

# On a Generalized Graph Coloring/Batch Scheduling Problem

Giorgio Lucarelli<sup>1</sup>, Ioannis Milis

Dept. of Informatics, Athens University of Economics and Business, 104 34, Athens, Greece, {gluc, milis}@aueb.gr

Vangelis Th. Paschos<sup>2</sup>

LAMSADE, Université Paris-Dauphine, 75016 Paris, France, paschos@lamsade.dauphine.fr

---

**Abstract:** We study a batch scheduling problem where jobs are no more independent but they are subject to (in)compatibility constraints described by an underlying graph. The problem is equivalent to a generalized weighted graph coloring problem. We study the frontier between polynomial and NP-variants of the problem as well as the approximability of NP-hard variants.

*Keywords:* Algorithmics, Batch Scheduling

---

## 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 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. 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 to be minimized.

This problem is known as Time Slot Scheduling and it is equivalent to the parallel batch scheduling problem with (in)compatibilities between jobs, which is denoted as  $1 \mid p\text{-batch, graph} \mid C_{max}$ . Jobs are no more independent but they correspond to the edges of a weighted *graph*. Edge weights correspond to processing (transmission) times of jobs and the graph  $G$  describes (in)compatibilities between jobs: jobs corresponding to adjacent edges cannot be scheduled in the same batch (configuration).

The problem can be also formulated in graph-theoretic terms with senders and receivers corresponding to the vertices  $V$  of a graph  $G(V, E)$ , messages to its edges  $E$ , the lengths of messages to the weights of edges  $w(e)$  and configurations to matchings. In this context, we ask for a partition  $M = \{M_1, M_2, \dots, M_s\}$  of the set of edges of the graph  $G$  into matchings, each one of weight  $w(M_i) = \max\{w(e) \mid e \in M_i\}$ , such that  $w(M) = \sum_{i=1}^s w(M_i)$  is minimized. In the version we study in this paper preemptions are not allowed. In the following we shall refer to this problem as Batch Scheduling-Edge Coloring (BS-EC) problem.

It is known that the BS-EC problem is strongly NP-hard and  $7/6$ -inapproximable even for cubic planar bipartite graphs and edge weights restricted to be 1, 2 or 3 [6, 9]. On the other hand, a 2-approximation algorithm has been presented in [9] for bipartite graphs, which, in fact, applies also for general graphs. In addition, a  $\frac{2\Delta-1}{3}$ -approximation algorithm, for bipartite graphs

---

<sup>1</sup>This research project is co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%).

<sup>2</sup>Part of this work has been carried out while the author was visiting the Department of Informatics of Athens University of Economics and Business.

of maximum degree  $\Delta$ , has been presented in [5], which gives an approximation ratio of  $5/3$  for  $\Delta = 3$ . A new algorithm for bipartite graphs of  $\Delta = 3$ , presented in [6], achieves an approximation ratio equal to the  $7/6$ -inapproximability result. Furthermore, an algorithm presented in [7] achieves approximation ratios less than two for bipartite graphs with  $4 \leq \Delta \leq 7$ . The preemptive variant of the BS-EC problem on bipartite graphs has been also studied [4, 1, 3].

Moreover, a large body of work is concentrated on the analogous Batch Scheduling-Vertex Coloring (BS-VC) [2, 8, 5, 6, 7, 10, 11]. In this problem jobs correspond to the weighted vertices of a given graph  $G$  and adjacent jobs (vertices) cannot be scheduled in the same batch. In graph theoretic terms we ask for a partition of the vertices of  $G$  into independent sets, each one of weight equal to the maximum weight of its vertices, so that the total weight of the partition is minimized. It is clear that the BS-EC problem on a general graph  $G$  is equivalent to the BS-VC problem on the line graph,  $L(G)$ , of  $G$  and thus any algorithm for the BS-VC problem applies also to the BS-EC 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.

In this paper we study the BS-EC problem. In the next section we give the notation we use and some preliminaries. As the complexity of the problem on trees remains open, in Section 3, we present a polynomial algorithm for trees of bounded degree and a polynomial algorithm for stars of chains. Finally in Section 4 we present an approximation algorithm for bipartite graphs which beats the known ones for bipartite graphs of maximum degree  $\Delta \leq 12$ .

## 2 Preliminaries

In the following we consider the non-preemptive BS-EC problem on an incompatibility weighted graph  $G = (V, E)$ . By  $d(v)$ ,  $v \in V$ , we denote the degree of vertex  $v$  and by  $\Delta(G)$  (or simply  $\Delta$ ) 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  $\Delta'$ ) denotes the maximum edge degree. Moreover, we consider the edges of  $G$  sorted in non-increasing order of their weights,  $w(e_1) \geq w(e_2) \geq \dots \geq w(e_m)$ . Thus,  $e_1$  denotes the heaviest edge of  $G$ .

By  $OPT$  we denote the cost of an optimal solution to the BS-EC 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.  $w(M_1) \geq w(M_2) \geq \dots \geq w(M_s)$ , and for the optimal solution  $w_1^* \geq w_2^* \geq \dots \geq w_{s^*}^*$ .

**Proposition 1** *For the number of matchings,  $s^*$ , in an optimal solution of the BS-EC problem on a graph  $G$ , it holds that  $\Delta \leq s^* \leq \Delta' - 1 \leq 2\Delta - 1$ .*

**Proof:** Obviously, any solution consists of at least  $\Delta$  matchings. Assume that an optimal solution consists of  $\Delta'$  or more matchings. As any edge of  $G$  has at most  $\Delta' - 2$  neighbor edges, it follows that 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 (recall that matchings are considered in non-increasing order of their weights). ■

In the case where all edges have the same length the BS-EC problem is equivalent to the classic *edge coloring* problem, where the objective is to minimize the number of colors (matchings) required in order to assign different colors to neighbor edges. It is well known that this problem is NP-hard in general graphs, but it is solvable in polynomial time in bipartite graphs, using as many colors as the maximum degree of the graph. Applying such an algorithm for a weighted bipartite graph we obtain a  $\Delta$ -matchings solution, in general non optimal, to the BS-EC problem.

The BS-EC problem is also polynomial for graphs of maximum degree  $\Delta = 2$ . This result follows from the same variant of the BS-VC problem. In [7] has been presented an  $O(|V|^2)$  algorithm for the BS-VC problem on chains, which can be easily adapted for the BS-VC problem on graphs of maximum degree  $\Delta = 2$  (collections of chains and cycles). On the other hand, if  $G$  is a graph of maximum degree two then its line graph  $L(G)$  is also a graph with  $\Delta(L(G)) = 2$ .

**Theorem 1** *An optimal solution to the BS-EC 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.*

### 3 Trees

As the complexity of the BS-EC problem on trees remains open, 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 BS-EC problem on trees of bounded degree. We also present a polynomial algorithm for stars of chains.

#### 3.1 Feasible $k$ -Coloring and bounded degree trees

The Feasible  $k$ -Coloring problem is formally defined as following. An analogous problem is also defined and solved in [11] for the BS-VC problem on trees.

**Feasible  $k$ -Coloring:**

**Instance:** A tree  $T(V, E)$ , a weight function  $w(e) : E \rightarrow \mathbb{N}$  and a sequence of  $k$  integer weights  $a_1, a_2, \dots, a_k$ , such that  $a_1 \geq a_2 \geq \dots \geq a_k$ .

**Question:** Is there a partition of the edges  $E$  of  $T$  into exactly  $k$  matchings  $M_1, M_2, \dots, M_k$ , such that  $w(M_j) \leq a_j$ ,  $1 \leq j \leq k$ ?

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  $S(e) \subseteq \{M_1, M_2, \dots, M_k\}$  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:  $S(e) = \{M_j | 1 \leq j \leq k \text{ and } w(e) \leq a_j\}$

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  $w(M_0) = a_0 = 0$

2. For each  $e \in E \cup \{e_0\}$  in post-order do

3. For each matching  $M_j$  such that  $w(e) \leq a_j$  do

4. If there is coloring of  $T(e) \cup \{e\}$  such that  $e \in M_j$  then  $S(e) = S(e) \cup \{M_j\}$

5. If  $S(e) = \emptyset$  then return

6. Create a feasible coloring using the  $S(e)$ 's

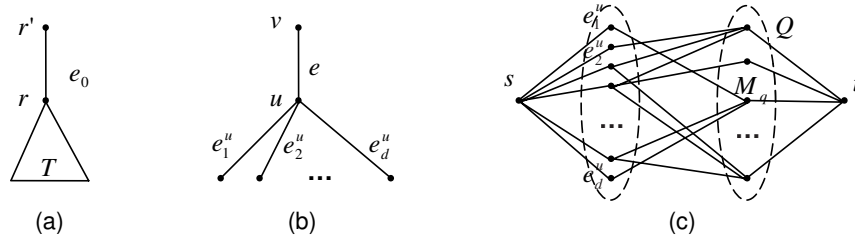


Figure 1: (a) The fictive edge  $e_0$ . (b) An instance of the graph in line 4 of Algorithm 1. (c) The flow network constructed in line 4 of Algorithm 1. All edges have weight equal to 1.

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  $E_u = \{e_1^u, e_2^u, \dots, e_d^u\}$  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.  $Q = \bigcup_{i=1}^d S(e_i^u) - \{M_j\}$ . 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 6, Algorithm 1 creates a partition of the edges of  $T$  into matchings, by assigning each edge  $e$  of  $T$ , in pre-order, to 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 polynomial complexity (e.g. Goldberg and Tarjan's algorithm of complexity  $O((|V| + k)^3)$ ). 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 BS-EC 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 run Algorithm 1
3. Return the best of the solutions found

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

**3.2 Stars of chains**

A star of chains consists of  $p$  chains  $C_1, C_2, \dots, C_p$  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 assume also that  $w(e_1^u) \geq w(e_2^u) \geq \dots \geq w(e_p^u)$ .

**Lemma 1** *For an optimal solution of the BS-EC 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.
- 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  $|M_j| > 1$  and  $e_2^u$  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  $e_3^u, e_4^u, \dots, e_{i-1}^u, e_{i+1}^u, \dots, e_p^u$   
(this creates a star  $T$  of 3 chains and a graph  $H$  of  $p - 3$  chains)
3. Find the optimal solution  $S^*(T)$  using Algorithm 2
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. 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.

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

## 4 Bipartite graphs

In this section we present an algorithm which improves the best known approximation ratios for the BS-EC problem in bipartite graphs of maximum degree  $4 \leq \Delta \leq 12$ .

Our algorithm generalizes the idea behind the  $7/6$ -approximation algorithm for bipartite graphs of  $\Delta = 3$ , proposed in [6], to general bipartite graphs. In fact, our algorithm splits repeatedly a given bipartite graph  $G$ , of maximum degree  $\Delta$ , into three edge induced subgraphs. To describe this partition as well as our algorithms let us introduce some additional notation. Recall that we consider the edges of  $G$  sorted in non-increasing order of their weights i.e.,  $w(e_1) \geq w(e_2) \geq \dots \geq w(e_m)$ . For this order of edges we denote by  $G_{j,k}$ ,  $j \leq k$ , the subgraph of  $G$  induced by the edges  $e_j, e_{j+1}, \dots, e_k$ . By  $\Delta_{j,k}$  we denote the maximum degree of graph  $G_{j,k}$ . By convention, we define  $G_{m+1,m}$  as an empty graph. By  $j_q$  we denote the maximum index such that  $\Delta_{1,j_q} = q$ . It is clear that  $j_1 < j_2 < \dots < j_\Delta = m$ .

In general our algorithm examines, for each pair of indices  $j, k$ ,  $j = 1, 2, \dots, j_{\Delta-1}$ ,  $k = j + 1, \dots, m$ , a partition of graph  $G$  into three edge induced subgraphs: the graph  $G_{1,j}$ , induced by the  $j$  heaviest edges of  $G$ , the graph  $G_{j+1,k}$ , induced by the next  $k - j$  edges of  $G$ , and the graph  $G_{k+1,m}$ , induced by the  $m - k$  lightest edges of  $G$ . For each one of these partitions the algorithm checks the existence of two different specific matchings in graph  $G_{j+1,k}$ . For each one of these matchings, if exists, a solution for the BS-EC problem on graph  $G$  is obtained. Finally, the algorithm returns the best among all the solutions found.

### Algorithm 5 (G)

1. For  $j = 1, 2, \dots, j_{\Delta-1}$  do
2.   For  $k = j + 1$  to  $m$  do
3.     If there is a matching  $M$  in  $G_{j+1,k}$  such that each vertex in  $G_{1,k}$  of degree  $\Delta$  is saturated
4.     Create a solution for  $G_{1,k}$  by concatenating a solution of  $\Delta - 1$  matchings for  $G_{1,k} - M$  and the matching  $M$
5.     Complete greedily this solution with the edges of  $G_{k+1,m}$
6.     If  $j \leq j_2$  then find an optimal solution  $S_{1,j}$  for  $G_{1,j}$  using Theorem 1
7.     else find a solution  $S_{1,j}$  for  $G_{1,j}$  by calling Algorithm 5 ( $G_{1,j}$ )
8.     If there is a matching  $M'$  in  $G_{j+1,k}$  such that each vertex in  $G_{j+1,k}$  of degree  $\Delta$  is saturated and  $M'$  fits in  $S_{1,j}$
9.     Create a solution  $S_{j+1,k}$  with  $\Delta - 1$  matchings for  $G_{j+1,k} - M'$
10.    Concatenate  $S_{1,j}$  and  $S_{j+1,k}$  and complete greedily this solution with the edges of  $G_{k+1,m}$
11. Return the best solution found in Steps 5 and 10

In Lines 5 and 10 the algorithm completes a partial solution by examining the remaining lightest edges one by one and assigning them to the first matching where they fit. If such a matching does

not exist then a new one is created. As both partial solutions consist of at most  $2\Delta - 1$  matchings, the complete solutions obtained will consist also of at most  $2\Delta - 1$  matchings, by the arguments in the proof of Proposition 1.

The following lemma bounds the weight  $W$  of the solution obtained by Algorithm 5. By  $\varrho_q$  we define the approximation ratio of our algorithm for a graph with maximum degree  $q$ . By definition,  $\varrho_1 = \varrho_2 = 1$ , since for graphs of maximum degree 1 or 2 the BS-EC problem can be solved in polynomial time. Recall that, w.l.o.g., we consider the matchings of an optimal solution in non-increasing order of their weights, i.e.,  $w_1^* \geq w_2^* \geq \dots \geq w_{s^*}^*$ .

**Lemma 2** *Algorithm 5 returns a solution of cost:*

$$W \leq \min \left\{ (\Delta - 1) \cdot w_1^* + w_\Delta^* + (\Delta - 1) \cdot w_{\Delta+1}^*, \min_{1 \leq q \leq \Delta-1} \left\{ \varrho_q \cdot \sum_{i=1}^q w_i^* + (\Delta - 1) \cdot w_{q+1}^* + (\Delta - q) \cdot w_{\Delta+q}^* \right\} \right\}$$

**Proof:**

For the first term in the lemma's inequality consider the solution obtained by the Lines 3–5 of the algorithm in the iteration  $(j, k)$  where  $w(e_{j+1}) = w_\Delta^*$  and  $w(e_{k+1}) = w_{\Delta+1}^*$  (note that if  $k = m$  the algorithm examines the case where  $w_{\Delta+1}^* = 0$ ). In this iteration the matching  $M$  exists, since in the optimal solution the edges of  $G_{1,k}$  belong in at most  $\Delta$  matchings and the  $\Delta$ -th matching contains edges of weight at most  $w(e_{j+1})$ . The weight of  $M$  is at most  $w_\Delta^*$ , while the weight of the solution found for  $G_{1,k} - M$  is bounded by  $(\Delta - 1) \cdot w_1^*$ . The greedy step in Line 5 creates at most  $\Delta - 1$  matchings, each one of weight at most  $w_{\Delta+1}^*$ . Therefore,  $W \leq (\Delta - 1) \cdot w_1^* + w_\Delta^* + (\Delta - 1) \cdot w_{\Delta+1}^*$ .

For the second term in the lemma's inequality, consider the solutions obtained by the Lines 6–10 of the algorithm for  $\Delta - 1$  different iterations  $(i, k)$ . For  $j \leq j_q$ , consider the iteration  $k$  where  $w(e_{j+1}) = w_{q+1}^*$  and  $w(e_{k+1}) = w_{\Delta+q}^*$ . In this iteration, the matching  $M'$  exists, since in the optimal solution the edges of  $G_{j+1,k}$  belong in at most  $\Delta - 1$  matchings. If  $q = 1$  or  $2$ , the algorithm creates an optimal solution for  $G_{1,j}$  using Theorem 1, since  $\Delta_{1,j} \leq 2$ . If  $q \geq 3$ , the algorithm creates an approximation solution for  $G_{1,j}$ . In both cases, the edges in  $G_{1,j}$  are a subset of the edges of the  $q$  heaviest matchings in the optimal solution. Thus, it holds that  $S_{1,j} \leq \varrho_q \cdot \sum_{i=1}^q w_i^*$ , where  $\varrho_1 = \varrho_2 = 2$ . The weight of the solution found for  $G_{j+1,k} - M'$  is bounded by  $(\Delta - 1) \cdot w_{q+1}^*$ , since  $\Delta(G_{j+1,k} - M') \leq \Delta - 1$  and for each edge  $e \in G_{j+1,k} - M'$  it holds that  $w(e) \leq w_{q+1}^*$ . The greedy step in Line 10 creates at most  $\Delta - q$  matchings, each one of weight at most  $w_{\Delta+q}^*$ . Summing up the costs of these three partial solutions the lemma follows. ■

The Algorithm 5 performs  $O(|E|^2)$  iterations. The recursion in Line 7 is of depth 1, since for each partition  $j', k'$  of the graph  $G_{1,j}$  the solution for the graph  $G_{1,j'}$  is already computed in some previous iteration. The matchings  $M$  and  $M'$  can be computed in polynomial time by generalizing the ideas proposed in [6] for bipartite graphs of  $\Delta = 3$ .

For  $\Delta = 4$ , Algorithm 5 returns a solution of cost  $W$ , for which holds that:

$$\begin{aligned} W &\leq 3 \cdot w_1^* + w_4^* + 3 \cdot w_5^* \\ W &\leq w_1^* + 3 \cdot w_2^* + 3 \cdot w_5^* \\ W &\leq w_1^* + w_2^* + 3 \cdot w_3^* + 2 \cdot w_6^* \\ W &\leq 7/6 \cdot (w_1^* + w_2^* + w_3^*) + 3 \cdot w_4^* + w_7^* \end{aligned}$$

Multiplying these inequalities by  $44/458$ ,  $66/458$ ,  $99/458$  and  $138/458$ , respectively, and adding them we obtain a ratio  $\varrho_4 = 458/347 \simeq 1.32$  for the BS-EC problem in graphs with maximum degree  $\Delta = 4$ . In a same way, we can compute the ratio  $\varrho_\Delta$  for different values of  $\Delta$ .

The following table summarizes the approximation ratio achieved by Algorithm 5 as well as the previous best known ratio for  $\Delta = 3, 4, \dots, 12$ .

$\Delta$	3	4	5	6	7	8	9	10	11	12
previous ratio	1.17 <sup>[6]</sup>	1.61 <sup>[7]</sup>	1.75 <sup>[7]</sup>	1.86 <sup>[7]</sup>	1.95 <sup>[7]</sup>	2 <sup>[9]</sup>	2 <sup>[9]</sup>	2 <sup>[9]</sup>	2 <sup>[9]</sup>	2 <sup>[9]</sup>
our ratio	1.17	1.32	1.45	1.56	1.65	1.74	1.81	1.87	1.93	1.98

Algorithm 5 gives better results than the approximation ratio  $(2\Delta - 1)/3$  given in [5], for any  $\Delta$ , as well as than the 2-approximation algorithm given in [9], for bipartite graphs of  $\Delta \leq 12$ . Furthermore, for bipartite graphs of  $4 \leq \Delta \leq 7$  our algorithm improves the best known results given in [7]. For bipartite graphs of maximum degree  $\Delta = 3$ , Algorithm 5 reduces to the 7/6-approximation algorithm proposed in [6].

## References

- [1] F. Afrati, T. Aslanidis, E. Bampis, I. Milis (2005), Scheduling in Switching Networks with Set-Up Delays, *Journal of Combinatorial Optimization* **1**, 49 – 57.
- [2] P. Brucker, A. Gladky, H. Hoogeveen, M. Koyalyov, C. Potts, T. Tautenham, S. van de Velde (1998), Scheduling a Batching Machine, *Journal of Scheduling* **1**, 31 – 54.
- [3] J. Cohen, E. Jeannot, N. Padoy, F. Wagner (2006), Messages Scheduling for Parallel Data Redistribution between Clusters, *IEEE Trans. on Parallel and Distributed Systems* **17**, 1163 – 1175.
- [4] P. Crescenzi, X. Deng, Ch. H. Papadimitriou (2001), On Approximating a Scheduling Problem, *Journal of Combinatorial Optimization* **3**, 287 – 297.
- [5] M. Demange, D. de Werra, J. Monnot, V. Th. Paschos (2002), Weighted Node Coloring: When Stable Sets Are Expensive, In: *Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, 114 – 125.
- [6] D. de Werra, M. Demange, B. Escoffier, J. Monnot, V. Th. Paschos (2004), Weighted Coloring on Planar, Bipartite and Split Graphs: Complexity and Improved Approximation, In: *International Symposium on Algorithms and Computation (ISAAC)*, 896 – 907.
- [7] B. Escoffier, J. Monnot, V. Th. Paschos (2006), Weighted Coloring: further complexity and approximability results, *Information Processing Letters* **97**, 98 – 103.
- [8] G. Finke, V. Jost, M. Queyranne, A. Sebő (2004), Batch processing with interval graph compatibilities between tasks, *Cahiers du laboratoire Leibniz*; available at <http://www-leibniz.imag.fr/NEWLEIBNIZ/LesCahiers/index.xhtml>.
- [9] A. Kesselman, K. Kogan (2004), Non-preemptive scheduling of optical switches, In: *IEEE Global Telecommunications Conference (GLOBECOM)* **3**, 1840 – 1844.
- [10] S. V. Pemmaraju, R. Raman, K. R. Varadarajan (2004), Buffer minimization using max-coloring, In: *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 562 – 571.
- [11] S. V. Pemmaraju, R. Raman (2005), Approximation Algorithms for the Max-coloring Problem, In: *International Colloquium on Automata, Languages and Programming (ICALP)*, 1064 – 1075.