On Computing an Optimal Semi-matching

František Galčík, Ján Katrenič, and Gabriel Semanišin[⋆]

Institute of Computer Science,
P.J. Šafárik University, Faculty of Science,
Jesenná 5, 041 54 Košice, Slovak Republic
{frantisek.galcik,jan.katrenic,gabriel.semanisin}@upjs.sk

Abstract. The problem of finding an optimal semi-matching is a generalization of the problem of finding classical matching in bipartite graphs. A *semi-matching* in a bipartite graph $G = (U, V, E)$ with n vertices and m edges is a set of edges $M \subseteq E$, such that each vertex in U is incident to at most one edge in M. An *optimal semi-matching* is a semi-matching with $deg_M(u) = 1$ for all $u \in U$ and the minimal value of $\sum_{v \in V} \frac{deg_M(v).(deg_M(v)+1)}{2}$. We propose a schema that allows a reduction of the studied problem to a variant of the maximum bounded-degree semi-matching problem. The proposed schema yields to two algorithms for computing an optimal semi-matching. The first one runs in time $O(\sqrt{n} \cdot m \cdot \log n)$ that is the same as the time complexity of the currently best known algorithm. However, our algorithm uses a different approach that enables some improvements in practice (e.g. parallelization, faster algorithms for special graph classes). The second one is randomized and it computes an optimal semi-matching with high probability in $O(n^{\omega} \cdot \log^{1+o(1)} n)$, where ω is the exponent of the best known matrix multiplication algorithm. Since $\omega \leq 2.38$, this algorithms breaks through $O(n^{2.5})$ barrier for dense graphs.

1 Introduction

We consider unweighted bipartite graphs without loops, multiple edges, and isolated vertices. In general we use standard graph theory terminology and notations. If it is not stated otherwise, the symbols n and m stand for the number of vertices and the number of edges of a graph respectively. By $N(X)$ we understand the open neighborhood of a set X, subset of the set of vertices. If v is a vertex belonging to a set W then we refer it as a W-vertex. A *semi-matching* M of a bipartite graph $G = (U, V, E)$ is a set of edges such that each vertex of U is incident to at most one edge of M. A semi-matching M is a *maximum semi-matching* if each vertex of U is incident to exactly one edge of M. We denote by $SM(G)$ the set of all maximum semi-matchings of G. It is easy to see that any

[⋆] The research of the authors was supported in part by the Slovak Research and Development Agency under contracts APVV-0035-10 "Algorithms, Automata, and Discrete Data Structures" and SK-SI-0014-10 "Contemporary graph invariants and applications" and in part by the VVGS grant No. I-10-003-10.

© Springer-Verlag Berlin Heidelberg 2011

bipartite graph without isolated vertices contains a maximum semi-matching and therefore $SM(G)$ is non-empty.

The studied problem is motivated by the following off-line load balancing scenario: Given a set of tasks and a set of machines, each of which can process a subset of tasks. Each task requires one unit of processing time and must be assigned to some machine that can process it. The tasks has to be assigned in such a manner that minimizes some optimization objective. One natural goal is to process all tasks with the *minimum total completion time*. Another goal is to minimize the *average completion time*, or *total flow time*, which is the sum of time units necessary for completion of all jobs (including the units while a job is waiting in the queue).

Let M be a semi-matching. Let us denote by $deg_M(v)$ the degree of a vertex $v \in U \cup V$ in the subgraph of a graph G induced by M. (The definition of a maximum semi-matching immediately implies that if $M \in SM$, then $|M| = |U| = \sum_{v \in V} deg_M(v)$.) For any maximum semi-matching M we define the *cost* of M, denoted by $cost(M)$, as follows: For any machine $v \in V$, its cost with respect to the semi-matching M is

$$cost_M(v) = \sum_{i=1}^{deg_M(v)} i = \frac{deg_M(v) \cdot (deg_M(v) + 1)}{2}.$$

Intuitively, $cost_M(v)$ is the total completion time of jobs assigned to the machine v. The cost of the maximum semi-matching M is the sum of costs taken over all machines:

$$cost(M) = \sum_{v \in V} cost_M(v).$$

An *optimal semi-matching* is a maximum semi-matching with the minimum possible cost among all maximum semi-matchings.

The problem of finding an *optimal semi-matching* is studied from early 70s when an $O(n^3)$-algorithm was developed independently by Horn [5] and Bruno et al. [2]. In the next period, no significant progress has been reported besides the results related to some special cases of the problem and its variations. The problem received considerable attention in the past few years. Harvey et al. [4] shown that an optimal semi-matching minimizes simultaneously the maximum number of tasks assigned to a machine, the flow time and the variance of loads on the machines. They gave also a characterization of an optimal assignment based on cost-reducing paths and provided two algorithms for finding an optimal semi-matching in time $O(n \cdot m)$. Their first algorithm builds the semi-matching step by step starting with an empty semi-matching. Subsequently, in each step, it finds an augmenting path from a free U-vertex to a vertex in V with the smallest possible degree. The second proposed algorithm starts with an arbitrary semi-matching M, where in each step, the algorithm finds a cost-reducing path. The authors provided the $O(\min\{n^{3/2}, m \cdot n\} \cdot m)$ upper bound for running time of this algorithm [4].

The unweighted semi-matching problem was recently generalized to the quasi-matching problem by Bokal et al. [1]. In this problem, a function g is provided, and each vertex $u \in U$ is required to be connected to at least $g(u)$ vertices in V. Observe that the maximum semi-matching problem corresponds to the case when $g(u) = 1$ for each $u \in U$. Bokal et al. [1] used another definition of the optimality and shown that a maximum semi-matching is an optimal semi-matching if and only if its degree distribution is lexicographically minimal. More precisely, let $G = (U, V, E)$ be a bipartite graph, $X \subseteq V$, and $M \subseteq E$. Denote by $d_M(X)$ the sequence $(d_1, d_2, \dots, d_{|X|})$, $d_1 \geq d_2 \geq \dots \geq d_{|X|}$, of degrees of vertices from X in the subgraph induced by M. We say that a maximum semi-matching M_1 is lexicographically smaller than a maximum semi-matching M_2 (denoted by $M_1 <_{lex} M_2$), if the sequence $d_{M_1}(V)$ is lexicographically smaller than the sequence $d_{M_2}(V)$. Analogously we define the relation $M_1 \leq_{lex} M_2$. A maximum semi-matching $M \in SM(G)$ is a *lexicographically minimal semi-matching*, if $M \leq_{lex} M'$ for all $M' \in SM(G)$. A set of all lexicographically minimal semi-matchings of the graph G is denoted as $LSM(G)$.

Very recently, Fakcharoenphol, Laekhanukit and Nanongkai [3] presented $O(\sqrt{n} \cdot m \cdot \log n)$ algorithm for the optimal semi-matching problem, which improves the previous $O(n \cdot m)$ algorithm by Harvey et al. [4]. Their algorithm uses the same reduction to the min-cost flow problem as in [4]. However, instead of canceling one negative cycle in each iteration, the algorithm exploits the structure of the graphs and the cost functions to cancel many negative cycles in a single iteration.

In this paper, we present a new algorithm for finding an optimal semi-matching with respect to optimality criterion due to [1]. Our algorithm is based on the divide-and-conquer strategy that yields to some practically useful properties of the algorithm. In the algorithm, we reduce the problem of computing a lexicographically minimal semi-matching to several computations of a specific variant of the maximum bounded-degree semi-matching problem. This problem is referred as a $BDSM$ problem. The proposed algorithm runs in time $O(T_{BDSM}(n, m) \cdot \log n)$, where $T_{BDSM}(n, m)$ is the time complexity in which the $BDSM$ problem can be solved. Hence our algorithm can viewed also as a schema that utilizes an algorithm for the $BDSM$ problem as a black-box subroutine, i.e. each algorithm for $BDSM$ problem results in an algorithm for computing an optimal semi-matching.

As a consequence, applying known algorithms we get two algorithms for computing an optimal semi-matching. The first one is deterministic and runs in time $O(\sqrt{n} \cdot m \cdot \log n)$ that is the same as the running time of the algorithm from [3]. The second one is randomized and computes an optimal semi-matching with high probability in time $O(n^{\omega} \cdot \log^{1+o(1)} n)$, where ω is the exponent of the best known matrix multiplication algorithm. Since $\omega < 2.38$, for dense graphs this algorithm breaks the $O(n^{2.5})$ upper-bound known before.

2 Preliminaries

The algorithm proposed in this paper is in fact a schema that reduces the problem of computing a lexicographically minimal semi-matching to several instances of a variant of the maximum bounded-degree semi-matching problem (shortly BDSM) that can be stated as follows:

PROBLEM [BDSM]
INSTANCE: A bipartite graph $G = (U, V, E)$ with $n = |U| + |V|$ vertices and $m = |E|$ edges; a capacity mapping $c : V \to \mathbb{N}$ satisfying $\sum_{v \in V} c(v) \le 2 \cdot n$.
QUESTION: Find a semi-matching M in G with maximum number of edges such that $deg_M(v) \le c(v)$ for all $v \in V$.

By $ComputeBDSM(G, c)$ we shall denote an algorithm for solving the instance of $BDSM$ problem.

2.1 Balancing Subroutine

In this subsection, we describe a balancing subroutine. The balancing subroutine provides a method for a transformation of a semi-matching M to a new semi-matching M^* in such a way that for a given semi-matching M_f and a subset W of V we guarantee that all its vertices are loaded at least as in the semi-matching M_f. Moreover, the proposed method is time-efficient and in some sense guarantees that the transformation affects only a small number of V-vertices.

```
Function Balance($G$, $M$, $M_f$, $W$)
  Input: a bipartite graph $G = (U, V, E)$, semi-matchings $M$ and $M_f$ in $G$,
         and a subset $W \subseteq V$
  Output: a semi-matching $M_b$ satisfying properties stated in Lemma 1

  foreach $w \in W$ do  $R_w \leftarrow \{u \in U | (u, w) \in M_f \setminus M\}$;
  foreach $u \in U$ do  $p(u) \leftarrow nil$;
  foreach $(u, v) \in M$ do  $p(u) \leftarrow v$;
  $Q \leftarrow \{w \in W | deg_M(w) < deg_{M_f}(w)\}$;

  $X \leftarrow M$;
  while $Q \neq \emptyset$ do
      $w \leftarrow$ arbitrary vertex from $Q$;
      $Q \leftarrow Q \setminus \{w\}$;
      $u \leftarrow$ arbitrary vertex from $R_w$;
      $R_w \leftarrow R_w \setminus \{u\}$;

      if $p(u) \neq nil$ then
          $X \leftarrow X \setminus \{(u, p(u))\}$;
          if $p(u) \in W \wedge deg_X(p(u)) < deg_{M_f}(p(u))$ then  $Q \leftarrow Q \cup \{p(u)\}$;
      end

      $X \leftarrow X \cup \{(u, w)\}$;
      $p(u) \leftarrow w$;
      if $deg_X(w) < deg_{M_f}(w)$ then $Q \leftarrow Q \cup \{w\}$;
  end
  return $X (= M_b)$
```

Lemma 1. *Let M and M_f be semi-matchings of a bipartite graph $G = (U, V, E)$ and let $V_{M_f} = \{v \in V | \exists u \in U, \exists v' \in V, (u, v) \in E \wedge (u, v') \in M_f\}$. Then for any subset $W \subseteq V$, a semi-matching M_b of G satisfying*

(1) *$deg_{M_b}(v) = deg_{M_f}(v)$, for each $v \in W$ with $deg_M(v) \leq deg_{M_f}(v)$,*
(2) *$deg_{M_f}(v) \leq deg_{M_b}(v) \leq deg_M(v)$, for each $v \in W$ with $deg_M(v) > deg_{M_f}(v)$,*
(3) *$deg_{M_b}(v) \leq deg_M(v)$, for each $v \in V \setminus W$ with $v \in V_{M_f}$,*
(4) *$deg_{M_b}(v) = deg_M(v)$, for each $v \in V \setminus W$ with $v \notin V_{M_f}$, and*
(5) *$|M_b| \geq |M|$*

can be obtained as a results of the function $Balance(G, M, M_f, W)$ in linear time $O(|V(G)| + |E(G)|)$.

2.2 Dividing Subroutine

In the proposed algorithm, we apply the divide-and-conquer strategy. The key element of the algorithm is a dividing construction that allows to reduce the problem of finding a lexicographically minimal semi-matching of a bipartite graph G into two problems of finding a lexicographically minimal semi-matchings of induced subgraphs of G.

The dividing strategy is based on the following observation: Let us assume that we have already computed a maximum semi-matching M of a bipartite graph $G = (U, V, E)$ in the situation when the maximum load (maximal number of incident matching edges) of each V-vertex is bounded from above by a constant *cut* (the value of *cut* will be determined in the main algorithm, where *cut* is used as a variable). Obviously, the computed semi-matching M is not necessary maximum. However, one can expect that if a V-vertex v is not fully loaded in M (i.e., $deg_M(v) < cut$), an enlargement of the upper bound for the load of the vertex v would not help to find a larger semi-matching. On the other hand, if $deg_M(v) = cut$, it could be possible to enlarge M after increasing the allowed capacity (maximum allowed load) of v. As we shall show later, according to a computed semi-matching and an actual load of V-vertices, we can even divide the graph into two disjoint subgraphs in such a way, that the loads of vertices in all lexicographically minimal semi-matchings in one of them are upper-bounded by a given constant *cut*, and the loads of vertices in all lexicographically minimal semi-matchings in the other subgraph are lower-bounded by *cut*.

The following theorem states a few properties of our main algorithm that are crucial for estimation of the total time complexity of the algorithm.

Theorem 1. *Let $down, cut, up$ be non-negative integers such that $down < cut < up$ and $cut \leq 2 \cdot down$. Let $G = (U, V, E)$ be a bipartite graph such that for any lexicographically minimal semi-matching $M \in LSM(G)$ of G it holds $down \leq deg_M(v) \leq up$ for all $v \in V$. Let M_f be a semi-matching in G that satisfies $deg_{M_f}(v) \geq down$ for each vertex $v \in V$. Then there exist bipartite graphs $G^- = (U^-, V^-, E^-)$, $G^+ = (U^+, V^+, E^+)$ and semi-matchings $M_f^- \subseteq E^-$ in G^-, $M_f^+ \subseteq E^+$ in G^+ such that the following properties hold*

(1) *G^-, G^+ are disjoint and induced subgraphs of G, such that $U^- \cup U^+ = U$, $V^- \cup V^+ = V$, $U^- \cap U^+ = \emptyset$, $V^- \cap V^+ = \emptyset$,*
(2) *$\forall v \in V^-$: $deg_{M_f^-}(v) \geq down$,*
(3) *$\forall v \in V^+$: $deg_{M_f^+}(v) = cut$,*
(4) *$\forall M^- \in LSM(G^-)$, $\forall v \in V$: $down \leq deg_{M^-}(v) \leq cut$,*
(5) *$\forall M^+ \in LSM(G^+)$, $\forall v \in V$: $cut \leq deg_{M^+}(v) \leq up$, and*
(6) *$\forall M^- \in LSM(G^-)$, $\forall M^+ \in LSM(G^+)$: $M^- \cup M^+ \in LSM(G)$.*

Moreover, graphs G^+ and G^- can be constructed in time $T_{BDSM}(n, m) + O(n + m)$, where $n = |U| + |V|$, $m = |E|$ and $T_{BDSM}(n, m)$ is time complexity for solving an instance of the Problem BDSM for a graph with n vertices and m edges.

By $Divide(G, down, cut, up, M_f)$ we shall denote an algorithm that constructs graphs G^-, G^+ and semi-matchings M^-, M^+ as claimed in Theorem 1.

3 The Main Algorithm

Our algorithm for computing a lexicographically minimal semi-matching is an implementation of the divide-and-conquer strategy whose division stage is based on Theorem 1. The key computational element of the proposed algorithm is a recursive subroutine $ComputeBLSM(G, down, up, M_f, l)$ that expects five parameters. First three parameters are a bipartite graph $G = (U, V, E)$ and two non-negative integers *down* and *up*, such that for an arbitrary lexicographically minimal semi-matching M of G it holds that $down \leq deg_M(v) \leq up$ for all $v \in V$. We use here a special value $\diamond_\infty$ to denote that no upper-bound on the maximum degree of vertices in a lexicographically minimal semi-matching is given. In all inequalities, we assume that $x \leq \diamond_\infty$ is valid for all $x \in \mathbb{N}$. Observe that the whole computation starts with the parameter *down* set to 0 and the parameter *up* set to $\diamond_\infty$. Note also that it follows from the algorithm that $\diamond_\infty$ can be practically set to the value $2 \cdot n$ without affecting the computation. The fourth parameter is a semi-matching M_f in G satisfying $deg_{M_f}(v) \geq down$ for all $v \in V$. This semi-matching is important for the *Divide* subroutine. Moreover, it is utilized to transfer some results of already realized computation to subproblems. The last parameter l is not related to the computation. It is added as a supplementary element used in the time complexity analysis and denotes a level (depth of recursive call) of the current subproblem.

Algorithm 1. ComputeLSM(G)

```
Input: a bipartite graph $G = (U, V, E)$;
Output: a lexicographically minimal semi-matching $M$;
return $ComputeBLSM(G, 0, \diamond_\infty, \emptyset, 0)$ ;
```

```
Function ComputeBLSM(G, down, up, M_f, l)
  Input: a bipartite graph $G = (U, V, E)$, $down \in \mathbb{N}$, $up \in \mathbb{N} \cup \{\diamond_\infty\}$, s.t.,
         $\forall M \in LSM(G), \forall v \in V : down \leq deg_M(v) \leq up$, and a
         semi-matching $M_f$ in $G$, s.t., $\forall v \in V : deg_{M_f}(v) \geq down$;
  Output: a lexicographically minimal semi-matching $M$ of $G$

  //the trivial case;
  if $G = \emptyset$ then return $\emptyset$ ;
  if $up \neq \diamond_\infty \wedge up - down \leq 1$ then
      $M \leftarrow ComputeBDSM(G, c : \{c(v) \rightarrow up | v \in V\})$ ;
      $M \leftarrow Balance(G, M, M_f, V)$ ;
      return $M$ ;
  end

  //recursive case;
  if $up \neq \diamond_\infty$ then
      $cut \leftarrow \lfloor (down + up)/2 \rfloor$ ;
  else
      $cut \leftarrow max\{2 \cdot down, 1\}$ ;
  end
  $(G^-, M_f^-, G^+, M_f^+) \leftarrow Divide(G, down, cut, up, M_f)$;
  $M^- \leftarrow ComputeBLSM(G^-, down, cut, M_f^-, l + 1)$;
  $M^+ \leftarrow ComputeBLSM(G^+, cut, up, M_f^+, l + 1)$;
  return $M^- \cup M^+$;
```

Let us analyze the correctness of the algorithm. In what follows, we shall use the symbol ? to denote a parameter whose value is not important in an actual context.

Lemma 2. *Let $G = (U, V, E)$ be a bipartite graph and M be a maximum semi-matching of G such that $\forall v, w \in V$: $deg_M(v) - deg_M(w) \leq 1$. Then $M \in LSM(G)$.*

Proof. Follows from Theorem 3.1 in [4] which shows that a semi-matching is optimal if and only if no cost reducing path exists. □

Lemma 3. *Let l_{max} be the maximal value of the parameter l that occurs in a call of the subroutine $ComputeBLSM$ during the computation of $ComputeLSM(G)$. Then, $l_{max} = O(\log n)$ where $n = |U| + |V|$.*

Proof. First we show, using mathematical induction with respect to l, that if a subproblem $ComputeBLSM(?, down, \diamond_\infty, ?, l)$, $l \geq 1$, occurs during the computation of $ComputeLSM(G)$, then it holds $down = 2^{l-1}$. The call of $ComputeBLSM(G, 0, \diamond_\infty, \emptyset, 0)$ can yield only to the computation of a subproblem $ComputeBLSM(?, 1, \diamond_\infty, ?, 1)$ and in this case the induction hypothesis is valid. For $l > 1$, a subproblem $ComputeBLSM(?, down, \diamond_\infty, \emptyset, l - 1)$ can yield only to the computation of a subproblem $ComputeBLSM(?, 2 \cdot down, \diamond_\infty, \emptyset, l)$.

From the induction hypothesis, it follows $2 \cdot down = 2 \cdot 2^{l-2} = 2^{l-1}$. Similarly, we can show by induction on l and utilizing Theorem 1 that if a problem $ComputeBLSM(G', down, \diamond_\infty, ?, l)$ occurs in computation then G' is a subgraph of G. For $down \geq n = |U| + |V|$, the precondition of $ComputeBLSM$ implies that an input graph $G' = (U', V', E')$ is empty. Indeed, if for any semi-matching $M \in LSM(G')$ it holds $deg(v) \geq deg_M(v) \geq down \geq n$ for all $v \in V' \neq \emptyset$, we get a contradiction to $deg(v) < n$ (as G' is a subgraph of G). Hence, if a subproblem $ComputeBLSM(G', ?, \diamond_\infty, ?, l)$ with $G' \neq \emptyset$ is a part of the computation, then $2^{l-1} = down < n$ which implies $l \leq \log n + 1$.

In order to prove formally an upper-bound for l_{max}, we shall show, by induction on l, that if a subproblem $ComputeBLSM(G', down, up, ?, l)$ with $up \neq \diamond_\infty$, $l \geq 1$, and $G' \neq \emptyset$, is a part of the computation then $up - down \leq \frac{n}{2^{l-2-\log n}} + 2 = \frac{n^2}{2^{l-2}} + 2$. Since $ComputeBLSM(?, 0, 1, ?, l)$ is the only subproblem with $l = 1$ and $up \neq \diamond_\infty$ that can occur in the computation, the claim is obvious for $l = 1$. So we can assume that the induction hypothesis holds for $l = 1$. Let us assume now that $up - down \leq \frac{n^2}{2^{l-3}} + 2$ for all subproblems $ComputeBLSM(?, down, up, ?, l-1)$ at level $l - 1$. Each subproblem $ComputeBLSM(?, down, up, ?, l-1)$ can be divided into at most two subproblems $ComputeBLSM(?, down, cut, ?, l)$ and $ComputeBLSM(?, cut, up, ?, l)$ where $cut = \lfloor (down + up)/2 \rfloor$. By an application of the induction hypothesis, for a subproblem $ComputeBLSM(?, down, cut, ?, l)$, we get $cut - down = \lfloor (down + up)/2 \rfloor - down \leq (up - down)/2 \leq \frac{n^2}{2^{l-2}} + 1$. Similarly, for a subproblem $ComputeBLSM(?, cut, up, ?, l)$, we get $up - cut \leq (up - down)/2 + 1 \leq \frac{n^2}{2^{l-2}} + 2$. Hence, the claim holds for the both subproblems. It remains to analyze the case when a subproblem $ComputeBLSM(?, down, up, ?, l)$ is forced by $ComputeBLSM(?, down, \diamond_\infty, ?, l-1)$. We already know that the call $ComputeBLSM(?, down, \diamond_\infty, ?, l-1)$ implies $down = 2^{l-2}$ and $l - 1 \leq \log n + 1$. The problem $ComputeBLSM(?, down, \diamond_\infty, ?, l-1)$ can force only a subproblem $ComputeBLSM(?, down, 2 \cdot down, ?, l)$. For this subproblem, we get $2 \cdot down - down = down = 2^{l-2} \leq n$. Since $l - 2 \leq \log n$, it implies $n \leq \frac{n^2}{2^{l-2}}$. It means that in this case the claim holds too.

Hence, if a subproblem $ComputeBLSM(?, down, up, ?, l)$ with $up \neq \diamond_\infty$ occurs during a computation, it holds $up - down \leq \frac{n}{2^{l-2-\log n}} + 2$. For $l > 2 + 2 \cdot \log n$, it follows $\frac{n}{2^{l-2-\log n}} + 2 < 3$ and $up - down \leq 2$. In this case the construction of subroutine $ComputeBLSM$ guarantees that if a new subproblem is forced then it is the trivial one. Since there are no subproblems with $up \neq \diamond_\infty$ for $l > 3 + 2 \cdot \log n$ and no subproblem with $up = \diamond_\infty$ for $l > \log n + 3$, the assertion follows. □

Lemma 4. *The algorithm ComputeLSM computes a lexicographically minimal semi-matching of G.*

Proof. We have to show that the computation is finite and a semi-matching returned by $ComputeBLSM(G, 0, \diamond_\infty, \emptyset, 0)$ is lexicographically minimal. Since each problem $ComputeBLSM(?, ?, ?, ?, l)$ can yield to the computation of at most two subproblems $ComputeBLSM(?, ?, ?, ?, l+1)$ and from Lemma 3 we have $l \in O(\log n)$, the computation is finite.

Now, we show the correctness of the algorithm. In particular, we show that the subroutine $ComputeBLSM(G, down, up, M_f, l)$ returns a semi-matching M such that $M \in LSM(G)$. We show it by a backward induction on l. In what follows, G refers to the first parameter of a given subproblem $ComputeBLSM$ and not to a parameter of $ComputeLSM$ which initiates the whole computation.

Let us analyze the induction base, i.e., when $l = l_{max}$ and l_{max} is the maximal value of the parameter l that occurs in the computation. Since no subproblem at level $l_{max}+1$ is forced, either the input graph is empty or $up - down \leq 1$. In the former case is obvious. In the latter case when $up - down \leq 1$, Lemma 1 implies that $down \leq deg_M(v) \leq up$ for all $v \in V$. Let M' be an arbitrary semi-matching such that $M' \in LSM(G)$. Precondition of $ComputeBLSM$ implies $deg_{M'}(v) \leq up$ for all $v \in V$. If $|M'| > |M|$, we get a contradiction with maximality of a semi-matching computed by $ComputeBDSM$. Indeed, M' is a feasible solution for the $BDSM$ problem. Hence, due to maximality of M' we have $|M'| = |M|$. It follows that the semi-matching M is a maximum semi-matching for G. And therefore Lemma 2 implies that $M \in LSM(G)$.

Let us analyze the inductive step. If no subproblem is forced, the proof is the same as in the induction base. In the complementary case, which corresponds to a recursive step of the algorithm, two new subproblems are forced. Let G^-, G^+, M_f^-, and M^+ be the results of the subroutine *Divide*. Note that all assumptions of Theorem 1 are satisfied due to preconditions of $ComputeBDSM$ and the fact, which can be easily shown by induction on l, that $up \leq 2 \cdot down + 1$ in the case when $up \neq \diamond_\infty$. Therefore, properties (2) and (4) of Theorem 1 imply that preconditions of $ComputeBLSM(G^-, down, cut, M_f^-, l+1)$ are satisfied as well. Similarly, preconditions of $ComputeBLSM(G^+, cut, up, M_f^+, l+1)$ are satisfied due to properties (3) and (5) of Theorem 1. The new subproblems are at level $l+1$. Hence we can apply the induction hypothesis and we get $M^- \in LSM(G^-)$ and $M^+ \in LSM(G^+)$. Finally, due to the property (6) of Theorem 1 the returned semi-matching $M^- \cup M^+$ belongs to $LSM(G)$. □

Lemma 5. *Let $\mathcal{G}_l$ be a collection of subgraphs of G such that $G' \in \mathcal{G}_l$ if and only if a subproblem $ComputeBLSM(G', ?, ?, ?, l)$ occurs in the computation of $ComputeLSM(G)$. Then, all graphs in $\mathcal{G}_l$ are disjoint and $\bigcup_{G' \in \mathcal{G}_l} G'$ is a subgraph of G.*

Proof. We show the assertion of the lemma by induction on l. For $l = 0$, the claim is trivial. Let us assume that the claim holds for $l - 1$. Clearly, if $G' \in \mathcal{G}_l$, then there is a subproblem $ComputeBLSM(G^*, ?, ?, ?, l{-}1)$ that forces a computation $ComputeBLSM(G', ?, ?, ?, l)$. As we can see in the subroutine $ComputeBLSM$, new subproblems are generated as the result of the *Divide* subroutine. In particular, a subproblem $ComputeBLSM(G^*, ?, ?, ?, l-1)$ can force only subproblems $ComputeBLSM(G^-, ?, ?, ?, l)$ and $ComputeBLSM(G^+, ?, ?, ?, l)$. Then, due to the property (1) of Theorem 1, both graphs G^- and G^+ are disjoint subgraphs of G^* and the claim follows. □

The proposed algorithm is a recursive algorithm that is normally executed in the DFS like manner. However, to complete the analysis of time complexity, let us consider a computation in which the computational tree is traversed in BFS-like manner. In particular, the computation works in stages. In a stage l, we process all subproblems $ComputeBLSM(?, ?, ?, ?, l)$ at level l. Each subproblem is preprocessed and subproblems at level $l + 1$ are constructed. Consequently, execution in the stage l is interrupted and we start to solve subproblems at level $l + 1$ and the stage $l + 1$ starts. After the stage $l + 1$ is completed, we continue with the stage l. From results obtained at the stage $l + 1$ we construct results for all subproblems at level l and the stage l is completed.

The motivation for considering BFS-like computation is the following. Observe that in order to compute $ComputeBLSM(G', ?, ?, ?, l)$, we execute the subroutine $ComputeBDSM(G', ?)$ that computes a solution for the $BDSM$ problem. Let us denote the graph $\bigcup_{G' \in \mathcal{G}_l} G'$ by $G^l = (U^l, V^l, E^l)$. By Lemma 5, the graph G^l is a subgraph of G that consists of components corresponding to graphs in $\mathcal{G}_l$. Therefore, instead of executing the subroutine $ComputeBDSM$ for each graph $G' = (U', V', E') \in \mathcal{G}_l$ separately, we can execute the subroutine $ComputeBDSM$ with the graph G^l as an input. Since vertex sets of all graphs in $\mathcal{G}_l$ are disjoint, we can set capacities for all V-vertices separately. Observe that $\sum_{v \in V(G^l)} c(v) = \sum_{G' \in \mathcal{G}_l} \sum_{v \in V'} c(v) \leq \sum_{G' \in \mathcal{G}_l} 2 \cdot (|U'| + |V'|) = 2 \cdot (|U^l| + |V^l|)$. Hence the precondition of $ComputeBDSM$ is satisfied for input graph G^l. It means that using a BFS-like computational approach we can merge computations of bounded-degree semi-matchings for individual graphs occurring in subproblems at a given level to one computation of a maximum bounded-degree semi-matching at the given level.

Theorem 2. *Let $G = (U, V, E)$ be a bipartite graph. A lexicographically minimal semi-matching of G can be computed in time $O((n + m + T_{BDSM}(n, m)) \cdot \log n)$ where $n = |U| + |V|$, $m = |E|$, and $T_{BDSM}(n, m)$ is the time complexity of an algorithm for the BDSM problem. Moreover, the algorithm for the BDSM problem is applied $O(\log n)$ times during the computation.*

Proof. Let us analyze the modified algorithm that realizes a BFS-like computation of $ComputeLSM$. The correctness of the algorithm follows from Lemma 4. Note that, in order to allow sharing some computational parts, we changed only the order in which independent subproblems are computed. Lemma 3 implies that there are $O(\log n)$ stages in the computation. If we do not take into account time spent by the run of $ComputeBDSM$, due to Theorem 1 the time complexity of each subproblem is linear in size of the input subgraph. From Lemma 5, we have that $\bigcup_{G' \in \mathcal{G}_l} G'$ is a subgraph of G and graphs in $\mathcal{G}_l$ are disjoint. Hence $\sum_{G' \in \mathcal{G}_l} O(|V(G')| + |E(G')| = O(|V(G)| + |E(G)|) = O(n + m)$. In each stage, the algorithm $ComputeBDSM$ is executed with a subgraph of G as an input. Therefore, the computation of $ComputeBDSM$ is in time $T_{BDSM}(n, m)$. It follows that the total time complexity of a stage is $O(n + m) + T_{BDSM}(n, m)$. Taking into account the number of stages, the claim follows. $\square$

Note that the previous theorem holds even in the case of the standard DFS-like computation under assumption that the time complexity $T_{BDSM}(n, m)$ satisfies that $\sum_i T_{BDSM}(n_i, m_i) = O(T_{BDSM}(n, m))$ for all n_i, m_i such that $\sum_i n_i = n$ and $\sum_i m_i = m$.

In what follows, we show how to construct efficient algorithms for finding optimal semi-matching by an application of known algorithms for other problems.

Theorem 3. *Let $G = (U, V, E)$ be a bipartite graph with n vertices and m edges. A lexicographically minimal semi-matching of G can be computed in time $O(\sqrt{n} \cdot m \cdot \log n)$.*

Proof. In [6], Katrenič and Seminišin investigated a problem of finding a maximum (f, g)-semi-matching of a given graph G. The special case of this problem is a computation of a maximum $(1, c)$-semi-matching. A semi-matching M is a $(1, c)$-semi-matching of a bipartite graph $G = (U, V, E)$ if and only if $deg_M(u) \leq 1$ for all $u \in U$ and $deg_M(v) \leq c(v)$ for all $v \in V$. It is easy to see that any instance (G, c) of the $BDSM$ problem can be solved by finding a maximum $(1, c)$-semi-matching of the graph G. The authors of [6] showed that a maximum $(1, c)$-semi-matching of a graph G can be constructed in time $O(\sqrt{n} \cdot m)$, hence $T_{BDSM}(n, m) = O(\sqrt{n} \cdot m)$. Finally, Theorem 2 implies that a lexicographically minimal semi-matching can be computed in time $O(\sqrt{n} \cdot m \cdot \log n)$. □

Proposition 1. *For any bipartite graph $G = (U, V, E)$ with n vertices and a capacity mapping c, the $BDSM$ problem can be solved with high probability in time $O(n^\omega)$, where ω is the exponent of the best known matrix multiplication algorithm.*

Proof. We use a simple reduction to the problem of maximum matching in bipartite graphs. First, we create a new graph G' from the graph $G = (U, V, E)$ by splitting each vertex $v \in V$ exactly $c(v)$ times. The new graph has $|U| + \sum_{v \in V} c(v)$ vertices, which is at most $3 \cdot n$ due to precondition about the mapping c. Next, we compute the maximum matching M' in G'. In [7], the authors shown that a maximum matching can be computed in time $O(n^\omega)$ with high probability. Finally, from M' we construct a semi-matching M for G by merging the corresponding vertices. □

Theorem 4. *Let $G = (U, V, E)$ be a bipartite graph with n vertices and m edges. A lexicographically minimal semi-matching of G can be computed with high probability in time $O(n^\omega \cdot \log n \cdot \log \log n)$, where ω is the exponent of the best known matrix multiplication algorithm.*

Proof. From Proposition 1, $T_{BDSM}(n, m) = O(n^\omega)$. Applying Theorem 2, there are $O(\log n)$ instances of the $BDSM$ problem solved during the computation. Executing the randomized algorithm for the $BDSM$ problem $\log \log n$ times on each problem instance, the claim follows from Theorem 2. □

4 Conclusion

We presented a schema reducing the problem of computing an optimal semi-matching to a variant of the maximum bounded-degree semi-matching problem.

This problem can be efficiently solved utilizing known algorithms for maximum (f, g)-semi-matching or maximum matching using Gaussian elimination. The schema is based on a divide-and-conquer strategy. The problem is recursively divided into smaller independent subproblems that can be solved separately. This property can lead to a construction of simple and efficient parallel algorithms for computing an optimal semi-matching. Another its useful property follows from reduction to a variant of the maximum bounded-degree semi-matching problem. It shows that the time complexity of finding an optimal semi-matching is at most $O(\log n)$ times worse than the time complexity of solving the $BDSM$ problem, which matches best known complexity upper bounds for maximum matchings in bipartite graphs.

References

1. Bokal, D., Brešar, B., Jerebic, J.: A generalization of Hungarian method and Hall's theorem with applications in wireless sensor networks. IFMF, University of Ljubljana, Slovenia, Preprint Series 47(1102), 15 (2009)
2. Bruno, J., Coffman Jr., E.G., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. Commun. ACM 17, 382–387 (1974)
3. Fakcharoenphol, J., Laekhanukit, B., Nanongkai, D.: Faster Algorithms for Semi-Matching Problems (Extended Abstract). In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 176–187. Springer, Heidelberg (2010)
4. Harvey, N.J.A., Ladner, R.E., Lovász, L., Tamir, T.: Semi-matchings for bipartite graphs and load balancing. Journal of Algorithms 59(1), 53–78 (2006)
5. Horn, W.A.: Minimizing average flow time with parallel machines. Operations Research 21(3), 846–847 (1973)
6. Katrenič, J., Semanišin, G.: A generalization of Hopcroft-Karp algorithm for semi-matchings and covers in bipartite graphs, Technical report (2010)
7. Mucha, M., Sankowski, P.: Maximum matchings via gaussian elimination. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 248–255. IEEE Computer Society Press, Washington, DC, USA (2004)