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

2008－SLDM－133 (10)
2008 / 1 / 16

2次割当問題に対するシストリックアルゴリズムに基づくハードウェア解法

木村 義洋† 若林 真一† 永山 忍†

† 広島市立大学 大学院 情報科学研究科
〒 731-3194 広島市安佐南区大塚東 3-4-1

あらまし 2次割当問題 (Quadratic Assignment Problem, QAP) に対し，タブー探索法に基づくヒューリスティック解法をハードウェアとして実現し，FPGA 上に実装することで問題を高速に解くことを提案する．提案するハードウェア解法はタブー探索法をシストリックアルゴリズムとして実現することにより，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することで計算時間を短縮する．提案手法を FPGA 上に実現し，ソフトウェア解法と比較することにより提案手法の有効性を示した．
キーワード 2次割当問題，タブー探索法，シストリックアルゴリズム，FPGA

Solving the Quadratic Assignment Problem by Hardware Based on a Systolic Algorithm

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

† Graduate School of Information Sciences, Hiroshima City University
3-4-1, Ozuka-higashi, Asaminami-ku, Hiroshima 731-3194, Japan

Abstract For the quadratic assignment problem (QAP), a heuristic algorithm based on tabu search, which is implemented as hardware on FPGAs, is proposed to solve the problem efficiently. The proposed hardware algorithm is a systolic algorithm, in which multiple neighborhood solutions are evaluated in parallel, and for each solution, the objective function is evaluated in a pipeline fashion so as to shorten the computation time. The proposed method was implemented on an FPGA chip, and its effectiveness was shown.
Key words quadratic assignment problem, tabu search, systolic algorithm, 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 [9]．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 [1]．本研究では，タブー探索法に基づく QAP に対するヒューリスティック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．

QAP に対する発見的手法の 1 つにタブー探索法 [2] 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．QAP のベンチマークとして知られている QAPLIB [10] 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 [7] 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 [6]．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 を用いると自分が実現したい機能をもつ 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きている．

我々は，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，文献 [3], [8] において，タブー探索法に基づくヒューリスティック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した．この手法では，

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計算時間を短縮する．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った．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利用可能になった．

本稿においては，我々が文献 [3], [8] において提案した手法をさらに改良し，より高速に QAP の近似解を求めるハードウェア解法を提案する．本論文の提案手法も我々の以前の手法と同様，タブー探索法に基づいており，タブー探索法の高速実行を実現するため，タブー探索法においてもっとも計算時間を要する近傍解の生成をシストリックアルゴリズムを用いて高速に実行する．このため，我々の従来手法よりさらに高速にタブー探索を実行することが可能になった．また，提案手法はプロセッシングユニットを 1 次元配列状に接続したアーキテクチャ上で実現されているため，大規模な問題に対しても容易に提案手法を複数の FPGA 上で実現可能になる，という特徴を持つ．

以下では，まず，2. において QAP の定義を与え，3. でタブー探索法について述べる．4. で本論文で提案する QAP に対するタブー探索法に基づくシストリックアルゴリズムを説明する．次に 5. で提案ハードウェア解法の実験的評価について述べ，最後に 6. 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 $\{1,2,\ldots,n\}$ 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．

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

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．また，要素数 n を問題サイズと呼ぶ．本研究では，行列 A と行列 B はいずれも対称行列とし，行列の対角要素はすべて 0 であると仮定する．

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 1, 2, 3, 4 で表し，セル間の接続関係を図 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．

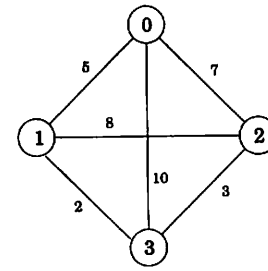

図 1 セル間の接続関係

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 で表すとすれば，図 1 で表される 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 1, 2, 3, 4 という順に 1 次元配置した時，このセル配置は $\pi=(1,2,3,4)$ 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 1,3 間はセル間の距離が 2, セル間を接続するネット数が 7 本であるため，コストは 2×7 と計算される．同様に，セル配置が $\pi=(2,3,1,4)$ 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，この配置の方が優れた配置である．実際に，この問題例に対しては，$\pi=(2,3,1,4)$ 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 $N(x)$ 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．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，などがある．

4. QAP専用ハードウェア

4.1 QAPに対するタブー探索法

本研究で提案するQAPに対するタブー探索法では，要素(ユニット)の順列ϕを現在の解として保持し，現在の解から近傍解への移動はϕ内のユニット対rとsの交換で実現する．タブーリストは，交換することが一時的に禁止されるユニット対(r,s)で構成される．近傍解への移動において，ユニット対の交換前の割当コストが知られている場合，移動後の近傍解のコストを式(1)に基づいて最初から計算する必要はなく，交換したユニットrとsに関する割当コストの差分を求めることで計算することが可能である．割当コストの差分計算の式を以下に示す．

QAPの許容解である順列ϕにおいて，ユニットiとjを交換したときの目的関数値の差分を$\Delta(\phi,i,j)$する．その中で，タブーリストの制約を満たす移動として，ユニットrとsを交換して順列πに移動したとする．このとき，ϕからπへ移動する際に求めたϕの全ての近傍の差分$\Delta(\phi,i,j)$を記憶しておけば，πにおいて，ユニットuとvを交換したときの差分$\Delta(\pi,u,v)$は以下の式で表される．ただし，u，vはr，sとは異なるものとする[7]．

$$\Delta(\pi,u,v)=\Delta(\phi,u,v)+2(a_{rv}-a_{ru}+a_{su}-a_{sv}) \times(b_{\pi(s)\pi(u)}-b_{\pi(s)\pi(v)}+b_{\pi(r)\pi(v)}-b_{\pi(r)\pi(u)}) \quad (4)$$

また，uかvがrかsに等しい場合の差分計算方法は以下の式で計算される[7]．

$$\Delta(\pi,u,v) = 2\Sigma_{k\neq u,v}(a_{vk}-a_{uk})\times (b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)}) \quad (5)$$

4.2 提案アルゴリズムの概要

本論文で提案するQAPに対するハードウェアアルゴリズムの概要を以下に示す．アルゴリズムはn個のプロセッシングユニット(PU)を一次元配列上に接続したアーキテクチャ上で実行される．提案アルゴリズムのブロック図を図2に示す．提案ハードウェアアルゴリズムは同一構造のPUを局所的な相互接続網で規則正しく接続したアーキテクチャ上で実行されるので，シストリックアルゴリズムとみなすことができる[5]．

4.3 プロセッシングユニットの動作

以下では各PUの動作を表す．

u番目のPUには，問題の入力行列の$A[*,u]$，$B[*,\phi(u)]$，および，前回の差分$\Delta(\phi,u,*)$が，各PUのメモリ内に記憶されているものとする．

最初のクロック$t=1$においてPUの変数vの値をuに設定し，

$$A(u)=a_{su}-a_{ru} \quad (6)$$

$$B(u)=b_{\pi(s)\pi(u)}-b_{\pi(r)\pi(u)} \quad (7)$$

を計算するとともに，これらの値をそれぞれ変数C，Dに設定する($C=A(u)$, $D=B(u)$)．

次のクロック$t=2$においては，vをデクリメント($v=0$になる場合は$v=n$)し，左隣のPUのC，Dの値を右シフトする(右端のPUの値は左端のPUにシフトする)．次に各PUにおいて，以下の式を計算する．

$$\Delta(\pi,u,v)=\Delta(\phi,u,v)+2(A(u)-C)(B(u)-D) \quad (8)$$

以下，同様にして$t=3$，$t=4$まで実行することで，式(4)の値を計算する．

このシストリックアルゴリズムにより，全ての近傍に対し，式(4)を$O(n)$計算時間で評価できる．

次に，uかvがrかsに等しい場合の式(5)を計算するPUの動作を示す．シストリックアルゴリズムは，式(4)の場合と同じく，n個のPUを1次元配列状に接続したアーキテクチャ上で実行される．k番目のPUには入力配列$A[*,k]$，$B[*,\phi(k)]$がメモリに格納されている．各PUは変数u，v，およびPUの番号を表す定数kを持つ．式(4)の場合は定数uがPUの番号に対応していたが，ここではkが対応していることに注意する．各クロックにおいて1番目のPUで，uとvに計算対象となる近傍を設定する．2番目以降のPUは，uとvの値を左隣のPUから受け取る．各クロックにおいて，1番目のPUでは

$$E = (a_{vk}-a_{uk})\times (b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)}) \quad (9)$$

を計算する．2番目以降のPUでは，1クロック前の左隣のPUのEの値を受け取り(E_{in}とする)，以下の式を計算する．

$$E = E_{in}+(a_{vk}-a_{uk})\times (b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)}) \quad (10)$$

ただし，いずれの場合も$u=k$，あるいは$v=k$の場合は加算される積項の値を0とする．このようにして計算を進めていき，右端のPUで式(10)が計算されたならば，その結果は$E=\Delta(\pi,u,v)$となるので，$F=E$，$p=u$，$q=v$として，次のクロックからクロックごとに1PUずつ(F,p,q)の3項組を左にシフトしていき，$k=p$が成り立つPUにおいてFの値を$\Delta(\pi,p,q)$に格納する．式(10)の計算のための値の右シフトと，結果の格納のための左シフトは並列処理で同時に行う．な

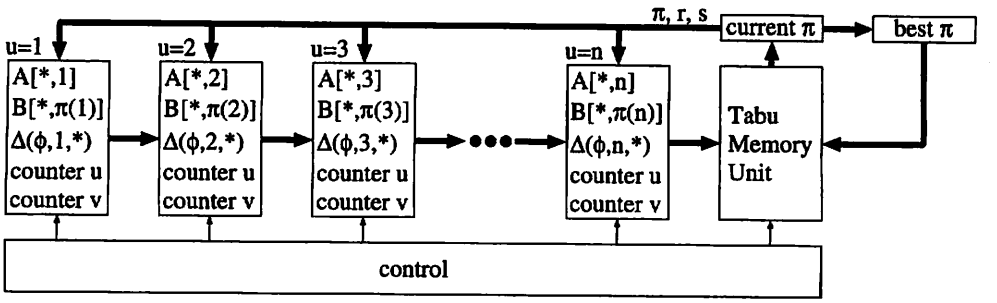

図 2 提案シストリックアレイ

	$u=1$	$u=2$	$u=3$	$u=4$
	$A[*,1]$	$A[*,2]$	$A[*,3]$	$A[*,4]$
	$B[*,\pi(1)]$	$B[*,\pi(2)]$	$B[*,\pi(3)]$	$B[*,\pi(4)]$
	$\Delta(\phi,1,*)$	$\Delta(\phi,2,*)$	$\Delta(\phi,3,*)$	$\Delta(\phi,4,*)$
$t=1$	$v=1$	$v=2$	$v=3$	$v=4$
	$a_{s1}-a_{r1}$	$a_{s2}-a_{r2}$	$a_{s3}-a_{r3}$	$a_{s4}-a_{r4}$
	$b_{\pi(s)\pi(1)}-b_{\pi(r)\pi(1)}$	$b_{\pi(s)\pi(2)}-b_{\pi(r)\pi(2)}$	$b_{\pi(s)\pi(3)}-b_{\pi(r)\pi(3)}$	$b_{\pi(s)\pi(4)}-b_{\pi(r)\pi(4)}$
$t=2$	$v=4$	$v=1$	$v=2$	$v=3$
	$\Delta(\pi,1,4)$	$\Delta(\pi,2,1)$	$\Delta(\pi,3,2)$	$\Delta(\pi,4,3)$
$t=3$	$v=3$	$v=4$	$v=1$	$v=2$
	$\Delta(\pi,1,3)$	$\Delta(\pi,2,4)$	$\Delta(\pi,3,1)$	$\Delta(\pi,4,2)$
$t=4$	$v=2$	$v=3$	$v=4$	$v=1$
	$\Delta(\pi,1,2)$	$\Delta(\pi,2,3)$	$\Delta(\pi,3,4)$	$\Delta(\pi,4,1)$

図 3 式 (4) を計算するシストリックアルゴリズムの動作 ($n=4$).

お，図 4 においては，計算結果を左シフトする過程は省略している．

図 4 に示したシストリックアルゴリズムは，ある $u=r$ と $v=1,2,\ldots,n$ に対して $\Delta(\pi,u,v)$ を $O(n)$ のクロック数で計算可能である．同様に，ある $v=s$ と $u=1,2,\ldots,n$ に対して $\Delta(\pi,u,v)$ を $O(n)$ のクロック数で計算できる．さらに，各 PU が持つメモリは式 (4) を計算するシストリックアルゴリズムと同じであるので，同一のアーキテクチャ上で，現在の許容解である順列 π のすべての近傍を $O(n)$ のクロック数で計算できる．

また，図 4 のシストリックアルゴリズムにおいて，左端の PU において $u=1,2,\ldots,n$ と $v=1,2,\ldots,n$ のすべての組合せで式 (5) を計算することで，各 PU の $\Delta(\pi,u,v)$ の初期値を $O(n^2)$ のクロック数で計算することができる．さらに，行列の対称性に注意してアルゴリズムを構成すれば，必要なクロック数を半減することが可能になる．

現在の許容解のすべての近傍解に対して，割当コストの差分値を計算した後，差分値の大きいユニット対から順にタブーリストと比較を行い，タブーでなければ対応するユニットの順列を次のステップの許容解として選択し，タブーリストの更新を行う．以降，アルゴリズムの終了条件が成立するまで近傍解への移動を繰り返す．詳細は紙面の都合上，省略する．

5. 実験と評価

5.1 実験結果

提案ハードウェア解法の評価実験を行った．QAP のベンチマーク [10] からサイズが 32，64，128 の問題を選んだ．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，提案ハードウェア解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．

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 評価ボードには Altera 社の Quartus II Version5.0，FPGA 評価ボードには三菱電機マイコン機器ソフトウエア株式会社製 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．比較対象のソフトウェアは，C 言語で作成したプログラムを gcc version 3.3.6 に-O3 最適化オプションをつけてコンパイルし，パーソナルコンピュータ (CPU:Pentium4

	$k=1$ $A[*,1]$ $B[*,\pi(1)]$	$k=2$ $A[*,2]$ $B[*,\pi(2)]$	$k=3$ $A[*,3]$ $B[*,\pi(3)]$	$k=4$ $A[*,4]$ $B[*,\pi(4)]$
$t=1$	$u=r,\ v=1$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$			
$t=2$	$u=r,\ v=2$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=1$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$		
$t=3$	$u=r,\ v=3$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=2$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=1$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	
$t=4$	$u=r,\ v=4$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=3$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=2$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=1$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$
$t=5$	$u=1,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=4$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=3$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=2$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$
$t=6$	$u=2,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=1,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=4$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=3$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$
$t=7$	$u=3,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=2,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=1,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=r,\ v=4$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$
$t=8$	$u=4,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=3,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=2,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=1,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$
$t=9$		$u=4,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=3,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=2,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$
$t=10$			$u=4,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$	$u=3,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$
$t=11$				$u=4,\ v=s$ $(a_{vk}-a_{uk})\times$ $(b_{\pi(v)\pi(k)}-b_{\pi(u)\pi(k)})$

図 4　式 (5) を計算するシストリックアルゴリズムの動作 ($n=4$)

3.4GHz，メインメモリ 3GB) 上で実行した．

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 数は回路を実現するのに要した FPGA の論理エレメント数を示し，メモリはメモリの使用数 (bits)，周波数は回路の最高動作周波数を示す．表 2 において，表中のソフトとハードはそれぞれソフトウェアと評価ボードの実行時間 (秒)，速度向上比は速度向上比=(ソフトウェアの実行時間/評価ボードの実行時間) を示す．

表 1 合成結果

サイズ	LE 数	メモリ [bits]	周波数 [MHz]
32	6032 (11%)	10240 (0.2%)	107
64	10576 (19%)	40960 (0.8%)	103
128	21770 (38%)	163200 (3.1%)	87

表 2 実行時間に関する実験結果

サイズ	ソフト [sec]	ハード [sec]	速度向上比
32	2.03	0.23	8.83
64	7.82	0.46	17.00
128	30.90	1.06	29.15

5.2 評価と考察

表 1 で示した実験結果より，パーソナルコンピュータの CPU のクロック周波数は FPGA の動作周波数の 30 倍以上であるにも関わらず，ハードウェアで実現した方がソフトウェアで実現した場合より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，問題サイズに対する実行時間の増加を考慮すると，問題のサイズが大きくなるにつれてハードウェアで実現した方がより有利になることがわかる．具体的には，問題サイズが 2 倍になると，速度向上比は 2 倍に改善されている．このことは，タブー探索法における 1 回の解の移動において，ソフトウェア解法における近傍解の生成時間は $O(n^2)$ であるのに対し，提案ハードウェア解法における近傍解の生成時間は $O(n)$ であることからも説明される．

また，本実験においては，探索回数を 100,000 回に設定して提案手法を実行したが，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解の探索回数が 100,000 回では解空間の探索には不十分であり，優良解を得るためには更なる探索が必要である可能性が高くなる．このため，サイズの大きな問題においては，実用時間内ではソフトウェアは優良解を生成できなくなる可能性が高いがそのような問題においてもハードウェア解法では実用時間内に優良解を生成可能である．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．

さらに大きなサイズの問題を解く場合、1 チップの FPGA に実装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 が市場に出回る可能性は高く，それらの FPGA を用いることで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，本研究で使用した FPGA よりも多数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は，基本的には一次元のシストリックアーキテクチャであるため，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と考えられる．

6. 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シストリックアルゴリズムによる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，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によって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示されており，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に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も興味深い課題である．

本研究の一部は平成 19 年度科学研究費補助金基盤研究 (C)(課題番号 18500042)，および平成 19 年度広島市立大学特定研究費により行った．

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] 木村毅洋, 若林真一, 永山忍, "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," 電子情報通信学会 VLSI 設計技術研究会技術研究報告, VLD2006-91 (2007).

[4] Y. Kimura, S. Wakabayashi, S. Nagayama, "A Systolic Algorithm for the Quadratic Assignment Problem and its FPGA Implementation," *Proc. International Conference on Field Programmable Technology*, pp.261–264 (2007).

[5] H. T. Kung, "Why systolic architectures?," *IEEE Computer*, Vol.15, No.1, pp.37–45 (1982).

[6] 末吉敏則, 天野英晴 (編著), "リコンフィギャラブル・システム", オーム社 (2005).

[7] E.Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel Computing*, 17, pp.443–455 (1991).

[8] S. Wakabayashi, Y. Kimura, S. Nagayama, "FPGA implementation of tabu search for the quadratic assignment problem," *Proc. International Conference on Field Programmable Technology*, pp.269–272 (2006).

[9] 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).

[10] http://www.seas.upenn.edu/qaplib/