Discrete Applied Mathematics 27 (1990) 267–276
North-Holland

EFFICIENT ALGORITHMS FOR PATH PARTITIONS

Craig WILLIAMS and Dana RICHARDS

Department of Computer Science, University of Virginia, Thornton Hall, Charlottesville, VA 22903, USA

Received 15 March 1988
Revised 24 January 1989

An $[a, b]$ path partition is a decomposition of the edges of a graph into a independent path sets and b matchings, where an independent path set is a set of paths that do not intersect each other. The path partition problem is related to other edge partition problems, such as edge coloring. It is shown that every graph with maximum degree 3 has a $[2, 0]$ path partition, and every graph with maximum degree 4 has a $[2, 1]$ path partition. Algorithmic proofs are given. The algorithms have linear-time sequential implementations and log-time parallel implementations.

1. Introduction

We present a new property of graphs, the path partition, and present algorithmic proofs for some cases of small maximum degree. Path partitions are related to edge-coloring and other edge partition problems. As an example of a path partition problem we ask whether the edges of a cubic graph can be colored blue and red so that there are only red paths and blue paths and no blue (red) path intersects with another blue (red) path? Alternatively, can we decompose a cubic graph into a set of disjoint paths so that the paths can be partitioned into a red set and a blue set, where no paths of the same color intersect? We show that such a partition exists by giving an algorithm that always finds such a partition. We present efficient sequential and parallel algorithms. Similar results are given for regular graphs of degree 4, and a generalization for larger degrees is proposed.

Let an *independent path set* be a set of vertex disjoint paths. A matching is a special independent path set in which the length of each path is 1. An $[a, b]$ *path partition* is a partition of the edges of a graph into a independent path sets and b matchings. We prove that any simple graph G with $\Delta(G) = 3$ has a $[2, 0]$ path partition and that any simple graph with $\Delta(G) = 4$ has a $[2, 1]$ path partition, where $\Delta(G)$ is the maximum degree in the graph G. In a sense these results are best possible. There are graphs with $\Delta(G) = 3$ which do not have a $[1, 1]$ path partition and there are graphs with $\Delta(G) = 4$ which do not have a $[2, 0]$ path partition. We restrict our attention in this paper to simple graphs. Unless otherwise specified, *graph* refers to a simple graph.

There is no known previous work on this problem. However our problem is

related to a number of other edge partition problems. Other edge partition problems include finding the path number of a graph, determining whether a graph has a [1,2]-factor, and coloring the edges of a graph. The *path number* of a graph G is the minimum number of edge disjoint paths whose union is G. The results on path numbers provide very loose upper bounds on the minimum number of independent paths into which the edges of the graph can be partitioned. For example, it is known that the edges of any cubic graph with n vertices can be covered by $\frac{1}{2}n$ edge-disjoint paths [8].

A $[1,2]$-*factor* of a graph is a spanning subgraph where each component is a path or cycle. By removing an edge from every cycle of a [1,2]-factor a *corresponding* independent path set is produced that spans the graph. In general an $[r, r+1]$-*graph* is a graph in which each vertex has degree r or $r+1$, and a $[k, k+1]$-*factor* is a spanning $[k, k+1]$-subgraph. Tutte proved that every r-regular general graph has a $[k, k+1]$-factor where k is an integer such that $0 \leq k \leq r$ [10]. Thomassen extended this result to show that if G is a general $[r, r+1]$-graph, G has a $[k, k+1]$-factor where k is an integer such that $1 \leq k \leq r$ [9].

Tutte's theorem implies that cubic graphs have a [2, 1] path partition. Removing the independent path set corresponding to a [1,2]-factor leaves a graph with maximum degree 2 which may contain cycles. Removing one edge from each cycle partitions this graph into a matching and a second set of independent paths. Similarly, using both Tutte and Thomassen's theorems, we can show that a 4-regular graph has a [3, 1] path partition. Removing the independent path set corresponding to a [1,2]-factor leaves a [2,3]-graph. Removing a second independent path set leaves a [1,2]-graph which can be partitioned, as above, into a matching and a third set of independent paths.

Results on edge-coloring of graphs provide similar results. An *edge-coloring* of a graph is an assignment of a color to each edge so that at every vertex no pair of incident edges is colored the same. The *chromatic index*, denoted $\chi'(G)$, is the minimum number of colors needed to edge-color G. It is easy to see that $\chi'(G) \geq \Delta(G)$. Vizing proved that for every graph, $\chi'(G) \leq \Delta(G) + 1$ [11]. Determining whether $\chi'(G)$ is $\Delta(G)$ or $\Delta(G)+1$ is NP-complete [5], but there are polynomial-time algorithms for finding a $\Delta(G)+1$ edge-coloring, e.g., [1].

The fact that the edge set of every graph with $\Delta(G) = 3$ is 4-colorable implies that each such graph has a [2, 1] path partition. Recolor the edges originally colored 1 and 2 red and recolor the edges originally colored 3 and 4 blue. Let B be a set of edges formed by removing an edge from each red or blue cycle. Since no two cycles intersect in a cubic graph, the edges removed to eliminate cycles form a matching. Similarly, a 5-coloring of the edges of a graph with $\Delta(G) = 4$ gives rise to a [3, 1] path partition. Begin as above. Note that the edges colored 5 combined with the set B form a set of paths and cycles, since no three can meet at one vertex. These edges can be partitioned into a third independent path set and a matching.

Note that our results give a stronger characterization than Vizing's theorem for graphs of maximum degree 3 and 4. A [2,0] path partition of a graph implies the

graph is 4-edge-colorable and a [2, 1] path partition implies the graph is 5-edge-colorable but the converse is not true. For example, a 4-edge-colorable 4-regular graph cannot have a [2, 0] path partition. (In fact no 4-regular graph can, since each vertex must be an internal point on two paths.) Further, there are 5-edge-colorable 5-regular graphs that do not have a [2, 1] path partition. A [2, 1] path partition of a 5-regular graph implies that the matching must be a perfect matching, but removing a perfect matching reduces the problem to finding a [2, 0] path partition of a 4-regular graph.

The proofs are constructive; we exhibit algorithms for finding the path partitions. Let the graph G have n vertices and m edges. These algorithms can be executed in $O(n+m)$ time on a RAM and in $O(\log n)$ time using $O(n+m)$ processors on a concurrent-read concurrent-write shared memory model of computation, i.e., a CRCW PRAM. The partitions can then be used to find a $\Delta(G)+1$ edge-coloring within the same time bounds. Karloff and Shmoys have shown that edge-coloring a graph with $\Delta(G)+1$ colors is in NC when $\Delta(G)=O(\log^{O(1)} n)$ [6]. They give a $O(\Delta(G)^{O(1)}\log^{O(1)} n)$ time algorithm using $O(n^{O(1)})$ processors on a CRCW PRAM and an $O(\log n)$ algorithm using $O(n+m)$ processors for coloring multigraphs of maximum degree 3. Since the constant exponents are greater than 1, the algorithm we describe is faster for the special case where $\Delta(G)$ is 4.

2. Constructing the path partition for $\Delta(G)=3$

Let $G=(V,E)$ be a cubic graph. Algorithm 1 partitions G into two independent path sets, a set of red paths and a set of blue paths.

Algorithm 1: Finding a [2, 0] path partition for a 3-regular graph.

Step 1. Obtain an initial coloring of G such that for all $v \in V$, v is incident to either two red and one blue or one red and two blue edges.

1.1 Construct G' from G by adding a special vertex v' and an edge from v' to every vertex in G.

1.2 Find an Eulerian circuit in G'. (Note that G' is Eulerian.)

1.3 Alternately color the edges of the Euler circuit red and blue. Delete v' and the edges incident to v' to obtain the original graph G. (The edges of G are partitioned into red paths and cycles and blue paths and cycles.)

Step 2. Eliminate all original cycles, i.e., all cycles created in the initial coloring.

2.1 Identify all cycles.

2.2 Choose one edge on every red (blue) cycle and flip it, i.e., color it blue (red).

Step 3. Eliminate all new cycles, i.e., cycles created by the preceding step.

3.1 Identify all cycles.

3.2 For each red cycle, choose one edge that was originally in a blue cycle and "shift" the red edge along that blue cycle, i.e., recolor the edge blue and color an adjacent blue edge red. Perform the analogous step for blue cycles.

Theorem 2.1. *Any cubic graph has a* [2,0] *path partition.*

Proof. Let G_{red} (G_{blue}) be the subgraph of G induced by the red (blue) edges of G. We show that Algorithm 1 constructs a partition of G into two independent path sets by showing that after step 3, G_{red} and G_{blue} have the following properties:

(1) $\Delta(G_{red}) = \Delta(G_{blue}) = 2$.
(2) G_{red} and G_{blue} are acyclic.

Since every edge is colored either red or blue, the edges of G_{red} and G_{blue} partition E into two independent path sets.

Property 1 is clearly true after Step 1. In the remaining steps, only edges on the original cycles can change color. Every edge on an original red (blue) path remains red (blue). In Step 2, each original cycle chooses one edge to flip. In Step 3 some subset of the original cycles restore the original color to their flipped edge and flip an adjacent edge along the cycle, in effect changing their initial choice as to which edge to flip. Since at most one edge along any original red (blue) cycle is colored blue (red) and all the blue (red) edges incident to the cycle are on original paths and therefore remain blue (red), no vertex has degree 3 in G_{red} (G_{blue}).

We show G_{red} is acyclic at the end of Step 3; G_{blue} is handled similarly. Since the shift operation eliminates every red cycle existing at the end of Step 2 we show that the shift operation cannot introduce red cycles. First consider a shift along a blue cycle. Let $\{v_i, v_j\}$ be an edge on an original blue cycle colored red in Step 2 and let $\{v_j, v_k\}$ be an adjacent edge colored red in Step 3 as a result of the shift operation. At the beginning of Step 3 $\{v_i, v_j\}$ is on a new red cycle and v_k is the endpoint of an original red path. Note that since G is a simple graph $v_k \neq v_i$. Coloring $\{v_j, v_k\}$ red in the shift operation can create a cycle only if a red path connects v_j and v_k that avoids edge $\{v_j, v_k\}$. However v_j is still connected by a red path to v_i and v_i now has degree 1 in G_{red}. Now consider a shift along a red cycle. The new red edge will have an endpoint with degree 1 in G_{red} and hence cannot contribute to a red cycle. □

Corollary 2.2. *Any graph with* $\Delta(G) = 3$ *has a* [2,0] *path partition.*

Proof. Every graph G with $\Delta(G) = 3$ can be converted into a cubic graph by the addition of a linear number of edges and vertices. Any [2,0] partition of the resulting graph H yields a [2,0] partition of G. □

The algorithm runs in linear time on a uniprocessor, i.e., a RAM. Steps 2.2 and 3.2 each take constant time per cycle and the remaining steps take time linear in the number of edges. The algorithm can be directly cast as a parallel algorithm in the shared memory model of parallel computation; in particular we use CRCW PRAM model. By using a known parallel algorithm to find the Euler circuit and standard techniques based on recursive doubling to color the Euler circuit and identify cycles, the algorithm can be executed in parallel on a CRCW PRAM with $O(n+m)$ processors in $O(\log n)$ time. (For a discussion of recursive doubling techniques see [7].) Atallah and Vishkin, and Awerbuch, Israeli, and Shiloach have described parallel algorithms for finding an Euler circuit in $O(\log n)$ time on a CRCW PRAM [2, 3].

Karloff and Shmoys's algorithm for edge-coloring multigraphs of maximum degree 3 also obtains the initial coloring using the technique of alternately coloring the edges of an Euler circuit, but breaks only odd cycles, leaving even cycles intact, and thus does not induce a partition of the edges into two independent path sets. (Indeed, cubic multigraphs cannot, in general, be partitioned into two independent path sets.)

3. Constructing the path partition for $\Delta(G)=4$

Let $G=(V,E)$ be a 4-regular graph. Algorithm 2 partitions G into two independent path sets, one red and one blue, and a matching consisting of edges colored green. Note that a [2, 1] path partition is the best possible. A [2, 0] path partition of G would imply that every vertex is incident to two edges from each set. If no vertex is the endpoint of a path, each set is a collection of cycles, not paths.

Algorithm 2: Finding a [2, 1] path partition for 4-regular graphs.

Step 1. Obtain an initial coloring of G such that for all $v \in V$, v is incident to two red and two blue edges.

1.1 Find an Eulerian circuit in G.

1.2 Alternately color the edges of the Euler circuit red and blue. (The edges of G are partitioned into red cycles and blue cycles.)

Step 2. Mark one edge on every cycle as a special edge.

2.1 Identify all cycles.

2.2 Choose one edge on every red cycle and mark it as the cycle's *special* edge.

2.3 Choose one edge on every blue cycle and mark it as the cycle's *special* edge. Choose an edge adjacent to as many red special edges as possible.

Step 3. Eliminate all special cycles and recolor paths of special edges.

3.1 For each cycle of special edges select two adjacent edges, a blue $\{x, v\}$ and a red $\{v, u\}$. Let $\{v, w\}$ be the other blue edge incident on v. If w is incident to a special edge, then remove the special mark from

$\{x, v\}$ and mark $\{v, w\}$ special. (This has the effect of breaking that special cycle and splicing it onto another path.) Otherwise flip the color of every edge on the cycle and remove the special mark from every edge on the special cycle except for $\{x, v\}$ and $\{v, u\}$. (This step combines all the red and blue cycles that contribute an edge to the special cycle into just a red and a blue cycle, each with a single special edge adjacent to the special edge of the other cycle.)

3.2 Color every special edge that is incident to the endpoint of a path of special edges green and flip the color of all the other special edges. (This step breaks each of the red and blue cycles with a special edge on the path; either with a green edge or by splicing the cycles to form one long red and one long blue path.)

Step 4. Eliminate all green paths of length 2.

4.1 For each green path of length 2 color blue the edge that was red at the beginning of step 3.2.

4.2 For each *initial* blue cycle, choose one endpoint of the special edge as the *special* vertex, where an *initial* blue (red) cycle is a blue (red) cycle existing at the beginning of step 3.2. If possible, choose the special vertex so that of the two incident edges that are not on the initial blue cycle one is colored red and the other is colored blue.

4.3 For every initial blue cycle, if the special edge was the middle edge on a path of special edges of length 3 (i.e., if the special edge is now red and both vertices incident to the special edge are incident to two red edges, one green and one blue edge) and there is a vertex on the cycle incident to three blue edges, recolor the special edge blue.

4.4 For every initial blue cycle, traverse the cycle visiting every vertex in such a way that the special vertex is visited first and last and the special edge is traversed last. Whenever a vertex incident to three blue edges is visited, recolor the edge just traversed green if the last vertex visited is incident to two red edges and red otherwise.

Theorem 3.1. *Any* 4*-regular graph has a* [2, 1] *path partition.*

Proof. Let G be a 4-regular graph. We show that Algorithm 2 constructs a [2, 1] path partition for G by showing that at the end of the algorithm G_{blue} and G_{red}, as defined above, and G_{green}, defined analogously, have the following properties:

(1) $\Delta(G_{\text{red}}) = \Delta(G_{\text{blue}}) = 2$.

(2) $\Delta(G_{\text{green}}) = 1$.

(3) G_{red} and G_{blue} are acyclic.

We define the *red degree* of a vertex to be the degree of the vertex in G_{red} and define blue degree and green degree similarly.

Notice that the effect of step 3.1 is to leave a coloring and marking that could have been produced by Steps 1 and 2 that just happened to have no cycles of special

edges. Actually step 3.1 does not simulate step 2.3 faithfully in that blue edges marked by step 3.1 as special edges may not be adjacent to as many special red edges as possible, but we only require that the following weaker claim holds.

Claim. *After step* 3.1, *if* $\{v, u\}$ *is a special blue edge, and u is an endpoint of a path of special edges, and* $\{v, w\}$ *is also blue, then there is no special edge incident to w. Further, after step* 3.1, *if any vertex of a blue cycle is incident to a special red edge, then the special edge of the blue cycle is adjacent to a special red edge.*

This follows immediately from the details of steps 2.3 and 3.1.

$\Delta(G_{\text{red}}) = 2$. At the beginning of Step 4 no vertex has red degree greater than 2. In the remainder of the algorithm only step 4.4 can color an edge red and step 4.4 can color an edge red only if one endpoint has blue degree 3 and the other has red degree not equal to 2. Given that there are initially no vertices of red degree greater than 2, step 4.4 cannot create a vertex of red degree greater than 2 and $\Delta(G_{\text{red}}) = 2$.

$\Delta(G_{\text{blue}}) = 2$. At the beginning of Step 4 no vertex has blue degree greater than 2. Step 4.1 introduces one vertex of blue degree 3 for every path of special edges of length 2, but step 4.4 decreases the blue degree of every such vertex by recoloring an edge incident to the vertex red or green. Every edge recolored by step 4.4 was blue so every recoloring reduces the blue degree of the incident vertices. To see that every edge recolored by step 4.4 was blue note that the only edge on an initial blue cycle that can be not blue at the beginning of step 4.4 is the special edge and that step 4.4 can recolor an edge only if it is on a blue initial cycle and is not special. The special edge cannot be recolored by step 4.4 because neither endpoint can have blue degree 3. (The special edge, with one exception, is not blue and at least one of the edges at each endpoint remains red. The exception is the case described in step 4.3 in which the special edge is blue but both endpoints are incident on two nonblue edges.) Since every vertex in G is on an initial blue cycle, every vertex with blue degree 3 is visited in step 4.4 and has its blue degree reduced.

$\Delta(G_{\text{green}}) = 1$. The green edges are a matching at the beginning of step 4.2. The only remaining opportunity to color an edge green is in step 4.4, and it recolors an edge green only if the vertex currently visited, w, has blue degree 3 and the previous vertex, v, has red degree 2. We show that neither w nor v is already incident to a green edge. The only edge incident to w that is not blue must be red since every edge on an initial red cycle except the special edge remains red. We show that the green degree of v is 0 by contradiction. If we assume that before edge $\{v, w\}$ is recolored v already has green degree 1 and also has red degree 2, then v must be the special vertex of its initial blue cycle. (If v is not special, then it will be incident not to *one*, but to *two* blue edges. We showed above that the edge from v to w that will be recolored green in step 4.4 is blue. If v is not special and v did not have blue degree 3 when it was visited in step 4.4, then the edge preceding v on the initial blue cycle is also blue. If v is not special and it did have blue degree 3, then one of the two

edges incident to v that are not on the blue cycle is blue.) Given that v is the special vertex and has green degree 1 and red degree 2, there are two cases to consider, either the special edge, $\{v,u\}$, of the blue initial cycle is red or it is green.

Assume that $\{v,u\}$ is green. Both of the edges incident to v that belong to the initial red cycle containing v are red and neither is special. Therefore, since w was incident to a special edge we know from the above claim that u must be incident to a special edge from an initial red cycle. The edges incident to u that are not on the initial blue cycle are therefore the special edge from an initial red cycle and an edge that has remained red from the same cycle. This special edge must be blue; no special edge from a red initial cycle remains red and the edge cannot be green. (The edge could have been colored green in step 3.2 if it was incident to the endpoint of a path of special edges, but it would then have been recolored blue in step 4.1 since the adjacent edge $\{v,u\}$ is also incident to the endpoint of a path of special edges.) By the rule for selecting a special vertex (step 4.2) u must be chosen as the special vertex, contradicting the assumption that v is the special vertex.

Assume instead that $\{v,u\}$ is red. Of the edges incident to v that are not on the initial blue cycle one is red and the other is green. It follows that u must have been incident to a special edge from an initial red cycle. The edges incident to u that are not on the initial blue cycle must be either red and blue or red and green. They cannot be red and green because then $\{v,u\}$ would have been recolored blue in step 4.3. Therefore, they are red and blue. But the rule for selecting a special vertex would then dictate that u be selected as the special vertex, again contradicting the assumption that v is the special vertex.

We have shown that whenever an edge is colored green in step 4.4 neither of endpoints is already incident to a green edge. Step 4.4 therefore preserves the property that the green edges form a matching.

G_{red} is acyclic. Suppose C is a cycle in G_{red}. It is easy to see that no red cycle exists at the beginning of Step 4, so at least one edge of C must be an edge colored red in 4.4. Let $\{t,x\}$ be that edge and x be the vertex with blue degree 3 which forced the recoloring of $\{t,x\}$ and let B be the initial blue cycle containing x. Cycle C must leave B at x. (The other edge incident to x along B cannot also be red or x would have red degree 3.) Let $\{x,v\}$ on the initial red cycle R be the blue edge incident to B at x. Edge $\{x,v\}$ is an edge on a path of special edges of length 2. Since no red path can leave an initial red cycle except at the vertices incident to the cycle's special edge and the red degree of v is 1, the sequence of red edges leaving B at x must end at v. (The red degree of v before step 4.4 is 1, and that step will not recolor any edge incident to v. In particular, the blue edge $\{v,w\}$ from the initial blue cycle containing v will remain blue since w cannot have blue degree 3, by the claim given above.) Therefore C cannot exist and G_{red} is acyclic.

G_{blue} is acyclic. Assume G_{blue} contains a cycle, C. Since no blue cycle exists at the beginning of Step 4 and only steps 4.1 and 4.3 in the remainder of the algorithm can color an edge blue, at least one edge on C must have been colored blue in step 4.1 or 4.3. Assume that C contains an edge colored blue in step 4.1 and let $\{u,v\}$

be that edge. Edge $\{u, v\}$ is the special edge of a red cycle and is an edge on a path of special edges of length 2. Let $\{v, w\}$ be the other edge on that path; $\{v, w\}$ is on an initial blue cycle B, and was colored green in step 3.2. Vertex w is the endpoint of a special path and is incident to two red edges. If no edge incident to B was colored blue in step 4.1, then $\{u, v\}$ is on a blue path ending at w. Therefore, C enters B at vertex v, uses one or more blue edges in B and leaves B on an edge also colored blue in 4.1 at some vertex x on B. But by the rule for selecting a special vertex, v is B's special vertex. (The edges incident to w that are not on B are both red and the edges incident to v that are not on B are red and blue.) Since v is B's special vertex the edge encountered before x on the traversal of B is colored green or red in step 4.4 and the blue path leaving v ends before reaching x. Therefore no cycle of blue edges can contain an edge colored blue in step 4.1. If C avoids edges colored in 4.1, it must contain at least one edge colored in step 4.3. Let $\{v, w\}$ on the initial blue cycle B be that edge. Cycle B is not the same as C since we do not recolor $\{v, w\}$ blue unless we know that we will color some edge of B green or red in step 4.4. Therefore C uses the edge $\{v, w\}$ and leaves B on two blue edges incident to B. We have shown that no edge of C can be an edge colored blue in 4.1. The only other points at which blue edges can touch B are at the vertices incident to B's special edge, $\{v, w\}$. But $\{v, w\}$ is the middle edge in a path of special edges of length 3 and the edges incident to $\{v, w\}$ that are not on B are red and green. Therefore G_{blue} is acyclic. □

Corollary 3.2. *Any graph G with $\Delta(G) = 4$ has a $[2, 1]$ path partition.*

Proof. Any graph G with $\Delta(G) = 4$ can be converted into a 4-regular graph by the addition of a linear number of edges and vertices. Any $[2, 1]$ partition of the resulting graph H yields a $[2, 1]$ partition of G. □

Algorithm 2, like the previous algorithm, has efficient sequential and parallel implementations. In a sequential version, it is clear in each step every edge is seen a constant number of times. Hence it runs in linear time. The only nontrivial steps for a parallel implementation involve cycle processing, such as finding and coloring Euler circuits and identifying cycles of special edges. A nontrivial section is step 4.4. For this step each block of contiguous nodes of blue degree 3 is identified. The first (in the direction specified by the special vertex) node before each block is checked and its number of red edges determines the coloring of all the blue edges for that block (by a simple parity check). This can be done with standard recursive doubling code. Hence the algorithm can be executed on a CRCW PRAM with $O(n + m)$ processors in $O(\log n)$ time.

4. Conclusions

We have presented a new graph property, the $[a, b]$ path partition, and have given

sharp results for graphs with maximum degree ≤ 4. How can our theorems be generalized? The most obvious conjecture, for any graph G with $\Delta(G)=k$, is that there exists a $[\frac{1}{2}(k+1), 0]$ path partition when k is odd, and there exists a $[\frac{1}{2}k, 1]$ path partition when k is even. For reasons such as stated in the first section, these conjectures are best possible. Of course these imply Vizing's theorem.

It is not clear how the algorithmic proofs could be extended. The proof for 4-regular graphs is really quite complex and the details become worse for larger k. However, we can begin by generalizing Step 1, when k is even, by finding an edge coloring using $\frac{1}{2}k$ colors, with each color occurring twice at each vertex. A theorem of Petersen states that it is always possible. An algorithmic proof, using consecutive Euler circuit computations, shows that it can be determined in polynomial time [4, p. 230].

We have an unpublished linear-time algorithm for cubic graphs which can transform an arbitrary [2, 1] path partition, which is relatively easy to find, into a [2, 0] path partition. Another algorithmic approach for a general result would be to extend this sort of algorithm to graphs with larger degrees. We have not been able to do that.

References

[1] E. Arjomandi, An efficient algorithm for colouring the edges of a graph with $\Delta + 1$ colours, INFOR 20 (1982) 82–101.

[2] M. Atallah and U. Vishkin, Finding Euler tours in parallel, J. Comput. System Sci. 29 (1984) 330–337.

[3] B. Awerbuch, A. Israeli and Y. Shiloach, Finding Euler Circuits in Logarithmic Parallel Time, Proceedings 16th ACM Symposium on Theory of Computing (1984) 249–257.

[4] C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1973).

[5] I. Holyer, The NP-completeness of edge-coloring, SIAM J. Comput. 10 (1981) 718–720.

[6] H.J. Karloff and D.B. Shmoys, Efficient parallel algorithms for edge coloring problems, J. Algorithms 8 (1987) 39–52.

[7] C.P. Kruskal, L. Rudolph and M. Snir, The power of parallel prefix, IEEE Trans. Comput. 34 (1985) 965–968.

[8] L. Lovász, On covering of graphs, in P. Erdos et al., eds., Theory of Graphs (Academic Press, New York, 1968) 231–236.

[9] C. Thomassen, A remark on the factor theorems of Lovász and Tutte, J. Graph Theory 5 (1981) 441–442.

[10] W.T. Tutte, The subgraph problem, Annals of Discrete Mathematics 3 (North-Holland, Amsterdam, 1978) 289–295.

[11] V.G. Vizing, On an estimate of the chromatic class of a p-graph, Diskret. Analiz. 3 (1964) 25–30 (in Russian).