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

2003－ＳＬＤＭ－110　(3)
2003／5／15

計算アルゴリズムの局所的類似性とそのデータパス合成への応用

厚見　吉彦†　　大橋　功治†　　金子　峰雄†

† 北陸先端科学技術大学院大学 情報科学研究科　〒923–1292 石川県能美郡辰口町旭台 1–1
E-mail: †{y-atsumi,koohashi,mkaneko}@jaist.ac.jp

あらまし　本稿では，結線数を最小化する観点からの演算，データの演算器，レジスタへの割り当て (資源割り当て) の優先尺度として，演算間あるいはデータ間の構造的類似度・相違度を提案する．提案した類似度・相違度の資源割り当てに対する有効性を評価するために，クリーク分割に基づいた資源割り当て手法に適用した．スケジュール結果から得られる生存期間から，類似度を枝重みとする両立グラフを構成して，クリーク数最小，かつクリークの枝重みの和を最大にするようなクリーク分割問題として，資源割り当てを定式化する．データパス合成実験の結果から，結線数最小のデータパスを生成することを確認した．

キーワード　類似度, 相違度, データパス合成, 資源割り当て, クリーク分割

Local Similarity in Computation Algorithm and Its Application to Data-path Synthesis

Yoshihiko ATSUMI†, Koji OHASHI†, and Mineo KANEKO†

† Japan Advanced Institute of Science and Technology　1–1 Asahidai, Tatsunokuchi, Nomi–gun, Ishikawa, 923–1292 Japan
E-mail: †{y-atsumi,koohashi,mkaneko}@jaist.ac.jp

Abstract　In this paper, novel measures to evaluate a preference in resource assignment, named similarity and dissimilarity between two objects, are proposed. To evaluate the efficacy of our similarity and dissimilarity measures in resource assignment optimization, we applied them to clique partitioning based resource assignment. Edges in a compatibility graph are weighted by the similarity measure, and resource assignment is formulated as to find a clique partitioning of the compatibility graph so that the number of cliques is minimized and total edge weight of those cliques is maximized. Experimental results show that our proposed method provides a data-path with the minimum number of nets.

Key words　Similarity, Dissimilarity, Data-path synthesis, Resource assignment, Clique partitioning

1. まえがき

高位合成は集積システム階層設計の上位に位置し，アルゴリズムレベル動作記述から，演算器，レジスタ，バスなどで構成される構造レベルのレジスタ転送レベル動作記述に合成する技術であり [1]，演算器の性能 (スピード，面積，消費電力) がシステム全体の性能を決める支配的要因であった従来の集積システム設計においては，演算スケジュールと演算器数の最適化が重要であった．しかし配線が極細化された現在及び将来の集積システムにおいては，配線に起因する消費電力や信号伝搬遅延を考慮した最適化がシステム性能にとって重要となって来ている．一般に，トポロジカルな情報だけからモジュール間の結線構造の良否を判断することは難しいが，結線の本数を少なくすることが，レイアウト設計の容易さ，レイアウトの小面積化，消費電力，信号伝搬遅延の最小化の点から，最も重要な設計目標の 1 つとなる．

VLSI 設計において資源割り当ては，各演算を実行する演算器および各変数を保持するレジスタを選択し，それらの接続関係を決定する重要なタスクである．従来の資源割り当て問題においては，整数線形計画問題 (ILP) やクリーク分割問題などに定式化し，それを解く手法が多く提案されている [1]〜[7]．これらは演算スケジュール結果から演算，データの生存時刻を算出し，生存期間が重なる演算，データの同一資源への割り当てを制限した手法である．文献 [1] では，両立可能な 2 つのデータを仮に同一のレジスタに割り当てた場合の結線構造を評価し，その評価値を枝重みとする両立グラフを構成し，演算器割

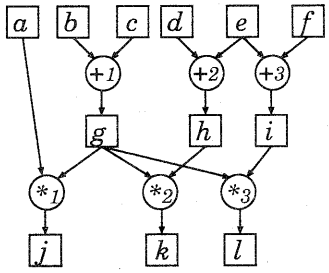


図 1　先行制約グラフ.

り当て後のレジスタ割り当てを重み付きクリーク分割として定式化し，その発見的アルゴリズムが示されている．一方，資源割り当てをスケジュールより優先して行ない，スケジュールによって制限されない資源割り当て解空間を結線構造に関わる諸特性を正確に評価しながら探索する資源割り当て優先探索手法が提案されている [8], [9]．スケジュールを開始点とする逐次的な処理を行なう従来の合成システムと比べて，配線資源が少ないデータパスを生成することが報告されているが，その資源割り当て探索は分枝限定法や Simulated Annealing 法に基づいているため，多大な探索時間を要する．

本稿では，結線数を最小化する観点からの演算，データの演算器，レジスタへの割り当ての優先尺度として，演算間やデータ間の類似度・相違度を提案する．また，提案した類似度・相違度の資源割り当てに対する有効性を評価するために，クリーク分割を用いた資源割り当て手法に適用した．資源割り当てに当たっては，先ずスケジュール結果から得られる生存期間から両立グラフを構成して，これをクリーク分割することで資源割り当てを決定する．この時，両立グラフの枝に提案する類似度を重みとして付け，クリーク数を最小にし，かつ分割されたクリークの枝重み和を最大にするようなクリーク分割を求める．

以下，2 節では本稿で取り扱う用語やモデルを定義し，3 節にて類似度・相違度を提案する．4 節にて類似度用いた資源割り当て手法を提案し，その実験結果を 5 節にて示す．6 節では，まとめと今後の課題について述べる．

2. 準　　備

- 先行制約グラフ

  入力とする計算アルゴリズムは有向グラフ $G = (V_G, A_G)$ で表されるものとする．ここで，V_G は演算集合 V_O とデータ集合 V_D の和集合，A_G はデータ頂点から演算頂点に向かう枝の集合 A_I と演算頂点からデータ頂点に向かう枝の集合 A_O の和集合である．各演算，データにはタイプ $t : V_G \to \mathcal{T}$ が与えられている．但し $\mathcal{T}$ は演算タイプ集合 $\mathcal{T}_O = \{+, -, \times, \cdots\}$ とデータタイプ集合の和集合である．ここでは簡単化のために，各データのタイプは全て同じであると仮定する．図 1 に先行制約グラフの例を示す．

- データパスモデル

  ここではマルチプレクサタイプのアーキテクチャを対象とする．構成要素は演算器，レジスタ，複数入力を制御するマルチプレクサ，資源間結線である．図 2 にデータパスモデルの例を示す．

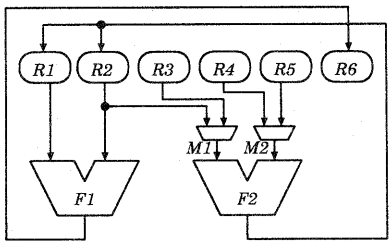


図 2　データパス.

- 演算スケジュール $\sigma : V_O \to Z$.

- 資源割り当て $\rho : V_O \to \mathcal{F}$, $\xi : V_D \to \mathcal{R}$, 但し $\mathcal{F}$ は演算器の集合，$\mathcal{R}$ はレジスタの集合である．

3. グラフ構造における類似度・相違度

1 つの演算に対して，入力となるデータを保持するレジスタから演算器へのデータ転送と演算器から出力データを書き込むレジスタへのデータ転送は，必要不可欠である．こうしたデータ転送の総和は入力の計算アルゴリズム (先行制約グラフ) のみで決まり，データパス構造に依存しない．従ってモジュール間の結線数を減らすためには，なるべく多くのデータ転送が 1 つの結線を共有して行なわれる事が重要となる．ここで，データ転送の構造が似通ったもの同士を同一資源に割り当てることで，効率的に複数のデータ転送の結線共有が行なわれ，似ていないもの同士を同一資源に割り当てた場合には，どちらか一方のみのデータ転送に使われる結線が増加して，効率的な結線共有が阻害されると考える．

上記の考え方に基づいて，結線数最小化の観点から，演算，データの演算器，レジスタへの割り当て (資源割り当て) の優先尺度として，演算間やデータ間の類似度・相違度を提案する．

3.1 類似度・相違度

はじめに，類似度・相違度を定義する際に重要となるタイプ付き最大同型部分グラフ問題は以下のようなものである．

[タイプ付き最大同型部分グラフ問題]

入力:　グラフ $G_1 = (V_{G_1}, A_{G_1})$, $G_2 = (V_{G_2}, A_{G_2})$.

出力:　部分グラフ $G'_1 = (V'_{G_1}, A'_{G_1}) \subseteqq G_1$, $G'_2 = (V'_{G_2}, A'_{G_2}) \subseteqq G_2$. 但し，任意の頂点 $v_i \in V'_{G_1}$ に対して，同じタイプの頂点 $\varphi(v_i) \in V'_{G_2}$ (すなわち $t(v_i) = t(\varphi(v_i))$) が存在し，$(v_j, v_k) \in A'_{G_1} \Leftrightarrow (\varphi(v_j), \varphi(v_k)) \in A'_{G_2}$ であるような 1 対 1 写像 $\varphi : V'_{G_1} \to V'_{G_2}$ が存在する．

目的:　枝数 $|A'_{G_1}| = |A'_{G_2}|$ 最大.

部分同型判定問題は $\mathcal{NP}$ 完全であることが知られており [10], タイプ付き最大同型部分グラフ問題は $\mathcal{NP}$ 困難である．

[定義 1]　先行制約グラフ $G = (V_G, A_G)$ において，枝の向きを無視し，2 頂点間の最短パス中に含まれる枝の数を距離とし，

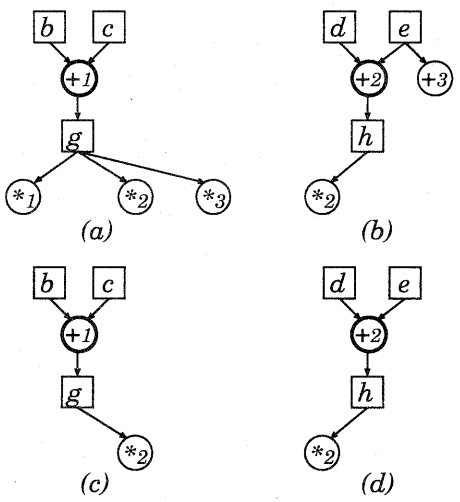


図 3 $+_1$ と $+_2$ の類似度と相違度.

基準点 $v \in V_G$ からの距離が k 以下の頂点集合から誘導される G の部分グラフを v の k 近傍グラフと呼ぶ.

[定義 2] 先行制約グラフ $G=(V_G, A_G)$ において，タイプが同じである 2 つの頂点 $v_1, v_2 (t(v_1)=t(v_2))$ に対して，v_1, v_2 の k 近傍グラフをそれぞれ $G_1=(V_{G_1}, A_{G_1}), G_2=(V_{G_2}, A_{G_2})$ とする．基準点の写像が指定され $(\varphi(v_1)=v_2)$，かつ $v_1 \in V_{G_1}$ と $v_2 \in V_{G_2}$ をそれぞれ含むタイプ付き最大同型連結部分グラフを $G'_1=(V'_{G_1}, A'_{G_1}) \subseteqq G_1$，$G'_2=(V'_{G_2}, A'_{G_2}) \subseteqq G_2$ とする．この時，枝数 $|A'_{G_1}|=|A'_{G_2}|$ を v_1 と v_2 の**類似度** $s(v_1, v_2)$ とし，$|A_{G_1} \setminus A'_{G_1}| + |A_{G_2} \setminus A'_{G_2}|$ を v_1 と v_2 の**相違度** $d(v_1, v_2)$ とする.

図 1 の先行制約グラフにおいて，同じタイプの頂点 $+_1$ と $+_2$ の 2 近傍グラフをそれぞれ図 3(a)，(b) に示す.

この 2 つのグラフにおいて，$+_1$ の写像先が $+_2$ であって，かつ $+_1$ と $+_2$ をそれぞれ含むタイプ付き最大同型連結部分グラフを図 3(c)，(d) に示す．図 3(c)，(d) のグラフの枝数はそれぞれ 4 であり，$+_1$ と $+_2$ の類似度 $s(+_1, +_2)$ は 4 と計算される．同様に，図 3(a)，(b) のグラフの総枝数が 11 であることから，相違度 $d(+_1, +_2)$ は 3 となる.

3.2 計算アルゴリズム

2 つのグラフから基準点が指定されても，最大同型部分グラフ問題は依然 $\mathcal{NP}$ 困難である．ここでは類似度の計算に必要な最大同型連結部分グラフ問題を分枝限定法を用いて解くものとする．但し，一方のグラフの頂点から他方のグラフの頂点への写像を分枝限定法にて探索するとすれば，基準点との連結成分を別途計算する必要があり，また無駄な探索も多くなると予想される．そこで，一方のグラフの枝から他方のグラフの枝への写像を探索することとする.

すなわち，探索途中での同型連結部分グラフを $G'_1=(V'_{G_1}, A'_{G_1})$，$G'_2=(V'_{G_2}, A'_{G_2})$ とする時，

SIMILARITY(G_1, G_2, v_1, v_2)

1 $\varphi(v_1)=v_2$，$V'_{G_1}=\{v_1\}$，$V'_{G_2}=\{v_2\}$，$s(v_1, v_2)=0$.
2 BAB(G_1, G_2, G'_1, G'_2) を実行.
3 v_1 と v_2 の類似度 $s(v_1, v_2)$ を出力.

BAB(G_1, G_2, G'_1, G'_2)

1 枝 $(v_i, v_j) \in A'_{G_1}, v_i \in V'_{G_1}, v_j \notin V'_{G_1}$，あるいは，$v_i \notin V'_{G_1}, v_j \in V'_{G_1}$ を 1 つ選ぶ.
このような枝がない時，$|V'_{G_1}|$ を評価し，必要に応じて $s(v_1, v_2)$ を更新し，return.
2 選ばれた枝が $(v_i, v_j), v_i \in V'_{G_1}, v_j \notin V'_{G_1}$ であるとき，
2.1 各枝 $(\varphi(v_i), v_x) \in A_{G_2}, v_x \notin V'_{G_2}$ に対して，以下を実行する.
2.1.1 $\varphi(v_j)=v_x$,
$V'_{G_1} \leftarrow V'_{G_1} \cup \{v_j\}$，$V'_{G_2} \leftarrow V'_{G_2} \cup \{v_x\}$.
2.1.2 各枝 $(v, v_j) \in A_{G_1}, v \in V'_{G_1}$ に対して，$\varphi(v) \neq$ NULL, かつ枝 $(\varphi(v), \varphi(v_j)) \in A_{G_2}$ が存在するならば,
$A'_{G_1} \leftarrow A'_{G_1} \cup \{(v, v_j)\}$,
$A'_{G_2} \leftarrow A'_{G_2} \cup \{(\varphi(v), \varphi(v_j))\}$.
各枝 $(v_j, v) \in A_{G_1}, v \in V'_{G_1}$ に対して，$\varphi(v) \neq$ NULL, かつ枝 $(\varphi(v_j), \varphi(v)) \in A_{G_2}$ が存在するならば,
$A'_{G_1} \leftarrow A'_{G_1} \cup \{(v_j, v)\}$,
$A'_{G_2} \leftarrow A'_{G_2} \cup \{(\varphi(v_j), \varphi(v))\}$.
2.1.3 BAB(G_1, G_2, G'_1, G'_2).
2.1.4 $\varphi(v_j)=$ NULL，2.1.1，2.1.2 にて追加した頂点，枝を削除.
2.2 $\varphi(v_j)=$ NULL として，BAB(G_1, G_2, G'_1, G'_2).
3 1 で選ばれた枝が $(v_i, v_j), v_i \notin V'_{G_1}, v_j \in V'_{G_1}$ であるとき，2 において v_i と v_j の役割を入れ替えた計算処理 (詳細は略).
4 return.

図 4 類似度計算アルゴリズム.

$(v_i, v_j) \in A'_{G_1}, v_i \in V'_{G_1}, v_j \notin V'_{G_1}$

(あるいは, $(v_i, v_j) \in A'_{G_1}, v_i \notin V'_{G_1}, v_j \in V'_{G_1}$)

と，これに対応して G_2 の枝集合

$$\{(\varphi(v_i), v) | (\varphi(v_i), v) \in A_{G_2}, v \notin V'_{G_2}\} \triangleq A''_{G_2}$$
$$(\{(v, \varphi(v_j)) | (v, \varphi(v_j)) \in A_{G_2}, v \notin V'_{G_2}\} \triangleq A''_{G_2})$$

に注目し，枝 $(v_i, v_j) \in A_{G_1}$ の A''_{G_2} の枝 $(\varphi(v_i), v_x) \in A''_{G_2}$ への写像 ($(v_y, \varphi(v_j)) \in A''_{G_2}$ への写像) の追加 (これに伴って，$\varphi(v_j)=v_x (\varphi(v_i)=v_y)$ を追加)，あるいは，枝 $(v_i, v_j) \in A_{G_1}$ の NULL への写像の追加を分枝とする.

これにより，G'_1, G'_2 が基準点を含む連結グラフであることを保ったまま探索することが可能となる.

4. 局所的類似性を用いた資源割り当て手法

4.1 資源割り当て問題

配線が極細化された現在及び将来の集積システムにおいては，配線に起因する消費電力や信号伝搬遅延を考慮した最適化がシステム性能にとって重要となって来ている．一般に，モジュール間の結線構造の良否を判断することは難しいが，結線の本数を少なくすることが，レイアウト設計の容易さ，レイアウトの小面積化，消費電力，信号伝搬遅延の最小化の点から，最も重

要な設計目標となる．

VLSI 設計において資源割り当ては，各演算を実行する演算器および各変数を保持するレジスタを選択し，それらの接続関係を決定する重要なタスクである．ここではスケジュール後の資源割り当てを想定し，次のように定式化する．

[資源割り当て問題]

入力: 先行制約グラフ $G=(V_G,A_G)$，演算スケジュール σ.

出力: 資源割り当て ρ,ξ.

目的: 結線数最小.

演算スケジュールが指定された資源割り当ては，しばしばクリーク分割問題として定式化される．文献[1]では，演算器割り当て後のレジスタ割り当てを重み付きクリーク分割問題として定式化している．その枝重みは，枝の両端点に対応するデータが同一のレジスタに割り当てられたと想定し，結線数の減数とマルチプレクサの入力の増減数によって決定される．結線構造を正確に評価しているが，あくまでも演算器割り当てが決まった後のデータのレジスタへの割り当てのみの適用にとどまっている．

4.2 スケジュール指定下での類似度・相違度

我々が 2 つの頂点間の類似度として，これら頂点を基準点とする同型連結部分グラフに注目する背景には，同型を規定する写像 φ にて対応づけられる頂点同士を同一の演算器あるいはレジスタに割り当てることにより，同型連結部分グラフの対応するどの枝対も同一の結線を共有できるとの想定がある．ところでスケジュールが与えられている時には，任意の演算対あるいはデータ対が同一の資源を共有できる訳ではなく，生存期間に重なりがあるもの同士は共有できないことになる．

このことを類似度を求める際の最大同型連結部分グラフに反映させることにする．すなわち，スケジュール指定下での類似度・相違度を以下のように定義する．

[定義 3] 先行制約グラフ $G=(V_G,A_G)$ と生存期間に重なりがなくタイプが等しい 2 つの頂点 $v_1,v_2\in V_G$ 対して，v_1,v_2 の k 近傍グラフを $G_1=(V_{G_1},A_{G_1}),G_2=(V_{G_2},A_{G_2})$ とする．連結部分グラフ $G_1'=(V_{G_1}',A_{G_1}')\subseteqq G_1$，$G_2'=(V_{G_2}',A_{G_2}')\subseteqq G_2$ に対して，以下の条件を満たす $\varphi:V_{G_1}'\to V_{G_2}'$ が存在するとき，これをタイプと生存期間付き同型連結部分グラフと呼ぶ．

1 $v_1\in V_{G_1}',v_2\in V_{G_2}',\varphi(v_1)=v_2$.
2 $(v_i,v_j)\in A_{G_1}'\Leftrightarrow(\varphi(v_i),\varphi(v_j))\in A_{G_2}'$.
3 $\forall v_i\in V_{G_1}',t(v_i)=t(\varphi(v_i))$.
4 $\forall v_i\in V_{G_1}',v_i$ の生存期間と $\varphi(v_i)$ の生存期間に重なりがない．

[定義 4] タイプと生存期間付き同型連結部分グラフの中で枝数最大のものを $G_1'=(V_{G_1}',A_{G_1}')$，$G_2'=(V_{G_2}',A_{G_2}')$ とするとき，枝数 $|A_{G_1}'|=|A_{G_2}'|$ を v_1 と v_2 の**類似度** $s(v_1,v_2)$ とし，$|A_{G_1}\setminus A_{G_1}'|+|A_{G_2}\setminus A_{G_2}'|$ を v_1 と v_2 の**相違度** $d(v_1,v_2)$ とする．

4.3 重み和最大クリーク分割問題

資源割り当て問題を取り扱うに当たって，演算スケジュール結果と先に定義した類似度を利用して，以下に定義する演算/データ両立グラフを導入する．

先行制約グラフにおいて，2 つの演算の生存期間が重複せず，同じ型の演算器で実行できる時，これらの演算は両立可能であるという．

[定義 5] 演算両立グラフは枝重み付き無向グラフ $G_O=(V_O,E_O)$ である．ここで頂点 $v_i\in V_O$ は演算，枝 $(v_j,v_k)\in E_O\subset V_O\times V_O$ は演算 v_j,v_k が両立可能であることを表し，演算 v_j,v_k の類似度 $s(v_j,v_k)$ をその枝の重みとする．

同様に，2 つのデータの生存期間が重複しなければ，これらのデータは両立可能であるという．

[定義 6] データ両立グラフは枝重み付き無向グラフ $G_D=(V_D,E_D)$ である．ここで頂点は $v_i\in V_D$ データ，枝 $(v_j,v_k)\in E_D\subset V_D\times V_D$ はデータ v_j,v_k が両立可能であることを表し，データ v_j,v_k の類似度 $s(v_j,v_k)$ をその枝の重みとする．

両立グラフの定義から両立グラフ中のクリーク (完全部分グラフ) を 1 つの演算器 (又はレジスタ) を共有する演算 (又はデータ) に対応させることができる．先に述べたように，結線数最小化の観点から，データ依存の構造が似通ったもの同士を同一資源に割り当てるという方針から，1 つの演算器 (又はレジスタ) の共有に関する評価尺度としてクリークの枝重み和を考える．

[定義 7] クリーク $C_i=(V_i,E_i)$ の重み $w(C_i)$ を，

$$w(C_i)=\sum_{(v_i,v_j)\in E_i}s(v_i,v_j)$$

とする．

演算器数最小のもとで，結線数を最小にする演算器割り当てとして以下の問題を考える．

[重み和最大クリーク分割問題]

入力: 演算両立グラフ $G_O=(V_O,E_O)$.

出力: G_O のクリーク分割 $C=\{C_1,C_2,\cdots,C_n\}$. ただし，クリーク $C_i=(V_i,E_i)$ とすると，$V_O=V_1\cup V_2\cup\cdots\cup V_n$，かつ $V_i\cap V_j=\emptyset,\forall i\neq j$.

目的: クリーク数 n が最小のなかで，クリークの重みの総和 $\sum_{i=1}^{n}w(C_i)$ が最大.

レジスタ割り当てについては，データ両立グラフを用いて同様に定義される．

図 5 に演算両立グラフの例を示す．このグラフに対して，最小となるクリーク数 2 のクリーク分割は，$\{\{+_1,+_2\},\{+_3,+_4\}\}$，$\{\{+_1,+_3\},\{+_2,+_4\}\}$，$\{\{+_1,+_2,+_3\},\{+_4\}\}$，$\{\{+_1\},\{+_2,+_3,+_4\}\}$，の 4 通りだけである．また，クリークの重みの総和は，それぞれ 8，3，7，6 である．従って，クリーク数が最小で，重み和が最大のクリーク分割は図 5 に示す $\{\{+_1,+_2\},\{+_3,+_4\}\}$ である．

4.4 重み和最大クリーク分割アルゴリズム

重み和最大クリーク分割問題を解く発見的アルゴリズムを図 6 に示す．このアルゴリズムは重み和最小クリーク分割のアルゴリズム[7]を重み和が最大となるように変更したものであり，基本的な動作は同じである．

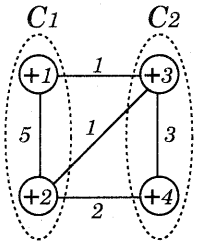


図 5　重み和最大クリーク分割.

はじめに各頂点にクリークを割り当てる．select_pair1(C) では，両立グラフの枝重みが大きいものを優先して，同じクリークにできる最初の組を選択し，それらのクリークをマージする．ここで 2 つのクリーク C_i, C_j をマージする (merge(C_i, C_j)) とは，両立グラフにおいて，クリーク C_i, C_j に対応する頂点を v_i, v_j として，

$$E_c \leftarrow E_c \setminus \{\{\{v_i, v\} \in E_c : \{v_j, v\} \notin E_c\} \cup \{\{v_j, v\} \in E_c\}\}$$
$$V_c \leftarrow V_c \setminus \{v_j\}$$

なる変更を行なうことを表している．select_pair1(C) を L_{max} 回繰り返す (すなわち，最初の組の選択を変更する) ことによって，最適なクリーク分割をより広く探索している．

select_pair2(C) では，優先度 $H(h_c, h_w)$ に基づいて，同じクリークにできる組を選択し，それらのクリークをマージする．これを同じクリークにできる組がなくなるまで行なう．(すなわち，両立グラフに枝がなくなるまでクリークのマージを行なう) ここで優先度 h_c は (α, β) の 2 項組からなり，

$$\alpha(v_i, v_j) = |\{v \in V_c : \{v, v_i\} \in E_c \wedge \{v, v_j\} \in E_c\}|$$
$$\beta(v_i, v_j) = |\{v \in V_c : (\{v, v_i\} \in E_c \wedge \{v, v_j\} \notin E_c) \vee (\{v, v_i\} \notin E_c \wedge \{v, v_j\} \in E_c)\}|$$

である．すなわち，選択したマージするクリークの組によって，α は他にマージできる可能性のあるクリーク数，β はマージできなくなるクリーク数を表している．従って，大きい α で，かつ小さい β をもつ組を選択するとによって，クリーク数最小の分割に指向する．

一方，優先度 h_w はマージすることによってクリークの重みが増える量を表し，クリークの重みの和を大きくするために評価する．

このクリーク分割は，以前よりもクリーク数が小さい (N(C) > N(C_{best})) か，またはクリーク数が同じでクリークの重みの和が大きい ((N(C) == N(C_{best})) && (W(C) < W(C_{best}))) 分割ならば，クリーク分割の解 C_{best} を更新していく．全体の繰り返し回数 L_{max} については，実験的に評価するものとする．

5. 実験結果

提案した資源割り当て手法を C 言語にて計算機実装を行なった．先行制約グラフの各頂点の 2 近傍グラフにおける類似度を両立グラフの枝重みとした．重み和最大クリーク分割は，

```
WEIGHTED_MAX_CLIQUE(G_c = (V_c, E_c))
{
    while(L < L_max) {
        ++L;
        C = {V_c};

        (C_s1, C_s2) = select_pair1(C);
        merge(C_s1, C_s2);

        while(Exist a pair of merged cliques) {
            (C_i, C_j) = select_pair2(C);
            merge(C_i, C_j);
        }

        if((N(C) > N(C_best)) ||
           ((N(C) == N(C_best)) && (W(C) < W(C_best)))) {
            C_best = C;
        }
    }
}
```

図 6　重み和最大クリーク分割の発見的アルゴリズム.

表 1　Differential Equation における実験結果.

	T_r	FU	R	#CON
手法 1	7	1ALUs,2×	5	14
手法 2	7	1ALUs,2×	5	14

手法 1:　生存期間を考慮して最初に全頂点間の類似度を計算しておいて，クリーク分割を行なう方式

手法 2:　1 つのタイプに関して類似度計算とクリーク分割を行なって資源割り当てを決定し，その結果を先行制約グラフに反映させて，次のタイプの類似度計算とクリーク分割を行なう方式

について，資源割り当てを求める実験を行った．

はじめに，先行制約グラフとしてベンチマークの Differential Equation と 1 つのパイプラインスケジュールを入力として用いた．手法 1 と手法 2 それぞれの実験結果を表 1 に示す．表中の T_r はパイプラインスケジュールにおける繰り返し周期，FU は演算器数，R はレジスタ数を表している．#CON はモジュール間の結線器数を表している．ただし，乗算器への定数入力部を除外している．実験結果では，手法 1 と手法 2 ともに最小の結線数を得ることができた．

次に，先行制約グラフとして図 7 に示される Four-Order All-Pole Lattice Filter と 1 つのパイプラインスケジュールを入力として用いた．手法 1 と手法 2 それぞれの実験結果を表 2 に示す．手法 1 においては，最小の結線数から 1 本多い結果を得たが，手法 2 においては最小の結線数を得ることができた．手法 2 では，1 つのタイプの資源割り当てを決定する度に類似度を再計算しているため，類似度を更新しない手法 1 に比べ，より正確なグラフの類似性を評価することができ，より少ない結線数を得ることができたと考える．ただし，本実験ではタイ

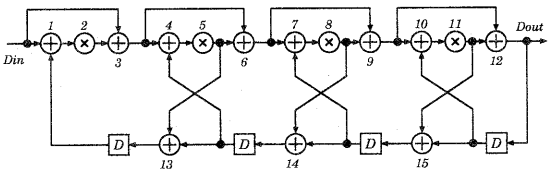


図 7　Four-Order All-Pole Lattice Filter.

表 2　Four-Order All-Pole Lattice Filter における実験結果.

	T_r	FU	R	#CON
手法 1	8	3+,2×	8	26
手法 2	8	3+,2×	8	25

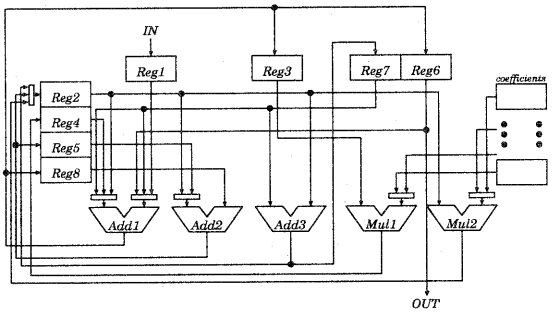
プ毎の決定順序は加算，乗算，データと行ない類似度を更新したが，どの決定順序がより良い解を生成するかは解析できていない．手法 1，手法 2 にそれぞれによって合成されたデータパスを図 8，図 9 に示す．

手法 2 では最小の結線数を得ることができたが，2 種類のベンチマークはともに小規模であり，またそれぞれ 1 つのスケジュールでしか実験を行なっていない．大規模なベンチマーク，及び様々なスケジュールにおける実験は，今後の課題として残されている．

6. ま と め

本稿では，結線数を最小化する観点からの演算，データの演算器，レジスタへの割り当ての優先尺度として，演算間やデータ間の類似度・相違度を提案した．類似度を枝重みとして利用した両立グラフを構成し，資源割り当てを重み和最大クリーク分割問題として定式化した．小規模ながら 2 種類のベンチマークを用いて実験を行ない，提案手法が結線数最小のデータパスを生成することを確認した．

今後，資源割り当てをスケジューリングに優先して行なうデータパス合成手法に対しても，提案した類似度・相違度の活用方法を考案し，その有効性を評価する予定である．



$Add1 : +_1, +_3, +_{10}, +_{12}, +_{15}$　$Add2 : +_4, +_6, +_7, +_{14}$

$ADD3 : +_9, +_{13}$

$MUL1 : \times_2, \times_{11}$　$MUL2 : \times_5, \times_8$

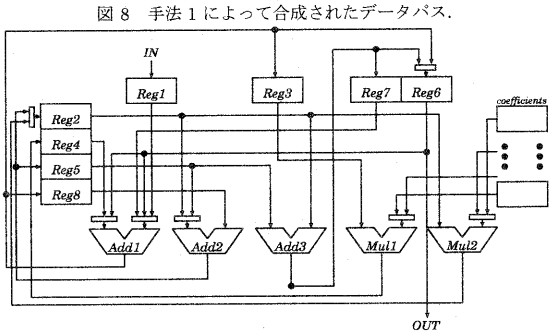
$Reg1 : D_{in}$　$Reg2 : D_4, D_5, D_7, D_8, D_{13}$

$Reg3 : D_1, D_{10}$　$Reg4 : D_2, D_{11}$

$Reg5 : D_6, D_{14}$　$Reg6 : D_{12}$

$Reg7 : D_9$　$Reg8 : D_3, D_{15}$

図 8　手法 1 によって合成されたデータパス.



$Add1 : +_1, +_3, +_{10}, +_{12}, +_{15}$　$Add2 : +_4, +_6, +_7, +_{14}$

$Add3 : +_9, +_{13}$

$Mul1 : \times_2, \times_{11}$　$Mul2 : \times_5, \times_8$

$Reg1 : D_{in}$　$Reg2 : D_4, D_5, D_7, D_8$

$Reg3 : D_1, D_{10}$　$Reg4 : D_2, D_{11}$

$Reg5 : D_6, D_{14}$　$Reg6 : D_{12} D_{13}$

$Reg7 : D_9$　$Reg8 : D_3, D_{15}$

図 9　手法 2 によって合成されたデータパス.

謝　辞

本研究を進めるにあたり，貴重な助言をいただいた北陸先端科学技術大学院大学 情報科学研究科 田湯智助手に深く感謝する．本研究の一部は，日本学術振興会科学研究費補助金 基盤 (C) 課題番号 14550321 の援助を受けた．

文　献

[1] P. Michel, U. Lauther, P. Duzy, "The synthesis approach to digital system design," Kluwer Adademic, 1992.

[2] M. Rim, R. Jain and R. De Leone, "Optimal Allocation and Binding in High-Level Synthesis", 29th ACM/IEEE Design Automation Conference, 1992.

[3] P. Ellervee, S. Kumar, A. Hemani, "Comparison of Four Heuristic Algorithems for Unified Allocation and Binding in High-Level Synthesis", NORCHIP'97 - The 15th NORCHIP Conference, November 10-11, pp.60–66, 1997.

[4] F.J. Kurdahi, A.C. Parker, "REAL: A program for register allocation," Proc. 24th ACM/IEEE Design Automation Conference, pp.511-516, 1987.

[5] C.-Y. Huang, Y.-S. Chen, Y.-L. Lin, Y.-C. Hsu, "Data path allocation based on bipartite weighted matching," Proc. 27th ACM/IEEE Design Automation Conference, pp.499-504, 1990.

[6] F.-S. Tsay, Y.-C Hsu, "Data path construction and refinement," In Digest of Technical Papers, ICCAD, pp.308–311, 1990.

[7] 高橋智也, 井上智生, 藤原秀雄, "無閉路部分スキャン設計に基づくデータパスのテスト容易化高位合成におけるバインディング手法", 電子情報通信学会論文誌, D-I, Vol.J83-D-I, NO2, pp.282–292, 2000 年 2 月.

[8] K. Oohashi, M. Kaneko, S.Tayu, "Assignment-Space Exploration Approach to Concurrent Data-Path/Floorplan Synthesis," Proc. ICCD2000, pp.370-375, 2000.

[9] T. Yorozuya, K. Ohashi, M. Kaneko, "Assignment-Driven Loop Pipeline scheduling and Its Application to Data-Path Synthesis," IEICE Trans. Fundamentals, Vol.E85-A, No.4, pp.819–826, April 2002.

[10] Michael R.Garey and David S.Johnson, "Computers and Intractability: a Guide to the Theory of NP-Completeness", Freeman, 1991.