プログラミング-言語・基礎・実践- 21－19
(1995. 3. 10)

バス結合型並列処理システムによるハミルトン閉路求解

山瀧 倫明　　黒川 恭一　　松原 隆　　古賀 義亮

防衛大学校 情報工学教室

〒２３９ 神奈川県横須賀市走水１－１０－２０
電子メール：MATUBARA@JPNNDA.BITNET

あらまし ＤＳＰを用いて構成した複数のＰＥをバス結合した並列処理システムを１９９３年に試作した[2]．グラフ理論の分野からハミルトン閉路求解問題を取り上げ，求解アルゴリズムであるエニュメレーションメソッド及びマルチパスメソッドを本システムに実装してハミルトン閉路を求めた．各アルゴリズムの処理能力及びに本システムの並列処理能力とその制約要因等について報告する．

キーワード　並列処理，バス結合，ＤＳＰ，ハミルトン閉路

Experimental results on Hamiltonian Cycles by using a Bus Connected Parallel Processing System

Michiaki YAMATAKI　Takakazu KUROKAWA　Takashi MATSUBARA　Yoshiaki KOGA

Department of Computer Science, National Defense Academy

1-10-20 Hashirimizu, yokosuka 239, Kanagawa
E-mail : MATUBARA@JPNNDA.BITNET

Abstract　A parallel processing system was experimentally produced in 1993. This system consists of bus connected PE's composed of DSP. We try to get Hamiltonian Cycles well-known in the field of graph theory by using two algorithms, enumeration and multi-path methods. In this paper, we report experimental results given by those algorithms and parallel processing of this system and give some restrictions of parallel processing on this problem.

key words　Parallel Processing, Bus Connection, DSP, Hamiltonian Cycles

1．　はじめに

近年，各種の応用分野で，より高速な処理を要求する場合があり，それに答えるため専用プロセッサによる並列処理システムの研究開発が行われている[1]．

当研究室では1993年に，ＤＳＰを用いて構成したＰＥをバスで結合した並列処理システムを試作し[2]，信号処理関連のアプリケーションによって性能評価を行っている[3]．本報告では，グラフ理論の分野からハミルトン閉路求解問題を取り上げ，これを解くためにエニュメレーションメソッド及びマルチパスメソッドという２つのアルゴリズムを本システムに実装し，ハミルトン閉路を求めた結果を報告する．

2．　システムの概要

本システムは，複数のＰＥをバス結合したＭＩＭＤ型並列処理システムである．バス結合方式による並列処理システムの構成は，ＰＥの増設によるシステムの拡張が容易であること，全てのＰＥへデータをブロードキャストすることが可能であること等の特長がある．

2.1　ハードウエア構成

ハードウエアの構成は図１のとおりである．データの入出力及びシステムの監視用にホストとしてパーソナルコンピュータ（ＰＣ－２８６ＶＧ）を置き，システムバスは，３２ビットデータバス，１５ビットアドレスバス，２１本のコントロールバスと６ビットアービトレーションバスから構成されている．ＰＥの実装数は現在のところ，４ＰＥであるが最大６４ＰＥまで拡張可能である．

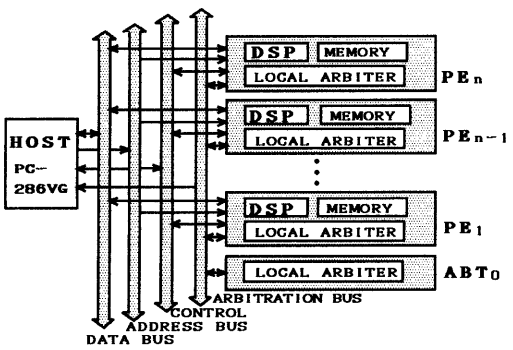

図１　ハードウエア構成

2.2　ＰＥの概要

ＰＥ（プロセッシングエレメント）は，３２ビットＤＳＰを中心とし，３２ｋワードのインストラクションＲＡＭ及びデータＲＡＭ，１チップのＰＡＬ上に構成した６ビットローカルアービタ，インタフェース回路等から構成されている．

2.3　ＤＳＰの概要

ＤＳＰはＮＥＣ製のμＰＤ７７２４０を使用している．このＤＳＰの処理データは３２ビット浮動小数点データ，２４ビット固定小数点データである．処理速度は９０ｎｓ／１インストラクションを限度としているが，本システムではシステムクロック（８ＭＨｚ）の制約により，１２５ｎｓ／１インストラクションとなっている．このＤＳＰは内部の乗算器（３２ビット×３２ビット）で高速積演算が行えること，インストラクションＲＯＭ及びデータＲＯＭにマトッリクス演算ライブラリがあらかじめ書き込まれていることや，３ステージのパイプライン処理が行えること等から，高速，高精度の要求されるディジタル信号処理に適している．さらに２組の独立したデータＲＡＭ（５１２×３２ビット）をレジスタと同様に使用できるなど信号処理以外の複雑な処理にも十分対応できるものである[4]．

2.4　ホストとＰＥ間の通信

ホストとＰＥ間の通信機能としては，次の４機能がある．

(1) ホストから特定のＰＥまたは全ＰＥへのデータのダウンロード．

(2) ホストから特定のＰＥまたは全ＰＥへの処理開始信号の送信．

(3) ＰＥからホストへの全ＰＥの処理終了信号及びアービトレーション信号の送信．

(4) ＰＥのデータのホストへのアップロード．

2.5　ソフトウエア及び開発環境

ソフトウエア構成はホスト用とＰＥ用に分かれており，ホスト用プログラムの機能はＰＥとの通信，ＰＥの処理時間（ｍｓｅｃ単位）測定，ＰＥ用プログラムの開発支援であり，またＰＥ用プログラムの機能はホストとの通信，ハミルトン閉路求解処理である．

開発環境は，開発マシンとしてＰＣ－２８６ＶＧ，ＯＳはＭＳ－ＤＯＳ，言語に関しては，

ホスト用はC言語，ＰＥ用はμＰＤ７７２４０アセンブリ言語[6]である．ＰＥ用プログラムの開発支援機能としては，ＰＥの処理終了後，ホストでＰＥのデータＲＡＭの記憶内容を表示することが可能である．

3．　ハミルトン閉路の求解

3.1　ハミルトン閉路

ハミルトン閉路とは，頂点とａｒｃからなるいわゆる有向グラフにおける全ての頂点をちょうど一度づつ通る閉路のことである．例として頂点数が６のグラフを図２(a)に示す．このグラフには図２(b)に太線で示すようにハミルトン閉路が２つ存在する．

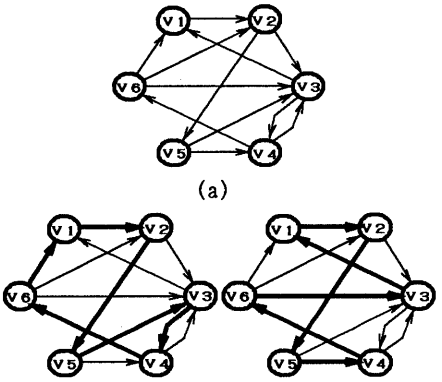

図２　グラフのハミルトン閉路

3.2　ハミルトン閉路求解アルゴリズム

3.2.1　エニュメレーションメソッド

アルゴリズムを以下のStep0からStep7に示す．

Step0:最初の頂点を通過頂点リストに記入する．
Step1:通過頂点リストが空であればENDへ．
Step2:現在の頂点の既選択ａｒｃリストに記入されていないａｒｃがあればそのａｒｃを選択し，既選択ａｒｃリストに記入する．無ければ，後進*しStep1へ．
Step3:選択したａｒｃに従って前進する．
Step4:現在の頂点が通過頂点リストに存在しなければ記入する．存在すれば記入せず後進し，Step1へ．
Step5:通過頂点リストの記入頂点数が全頂点数より少なければStep1へ．
Step6:現在の頂点から最初の頂点へのａｒｃがあればハミルトン閉路であるから出力する．
Step7:後進*しStep1へ．
End　:

注，後進*:後進するとき，当該既選択ａｒｃリストを消去し，通過頂点リストから当該頂点を消去することを表す．

このアルゴリズムの概略のイメージは，グラフの中で，任意の頂点から出発して，各頂点でａｒｃを順序よく，もれなく，選択，前進し，必要に応じて，（ハミルトン閉路でないことが明かになった場合，またハミルトン閉路を発見して次のハミルトン閉路の探索を開始する場合）後進して，ハミルトン閉路を全て探すというものであって，総当たり法とも言うべき方法である．Step0からStep7に示したようにアルゴリズムは比較的単純であり，この点は必ずしも十分ではない開発環境において，ＤＳＰのアセンブリ言語で記述する際には利点となる．しかしグラフが大規模になると処理時間が急激に増加する欠点がある．

3.2.2　マルチパスメソッド

マルチパスメソッドのアルゴリズムはエニュメレーションメソッドを基本とし，そのアルゴリズムの各段階でグラフの退化処理を行うことによって複数のパスが生成され，最終的にはそのパス同士が接続されハミルトン閉路となるものである．その名前はハミルトン閉路を探索する過程で，複数のパス（マルチパス）が生成されることに由来する．このアルゴリズムのキーポイントは統合可能な頂点は統合し，削除可能なａｒｃは削除することによってグラフを可能な限り退化することである．退化による頂点の統合は前進及び後進回数，またａｒｃの削除はａｒｃ選択回数の減少をもたらし，処理時間が大幅に短縮される．このような利点の反面，プログラミングはエニュメレーションメソッドに比較して複雑化する．頂点の統合及びａｒｃ削除の方法を次に述べる．

3.2.2.1　頂点の統合

図３に示すように，出力次数１である頂点Ｖ１の出力ａｒｃの行先の頂点Ｖ２の入力次数が１であるとき，Ｖ１とＶ２を統合することができる．

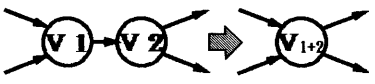

図３　頂点の統合

3.2.2.2　ａｒｃの削除

(1) 図4に示すように，ある頂点の出力ａｒｃを選択したとき，その頂点で選択されなかった他の出力ａｒｃを削除することができる．

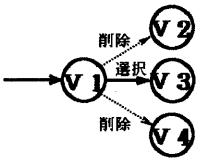

図4　非選択ａｒｃの削除

(2) 図5に示すように，生成されているパスがまだハミルトンパス（グラフの全ての頂点をちょうど1回通る道）になっていないとき，そのパスの最終頂点から開始頂点へのａｒｃがあれば削除することができる．

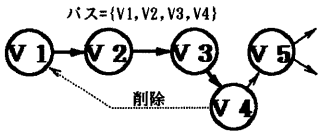

図5　開始頂点へのａｒｃ削除

(3) 図6に示すように，頂点の入力次数が1であれば，当該ａｒｃの元の頂点における他の出力ａｒｃは削除することができる．このような頂点は，グラフに初めから存在することも，またａｒｃ削除処理の結果生じることもある．

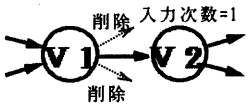

図6　出力ａｒｃ決定によるａｒｃ削除

(4) 図7に示すように，頂点の出力次数が1であれば，当該ａｒｃの行先の頂点における他の入力ａｒｃは削除することができる．このような頂点は，本報告において取り扱っているグラフについては出力ａｒｃ数＝3～5に設定しているため，初めから存在することはないが，ａｒｃ削除処理の結果生じることがある．

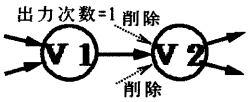

図7　入力ａｒｃ決定によるａｒｃ削除

3.2.2.3　ａｒｃ削除の連鎖

前項で示したａｒｃ削除の結果，新たに入力（出力）次数＝1の頂点が生じたときは再びそれぞれに応じてａｒｃ削除が可能である．これについては次の重要な事項がある．

頂点の入力（出力）次数が1であることに伴う，当該ａｒｃの元の頂点（行先の頂点）の他の出力（入力）ａｒｃ削除による連鎖反応によって生じる可能性があるのは，入力（出力）次数1の頂点であり．その逆，つまり出力（入力）次数1の頂点は生じない．すなわち頂点の入力（出力）次数＝1によるａｒｃ削除処理は連鎖反応によってそれ自体が連続する可能性があるということであり，これは処理の性能向上に有利である．

3.2.2.4　ａｒｃ削除の効果

ａｒｃ削除処理をアルゴリズムの各段階で行いつつ前進することによって，得られる効果は次のとおりであり図8に示す．

(1) パスの中間頂点からパス以外への出力ａｒｃ，またはパス以外から中間頂点への入力ａｒｃは存在しない．すなわち，通って来た道は1本のパスになっている．

(2) パスの開始頂点の出力ａｒｃでパスに接続されないａｒｃは削除されている．

(3) パスの最終頂点に終端しているａｒｃでパスに接続されないａｒｃは削除されている．

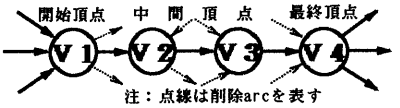

図8　ａｒｃ削除の効果

3.2.2.5　削除ａｒｃの復旧

ハミルトン閉路の探索過程において，削除されたａｒｃであっても，”後進”によって，削除されるに至った原因が解消されれば復旧しなければならない．このため，ａｒｃを削除するときは，削除されるに至った原因をそのａｒｃ毎に記録しておき，復旧するときは，それを参照して行なう必要がある．削除ａｒｃの復旧処理の方法は，ａｒｃ削除と同様に処理時間を左右する重要な事項であり，実際の処理においては，”前進”するとき，ａｒｃの削除処理が生じた場合は，削除するａｒｃの名前を専用のスタックにプッシュしておき，”後進”するときはポップして，そのａｒｃを復旧するという方法を用いている．

3.3　ホストでのグラフの処理方法

グラフのハミルトン閉路を並列処理システムを用いて求めるとき，各ＰＥで求めたハミルトン閉路の全ては，元のグラフに含まれていたハ

ミルトン閉路の全てと，集合として一致していなければならない．つまり並列処理するためグラフを分割するときにハミルトン閉路の重複及び抜けが無いことが必要である．これを満たすためホストでは次のようにグラフを分割処理する．まず，元のグラフの適当な頂点に注目して，そこからの出力ａｒｃを取捨選択することによって場合分けを行う．場合分けの数がＰＥ数に満たないときは，他の適当な頂点においても場合分けを行うことによって，ＰＥ数に一致させる．場合分けしなっかった頂点のａｒｃは当然そのままとする．このとき分割したグラフに対する処理の負荷がなるべく均等になるよう考慮することが必要である．グラフの分割処理方法について，具体例として２分割する場合を図９に示す．元のグラフの頂点Ｖ２からの出力ａｒｃを取捨選択し分割することによって２分割できる．このように分割することによって，元のグラフに存在するハミルトン閉路は漏れなく，かつ重複する事なく分割したグラフに移る．ただしこれによりハミルトン閉路が分割されたグラフ全てに分散されるとは限らない．

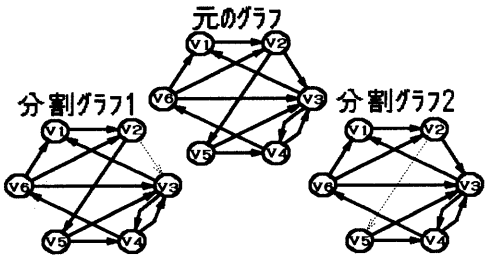

図９　グラフの分割処理

3.4　処理の概要

ハミルトン閉路求解処理の流れを図１０に示す．これはエニュメレーションメソッド及びマルチパスメソッドに共通のものである．

(1) ホストはグラフをＰＥ数に応じた数に分割処理し，分割されたグラフを各ＰＥにダウンロードする．

(2) 次にホストは全ＰＥに処理開始信号を送り，各ＰＥはアルゴリズムに従いハミルトン閉路を求めた後に処理終了信号をホストに返す．

(3) 各ＰＥを監視していたホストはＰＥからの処理終了信号を受けるとＰＥの処理結果をアップロードする．

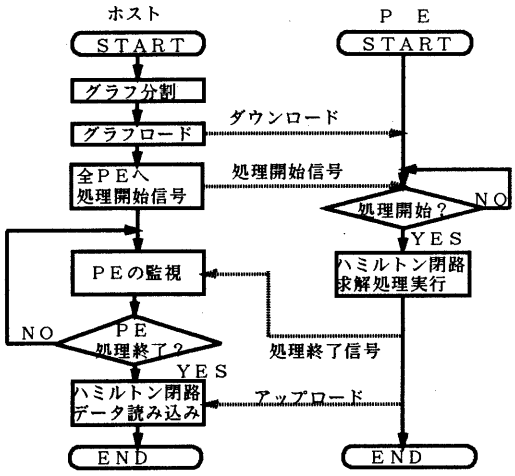

図１０　処理の流れ

4．　評　価

4.1　実装順序

本システムへの実装は，プログラミングの難易度から，エニュメレーションメソッド，マルチパスメソッドの順で行った．

4.2　グラフの作成

ハミルトン閉路を求める対象とするグラフを作成するため，事前にグラフにおけるハミルトン閉路数の一般的傾向を調べた．その結果によると頂点数を固定して１頂点あたりのａｒｃ数を増化させると，それに伴いハミルトン閉路数は急激に増加し，また１頂点あたりのａｒｃ数を固定して頂点数を増加させた場合も，同様である．ただし頂点数に比較して１頂点あたりのａｒｃ数が小さいと，ハミルトン閉路が存在しないグラフが多くなる．これらの事実を踏まえて，対象とするグラフは次のように作成する．

(1) 各頂点からの出力ａｒｃ数（出力次数）は３から５の範囲でランダムに変化させる．

(2) 出力ａｒｃの行先もまたランダムに変化させる．ただし自己ループは無いものとする．

(3) 出力ａｒｃ数及び行先をランダムに変化させると，ａｒｃの行先が特定の頂点に集中してしまうことがある．そこでグラフのデータ構造を簡単にするため，入力ａｒｃ数は１０を限度としてそれを越えるものは扱わない．

4.1　エニュメレーションメソッドによるハミルトン閉路求解

本システムと他システムの処理能力を比較するため，頂点数が１０，１５，２０の各々の場

合について，１０００個のグラフのハミルトン閉路をすべて求め，１閉路を求めるのに要した平均時間を図１１に示す．処理時間の大きいものから順にホストのみで処理した場合，参考文献[6]に記載されているＣＤＣ６６００，次に４ＰＥ実装時の本システムとなっている．

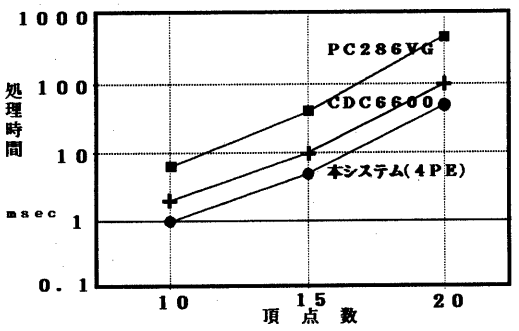

図１１　各システムの処理能力

次に本システムにおいて，ＰＥ数を変化させた場合の処理時間を図１２に示す．処理時間はＰＥ数が１の場合に正規化して表現している．ダウンロード時間はＰＥでの処理時間に比較して小さいためこの図には現れていない．ＰＥでの処理時間は，ＰＥ数の増加につれて負荷が分散されるため減少している．またアップロード時間は，3.3項で述べたようにハミルトン閉路数がグラフを分割しても変化しないため，ほぼ一定となっている．結局，４ＰＥの台数効果として２．７が得られている．

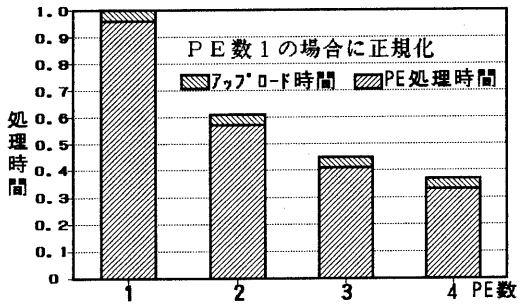

図１２　ＰＥ数と処理時間の関係

ここで台数効果を制約している要因を検討すると，まず負荷のアンバランスが考えられる．これは複数のＰＥにハミルトン閉路求解処理を行わせるためグラフを分割処理したときに，分割されたグラフの持つ負荷がアンバランスになり各ＰＥの処理終了時間に差が生じることである．この具体例を図１３に示す．元のグラフの持つ負荷を１とし，これを４分割するとき，図１３に示すようにアンバランスになったとすると，各ＰＥでの処理が終了するのは，４分割されたグラフの中で最大負荷のもので決まってしまう．理想的に均等分割された場合に比較して，処理時間が増加する割合を処理時間増加係数と呼ぶことにする．この例の場合では０．３５÷０．２５で１．４となる．

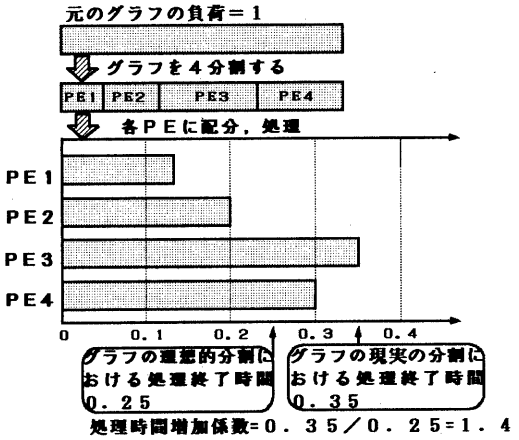

図１３　負荷のアンバランス

そこで２０００個のグラフについて3.3項に示した方法でグラフを４分割した場合の処理時間増加係数を調べると図１４のようになる．ｘ軸は処理時間増加係数でｙ軸はそれに対する度数である．下側９５％の平均を取ると１．３７になっており，先ほどの台数効果のデータから逆に推定した値は１．３８となっているので，４ＰＥにおける台数効果はほとんどこの処理時間増加係数によって決定されていると言える．

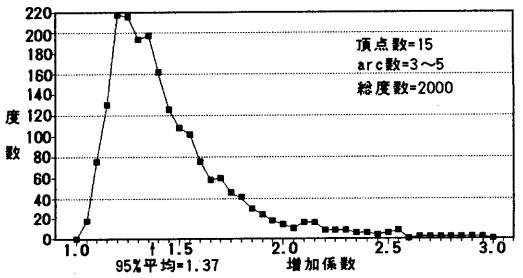

図１４　処理時間増加係数の分布

次にダウンロード時間は，ＰＥ数が少ないときは問題にならないがＰＥ数の増加に伴い無視できなくなる．アップロード時間に関しては，先に述べたように，ＰＥ数による変化はほとんど無い．

以上の検討結果をもとに，本システムのＰＥ数を増加させた場合の処理時間及び台数効果を、次の前提条件のもとで推定した．その結果を図１５に示す．

(1) ダウンロード時間

Tdwn=Tgrf(1+Kgrf×n)÷Ncir

Tgrf=Tw×Kact×(1+VER(1+ARC))

(2) ＰＥ処理時間

Tpe(n)=Tpe(1)×Kpe(n)÷n

(3) アップロード時間

Tup=Kact×Tw×(1+VER)

(4) 定数等

VER　:頂点数=15

ARC　:平均ARC数=4

Kgrf　:グラフ分割処理の処理時間係数=0.5

Ncir　:グラフの平均ハミルトン閉路数=480

Tw　:データ転送時間=0.022msec/word

Kact　:実処理時の増加係数=1.2

Tpe(1):１ＰＥ処理時間=9.492msec

Kpe(n):処理時間増加係数=$n^{0.2273}$

n　:ＰＥ数1,2,4,...

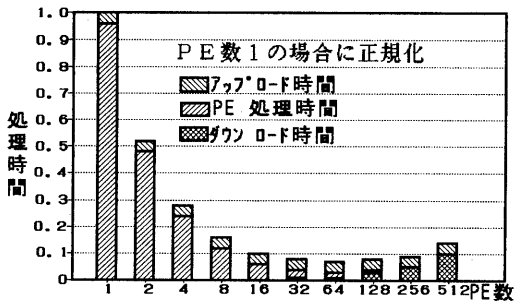

図１５　ＰＥ数と推定処理時間の関係

この結果から，本システムを６４ＰＥに拡張したとき，台数効果として最良の１６が得られ，また６４ＰＥ以上ではダウンロード時間が支配的になり台数効果が減少すると推定される．

4.2 マルチパスメソッドによる評価

4.2.1 マルチパスメソッド実装の動機

エニュメレーションメソッドを本システムに実装した場合には，上に記したような能力を有することが明らかになった．しかし対象とするグラフの頂点数の増加に伴い処理時間は急激に増加し，実験を行う場合にも２５頂点程度のグラフを取り扱うのが限度である．そこで処理能力の向上が期待できるマルチパスメソッドを本システムに実装し，より大規模なグラフのハミルトン閉路を求めることとした．

4.2.2 マルチパスメソッドとエニュメレーションメソッドの比較

それぞれの方法で，本システムの１個のＰＥを用いて，複数のグラフの全てのハミルトン閉路を求めた．その時の１ハミルトン閉路当たりの探索所用時間を図１６に示す．頂点数をＶとしたとき，所用時間はエニュメレーションメソッドは，

$t = 0.015 \times 10^{V/6}$[msec]　　（１）

マルチパスメソッドは

$t = 0.75 \times 10^{V/30}$[msec]　　（２）

で近似的に表現できる．ただしこれは4.2項で述べた条件で作成した頂点数が５０までの範囲のグラフでいえることであり，必ずしも一般的にいえることではない．

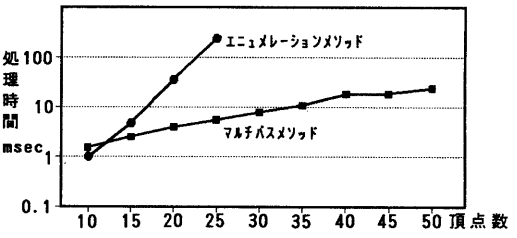

図１６　頂点数と処理時間の関係

4.2.3 マルチパスメソッドによる台数効果

マルチパスメソッドを複数のＰＥに実装したときのＰＥ数と処理時間の関係を図１７に示す．

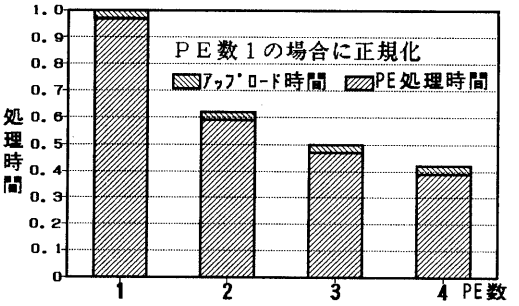

図１７　ＰＥ数と処理時間の関係

処理時間の割合については基本的にはエニュメレーションメソッドの場合と同じ傾向を示しており４ＰＥの台数効果は２．３となっている．台数効果がエニュメレーションメソッドの場合に比較して下がっているのは，エニュメレーションメソッドが総当たり的な探索を行うのに対して，マルチパスメソッドがグラフの退化処理を行っていることがグラフ分割における負荷の

不均等を大きくして，台数効果を下げていることが考えられる．

4.2.4　ＰＥ増設時の処理能力の推定

本システムのＰＥを増設した場合の処理能力を推定する．本システムにおける処理時間は

(1)　ダウンロード時間

(2)　ＰＥ処理時間

(3)　アップロード時間

に分かれている．この(1),(2),(3)について，ある程度の誤差は生じるがｎ個のＰＥの場合の処理時間を求めればよい．まず始めにグラフを3.3項で示した方法でｎ分割する．分割したグラフを順次，ある１個のＰＥにダウンロードし，処理させ，結果をアップロードし各々の時間を測定する．この数値からｎ個のＰＥで並列処理させた場合の時間を求めるには，ダウンロード及びアップロード時間については各々の和をとれば良い．またＰＥの処理時間は分割したグラフの内の最大負荷の処理時間で決まる．頂点数１５の１０００個のグラフについて，６４ＰＥまでを上に述べた方法で，また，それ以上のＰＥ数の場合についてはそれ以下のデータから推定して求めた結果を図１８に示す．

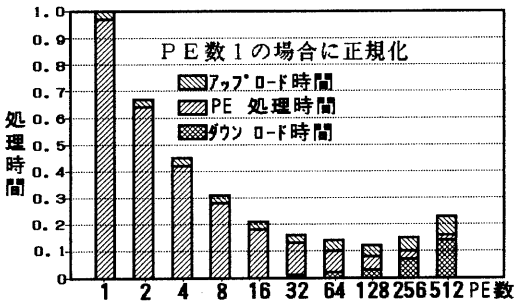

図１８　ＰＥ数と推定処理時間の関係

この結果は１２８ＰＥで最良となっているが，本システムのＰＥの最大実装数は６４であることから，６４ＰＥまで拡張して台数効果は７．１が得られることが明らかになった．

4.3　評価のまとめ

エニュメレーションメソッド及びマルチパスメソッドの台数効果は１５頂点のグラフについて求めたものである．ここで頂点数の異なる他のグラフについても台数効果は大きく変化しないと仮定して両方法の処理能力を求めたものを図１９に示す．これによるとエニュメレーションメソッドはマルチパスメソッドに台数効果及び小規模なグラフに対する処理能力は勝っているが，大規模なグラフに対してはマルチパスメソッドの方が優れていることがわかる．

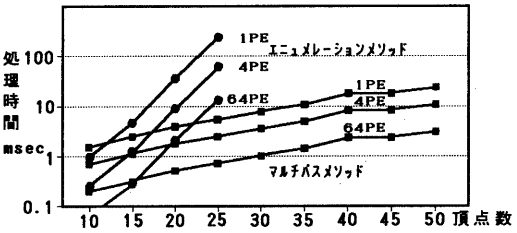

図１９　複数ＰＥでの処理能力

5．　おわりに

ハミルトン閉路を求めるため，当研究室で試作したバス結合型並列処理システムに，エニュメレーションメソッドとマルチパスメソッドを実装した．その各々について，台数効果は４ＰＥで２．７及び２．３が得られた．その制約要因はグラフ分割による負荷のアンバランスが最も多く，ついでダウンロード時間の増大である．さらに本システムを拡張した場合の台数効果は，６４ＰＥで各々１６と７．１が得られることが推定される．また頂点数の大きい，大規模なグラフを扱うときはマルチパスメソッドの方が高速であることが明らかになった．

今後は，マルチパスメソッドの改善と，本システムを用いて処理するときの並列処理能力の向上策の検討を行う予定である．

参考文献

[1] 吉川和宏等："マルチDSPシステムのアーキテクチャと並列信号処理開発環境"，情報処理学会第43回(平成３年)全国大会予稿集，7U-1(1991)．

[2] 高橋，黒川，古賀:" バス結合方式による並列処理システム"，信学技報CPSY93-38(1993)．

[3] 楢原，尾田，松原，黒川，古賀:" バス結合型ＤＳＰによる信号処理"，情報処理学会第48回(平成6年前期)全国大会，7B-6(1994)．

[4] NEC μPD77240ディジタル・シグナル・プロセッサ　ユーザーズ・マニュアル(1991)

[5] NEC RA77240アセンブラ・パッケージ　ユーザーズ・マニュアル(1991)

[6] N.Christofides:"Graph Teory-an Algorithmic Approach"，p.214，Academic Press (1975)．