社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2007－ＳＬＤＭ－128（７）
2007／1／17

2次割当問題に対するタブー探索法に基づくFPGAを用いたハードウェア解法

木村　義洋†　　若林　真一†　　永山　忍†

†広島市立大学情報科学部　〒731-3194 広島市安佐南区大塚東 3-4-1
E-mail: {wakaba,s_naga}@ce.hiroshima-cu.ac.jp

あらまし　本研究では，2次割当問題 (QAP) に対し，タブー探索法に基づくハードウェア解法を提案する．提案アルゴリズムはFPGAの大規模内部メモリを効率よく利用することで複数の近傍解を並列処理により同時に評価し，かつ，各解に対する目的関数の評価をパイプライン処理で実行することで近傍解の評価時間を大幅に短縮している．この結果，従来のソフトウェア解法と比較して非常に短い実行時間でタブー探索法に基づく近似解を得ることを可能とした．また，FPGAのプログラム可能性を利用することで，問題サイズとFPGAチップの規模を考慮した最適なハードウェア構成が実現可能になる．提案手法を Verilog-HDL ハードウェア記述言語を用いて設計し，FPGA 上に実現して性能を評価した．

キーワード　2次割当問題，タブー探索法，FPGA

A Hardware Algorithm for the Quadratic Assignment Problem Based on Tabu Search Using FPGAs

Yoshihiro KIMURA†, Shin'ichi WAKABAYASHI†, and Shinobu NAGAYAMA†

† Faculty of Information Sciences, Hiroshima City University
3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima 731-3194, Japan
E-mail: {wakaba,s_naga}@ce.hiroshima-cu.ac.jp

Abstract　In this paper, a hardware algorithm for the quadratic assignment problem (QAP) based on tabu search was proposed. The proposed algorithm effectively utilizes internal RAMs in FPGAs so that multiple neighborhood solutions are evaluated in parallel, and each neighborhood solution is evaluated in a pipeline fashion. From those features of the proposed algorithm, execution time of the tabu search for the QAP can be significantly shortened compared with its software implementation. Furthermore, utilizing the programability of FPGA devices, an optimal circuit structure of the proposed method can be easily implemented for a given instance of the problem and the size of a FPGA chip to be used. The proposed method was designed with the Verilog-HDL, and its performance was experimentally evaluated.

Key words　quadratic assignment problem, tabu search, FPGA

1. はじめに

2次割当問題 (Quadratic Assignment Problem, QAP) は NP 困難な組合せ最適化問題の1つであり，数理計画法等の通常の手法では最適解を求めることが困難であることが知られている [1]．一方，QAP は巡回セールスマン問題，VLSI レイアウト設計におけるセル配置問題など多くの問題に応用できるため，これまでに多くの研究が行われている．本研究では，タブー探索法 [1] に基づく QAP に対する解法 [5] をハードウェアで実現し，FPGA 上に実装することを提案する．

QAP に対する発見的手法の1つにタブー探索法に基づく解法がある．タブー探索法は NP 困難な組合せ問題に対する頑健な発見的手法として知られている [2], [8]．QAP のベンチマークとして知られている QAPLIB [9] において，QAPLIB の多くのベンチマーク問題の最良解はタブー探索法に基づく手法で得られている．しかしながら，QAP の問題サイズが大きくなるとタブー探索法では探索解の総数が莫大になり，実用的な時間内に優良解を得ることは困難になるという問題点がある．

一方，FPGA(Field Programmable Gate Array) とは，ユーザが手元で自由にプログラミングすることができる LSI である [4]．ここで，FPGA におけるプログラミングとは，FPGA はハードウェアでありながら，その回路構成をあたかもソフトウェアの

ごとくユーザが自由に変更できるという意味である．すなわちFPGAを用いると自分が欲しいと思う機能をもつLSIを手軽に手元で作ることができる．FPGAは回路の試作や論理エミュレータ等に利用されているほか，最近では少量生産の最終製品にもASICに替わるデバイスとして利用されることも多くなって来ている．

FPGAの応用分野として近年，注目を集めているもののひとつに，通常のソフトウェアプログラムでは解くことが非常に困難な組合せ問題をリコンフィギャラブルデバイスを用いて高速に解くことがある[3]．この解法では，与えられた組合せ問題を解く専用回路をFPGA上に構築し，ハードウェアで高速に問題を解く．著者らもグラフの最大クリーク問題および最小支配集合問題に注目し，与えられたグラフの最大クリークおよび最小支配集合を求めるインスタンスに特化したハードウェア解法を提案している[6],[7]．

本研究では，QAPに対するタブー探索法の計算時間に関する問題点を解決することを目的として，タブー探索法に基づくQAPの高速解法をハードウェアで実現し，FPGA上に実装することを提案する．FPGAの大規模内部メモリを効率よく利用して複数の近傍解を並列処理により同時に評価し，かつ，各近傍解に対する目的関数の評価をパイプライン処理で実行することで近傍解の評価時間を短縮する．このため，従来のソフトウェア解法と比較して非常に短い実行時間で解の探索が可能となる．また，FPGAのプログラム可能性を利用することで，問題サイズとFPGAチップの規模を考慮した最適なハードウェア構成が利用可能になる．

以下では，まず，2.においてQAPの定義を与え，QAPの問題例を示す．次に3.でタブー探索法について述べ，4.でQAPのタブー探索法への適用について述べる．さらにQAPに対するタブー探索法に基づくハードウェア解法を5.で提案する．6.においてQAPに対する提案ハードウェア解法をFPGAボード上に実現し，同じ動作をするソフトウェアプログラムとの比較実験を行い，計算時間に関して評価した実験結果を示す．最後に7.で本論文をまとめる．

2. 2次割当問題

2次割当問題(QAP)とは，n個の要素とn個の場所が与えられたとき，目的関数を最小にする要素の場所への割当を求める問題である．すなわち，この問題の目的は，n個の要素$\{0,1,2,\ldots,n-1\}$と2つの$n\times n$行列$A=(a_{ij})$と$B=(b_{ij})$が与えられた時，目的関数(割当コスト)を表す式(1)を最小化する要素への割り当てπと，その時の割当コストを求めることである．

$$F(\pi)=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}a_{ij}b_{\pi(i)\pi(j)} \tag{1}$$

ここでπはn個の要素のある順列を表す．また，行列Aは距離行列と呼ばれ，a_{ij}は場所iとjの間の距離を示す．行列Bはフロー行列と呼ばれ，b_{kl}は要素kとlの間のフローを表す．要素数nを問題サイズと呼ぶ．

QAPについて，LSIの1次元配置問題を例として説明する．今，4つのセル(モジュール)から構成されるネットリストが与えられているとし，隣り合うマスの距離が1で直線状に配置されたマス列に，セル間を接続するネットの総配線長ができるだけ短くなるように，4つのセルを配置する問題を考える．回路を構成する4つのセルをそれぞれ0, 1, 2, 3で表わし，セル間の接続関係(ネットリスト)を図1とする．セル間の数字はセル間を接続するネットの総数を表している．

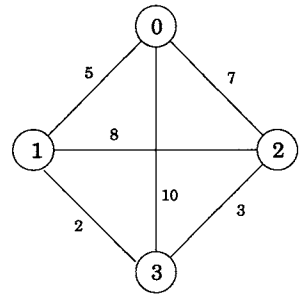

図1 セル間の接続関係

この1次元配置問題をQAPで定式化する．各セルを要素，セルが配置されるマスを場所とし，マス間の距離を行列A，セル間のネット数を行列Bで表わすとすれば，図1で表わされる1次元配置問題の行列AとBはそれぞれ式(2)，式(3)のようになる．

$$A=\begin{pmatrix}0&1&2&3\\1&0&1&2\\2&1&0&1\\3&2&1&0\end{pmatrix} \tag{2}$$

$$B=\begin{pmatrix}0&5&7&10\\5&0&8&2\\7&8&0&3\\10&2&3&0\end{pmatrix} \tag{3}$$

4つのセルのセル配置を順列πで表わすとき，セル間の総配線長は式(1)で表わされる．例えば，4つのセルを0, 1, 2, 3という順に1次元配置した時，このセル配置は$\pi=\{0,1,2,3\}$で表わされ，この場合のセル間の総配線長(コスト)は$(1\times5+2\times7+3\times10+1\times8+2\times2+1\times3)\times2=128$となる．例えば，セル0, 2間はセル間の距離が2，セル間を接続するネット数が7本あるため，コストは2×7と計算される．同様に，セル配置が$\pi=\{1,2,0,3\}$である場合の総配線長は$(1\times8+2\times5+3\times2+1\times7+2\times3+1\times10)\times2=94$となる．前記のセル配置のコストよりもコストが下がるため，この配置のほうが優れた配置である．実際，この問題例に対しては，$\pi=\{1,2,0,3\}$のセル配置が最適解である．

3. タブー探索法

タブー探索法とは，組合せ最適化問題を解くための発見的解法の一つであり，許容解 x の近傍 $N(x)$ 全体の中で，x 以外の最良の解を次の解として選び，この操作の繰り返しで最適解を求めていくものである [2]．タブー探索法では，最近探索した解をタブーリストと呼ばれるメモリに一定期間保存しておき，その解への探索を禁止するという特徴がある．単純に近傍解の中から最良解を選択し続けるとすぐに局所解に陥るため，再探索を禁止することによって局所解からの脱出を図る．また，選択された解が最適解であっても次の解への移動が強制される．このため，現在の評価値よりも悪い評価値へ移動することもある．終了条件としては，解の更新をあらかじめ決められた回数行った際に終了する，最良の評価値が一定期間更新されなくなったら終了する，等がある．

4. QAP に対するタブー探索法

タブー探索法を QAP に適用する場合，許容解は n 個の要素の順列 ϕ である．近傍解への移動は，現在の順列 ϕ のユニット r と s 一組の交換によって行う．交換前のコストがすでに計算されている場合，近傍解のコストを式 (1) で最初から計算する必要はなく，交換したユニット r と s についての差分を求めることで計算することが可能である．交換した後の解（近傍解）を π とする．この時の差分式は以下の式 (4) ように与えられる．

$$\Delta(\phi,r,s)=\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(a_{ij}b_{\phi(i)\phi(j)}-a_{ij}b_{\pi(i)\pi(j)}) \qquad (4)$$

与えられている行列が対称行列で対角成分が全て 0 であるなら，ユニット r と s を交換した時の差分式は式 (5) で与えられる．

$$\Delta(\phi,r,s)=\sum_{k\neq r,s}(a_{sk}-a_{rk})(b_{\phi(s)\phi(k)}-b_{\phi(r)\phi(k)}) \qquad (5)$$

ただし，$\pi(r)=\phi(s)$，$\pi(s)=\phi(r)$，$\pi(k)=\phi(k)$，$0\leq k<n$，$k\neq r$，$k\neq s$ である．

QAP の応用問題の多くは，定式化すると与えられる行列は対称行列で対角成分が全て 0 となるため，本研究では差分式は式 (5) で表されると仮定する．

QAP に対するタブー探索法におけるタブーリストの構成としては，近傍解として採用した解をリストに保存することが考えられる．しかし，この方法ではリストに必要なメモリが膨大になり，かつタブーリストの効率の良い探索が困難になる，という問題点がある．この問題点を解決するために，文献 [5] では，解を全て記憶するのではなく，交換したユニットの対をタブーとして記憶することを提案し，解を全て記憶する本来のタブーリストと同様のリストを実装し，効果的な探索を実現している．そこで，本研究においてもタブーリストは近傍解を生成するために交換したユニット対で構成することとした．

5. QAP 専用ハードウェア

本研究では，タブー探索法に基づく QAP の高速解法解法を FPGA を用いて実現し，ハードウェアで QAP を解くことの有効性を示す．本節では，提案ハードウェアの回路構成について述べる．

5.1 回路構成

本研究では，式 (5) の差分計算に並列処理やパイプライン演算を導入することにより，ソフトウェア解法より高速にタブー探索法を実行することを目指す．問題サイズ (n) が 16 の場合の提案ハードウェアのブロック図を図 2 に示す．

QAP の問題サイズを n とするとき，提案ハードウェアは n 個の差分計算ユニット (DCU) と 1 個のタブーメモリユニット (TMU) から構成される．以下にそれぞれのユニットの概要を示す．

5.1.1 差分計算ユニット (DCU)

回路は n 個の差分計算ユニット (DCU) を持ち，それぞれの DCU は式 (5) における各 k に対応した計算を行う．すなわち，行列 A と行列 B の値が格納されているメモリから計算に用いる値を受取り，減算器，乗算器に値を渡し，式 (5) に基づいて差分値を求める．

なお，行列 A と行列 B の値は列ごとに分散して記憶されている．行列の値を列ごとに分散して記憶することによって，行列全てを記憶するよりもメモリサイズが削減できる．また，行全ての値の操作が同時に可能となるという利点もある．

DCU のブロック図を図 3 に示す．入力を行列 A，行列 B の値と交換するペア s，r とする．何も書かれていない四角はパイプラインレジスタである．台形は各演算器を示す．

1 サイクル目で行列 A が格納されているメモリから a_{sk} を，行列 B が格納されているメモリから $b_{\phi(s)\phi(k)}$ を読み出し，レジスタに保存する．次のサイクルで，行列 A が格納されているメモリから a_{rk} を，行列 B が格納されているメモリから $b_{\phi(r)\phi(k)}$ を読み出し，別のレジスタに保存する．以後，r を $n-1$ までインクリメントしつつパイプライン演算を行う．その次のサイクルで $a_{sk}-a_{rk}$ と $b_{\phi(s)\phi(k)}-b_{\phi(r)\phi(k)}$ を計算し，さらに次のサイクルで乗算を行う．最後に，k の値が r または s と同じ場合は差分計算を行わなくてよいので，計算結果を 0 にする．

図 2 に示すように，サイズ n の問題を解くために n 個の DCU が実装され，各ユニットで $k(0\leq k<n)$ に対応した計算を並列に行う．式 (3) の計算は各 (s,r)，$0\leq r,\ s<n$ についてパイプラインで計算される．n 個の DCU の出力は，その総和を求めタブー探索において次の近傍解への移動の候補となる．差分値と，その時の (s,r) の値が解の候補としてタブーメモリユニット (TMU) に送られ，タブーリスト内の全てのペアと比較される．ペアがタブーリスト内に存在しない，もしくはペアがタブーリスト内に存在していても今まで探索してきた中で見つけられた最良解を更新するならば，タブー探索において次の近傍解となる．条件を満たさなければ近傍解の候補から外され，次の候補が調べられる．

近傍解が更新されると，各 DCU が保持している行列 B の

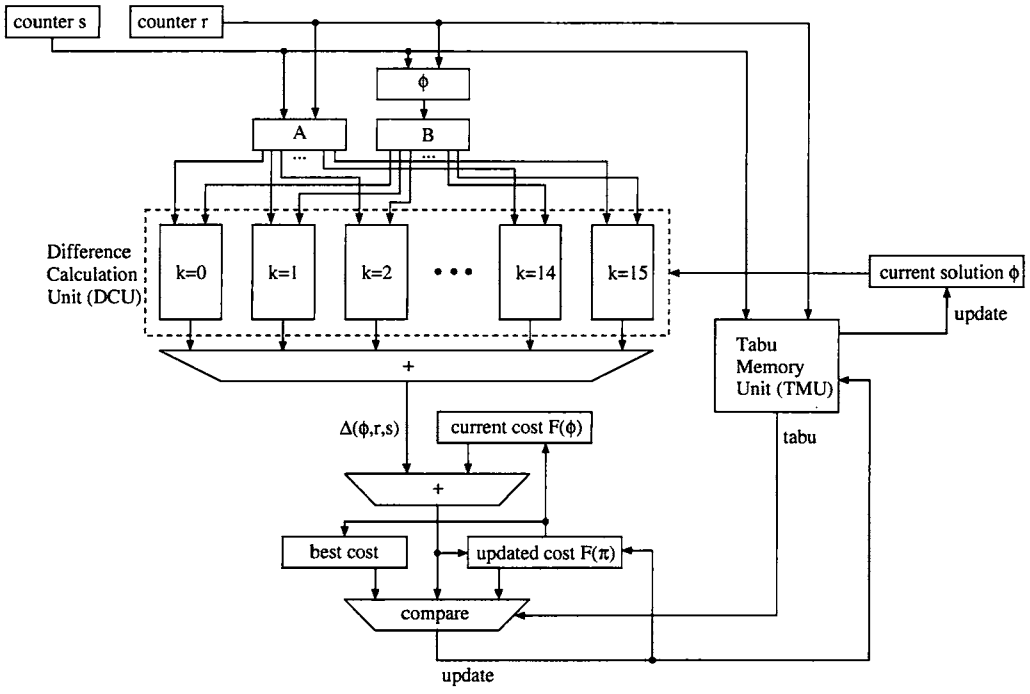

図 2　提案ハードウェア解法のブロック図 ($n = 16$)

列データを更新する．現在の近傍解から新しい近傍解にユニット r と s の交換で移動したとすれば，$k = \phi(r)$，$k = \phi(s)$ の DCU 同士がそれぞれ保持している行列 B の列データを交換する．データの交換をクロスバスイッチにて実現する方法も考えられる．メモリの値を書き換える方法に比べると，値の書き換えに必要な時間だけ実行時間が短くなるが，問題サイズが大きくなるにつれてスイッチ部分に必要なハードウェア規模が大きくなり，また，メモリの書き換えに必要な時間は近傍計算と比較すると非常に短い時間で実行できるため，本研究ではメモリの書き換えによってデータの交換を行うこととした．

5.1.2　タブーメモリユニット (TMU)

タブーメモリユニット (TMU) は許容解の更新に伴って交換されたペアを順に一定期間格納するキューと，キューに存在するペアのフラグを格納する二次元メモリの組み合わせで実現する．ブロック図を図 4 に示す．

現在の近傍解より良い解が見つかった場合，タブーメモリユニットには交換するペアの候補 (i, j) が送られる．交換しようとしているペアがタブーかどうかの決定は，(i, j) をアドレスとする二次元メモリ内のフラグの状態を見て決定する．メモリ内の値が 0 であればタブーではないので，交換するペアをキューに入れ，キューに入れられたペアに対応する二次元メモリの値を 1 にする．その後，キューから出ていくペアに対応する二次元メモリの値を 0 にする．

タブーリストをハードウェアで実現する場合，シフトレジスタのみでも実現は可能である．しかし，シフトレジスタのみで実現した場合，添字の対 (i, j) がリストに含まれているかどうかを定数時間で調べるためにはシフトレジスタを構成する全てのレジスタに比較器を付ける必要があり，ハードウェアの規模が大きくなる．一方，最近の FPGA はかなり大規模なメモリを内部に持っている．また，タブーリストの比較がキューの中身を全て比較する必要がなく，フラグの状態を見るだけで行えるため，短時間で実行できるというメリットもある．以上の理由により，上記の方式を採用した．

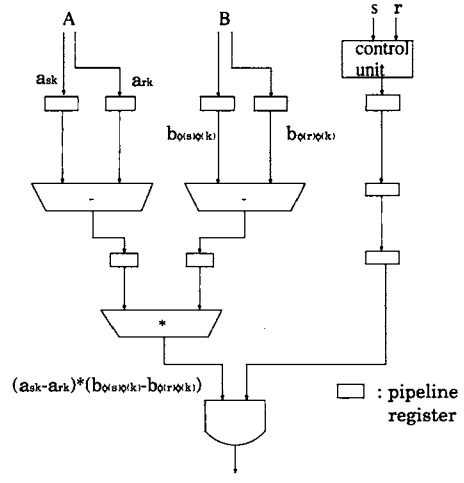

図 3　差分計算ユニット (DCU)

5.2　計算時間

問題サイズ n に対し，近傍解の個数は $n(n-1)/2$ 個存在す

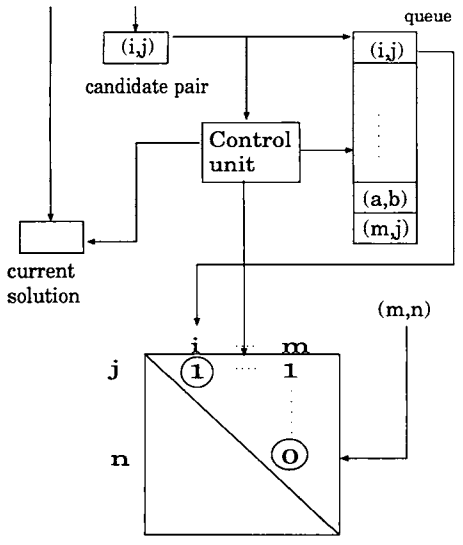

図4 タブーメモリユニット (TMU)

る．1 回の解の更新ために必要な近傍解計算をソフトウェアで実現した場合の計算時間は，各近傍解に対してそのコストの計算，すなわち式 (5) の評価に $O(n)$ の計算時間がかかるため，全体の計算時間は $O(n^3)$ となる．一方，提案ハードウェア解法では近傍解の個数は変わらないものの，各々の近傍解のコスト計算を並列処理，パイプライン処理することによって近傍解 1 個あたりの計算時間が $O(1)$ となり，全体での計算時間は $O(n^2)$ となる．よって，近傍解の更新回数 (探索の終了条件) を定数とすれば，ソフトウェアで実現した場合の全体の計算時間は問題サイズの 3 乗に比例するのに対し，提案ハードウェア解法では問題サイズの 3 乗に比例することがわかる．すなわち，問題サイズが大きくなるほど提案ハードウェア解法が有利になる．

6. 実験と評価

6.1 実験結果

提案ハードウェア解法の評価実験を行った．問題サイズを 32, 64, 128, 256 とし，サイズが 32, 64, 128 の問題は QAP のベンチマーク [9] から問題を選び，サイズが 256 の問題は新たにランダムに作成した．タブー探索法の終了条件としては，要素の交換による新たな近傍解への移動回数が 100,000 回に達するまでとした．また，タブーリストのサイズ (長さ) は予備実験を行なった結果，タブーリストのサイズを問題サイズ n に設定した場合に最も良好な結果が得られたので，そのように設定した．各問題サイズに対して，提案手法をハードウェア記述言語 Verilog-HDL を用いてハードウェア記述し，FPGA 設計ツールで論理合成を行なうことでハードウェアとして実現した．また，比較対象として，提案手法と同じ動作を行なうタブー探索法を C 言語を用いてプログラムとして実現した．ただし，提案手法では並列処理されている部分はソフトウェアではすべて逐次的に実行される．また，ハードウェア解法，ソフトウェア解法とも初期解としてはランダムに生成した同一の許容解を設定し，実行を開始した．

本研究における実験環境を以下に示す．提案ハードウェアを実現する FPGA 設計ツールには Altera 社の Quartus II Version5.0, FPGA 評価ボードには Power Medusa MU 200-SX60(ボード上の FPGA は Altera 社の EP1S60F 1020C7，論理エレメント数 57,120) を用いた．実験においては，FPGA のクロック周波数を 40MHz に設定した．比較対象のソフトウェアは，Sun Microsystems sun4u Sun Blade 1500(CPU:UltraSPARC, 1062MHz)，メインメモリ 2GB のワークステーション上で実行した．ハードウェア解法の実行時間は，回路中に実現したカウンタ回路を用いて計測し，ソフトウェア解法の実行時間は OS が提供している計時のためのシステムコマンドを用いて計測した．

提案ハードウェア解法の論理合成結果を表 1 に，ハードウェア解法とソフトウェア解法の実行時間に関する実験結果を表 2 に示す．表 1 において "LE 数" は回路の実現に要した FPGA の論理エレメント数であり，"メモリ量" は回路で使用したメモリ量である．また，LE 数，メモリ量においてカッコ内は FPGA の総 LE 数，総メモリ量に対する使用 LE 数，使用メモリ量の割合いを示す．

表 2 において，"ソフトウェア" と "ハードウェア" はそれぞれソフトウェア解法とハードウェア解法の実行時間 (単位は秒) であり，速度向上比は速度向上比=(ソフトウェア解法の実行時間/ハードウェア解法の実行時間) である．なお，元となるアルゴリズムはハードウェア解法，ソフトウェア解法とも同一であり，同一解を初期解としているため，求まった最終解も同一である．問題サイズが 32, 64, 128 の場合は，[9] で公表されている最良解のコストと同じコストの解が求まっている．

6.2 評価と考察

表 2 で示した実験結果より，ワークステーションの CPU のクロック周波数は FPGA のクロック周波数の約 26 倍であるにも関わらず，ハードウェア解法はソフトウェア解法よりも圧倒的に高速であることがわかる．これは，提案ハードウェア解法に導入した差分計算における並列処理とパイプライン処理が効果的に働いているためであると考えられる．また，ソフトウェア解法に対する提案ハードウェア解法の速度向上比は問題サイズの増加とともに増加している．これは，5.2 でも述べたように，問題サイズが大きい問題ほど提案ハードウェア解法の並列度が上がるためであり，問題サイズが 2 倍になると速度向上比は約 2 倍に改善される．

さらに，QAP の許容解は問題サイズの階乗通り存在するため，問題サイズが大きくなると探索の終了条件として要素の交換回数が 100,000 回では解空間の探索には不十分であり，優良な近似解を得るためには更なるペアの交換が必要である可能性が高くなる．例えば，問題サイズが 256 の場合，優良解を求めるためには 1 千万回の近傍解の更新が必要だとしても，提案ハードウェア解法では実用的な計算時間に収まるが，ソフトウェア解法ではその実現は極めて困難である．これらの結果，提案ハードウェア解法の有効性は示された．

表 1　論理合成結果

問題サイズ	Verilog-HDL 記述行数	合成時間 (s)	LE 数	メモリ量 (byte)
32	2300	194	2658(5 %)	9536 (0.2 %)
64	2800	343	4994(9 %)	36864 (1 %)
128	5000	773	9714(17 %)	148736 (3 %)
256	9500	2041	24540(42 %)	592896(11 %)

表 2　実行時間に関する実験結果

問題サイズ	ソフトウェア (s)	ハードウェア (s)	速度向上比
32	264	1.74	152
64	2153	6.02	358
128	17528	22.26	787
256	142320	85.46	1665

問題サイズが 256 より大きい場合，現在，市場に出回っている FPGA では 1 チップの FPGA に登載することが困難になる可能性がある．しかし，今後の半導体技術の進歩に伴い，さらに集積度の高い FPGA が市場に出て来る可能性は高く，それらの FPGA を用いればさらに大きなサイズの問題に対しても，提案ハードウェア解法を FPGA 上に実現することは可能になる．実際，本研究で実験に使用した FPGA の 3 倍以上の論理エレメント数を持つ FPGA が既に市販されている．また，提案ハードウェア解法の回路構成は図 2 に示すように基本的には 1 次元アーキテクチャであるため，提案解法の回路を分割して複数の FPGA 上に実現することも容易である．

提案ハードウェア解法をさらに高速化するためには FPGA のクロック周波数を向上させることが有効である．現在のハードウェア解法の回路構成を見直し，回路中のクリティカルパスの遅延を短縮することができればより高いクロック周波数での動作も可能になると予測している．

7. おわりに

本研究では，2 次割当問題に対するタブー探索法に基づくハードウェア解法を提案し，FPGA を用いて実現した．そして，ソフトウェア解法と比較することにより提案ハードウェア解法の有効性を示した．今後の課題としては，さらにサイズの大きな問題に対する提案ハードウェア解法の実装，それに伴って増加するメモリサイズや論理エレメント数の削減，クリティカルパスを見直すことで動作周波数を上げることによるさらなる高速化等が挙げられる．さらに，タブー探索法の頑健性を高め，解空間のより詳細な探索を実現するために，長期メモリの有用性が知られており [2]，ハードウェアでの実現が容易な長期メモリの実現方式の検討と提案手法への導入も考えられる．また，QAP が対象とする問題例の仕様が与えられた場合に，仕様で決められたデータ幅や演算精度に合わせた提案ハードウェア解法の回路記述の自動生成，および QAP 以外の問題に対するタブー探索法のハードウェア化なども今後の興味深い課題である．

謝辞　本研究の一部は平成 18 年度科学研究費補助金基盤研究 (C)(課題番号 18500042) により行った．

文　献

[1] E.Çela, "The Quadratic Assignment Problem : Theory and Algorithms," Kluwer Academic Publishers (1998).
[2] F.Glover, M.Laguna, "Tabu Search," Kluwer Academic Publishers (1997).
[3] M.Platzner, "Reconfigurable accelerators for combinatorial problems," Computer, 33, 4, pp.58–60 (2000).
[4] 末吉敏則, 天野英晴, "リコンフィギャラブルシステム", オーム社 (2005).
[5] E.Taillard, "Robust taboo search for the quadratic assignment problem," Parallel Computing, 17, pp.443–455 (1991).
[6] 若林真一, 菊池健司, "最大クリークを求めるデータ依存ハードウェアアルゴリズムの実装と評価", 情報処理学会システム LSI 設計技術研究会研究報告, 2004-SLDM-113-22 (2004).
[7] S.Wakabayashi, K.Kikuchi, "Solving the Minimum Dominating Set Problem with Instance-Specific Hardware on FPGAs," Proc. 2005 IEEE International Conference on Field Programmable Technology (FPT 2005), pp.69–76 (2005).
[8] 柳浦睦憲, 茨木俊秀, "組合せ最適化 — メタ戦略を中心として —", 浅倉書店 (2001).
[9] http://www.seas.upenn.edu/qaplib/