Online Maximum k-Coverage*

Giorgio Ausiello[1], Nicolas Boria[2], Aristotelis Giannakos[2], Giorgio Lucarelli[2], and Vangelis Th. Paschos[2,3]

[1] Dip. di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza"
ausiello@dis.uniroma1.it
[2] LAMSADE, CNRS UMR 7243 and Université Paris-Dauphine
{boria,giannako,lucarelli,paschos}@lamsade.dauphine.fr
[3] Institut Universitaire de France

Abstract. We study an online model for the maximum k-vertex-coverage problem, where given a graph $G = (V, E)$ and an integer k, we ask for a subset $A \subseteq V$, such that $|A| = k$ and the number of edges covered by A is maximized. In our model, at each step i, a new vertex v_i is revealed, and we have to decide whether we will keep it or discard it. At any time of the process, only k vertices can be kept in memory; if at some point the current solution already contains k vertices, any inclusion of a new vertex in the solution must entail the definite deletion of another vertex of the current solution (a vertex not kept when revealed is definitely deleted). We propose algorithms for several natural classes of graphs (mainly regular and bipartite), improving on an easy $\frac{1}{2}$-competitive ratio. We next settle a set-version of the problem, called maximum k-(set)-coverage problem. For this problem we present an algorithm that improves upon former results for the same model for small and moderate values of k.

1 Introduction

In the *maximum k-vertex-coverage* (MkVC) problem we are given a graph $G = (V, E)$ ($|V| = n$, $|E| = m$) and an integer k, and we ask for a subset $A \subseteq V$, such that $|A| = k$ and the number of edges covered by A is maximized. The MkVC problem is NP-hard, since otherwise the optimal solution for the vertex cover problem could be found in polynomial time: for each k, $1 \leq k \leq n$, run the algorithm for the MkVC problem and stop when all elements are covered.

In this paper we consider the following online model for this problem: at each step i, a new vertex v_i with its adjacent edges is revealed, and we have to decide whether we will include v_i in the solution or discard it. At any time of the process, only k vertices can be kept in memory, so if at some point the current solution already contains k vertices, any inclusion of any new vertex in the solution must be compensated with the definite deletion of one vertex of the current solution. Of course, a vertex that is not kept when it is revealed is also definitely deleted. To our knowledge, no online model for the MkVC problem has been studied until now.

* Research supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010.

O. Owe, M. Steffen and J.A. Telle (Eds.): FCT 2011, LNCS 6914, pp. 181–192, 2011.

A generalization of the MkVC problem is the *maximum k-(set)-coverage* (denoted by MkC) problem, where given a universe of elements $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, a collection of subsets of $\mathcal{E}$, $S = \{S_1, S_2, \ldots, S_n\}$, and an integer $k \leq n$, we ask for a subcollection $A = \{A_1, A_2, \ldots, A_{|A|}\} \subseteq S$, such that $|A| = k$ and the number of elements of E covered by A is maximized. The online model for the MkC problem is the same as for the MkVC.

Clearly, the MkVC problem is a special case of the MkC problem where: (i) each element belongs to exactly two sets and (ii) the intersection of any two sets of S has size at most one, since multiple edges are not permitted.

The weighted generalization of the MkC problem, denoted by WEIGHTED MkC, has been also studied in the literature. In this problem, each element $e_i \in \mathcal{E}$ has a non-negative weight $w(e_i)$, and the goal is to maximize the total weight of the elements covered by k sets.

The analogous online model for WEIGHTED MkC problem, where at each step i a set $S_i \in S$ together with its elements is revealed and only k such sets can be kept in memory, has been studied in [1], where an algorithm of competitive ratio $\frac{1}{4}$ is given. The authors in their so called *set-streaming model* assume that the universe of the instance is known *a priori*. Nevertheless, they do not use this information in the proposed algorithm.

In the classic offline setting, the MkC problem is known to be non approximable within a factor $1 - \frac{1}{e}$ [2]. On the other hand, even for the weighted version of the problem, an approximation algorithm of ratio $1 - \left(1 - \frac{1}{k}\right)^k$ is known [3]. This ratio tends to $1 - \frac{1}{e}$ as k increases, closing in this way the approximability question for the problem.

In [4] the inverse problem (i.e., the hitting set version of MkC), also called *maximum coverage problem*, has been studied: given a universe of elements $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, a collection of subsets of $\mathcal{E}$, $S = \{S_1, S_2, \ldots, S_n\}$, a non-negative weight $w(S_i)$ for each $S_i \in S$, and an integer k, a set $B \subseteq \mathcal{E}$ is sought, such that $|B| = k$ and the total weight of the sets in S that intersect with B is maximized. It is easy to see that this version is equivalent to the WEIGHTED MkC modulo the interchange of the roles between set-system and universe of elements. An algorithm of approximation ratio $1 - \left(1 - \frac{1}{p}\right)^p$ is presented in [4] for this problem, where p is the cardinality of the largest set in S. In the case where each set has cardinality equal to two then this problem coincides with the MkVC problem; hence a $\frac{3}{4}$ approximation ratio is implied by the algorithm in [4]. Several improvements for some restricted cases of the MkVC problem are presented in [5,6]. The MkVC problem is NP-complete in bipartite graphs (by a reduction from the *densest k-subgraph* problem [7]). Moreover, it is polynomial in trees by an application of the dynamic programming algorithm for the quadratic $0 - 1$ knapsack problem presented in [8] using appropriate weights.

In this paper we study the online model described above for both the MkVC and the MkC problems. In Section 2, we prove several negative results on the competitiveness of any algorithm for the model handled for both problems. In Section 3, we present algorithms for regular graphs, regular bipartite graphs, trees and chains, achieving non-trivial competitive ratios, improving upon an

easy $\frac{1}{2}$ competitiveness result holding for any graph. Finally, in Section 4 the k-(set)-coverage problem is handled. For this problem, we present an algorithm that improves upon former results for the same model for small and moderate values of k.

The following notations will be used in the sequel. They are based upon the definition of the MkVC problem and are easily extendable to the MkC problem.

For any $A \subseteq V$, we denote by $E(A)$ the set of edges covered by A and by $m(A) = |E(A)|$ the number of these edges. Let $SOL = m(A)$ be the number of edges covered by our algorithms. Moreover, we denote by $A^* \subseteq V$ an optimal subset of vertices and by $OPT = m(A^*)$ the number of edges covered by an optimal solution. The maximum degree (or the degree when it is regular) of the input graph $G = (V, E)$ is denoted by Δ. Dealing with MkC, Δ denotes the cardinality of a set of maximum size, that is, $\Delta = \max\{|S_i| : 1 \leq i \leq n\}$. For a subset $A \subseteq V$ and a vertex $v_i \in A$, we call *public* the edges incident to v_i and to another vertex in A and *private* the edges of v_i that are covered just by v_i in A. Finally, as it is common in the online setting, the quality of an algorithm is measured by means of the so-called *competitive ratio* representing the ratio of the value of the solution computed by the algorithm over the optimal value of the whole instance, i.e., the value of an optimal (offline) solution of the final instance.

Detailed proofs of the results are given in [9].

2 Negative Results

In this section we give negative results for the online maximum k-vertex-coverage problem and their corresponding adaptations for the maximum k-coverage problem. We start with a negative result for the case where we don't allow any "swaps", i.e., where the replacement of a vertex or set that belongs to the current solution by the newly revealed vertex or set is not permitted.

Proposition 1. *Any deterministic online algorithm that does not allow swaps cannot achieve a competitive ratio better than* $O\left(\frac{1}{(n-1)^{1/(k+1)}}\right)$*, for the* M$k$VC *problem, and better than* $O\left(\frac{1}{m^{1/(k+1)}}\right)$*, for the* M$k$C *problem.*

The next negative result for the MkVC problem fits the model addressed in the paper (swaps are allowed).

Proposition 2. *Any deterministic online algorithm cannot achieve a competitive ratio better than* $\frac{2k}{3k-2} \simeq \frac{2}{3}$ *for the* MkVC *problem.*

Proof. Assume that $2k-1$ vertices, $v_1^1, v_2^1, \ldots, v_{2k-1}^1$, of degree one and $2k-1$ vertices, $v_1^2, v_2^2, \ldots, v_{2k-1}^2$, of degree two are released, such that $(v_i^1, v_i^2) \in E$, $1 \leq i \leq 2k-1$, and that the algorithm selects $k' \leq k$ of them. Wlog, let $v_1^2, v_2^2, \ldots, v_{k'}^2$ be the vertices selected by the algorithm. Next the vertex v_3 of degree k' is released, where $(v_i^2, v_3) \in E$, $1 \leq i \leq k'$. The solution of the algorithm at this time is $2k'$, while the inclusion or not of v_3 does not play any role for

this value. Finally, $2k-1-k'$ vertices, $v^3_{k'+1}, v^3_{k'+2}, \ldots, v^3_{2k-1}$, of degree one are released, such that $(v^2_i, v^3_i) \in E$, $k'+1 \leq i \leq 2k-1$. In this last phase, the algorithm can increase its solution by at most $k-k'$ more edges. Hence, the final solution of the algorithm is at most $k+k'$. The optimum solution consists of the vertices $v^2_{k+1}, v^2_{k+2}, \ldots, v^2_{2k-1}, v_3$, and hence is of cardinality $2(k-1)+k'$. In all, $\frac{SOL}{OPT} = \frac{k+k'}{2(k-1)+k'} \leq \frac{2k}{3k-2}$. □

An analogous result can be proved for the MkC problem. Recall that for the offline version of the MkC problem an $1-\frac{1}{e} \simeq 0.63$-inapproximability result is known [2].

Proposition 3. *Any deterministic online algorithm cannot achieve a competitive ratio better than* $\frac{k+2\sqrt{k}+1}{2k+2\sqrt{k}+1} \simeq \frac{1}{2}$ *for the* MkC *problem even in the case where all sets have the same cardinality.*

3 Maximum *k*-Vertex-Coverage

In this section we deal with the online maximum k-vertex-coverage problem. Note, first, that there exists an easy $\frac{1}{2}$-competitive ratio for this problem. In fact, consider selecting k vertices of largest degrees. In an optimum solution all the edges are, at best, covered once, while in the solution created by this greedy algorithm, all the edges are, at worst, covered twice. Since the algorithm selects the largest degrees of the graph, the $\frac{1}{2}$-competitive ratio is immediately concluded.

Proposition 4. *There is a* $\frac{1}{2}$*-competitive ratio for the online* MkVC *problem.*

In the rest of this section we improve the $\frac{1}{2}$-competitive ratio for several classes of graphs. But first, we give an easy upper bound for the number of elements covered by any solution that will be used later. Its proof is straightforward.

Proposition 5. $OPT \leq k\Delta$.

3.1 Regular Graphs

The following preliminary result that will be used later holds for any algorithm for the MkVC problem in regular graphs.

Proposition 6. *Any deterministic online algorithm achieves a* $\frac{k}{n}$*-competitive ratio for the* MkVC *problem on regular graphs.*

Let us note that the result of Proposition 6 for the MkVC problem also holds for general graphs in the offline setting [6].

We now present an algorithm for the MkVC problem in regular graphs. Our algorithm depends on a parameter x which indicates the improvement on the current solution that a new vertex should entail, in order to be selected for inclusion in the solution. In other words, we replace a vertex of the current solution by the released one, only if the solution increases by at least $\lceil \frac{\Delta}{x} \rceil$ edges.

As we will see in what follows, the best value for x is $x = \frac{n+2k+\sqrt{4k^2+n^2}}{2n}$, leading to the following theorem.

ALGORITHM mkVC-R(x)
1: $A = \emptyset$; $B = \emptyset$;
2: **for** each released vertex v **do**
3: **if** $|A| < k$ **then**
4: $A = A \cup \{v\}$;
5: **if** $|E(\{v\}) \setminus E(B)| \geq \lceil \frac{\Delta}{x} \rceil$ **then**
6: $B = B \cup \{v\}$;
7: **else if** $|B| < k$ **and** $|E(\{v\}) \setminus E(B)| \geq \lceil \frac{\Delta}{x} \rceil$ **then**
8: Select a vertex $u \in A \setminus B$;
9: $A = A \cup \{v\} \setminus \{u\}$; $B = B \cup \{v\}$;
10: **return** A;

Theorem 1. ALGORITHM mkVC-R *achieves* 0.55*-competitive ratio.*

Proof (Sketch). Note that $B \subseteq A$ consists of the vertices that improve the solution by at least $\lceil \frac{\Delta}{x} \rceil$; b denotes the number of these vertices, i.e., $b = |B|$. We denote by y_1 the number of edges with one endpoint in B and the other in $V \setminus B$, and by y_2 the number of edges with both endpoints in B. By definition:

$$SOL \geq y_1 + y_2 = b\Delta - y_2 = \frac{b\Delta - y_1}{2} + y_1 = \frac{b\Delta + y_1}{2} \tag{1}$$

We shall handle two cases, depending on the value of b with respect to k.

If $b < k$ then each vertex $v \in V \setminus B$ is not selected by ALGORITHM mkVC-R(x) to be in B because it is adjacent to at most $\lceil \frac{\Delta}{x} \rceil - 1$ vertices of $V \setminus B$. Thus, there are at least $\Delta - \lceil \frac{\Delta}{x} \rceil + 1$ edges that connect v with vertices in B. Summing up for all the vertices in $V \setminus B$, it holds that $y_1 \geq (n-b)\left(\Delta - \lceil \frac{\Delta}{x} \rceil + 1\right)$, and considering also (1) we get:

$$SOL \geq (n-b)\left(\Delta - \left\lceil \frac{\Delta}{x} \right\rceil + 1\right) + y_2 \tag{2}$$

$$SOL \geq \frac{b\Delta + (n-b)\left(\Delta - \lceil \frac{\Delta}{x} \rceil + 1\right)}{2} \tag{3}$$

Using the upper bound for the optimum provided by Proposition 5 and expressions (2) and (3), respectively, we get the following ratios:

$$\frac{SOL}{OPT} \geq \frac{(n-b)\left(\Delta - \lceil \frac{\Delta}{x} \rceil + 1\right) + y_2}{k\Delta} \geq \frac{n(x-1) - b(x-1)}{kx} \tag{4}$$

$$\frac{SOL}{OPT} \geq \frac{\frac{b\Delta + (n-b)\left(\Delta - \lceil \frac{\Delta}{x} \rceil + 1\right)}{2}}{k\Delta} \geq \frac{n(x-1) + b}{2kx} \tag{5}$$

Observe that the righthand side of (4) decreases with b while that of (5) increases; thus, the worst case occurs when righthand sides of them are equal, that is $\frac{n(x-1)-b(x-1)}{kx} = \frac{n(x-1)+b}{2kx} \Leftrightarrow b = \frac{n(x-1)}{2x-1}$ and hence:

$$\frac{SOL}{OPT} \geq \frac{n(x-1) + \frac{n(x-1)}{2x-1}}{2kx} = \frac{n(x-1)}{k(2x-1)} \tag{6}$$

If $b = k$, then trivially it holds that:

$$\frac{SOL}{OPT} \geq \frac{k\left\lceil \frac{\Delta}{x} \right\rceil}{k\Delta} \geq \frac{1}{x} \tag{7}$$

Note that (6) increases with x while (7) decreases; therefore, for the worst case we have $\frac{n(x-1)}{k(2x-1)} = \frac{1}{x} \Leftrightarrow x = \frac{n+2k+\sqrt{4k^2+n^2}}{2n}$. In all, it holds that:

$$\frac{SOL}{OPT} \geq \frac{2n}{n+2k+\sqrt{4k^2+n^2}} \tag{8}$$

If $k < 0.55n$, the ratio of (8) leads to $\frac{SOL}{OPT} \geq \frac{2n}{n+2(0.55n)+\sqrt{4(0.55n)^2+n^2}} = 0.55$. On the other hand, the ratio provided in Proposition 6 that holds for any algorithm, for $k > 0.55n$, gives $\frac{SOL}{OPT} \geq \frac{k}{n} \geq \frac{0.55n}{n} = 0.55$. □

Let us note that, as it can be easily derived from (8), *when $k = o(n)$ the competitive ratio of* Algorithm m$k$vc-R *is asymptotical to 1.*

3.2 Regular Bipartite Graphs

A better ratio can be achieved if we further restrict in regular bipartite graphs. A key-point of such an improvement is that the maximum independent set can be found in polynomial time in bipartite graphs (see for example [10]). In what follows in this section, we consider that the number of vertices, n, is known a priori.

Our Algorithm m$k$vc-B initializes its solution with the first k released vertices. At this point, a maximum independent set B, of size $b \leq k$, in the graph induced by these k vertices is found. The vertices of this independent set will surely appear in the final solution. For the remaining $k - b$ vertices we check if they cover at least $\frac{\frac{n\Delta}{2}-b\Delta}{\left\lceil \frac{n-b}{k-b} \right\rceil}$ edges different from those covered by the independent set B; if yes, we return the solution consisting of the b vertices of the independent set and these $k - b$ vertices. Otherwise, we wait for the next $k - b$ vertices and we repeat the test. In Algorithm m$k$vc-B, $G[A]$ denotes the subgraph of G induced by the vertex-subset A.

Theorem 2. Algorithm m$k$vc-B *achieves a* 0.6075*-competitive ratio.*

Proof (Sketch). Let us call *batch* the set of the $k-b$ vertices of $A \setminus B$ in Lines 5–10 of Algorithm m$k$vc-B.

The solution computed by this algorithm contains a maximum independent set of size b. Since the input graph is bipartite, it holds that $b \geq \frac{k}{2}$.

The number of edges of the graph uncovered by the vertices of the maximum independent set is in total $\frac{n\Delta}{2} - b\Delta$. Any of these edges is covered by vertices belonging to at least one of the $\left\lceil \frac{n-b}{k-b} \right\rceil$ batches. Hence, in average, each batch covers $\frac{\frac{n\Delta}{2}-b\Delta}{\left\lceil \frac{n-b}{k-b} \right\rceil}$ of those edges; so there exists a batch that covers at least $\frac{\frac{n\Delta}{2}-b\Delta}{\left\lceil \frac{n-b}{k-b} \right\rceil}$ of

ALGORITHM MkVC-B
1: $A = \{$the first k released vertices$\}$;
2: Find a maximum independent set $B \subseteq A$ in $G[A]$; $b = |B|$;
3: **for** each released vertex v **do**
4: **if** $|A| = k$ **then**
5: **if** $m(A) \geq b\Delta + \frac{\frac{n\Delta}{2} - b\Delta}{\left\lceil \frac{n-b}{k-b} \right\rceil}$ **then**
6: **return** A;
7: **else**
8: $A = B$;
9: **else**
10: $A = A \cup \{v\}$
11: **return** A;

them. Therefore, the algorithm covers in total at least $b\Delta + \frac{\frac{n\Delta}{2} - b\Delta}{\left\lceil \frac{n-b}{k-b} \right\rceil}$ edges. Using Proposition 5, we get $\frac{SOL}{OPT} \geq \frac{b\Delta + \frac{\frac{n\Delta}{2} - b\Delta}{\left\lceil \frac{n-b}{k-b} \right\rceil}}{k\Delta} = \frac{b + \frac{\frac{n}{2} - b}{\left\lceil \frac{n-b}{k-b} \right\rceil}}{k}$ and since this quantity increases with b it holds that:

$$\frac{SOL}{OPT} \geq \frac{\frac{k}{2} + \frac{\frac{n}{2} - \frac{k}{2}}{\left\lceil \frac{n-\frac{k}{2}}{k-\frac{k}{2}} \right\rceil}}{k} = \frac{k + \frac{n-k}{\left\lceil \frac{2n-k}{k} \right\rceil}}{2k} \tag{9}$$

If $k \leq 0.6075n$, then (9) leads to $\frac{SOL}{OPT} \geq 0.6075$. Otherwise, using Proposition 6 we get the same ratio and the theorem is concluded. □

Note that by (9), ALGORITHM MkVC-B *achieves a competitive ratio asymptotical to* $\frac{3}{4}$ *when* $k = o(n)$.

3.3 Trees and Chains

In this section we give algorithms that further improve the competitive ratios for the MkVC problem in trees and chains. Dealing with trees the following result holds.

Proposition 7. *The* MkVC *problem can be solved within* $\left(1 - \frac{k-1}{\Delta^*}\right)$*-competitive ratio in trees, where* Δ^* *is the sum of the* k *largest degrees in the tree. The ratio is tight.*

Note that, if the number of vertices of degree greater than 1 is $r < k$ then our algorithm finds an optimum solution using just r vertices, since the edges that are adjacent to the leaves are covered by their other endpoints.

Furthermore, in the case where all the internal vertices of the tree have the same degree Δ, the ratio provided by Proposition 7 becomes $\left(1 - \frac{k-1}{k\Delta}\right)$. This ratio is better than the ratio proved for regular bipartite graphs in Theorem 2 for any $\Delta \geq 3$, but it is worse for $\Delta = 2$, i.e., in the case where the input graph is a chain.

An improvement for the MkVC problem in chains follows. The main idea of the algorithm is to partition the solution, A, into two disjoint parts, whose size is dynamically adjusted: the set B of vertices that contribute two edges in $E(A)$ and the set C of vertices that contribute one edge in $E(A)$.

ALGORITHM MkVC-C

1: $A = \emptyset$; $B = \emptyset$; $C = \emptyset$; In any step $A \equiv B \cup C$;
2: **for** each released vertex v **do**
3: **if** $|B| \leq k$ **and** v adds two new edges to the solution **then**
4: **if** $|A| = k$ **then**
5: Delete an arbitrary vertex from C;
6: $B = B \cup \{v\}$;
7: **else if** $|A| < k$ **and** v adds one new edge to the solution **then**
8: $C = C \cup \{v\}$;
9: **if** the inclusion of v in A has as a result three consecutive vertices to appear in A **then**
10: Move v from C to B; Remove the middle vertex from A;
11: **return** A;

Proposition 8. *For the* MkVC *problem in chains,* ALGORITHM MkVC-C *returns the (offline) optimum, if* $k < \lceil \frac{n}{3} \rceil$ *or* $k \geq \lceil \frac{2n}{3} \rceil$*, and achieves a* 0.75*-competitive ratio, if* $\lceil \frac{n}{3} \rceil \leq k < \lceil \frac{2n}{3} \rceil$.

4 Maximum k-(set)-Coverage

In this section we present ALGORITHM MkC for the online maximum k-(set)-coverage problem. It initializes by selecting the first k released sets. Then, each time a new set P is released, the algorithm will update the current solution A_j only if for some suitably selected set Q from A_j, the solution obtained after having in A_j the set Q replaced by P covers at least $m(A_j)\left(\frac{k+1}{k}\right)$ elements. We prove that ALGORITHM MkC achieves competitive ratio strictly greater than $\frac{1}{4}$, tending to $\frac{1}{4}$ as k increases. Recall that the algorithm presented in [1] achieves also an $\frac{1}{4}$-competitive ratio. However, our analysis is tight and gives better results for moderately large values of k.

To analyze ALGORITHM MkC, let A_z be the solution computed after having all the sets released, i.e., $SOL = m(A_z)$. Fix also, an optimum solution A^*.

ALGORITHM MkC

1: $j = 1$; $A_j = \{$the first k released sets$\}$;
2: **for** each released set P **do**
3: Find the set $Q \in A_j$ that covers privately the smallest number of elements in A_j;
4: **if** $m(A_j \setminus \{Q\} \cup \{P\}) > m(A_j) + \frac{m(A_j)}{k}$ **then**
5: $j = j + 1$; $A_j = A_{j-1} \setminus \{Q\} \cup \{P\}$;

We distinguish the following two types of *bad events* that may happen during the execution of the algorithm upon arrival of a set P:

(a) $P \in A^*$ and ALGORITHM MKC does not select it, and
(b) $P \notin A^*$ and ALGORITHM MKC discards $Q \in A^*$ in order to insert P into its current solution.

The total number of bad events of both types is a measure of the distance between A_z and A^*; clearly, at most k such events may happen. Notice also that more than one bad event of type (a) may happen while the current solution is kept unchanged, i.e. may correspond to some value of j in the algorithm, while at most one bad event of type (b) might correspond to it; so, let $\ell = |\{j :$ some bad event of any type happens when the current solution is $A_j\}|$. Let A_{j_i}, $1 \le i \le \ell$, $1 \le j_i \le z$, be the i-th of these current solutions, and k_i, $1 \le i \le \ell$, be the number of events occurred with A_{j_i} being the current solution.

We will now provide an upper bound to $OPT = m(A^*)$ by some expression involving these "event-stroken" A_{j_i}s, $1 \le i \le \ell$. Consider that the s-th bad event corresponds to j_i, i.e., $\sum_{r=1}^{i-1} k_r < s \le \sum_{r=1}^{i} k_r$. Let P_s be the new set that arrives while the current solution is A_{j_i} and Q_s be the set that covers privately the smallest number of elements in A_{j_i}. Let, also, $\widetilde{Q}_s \subseteq Q_s$ be the set of private elements of Q_s in A_{j_i}.

If the event is of type (a) then $P_s \in A^*$ is not selected and it covers a subset of the elements in $E(A_{j_i} \setminus \{Q_s\})$ plus its private elements, $\widetilde{P}_s \subseteq P_s$, in $E(A_{j_i} \setminus \{Q_s\} \cup \{P_s\})$. Note that it is $m(\widetilde{P}_s) \le m(\widetilde{Q}_s) + \frac{m(A_{j_i})}{k}$, otherwise P_s would be selected by the algorithm. Moreover, $m(\widetilde{Q}_s) \le \frac{m(A_{j_i})}{k}$, since Q_s has the smallest private part in A_{j_i}, and hence $m(\widetilde{P}_s) \le \frac{2m(A_{j_i})}{k}$.

In all we get the following inclusion relation $E(A^*) \subseteq \bigcup_{i=1}^{\ell} E(A_{j_i}) \cup \bigcup_s \widetilde{P}_s$ with s varying on the indices of (a)-type bad events (remark that the sets of the optimum removed after a (b)-type bad event are always represented in the first term of the expression, since the union varies among all i for A_{j_i}s). This reduces trivially to $E(A^*) \subseteq E(A_{j_\ell}) \cup \bigcup_{i=2}^{\ell} \left[E(A_{j_{i-1}}) \setminus E(A_{j_i})\right] \cup \bigcup_s \widetilde{P}_s$. Thus, for the value of the optimum solution A^* we have the following bound:

$$OPT \le m(A_{j_\ell}) + \sum_{i=2}^{\ell} m(E(A_{j_{i-1}}) \setminus E(A_{j_i})) + \sum_{i=1}^{\ell} \left(k_i \frac{2m(A_{j_i})}{k}\right)$$

Claim 1. $m(E(A_{j_{i-1}}) \setminus E(A_{j_i})) \le \frac{|A_{j_{i-1}} \setminus A_{j_i}|}{k} m(A_{j_{i-1}})$, $2 \le i \le \ell$.

Using Claim 1 and since $m(A_{j_\ell}) > m(A_{j_i})$, $1 \le i \le \ell - 1$, and $\sum_{i=1}^{\ell} k_i = k$ we get:

$$OPT \le m(A_{j_\ell}) + \sum_{i=2}^{\ell} \frac{|A_{j_{i-1}} \setminus A_{j_i}|}{k} m(A_{j_{i-1}}) + \sum_{i=1}^{\ell-1} \frac{2m(A_{j_i})}{k} + (k - \ell + 1)\frac{2m(A_{j_\ell})}{k}$$

By definition, it holds that $j_\ell \leq z$ and hence $m(A_{j_\ell}) \leq m(A_z) = SOL$. Moreover, by ALGORITHM MkC, $m(A_{j_\ell}) \geq \left(1+\frac{1}{k}\right)^{j_\ell - j_i} m(A_{j_i})$. Thus, we have:

$$\frac{SOL}{OPT} \geq \frac{1}{3 + \frac{1}{k}\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}+2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} - \frac{2(\ell-1)}{k}} \tag{10}$$

Claim 2. *For any $\ell \geq 2$, it holds that $\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}+2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} \leq \frac{g(\ell)}{\ln\left(1+\frac{1}{k}\right)}$, where $g(\ell) = \frac{\left(1+\frac{1}{k}\right)^2}{e} \cdot e^{g(\ell-1)}$ and $g(2) = \frac{\left(1+\frac{1}{k}\right)^2}{e}$.*

Using Claim 2 and (10), we get $\frac{SOL}{OPT} \geq \frac{1}{3+\frac{1}{k}\left[\frac{g(\ell)}{\ln\left(1+\frac{1}{k}\right)} - 2(\ell-1)\right]}$, where $g(\ell) = \frac{\left(1+\frac{1}{k}\right)^2}{e} \cdot e^{g(\ell-1)}$ and $g(2) = \frac{\left(1+\frac{1}{k}\right)^2}{e}$. This quantity is minimized for some $\ell = o(k)$. The ratio r achieved by ALGORITHM MkC for different values of k is shown in Table 1.

Table 1. Approximation ratio of ALGORITHM MkC

k	2	3	5	10	30	50	100	300	500	1000
r	0.333	0.324	0.314	0.300	0.282	0.275	0.268	0.261	0.258	0.256

To see that the ratio achieved by ALGORITHM MkC is always greater than $\frac{1}{4}$, consider the following expression for the ratio, slightly coarser than (10):

$$\frac{SOL}{OPT} \geq \frac{1}{3 + \frac{1}{k}\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} + \frac{1}{k}\sum_{i=2}^{\ell}\left(\frac{2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} - 2\right)}$$

Note first that if $\ell = 1$ then both sums in the denominator become zero and hence we have a $\frac{1}{3}$-competitive ratio. For $\ell \geq 2$ we may proceed to the following analysis. For the first sum, by a similar argument as in Claim 2 we can prove that $\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} \leq \frac{g(\ell)}{\ln\left(1+\frac{1}{k}\right)}$, where $g(\ell) = \frac{1}{e} \cdot e^{g(\ell-1)}$ and $g(2) = \frac{1}{e}$. It is easy to see by simple induction that $g(\ell) \leq 1$ for any $\ell \geq 2$ and hence $\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} \leq \frac{1}{\ln\left(1+\frac{1}{k}\right)} \leq k$. For the second sum, we have:

$$\sum_{i=2}^{\ell}\left(\frac{2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} - 2\right) \leq \sum_{i=2}^{2}\left(\frac{2}{\left(1+\frac{1}{k}\right)} - 2\right) = \frac{2k}{k+1} - 2 = -\frac{2}{k+1}$$

Therefore, using these bounds to the ratio we get $\frac{SOL}{OPT} \geq \frac{1}{4} + \frac{1}{4}\frac{1}{2k(k+1)-1}$.

It is hopefully clear from the previous discussion, that the analysis of ALGORITHM MkC works as well for the WEIGHTED MkC problem, up to the assumption that $m(\cdot)$ in ALGORITHM MkC denotes the total weight of the elements rather than their number.

We conclude this section by providing a tight example for the ratio achieved by ALGORITHM MkC. The idea of the example strongly relies upon the proof given above, which indicates the "critical" values of ℓ and j_i, $1 \leq i \leq \ell$. For simplicity, we will consider a case where $k = 3$, but it is easy to extend our example for any k, by appropriately choosing values for ℓ and j_i.

For $k = 3$, one can see that the ratio of ALGORITHM MkC is minimized when $\ell = 2$ and $j_2 - j_1 = 1$. Hence, consider the scenario shown in Figure 1. Let

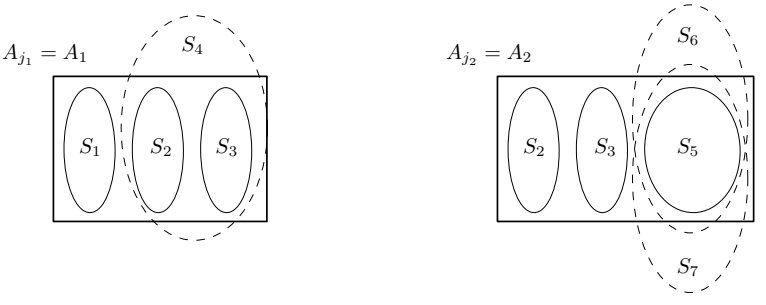

Fig. 1. A tight example for ALGORITHM MkC when $k = 3$

$A_1 = \{S_1, S_2, S_3\}$ be the solution after the first three sets have been released. These sets are disjoint and each one covers c elements. Next, the set S_4 appears, which covers $2c - \epsilon$ new elements plus all elements in S_2 and S_3. The algorithm does not select S_4, since it may choose S_1 as a candidate for swapping; in this case the new solution would cover $m(\{S_2, S_3, S_4\}) = 4c - \epsilon$ elements which is smaller than $m(A_1) + \frac{m(A_1)}{k} = 4c$. Then, the set S_5 is released, which is disjoint to the previous sets and covers $2c$ elements. Thus, the algorithm replaces S_1 by S_5, and the new solution is $A_2 = \{S_2, S_3, S_5\}$. Finally, S_6 and S_7 are released, each one covering the elements in S_5 plus $\frac{8c}{3} - \epsilon$ new elements. ALGORITHM MkC does not select any of them, since they do not satisfy the algorithm's criterion.

So, the final solution, A_2, covers $m(S_2) + m(S_3) + m(S_5) = 4c$ elements. The optimal solution consists of sets S_4, S_6 and S_7 and covers $OPT = (4c - \epsilon) + (2c + \frac{8c}{3} - \epsilon) + (\frac{8c}{3} - \epsilon) = \frac{34c}{3} - \epsilon$ elements. Therefore, the ratio achieved by ALGORITHM MkC is $\frac{SOL}{OPT} = \frac{4c}{\frac{34c}{3} - \epsilon} \simeq \frac{12}{34} = 0.353$.

Note that the gap between this ratio and the ratio 0.324 given in Table 1 is due to the fact that the elements of S_1 do not appear to the optimal solution. Indeed, if S_1 was included to the optimal, then $OPT = \frac{37c}{3} - \epsilon$ and $\frac{SOL}{OPT} = \frac{4c}{\frac{37c}{3} - \epsilon} \simeq \frac{12}{37} = 0.324$. This gap decreases as $k \to \infty$.

5 Conclusions

There exist several interesting questions arising from the results presented in this paper. The first of them is to improve the easy $\frac{1}{2}$-competitive ratio for MkVC in

general graphs and the (less easy) worst-case $\frac{1}{4}$-competitive ratio in set systems. Another open question is to provide tighter upper bounds for the on-line model handled in regular graphs. We still do not see how one can improve the analysis of Algorithm mkc in the case of equal cardinalities, or how to tighten the upper bound of Proposition 3 in Section 2, in order to match (or to get closer to) the competitive ratio of Algorithm mkc. Let us note that an algorithm in the spirit of Algorithm m$k$vc-R of Section 3.1 for the case of equal-cardinality sets, only achieves ratio $\frac{1}{\sqrt{k}}$.

References

1. Saha, B., Getoor, L.: On maximum coverage in the streaming model & application to multi-topic blog-watch. In: DM 2009, pp. 697–708. SIAM, Philadelphia (2009)
2. Feige, U.: A threshold of $\ln n$ for approximating set cover. Journal of the ACM 45, 634–652 (1998)
3. Hochbaum, D.S., Pathria, A.: Analysis of the greedy approach in problems of maximum k-coverage. Naval Research Logistics 45, 615–627 (1998)
4. Ageev, A.A., Sviridenko, M.I.: Approximation algorithms for maximum coverage and max cut with given sizes of parts. In: Cornuéjols, G., Burkard, R.E., Woeginger, G.J. (eds.) IPCO 1999. LNCS, vol. 1610, pp. 17–30. Springer, Heidelberg (1999)
5. Feige, U., Langberg, M.: Approximation algorithms for maximization problems arising in graph partitioning. Journal of Algorithms 41, 174–211 (2001)
6. Han, Q., Ye, Y., Zhang, H., Zhang, J.: On approximation of max-vertex-cover. European Journal of Operational Research 143, 342–355 (2002)
7. Corneil, D.G., Perl, Y.: Clustering and domination in perfect graphs. Discrete Applied Mathematics 9, 27–39 (1984)
8. Rader Jr., D.J., Woeginger, G.J.: The quadratic 0-1 knapsack problem with series-parallel support. Operations Research Letters 30, 159–166 (2002)
9. Ausiello, G., Boria, N., Giannakos, A., Lucarelli, G., Paschos, V.T.: Online maximum k-coverage. Technical Report 299, Université Paris-Dauphine, Cahiers du LAMSADE (2010)
10. Paschos, V.T.: A survey of approximately optimal solutions to some covering and packing problems. ACM Computing Surveys 29, 171–209 (1997)