2009－AL－122 (3)
2009／1／30

オイラー経路の一つを求める並列アルゴリズム

松本 吉實、 多田 昭雄
崇城大学 情報学部

概要

オイラー経路を求める逐次アルゴリズムとしてフラーリーのアルゴリズムがあるが、効率の良い並列アルゴリズムは見当たらない。本稿では、CREW-PRAM 計算機モデルのもとで、オイラーグラフにおいて一つのオイラー経路を求める並列アルゴリズムを提案する。具体的には、はじめに、各辺が出節点と入節点で与えられた無向辺に辺番号を付与し、これに出節点と入節点を入れ換えた逆向き辺を追加して、節点番号と辺番号で整列する。そして、各節点において先頭から順に、２辺を１組として各辺を接続すると、複数の部分ループが構成される。次に、ブロードキャストを利用してこの部分ループ内の最少辺番号を特定する。この最少番号を持つ辺の節点において２つの部分ループを接続しつつ、一つのオイラー経路を求める並列アルゴリズムである。提案する並列アルゴリズムの計算量は、節点数 n, 辺数 m とする時、プロセッサ 数 $O(m)$,時間量が $O(\log^2 m)$となる。

Parallel Algorithm for Finding an Eulerian Path

Yoshimi Matsumoto, Akio Tada
Faculty of Computer and Information Sciences, Sojo University

Abstract There is Fleury's algorithm as a sequential algorithm for finding an Eulerian path. However we cannot find a parallel algorithm for this problem. In this paper, we propose an efficient parallel algorithm for finding an Eulerian path in an undirected graph with n vertices and m edges on a CREW-PRAM model. Namely, the proposed algorithm initially gives an edge number for each edge which is composed of out-vertex number and in-vertex number, and appends reverse edges which changed out- and in-vertex number. Then these edges with vertex numbers and edge numbers are sorted. Next, after we connect a pair of two edges from the top for each vertex, then some partial loops are constructed. At last, these loops are joined into one Eulerian path. The proposed algorithm requires $O(m)$ processors and $O(\log^2 m)$ time.

1 はじめに

オイラーグラフは、一筆書きとなるオイラー経路を持つことが定理[1]として確立されている。オイラーグラフの条件は、次の２つのいずれかに該当する場合である。

(a) 全ての節点での次数が偶数の場合

(b) 奇数の次数の節点が２つ、他の節点は全て偶数の場合

オイラー経路を求める逐次アルゴリズムとしてフラーリーのアルゴリズム[1]があるが、効率の良い並列アルゴリズムは見当たらない。本稿では、オイラーグラフ（節点 n、辺数 m）において一つ

のオイラー経路を CREW-PRAM モデルのもとで求めるアルゴリズムを提案する。

具体的には、はじめに２節点で表す各辺に辺番号を付与し、接続作業用配列上で接続作業および接続補正作業を行う。接続作業は各節点内での辺の配列順番号が奇数番号と偶数番号の辺をペアとして接続する。この結果複数の部分ループが構成される。そして、奇数番号、偶数番号を利用して接続補正をすることにより一つのループ、つまりオイラー経路を求める並列アルゴリズムである。計算量は、プロセッサ数 $O(m)$、時間量 $O(\log^2 m)$ となる。

2　提案する並列アルゴリズムの概要

本稿において今回提案する並列アルゴリズムは、対象とするグラフが無向連結グラフで、条件(a) に該当するオイラーグラフで、かつ、自己ループおよび多重辺を含まないグラフとする。

対象とするオイラーグラフが、条件(b)に該当する場合は奇数節点間にダミー辺１個を追加する。これにより、全ての節点での次数が偶数となり、オイラーグラフの条件(a)に該当することになる。その結果、条件(a) (b)どちらに該当するオイラーグラフでも同一のアルゴリズムで対応できる。条件(b)の場合は経路完成後にダミー辺を取り除くと目的のオイラー経路が得られる。

以下に並列アルゴリズムの概要を示す。

２節点で表された各辺に辺番号を付与する。各節点では、辺番号順に整理する。この順番号が奇数番号か偶数番号かを利用して、以下の接続作業および接続補正作業を行う。

（１）奇数順番号の辺は、自身の奇数順番号 ＋１ 番の偶数順番号の辺と接続する。これを全辺同時に行う。この結果、複数の部分ループが構成される。

（２）ブロードキャスト[3]を利用してループ内の辺番号の最少番号を求め、この番号をループ番号としてループ内各辺が共有する。

（３）各ループとも自己のループ番号と同じ辺番号を持つ辺を特定する（以下、この辺を最少辺番号辺と呼ぶ)。最少辺番号辺を持つ節点において、他の番号のループと接続を補正する。接続補正を要求できるのは最少辺番号辺で１回のみとする。要求を受けるループ番号は複数回でも可とする。

（４）再度（２）を行う。

（５）全辺のループ番号が１か否かを判定する。否の場合は(３)（４）を繰り返す。

（６）全辺のループ番号が１ならばオイラー経路が完成する。ダミー辺が追加されてあればこれを削除する。

3　準備１：配列メモリの確保および入力グラフデータの整理

対象とするオイラーグラフに、あらかじめ各節点に節点番号をつける。その際なるべく次数の高い節点（次数は辺数を表す）から番号をつける。一つの辺は２つの節点番号の組(V1 、V2)で表わされる(以下　節点番号の小さい方 V1 を S サイト、大きい方 V2 を L サイトとする)。具体例として、図１に入力グラフの例を示し、表１にその入力配列の例を示す。

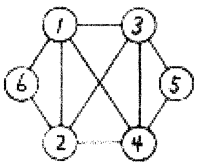

図１　入力グラフの例

番号	1	2	3	4	5	6	7	8	9	10
V1	1	1	1	1	2	2	2	3	3	4
V2	6	2	4	3	6	3	4	4	5	5

表１　入力データ

次に、以下の配列メモリを準備する。

(a)　辺データ整理用配列１（２次元配列、１次元は入力順番号）（表２、表３）

辺データ整理用配列２（３次元配列、１次元は節点番号、２次元は辺順番号）（表６）

(b)　接続作業用配列（３次元配列、１次元は節点番号、２次元は節点内辺順番号）（表７）接続および接続補正の作業は、全てこの配列上で行

う。

(c) 辺接続データ用配列（２次元配列、１次元は辺番号）（表９）辺の接続に関するデータを保持する。各辺が保持している、節点番号と接続相手の辺番号をポインタとして接続辺のループを取り出すことができる。また、接続辺のループはブロードキャストにも利用する。

(d) 節点番号探査用配列（２次元配列、１次元配列番号）（表４）

(e) 辺データ保持用メモリ（１次元配列 3 個）（自己用、Ｓサイト用、Ｌサイト用）（表５）

入力グラフのデータは、辺データ整理用配列１および辺データ整理用配列 2 で次のように整理をする。

（１） 入力データは、辺データ整理用配列１へのデータ入力順に辺番号を付与する（本稿では説明の便宜上、節点番号と区別するために辺番号を 100 番台の数値として表記する）。.

（２）各辺に辺プロセッサとして各１個を割り当てる。

（３）辺データ整理用配列１（表２）に節点番号（V1、V2）が逆向きの辺(表３)を追加し、節点番号順に整列[4]する。整理された辺データを辺データ整理用配列２に移し、同一節点内で辺番号順に整列する。最終的には辺データ整理用配列２では表６のように整理される。

（４）節点内での最大次数（節点に接する最大辺数）を求める。この最大辺数を、接続作業用配列の２次元数とする。

配列番号		1	2	3	4	5	6	7	8	9	10
1	節点 V1	1	1	1	1	2	2	2	3	3	4
2	辺番号	101	102	103	104	105	106	107	108	109	110
3	節点 V2	6	2	4	3	6	3	4	4	5	5

表２ 辺データ整理用配列１

配列番号		11	12	13	14	15	16	17	18	19	20
1	節点 V2	6	2	4	3	6	3	4	4	5	5
2	辺番号	101	102	103	104	105	106	107	108	109	110
3	節点 V1	1	1	1	1	2	2	2	3	3	4

表３ （追加）逆向き辺データ

配列番号	1	2	3	4	5
ループ番号					
節点番号					
接続相手番号					

表４ 節点番号探査用配列

自己 辺番号			自己 ループ番号		
S	1 相手辺番号		L	1 相手辺番号	
	2 相手順番号			2 相手順番号	
	3 節点番号			3 節点番号	
	4 順番号			4 順番号	
	5 辺番号			5 辺番号	
	6 ループ番号			6 ループ番号	
	7 補正辺			7 補正辺	
	8 完成チェック			8 完成チェック	

表５ 辺データ保持用メモリ

以下、各表における補足を記述する。

・表２と表３の数値は表１から得られた記入例である。表 6 から表 12 の数値は図２から得られた記入例である。

・V1 は節点番号の小さいサイトを表す

・V2 は節点番号の大きいサイトを表す

・S は V1 を表す

・L は V2 を表す

・順番号は各節点内の辺の順番号を表す（表６で割り付けられる順番号）

・ループ番号はループ内の最少辺番号で表す

準備２：辺プロセッサの作業内容

辺プロセッサは、辺データ整理用配列２からＳサイトおよびＬサイトの節点内の順番号を読み取る。この順番号が奇数であれば奇数辺プロセッサ、偶数であれば偶数辺プロセッサとして、それぞれ接続作業または接続補正作業を行う。１つの辺プロセッサは S、L サイトそれぞれで、順番号が奇数であれば奇数辺プロセッサ、偶数であれば偶数辺プロセッサとする。従って、一つのプロセッサが両サイト共に、奇数辺プロセッサ或いは偶数辺プロセッサの場合もある。辺プロセッサは、適宜、準備１の配列の中から、必要とするデータを読み取り、そのデータを自身が持つ辺データ保持用メモリに書き込む。また、適宜、辺データ保持用メモリに持つデータを、目的とする配列の該当メモリに書き込む。

4　提案する並列アルゴリズムの詳細化

図２のオイラーグラフ（節点数 24、辺数、45、奇数節点 2 他は全て偶数のオイラーグラフ、節点 3 と節点 20 の奇数節点間にダミー辺を追加）をモデルとしてアルゴリズムを説明する。

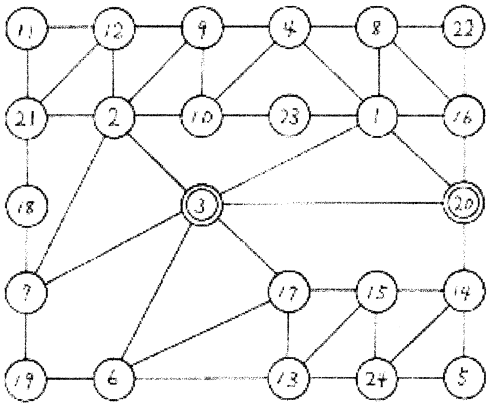

図２　モデルとするオイラーグラフ

（１）入力グラフのデータを３項の準備１に従って辺データ整理用配列１および辺データ整理用配列２上で整理する。次に、準備２により各辺プロセッサは必要なデータを読み取り、辺プロセッサの S サイト、L サイトの役割を確定する。さらに必要なデータを接続作業用配列の割り振られた該当メモリに書き込む（以下、配列内に割り当てられた番号を(　)で示す）（表７）。

（２）接続作業用配列（表７）上で、奇数辺プロセッサは同一節点内で自身の順番号 ＋1 番の配列メモリ(4)(5)から、その辺番号と順番号を自身の辺データ保持用メモリ(1)(2)に読む込み、これを接続作業用配列の自身の順番号の該当メモリ(1)(2)に書き込む。偶数辺プロセッサは自身の順番号 － 1 番の配列メモリから同様の作業をする。この結果を各辺プロセッサは読み取り、さらに、辺接続データ用配列の該当メモリに書き込む（表９）。この接続作業により複数の部分ループができる（図３、表８）。

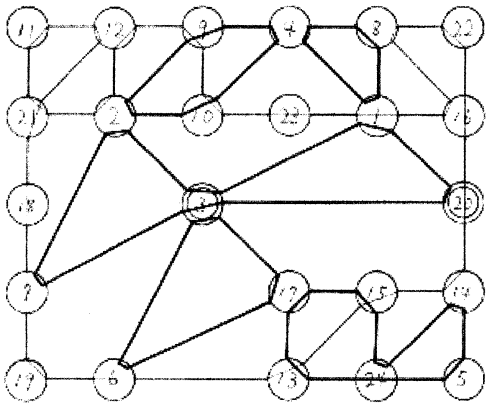

図３　部分ループ図

節点番号	1						2					
順番号	1	2	3	4	5	6	1	2	3	4	5	6
節点番号	1	1	1	1	1	1	2	2	2	2	2	2
辺番号	101	102	103	104	105	106	107	108	109	110	111	112
節点番号	8	4	23	16	20	3	3	7	21	12	9	10

21				22		23	
1	2	3	4	1	2	1	2
21	21	21	21	22	22	23	23
109	133	134	145	127	143	103	131
2	11	12	18	8	16	1	10

表６　辺データ整理用配列２　（奇数偶数の順番号、一部略記）

節点番号		1						2					
順番号		1	2	3	4	5	6	1	2	3	4	5	6
1	相手辺番号	102	101	104	103	106	105	108	107	110	109	112	111
2	相手順番号	2	1	4	3	6	5	2	1	4	3	6	5
3	節点番号	1	1	1	1	1	1	2	2	2	2	2	2
4	順番号	1	2	3	4	5	6	1	2	3	4	5	6
5	辺番号	101	102	103	104	105	106	107	108	109	110	111	112
6	ループ番号												
7	補正辺												
8	完成チェック												

22		23		24			
1	2	1	2	1	2	3	4
143	127	131	103	137	121	142	138
2	1	2	1	2	1	4	3
22	22	23	23	24	24	24	24
1	2	1	2	1	2	3	4
127	143	103	131	121	137	138	142

表７ 接続作業用配列（全辺同時接続、一部略記）

節点番号		4				5		6				7				8			
順番号		1	2	3	4	1	2	1	2	3	4	1	2	3	4	1	2	3	4
1	相手辺番号	117	102	119	118	121	120	122	115	124	123	113	108	126	125	117	101	128	127
2	相手順番号	2	1	4	3	2	1	2	1	4	3	2	1	4	3	2	1	4	3
3	節点番号																		
4	順番号	1	2	3	4	1	2	1	2	3	4	1	2	3	4	1	2	3	4
5	辺番号	102	117	118	119	120	121	115	122	123	124	108	113	125	126	101	117	127	128
6	ループ番号	1	1	11	11	20	20	15	15	3	3	5	5	3	3	1	1	3	3
7	補正辺																		
8	完成チェック																		

表８　取得ループ番号（接続作業用配列、一部略記）

（３）辺接続データ用配列（表９）のメモリ(5)（辺番号）上でブロードキャストにより各辺の辺番号をループ内の最少辺番号に収束させ、これをループ番号とする。これにより部分ループが検出される。具体的なアルゴリズムは、次項５の部分ループ検出アルゴリズムに記す。

（４）ループ番号と同一の番号を持つ辺プロセッサは、接続作業用配列上の自身の節点内で、２つの部分ループを一つにする接続補正相手を特定し、接続補正作業を行う（作業は全節点で並列処理する）。接続補正作業アルゴリズムは次項６で具体的に記す。

（５）　再度（３）の作業を行う。

（６）　各辺プロセッサは接続作業配列上で接続作業用配列メモリ(6)のループ番号が 1 ならば 0 を、1 でなければ 1 、を接続作業用配列上の、メモリ(8)（完成チェック）に書き込む。

（７）　メモリ(8)（完成チェック）の数値の合計が 0 ならばオイラー経路が完成、そうでなければ、（３）から（７）を繰り返す。

（８）　オイラー経路の完成が確認できたら、奇数節点 2 のオイラーグラフであれば、ダミー辺を削除する。

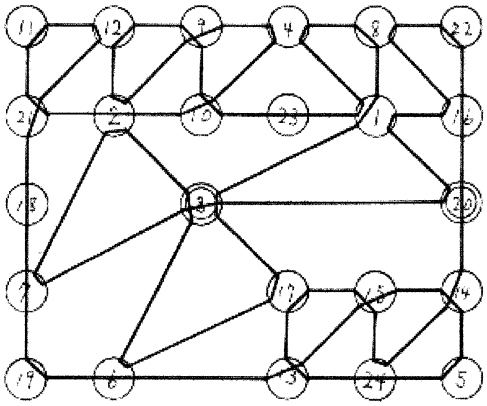

図４　完成オイラー経路

配列番号				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	S	1	相手辺番号	106	103	102	105	104	101	108	107	112	111	110	109	114	113	107	106	102	119	118	121	120
2		2	相手順番号	6	3	2	5	4	1	2	1	6	5	4	3	4	3	2	1	1	4	3	2	1
3		3	節点番号	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4	4	4	5	5
4		4	順番号	1	2	3	4	5	6	1	2	3	4	5	6	3	4	5	6	2	3	4	1	2
5		5	辺番号	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121
6	L	4	順番号	1	1	1	1	1	1	2	1	1	1	1	1	2	2	1	1	2	2	2	1	1
7		3	節点番号	8	4	23	16	20	3	3	7	21	12	9	10	7	20	6	17	8	9	10	14	24
8		2	相手順番号	2	2	2	2	2	6	5	2	2	2	2	2	1	1	2	2	1	1	1	4	2
9		1	相手辺番号	117	117	131	128	114	116	115	113	133	130	118	119	108	105	122	122	101	111	112	140	137

表９　辺接続データ用配列（接続補正完成、１回目の全辺同時接続分は省略）の１

配列番号			22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
1		1	115	124	123	126	125	128	127	130	129	129	133	132	132	123	137	136	139	138	120	142	141	144	143	125
2	S	2	1	4	3	4	3	4	3	4	3	3	2	1	3	1	4	3	3	2	1	4	3	4	3	1
3		3	6	6	6	7	7	8	8	9	9	10	11	11	12	13	13	13	14	14	14	15	15	16	16	18
4		4	2	3	4	3	4	3	4	3	4	4	1	2	4	2	3	4	2	3	4	3	4	3	4	2
5		5	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145
6		4	2	1	1	1	2	1	2	3	2	2	3	2	3	1	3	2	3	3	2	4	4	2	4	4
7	L	3	17	13	19	18	19	22	16	10	12	23	12	21	21	15	17	24	24	20	15	17	24	22	20	21
8		2	1	2	2	2	1	2	1	4	1	1	4	1	4	2	4	1	4	4	1	3	3	1	3	3
9		1	116	135	126	145	124	143	104	131	110	103	134	109	145	140	141	121	142	144	135	136	138	127	139	134

表 9　辺接続データ用配列（接続補正完成，１回目の全辺同時接続分は省略）の２

5　部分ループの検出アルゴリズム

部分ループを次により検出する。

辺接続データ用配列上での各辺は、接続節点番号および接続相手の辺番号を保持しているので、これをポインタとしてループ内の辺番号を接続順に取り出すことができる。辺接続データ用配列よりループ番号１（１回目接続）のループを取り出すと図５(a)のようになる。同様にして全ループを取り出すことができる。（表９の辺接続データ用配列は完成データ、一回目の接続データは省略する）

そこで、この配列上のポインタ[5]を利用して、辺接続データ用配列のメモリ(5)上で、各辺は自身の辺番号をブロードキャストにより送り出す。この時、ループ内の最少番号に収束させると、各部分ループ毎の最少番号が表れる。この番号を各辺プロセッサは読み取り、これを接続作業用配列のメモリ(6) に書き込む。これにより全辺同時接続により構成された各部分ループが把握できる。その後再度、各辺プロセッサは辺接続データ用配列のメモリ(5) に配列番号（自己の辺番号）を書き込む。

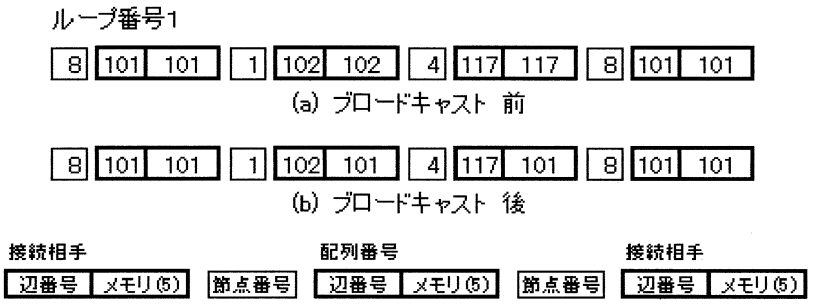

図５　部分ループの辺連結およびブロードキャスト結果のループ番号

6　接続補正作業のアルゴリズム

接続補正作業の詳細アルゴリズムを以下に記す。

（１）　接続補正要求をする辺プロセッサの決定

収束された番号と同じ番号を持つ辺プロセッサのＳサイトが、（２）以下の接続補正の作業の準備をする。接続作業用配列の自身のメモリ(7)(補正辺) に記号 *** を書き込む（表 10)。

同一節点内に接続補正を要求するルート番号の辺が複数ある場合、次の（２）の作業はループ番号の最も大きい辺プロセッサが行う。接続補正相手は自己のループ番号より小さいループ番号とする。ただし（２）(d)の場合を除く。ループ番号１をもつ辺番号１の辺プロセッサは接続補正の要求をしない。

（２）接続補正相手辺の決定

接続補正を要求できるのは、各辺１回のみとする。接続の要求を受ける辺は同一ルート番号のどの節点でも、また複数回でも可とし、次により決める。

(a)　同一節点内に接続補正を受けることができるルート番号が複数ある場合は、最も小さい番号を選択する。

(b) 同一節点内に接続補正を要求される同じルート番号が複数ある場合は、最も小さい番号辺を選択する。

(c) 同一節点内に接続補正を要求するルート番号が複数ある場合でも、接続を受けるルート番号は１つのみとする。

(d) 接続補正相手辺が自身の節点内にない場合は、まず同一辺プロセッサの反対Ｌサイトを探す。 見つからない場合は、接続補正を要求する自身のループ内で、他のループ番号を持つ節点の辺プロセッサが、その節点番号と他のループ番号を節点番号探査用配列に書き出す（必ず、１個以上は存在する)。この中からルート番号の最も小さい節点を選択する。

(e) 接続補正を受ける辺のメモリ(7)（補正辺）に、記号 * を書き込む（表 10)。

（３） 接続補正方法

（２）により接続補正相手が決定したら、各辺プロセッサは次により接続補正作業を行う。

(a) 各節点内で接続補正が定まった辺では現在の接続を解除する。これは、メモリ(7)（補正辺）に * または *** を持つ辺およびその接続相手辺でのメモリ[1][2] に"null"を書き込む。解除が確定した辺全てのメモリ[7]に記号 *** を書き込む（表 11)。

(b) 記号 *** を持つ偶数辺プロセッサは自身の順番号 +(1 + r * 2) の奇数番号辺で記号 *** を持つ奇数番号辺を探し、この辺と接続する。同一節点内に奇数番号辺が見つからない場合は r = 0 にリセットして（１＋ｒ＊２）の番号の奇数番号辺で記号 *** を持つ奇数番号辺を探す。必ず奇数番号辺がある。なお、r の値は 0 から（偶数番号辺自身の番号 +1 + r * 2）の値がその節点番号の最大値を超えない範囲とする。 従って最大でも（最大順番号 / 2）となる。これを超えた場合に r = 0 にリセットし（１＋ｒ＊２）の奇数番号辺を探す。

(c) 記号 *** を持つ偶数辺プロセッサは（２）の作業を同時並列に行う。これでペアとなった奇数辺プロセッサおよび偶数辺プロセッサは相互に相手辺の辺番号および順番号を自身のメモリ(1)(2)に書き込む。さらに、各辺プロセッサとも辺接続データ用配列の該当メモリに書き込む（表９)。

節点番号	1						2						3						14			
順番号	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4
1 相手辺番号	102	101	104	103	106	105	108	107	110	109	112	111	107	106	114	113	116	115	138	120	140	139
2 相手順番号	2	1	4	3	6	5	2	1	4	3	6	5	2	1	4	3	6	5	2	1	4	3
3 節点番号	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	14	14	14	14
4 順番号	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4
5 辺番号	101	102	103	104	105	106	107	108	109	110	111	112	106	107	113	114	115	116	120	138	139	140
6 ループ番号	1	1	3	3	5	5	5	5	3	3	11	11	5	5	5	5	15	15	20	20	3	3
7 補正辺	*		***		***				*		***		*				***		***		*	
8 完成チェック																						

表 10　接続補正要求辺および補正相手辺

節点番号	1						2						3						14			
順番号	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4
1 相手辺番号							108	107							114	113						
2 相手順番号							2	1							4	3						
3 節点番号	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	14	14	14	14
4 順番号	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4
5 辺番号	101	102	103	104	105	106	107	108	109	110	111	112	106	107	113	114	115	116	120	138	139	140
6 ループ番号	1	1	3	3	5	5	5	5	3	3	11	11	5	5	5	5	15	15	20	20	3	3
7 補正辺	***	***	***	***	***	***			***	***	***	***	***	***			***	***	***	***	***	***
8 完成チェック																						

表 11　接続解除辺

	節点番号	1						2						3						14			
	順番号	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4
1	相手辺番号	106	103	102	105	104	101	108	107	112	111	110	109	116	115	114	113	107	106	140	139	138	120
2	相手順番号	6	3	2	5	4	1	2	1	6	5	4	3	6	5	4	3	2	1	4	3	2	1
3	節点番号	1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	14	14	14	14
4	順番号	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4
5	辺番号	101	102	103	104	105	106	107	108	109	110	111	112	106	107	113	114	115	116	120	138	139	140
6	ループ番号	1	1	3	3	5	5	5	5	3	3	11	11	5	5	5	5	15	15	20	20	3	3
7	補正辺	***	***	***	***	***	***			***	***	***	***	***	***			***	***	***	***	***	***
8	完成チェック																						

表 12　接続補正済辺

まとめと今後の課題

今回提案したアルゴリズムのポイントは接続補正部分のアルゴリズムである。節点内での接続補正アルゴリズムは比較的に単純化できた。今後の課題としたいのは、次の点が挙げられる。１回目の全辺並列接続で構成される部分ループの個数がｋ個とすると接続補正は(ｋ－1)個で完了する。しかも接続補正は並列処理ができる。しかし、このアルゴリズムの時間量は $O(\log^2 m)$とした。その理由は次の点にある。

２個の部分ループの接続補正は１つの節点のみで完了する。これを複数の節点で行うと１個のループとなる保障はない。１個のループで１つの節点のみの接続補正を実行するために、ループ番号と同じ辺番号を持つ最少辺番号辺のみに接続補正の要求が出来るとした。そこで必然的に節点が特定される。自己のループ番号より小さいループ番号との接続補正とし、これがないときは接続補正はしないこととした。これで、オイラー経路の完成が最初の１回目接続補正のみでは達成できない場合がある。

以上の理由で、時間量は $O(\log^2 m)$とした。今後の研究課題としたい。また次の点も今後の課題としたい。

最少辺番号辺の節点内に他のループ番号がない場合、ループ内で、接続補正が可能な節点番号を探すアルゴリズムとして CRCW-PRAM での arbitrary 解消法はアルゴリズムを単純化できる。しかしこのことのみで CRCW-PRAM モデルを採用するのは、このモデルの採用だけで時間量が$(\log m)$を乗算することになることに抵抗を感じたので採用しなかった。CREW-PRAM モデルでのより効率的なアルゴリズムを考究したい。

参考文献

[1] 一森哲男：グラフ理論、pp.64-70,共立出版株式会社(2002).

[2] 宮野悟：並列アルゴリズム、近代科学社

[3] Gibbons, A. and Rytter, W.:Efficent Parallel Algorithms, Cambridge University Press (1988).

[4] Cole, R.: Parallel Merge Sort, SIAM J. Comput., Vol.17, No.4, pp.770-785(1988).

[5] 奥村晴彦：アルゴリズム事典、pp.299-301 技術評論社