究は停滞する．しかしながら，現在でも計算機の主流であるノイマン型コンピュータの性能の向上は，あと十数年で滞ってしまうと考えられている．トランジスタの小型化によって，高速化が進んでいる一方で，計算時に排出される熱の増大，量子効果による計算エラーなどの問題が無視できないレベルになりつつあり，18から24ヶ月ごとに集積回路のトランジスタ数が2倍になるというムーアの法則 [1] も停滞していくと考えられている．微細加工技術の進歩は量子コンピュータ実現に関わる技術的課題を克服する可能性を示し始めた．また，Shor's アルゴリズム [2] の発見は量子コンピュータがRSA暗号を瞬間的に解く可能性を示した．これにより，再び量子コンピュータに関わる研究が盛んになっている．その一環として，量子コンピュータ上で量子計算を担う量子プロセッサの研究も進んでいるが，配置の最適化などの問題もあり，試行錯誤が続けられている．本研究における量子プロセッサとは，量子コンピュータ上で量子計算を物理的に実行する演算装置を指す．

1.2　本研究の目的

現在，有力と考えられている実装方式による量子プロセッサは，物理的に隣接した量子ビットの間でしか量子演算が行えないという制約がある．そのため，非隣接な量子ビットに対する量子演算は，量子ビット・スワッピングにより量子ビットを隣接状態にした上で実行する必要があるが，実行順序に関する考慮を伴わずにこれらの操作を実行すると，量子計算全体に必要なステップ数が膨大になり，結果として計算時間の増大を招く．また，量子ビットは量子情報を保持してからの時間経過や，物理的な操作に対して脆弱であるた

め，量子計算の総ステップ数だけでなく，量子ビット単体に対する操作回数も減らす必要がある．本研究では，これらの問題に対して，量子計算における個々の量子ビットの操作頻度や量子ビット間の依存関係に注目し，量子プロセッサ上で量子計算をより良い手順で実現するための効率的かつ汎用的な手法の構築を目的とする．この手法は将来の量子プログラミングで用いられる量子コンパイラの必須構成要素であり，大規模量子計算実現への道筋となる．

1.3 本研究の成果

本研究では，抽象的な量子回路により表記された量子プログラムを，様々な方式により物理的に実装された量子プロセッサ上において実行可能な状態に変換するための効率的かつ汎用的な手法の構築を目指し，時間経過を考慮しない静的量子変数配置アルゴリズムを設計・実装した．

1.4 本論文の構成

本論文は全 6 章から構成される．第 2 章では，本研究の背景となる量子情報科学について述べる．第 3 章では，本研究の主題となる Graph Embedding の関連事項について述べる．第 4 章では，第 3 章で述べた課題の要因を導き出し，その解決手法と設計及び開発した実装について述べる．第 5 章では，本提案手法と実装を定性的・定量的な側面から評価する．最後に第 6 章で本論文の結論を述べる．

第2章　量子情報科学

本章では，本研究の主要技術である量子情報科学の要素について順を追って説明する．量子情報科学とは量子力学を利用し情報処理や情報伝達を試みる，これまでの情報科学と異なった学問である．量子情報科学では，現在の情報科学を量子情報科学と区別するため” 古典情報科学” と呼ぶ.

2.1　重ね合わせ

重ね合わせの原理は量子情報に関するあらゆる考察、量子力学に関するほとんどの思考実験、さらにはパラドクスに関しても、最も中心的な役割を持つ。この重ね合わせの原理は “量子が複数の状態を確率的に併せ持ち，どの状態であるのか観測するまで決定されていない” という抽象的な文言によって説明される．この原理は、“二重スリット実験

ニ重スリットの手前に微弱な粒子源を設置し，ニ重スリットに向かい粒子を放出すると，粒子の強度に関わらず，一時に存在する粒子の平均数が 1 個以下という微弱な強度であっても，二重スリットの背後にある観測面に干渉縞が描かれる．この実験における粒子の量子状態は,

$$|\Psi\rangle = \frac{1}{\sqrt{2}}(|\Psi_{左}\rangle + |\Psi_{右}\rangle)$$

というコヒーレントな重ね合わせの状態で表記される．ただし，$|\Psi_{左}\rangle$ および $|\Psi_{右}\rangle$ はそれぞれ左スリット，右スリットが開いているときの状態を示す.

この実験の特色は，粒子一つだけがそれ自身と干渉するような微弱な強度においても観測される点である．その際，干渉性を失わずに粒子が左右どちらのスリットを通ったかどうかについては言明できない．また，粒子は局所性なものであるので左右のスリットを同時に通ったと言明もできない．量子情報科学ではこの原理を利用することで，古典情報科学には不可能な演算・処理を行う.

この原理は現在でも様々な解釈がなされている。

2.2 量子ビット

古典情報科学では情報の最小単位としてビットが用いられる．それに対して量子情報科学では量子ビットの概念を用いる．ビットは必ず 0 もしくは 1 のどちらか一意の値をとるのに対し，量子ビットでは 0 と 1 両方を “重ね合わせた” 状態をとる．

量子ビットの重ね合わせの状態を量子情報科学では，ケットベクトルを用いて次の様な線形結合で表す．

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

ここで，$\alpha, \beta \in \mathbb{C}$ は $|\alpha|^2 + |\beta|^2 = 1$ を満たす．この量子状態 $|\psi\rangle$ を観測した際 0 を得る確率が $|\alpha|^2$ であり，1 を得る確率が $|\beta|^2$ となる．

この重ね合わせ状態は，0 もしくは 1 をとる “確率” と，それぞれの確率波の “ 位相差” によって表記され，ブロッホ球と呼ばれる 3 次元単位球面にてしばしば表現される．ブロッホ球を図 2.1 に示す．

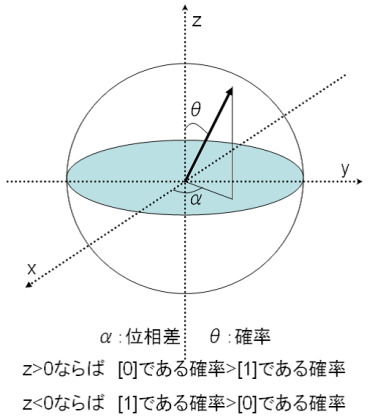


図 2.1: ブロッホ球

2.3 エンタングルメント

集団を構成する個々の対象が独立した量子状態を持たない集団を “量子もつれ状態” にある，という．“量子もつれ状態” にある集団を “Entanglement” と呼ぶ．例えば，複数の量子によって構成された Entanglement において，どれかの量子の値を測定すると残りの量子の値は測定しなくても確定的に予測できる．この性質はそれぞれの測定がどんなに離

れていようと成り立ち，量子力学が局所実在論に基づく古典論とは異なることを示している．

2.4 量子ゲート・量子回路

古典コンピュータ上の演算に AND ゲートや NOT ゲートといった古典ゲートが使用されるように，量子コンピュータ上の演算には量子ゲートが使用される．また，量子計算における原則として，量子ゲートはユニタリ行列で記述可能でなければならない．

量子ゲートの連続によって量子計算を表現する量子回路の見方を図 2.2 をによって説明する．横線はそれぞれ量子ビットを表し，その量子ビットは $|0\rangle$ と $|1\rangle$ の任意の重ね合わせ状態からなる．図の量子回路では，左から右に向かって時間経過を示す．縦線は量子ゲートを示す．量子ビットは量子ゲートを通過する度に，指定されたユニタリ変換を受ける．

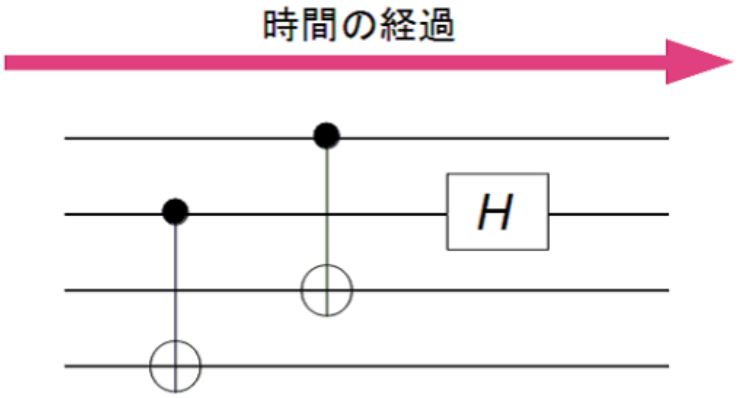


図 2.2: 量子回路の見方

以下の項で 1 量子ビット 2 量子ビット，および 3 量子ビット系に対する量子ゲートについていくつかの例を示す．

2.4.1 Pauli ゲート

1 量子ビットに対する最も基本的な操作ゲートとして Pauli ゲートがある．以下に Pauli ゲートを構成する σ_X，σ_Y および σ_Z の各ゲートが行う変換を示す．

$$\sigma_X : |1\rangle\langle 0| + |0\rangle\langle 1|, \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.2}$$

$$\sigma_Y : i|0\rangle\langle 1| - i|1\rangle\langle 0|, \quad \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} \tag{2.3}$$

$$\sigma_Z : |0\rangle\langle 0| - |1\rangle\langle 1|, \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.4}$$

ここで，Pauli ゲートは

$$\sigma_X\sigma_Y = -\sigma_Y\sigma_X = i\sigma_Z, \quad \sigma_Y\sigma_Z = -\sigma_Z\sigma_Z = i\sigma_X, \quad \sigma_Z\sigma_X = -\sigma_X\sigma_Z = i\sigma_Y \tag{2.5}$$

の各関係を満たす.

2.4.2 Hadamard ゲート

他に重要な 1 量子ビットに対する操作ゲートとして Hadamard ゲートがあり，以下の変換を行う.

$$H : \frac{1}{\sqrt{2}}\left(|0\rangle\langle 0| + |1\rangle\langle 0| + |0\rangle\langle 1| - |1\rangle\langle 1|\right) = \quad \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.6}$$

また，$|+\rangle, |-\rangle$ を Hadamard 基底とする．ここで,

$$|+\rangle = H|0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right), \quad |-\rangle = H|1\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \tag{2.7}$$

となる.

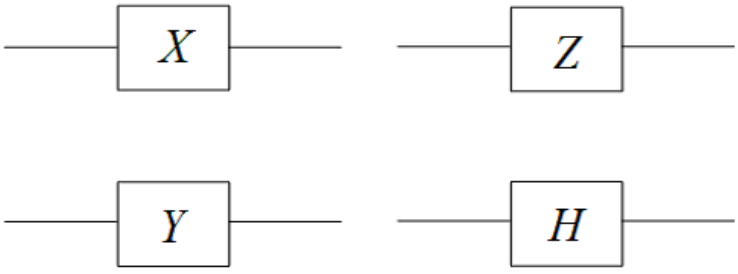


図 2.3: 左上からそれぞれ PauliX,Y,Z および Hadamard の各ゲート表記

表 2.1: CNOT 真偽値表

Input $\|AB\rangle$	Output $\|A\rangle\|B \oplus A\rangle$
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

2.4.3 制御 NOT（CNOT）ゲート

制御 NOT ゲートは 2 量子ビット系に対する操作ゲートであり，制御量子ビットおよびターゲット量子ビットとして指定された二つの量子ビット A, B に対して以下の変換を行う．

$$CNOT^{(A,B)} : |0\rangle_A|0\rangle_B\langle 0|_A\langle 0|_B + |0\rangle_A|1\rangle_B\langle 0|_A\langle 1|_B + |0\rangle_A|1\rangle_B\langle 1|_A\langle 0|_B + |1\rangle_A|0\rangle_B\langle 1|_A\langle 1|_B,$$

$$Ucn = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.8}$$

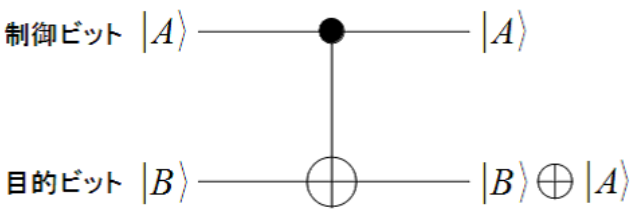


図 2.4: CNOT ゲート

2.4.4 Toffoli(CCNOT) ゲート

トフォリゲートは 3 量子ビット系に対する操作ゲートである．制御量子ビット 2 つおよびターゲット量子ビットとして指定された 3 つの量子ビット A, B, C に対して以下の変換

表 2.2: CCNOT 真偽値表

Input $\|ABC\rangle$	Output $\|A\rangle\|B\rangle\|C \oplus A \bullet B\rangle$
$\|000\rangle$	$\|000\rangle$
$\|001\rangle$	$\|001\rangle$
$\|010\rangle$	$\|010\rangle$
$\|011\rangle$	$\|011\rangle$
$\|100\rangle$	$\|100\rangle$
$\|101\rangle$	$\|101\rangle$
$\|110\rangle$	$\|111\rangle$
$\|111\rangle$	$\|110\rangle$

を行う.

$$Uccn = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.9}$$

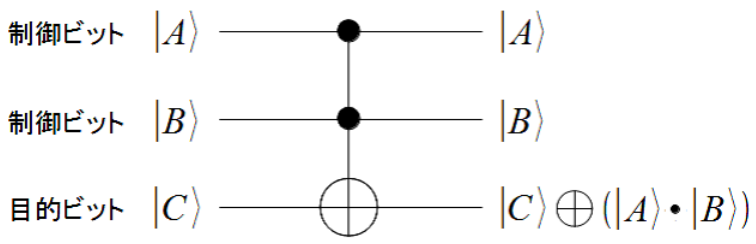


図 2.5: CCNOT ゲート

2.4.5 SWAP ゲート

二つの量子ビットの量子状態を交換するゲートを SWAP ゲートと呼び，指定された二つの量子ビット A, B に対する操作は

$$SWAP^{(A,B)} = CNOT^{(A,B)} \times CNOT^{(B,A)} \times CNOT^{(A,B)} \tag{2.10}$$

という三つの CNOT ゲートの組み合わせによって実現される．

ここで，NAND ゲートの組み合わせであらゆる古典ゲートを実現でき，同ゲートが古典コンピュータにおける万能古典ゲートであったように，量子コンピュータにおいてあらゆるゲートは制御 NOT ゲートと 1 量子ビットへの操作ゲートの組み合わせによって実現され，これらの組み合わせが万能量子ゲートとなる．

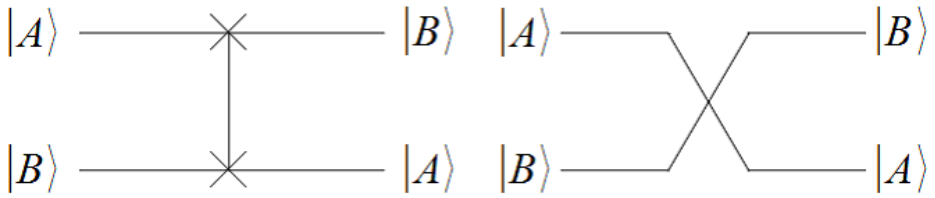


図 2.6: SWAP ゲート，左右どちらかの表記が用いられる

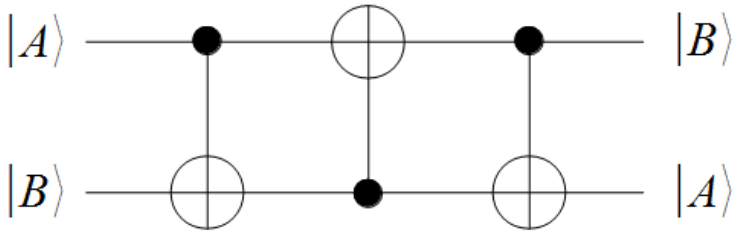


図 2.7: SWAP ゲートは CNOT ゲート 3 つで実現される

2.5 量子アルゴリズム

量子コンピュータが提唱された 1980 年代より，量子コンピュータで用いることのできる量子アルゴリズムが求められていた．

現在までに発見された著名な量子アルゴリズムとして，Peter Shor が提唱した Shor のアルゴリズムと Lov Grover の提唱した Grover のアルゴリズムが挙げられる．

Shor のアルゴリズムでは素因数分解問題を効率的に解くことができる．古典的コンピュータでの素因数分解問題は現在まで効率的に計算する方法が見つかっておらず，最も効率的とされる方法でも $O(e^{(nklog^2n)^{1/3}})$ であると知られている．

その計算の困難性により RSA 暗号を始めとする多くの暗号方式にも利用されている．一方，量子コンピュータでは離散フーリエ変換を高速に計算可能であることから素因数分解問題が効率的に計算できることが分かっている．

よって Shor のアルゴリズムを実装した量子コンピュータの実現によって RSA 暗号などの素因数分解の困難さに基づいた安全性が失われる可能性が高い．

Grover のアルゴリズム [3] は探索問題を効率的に解くことができる．例えば，線形探索での探索問題計算量は $O(n)$ であるが Grover のアルゴリズムでは $O(\sqrt{n})$ で求められる．

2.6 量子プロセッサ

量子コンピュータも古典コンピュータと同様に，アルゴリズム実行命令等を請け負うソフトウェア部分とソフトウェアからの命令を実行するハードウェア部分に分けられる．本研究では，量子コンピュータのハードウェア部分を量子プロセッサと呼ぶ．現在，量子プロセッサは様々なモデルが考えられている．

その中でも，一列に量子ビットを並べ隣接した量子ビット同士のみでのインタラクションを許す Linear Nearest Neighbor Architecture(以下，LNN アーキテクチャ)[4] での量子プロセッサモデルは実現度が高いとされている．

第3章　量子変数配置アルゴリズム

実行順序の固定されている演算群と固定されていない演算群から構成される大規模な量子計算に対する量子変数配置最適化は計算量的困難さを持ち，厳密解を求めることは難しい．そこで本研究では，近似解探索アルゴリズムを構築した上で，さまざまな物理構成を持つ量子プロセッサに適用できるよう拡張し，汎用量子変数配置アルゴリズムの構築を目指す．また，量子変数配置最適化問題に対する既存研究 [5] として，Graph Embedding という概念を用いての解決が提案されており，本研究でも同概念を用いての解決を目指す．

3.1　量子変数配置アルゴリズム

量子プロセッサには，物理的に隣接していない実量子ビット間での量子演算が不可能である，という共通した制約が存在する．これは，量子演算を行うために量子の持つ存在確率の不確定性を利用して実量子ビット同士を直接作用させていることに起因する．

また，実量子ビットは，時間経過や操作といった外部からの擾乱によって重ねあわせ状態が失われてしまい，品質が低下するという脆弱性を持っている．これにより，大規模・高精度の量子計算を実現するには，個々の量子ビットの操作頻度や量子ビット間の依存関係を解析した上で，量子計算１サイクルあたりの計算ステップ数と各実量子ビットへの操作回数が可能な限り少なくなるように量子変数を実量子ビットに割り当てる必要がある．この問題は，現在のコンパイラでの変数をストレージに割り当てる問題や，変数の移動が明確に有向に沿う点からハードウェア設計上の配線配置の問題と相似している．

3.1.1　量子プロセッサ上での量子プログラムの実行例

量子プロセッサ上で，非隣接な量子ビットに対する量子演算は，量子ビット・スワッピングにより実量子ビットを隣接状態にした上で実行する．よって，ある量子プログラムの実行を試みた時，その量子プログラムに含まれる量子演算のみでの実行はできない場合が多い．例えば，図 3.1 に見られる量子プログラムを LNN アーキテクチャ上で実現し，かつプログラムの最初と最後の量子変数配置を等しくするためには図 3.2 への様に変換を行

う必要がある．この時，目的の量子プログラム上では３個の量子演算のみを必要としていないのに対して実際には８個のスワッピングを含めた１１個の量子演算が必要になっていることが分かる．

このような，全てのゲート操作が指定された量子プロセッサ上での操作が可能な位置にある状態の量子プログラムを，本研究では実行可能な量子プログラムと定義する．

前述の通り量子演算の数の増加は，量子ビットの品質や計算時間などの面に悪影響を与える．よって，量子プログラムに対してより少ない量子演算数で実行するため何らかの方法を構築する必要がある．

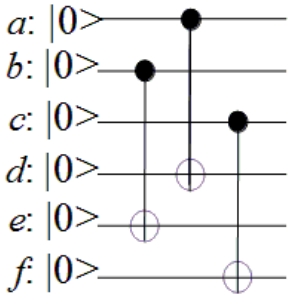


図 3.1: 量子プログラム例. ６個の量子ビットと３個の量子演算より成る．

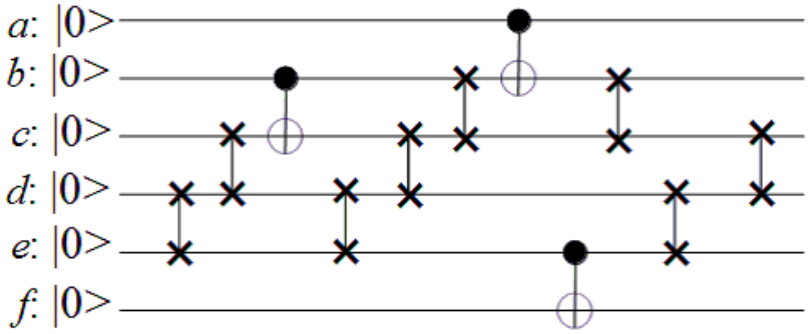


図 3.2: 図 3.1 上の量子プログラムの LNN アーキテクチャでの実現.

将来のさまざまな大規模量子計算の実現のため，現在のノイマン型コンピュータの性能を引き出すための有力な手段としてコンパイラの改良があることと同様に，さまざまな大規模量子計算を実現するためにはイオントラップや光共振器などの量子プロセッサを構成するデバイス技術の開発だけでなく，ソフトウェア開発も並行して行う必要がある．ここで，高性能な量子コンパイラを開発するためには本研究の仮想量子ビットを実量子ビットへと配置するアルゴリズムの構築が必要条件となる．

本研究では，以下において量子計算プログラム上の量子ビット（以下，量子変数）を，量子プロセッサ上に実装された量子ビット（以下，実量子ビット）に効率的に割り当てる手法を量子変数配置アルゴリズムと定義する．以下に本研究の問題解決のためのアプローチを挙げる．

3.2 Graph Embedding

Graph Embedding(グラフ埋め込み) とは，グラフ理論における概念の一つである．グラフを表現する図形において，エッジ同士が共有するノード以外では交わらないとき，その図形をグラフの実現という．実現されたグラフの例を以下の図 3.3 に示す．実現されたグラフは埋め込み可能グラフと呼ばれる．

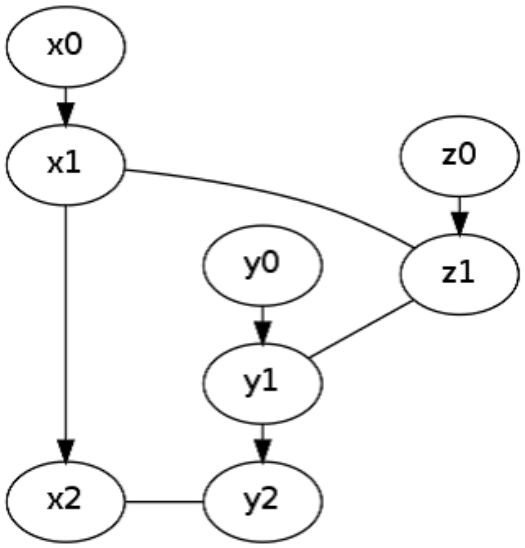


図 3.3: 実現されたグラフの例．全てのエッジ同士が交わらない．

Graph Embedding とは，図 3.4 に示すように，任意の実現されたグラフである guest graph が持つノードやエッジを，相関関係を維持したまま host graph へと埋め込むグラフ理論での概念を指す．

Graph Embedding を用いることで，量子計算プログラムにおける変数である仮想量子ビット間の依存関係と，実量子ビット群を Graph で表現可能になる．まず，仮想量子ビット間の依存関係を Graph で表現するため，ステップ毎の仮想量子ビットをノードとして扱い，仮想量子ビット間に存在する演算・時間経過の関係をエッジとして扱う．次に，実量子ビット群を Graph で表現するため，実量子ビットをノードとして扱い，隣接してい

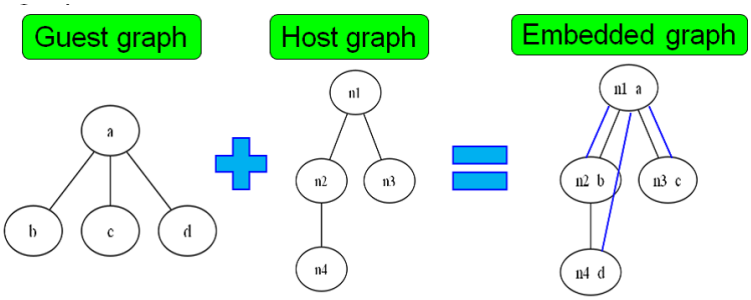


図 3.4: Graph embedding の実行．左のグラフが中心のグラフへと埋め込まれたグラフが右のグラフである．

る実量子ビット，即ち演算可能な状態にある実量子ビット間の関係をエッジとして扱う．以下の図 3.5 に例を示し，実量子ビットのグラフ化の様子を図 3.6 に示す．

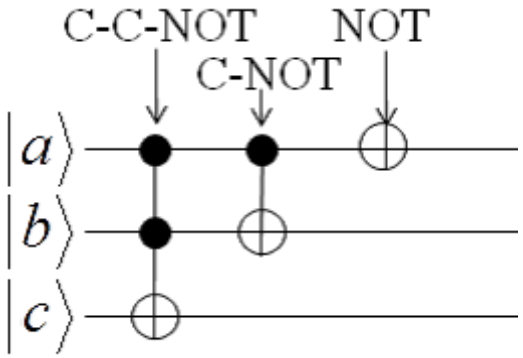
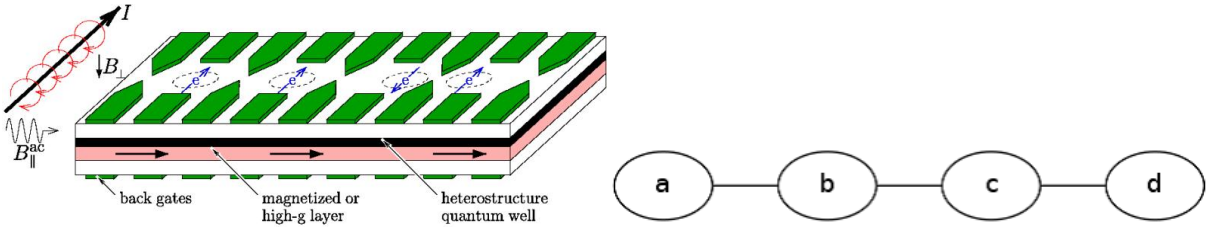


図 3.5: 仮想量子ビットの依存関係のグラフ化

これら 2 種類の Graph を用いることで，「仮想量子ビットを実量子ビット群に配置する」という問題は「仮想量子ビットの依存関係 (以下 guest graph) を実量子ビット群 (以下 host graph) に結び付ける」という Graph 理論の問題に置き換えられる．ここで，両グ



(a) 量子プロセッサの図．[6] より引用　　(b) Host graph

図 3.6: 実量子ビットのグラフ化．

ラフはそれぞれ量子計算プログラムと量子プロセッサの構成に依拠して与えられる．

3.3 研究目標

これまでの研究で，時間経過を考慮しない場合の量子変数配置アルゴリズムを作成した．同アルゴリズムに対する評価手法の確立と量子プログラムの時間経過に対応するため量子ビットスワッピングの最適化を含めたアルゴリズムの拡張を行う．また，あらゆる量子プロセッサモデルに対応したアルゴリズムの構築を行う．

量子プロセッサモデルの例として D-Wave 社の強磁性モデルの量子プロセッサ [7] や格子型 (二次元モデル) などを初めとして，近年では提案される量子プロセッサのモデルも複雑性を増してきた．しかし，既存研究においての量子変数配置アルゴリズムでは LNN アーキテクチャ上での実現を主として考えられている．

だが現状，量子プロセッサの実現は LNN アーキテクチャ上のみでの実現は確実ではなく，どのモデルで実現されるかは予測不可能である．よって本研究ではどんな量子プロセッサのモデルでも作用可能である汎用量子変数配置アルゴリズムの構築を目指す．

第4章　実装

本章では，弟 3 章で述べた問題を解決するための設計および実装要件について述べる．

4.1　設計

本研究の設計を図 4.1 に示す．量子変数配置アルゴリズムは大きく 2 段階に分かれる．まず，擬似アセンブリ言語で書かれた量子計算プログラムを解析し，Guest graph を作成する．次に，Guest graph と，量子プロセッサの物理構成から作成される Host graph を Graph 埋め込みソフトウェアへと入力し，実量子ビットに対する仮想量子ビットの初期割り当てを示した Output graph が出力する．また，本研究での Graph 作成・読み込みは，グラフィック支援ソフト Graphviz を介して行われる．

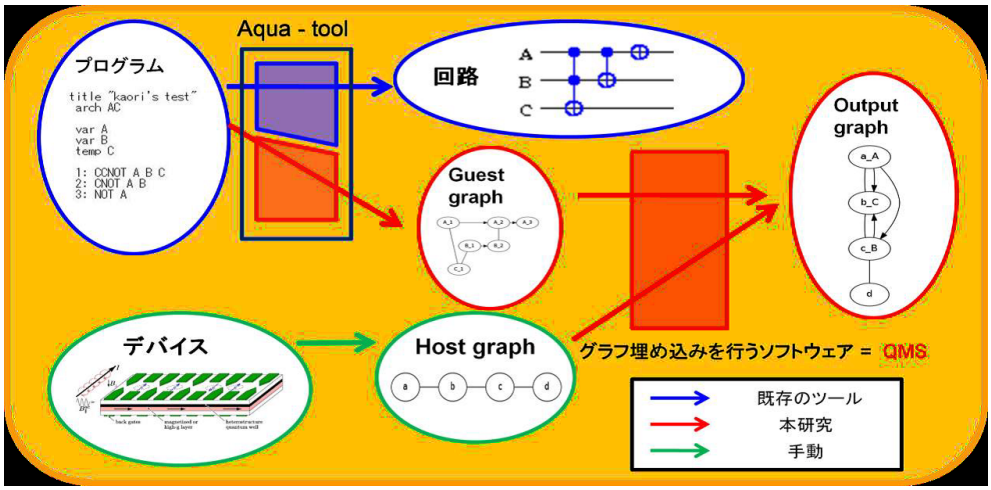
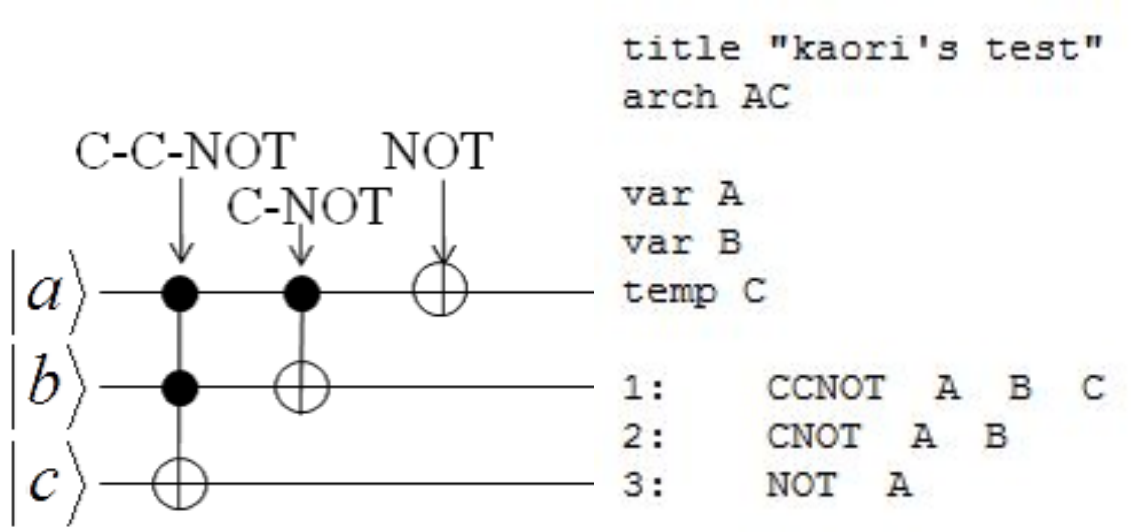


図 4.1: 本研究の設計イメージ．

4.1.1　Aqua-tools

本研究のインプットファイルとして与えられる量子プログラムは Aqua 言語と呼ばれる擬似量子アセンブラ言語で書かれることを想定する．Aqua 言語はステップ順に，演算さ

せたい量子ゲート名と量子変数名を与えることで量子プログラムを表記するアセンブラ言語に近似した単純な言語である．量子回路の Aqua 言語での表記例を以下の図 4.2 に示す．



(a) 量子回路例 (b) 左図の Aqua 言語表記

図 4.2: Aqua 言語表記例

この Aqua 言語で書かれた量子プログラムを読み込み，プログラムに基づいた量子回路を作成し画像として出力するソフトウェアが Aqua-tools である．本研究では量子プログラムをグラフ化するために Aqua-tools を拡張し，グラフィック支援ソフト graphviz 対応のグラフ言語での出力を可能にした．これにより，Aqua 言語で書かれた量子プログラムが Guest graph に変換される．本研究での Aqua-tools における拡張箇所である，Aqua 言語を graphviz 対応のグラフ言語へと変換する関数 embeddingPrettyPrint を図 4.3 に示す．
embeddingPrettyPrint

Aqua-tools を用いた一連の流れを例に示す．図 4.4(a) は，4 ビットで表現可能な 2 つの数字の加算を行う量子計算を Aqua 言語で記述したものである．これを Aqua-tools でグラフ化したものが図 4.4(b) である．図 4.4 において，ノードは「変数名 ー ステップ順」のラベルが付けられている．同じ変数名のステップ順の異なるノードは有向エッジによって結ばれ，エッジの方向により時間の経過を示す．無向エッジは同ステップ順においての依存関係のある変数対を示す．

h

```
void embeddingPrettyPrint(struct QuantumProgram *progp, char *fn){
  FILE *fp;
  int i, m, n, varn = 0;
  struct gate *gp;
  fprintf(fp, "digraph G{\n graph [rankdir = LR];\n\n");
  /* variables */
   .
   .
(中略)
   .
   .
 /* gates */
  for(i = 0;i < progp->gatecount;i++){
    gp = &progp->gatep[i];
    if(timefreeembedding){
      switch(gp->type){
      case NOT:
        fprintf(fp, " // NOT\n");
        fprintf(fp, " %s[gate = \"NOT\"];\n", arg2str(progp,
                    %&gp->args[0]));
        break;
      case CNOT:
        fprintf(fp, " // CNOT\n");
        fprintf(fp, " %s -> %s[dir = none, gate = \"CNOT\"];\n",
                    %arg2str(progp, &gp->args[0]), arg2str(progp,
                    %&gp->args[1]));
        break;
   .
   .
(続く)
```

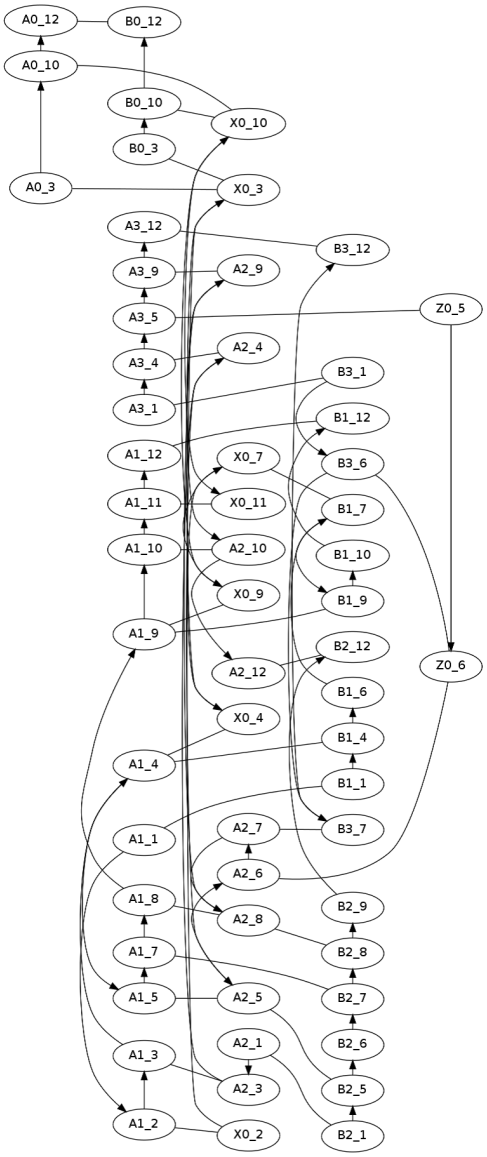
図 4.3: Aqua-tools 内拡張箇所の一部である，embeddingPrettyPrint 関数

```
title "CDKM 4-bit adder"
arch AC

var B0
var A0
temp X
var B1
var A1
var B2
var A2
var B3
var A3
var Z

1:      CNOT  A1  B1
        CNOT  A2  B2
        CNOT  A3  B3
2:      CNOT  A1  X
3:      CCNOT A0  B0  X
        CNOT  A2  A1
4:      CCNOT X  B1  A1
        CNOT  A3  A2
5:      CCNOT A1  B2  A2
        CNOT  A3  Z
6:      CCNOT A2  B3  Z
        NOT   B1
        NOT   B2
7:      CNOT  X  B1
        CNOT  A1  B2
        CNOT  A2  B3
8:      CCNOT A1  B2  A2
9:      CCNOT X  B1  A1
        CNOT  A3  A2
        NOT   B2
10:     CCNOT B0  A0  X
        CNOT  A2  A1
        NOT   B1
11:     CNOT A1  X
12:     CNOT  A0  B0
        CNOT  A1  B1
        CNOT  A2  B2
```

(a) 4ビットの加算プログラム



(b) 左図より生成された Guest graph

図 4.4: Guest graph 作成の例

4.2 Quantum Mapping Software

本研究で実装した Quantum Mapping Software(以下，QMS) は，量子プログラムを，指定された量子プロセッサ上において実行可能状態へと変換するために，Aqua-tools によって量子プログラムから作成された Guest graph と，プリセットされた量子プロセッサに対応した Host graph を選び，この 2 つのグラフを用いた Graph Embedding を行う．QMS を実行した結果として量子プログラム上の各変数の初期配置を示したテキストファイル，指定した量子プロセッサに対する aqua 言語によって表記された実行可能状態な量子プログラムおよび実行可能状態の量子プログラムの相関関係を示したグラフファイルを出力する．

4.2.1 実行可能状態な量子プログラムへの変換

QMS では，量子プログラムを実行可能状態な量子プログラムへと変換するために，量子プログラム内の全量子ゲートを隣接状態にする必要がある．このため各量子ゲートを隣接させるため SWAP ゲートを挿入する．

２量子ビットゲートの実行可能状態への変換

２量子ビットゲートの実行可能状態への変換は，２つの変数の位置の最短経路を元に隣接するまで SWAP ゲートを挿入することによって実行される．

３量子ビットゲートの実行可能状態への変換

３量子ビットゲートの実行可能状態は，中心に位置する Target bit および，２つの Control bit(Target bit に距離が近いものを Control bit1(以下，C1)，遠いものを Control bit2(以下，C2) と呼称する．) がそれぞれ隣接した状態を指す．３量子ビットゲートの実行可能状態への変換手順を以下に示す．

Step 1. Target bit を 2 つ以上の隣接ビットをもつ量子ビットへと配置する．

Step 2. Target bit に隣接する量子ビットの中から，C1 および C2 からの距離が最短な量子ビットを選び，それぞれの目的地ビットとする．

- C1 および C2 の目的地ビットが同一の場合，C1 の目的地ビットを他の量子ビットへと変更する．

Step 3. C1 を，C1 の目的地ビットへと SWAP を行い移動させる

- Target bit の位置が移動した場合，元の位置へと戻す.

Step 4. C2 を，C2 の目的地ビットへと SWAP を行い移動させる.

4.2.2 graphviz による視覚化

Graph Embedding の詳細は，グラフィック化支援ソフト graphviz を用いて視覚化される.

QMS は graphviz に対応したグラフ言語で書かれた Guest graph, Host graph の各ファイルを読み込み，Graph Embedding を行う．Graph Embedding により埋め込まれた Host graph を Output graph と呼ぶ．Output graph は量子プロセッサ上で実行可能な状態となった量子プログラムにおける変数，スワップ・ゲート操作および物理的な量子ビットの相関関係を示す.

例として，QMS での graphviz を利用した Guest graph の読み込みの様子を以下の図 4.7 に示す.

```
$ ./QMS Guestgraph.dot Hostgraph.dot       # コマンドラインでの入力
-------------------------------------------------------------------
#include <gvc.h>

// Guest graph read fron argv[1]
Graph_read(argv[1]);
tmp = FPopen("tmp.dot", "r");
tmp2 = FPopen("tmp.dot", "r");
```

図 4.5: graphviz ライブラリでの読み込み

4.3 静的アルゴリズム

本研究の最初の取り組みとして，量子プログラム上のステップ順及び量子ビット上のステップ順を考慮しない静的アルゴリズムの構築を行った．以下にアルゴリズムの概略と要素について述べる．

4.3.1 グラフの離心性・中心・直径

本アルゴリズムは，グラフ理論におけるグラフの離心性・中心・直径などの概念を利用する．これらの概念はすべて Connected graph 上で考えられる．

グラフの離心性は，個々のノードで定義される．あるノードについて，そのノード以外の全てのノードについて，グラフ上の最短距離を求め，得られた全ての最短距離のうち，最長のものをこのノードの離心性と定義する．この様子を以下の図 4.6 に示す．

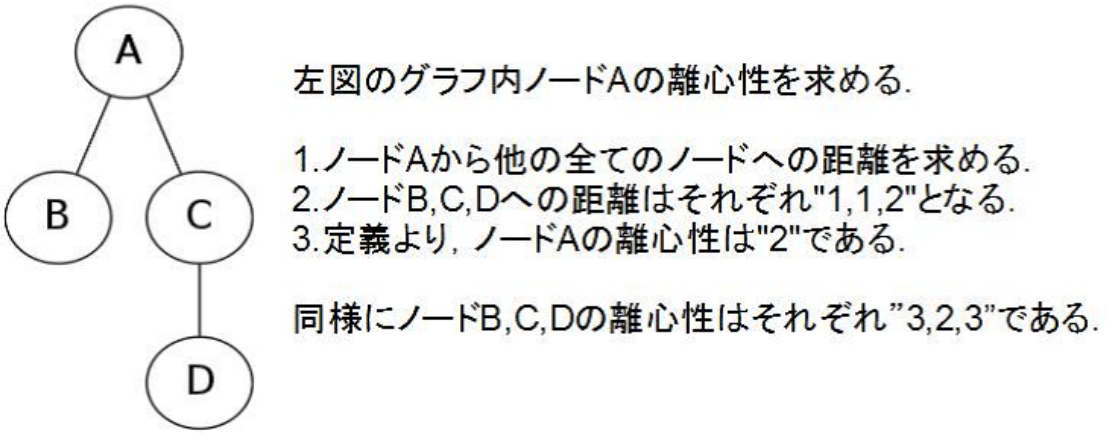


図 4.6: グラフの離心性.

グラフの直径・半径は，グラフの離心性によって与えられる．全てのノードの離心性について，その最大値を直径，最小値を半径とする．また，半径となる離心性を有する一つ以上のノードをグラフの中心と呼ぶ．

4.3.2 ノード間の距離

本研究では特に注釈の無い限り，Dijkstra's アルゴリズムの適用によって得られたグラフ上のノード間の最短距離をノード間の “距離” と定義する．Dijkstra's アルゴリズムはグ

ラフ上の 2 頂点間の最短経路問題を効率的に求めるアルゴリズムである [8]．また，本研究で使用されるグラフ上のエッジの重みは全て 1 で固定されている．

4.3.3 優先度

優先度とは，2 つのグラフの全てのノード対の親和性を示す指数である．例えば既に埋め込まれたノード対と距離の近いノード B は親和性が高いとみなされ，ノード B のノード A の近くにあるノードの優先度は高くなる．優先度の初期値は全ノード対において 0 である．

優先度の更新はノードを埋め込むたびに行われる．Guest graph ノードから選ばれた埋め込みノードを Vr とし，Host graph ノードより選ばれた埋め込み先ノードを Vt としたとき，任意の Guest graph ノード g と任意の Host graph ノード g から成るノード対 Vg,Vh に対しての優先度上下のパラメータ P(Vg,Vh) は次式によって与えられる．

$$P(Vg, Vh) = (\text{半径} - Vr \text{ と } Vg \text{ 間の距離}) + (\text{半径} - Vt \text{ と } Vh \text{ 間の距離})$$

この式によって得られた P(Vg,Vh) の値が優先度表の適切な位置に加算される．式の値が 0 以下になる場合，パラメータ値 P(Vg,Vh) は無視される．

Guest graph ノードから選ばれた埋め込みノードを Vr とし，Host graph ノードより選ばれた埋め込み先ノードを Vt とする．

優先度の更新は全ノードが埋め込まれるまで繰り返される．

4.3.4 アルゴリズム擬似コード

本アルゴリズムは，埋め込むノードの選択，埋め込み先のノードの選択，実行及び優先度更新という主に 3 つのステップにより成り立っている．その概略を以下に示す．

Host graph $G_1(V_1, E_1)$

Guest graph $G_2(V_2, E_2)$

$v_i \in V_2$ はそれぞれ優先度および離心性をパラメータに持つ

Step 1. 埋め込み元ノード (Vr) の選択
Guest graph 中で離心性が最も小さい未使用ノードを選択
IF 最小の離心性を持つノードが複数存在
THEN より優先度の合計値が小さいノードを選択

Step 2. 埋め込み先ノード (Vt) の選択
Vr に対する優先度が最も大きい Host graph 中のノードを選択
IF 最大の優先度を持つノードが複数存在
THEN 優先度の合計値が小さいノードを選択

Step 3. 埋め込みの実行
(a) Vr を Vt に埋め込む
(b) Vr-Vt 間の距離を基に, Guest graph および Host graph 中の各ノードの優先度を更新する

図 4.7: 静的アルゴリズム

第5章　評価

本章では，第三章の設計に基づいて第四章の実装を行った量子プログラムの実現可能状態への変換と静的量子変数配置アルゴリズムについて評価手法および評価結果を述べる．

5.1　実現可能な量子プログラムの評価

本節では，量子プログラムが実現可能状態な量子プログラムへと変換される際のステップ数に対して評価をとる．

5.1.1　変換する量子プログラム

実現可能状態な量子プログラムへと変換する量子プログラムとして，本節では４量子ビット同士の量子加算回路を用いる．

図 5.1 には例として 4 量子ビット同士の加算を行う加算機をあげる．

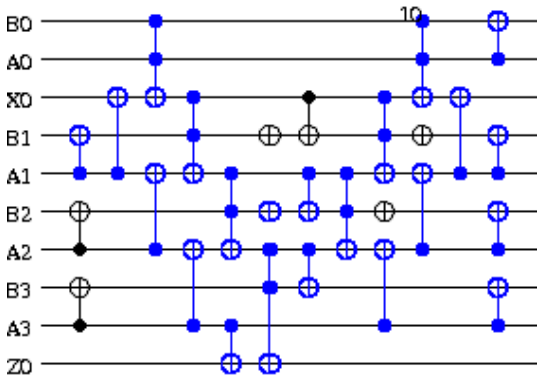


図 5.1:　４量子ビットの加算機

この量子プログラムは以下のプログラムで表記される．

また，本研究で使用する量子プログラムは簡単のために１ステップ中に１つのゲートのみを実行するものとする．上記の cdkm04 回路を１ステップ中に１つのゲートを満たす様

h

```
title "CDKM 4-bit adder"

arch AC
var B0
（中略）
var A3
var Z

1:      CNOT  A1  B1
        CNOT  A2  B2
        CNOT  A3  B3
2:      CNOT  A1  X
3:      CCNOT A0  B0  X
        CNOT  A2  A1

(中略)

10:     CCNOT B0  A0  X
        CNOT  A2  A1
        NOT   B1
11:     CNOT A1  X
12:     CNOT  A0  B0
        CNOT  A1  B1
        CNOT  A2  B2
        CNOT  A3  B3
```

図 5.2: cdkm04 プログラムの aqua 言語表記

に編集した new-cdkm04（ステップ数 28，使用変数の数 10）を用いて以下に評価を行う．このとき，cdkm04 回路と new-cdkm04 回路の実行結果は全く同じである．

new-cdkm04 回路を図 5.3 に示す．また，これをグラフ化したものを図 5.4 に示す．グラフの図中において，エッジはゲート操作を表す．

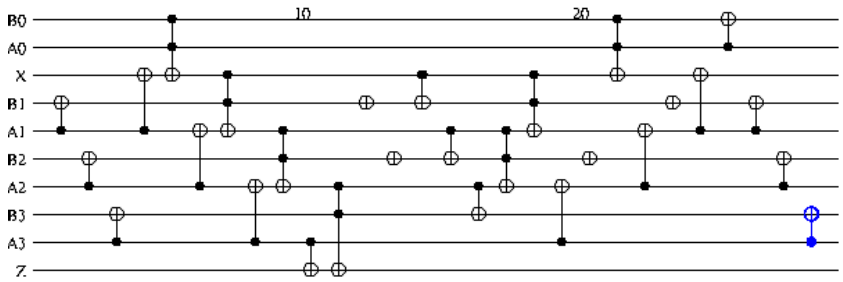


図 5.3: new-cdkm04 の回路図

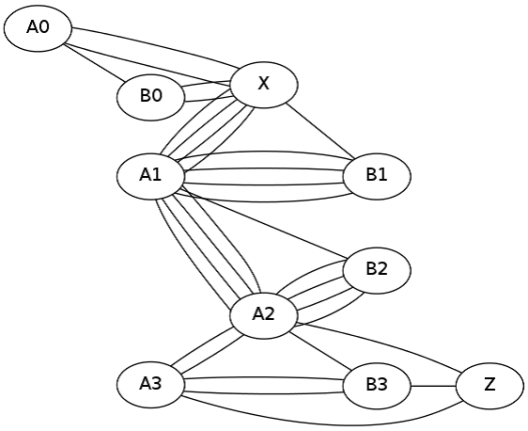


図 5.4: cdkm04 回路のグラフ化

5.1.2 1D モデルへの実現可能状態への変換

はじめに 1 次元に量子ビットが並び，隣接したもの同士のみでの演算が可能である 1D モデルについて評価を行う．この想定している 1D モデルは LNN アーキテクチャと呼ばれる．ここでは例として，1 次元に並んだ 1 2 量子ビットからなる量子プロセッサのモデルを用いる．このモデルをグラフ化したものを図 5.9 に示す.

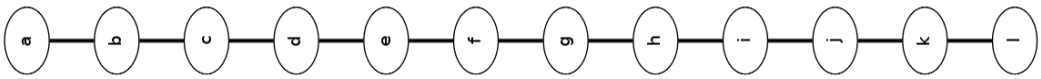


図 5.5: 1 2 量子ビットから成る 1 次元モデルのグラフ化

1D モデル実行結果

new-cdkm04 回路を 1 2 量子ビット 1D モデルへの実行可能状態へと変換した結果を示す.

Embedding されたグラフ：図 5.6 に示す.

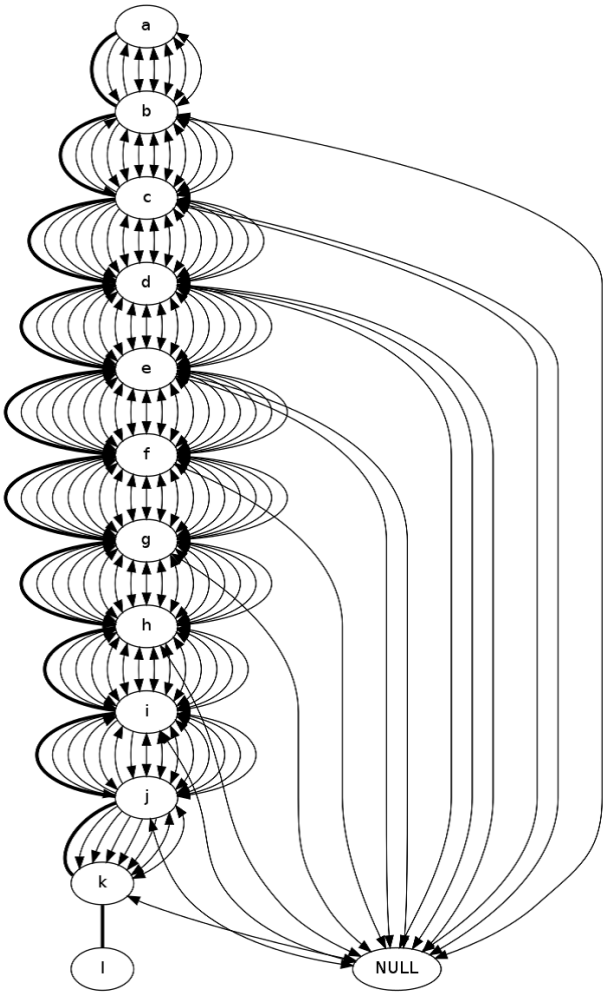


図 5.6: cdkm04 回路のグラフを 1D モデルのグラフへと埋め込んだグラフ．両端に矢のついたエッジは SWAP ゲートを，片方に矢のついたエッジはその他のゲートを，太いエッジは物理的な関係をそれぞれ示す.

初期配置：図 5.7 に示す.
実行可能状態な回路とプログラムの一部：（ステップ数：140）

```
"配置する変数名" -> "配置される量子ビット名"
() 内は，変数を配置するステップを表す.

var A1 -> a   (STEP:1)
var B1 -> b   (STEP:1)
var A2 -> c   (STEP:2)
var B2 -> d   (STEP:2)
var A3 -> e   (STEP:3)
var B3 -> f   (STEP:3)
var X  -> g   (STEP:4)
var A0 -> h   (STEP:5)
var B0 -> i   (STEP:5)
var Z  -> j   (STEP:10)
```

図 5.7: 1D 初期配置表

```
1:      CNOT  A1  B1
2:      CNOT  A2  B2
3:      CNOT  A3  B3
4:      SWAP  A1  B1
5:      SWAP  A1  A2
6:      SWAP  A1  B2
7:      SWAP  A1  A3
8:      SWAP  A1  B3
9:      CNOT  A1  X
```

図 5.8: 1D で実行可能状態にあるプログラムの一部

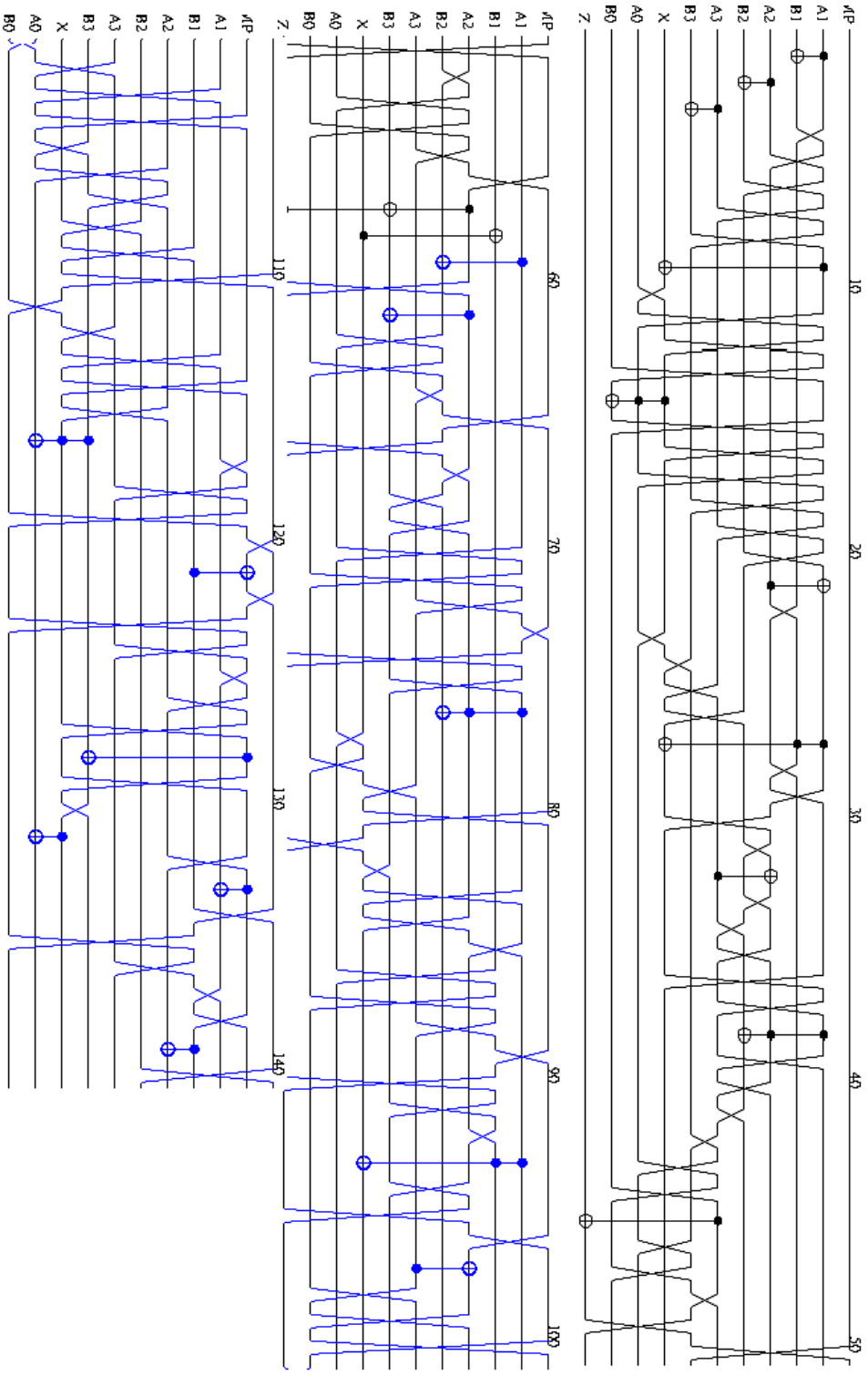


図 5.9: 1D で実行可能状態にある量子回路

5.1.3 2D モデルへの実現可能状態への変換

２次元に量子ビットが並び，隣接したもの同士のみでの演算が可能である 2D モデルについて評価を行う．ここでは例として，３×４の格子上に並んだ１２量子ビットからなる量子プロセッサのモデルを用いる．このモデルをグラフ化したものを図 5.10 に示す.

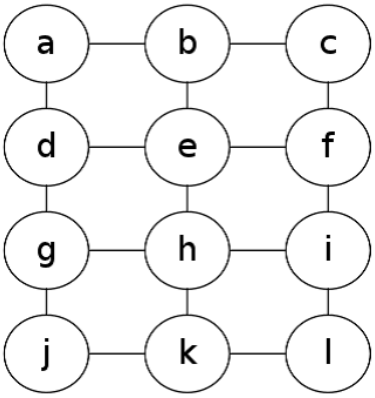


図 5.10: １２量子ビットから成る２次元モデルのグラフ化

2D モデル実行結果

new-cdkm04 回路を１２量子ビット 2D モデルへの実行可能状態へと変換した結果を示す.
Embedding されたグラフ：図 5.11 に示す.
初期配置：図 5.12 に示す.
実行可能状態な回路とプログラムの一部：(ステップ数：54)

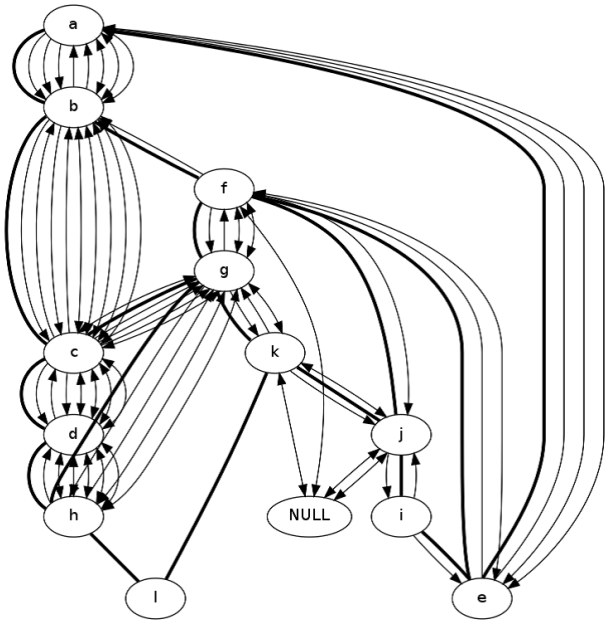


図 5.11: cdkm04 回路のグラフを 2D モデルのグラフへと埋め込んだグラフ．両端に矢のついたエッジはSWAP ゲートを，片方に矢のついたエッジはその他のゲートを，太いエッジは物理的な関係をそれぞれ示す．

```
"配置する変数名" -> "配置される量子ビット名"
() 内は，変数を配置するステップを表す．

var A1 -> a   (STEP:1)
var B1 -> b   (STEP:1)
var A2 -> c   (STEP:2)
var B2 -> d   (STEP:2)
var A3 -> e   (STEP:3)
var B3 -> f   (STEP:3)
var X  -> g   (STEP:4)
var A0 -> h   (STEP:5)
var B0 -> i   (STEP:5)
var Z  -> i   (STEP:10)
```

図 5.12: 2D 初期配置表

```
1:      CNOT  A1  B1
2:      CNOT  A2  B2
3:      CNOT  A3  B3
4:      SWAP  A1  B1
5:      SWAP  A1  A2
6:      CNOT  A1  X
```

図 5.13: 2D で実行可能状態にあるプログラムの一部

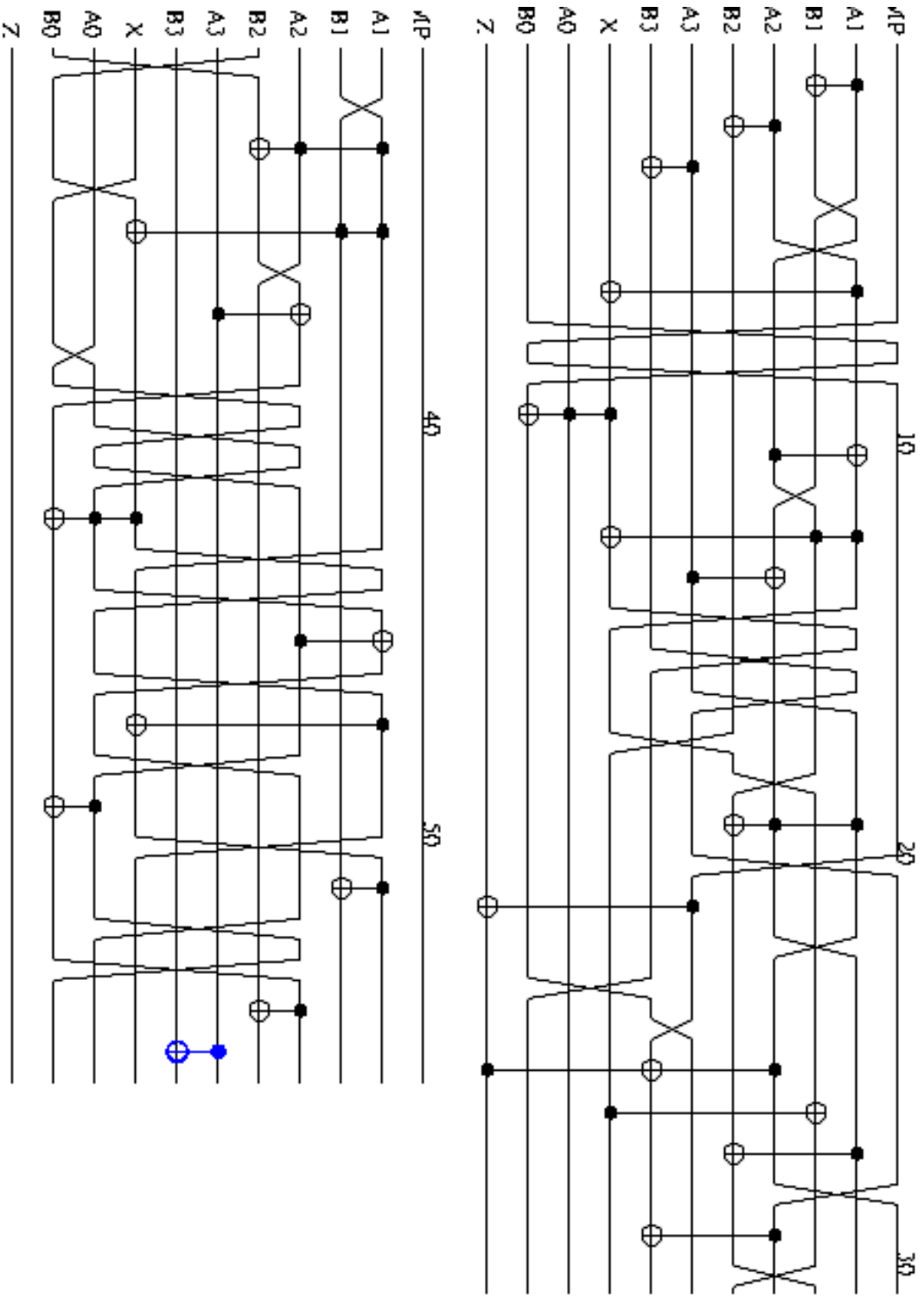


図 5.14: 2D で実行可能状態にある量子回路

5.2 静的アルゴリズムの評価

本節では，静的アルゴリズムの評価を行うために，Guest graph として量子回路を aqua-tools によってステップ順を考慮しないグラフに作成したものを用い，それを Host graph として 1-D モデル・2-D モデル・D-wave モデル 3 種類の量子プロセッサから作成したグラフへと埋め込む．評価に対して，得られた Output graph 内のエッジが 2 以上の距離に関してはそのステップ実行の都度，SWAP を繰り返しエッジの距離が 1 に，即ち隣接するまで行う必要がある．

以下にそれぞれの概要および結果について述べる．

5.2.1 埋め込む量子プログラム

Guest graph の量子回路に，本節では前節と同様に量子加算回路を用いる．また，この量子回路を時間発展を無視したグラフにしたものが図 5.15 である．

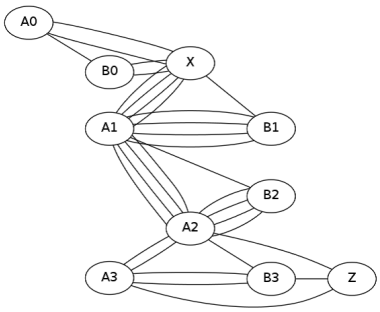


図 5.15: 静的グラフ．本節ではこのグラフを Guest graph として用いる．

5.2.2 1-D モデルでの評価

本節では，2-D 以上のモデルを作成する上で必須となる意味から最も基本的といえる 1-D モデルでの評価を行った．1-D モデルでは直列に量子ビットが並び，隣接している量子ビット同士のみの作用が可能となっている．本モデルをグラフ化したものを図 5.19 に示す．

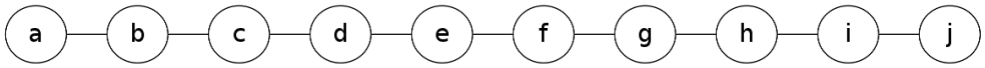


図 5.16: 1 次元プロセッサモデル．直列に並んだ１０個の量子ビットのモデルに対応している．

評価概要

Guest graph ノード数 ••• 10
Guest graph エッジ数 ••• 31
Host graph ノード数 ••• 10

評価は Guest graph のエッジの Output graph 上での総距離を用いる．これをアルゴリズムを用いず、単純に端から割り当てた場合の総距離と比較する．

評価結果

・アルゴリズム適用
エッジ総距離 ••• 48

・アルゴリズム不適用
エッジ総距離 ••• 48

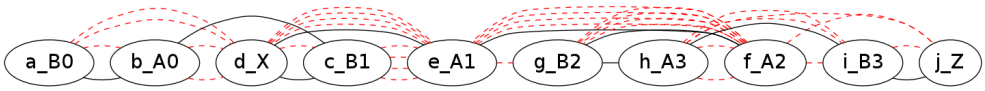


図 5.17: 1 次元プロセッサモデルに 4 量子ビット加算機を埋め込んだ図．ノード名は [埋め込み前実量子ビット名_埋め込み先変数名] となっている．実線は量子プロセッサの作用可能である量子ビットの関係性を，破線は埋め込んだ量子変数の作用があることを示す．

5.2.3 2-D モデルでの評価

1 つの量子ビットにつき最大 4 つの隣接する量子ビットに作用が可能である．本モデルをグラフ化したものを図 5.18 に示す.

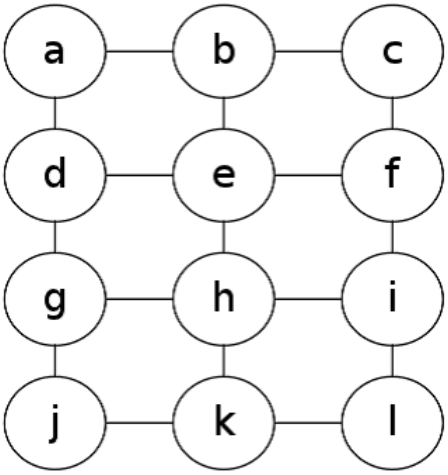


図 5.18: 2 次元プロセッサモデル．３×４のラティス状に並んだ１２個の量子ビットのモデルに対応している.

評価概要

Guest graph ノード数 ･･･ 10
Guest graph エッジ数 ･･･ 31
Host graph ノード数 ･･･ 12

評価は Guest graph のエッジの Output graph 上での総距離を用いる．これをアルゴリズムを用いず、単純に端から割り当てた場合の総距離と比較する.

評価結果

・アルゴリズム適用
エッジ総距離 ･･･ 38
・アルゴリズム不適用

エッジ総距離 ••• 50

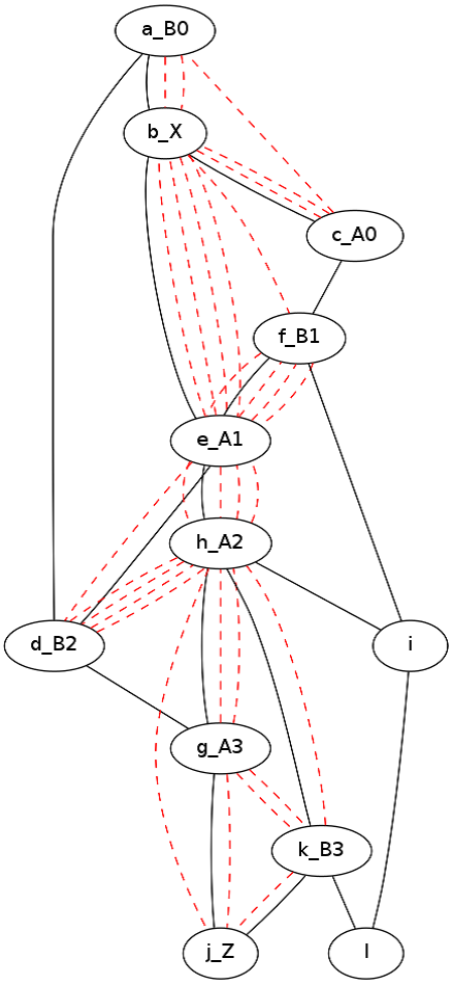


図 5.19: ２次元プロセッサモデルに４量子ビット加算機を埋め込んだ図．ノード名は [埋め込み前実量子ビット名_埋め込み先変数名] となっている．実線は量子プロセッサの作用可能である量子ビットの関係性を，破線は埋め込んだ量子変数の作用があることを示す．

5.2.4 D-wave モデルでの評価

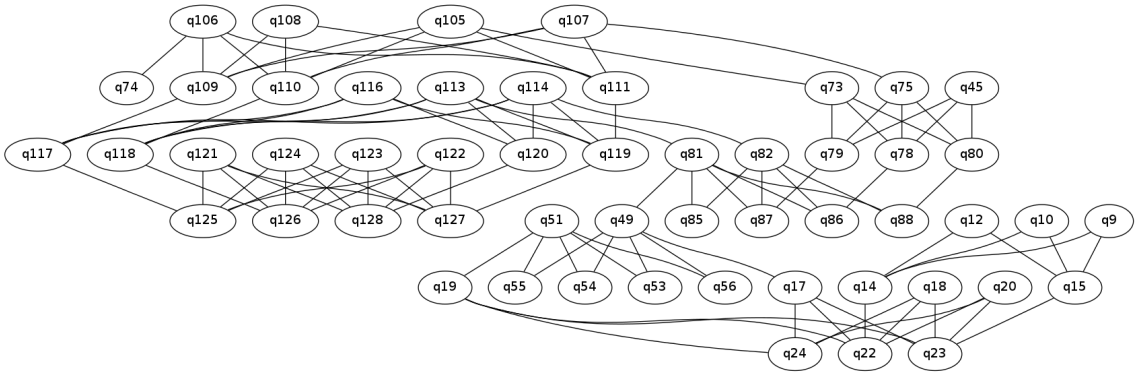


図 5.20: D-wave モデルをグラフ化した図.

D-wave 社の強磁性モデルは 128 量子ビットから構成されており，8 量子ビットずつ 16 のグループに分かれている．このような複雑な量子プロセッサのモデルを用いることにより，大きな量子プログラムを柔軟に実行可能状態へとする道筋になると考えられる.

評価概要

Guest graph ノード数 ··· 10
Guest graph エッジ数 ··· 31
Host graph ノード数 ··· 12

評価は Guest graph のエッジの Output graph 上での総距離を用いる．これをアルゴリズムを用いず、単純に端から割り当てた場合の総距離と比較する.

評価結果

・アルゴリズム適用
エッジ総距離 ··· 36
・アルゴリズム不適用]
エッジ総距離 ··· 51

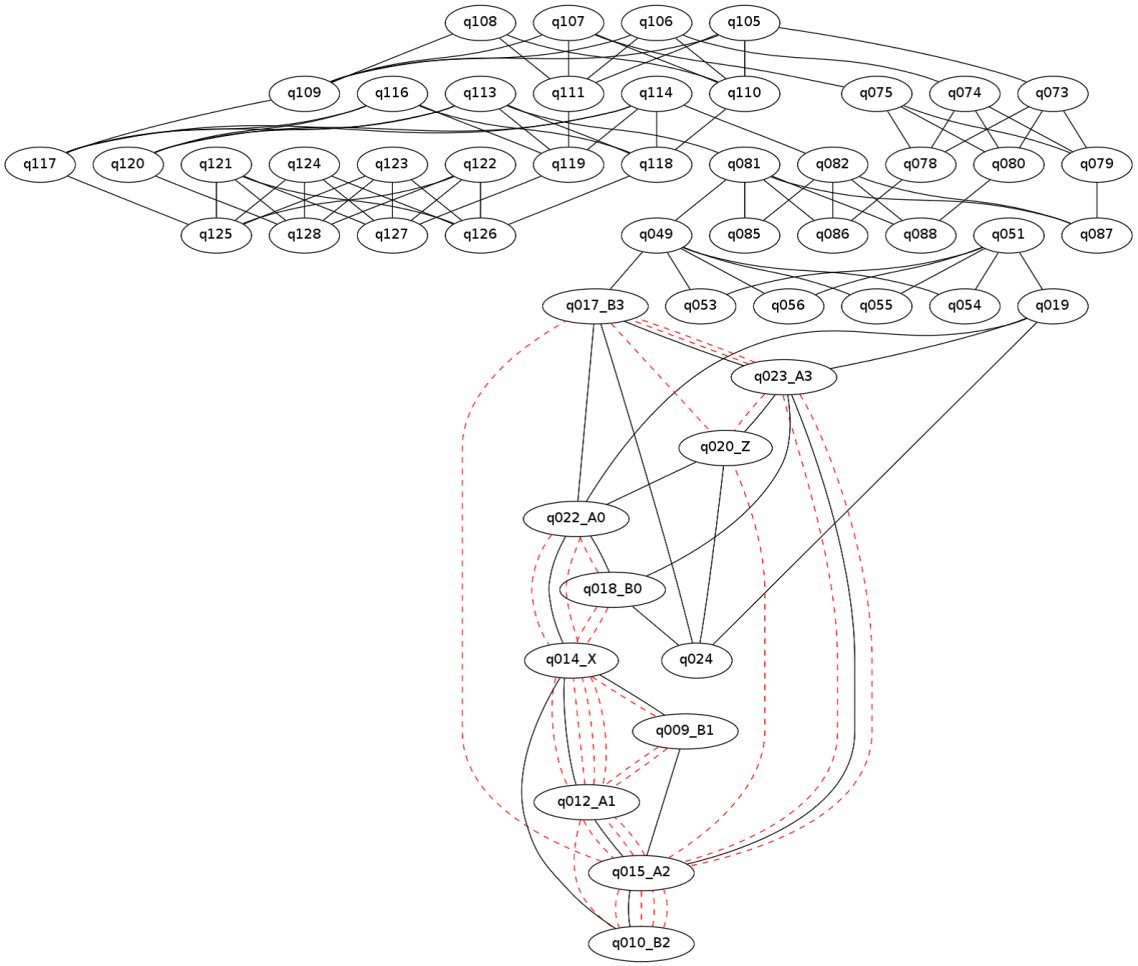


図 5.21: D-wave モデルに cdkm04 を埋め込んだ図. 実線は量子プロセッサの作用可能である量子ビットの関係性を，破線は埋め込んだ量子変数の作用があることを示す.

第6章　結論

6.1　まとめ

本研究では，量子計算を行う際に，量子プログラムが必要とする量子ビットを量子プロセッサにおけるどの物理量子ビットに当てはめるのか，という量子回路最適化問題に対して，量子プロセッサの実装方式に依存することなく，時間経過を伴わない場合での最適な配置を得ることが可能な汎用量子変数配置アルゴリズムを構築した．

量子回路最適化問題に対する既存のアプローチは，計算に用いる量子回路長自体の縮減が中心であり，また，量子プロセッサの方式を制限したり，考慮されていない場合が多かった．

これに対して本研究では，グラフ埋め込みを応用することで（擬似量子言語による）量子プログラムに対して物理量子ビットと対応した地図を作成することによりこの問題を汎用的に解決した．

これは将来の量子コンパイラの開発に向けた基礎理論の整備における重要な成果であると考えられる．

6.2　今後の展望

本研究での最終的な目的は時間経過を伴い最適な配置を行う動的汎用量子変数配置アルゴリズムの構築である．達成のためのプロセスとして，次の段階が挙げられる．

・ノード隣接のために実行される SWAP 回数の最小化問題解決
・同ステップでの実行を行うための制御を見越したアルゴリズム構築

これらの実現により動的汎用量子変数配置アルゴリズムの構築が可能になると予想される．

謝辞

本論文の作成にあたり，ご指導頂いた慶應義塾大学環境情報学部教授村井純博士，同学部教授徳田英幸博士，同学部教授中村修博士，同学部教授武田圭史博士，同学部准教授楠本博之博士，同学部准教授高汐一紀博士，同学部准教授三次仁博士，同学部准教授植原啓介博士，同学部専任講師中澤仁博士に感謝致します．

Rodney D. Van Meter III 博士に重ねて感謝致します．博士には私が研究室に入室した当初より御指導御助言を頂きました．

Clare Horsman 博士に感謝致します．博士には多くの御指導御助言を頂きました．慶應義塾大学政策メディア研究科前期博士課程永山翔太氏に感謝致します．氏には，普段の研究室での生活から，研究の進め方についてまで御指導御助言を頂きました．

東京大学佐藤貴彦氏に感謝致します．氏には，日本語の使い方や常識などを教わりました．

研究室で活動を共にしました慶應義塾大学環境情報学部 Phan Tiem Trung 氏，藤森将一氏，福山翔一郎氏，杉山智紀氏，村田紘司氏，水谷伊織氏に感謝致します．

また，執筆を共にしました研究会同期の皆様に感謝致します．

以上をもって謝辞と致します．

参考文献

[1] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), 1965.

[2] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 35:124–134, 1994.

[3] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING*, pages 212–219. ACM, 1996.

[4] Austin G. Fowler, Simon J. Devitt, and Lloyd C. Hollenberg. Implementation of Shor's algorithm on a linear nearest neighbor qubit array. *Quantum Information and Computation*, 4(4):237, 2004.

[5] Byung-Soo Choi and Rodney Van Meter. On the effect of quantum interaction distance on quantum addition circuits. *J. Emerg. Technol. Comput. Syst.*, 7:11:1–11:17, 2011.

[6] Daniel Loss and David P. DiVincenzo. Quantum computation with quantum dots. *Phys. Rev. A*, 57:120–126, 1998.

[7] Zhengbing Bian, Fabian Chudak, William G. Macready, and Geordie Rose. The ising model: teaching an old problem new tricks. 2011.

[8] E. W. Dijkstra. A note on two problems in connexion with graphs. 1(1):269–271, 1959.