Distributed 2-Approximation Algorithm for the Semi-matching Problem*

Andrzej Czygrinow[1], Michal Hańćkowiak[2], Edyta Szymańska[2], and Wojciech Wawrzyniak[2]

[1] School of Mathematical and Statistical Sciences, Arizona State University, Tempe, AZ,85287-1804, USA
andrzej@math.la.asu.edu
[2] Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland
{mhanckow,edka,wwawrzy}@amu.edu.pl

Abstract. In this paper we consider the problem of matching clients with servers, each of which can process a subset of clients. It is known as the *semi-matching* or *load balancing* problem in a bipartite graph $G = (V, U, E)$, where U corresponds to the clients, V to the servers, and E is the set of available connections between them. The goal is to find a set of edges $M \subseteq E$ such that every vertex in U is incident to exactly one edge in M. The *load* of a server $v \in V$ is defined as $\binom{d_M(v)+1}{2}$ where $d_M(v)$ is the degree of v in M, and the problem is to find an optimal semi-matching, i.e. a semi-matching that minimizes the sum of the loads of the servers. An optimal solution can be found sequentially in polynomial time but the distributed complexity is not well understood. Our algorithm yields $(1+\frac{1}{\alpha})$-approximation (where $\alpha = \max\left\{1, \frac{1}{2}\left(\frac{|U|}{|V|}+1\right)\right\}$) and has time complexity $O\left(\Delta^5\right)$, where Δ is the maximum degree of a vertex in V. In particular, for $\Delta = O(1)$ it gives constant approximation with constant time complexity. We also give a fast algorithm for the case when Δ is large and the degrees in V and U satisfy some additional properties. Both algorithms are deterministic.

1 Introduction

In this paper we restrict our attention to a bipartite graph $G = (V, U, E)$ with bipartition $V \cup U$ and edge set $E \subseteq V \times U$. We denote $|U| = n$ and $|V| = m$ and refer to the vertices of U as clients and to the vertices of V as servers. In what follows we assume that vertices have unique identifiers from $\{1, \ldots, n+m\}$ and know the maximum degree $\Delta = \Delta_V(G)$ of a vertex in V.

Recall that a *matching* in a bipartite graph $G = (V, U, E)$ is a set $M \subseteq E$ of disjoint edges. A matching M is called *maximal* if there is no matching M' such that M is a proper subset of M', and a matching M is called *maximum* if there is no matching M' with $|M'| > |M|$.

* The research supported by grant N N206 565740.

M.K. Aguilera (Ed.): DISC 2012, LNCS 7611, pp. 210–222, 2012.

We are interested in a relaxation of the maximum bipartite matching problem. A *semi-matching* in a bipartite graph $G = (V, U, E)$ is a set of edges $M \subseteq E$ such that every vertex $u \in U$ is incident with exactly one edge in M. In this way a semi-matching provides an assignment of each client to a server that it is connected to. This also implies that for a semi-matching to exist each vertex in U must have degree at least one in G. For a semi-matching M and every vertex $v \in V$ we denote by $d_M(v)$ the number of edges in M incident do v, which corresponds to the number of clients that have to be processed by a server associated with v. With this setting, the total completion time (including the waiting time) of a server v for its $d_M(v)$ clients, which are served in an arbitrary sequential order is equal to $1+2+\cdots+d_M(v) = \binom{d_M(v)+1}{2}$. Therefore, we define the *cost* of a semi-matching M as

$$\text{cost}(M) = \sum_{v \in V} \binom{d_M(v)+1}{2}.$$

A semi-matching with minimum total cost is called an *optimal semi-matching*. This, in turn, corresponds to the total completion time of serving all clients by the servers.

An optimal solution to the problem can be found in polynomial time by sequential algorithms (see Sec. 1.1 for more details). In this paper we analyze the distributed complexity of the optimal semi-matching problem.

We consider a synchronous, message-passing model of computations (referred to as *LOCAL* in [11]). In this model a graph is used to represent an underlying network. The vertices of the graph correspond to computational units, and edges represent communication links. The network is synchronized and in one round a vertex can send and receive messages from all of its neighbors. In addition, in the same round, a vertex can perform some local computations. The running time of the algorithm is the number of rounds needed to solve a problem. We restrict our attention to deterministic algorithms.

1.1 Related Work

The semi-matching problem known also as the load balancing problem has been extensively studied under various names in the scheduling literature. Recently it has received renewed attention after the paper by Harvey, Ladner, Lovász, and Tamir [6], where the name *semi-matching* was introduced. In the same paper the authors proposed two sequential polynomial time algorithms. The first algorithm generalizes the Hungarian method for computing maximum bipartite matchings, while the second is based on a notion of so called *cost reducing paths*. The best running time of the latter algorithm is $O(|E|\sqrt{n+m}\log(n+m))$ and was obtained in [3]. Also, a new approach to this problem was recently proposed in [7]. The weighted version of a related problem to find a semi-matching that minimizes the maximum load among all vertices in V was considered in [3].

The problem, if solved in the distributed setting, can be used, for example, to construct a load balanced data gathering tree in sensor networks [12]. All the

sequential algorithms improve an initial semi-matching to get an optimal one by using some global structures such as cost reducing paths or breadth-first search trees and cannot be applied in the *LOCAL* setting. At the same time, known distributed algorithms for the matching problem are either randomized [9] or rely on techniques that are specific to matchings [5].

As our first approach to the problem, we observed in [2] that in the distributed model of computation the optimal solution requires $\Omega(|V|)$ rounds, and proposed a greedy algorithm which yields $O(1)$-approximation of an optimal semi-matching in time $O(\Delta^2)$. In this paper the approximation ratio is reduced to two via a modification of both, the algorithm and its analysis. Moreover, we give an alternative algorithm GreedySM (see Sec. 3), which is much faster in the case when Δ is large, the degrees of vertices of V do not differ much from each other, and the degrees on U are bounded from above.

1.2 Main Result

Our main result is summarized in the following theorem. Let M^* denote an optimal semi-matching in a bipartite graph $G = (V, U, E)$.

Theorem 1. *In every bipartite graph $G = (V, U, E)$ with the maximum degree in V equal to Δ the algorithm* SemiMatch *(described in Sec. 2.2) finds a semi-matching M such that*

$$cost(M) \leq \left(1 + \frac{1}{\alpha}\right) cost(M^*), \qquad \text{where } \alpha = \max\left\{1, \frac{1}{2}\left(\frac{|U|}{|V|} + 1\right)\right\},$$

and the time complexity of this algorithm is $O\left(\Delta^5\right)$.

1.3 Organization

The rest of the paper is structured as follows. The next section is devoted to the main algorithm SemiMatch and its analysis for arbitrary bipartite graphs. In the last section we give a fast algorithm in the case when Δ is large and some additional conditions on the degrees of vertices of V and U are satisfied.

2 Main Algorithm

Before stating the main algorithm SemiMatch and proving Theorem 1 we need to introduce some more notation.

2.1 Notation and Non-swappable Semi-matchings

The crucial role in our proof is played by semi-matchings which are called *non-swappable*. This property is formally defined below. Observe that if for an arbitrary semi-matching M there exist two vertices $v, w \in V$ connected by a path P consisting of edges alternating between M and $E \setminus M$ and such that v is matched

by M and w is not, and, moreover, $d_M(v) \geq d_M(w) + 2$ then we can lower the cost of M by switching the non-matched edges of P to M and vice-versa, i.e. by taking the symmetric difference of P and M. In such a case the path P is called *cost reducing*. It was proved in [6] that if no cost reducing path exists for a given semi-matching M then M is optimal. These paths, however, can be very long and thus impossible to detect efficiently in the distributed setting. Therefore, we restrict our attention to cost reducing paths of length two only. A semi-matching is non-swappable if there is no cost reducing path of length two.

Definition 1. *Let $G = (V, U, E)$ be a bipartite graph and M be a semi-matching. We say that M is* non-swappable *if for all $v, w \in V, u \in U$ such that $vu \in M$ and $wu \in E \setminus M$ it holds that $d_M(v) \leq d_M(w) + 1$. A path $P = vuw$ not satisfying this condition, i.e. a cost reducing path of length two, is called a* bad path.

Let M^* be an optimal semi-matching in $G = (V, U, E)$. Semi-matchings which are non-swappable form a good approximation of the optimal solution as it is indicated in Theorem 2 below.

Theorem 2. *For any non-swappable semi-matching M in $G = (V, U, E)$,*

$$cost(M) \leq \left(1 + \tfrac{1}{\alpha}\right) cost(M^*), \qquad \text{where } \alpha = \max\left\{1, \frac{1}{2}\left(\frac{|U|}{|V|} + 1\right)\right\}.$$

In order to prove Theorem 2 we need the following result.

Fact 1. *For any semi-matching M in $G = (V, U, E)$ it holds*

$$cost(M) \geq \alpha |U|, \qquad \text{where } \alpha = \max\left\{1, \frac{1}{2}\left(\frac{|U|}{|V|} + 1\right)\right\}.$$

Proof.

$$\text{cost}(M) = \sum_{v \in V} \binom{d_M(v) + 1}{2} \geq \sum_{v \in V} d_M(v) = |U|,$$

as well as, by Cauchy-Schwartz inequality

$$\text{cost}(M) = \sum_{v \in V} \binom{d_M(v) + 1}{2} \geq |V| \binom{\frac{\sum_{v \in V} d_M(v)}{|V|} + 1}{2} = \tfrac{1}{2}|U| \left(\tfrac{|U|}{|V|} + 1\right).$$

Proof (Proof of Theorem 2.). Let M be a non-swappable semi-matching and let M^* be an optimal semi-matching in G. For any $u \in U$ set v_u, v_u^* in such a way that $uv_u \in M$ and $uv_u^* \in M^*$. Define an auxiliary multi-digraph D such that for any $u \in U$ there is an arc $(v_u, v_u^*) \in D$ if $v_u \neq v_u^*$. We assume that D is connected, as otherwise we could analyze each component separately (note that $\text{cost}(M)$ is additive). Let $d_D^+(v), d_D^-(v)$ denote the out-degree and the in-degree of vertex v respectively. Consider two subsets of V, $V^+ = \{v \in V : \ d_D^+(v) > d_D^-(v)\}$, $V^- = \{v \in V : \ d_D^-(v) > d_D^+(v)\}$. Let $d = \sum_{v \in V^+}(d^+(v) - d^-(v)) = \sum_{v \in V^-}(d^-(v) - d^+(v))$. Clearly, if $V^+ = V^- = \emptyset$ then $\text{cost}(M) = \text{cost}(M^*)$. In the other case we use the following theorem(see [1], p. 84-85) about an Eulerian cover by arc-disjoint open trails.

Theorem 3. *For every connected directed multigraph D with $d > 0$ there exist (edge disjoint) open trails $P_1, P_2, \ldots, P_d$ which cover D, i.e. $D = \bigcup_{j=1}^{d} P_j$ and $P_i \cap P_j = \emptyset$(edgewise), where the beginnings of $P_1, P_2, \ldots, P_d$ are in V^+, while the ends are in V^-.*

For every $e = (v, v^*) \in D$, let u_e be such that $v = v_{u_e}, v^* = v^*_{u_e}$. Construct a sequence of semi-matchings $M_1, \ldots, M_d$ obtained by swapping the edges of current semi-matching and the optimum M^* corresponding to the arcs of the trails. Thus, $M_1 = (M \setminus \{\{u_e v_{u_e}\} : e \in P_1\}) \cup \{\{u_e, v^*_{u_e}\} : e \in P_1\}, \ldots, M_d = (M_{d-1} \setminus \{\{u_e v_{u_e}\} : e \in P_d\}) \cup \{\{u_e, v^*_{u_e}\} : e \in P_d\} = M^*$. Let v, w be the beginning and end of P_1, respectively. Then it holds $d_M(v') = d_{M_1}(v')$ for every $v' \neq v, w$ but $d_{M_1}(v) = d_M(v) - 1$, $d_{M_1}(w) = d_M(w) + 1$ and $d_M(w) \geq d_M(v) - |P_1|$ because M is non-swappable.

So, the cost changes as follows

$$\operatorname{cost}(M) - \operatorname{cost}(M_1) = \binom{d_M(w)+1}{2} - \binom{d_M(w)+2}{2} +$$

$$+ \binom{d_M(v)+1}{2} - \binom{d_M(v)}{2} = \tfrac{1}{2}\left[-2(d_M(w)+1) + 2d_M(v)\right]$$

$$= d_M(v) - d_M(w) - 1 \leq |P_1| - 1.$$

Note that M_1 is also non-swappable as no new bad paths were formed. Analogously, $\operatorname{cost}(M_{j-1}) - \operatorname{cost}(M_j) = (d_{M_{j-1}}(v) - d_{M_{j-1}}(w) - 1) \leq (|P_j| - 1)$ for every $j = 2, \ldots, d$. In consequence, $\operatorname{cost}(M) - \operatorname{cost}(M^*) \leq \sum_{j=1}^{d}(|P_j| - 1) = |D| - d \leq |D| \leq |U|$. Further, by Fact 1, we obtain $\operatorname{cost}(M) \leq \operatorname{cost}(M^*) + |D| \leq \left(1 + \frac{1}{\alpha}\right)\operatorname{cost}(M^*)$.

For arbitrary, non-swappable semi-matching the above theorem yields the following.

Corollary 1. *For any non-swappable semi-matching M in G, $cost(M) \leq 2 \cdot cost(M^*)$.*

2.2 An Approximation Algorithm and Its Analysis

We are now ready to present the algorithm SemiMatch (see the pseudocode below) returning a non-swappable semi-matching which, by Theorem 2, is a $(1 + 1/\alpha)$-approximation of the optimum. It starts by finding an arbitrary semi-matching M in $G = (V, U, E)$ (step 1) and systematically eliminating all cost reducing paths of length two with respect to M (i.e. bad paths as in Def. 1) in such a way that no new such paths are formed. This is quite a challenging task and requires a systematic approach in which we consider bad paths that end in vertices of degree $0, 1, \ldots \Delta$ in M (the loop in step 2) and remove them by the swapping operation (step 7). As a result, M is modified and possibly new bad paths are created (step 9) and are again eliminated (step 11). This process, as we show below, ends after finitely many iterations. Unfortunately, we do not

know how to do it for bad paths of length greater than two, which could possibly improve the approximation ratio.

To give a formal analysis we need some more notation. Let M be a semi-matching in a bipartite graph $G = (V, U, E)$. Then we set

$$V_k := V_k(M) = \{v \in V | d_M(v) = k\}.$$

In addition, $V_{\leq k} = \{v \in V | d_M(v) \leq k\}$ and $V_{<k}, V_{\geq k}, V_{>k}$ are defined analogously. Note that as M changes during the execution of the algorithm so do the above sets. To keep track of the local values of the degrees of vertices in the algorithm we also use the sets L_k corresponding to V_k.

Recall that an M-alternating path $P = vuw$ with $vu \in M$ is bad (Def. 1) if it is cost reducing, i.e. $d_M(v) - d_M(w) \geq 2$. For every such P we set $Start(P) = \{v\}$, $End(P) = \{w\}$. For two (not necessarily disjoint) sets $A, B \subset V$ we use $Bad(A, B)$ to denote the set of all bad paths from some $x \in A$ to some $y \in B$. Observe that in this setting the condition $Bad(V, V) = \emptyset$ with respect to a current semi-matching implies that this semi-matching is non-swappable (Thm. 5). To meet this condition the algorithm eliminates bad paths by swapping, in parallel, bad paths which do not interfere with each other. Formally, we use a simple procedure denoted by $Bad_{ind}(A, B)$ to find a maximal set of paths from $Bad(A, B)$ such that for any two paths $P, P' \in Bad_{ind}(A, B)$, $Start(P) \cap Start(P') = \emptyset$ and $End(P) \cap End(P') = \emptyset$. Since M is a semi-matching, the paths constructed by $Bad_{ind}(A, B)$ are also internally disjoint. What is more, since $\Delta(V) \leq \Delta$, $Bad_{ind}(A, B)$ can be performed easily in $O(\Delta)$ rounds for any $A, B \subseteq V$. Finally, if M is a semi-matching and Z is a set of bad paths, then $Z \oplus M$ is the semi-matching obtained from M by deleting the edge $vu \in M$ and adding uv' for every path vuv' in Z. Clearly, $Z \oplus M$ is a semi-matching as the degree of every u stays one.

Algorithm 1. SemiMatch

```
1: ∀u ∈ U pick an arbitrary edge e_u incident to u and let M = ⋃_{u∈U} e_u.
2: for k = 0 to Δ − 2 do
3:     for i = 0 to 2Δ do
4:         ∀_{v∈V} l(v) = d_M(v), ∀_{t=0,...,Δ} L_t = {v ∈ V | l(v) = t}        ▷ Layers
5:         X = Bad_ind(V_{>k+1}, V_k)                    ▷ Maximal set of disjoint paths
6:         S = Ends(X), S^c = V \ S
7:         M = M ⊕ X                                              ▷ Applying X to M
8:         for j = 0 to 2Δ^2 do
9:             Y = ⋃_{t=1}^{k} Bad_ind(L_t ∩ S, L_{t−1} ∩ S^c)
10:            S = S ∪ Ends(Y) \ Starts(Y), S^c = V \ S
11:            M = M ⊕ Y                                          ▷ Applying Y to M
12:        end for
13:    end for
14: end for
15: return M
```

We now proceed with the analysis of SemiMatch. Our goal is to prove that it terminates in $O(\Delta^5)$ steps (via Thm. 4) and returns a semi-matching that is non-swappable (via Thm. 5). At the end Theorems 4 and 5 together with Theorem 2 will yield our main result, Theorem 1.

Fix n and let $C(k,i)$ be the smallest integer C such that after C iterations of the loop 8-12 the set $Y := \bigcup_{t=1}^{k} Bad_{ind}(L_t \cap S, L_{t-1} \cap S^c)$ is empty. The following theorem provides an upper bound on the value of $C(k,i)$. It can be easily proved that $C(k,i)$ is finite. Our next result provides a specific bound in terms of Δ.

Theorem 4. $C(k,i) \leq 2\Delta^2$.

The next theorem guarantees that the algorithm returns a non-swappable semi-matching after termination.

Theorem 5. *After all iterations of the loop 2–14, $Bad(V,V) = \emptyset$. In particular, the semi-matching returned in Step 15 is non-swappable.*

First we show how the degree of a vertex in a semi-matching changes in the course of the algorithm. Note that in SemiMatch the label $l(v)$ of a vertex v does not change during the execution of loop 8-12 but $d_M(v)$ may change.

Fix k, i and let $d_M(v)$ denote the degree of v (in M) at the beginning of the i-th iteration (step 3) and let $d_M^{(j)}(v)$ be the degree of v at the beginning of the j-th iteration (step 8) ($j = 0, \ldots, C+1$, where $j = 2\Delta^2 + 1$ gives the degree after all iterations). For $T \in \{S, S^c\}$, we say that a vertex v *has state* T at a given time if $v \in T$ at this time.

Fact 2. *The following inequalities hold for every j.*

a) If $d_M(v) \geq k+2$, then $-1 \leq d_M^{(j)}(v) - d_M(v) \leq 0$.
b) If $d_M(v) = k+1$, then $d_M^{(j)}(v) - d_M(v) = 0$.
c) If $d_M(v) \leq k$, then $0 \leq d_M^{(j)}(v) - d_M(v) \leq 1$.

Proof. If $d_M(v) = k+1$, then v cannot be the beginning or the end of any path in X or Y and its degree does not change. If $d_M(v) \geq k+2$, then v can be only the beginning of at most one path in $X \cup Y$. Therefore, for every j, $d_M(v) \geq d_M^{(j)}(v) \geq d_M(v) - 1$. Now assume that $d_M(v) \leq k$. First, note that if $v \in S$ in step 6, then its degree increases by exactly one in step 7. Finally, in view of step 9, only paths from vertices in S to vertices in S^c are used. If $P \in Y$ and $P = vuw$, then w has state S^c, v has state S in step 9 and $v \in S^c$, $w \in S$ in step 10. Consequently, the degree of any vertex in $V_{<k+1}$ cannot decrease and can increase by at most one.

Next we show three lemmas determining the elimination of bad paths in the execution of the algorithm, which are used in the proof of Theorem 5.

Lemma 1. *For every k and i, if $Bad(V, V_{<k}) = \emptyset$ at the beginning of the i-th iteration (step 3), then $Bad(V, V_{<k}) = \emptyset$ after this iteration.*

Proof. Suppose, to the contrary, that there is a path $vuv' \in Bad(V, V_{<k})$ after the i-th iteration. First note that $v' \in \bigcup_{i \leq k} L_i$. Indeed, by Fact 2, every vertex in $V_{>k}$ has degree at least $k+1$ in all of the iterations of loop 8-12. Thus, we shall consider two cases based on $l(v)$.

- *Case 1:* $l(v) \geq k+1$. Then, in step 4, $d_M(v) \geq k+1$ and $vu \in M$ at the beginning of the iteration as v cannot be the endpoint of any path in X or Y. Thus vuv' is in $Bad(V, V_{<k})$ at the beginning of the i-th iteration contradicting the assumption.
- *Case 2:* $l(v) \leq k$. First suppose that $vu \notin M$ at the beginning of the iteration. Then, at the same time there must be another path wuv with $wu \in M$ for some w that satisfies $l(w) > l(v)$. Since $l(v) > l(v')$, $wuv' \in Bad(V, V_{<k})$ at the beginning of the iteration. Thus, we may assume that $vu \in M$ at the beginning of the iteration (step 9). Since there are no bad paths in step 5, we have $l(v) = l(v') + 1$ and after all iterations $v \in S, v' \in S^c$, that is vuv' could be added to Y, contradicting the fact that after C iterations Y is empty.

Lemma 2. *Let $Bad(V, V_{\leq k-1}) = \emptyset$ at the beginning of the k-th iteration (step 2) for some k. Then for every $i \in \{0, \ldots, 2\Delta\}$ and every $v \in V_k$ after steps 4-12 of the i-th iteration at least one of the following conditions is satisfied.*

a) $Bad(V, v) = \emptyset$.
b) $|Bad(V, v)|$ *decreases by at least one in the i-th iteration.*
c) $\max\{d_M(w) | \, w \in Starts(Bad(V, v))\}$ *decreases by at least one in the i-th iteration.*

Proof. Fix k. First assume that $v \notin L_k$ at the beginning of the i-th iteration (step 4) for some i. Fact 2 implies that $l(v) = k-1$. If at the end of the i-th iteration there exists a path $wuv \in Bad(V, v)$, then $l(w) > k+1$ and so $wu \in M$ in step 4 as w cannot be the endpoint of any bad path in the i-th iteration. Thus $wuv \in Bad(V, v)$ at the beginning of the k-th iteration contradicting the assumption. Thus, $Bad(V, v) = \emptyset$ in the i-th iteration.

Now suppose that $v \in L_k$ at the beginning of the i-th iteration (step 4). First, we prove that no new paths are added to $Bad(V, v)$ in the i-th iteration. Suppose, to the contrary, that wuv is added to $Bad(V, v)$. Since the degree of v has not changed in this iteration, either wu is added to M in this iteration or the degree of w (in M) increases by one. However, $w \in V_{>k}$ and no new edges incident to w are added to M excluding both possibilities, so the path wuv cannot be added in this iteration. Assume $Bad(V, v) \neq \emptyset$. If $v \in Ends(X)$ after step 6, then at least one path from $Bad(V, v)$ is deleted in step 7. If $v \notin Ends(X)$, by maximality of X, for every $x \in Starts(Bad(V, v))$, $x \in Starts(X)$. Since $l(x) > k+1$, $x \notin Ends(Y)$ for any Y and so $d_M(x)$ decreases by at least one in this iteration.

Lemma 3. *Let $Bad(V, V_{\leq k-1}) = \emptyset$ at the beginning of the k-th iteration for some k. Then $Bad(V, V_{\leq k}) = \emptyset$ after the k-th iteration.*

Proof. Assume $Bad(V, V_{\leq k-1}) = \emptyset$ at the beginning of the k-th iteration. In view of Lemma 1, $Bad(V, V_{\leq k-1}) = \emptyset$ after the k-th iteration. Let $v \in V_k$

after the k-th iteration. By Lemma 2, in each of the iterations of the loop 3-12 at least one of the conditions is satisfied and once $Bad(V, v) = \emptyset$, we cannot add new bad paths to $Bad(V, v)$. Since $d_G(w) \leq \Delta$ for every $w \in V$, $0 \leq |Bad(V, v)| + \max\{d_M(w) | \, w \in Starts(Bad(V, v))\} \leq 2\Delta$ and so, by Lemma 2, after 2Δ iterations $Bad(V, v) = \emptyset$.

Proof (Proof of Theorem 5). First note that after the k-th iteration, $Bad(V, V_{\leq k}) = \emptyset$. Indeed, by induction on k, if $k = 0$, then $Bad(V, V_{-1}) = \emptyset$ and so, by Lemma 3, $Bad(V, V_{\leq 0}) = \emptyset$. For the inductive step, assume that $Bad(V, V_{\leq k-1}) = \emptyset$ and thus, by Lemma 3, $Bad(V, V_{\leq k}) = \emptyset$. Since $\Delta(V) \leq \Delta$ and there are no bad paths ending in vertices of degree at least $\Delta - 1$, after all iterations of the algorithm we have $Bad(V, V) = Bad(V, V_{\leq \Delta - 2}) = \emptyset$.

Next we establish some facts and lemmas necessary to prove Theorem 4. To prove that $C(k, i) \leq 2\Delta^2$ we will use the following auxiliary directed multigraph. Fix k, i and assume that $Bad(V, V_{<k}) = \emptyset$ at the beginning of the i-th iteration of 3-13. Let $H = H_{i,k} = (V_H, E_H)$ be defined as follows, $V_H = L_{\leq k}$ and for $v, w \in V_H$ there is an arc from v to w (with label u) if the path vuw was is in Y in one of the iterations 8-12. Note that if $(v, w) \in E_H$, then $l(v) > l(w)$ and $vu \in M$, $uw \notin M$ prior to applying vuw to M. We first state the following simple fact about H.

Fact 3. *The multigraph H is acyclic and the longest directed path in H has length at most $\Delta - 2$. The maximum out-degree of H and the maximum in-degree of H are at most Δ.*

Proof. If (v, w) is an arc, then $l(v) > l(w)$ and so there are no cycles and the longest path in H has length at most $\Delta - 2$ as $k \leq \Delta - 2$. There is at most one arc in H from v to w which is labeled with u and since the maximum degree of vertices in V is Δ, the max out-degree in H is at most Δ. The same applies to the in-degree.

Recall that for $T \in \{S, S^c\}$, we say that a vertex v has state T at a given time if $v \in T$ at this time. For a directed path P in H, let $q(P)$ be the sum of all changes of states of all vertices in P. In view of Fact 3 we have

$$0 \leq q(P) \leq 2\Delta^2. \tag{1}$$

Now consider the set S in step 6. In step 7 and step 11, if $vuw \in X$ (or Y), then w acquires the S-state that was previously on v and we say that S *moves from v to w*. In addition, we say that we *apply* the path vuw. We need the following important lemma.

Lemma 4. *Fix k, i and assume that $Bad(V, V_{<k}) = \emptyset$ at the beginning of the i-th iteration (steps 3-13). If there exists a bad path vuv' such that $v \in S$ and $v' \in S^c$ for some iteration j_0 (of the loop 8-12) then vuv' is a bad path in every iteration $j < j_0$ of the loop 8-12.*

Proof. Fix j_0 and suppose that vuv' is a bad path such that $v \in S$ and $v' \in S^c$ in the iteration j_0. Then, since vuv' is a bad path in one of the iterations, Fact 2 implies $d_M(v) \geq d_M(v') + 2$ in the iteration j_0. If $uv \notin M$ for some $j < j_0$, then there is w with $l(w) > l(v)$ such that $wu \in M$ for all $j < j_0$ and wuv' is a path in $Bad(V, V_{<k})$ at the beginning of the i-th iteration contradicting the assumption of the lemma.

Returning to the main line of reasoning towards the proof of Theorem 4 fix k and i. Further, for a vertex $v \in S$ in step 6, let $P = v_1 \dots v_s$ be the path of successive moves of the state S that was originally on $v_1 := v$. Thus, in step 6, v_1 has S and in steps 8-12, S will move by applying bad paths $v_i u_i v_{i+1}$ until it reaches v_s which is its final destination. To prove that in $O(\Delta^2)$ steps S will reach v_s we extend P to $P' = v_1, \dots v_s, \dots, v_{s+l}$ using the following operation. Let j_p be the largest index j such that in the j-th iteration state S on v_{s+p} is moved to a vertex w and let v_{s+p+1} be this vertex w. Continue extending P' if possible. Note that P' is a directed path in the multigraph H introduced earlier. The following lemma holds for P'.

Lemma 5. *For every iteration of the loop* $8-12$ *either at least one vertex on* P' *changes its state or state* S *is on its final destination* v_s*.*

Proof. Consider an iteration of the loop 8-12 and suppose that at the beginning of this iteration the state S that originated at v_1 is on v_i for some $i \in \{1, \dots, s-1\}$. We prove that there exists a $j > i$ such that $v_j \in P'$ has state S^c. (Note that i, j have now nothing to do with the indices of the loops in the algorithm.) First observe that if for every $j \in \{s+1, \dots, s+p\}$, the state of v_j is S, then these are final states of these vertices. By maximality of P', the vertex v_{s+p} has its final state. Now suppose, $v_{s+i}, \dots, v_{s+p}$ have their final states (all S). If $v_{s+i-1} \in S$, then the state S cannot move from v_{s+i-1} to v_{s+i} and since, by definition of P', the vertex v_{s+i} was the last recipient of the S-state from v_{s+i-1}. So v_{s+i-1} will not change its state. Now, if all $v_{s+1}, \dots, v_{s+p}$ are in state S, then v_s is in S^c as by definition of P', v_{s+1} was the last recipient of an S-state from v_s.

Further, since $i < j$, v_i has state S, and v_j has state S^c, there is an index $i \leq l < j$ such that $v_l \in S$, $v_{l+1} \in S^c$. Thus, by Lemma 4, the path $v_l u_l v_{l+1}$ is a bad path in this iteration. By maximality of Y in step 9, either v_l or v_{l+1} is the endpoint of a path from Y.

Proof (Proof of Theorem 4). For a fixed k, i and a vertex $v \in S$ in step 6, let P' be defined as before. By (1), $q(P') \leq 2\Delta^2$ and, in view of Lemma 5, in $2\Delta^2$ iterations the state originated at v will reach its final destination. Since every bad path is obtained from some $v \in S$, after $2\Delta^2$ iterations there will be no bad paths in Y, that is $Y = \emptyset$.

3 Semi-matchings via the Minimum Sum Set Cover

In this section we present an alternative approach to the problem of computing a semi-matching in a distributed setting. We give an algorithm for approximating

the optimal semi-matching with a slightly modified definition of the cost (see the proof of Theorem 6 for details) in a graph $G = (V, U, E)$ with degrees on U bounded from above and with an additional assumption on degrees of vertices in V. The method relies on a reduction to the *Minimum Sum Set Cover* (MSSC) problem that we shall define next. Given a hypergraph $H = (V_H, E_H)$, a solution to the MSSC problem is a bijection $\phi : V_H \to \{1, \ldots, |V_H|\}$ and the cost of ϕ, $cost_{MSSC}(\phi) = \sum_{e \in E_H} \min\{\phi(v) | v \in e\}$. We let $opt_{MSSC}(H) = \min_\phi cost_{MSSC}(\phi)$ and call a solution ϕ *optimal* if $cost_{MSSC}(\phi) = opt_{MSSC}(H)$. It is known (see [4]) that a greedy algorithm solving this problem yields a 4-approximation. The algorithm in [4] works as follows. In the i-th ($i = 1, \ldots, |V_H|$) iteration a vertex v of the maximum degree in the current hypergraph is selected, $\phi(v) := i$, and all edges containing v are removed from this hypergraph. By a slight modification of the proof in [4] one can show that, if instead of selecting a vertex with the maximum degree, the procedure picks a vertex v with $d_H(v) \geq \Delta_H/2$, then the approximation ratio is nine instead of four, that is the obtained bijection ϕ satisfies

$$cost_{MSSC}(\phi) \leq 9 \cdot opt_{MSSC}(H). \tag{2}$$

Now we describe the reduction, which takes an instance $G = (V, U, E)$ with $U = \{u_1, \ldots, u_{|U|}\}$ of a semi-matching problem, and returns an MSSC instance using f-matchings. Given $f \in Z^+$, an f-matching in G is a set $Q \subseteq E$, such that $d_Q(v) \leq f$ for every $v \in V$ and $d_Q(u) \leq 1$ for every $u \in U$. Formally, given $G = (V, U, E)$ construct a hypergraph $H = H(G) = (V_H, E_H)$, where V_H is the set of all f-matchings in G and $E_H = \{e_1, \ldots, e_{|U|}\}$, where e_i is the set of all f-matchings $Q \in V_H$ with $d_Q(u_i) = 1$. Observe that, for $e_i \in E_H$, $v \in V_H$ and an f-matching Q in G corresponding to v, the fact that v belongs to e_i is equivalent to $d_Q(u_i) = 1$, and, therefore $d_H(v) = |Q|$. Moreover, the operation of removing an edge e_i from H is equivalent to removing the corresponding vertex u_i from U. As a result of this, the degrees of all vertices in e_i decrease by one and at the same time the sizes of all f-matchings containing u_i decrease by one. Notice also, that the set V_H remains unchanged during such an operation while its elements (f-matchings) might decrease in size.

The greedy algorithm finding an MSSC in H can be now rewritten as the following procedure GreedySM in G: Find a maximal f-matching Q in G, delete from U all vertices u with $d_Q(u) = 1$, and continue until G is empty. The result of GreedySM is the union M of all maximal f-matchings Q computed in the course of the procedure. Note that any maximal f-matching is a $\frac{1}{2}$-approximation of a maximum f-matching in G.

Next we relate an arbitrary semi-matching in G with a solution to the MSSC problem in $H(G)$ using a labeling. For a semi-matching M in G let ϕ_M be a solution to the MSSC problem in H defined as follows. Consider the sequence $s := 1, 1, \ldots, 1, 2, 2 \ldots, 2, 3, \ldots$ where each i appears exactly f times. Every vertex $v \in V$, in parallel, labels the edges of M incident to v by successive numbers in s. Let k be the maximum label used in this process. Then, the edges of M with label i form an f-matching Q_i. Set $\phi_M(Q_i) := i$ and for every other

f-matching Q in G let $\phi_M(Q) = j$ for some $j > k$. In addition, if M is obtained by GreedySM and the labels of edges in M correspond to iterations (edges added in the i-th iteration have label i), then ϕ_M obtained as above is said to *agree* with GreedySM.

For a semi-matching M in G, $p \in \Re$, and $v \in V$, let $k_M(v) := p \cdot d_M(v)$ if $d_M(v) < f$ and let $k_M(v) := \frac{p}{f} d_M^2(v)$ if $d_M(v) \geq f$. Let $K(M,p) := \sum_{v \in V} k_M(v)$.

Lemma 6. *For a semi-matching M, $cost_{MSSC}(\phi_M) \in [K(M, \frac{1}{2}), K(M,2)]$.*

Proof. For an edge $e = \{u, v\} \in M$, $u \in U$, let $\psi(e)$ be the label of e assigned as described before. Then $Q_{\psi(e)}$ has the smallest value of ϕ_M from all f-matchings containing u and therefore, $cost_{MSSC}(\phi_M) = \sum_{e \in M} \psi(e)$. Let $v \in V$. If $d_M(v) < f$, then the sum of $\psi(e)$ over all e's in M incident to v is $d_M(v)$ and otherwise it is $\frac{1}{2}r(r+1)f + (r+1)(d_M(v) - fr) = (r+1)(d_M(v) - \frac{1}{2}fr)$, where $r = \lfloor \frac{d_M(v)}{f} \rfloor$. Finally, we have $\frac{d_M^2(v)}{2f} \leq (r+1)(d_M(v) - \frac{1}{2}fr) \leq \frac{2d_M^2(v)}{f}$.

In the analysis of the algorithm GreedySM we use also the following fact.

Fact 4. *Let M be a semi-matching such that for all $v \in V$ it holds $d_M(v) \geq t$. Then there exists an optimal semi-matching M^* with $d_{M^*}(v) \geq t$ for all $v \in V$.*

Theorem 6. *Let $a, b \in Z^+$, $\Delta = \Delta_V(G) > ab$.* GreedySM *finds a 36-approximation of the semi-matching problem in a graph $G = (V, U, E)$ that satisfies: $d(v) \in [\Delta/a, \Delta]$ for every $v \in V$ and $d(u) \leq b$ for every $u \in U$. The algorithm runs in $O(ab^2)$ rounds.*

Proof. Let $f := \lfloor \Delta/(ab) \rfloor$ and let M be obtained by GreedySM. Let ϕ_M be the solution to the MSSC problem obtained from M that agrees with GreedySM. Then, in view of the previous discussion and by (2), $cost_{MSSC}(\phi_M) \leq 9 \cdot opt_{MSSC}(H)$. To simplify computations we redefine

$$cost_{SM}(M) := \frac{2}{f} \sum_{v \in V} d_M^2(v).$$

By Lemma 6, $\frac{1}{4} cost_{SM}(M) \leq K(M, \frac{1}{2}) \leq cost_{MSSC}(\phi_M) \leq 9 \cdot opt_{MSSC}(H)$. On the other hand, by Hall's theorem there exists a semi-matching M' with $d_{M'}(v) \geq f$ for all $v \in V$. By Fact 4 there also exists an optimal semi-matching M^* with the same property. From Lemma 6, $cost_{MSSC}(\phi_{M^*}) \leq K(M^*, 2) = cost_{SM}(M^*)$ and so $opt_{MSSC}(H) \leq cost_{SM}(M^*)$. Thus, $cost_{SM}(M) \leq 36 \cdot cost_{SM}(M^*)$. The number of iterations of GreedySM is $O(ab)$ as in each iteration each vertex from V looses f or all incident edges. Finding a maximal f-matching can be done in $O(b)$ rounds (using procedures similar to those computing a maximal matching, see e.g. [13]).

Remark 1. *If $a = 1$, $b \in Z^+$, $\Delta > b$, then the algorithm* GreedySM *finds a 36-approximation of the semi-matching problem in every graph $G = (V, U, E)$, which is Δ-regular on V, in $O(b^2)$ rounds. In this case it outperforms the algorithm* SemiMatch*, which has time complexity $O(\Delta^5)$ independent of the value of b, the upper bound on the degrees of vertices in U.*

References

1. Andrasfai, B.: Introductory Graph Theory. Adam Hilger (1977)
2. Czygrinow, A., Hańćkowiak, M., Krzywdziński, K., Szymańska, E., Wawrzyniak, W.: Brief Announcement: Distributed Approximations for the Semi-matching Problem. In: Peleg, D. (ed.) DISC 2011. LNCS, vol. 6950, pp. 200–201. Springer, Heidelberg (2011)
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. Feige, U., Lovasz, L., Tetali, P.: Approximating Min Sum Set Cover. Algorithmica 40(4), 219–234 (2004)
5. Hańćkowiak, M., Karoński, M., Panconesi, A.: On the distributed complexity of computing maximal matchings. In: Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, San Francisco, CA, USA, pp. 219–225 (January 1998)
6. Harvey, N.J.A., Ladner, R.E., Lovasz, L., Tamir, T.: Semi-matchings for bipartite graphs and load balancing, J. Algorithms 59(1), 53–78 (2006)
7. Galčík, F., Katrenič, J., Semanišin, G.: On Computing an Optimal Semi-matching. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 250–261. Springer, Heidelberg (2011)
8. Linial, N.: Locality in distributed graph algorithms. SIAM Journal on Computing 21(1), 193–201 (1992)
9. Lotker, Z., Patt-Shamir, B., Rosén, A.: Distributed Approximate Matching. SIAM J. Comput. 39(2), 445–460 (2009)
10. Low, C.P.: An approximation algorithm for the load-balanced semi-matching problem in weighted bipartite graphs. Information Processing Letters 100(4), 154–161 (2006)
11. Peleg, D.: Distributed Algorithms, A Locality-Sensitive Approach. SIAM Press (2000)
12. Sadagopan, N., Singh, M., Krishnamachari, B.: Decentralized utility-based sensor network design. Mob. Netw. Appl. 11(3), 341–350 (2006)
13. Suomela, J.: Survey of Local Algorithms (manuscript), `http://www.cs.helsinki.fi/u/josuomel/doc/local-survey.pdf`