On the Complexity of the Max-Edge-Coloring Problem with Its Variant

Chang Wu Yu
Department of Computer Science and Information Engineering
Chung Hua University, Hsinchu, Taiwan 300, R.O.C.
{cwyu}@chu.edu.tw

Abstract

The max-edge-coloring problem (MECP) is finding an edge colorings {E_1, E_2, E_3, ..., E_z} of a weighted graph $G=(V, E)$ to minimize $\sum_{i=1}^{z} \max \{w(e_k) | e_k \in E_i\}$, *where $w(e_k)$ is the weight of e_k. In the work, we discuss the complexity issues on the new graph problem and its variants. Specifically, we design a 2-approximmation algorithm for the max-edge-coloring problem on biplanar graphs. Next, we show the splitting chromatic max-edge-coloring problem, a variant of MECP, is NP-complete even when the input graph is restricted to biplanar graphs. Finally, we also show that these two problems have applications in scheduling data redistribution on parallel computer systems.*

Keywords: *edge coloring, bipartite graphs, multi-graphs, algorithm design*

1. Introduction

The *max-edge-coloring* problem (MECP) is finding an edge colorings $\{E_1, E_2, E_3, \ldots, E_z\}$ of a weighted graph G to minimize $\sum_{i=1}^{z} \max \{w(e_k) | e_k \in E_i\}$, where $w(e_k)$ is the weight of e_k. We can also define its two variants by using the minimum number of coloring as follows. The *chromatic max-edge-coloring* problem (CMECP) is finding an edge colorings $\{E_1, E_2, E_3, \ldots, E_\Delta\}$ of a weighted graph G to minimize $\sum_{i=1}^{\Delta} \max \{w(e_k) | e_k \in E_i\}$, where Δ is the maximum degree of G and $w(e_k)$ is the weight of e_k. Given a weighted graph $G=(V, E)$, the *splitting chromatic max-edge-coloring* problem (SCMECP) is an decision problem whether we can add extra edges E' such that the resulting graph $G'=(V, E \cup E')$ satisfies the following four conditions:

(1) Another edge (s, t) can be added to E' only if there exists an edge (s, t) in E.

(2) The maximum degree of the resulting new graph is the same; that is, $\Delta(G)=\Delta(G')$.

(3) The value $\sum_{i=1}^{\Delta} \{w(e_k) | e_k = (s,t) \text{ in } E' \bigcup E\}$ equals the weight of (s, t) in E, where $w(e_k)$ is the weight of e_k.

(4) There is an edge coloring $\{E_1, E_2, E_3, \ldots, E_\Delta\}$ of $G'=(V, E \cup E')$ such that $\sum_{i=1}^{\Delta} \max \{w(e_k) | e_k \in E_i\} \le K$, where K is given and $w(e_k)$ is the weight of e_k.

To the best of our knowledge, the above three problems are defined formally for the first time. Although some heuristic algorithms have been

designed for the CMECP [19, 20, 21], both the MECP and the CMECP are still open to devise a polynomial-time algorithm even when the input graph is restricted to biplanar graphs. The first contribution of the work is showing the existence of 2-approximation algorithm for the MECP when the input graph is restricted to biplanar graphs. The SCMECP is a new graph problem defined here, which have applications in shortening the overall communication time for data redistribution in parallel systems. In this work, we will show the SCMECP is NP-complete even when the input graph is restricted to biplanar graphs.

Only a similar problem called the *max-coloring problem* [28, 29] can be found in literature. The problem is to find a proper vertex coloring of input graphs whose color classes C_1, C_2, …, C_k, minimize $\sum_{i=1}^{k} \max \{w(e) | e \in C_i\}$ where $w(e)$ is the weight of e. The max-coloring problem has been shown to be NP-hard on interval graphs [28]; however, there exist some approximation algorithms for the problem on many well-known subclasses of graphs including bipartite graphs, interval graphs, circle graphs, circular arc graphs, unit disk graph, chordal graphs, and permutation graphs [28, 29].

The rest of the paper is organized as follows. Section 2 presents necessary definitions and notations. Next, Section 3 presents a 2-approximation algorithm for the MECP when the input graph is restricted to biplanar graphs. The SCMECP is shown to be NP-complete even when the input graph is restricted to biplanar graphs in Section 4. The applications of these problems to scheduling problems for data redistribution in parallel systems are discussed in Section 5. Finally, Section 6 concludes the paper.

2. Definitions and notations

A *graph* G consists of a finite nonempty vertex set together with an edge set. A *bipartite graph* $G=(S, T, E)$ is a graph whose vertex set can be partitioned into two subsets S and T such that each of the edges has one end in S and the other end in T. A typical convention for drawing a bipartite graph $G=(S, T, E)$ is to put the vertices of S on a line and the vertices of T on a separate parallel line and then represent edges by placing straight line segments between the vertices that determine them. In this convention, a drawing is *biplanar* if edges do not cross, and a graph G is *biplanar* if it has a biplanar drawing [27]. A graph is *connected* if there is a path joining each pair of nodes. An *acyclic* graph is one that contains no cycles. A *forest* is an acyclic graph. A *tree* is a connected acyclic graph. A *component* of a graph is a maximal connected subgraph. The number of components of G is denoted by $\omega(G)$.

Let $N(v)$ denote the set of vertices which are adjacent to v in G. The ends of an edge are said to be *incident* with the edge. Two vertices which are incident with a common edge are *adjacent*. A *multi-graph* is a graph allowing more than one edge to join two vertices. The degree $d_G(v)$ of a vertex v in G is the number of edges of G incident with v. We denote the maximum degree of vertices of G by $\Delta(G)$. A *complete bipartite graph* $G=(S, T, E)$ is a graph such that each vertex of S is joined to each vertex of T; if $|S|=m$ and $|T|=n$, such a graph is denoted by $K_{m, n}$.

An ordering of S (T) has the *adjacency property* if for each vertex $v \in T(S)$, $N(v)$ contains consecutive vertices in this ordering. The graph $G=(S, T, E)$ is called a *doubly convex-bipartite graph* if there are orderings of S and T having the adjacency property [24]. A graph is called *interval*

graph if its vertex set can be represented by using a finite number of interval on a straight line and two vertices are connected by an edge when the corresponding intervals overlap at least partially.

The *line graph* $L(G)$ of a graph $G=(V, E)$ is defined so that there is a one-to-one correspondence between the vertices in $L(G)$ and the edges in G. That is, there is an edge joining two vertices in $L(G)$ when their corresponding edges in G are incident with a common vertex.

The coloring is *proper* if no two adjacent edges have the same color. An edge with identical ends is called a *loop*. A *k-edge coloring* of a loopless graph G is an assignment of k colors to the edges of G. G is *k-edge colorable* if G has a proper k-edge coloring. The *edge chromatic number* $\chi'(G)$, of a loopless graph G, is the minimum k for which G is k-edge-colorable. A subset M of E is called a *matching* in $G=(V, E)$ if its elements are links and no two are adjacent in G. Note that the each edge set with the same color in a proper edge coloring forms a matching. At last, most graph definitions used in the paper can be found in [22].

An algorithm that generates near-optimal solution is called an *approximation algorithm*. We say that an approximation algorithm has a ratio-bound of ρ, called *ρ-approximation algorithm*, if for any input of size, the cost C of the solution produced by the approximation algorithm is within a factor of ρ of the cost C^* of an optimal solution: $\max(C/C^*, C^*/C) \leq \rho$.

3. The 2-approximation algorithm for the max-edge-coloring problem when the input graph is restricted to biplanar graphs

The section gives a 2-approximation algorithm for the max-edge-coloring problem when the input graph is restricted to biplanar graphs. First, the following theorem presents additional properties of biplanar graphs.

Theorem 1: The following four statements are equivalent [25-27]:

(1) A bipartite graph G is biplanar.

(2) The graph G is a collection of disjoint caterpillars.

(3) The graph G contains no cycle and no double claw.

(4) The graph G^* that is the remainder of G after deleting all vertices of degree one, is acyclic and contains no vertices of degree at least three.

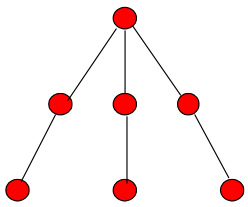

Figure 1. A double claw.

Here a *caterpillar* is a connected graph that has a path called the *backbone b* such that all vertices of degree larger than one lie on b; and a *double claw* graph is depicted in Figure 1.

The size of edge set of a general bipartite graph is at most $O(n^2)$, where n is the number of vertices in the graph. However, if the input graph can be drawn in a plane without crossing edges (i.e., it is a planar graph), the size of the edge set is less than $3n$-6 [22]. Since biplanar graphs are intrinsically planar, the size of the edge set of biplanar graphs is less than $3n$-6. In fact, a biplanar graph is a tree, the size of whose edge set is n-1.

Theorem 2: The line graph of a biplanar graph is an interval graph.

Proof: Given a biplanar graph $G=(S, T, E)$, we will construct an interval model to represent the line graph of G. Since G is biplanar, we have a biplanar drawing of G on two horizontal lines. By preserving the orderings of the biplanar drawing, this drawing can be further rearranged to satisfy the following two properties (See Figure 2 for example):

(1) Every vertex v of S and T has distinct $\chi(v)$ value and $1\le\chi(v)\le|S|+|T|$. Hereafter $\chi(v)$ denotes the x-coordinate of a vertex v in the biplanar drawing of G.

(2) The integer set $\{\chi(u)|u\in N(v)$ and $d(u)=1\}$ consists of consecutive integers for every vertex $v\in S\cup T$.

The first property can be achieved by scaling the x-coordinates of vertices properly, and the second by packing the degree-one vertices which incident to the other same vertex.

According to the drawing, each vertex of G is labeled by using its x-coordinate, and an interval $[x\text{-o.5}, y\text{+0.5}]$ is created for each edge (x, y) in E (See Figure 2 for example). The remainder is to show that the set of intervals represents the line graph of G. Suppose that two vertices are adjacent in $L(G)$ and the corresponding edges in G are $e_1=(u, v)$ and e_2. Without loss of generality, let $u\le v$ and the interval $[u\text{-o.5}, v\text{+0.5}]$ represents e_1. That implies that e_2 must be incident with u or v; and the interval created for e_2 is $[z, u\text{+0.5}]$ or $[v\text{-o.5}, z]$, both of which overlap with $[u\text{-o.5}, v\text{+0.5}]$.

On the other hand, given any pair of overlapped intervals $[x_1, y_1]$ and $[x_2, y_2]$ in the set, we have $x_1\le x_2<y_1\le y_2$. We claim that the corresponding two edges in G are incident with a common vertex; that also indicates the corresponding two vertices are adjacent in $L(G)$. Otherwise, the edges are not incident with a common vertex, and the created intervals for the edges must be $[x_1, y_1]$ and $[x_2, y_2]$ and $x_1< y_1<x_2<y_2$ according to the two properties of the drawing. A contradiction occurs.

Thus, the set of intervals represents the line graph of G, and $L(G)$ is an interval graph. ■

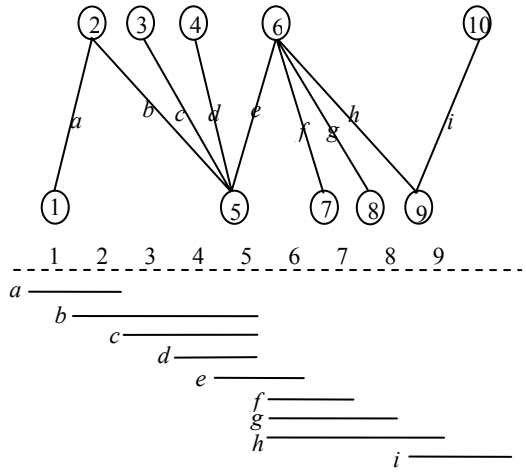

Figure 2. A biplanar graph G with the interval model of its line graph.

Based on above theorem, an algorithm for the max-edge-coloring problem is described as follows.

Algorithm AMEC:

Input: a biplanar graph $G=(S, T, E)$.

Output: an edge colorings $\{E_1, E_2, E_3, \ldots, E_z\}$ of G and the value $\sum_{i=1}^{z}\max\{w(e_k)|e_k\in E_i\}$, where $w(e_k)$ is the weight of e_k.

Step 1: Constructing the line graph $L(G)$ from G.

Step 2: Finding an interval model for $L(G)$ by applying an interval graph recognition algorithm.

Step 3: Finding a vertex coloring $\{C_1, C_2, C_3, \ldots, C_z\}$ of $L(G)$ by applying a 2-approximation algorithm for solving the max-coloring problem on interval graphs.

Step 4: Constructing an edge colorings $\{E_1, E_2, E_3, \ldots, E_z\}$ of G from $\{C_1, C_2, C_3, \ldots, C_z\}$ and compute the value $\sum_{i=1}^{z} \max\{w(e_k) | e_k \in E_i\}$.

The correctness and performance guarantee of algorithm AMEC are demonstrated in the next theorem.

Theorem 3: Algorithm AMEC is a 2-approximation algorithm for the max-edge-coloring problem.

Proof: Since an edge coloring of an graph G corresponds to a vertex coloring of its line graph $L(G)$, the *max-edge-coloring* problem of G can be transformed to the *max-coloring* problem of $L(G)$. Since $L(G)$ is an interval graph (by Theorem 2), we obtain a 2-approximation algorithm for max-edge-coloring problem of G by applying Pemmaraju et al 's 2-approximation algorithm for the max-coloring problem on interval graphs [28]. Finally, we conclude that Algorithm AMEC is a 2-approximation algorithm for the max-edge-coloring problem. ■

The time complexity of AMEC is discussed as follows. Step 1 requires $O(n^2)$ time because the step compares at most every pair in the edge set (whose size of edge set is $|E|=O(n)$) of G for constructing the edge set of $L(G)$. Step 2 can be implemented in $O(n^2)$ time if we apply Booth and Lueker's linear-time recognition algorithm for interval graphs [30]. Step 3 takes $O(n\log n)$ time if we apply Pemmaraju et al 's 2-approximation algorithm for the max-coloring problem on interval graphs [28]. Finally, Step 4 can be implemented in $O(n)$ time. The next theorem makes a summary.

Theorem 4: When the input graph is biplanar, AMEC is a 2-approxmiation algorithm for the max-edge-coloring problem requiring $O(n^2)$ time, where n is the size of the vertex set of the input graph.

4. The splitting chromatic max-edge-coloring problem is NP-complete when the input graph is restricted to biplanar graphs

When the input graph is restricted to biplanar graphs, we will prove that the SCMECP is NP-complete by transforming from the partition problem: Given a finite set A and a weight $s(a) \in Z^+$ for each $a \in A$, *the partition problem* is to ask whether there is a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$.

Theorem 5: The SCMECP is NP-complete even the input graph is restricted to biplanar graphs.

Proof: It is easy to see that SCMECP$\in$NP, since a nondeterministic algorithm need only guess the set of added edges E' with its associated weights and check whether the above four conditions are satisfied.

We will transform the partition problem to SCMECP. Let $A=\{a_1, a_2, \ldots, a_n\}$and given weight $s(a)$ for each $a \in A$ constitute an arbitrary instance of the partition problem. We can construct a weighted biplanar graph $G=(S, T, E)$ such that (1) vertex set $S=\{s_x, s_y, s_1, s_2, \ldots, s_n\}$ and $T=\{t_1, t_2\}$, (2)

edge set $E=\{(s_x, t_1), (s_y, t_1), (s_1, t_2), (s_2, t_2), \ldots, (s_n, t_2)\}$, and (3) the weights of (s_x, t_1) and (s_y, t_1) are $\sum_{a\in A} s(a)/2$, and that of (s_i, t_2) is $s(a_i)$ for i=1 to n.

Figure 3 illustrates the construction when $A=\{2, 3, 4, 6, 8, 9\}$ where n=6.

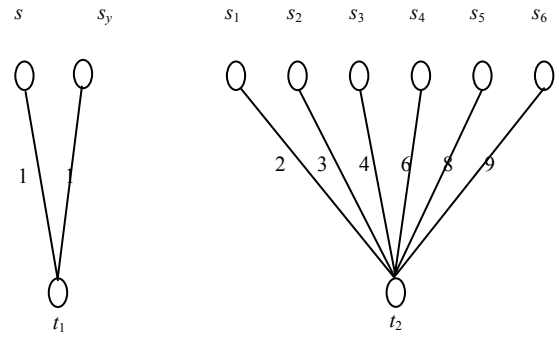

Figure 3. SMECP instance resulting from partition instance in which $A=\{2, 3, 4, 6, 8, 9\}$.

It is easy to show how the construction can be accomplished in polynomial time. All that remains to be shown is that there is a set subset $A'\subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(a)$ if and only if there is an edge coloring $\{E_1, E_2, E_3, \ldots, E_\Delta\}$ of $G'=(S, T, E\cup E')$ such that $\sum_{i=1}^{\Delta} \max\{w(e_k)|e_k \in E_i\} \le K=\sum_{a\in A} s(a)$.

Suppose that there is a set subset $A'=\{a_{o(1)}, a_{o(1)}, \ldots, a_{o(k)}\}\subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(a)$, obviously $\sum_{1\le i\le k} s(a_{o(i)}) = (\sum_{a\in A} s(a))/2 = K$. We can then construct the corresponding weighted biplanar graph $G=(S, T, E)$ according to the above discussions. Let the extra adding edges E' consist of $o(k)$-1 copies of (s_x, t_1) and n-$o(k)$-1 copies of (s_y, t_1). Consequently, the maximum degree of the resulting graph remains n; that is, $\Delta(G')=\Delta(G)=n$. Since $\sum_{1\le i\le k} s(a_{o(i)}) = (\sum_{a\in A} s(a))/2 = K$ and $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(a)$, we can reassign i^{th} edge of $o(k)$ (s_x, t_1) with weight $s(a_{o(i)})$ and the other n-$o(k)$ edges of (s_y, t_1) with weight $s(a_{o(i)})$.

There exists an edge coloring $\{E_1, E_2, E_3, \ldots, E_n\}$ of $G'=(S, T, E\cup E')$ where $E_i=\{(s_x, t_1), (s_{o(i)}, t_2)\}$ for $1\le i\le k$ and $E_i=\{(s_y, t_1), (s_{o(j)}, t_2)\}$ for $k+1\le j\le n$. Evidently, $\sum_{i=1}^{n} \max\{w(e_k)|e_k \in E_i\} = K$.

Figure 4 depicts the corresponding edge coloring of G' constructed from G shown in Figure 3.

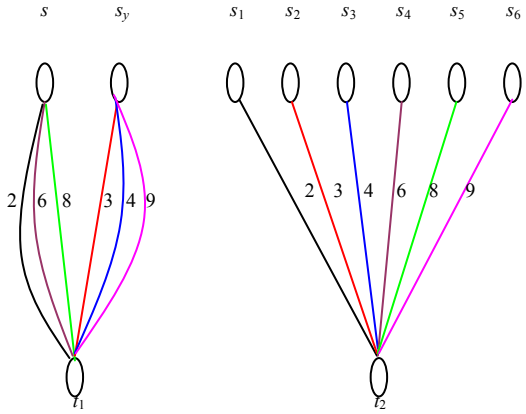

Figure 4. The corresponding edge coloring of G'.

Conversely, suppose that there is an edge coloring $\{E_1, E_2, E_3, \ldots, E_\Delta\}$ of $G'=(S, T, E\cup E')$ such that $\sum_{i=1}^{\Delta} \max\{w_k|e_k \in E_i\} \le K=\sum_{a\in A} s(a)$, where w_k is the weight of e_k. According to the graph construction of G, the weights of (s_x, t_1) and (s_y, t_1) equals $\sum_{a\in A} s(a)/2$, respectively, and that of (s_i, t_2) is $s(a_i)$ for i=1 to n. Consequently, there is a subset $A'=\{a_{o(1)}, a_{o(1)}, \ldots, a_{o(k)}\}\subseteq A$ such that $\sum_{a\in A'} s(a) = \sum_{a\in A-A'} s(a)$. Otherwise, $\sum_{i=1}^{\Delta} \max\{w_k|e_k \in E_i\} > K$, a contradiction

occurs. ■

5. Applications in scheduling data redistribution

We also find applications in scheduling problems for data redistribution in parallel systems for these problems in this section.

5.1 Applications of MECP and CMECP

Array redistribution is crucial for system performance because a specific array distribution may be appropriate for the current phase, but incompatible for the subsequent one. Many parallel programming languages thus support run-time primitives for rearranging a program's array distribution. Therefore developing efficient algorithms for array redistribution is essential for designing distributed memory compilers for those languages. While array redistribution is performed at run time, a trade-off occurs between the efficiency of the new data rearrangement for the coming phase and the cost of array redistributing among processors.

In irregular redistribution, messages of varying sizes are scheduled in the same communication step. Therefore, the largest size of message in the same communication step dominates the data transfer time required for this communication step.

A bipartite graph model will be introduced to represent data redistributions. Any data redistribution can be represented by a bipartite graph $G=(S, T, E)$, called a *redistribution graph*. Where S denotes source processor set, T denotes destination processor set, and each edge denotes a message required to be sent. For example, a Block-Cyclic(x) to Block-Cyclic(y) data redistribution from P processors to Q processors (denoted by $BC(x, y, P, Q)$) can be modeled by a bipartite graph $G_{BC(x, y, P, Q)}=(S, T, E)$ where $S=\{s_0, s_1, \ldots, s_{|s|-1}\}$ ($T=\{t_0, t_1, \ldots, t_{|t|-1}\}$) denotes the source processor set $\{p_0, p_1, \ldots, p_{|s|-1}\}$ (destination processor set$\{p_0, p_1, \ldots, p_{|t|-1}\}$) and we have $(s_i, t_j)\in E$ with weight w if source processor p_i has to send the amount of w data elements to destination processor p_j. For simplicity, we use $BC(x, y, P)$ to denote $BC(x, y, P, P)$. Figure 5 depicts the a data redistribution pattern $BC(1, 4, 4)$, and its corresponding redistribution graph $G_{BC(1, 4, 4)}$ is shown in Figure 6.

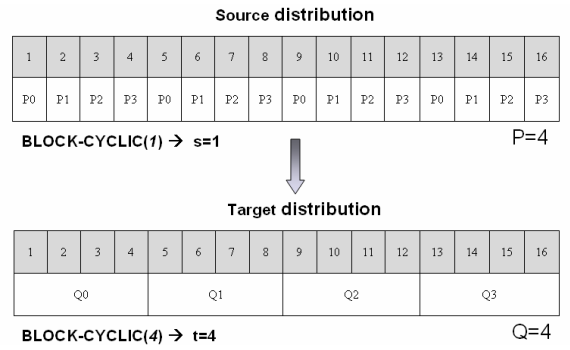

Figure 5. A data redistribution pattern $BC(1, 4, 4)$.

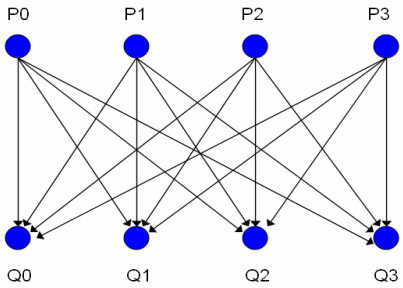

Figure 6. The redistribution graph $G_{BC(1, 4, 4)}$ is a complete bipartite graph.

Similarly, GEN_BLOCK data redistribution from P processors to Q processors (denoted by $GB(P, Q)$) can also be modeled by a bipartite graph $G_{GB(P, Q)}=(S, T, E)$. For example, a $GB(4, 4)$ with its redistribution graph $G_{GB(4, 4)}$ is depicted in Figure 7 and 8.

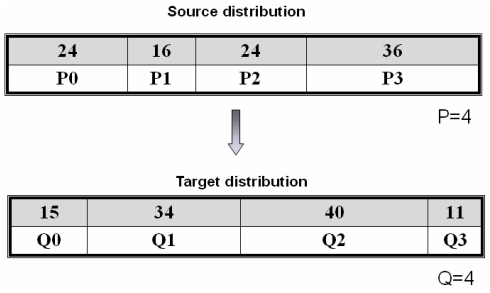

Figure 7. GEN_BLOCK data redistribution $GB(4, 4)$.

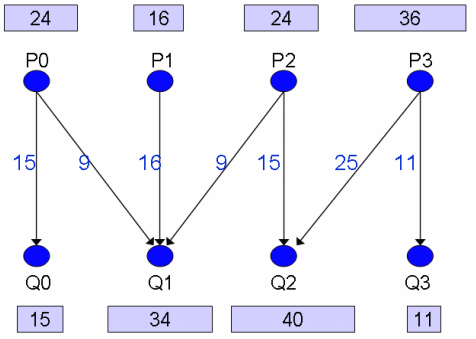

Figure 8. A redistribution graph $G_{GB(4, 4)}$.

Theorem 6: The redistribution graph of GEN-BLOCK is a biplanar graph.

Proof: Evidently the redistribution graph G is bipartite. The remainder is to show the planarity property. We may image that source processors and destination processors are assembled into two parallel lines (Figure 7 and 8). Subsequently consecutive array elements (described in the GEN-BLOCK format) allocates to a single source processor (and destination processor) one by one. Note that the elements in a source processor P_i must be reallocated to consecutive destination processors $\{Q_j, Q_{j+1}, \ldots, Q_k\}$ in the line; and the elements stored in the following source processor P_{i+1} may be reallocated to the consecutive destination processors from Q_k. Therefore, the resulting redistribution bipartite graph can be drawn without any crossing edge. ■

A set of conflict-free data communication can be represented by a matching of the given redistribution graph G. Thus, the data redistribution problem can be modeled as the MECP and CMECP.

5.2 Applications of SCMECP

Designing data redistribution scheduling algorithms for CMECP encounters a difficulty: shortening the overall communication time without increasing the number of communication steps at the same time. Unlike existing algorithms, Yu *et al* [31] presented an algorithm to partition large data segments into multiple small data segments and properly schedule them in different communication steps without increasing the number of total communication steps. For example, Figure 9 depicts a redistribution graph G with a possible scheduling.

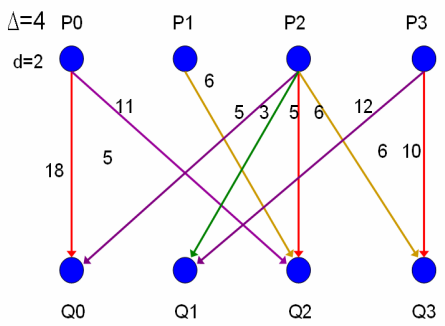

Figure 9. A redistribution graph G with maximum degree Δ=4.

Since G is bipartite, it is well known that $\chi'(G)=\Delta(G)$ [22]. That indicates that the minimum number of required communication steps (colors) equals the maximum degree Δ of the given distribution graph G. Therefore, we at least need four communication steps for the data redistribution since $\chi'(G)=\Delta(G)=4$. In addition, the overall cost of the scheduling is 38 (See Table 1).

Table 1. Costs of the scheduling correspond to the edge coloring in Figure 9.

	1(red)	2(yellow)	3(green)	4(purple)	Total
Cost	18	6	3	11	38

Note that the cost of Step 1 (colored in red) is dominated by the data segment (with 18 data elements) from P_0 to Q_0. Suppose that we partition the data of the segment into two data segments (with 9 and 9 data elements respectively) and transmit them in different steps; then the cost required for Step 1 is reduced to 10 (Currently the step is dominated by the data segment from P_3 to Q_3). We can represent the kind of message partition by adding a new edge (P_0, Q_0) in the original redistribution graph and sharing weight with the old edge (P_0, Q_0). Similarly, we can partition other large data segments into multiple small data segments if the maximum degree of the resulting redistribution graph remains unchanging. After several data partitions, the overall communication cost can be reduced to 29 (or equivalently 76%) and the number of required communication step is still minimized (see Figure 10 and Table 2).

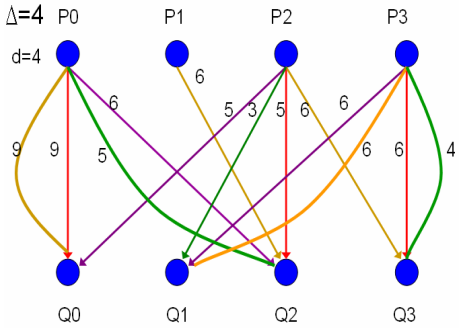

Figure 10. The resulting redistribution graph after partitioning long data segments.

Table 2. Costs of the scheduling after partitioning long data segments.

Step	1(red)	2(yellow)	3(green)	4(purple)	Total
Cost	9	9	5	6	29

Evidently, the above technique can be modeled by the SCMECP. We have shown the SCMECP is NP-complete even when the input graph is biplanar by Theorem 5.

6. Conclusions

In this work, we have designed the first 2-approximmation algorithm for the MECP on biplanar graphs. We also proved that the SCMECP is NP-complete even when the input graph is restricted to biplanar graphs. These two problems find applications in scheduling data redistribution on parallel computer systems. The authors believe that these newly defined graph problems deserve serious attention and can be applied to tackle more practical problems in diverse fields.

References:

[1] G. Bandera and E.L. Zapata, "Sparse Matrix Block-Cyclic Redistribution," *Proceeding of IEEE Int'l. Parallel Processing Symposium* (IPPS'99), San Juan, Puerto Rico, April 1999.

[2] Frederic Desprez, Jack Dongarra, and Antoine Petitet, "Scheduling Block-Cyclic Data redistribution," *IEEE Trans. on PDS*, vol. 9, no. 2, pp. 192-205, Feb. 1998.

[3] C.-H Hsu, S.-W Bai, Y.-C Chung, and C.-S Yang, "A Generalized Basic-Cycle Calculation Method for Efficient Array Redistribution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 12, pp. 1201-1216, Dec. 2000.

[4] C.-H Hsu, Dong-Lin Yang, Yeh-Ching Chung, and Chyi-Ren Dow, "A Generalized Processor Mapping Technique for Array Redistribution," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, vol. 7, pp. 743-757, July 2001.

[5] Minyi Guo, "Communication Generation for Irregular Codes," *The Journal of Supercomputing*, vol. 25, no. 3, pp. 199-214, 2003.

[6] Minyi Guo and I. Nakata, "A Framework for Efficient Array Redistribution on Distributed Memory Multicomputers," *The Journal of Supercomputing*, vol. 20, no. 3, pp. 243-265, 2001.

[7] Minyi Guo, I. Nakata, and Y. Yamashita, "Contention-Free Communication Scheduling for Array Redistribution," *Parallel Computing*, vol. 26, no.8, pp. 1325-1343, 2000.

[8] Minyi Guo, I. Nakata, and Y. Yamashita, "An Efficient Data Distribution Technique for Distributed Memory Parallel Computers," *JSPP'97*, pp.189-196, 1997.

[9] Minyi Guo, Yi Pan, and Zhen Liu, "Symbolic Communication Set Generation for Irregular Parallel Applications," *The Journal of Supercomputing*, vol. 25, pp. 199-214, 2003.

[10] Edgar T. Kalns and Lionel M. Ni, "Processor Mapping Technique Toward Efficient Data Redistribution," *IEEE Trans. on Parallel and Distributed Systems*, vol. 6, no. 12, December 1995.

[11] S. D. Kaushik, C. H. Huang, J. Ramanujam and P. Sadayappan, "Multiphase data redistribution: Modeling and evaluation," *Proceeding of IPPS*'95, pp. 441-445, 1995.

[12] S. Lee, H. Yook, M. Koo, and M. Park, "Processor reordering algorithms toward efficient GEN_BLOCK redistribution," *Proceedings of the ACM symposium on Applied computing*, 2001.

[13] Y. W. Lim, Prashanth B. Bhat, and Viktor K. Prasanna, "Efficient Algorithms for Block-Cyclic Redistribution of Arrays," *Algorithmica*, vol. 24, no. 3-4, pp. 298-330, 1999.

[14] Neungsoo Park, Viktor K. Prasanna, and Cauligi S. Raghavendra, "Efficient Algorithms for Block-Cyclic Data redistribution Between Processor Sets," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, No. 12, pp.1217-1240, Dec. 1999.

[15] Antoine P. Petitet and Jack J. Dongarra, "Algorithmic Redistribution Methods for Block-Cyclic Decompositions," *IEEE Trans. on PDS*, vol. 10, no. 12, pp. 1201-1216, Dec. 1999.

[16] L. Prylli and B. Touranchean, "Fast runtime block cyclic data redistribution on multiprocessors," *Journal of Parallel and Distributed Computing,* vol. 45, pp. 63-72, Aug. 1997.

[17] S. Ramaswamy, B. Simons, and P. Banerjee, "Optimization for Efficient Data redistribution on Distributed Memory Multicomputers," *Journal of Parallel and Distributed Computing,* vol. 38, pp. 217-228, 1996.

[18] Akiyoshi Wakatani and Michael Wolfe, "Optimization of Data redistribution for Distributed Memory Multicomputers," short communication, *Parallel Computing*, vol. 21, no. 9, pp. 1485-1490, September 1995.

[19] Hui Wang, Minyi Guo, and Daming Wei, "Divide-and-conquer Algorithm for Irregular Redistributions in Parallelizing Compilers", *The Journal of Supercomputing*, vol. 29, no. 2, 2004.

[20] Hui Wang, Minyi Guo, and Wenxi Chen, "An Efficient Algorithm for Irregular Redistribution in Parallelizing Compilers," *Proceedings of 2003 International Symposium on Parallel and Distributed Processing with Applications*, LNCS 2745, 2003.

[21] H.-G. Yook and Myung-Soon Park, "Scheduling GEN_BLOCK Array Redistribution," *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems*, November, 1999.

[22] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, Macmillan, London, 1976.

[23] R. Cole and J. Hopcroft, "On edge-coloring bipartite graphs," *SIAM J. Comput.* vol. 11, pp. 540-546, 1982.

[24] C. W. Yu and G. H. Chen, "Efficient parallel algorithms for doubly convex-bipartite graphs," *Theoretical Computer Science*, vol. 147, pp. 249-265, 1995.

[25] P. Eades, B. D. McKay, and N. C. Wormald, "On an edge crossing problem," *Proc. 9th Australian Computer Science Conference*, Australian National University, 1986, pp. 327-334.

[26] N. Tomii, Y. Kambayashi, and Y. Shuzo, "On planarization algorithms of 2-level graphs," *Papers of tech. group on electronic computers, IECEJ*, EC77-38, pp. 1-12, 1977.

[27] C. W. Yu, "On the complexity of the maximum biplanar subgraph problem," *Information Science*, vol. 129, pp. 239-250, 2000.

[28] Sriram V. Pemmaraju, Rajiv Raman, and Kasturi R. Varadarajan, "Buffer minimization using max-coloring," *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004, pp. 562-571.

[29] Sriram V. Pemmaraju and Rajiv Raman, "Approximation algorithms for the max-coloring problem," *Lecture Notes in Computer Science*, vol. 3580, pp. 1064-1075, 2005.

[30] K. S. Booth, and G. S. Lueker, "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms," *J. Comput. System Sci.*, vol. 13, pp. 335-379, 1976.

[31] C. W. Yu, Ching-Hsien Hsu, Kun-Ming Yu, Chiu Kuo Lian, and Chun-I Chen "Irregular Redistribution Scheduling by partitioning Messages," *Springer-Verlag Lecture Notes in Computer Science (LNCS)*, vol. 3740, pp. 295-309, 2005.