On the max-weight edge coloring problem

Giorgio Lucarelli, Ioannis Milis, Vangelis Th. Paschos

UNIVERSITE PARIS DAUPHINE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

CNRS UMR 7024

CAHIER DU LAMSADE

248

Mars 2007

On the max-weight edge coloring problem

Giorgio Lucarelli, Ioannis Milis, Vangelis Th. Paschos

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

On the max-weight edge coloring problem

Giorgio Lucarelli*, Ioannis Milis
Dept of Informatics, Athens University of Economics and Business, 104 34, Athens, Greece
{gluc, milis}@aueb.gr
Vangelis Th. Paschos†
LAMSADE, Université Paris-Dauphine, 75016, Paris, France
paschos@lamsade.dauphine.fr

Abstract

We study the following generalization of the classical edge coloring problem: Given a weighted graph, find a partition of its edges into matchings (colors), each one of weight equal to the maximum weight of its edges, so that the total weight of the partition is minimized. We explore the frontier between polynomial and NP-hard variants of the problem as well as the approximability of the NP-hard variants.

Keywords: weighted edge coloring, polynomial algorithms, approximation algorithms

1 Introduction

In several communication systems messages are to be transmitted in a single hop from senders to receivers through direct connections established by an underlying switching network. In such a system, a sender (resp. receiver) cannot send (resp. receive) more than one messages at a time, while the transmission of messages between different senders and receivers can take place simultaneously. The scheduler of such a system establishes successive configurations of the switching network, each one routing a non-conflicting

*Funded by the PENED 2003 Programme of the EU and the Greek General Secretariat for Research and Technology

†Part of this work has been carried out while the author was visiting the Department of Informatics of Athens University of Economics and Business

subset of the messages from senders to receivers. Given the transmission time of each message, the transmission time of each configuration equals to the heaviest message transmitted. Moreover, in practice there is a significant setup delay to establish each configuration. Note, for example, that in optical switches this overhead may dominate the transmission time. The aim of the scheduler is to find a sequence of configurations such that all the messages to be finally transmitted and the total transmission time (including setup delays) to be minimized.

This problem can be formulated in graph-theoretic terms as following: senders and receivers can be considered as the vertices V of a weighted graph $G = (V, E)$, whose edges E correspond to messages and their weights $w(e),\ e \in E$, to the lengths (transmission times) of messages. Although the graph G obtained is originally a weighted directed one, it can be considered as an undirected one, since the directions of its edges do not play any role in the objective function we study here. Clearly, a configuration corresponds to a matching or a color (in terms of edge coloring) of the graph G. In this context, we ask for a partition $M = \{M_1, M_2, \ldots, M_s\}$ of E into matchings (colors), each one of weight $w_i = \max\{w(e)|e \in M_i\}$, such that $W = \sum_{i=1}^{s} w_i$ is minimized.

Due to the fact that the weight w_i of each color is defined as the maximum weight of the edges colored i, in the following we shall refer to this problem as Max-weight Edge Coloring (MEC) problem.

Clearly, if all the edges of G are of the same weight then the MEC problem reduces to the classical *edge coloring* problem, where the objective is to minimize the number of colors (matchings) required in order to assign different colors to neighbor edges.

The MEC problem is equivalent to the parallel batch scheduling problem with incompatibilities between jobs. According to the standard notation for scheduling problems we denote this variant as $1 \mid p - batch,\ graph \mid C_{max}$. In this variant jobs are no more independent but they correspond to the edges of a weighted *graph*. Edge weights correspond to processing times of jobs and the graph G describes incompatibilities between jobs i.e., jobs corresponding to adjacent edges cannot be scheduled in the same batch (color).

In this paper we study the frontier between polynomial and NP-variants of the MEC problem as well as the approximability of its NP-hard variants. In the next section we review briefly the related work, while in Section 3 we give the notation we use and some preliminaries. As the complexity of the

problem on trees remains open, in Section 4 we present a polynomial algorithm for trees of bounded degree and in Section 5 a polynomial algorithm for stars of chains. Finally in Section 6 we present an approximation for bipartite graphs which beats the known one for bipartite graphs of maximum degree $\Delta \leq 7$.

2 Related work

It is known that the MEC problem is strongly NP-hard even for: (i) bipartite graphs of maximum degree three and edge weights restricted to be to 1, 2 or 3 [8, 10], (ii) k-regular bipartite graphs for $k \geq 3$ [5], and (iii) cubic bipartite planar graphs [4]. Yet another NP-hardness proof for an equivalent formulation of the MEC problem on bipartite graphs in terms of matrix decomposition has been proposed in [15].

Demange et al. in [5] have been also shown that the MEC problem on k-regular bipartite graphs cannot be approximated within a ratio less than $\frac{2^k}{2^k-1}$, which for $k = 3$ becomes 8/7. This inapproximability result has been improved, by the same authors in [4], to 7/6 for bipartite cubic planar graphs.

On the other hand, a 2-approximation algorithm has been presented in [10] for bipartite graphs. It is easy to see that the same algorithm applies also for general graphs. In addition, a 5/3-approximation algorithm for bipartite graphs of maximum degree $\Delta = 3$ has been presented in [5]. In general, this algorithm gives an approximation ratio equal to $\frac{2\Delta-1}{3}$ for bipartite graphs with $\Delta \geq 3$. In [4] has been presented a new algorithm for bipartite graphs of maximum degree $\Delta = 3$ that achieves an approximation ratio equal to the 7/6-inapproximability result.

A natural idea to decrease the cost of a solution to such a weighted edge coloring problem is to allow the division of each edge e of G into parallel edges of weights adding up to $w(e)$. In fact, this idea corresponds to the notion of preemption in scheduling problems: interrupt the execution of a job (the transmission of a message) and complete it later. The weight of each color M_i is now defined as $w_i = \max\{w_i(e)|e \in M_i\}$, where $w_i(e)$ is the portion of the e's weight belonging to matching M_i, and we ask for a partition M such that $\sum_{i=1}^{s} w_i(e) = w(e), \forall e \in E$, and $W = \sum_{i=1}^{s} w_i$ is minimized. We shall refer to this variant as preemptive MEC (p-MEC) problem.

The existence of a setup delay, d, encounter in practical applications to

establish each matching (configuration), does not make any difference for the MEC problem: this can be taken into account by increasing the weight of all edges of G by d. Thus, the weight of each matching in M will be also increased be d, incorporating the set-up delay for this matching.

On the contrary, the presence of such a parameter affects crucially the complexity of the p-MEC problem. In the absence of d the p-MEC problem is equivalent to the preemptive open shop scheduling problem which can be solved optimally in polynomial time [12]. However, in the presence of d the p-MEC problem becomes strongly NP-hard [8] and non approximable within a factor less than 7/6 [3]. Approximation algorithms of factors 2 and $2 - \frac{1}{d+1}$ have been presented in [1] and [3], respectively.

The analogous, to the MEC problem, Max-weight Vertex Coloring (MVC) problem has been studied more extensively in the literature during last years [2, 7, 5, 4, 6, 14, 13]. In the MVC problem we ask for a partition of the vertices of G into independent sets (colors), each one of weight equal to the maximum weight of its vertices, so that the total weight of the partition is minimized.

Note that the MEC problem, on a general graph G, is equivalent to the MVC problem on the line graph, $L(G)$, of G and thus any algorithm for the MVC problem applies also to the MEC problem. However, this is not true for special graph classes, since the line graph of a special graph (e.g. bipartite or tree) is not anymore in the same special class.

3 Notation and Preliminaries

In the following we consider the MEC problem on a weighted graph $G = (V, E)$. By $d(v),\ v \in V$, we denote the degree of vertex v and by $\Delta(G)$ (or simply Δ) the maximum vertex degree of G. We define the degree of each edge $e(u, v) \in E$ as $d(u, v) = d(u) + d(v)$ and $\Delta'(G)$ (or simply Δ') denotes the maximum edge degree.

It is well known that the classical *edge coloring* problem, is NP-hard even in cubic graphs [9], although its optimal solution is either Δ or $\Delta + 1$ [16]. On the other hand, it is solvable in polynomial time for bipartite graphs [11]. Obviously, applying such an algorithm on a weighted bipartite graph we obtain a Δ-colors solution, in general non optimal, to the MEC problem. For the number of matchings in an optimal solution of the MEC problem the following bound holds.

Proposition 1 *For the number of matchings, s^*, in an optimal solution it holds that $\Delta \leq s^* \leq \Delta' - 1 \leq 2\Delta - 1$.*

Proof: Any solution consists of at least Δ matchings, since there is a vertex with exactly Δ adjacent edges which belong in different matchings.

Assume that an optimal solution consists of Δ' or more matchings. Consider those matchings sorted in non-increasing order of their weights. Each edge of G has at most $\Delta' - 2$ neighbor edges. So, for each edge e in any $(\Delta' + i)$-th matching, $i > 0$, there is one of the first $\Delta' - 1$ matchings where e can be moved without increasing the weight of this matching.

The last part of the inequality follows directly by the definition of Δ'. ■

The MEC problem is also polynomial for graphs of maximum degree $\Delta = 2$. This result follows from the same variant of the MVC problem. In [6] has been presented an $O(|V|^2)$ algorithm for the MVC problem on chains, which can be easily adapted for the MVC problem on graphs of maximum degree $\Delta = 2$ (collections of chains and cycles). Moreover, if G is a graph of maximum degree two, then its line graph $L(G)$ is also a graph with $\Delta(L(G)) = 2$ and the following theorem holds.

Theorem 1 *An optimal solution to the* MEC *problem for graphs of maximum degree $\Delta = 2$ consists of at most three (i.e., two or three) matchings and can be found in $O(|E|^2)$ time.*

In the rest of this paper we consider the edges of G sorted in non-increasing order of their weights, $w(e_1) \geq w(e_2) \geq \ldots \geq w(e_m)$. Thus, e_1 denotes the heaviest edge of G.

By OPT we denote the cost of an optimal solution to the MEC problem. We also assume that in such an optimal solution the graph is decomposed into s^* matchings each one of weight w_i^*. Without loss of generality we consider the matchings of any solution in non-increasing order of their weights, i.e. $w_1 \geq w_2 \geq \ldots \geq w_s$, and for the optimal solution $w_1^* \geq w_2^* \geq \ldots \geq w_{s^*}^*$.

4 Trees

In this section we first present a polynomial algorithm for a related decision problem called Feasible k-Coloring. This algorithm is then used to derive a polynomial algorithm for the MEC problem on trees of bounded degree.

The Feasible k-Coloring problem is formally defined as following. An analogous problem is also defined and solved in [13] for the MVC problem

on trees.

Feasible k-Coloring:
Instance: A tree $T(V, E)$, a weight function $w(e) : E \to \mathbb{N}$ and a sequence of k integer weights $a_1, a_2, \ldots, a_k$, such that $a_1 \geq a_2 \geq \ldots \geq a_k$.
Question: Is there a partition of the set of edges E into exactly k matchings $M_1, M_2, \ldots, M_k$, such that $w_j \leq a_j,\ 1 \leq j \leq k$?

We consider the tree T rooted at an arbitrary vertex, r. For each edge $e = (v, u)$ we define u to be the most distant from r endpoint of e, and $T(e)$ to be the subtree of T rooted at u. We denote by $S(e) \subseteq \{M_1, M_2, \ldots, M_k\}$ to be the set of matchings in which edge e can belong in order the subtree $T(e) \cup \{e\}$ to be feasibly colorable.

Our algorithm initializes the sets $S(e)$ for each leaf edge e to contain the matchings that are heaviest than its weight $w(e)$. Moreover, a fictive edge e_0 of weight $w(e_0) = 0$ is connected to the root of the tree, as in Figure 1.a, in order to treat the root of the tree as the rest vertices. The Feasible k-Coloring algorithm follows.

Algorithm 1

```
1. Initialization:
     Leaf edges:  S(e) = {M_j | 1 ≤ j ≤ k and w(e) ≤ a_j}
     Rest edges:  S(e) = {}
     Add a fictive vertex r',
     a fictive edge e_0 = (r', r) with w(e_0) = 0
     and a fictive matching M_0 with w_0 = a_0 = 0
2. For each e ∈ E ∪ {e_0} in post-order do
3.   For each matching M_j such that w(e) ≤ a_j do
4.     If there is coloring of T(e) ∪ {e} such that e ∈ M_j then
5.       S(e) = S(e) ∪ {M_j}
6.   If S(e) = ∅ then return
7. Create a feasible coloring using the S(e)'s
```

In line 4, Algorithm 1 decides if a feasible coloring for the subtree $T(e) \cup \{e\}$ exists (see Figure 1.b). For each edge $e = (v, u)$, we define $E_u = \{e_1^u, e_2^u, \ldots, e_d^u\}$ to be the set of edges from u to its children (recall that u is the most distant from the root of the tree endpoint of e). Each edge e_i^u can belong in one of the matchings in its $S(e_i^u)$. Let Q be the union of the sets $S(e_i^u)$, but the matching M_j edge e is assigned to, i.e. $Q = \bigcup_{i=1}^{d} S(e_i^u) - \{M_j\}$.

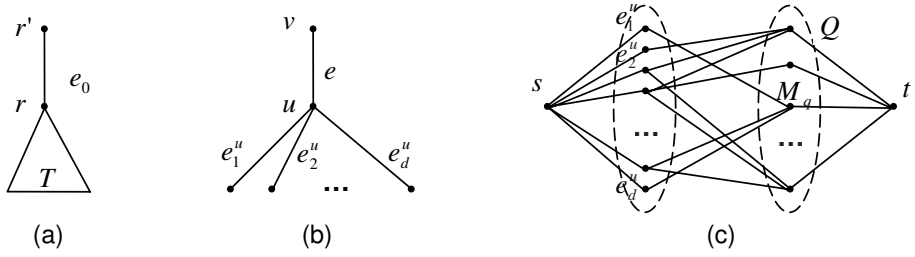

Figure 1: (a) The fictive edge e_0. (b) An instance of the graph in line 4 of Algorithm 1. (c) The flow network constructed in line 4 of Algorithm 1. All edges have weight equal to 1.

We create a bipartite graph $B(E_u, Q; S)$, where there is an edge between $e_i^u \in E_u$ and $M_q \in Q$ iff $M_q \in S(e_i^u)$. Then we create a flow network F by joining a source vertex s to each vertex in E_u and a terminal vertex t to each vertex in Q, as in Figure 1.c. We assign to all the edges in F a weight equal to 1. Then, it follows that there is coloring of $T(e) \cup \{e\}$ such that $e \in M_j$ iff there is in F an $s - t$ flow of value d.

In line 7, Algorithm 1 creates a partition of the edges of T into matchings. This can be done by considering the edges of T in pre-order and assigning an edge e in an arbitrary matching in its set $S(e)$.

Algorithm 1 performs $k \cdot |E|$ iterations and in each one of them runs a maximum flow algorithm of O(poly) time. Thus, the next Theorem follows.

Theorem 2 *There a polynomial time algorithm for the Feasible k-Coloring problem.*

Algorithm 1 can be used to solve the MEC problem on trees, as following.

Algorithm 2

1. `For` $k = \Delta$ `to` $\Delta' - 1$ `do`
2. `For all` $\binom{|E|}{k}$ `combinations of edge weights,`
   `such that` $w_1 \geq w_2 \geq \ldots \geq w_k$ `and` $w_1 = max\{w(e)|e \in E\})$ `do`
3. `Run Algorithm 1`
4. `Return the best of the solutions found`

Line 3 of Algorithm 2 is repeated $O(\Delta \cdot |E|^{\Delta})$ times, and therefore it is polynomial only for trees of polynomially bounded degree.

Theorem 3 *There a polynomial time algorithm for the* MEC *problem in trees with polynomially bounded degree.*

5 Stars of chains

A star consists of m edges $e_1, e_2, \ldots, e_m$ sharing a common endpoint. Obviously, the optimal solution to the MEC problem for such a weighted star is of cost $OPT = \sum_{i=1}^{m} w(e_i)$ and consists of exactly $\Delta = m$ matchings.

A star of chains consists of p chains $C_1, C_2, \ldots, C_p$ all starting from a common vertex, say u. We consider each chain $C_i,\ 1 \leq i \leq p$, starting from u with an edge e_i^u which we call start edge. We also assume w.l.o.g. that $w(e_1^u) \geq w(e_2^u) \geq \ldots \geq w(e_p^u)$.

Lemma 1 *For an optimal solution of the* MEC *problem on a star of chains the following hold:*
i) The number of matchings s^* equals to either p or $p+1$.
ii) Only $k \leq 3$ matchings have cardinality $|M_j| > 1$.
iii) At least the $k-1$ heaviest start edges appear in these k matchings.

Proof:
i) By Proposition 1, $\Delta \leq s^* \leq \Delta' - 1$. Here, $\Delta = p$ and $\Delta' = p + 2$.
ii) Assume that an optimal solution has more than three matchings of cardinality $|M_j| > 1$. Consider those matchings sorted in non-increasing order of their weights. Each non start edge e has at most 2 neighbor edges. So, such an edge e can be moved in one of the three first heaviest matchings, since its neighbor edges can belong in at most two different matchings.
iii) Consider first that $k = 2$. Assume that in the optimal solution the heaviest start edge e_1^u does not belong to neither of two matchings of cardinality $|M_j| > 1$. Then e_1^u can be either inserted in one of those two matchings (if this does not contain another start edge) or e_1^u can replace an existing start edge. In both cases the cost of the optimal solution decreases or remains the same.

Assume next that $k = 3$. As in the previous case e_1^u can be inserted in one of the three matchings of cardinality $|M_j| > 1$. Similarly, e_2^u can be inserted in one of the rest two of those matchings. ■

In the following we distinguish between two cases according to possible number of matchings in an optimal solution, i.e. $p+1$ or p.

If an optimal solution consists of $p+1$ matchings then it contains exactly one matching without any start edge. Algorithm 3 finds such an optimal schedule with $p+1$ matchings.

Algorithm 3

1. Remove from the star the $p-2$ lightest start edges
   (this creates a graph H consisting of $p-1$ chains)
2. Find an optimal solution $S^*(H)$ for the graph H,
   using Theorem 1
3. If there are 3 non empty matchings in $S^*(H)$ then
4. Return the solution consisting of these 3 matchings
   of $S^*(H)$ plus $p-2$ matchings each one containing
   one of the removed $p-2$ lightest start edges

Note that Algorithm 3 is possible to return $p-1$ matchings of $|M_i| = 1$, in the case where the one of the three matchings of $S^*(H)$ found in line 2 consists of a single edge. Taking into account Lemma 1, it follows that Algorithm 3 returns the optimal solution of $p+1$ matchings since: (i) the $p-(k-1)$ matchings of cardinality $|M_i| = 1$ contain the $p-(k-1)$ lightest start edges (one per each matching) and (ii) the cost of k matchings is optimal.

The complexity of Algorithm 3 is dominated by line 2 and by Theorem 1 it is $O(|E|^2)$.

If an optimal solution consists of p matchings, then each of them contains a start edge. Algorithm 4 returns such an optimal schedule with p matchings.

Algorithm 4

1. For $i = 3$ to p do
2. Remove $p-3$ start edges $e_3^u, e_4^u, \ldots, e_{i-1}^u, e_{i+1}^u, \ldots, e_p^u$
   (this creates a star T of 3 chains
   and a graph H of $p-3$ chains)
3. Find the optimal solution $S^*(T)$ using Theorem 3
4. If there are exactly 3 matchings in $S^*(T)$ then
5. Find the optimal solution $S^*(H)$ using Theorem 1
6. Combine the solutions $S^*(T)$ and $S^*(H)$
   into exactly 3 matchings
7. Find a solution for the initial star consisting of
   these 3 matchings plus $p-3$ matchings each one
   containing one of the removed $p-3$ start edges
8. Return the best solution found

Algorithm 2 is used in line 3 since T is a bounded degree tree with $\Delta = 3$ and it returns an optimal solution $S^*(T)$ of at least three matchings. In line

6, a 3-matchings optimal solution for the edges in T and H can be obtained by considering the matchings in both solutions in non-increasing order of their weights and merging the matchings of each solution having the same rank, since T and H are vertex-disjoint. The optimality of the solution that Algorithm 4 returns follows by Lemma 1 using the same arguments as for Algorithm 3.

The complexity of the algorithm is dominated by line 3 which takes $O(|E|^3)$ time and is executed $\Delta - 2$ times.

The optimal solution to the MEC problem on stars of chains is the best between the solution found by Algorithm 3 (optimal with $p+1$ matchings) and the one found by Algorithm 4 (optimal with p matchings), since, according to Lemma 1, the optimal solution consists of either $p+1$ or p matchings. Thus, the following theorem holds.

Theorem 4 *The* MEC *problem on stars of chains can be solved optimally in* $O(\Delta \cdot |E|^3)$ *time.*

6 Bipartite graphs

In this section we present an algorithm, denoted by $\mathcal{A}(G)$, that improves the 2 approximation ratio given in [10] for the MEC problem in bipartite graphs of maximum degree $\Delta \leq 7$. The main idea of our algorithm is the following: for a given bipartite graph G, of maximum degree Δ, run $\Delta - 1$ algorithms, $A_\Delta, A_{\Delta+1}, \ldots, A_{2\Delta-1}$, and select the best solution found. Each $A_{\Delta+k}$, $0 \leq k \leq \Delta - 1$, algorithm splits the graph G into two subgraphs G_1 and G_2 such that the graph G_1 contains heavy edges and has maximum degree $\Delta(G_1) \leq k$. Note that in general $\Delta(G_2) \leq \Delta(G)$. Given this splitting of G each algorithm $A_{\Delta+k}$ returns a solution of cost $W_{\Delta+k}$, by concatenating the following solutions of the MEC problem on G_1 and G_2: (i) for the graph G_2 the algorithm finds a Δ-matchings solution by solving the classical edge coloring problem on G_2, (ii) for the graph G_1 the algorithm finds a solution of at most $2\Delta(G_1) - 1$ matchings using (recursively) algorithm $\mathcal{A}(G_1)$.

For $k = 0$, G_2 coincides with G and therefore algorithm A_Δ returns a Δ-matchings solution found as in point (i) above. For the weight of such a solution it holds that $W_\Delta \leq \Delta \cdot w_1^*$, since for the weight, w_i, of any matching of this solution it holds that $w_i \leq w(e_1) = w_1^*$.

For $k = 1$, G_1 is a maximal matching of G created by examining its edges in non-increasing order of their weights. This matching is of weight $w_1 = w(e_1)$. Algorithm $A_{\Delta+1}$ uses also algorithm $A_\Delta(G_2)$. The cost of

the solution that algorithm $A_{\Delta+1}$ returns is $W_{\Delta+1} \leq w_1^* + \Delta \cdot w_2^*$, since no edge e of weight $w(e) > w_2^*$ belongs to G_2 (if such an edge belongs to G_2, then it is not in M_1 because a heavier one of its adjacent edges is in M_1, a contradiction).

In general, algorithm $A_{\Delta+k},\ k \geq 2$, repeatedly splits the graph G into graphs G_1 and G_2, with G_1 containing edges heavier than a parameter c, taking as values the weights of the edges of the graph G. The algorithm returns the best of the solutions found in these iterations.

Algorithm 5 - $A_{\Delta+k}$

`1. For each weight` $c = w(e_1), w(e_2), \ldots$ `do`
`2. Split` G `into edge induced subgraphs` $G_1 = \{e|w(e) > c\}$ `and` $G_2 = \{e|w(e) \leq c\}$
`3. If` $\Delta(G_1) \leq k$ `then`
`4. If` $\Delta(G_1) = 2$ `then find an optimal solution for the graph` G_1
`5. Else run` $\mathcal{A}(G_1)$
`6. Use` A_Δ `for` G_2
`7. Return the best solution found`

The case of $k = 2$ is treated in a different way than the cases of $k \geq 3$, since in this case $\Delta(G_1) = 2$, and an optimal solution for G_1 can be found by Theorem 1. Thus, for $k = 2$ we obtain

Lemma 2 *Algorithm $A_{\Delta+2}$ returns a solution of cost $W_{\Delta+2} \leq w_1^* + w_2^* + \Delta \cdot w_3^*$.*

Proof: $A_{\Delta+2}$ returns the best solution found with weight $W_{\Delta+2} = \min\{w_1 + w_2 + w_3 + \Delta \cdot w_4\}$ for different values of $c = w_4$. Consider the cost of the solution found by the algorithm in the iteration for which $w_4 = w_3^*$. In this iteration the edges of G_1 are a subset of the edges in the two heaviest matchings of the optimal solution. Therefore for this iteration it holds that $W_{\Delta+2} \leq w_1^* + w_2^* + \Delta \cdot w_3^*$, since $\Delta(G_1) = 2$ and thus the optimal solution for G_1 can be found by Theorem 1. ■

Using algorithms A_Δ, $A_{\Delta+1}$ and $A_{\Delta+2}$, algorithm $\mathcal{A}(G)$ returns a solution of cost $W = \min\{W_\Delta, W_{\Delta+1}, W_{\Delta+2}\}$, that is

$$W \leq \Delta \cdot w_1^*$$
$$W \leq w_1^* + \Delta \cdot w_2^*$$
$$W \leq w_1^* + w_2^* + \Delta \cdot w_3^*$$

Multiplying these inequalities by z_1, z_2 and z_3, respectively, and adding them we obtain

$$W \cdot (z_1 + z_2 + z_3) \leq w_1^* \cdot (z_1 \cdot \Delta + z_2 + z_3) + w_2^* \cdot (z_2 \cdot \Delta + z_3) + w_3^* \cdot (z_3 \cdot \Delta)$$

$$\frac{W}{OPT} \leq \frac{t}{z_1 + z_2 + z_3}$$

where $t = z_1 \cdot \Delta + z_2 + z_3 = z_2 \cdot \Delta + z_3 = z_3 \cdot \Delta$. Thus, the best approximation ratio is achieved when the quantity $\frac{t}{z_1+z_2+z_3}$ is minimized. Therefore, we want to:

$$\min \left\{ \frac{t}{z_1 + z_2 + z_3} \right\}$$

such that

$$\begin{aligned} &(z_1 \cdot \Delta + z_2 + z_3) = t \\ &(z_2 \cdot \Delta + z_3) = t \\ &(z_3 \cdot \Delta) = t \\ &z_1, z_2, z_3 \geq 0 \end{aligned}$$

Solving this problem we obtain a new approximation ratio for the MEC problem in bipartite graphs with $\Delta \geq 3$, which is:

$$\frac{W}{OPT} \leq \frac{\Delta^3}{3\Delta^2 - 3\Delta + 1} = \varrho_{\Delta+2}$$

This ratio is unbounded, but gives better results than the approximation ratio $(2\Delta - 1)/3$ given in [5], for any Δ, as well as than the 2-approximation algorithm given in [10], for $\Delta = 3$ and $\Delta = 4$. In fact, $\varrho_{\Delta+2}$ becomes $27/19 \simeq 1.42$ for $\Delta = 3$ and $64/37 \simeq 1.73$ for $\Delta = 4$. Note, however, that an 1.17-approximation algorithm is known [4] for bipartite graphs with $\Delta = 3$.

The behavior of algorithm $\mathcal{A}(G)$ for bipartite graphs of $\Delta \geq 4$, is improved if one allows the use of algorithms $A_{\Delta+3}$, $A_{\Delta+4}$ and so on. In fact, for $k = 3$ we have:

Lemma 3 *Algorithm $A_{\Delta+3}$ returns a solution of cost $W_{\Delta+3} \leq \varrho_{\Delta+2} \cdot (w_1^* + w_2^* + w_3^*) + \Delta \cdot w_4^*$.*

Proof: Similarly to the proof of Lemma 2, in some iteration of algorithm $A_{\Delta+3}$, the maximum edge in G_2 is equal to w_4^* and the edges of G_1 are a subset of the edges in the three heaviest matchings of the optimal solution. As $\Delta(G_1) = 3$, the algorithm $\mathcal{A}(G_1)$ return a $\varrho_{\Delta+2}$ approximate solution for

the graph G_1. ■

Working as above we obtain that algorithm $\mathcal{A}(G)$, for bipartite graphs with $\Delta \geq 4$, leads to an approximation ratio

$$\varrho_{\Delta+3} = \frac{19\Delta^4}{76\Delta^3 - 138\Delta^2 + 100\Delta - 27}.$$

This ratio improves the result for $\Delta = 4$, where the ratio becomes 1.61 from 1.73. Moreover, for $\Delta = 5$ this ratio becomes 1.82.

In general, the algorithm $\mathcal{A}(G)$ gives a better approximation ratio than 2 for bipartite graphs with $\Delta(G) \leq 7$. The following table summarizes the best approximation ratio achieved by our algorithm and the previous best known ratio for different values of Δ.

Δ	3	4	5	6	7	8
our ratio	1.42	1.61	1.75	1.86	1.95	2.03
previous ratio	1.17	2	2	2	2	2

In general, the complexity of algorithm $\mathcal{A}(G)$ is dominated by the complexity of $A_{2\Delta(G)-1}$, which calls recursively at most $|E|$ times algorithm $\mathcal{A}(G_1)$, where $\Delta(G_1) \leq \Delta(G) - 1$. The recursion depth is at most $\Delta - 2$, since the recursion stops in $A_{\Delta+2}$. Each iteration of each $A_{\Delta+k}$ algorithm, $0 \leq k \leq \Delta - 1$, calls algorithm A_Δ, which runs in polynomial time. Thus, algorithm A_Δ is called $O(|E|^{\Delta(G)-2})$ times, in total, by algorithm $\mathcal{A}(G)$.

Remark: Concerning general graphs, note that an edge coloring using $(\Delta+1)$ matchings can be found in polynomial time. Thus, modifying algorithm A_Δ to find such a coloring of $\Delta+1$ (instead of Δ) matchings, algorithm $\mathcal{A}(G)$ works also for general graphs. Furthermore, it beats the 2-approximation algorithm in [10] for graphs of $\Delta = 3$ and $\Delta = 4$, achieving ratios 1.73 and 1.93, respectively.

References

[1] F. N. Afrati, T. Aslanidis, E. Bampis, and I. Milis. Scheduling in switching networks with set-up delays. *J. Combinatorial Optimization*, 9(1):49–57, 2005.

[2] P. Brucker, A. Gladky, H. Hoogeveen, M. Koyalyov, C. Potts, T. Tautenham, and S. van de Velde. Scheduling a batching machine. *J. Scheduling*, 1(1):31–54, 1998.

[3] P. Crescenzi, X. Deng, and C. H. Papadimitriou. On approximating a scheduling problem. *J. Combinatorial Optimization*, 5(3):287–297, 2001.

[4] D. de Werra, M. Demange, B. Escoffier, J. Monnot, and V. Th. Paschos. Weighted coloring on planar, bipartite and split graphs: complexity and improved approximation. In Rudolf Fleischer and Gerhard Trippen, editors, *Proc. International Symposium on Algorithms and Computation, ISAAC'04*, volume 3341 of *Lecture Notes in Computer Science*, pages 896–907. Springer-Verlag, 2004.

[5] M. Demange, D. de Werra, J. Monnot, and V. Th. Paschos. Weighted node coloring: when stable sets are expensive. In L. Kučera, editor, *Proc. International Workshop on Graph Theoretical Concepts in Computer Science, WG'02*, volume 2573 of *Lecture Notes in Computer Science*, pages 114–125. Springer-Verlag, 2002.

[6] B. Escoffier, J. Monnot, and V. Th. Paschos. Weighted coloring: further complexity and approximability results. *Inform. Process. Lett.*, 97(3):98–103, 2006.

[7] G. Finke, V. Jost, M. Queyranne, and A. Sebő. Batch processing with interval graph compatibilities between tasks. Technical report, Cahiers du laboratoire Leibniz, 2004. Available at `http://www-leibniz.imag.fr/NEWLEIBNIZ/LesCahiers/index.xhtml`.

[8] I. S. Gopal and C. Wong. Minimizing the number of switchings in a SS/TDMA system. *IEEE Transactions On Communications*, 33(6):497–501, 1985.

[9] I. Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718–720, 1981.

[10] A. Kesselman and K. Kogan. Non-preemptive scheduling of optical switches. In *Proc. IEEE Global Telecommunications Conference, GLOBECOM'04*, volume 3, pages 1840–1844, 2004.

[11] D. König. ber graphen und iher anwendung auf determinantentheorie und mengenlehre. *Math. Ann.*, 77:453–465, 1916.

[12] E. L. Lawler and J. Labetoulle. On preemptive scheduling of unrelated parallel processors by linear programming. *J. Assoc. Comput. Mach.*, 25(4):612–619, 1978.

[13] S. V. Pemmaraju and R. Raman. Approximation algorithms for the max-coloring problem. In *Proc. ICALP'05*, number 3580 in Lecture Notes in Computer Science, pages 1064–1075. Springer-Verlag, 2005.

[14] S. V. Pemmaraju, R. Raman, and K. R. Varadarajan. Buffer minimization using max-coloring. In *Proc. Symposium on Discrete Algorithms, SODA'04*, pages 562–571, 2004.

[15] F. Rendl. On the complexity of decomposing matrices arising in satellite communication. *Oper. Res. Lett.*, 4(1):5–8, 1985.

[16] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz.*, 3:25–30, 1964.