Independent Set with Advice: The Impact of Graph Knowledge
(Extended Abstract)

Stefan Dobrev[1], Rastislav Královič[2], and Richard Královič[3,4]

[1] Institute of Mathematics, Slovak Academy of Sciences, Bratislava, Slovakia
[2] Department of Computer Science, Comenius University, Bratislava, Slovakia
[3] ETH Zurich, Switzerland
[4] Google Zurich, Switzerland

Abstract. We are interested in online graph problems where the knowledge of the underlying graph G (all arriving vertices are from G) has a profound impact on the size of the advice needed to solve the problem efficiently. On one hand, we show that, for sparse graphs, constant-size advice is sufficient to solve the maximum independent set problem with constant competitive ratio, even with no knowledge of the underlying graph. On the other hand, we show a lower bound of $\Omega(\log(n/a)/\log\log(n/a))$ on the competitive ratio of finding a maximum independent set in bipartite graphs if no knowledge of the underlying graph is available and if the advice is of size a. We complement the lower bounds by providing corresponding upper bounds.

1 Exposition and Motivation

Given a simple undirected graph $G = (V, E)$, a subset $I \subseteq V$ is called *independent* if the subgraph induced by I does not contain any edges, i.e., $\forall u, v \in I : \{u, v\} \notin E$. The problem of finding the independent set of maximum cardinality (MIS) is one of the most studied computational problems on graphs with applications in many areas ranging from computer vision, to coding theory, molecular biology, scheduling, or wireless networking, to name just a few. However, this problem is, in general, computationally hard: in [21] it is proven that, unless $NP = ZPP$, the problem[1] cannot be approximated within a factor of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ (i.e., there is no polynomial-time algorithm that would always find an independent set of size at least $opt/n^{1-\varepsilon}$).

We are interested in the *online* version of the problem. In online problems, the input is not known to the algorithm at the beginning, instead, it arrives piecewise. The algorithm must present a partial output to each chunk of the input before reading next chunk, and cannot revise its decision afterwards. More formally, the input $\boldsymbol{x}$ is a sequence of *requests* $\boldsymbol{x} = (x_1, \ldots, x_n)$. The output $\boldsymbol{y}$ is a sequence of *answers* $\boldsymbol{y} = (y_1, \ldots, y_n)$ computed by the algorithm in such a

[1] Actually, the paper proves inapproximability of Maximum-Clique problem, where the aim is to find the subset of vertices with largest cardinality that form a clique. However, the two problems are obviously equivalent w.r.t. approximation.

T. Erlebach and G. Persiano (Eds.): WAOA 2012, LNCS 7846, pp. 2–15, 2013.

way that each y_i is a function of $x_1, \ldots, x_i$ (for randomized algorithms also a function of the random bits used so far). The goal is to maximize or minimize a *cost* function defined over the whole output $\boldsymbol{y}$. A standard way of measuring the quality of online algorithms is to compare the (worst case) performance of the algorithm to the optimal solution of the instance (i.e., with known input). For maximization problems, the definition of *competitive ratio* states that an algorithm is *c-competitive* if for any input the cost of the output produced by the algorithm is at least[2] $1/c \cdot opt$. Note that in online problems the main concern is not the computational complexity, but the inherent loss of performance due to the unknown future. Online computation has received considerable attention over the past decades as a natural way of modeling real-time processing of data. For example, in resource scheduling problems a server has to handle a stream of requests (e.g., phone calls), each of them requiring certain subset of available resources (e.g., lines). The goal of maximizing the throughput of the server leads naturally to an online version of MIS (see e.g., [25] in the context of scheduling intervals). For an exposition to online problems, we refer the reader to [5].

When dealing with the online graph problems (and in particular with the online MIS), there are several ways how the input can be presented to the algorithm. A natural way (which we shall refer to as *unknown graph* model) is to present the graph vertex by vertex: Each time a new vertex v_i arrives, the algorithm learns the edges connecting v_i to already presented vertices $v_i, \ldots, v_{i-1}$. Before the next vertex v_{i+1} is presented, the algorithm must decide whether the vertex v_i will be included in the resulting independent set or not. Obviously, no deterministic algorithm can attain a good competitive ratio: consider an instance where at the beginning, a isolated vertices are presented. If the algorithm does not include any one of them to the independent set, no more vertices arrive. On the other hand, if the algorithm includes some of them to the independent set, $b \gg a$ vertices arrive such that they form a complete bipartite graph with the first a vertices. In [6] authors show that the competitive ratio of online MIS on the class of σ-bounded disk graphs is $\Theta(\min\{n, \sigma^2\})$ where the upper bound is attained by a simple first fit algorithm, and the lower bound holds also for randomized algorithms[3].

The inherent difficulty of the unknown graph model led to the study of various other models. In [2], authors use what we shall refer to as the *known supergraph* model: the graph G presented to the algorithm is a subgraph of some larger graph H that is known in advance to the online algorithm. Thus the situation is as follows: first, the graph $H = (V', E')$ is presented to the algorithm with $V' = v_1, \ldots, v_m$. Each request $x_i \in \{1, \ldots, m\}$ represents a vertex, and the algorithm must decide whether to include vertex v_{x_i} in the independent set or not. The optimal solution is the largest independent set of the subgraph

[2] Some works use a slightly relaxed definition by allowing an additive constant, i.e., the algorithm is c-competitive if there exists a constant α such that the cost of the worst-case output (for randomized algorithms the worst-case expected output) is at least $1/c \cdot opt - \alpha$.

[3] If the disk representation is given, the bound is $\Theta(\min\{n, \log \sigma\})$.

of H induced by the presented vertices. This model was motivated by routing problems in networks, where the topology of the network is usually known to the algorithms but the routing requests are not. It was proven in [2] that, even if the algorithms can use randomization, the competitive ratio is $\Omega(n^\varepsilon)$ for some constant ε. The conference version of the paper, [1], mentions a similar lower bound even in the case when preemption is allowed (i.e., the algorithm may, at a later stage, remove a vertex from the independent set in construction, but it cannot be reinserted). In [7], an $O(\log n)$-competitive algorithm for online MIS in chordal graphs is presented using the known supergraph model.

A number of other modifications of the unknown graph model has been studied. In [20], a model is considered where the algorithm is allowed to maintain multiple independent sets under construction, and each vertex can be assigned to at most $r(n)$ different sets. The largest set is chosen as the final output. The competitive ratio in this model is shown to be $\Theta(n/\log n)$ when $r(n)$ is polynomial in n, and $\Theta(n)$ if $r(n)$ is constant. In a more powerful *inheritance* variation, a lower bound $\Omega(n/\log^3 n)$ is proven for polynomial $r(n)$. The graphs used as lower bounds are split graphs (vertex set can be partitioned into an independent set and a clique), and at the same time interval graphs (subclass of chordal graphs). In yet another model from [13] the algorithm starts from a complete graph on n vertices, and each request removes an edge; the algorithm is allowed to add to the constructed independent set endpoint(s) of the removed edge. Again, in this model, a competitive ratio is bounded from below by $(n-1)/2$ and Δ where Δ is the maximum degree of the resulting graph. Still another model, from [9], considers the requests containing not one vertex, but a subset of vertices (together with all induced edges); the authors give an algorithm with a competitive ratio $O(n\sqrt{t(n)}/\log n)$, where $t(n)$ is the number of requests, and n is the number of vertices. In [19], a variation of the unknown graph model, in which the algorithm is at the beginning presented with a graph isomorphic to the presented graph G, is studied in the context of online coloring problems. The author also addresses the independent set: the algorithm is required to produce an online coloring of the input graph, and the largest color class is taken as the solution. Even with this relaxation, the competitive ratio of deterministic algorithms is proven to be $\Omega(n)$, and $\Theta(n/\log n)$ for randomized algorithms against oblivious adversary.

Several modifications of the unknown graph model share the same high-level idea: to enhance the online algorithm with some a-priori information about the input, either in general by using restricted class of input graphs or per-instance by providing a supergraph or an isomorphic graph. In the context of online and distributed algorithms where a crucial role is played by some missing information (about topology of the communication network or the future input), the notion of *advice* has recently become popular as a general way to treat the additional information. The idea of the approach is to augment the algorithm by some information about the instance, and to study the relation of the amount of the added information to the solution quality. There are no computational

constraints on the added information, only that it is a function of the instance and the algorithm at hand.

In the context of online algorithms, the original model from [11] has been developed in two ways: the model from [12] considers that the algorithm receives, with each request, a b-bit string of advice (a function of the whole input and the algorithm). Hence, the case with $b = 0$ corresponds to the classical online model, and the case with $b = \lceil \log |\mathcal{A}| \rceil$, where $\mathcal{A}$ is the action space of the algorithm, always gives an optimal algorithm (since the particular answer may be encoded in the advice of any request). In the model from [4] (see also [3, 22]), the whole advice is given to the algorithm at the beginning in a form of a binary string. A trivial upper bound, apart from specifying each action in $n\lceil \log |\mathcal{A}| \rceil$ bits, is to encode the whole input using the advice of size that corresponds to the Kolmogorov complexity of the input instance. However, as it is shown in [3, 4, 22], these trivial bounds can be in many cases substantially improved.

Our choice for the second model is mainly due to the fact that it makes it possible to analyze sublinear advice. While for problems where there are at most two possible actions for each request (as is the case of MIS; every vertex is either included in the set or not), one bit of advice per request is already sufficient to achieve optimum, it is interesting to know what can be done with smaller advice. We show, e.g., that in the case of bipartite graphs, $O(\log \log n)$ bits of advice already bring the competitive ratio down from $\Omega(n)$ to $O(\log n)$.

In a related context of distributed algorithms, in [14, 16–18, 23] the advice is a function that assigns a binary string to each node of a communication network. The only information about the network known to each node is its local information, and the advice string. The parameter under consideration is either the sum or the maximum of the lengths of the strings in all nodes, and the main question is how much advice is needed to (efficiently) perform communication tasks such as broadcasting in (radio) networks, graph searching, computing a proper vertex coloring of the network, or computing the spanning tree of the network. In [10, 15], there is a mobile entity (agent) performing some task in the network (e.g., exploration), and the advice is a binary string given to the agent based on the topology.

Our Contribution. In this paper we focus on the online MIS in the unknown graph and known supergraph models. While the first intuition is that the knowledge of the supergraph may be significant additional information[4] , it is easy to see that it does not affect the worst-case performance of deterministic algorithms, even restricted to a class of forests (which are planar and bipartite). Indeed, let the underlying graph be a forest containing m stars with k leaves. First, centers of the stars $c_1, c_2, \ldots$ are presented to the algorithm one by one until it selects some c_i for the first time. Then the leaves of the i-th star are presented, and the input ends. Because the competitive ratio is parameterized by the size of the presented graph G (and not in the size of the supergraph H),

[4] Although without any restrictions on the structure or size of the supergraph, one can always construct a "universal" supergraph that fools any deterministic algorithm; with randomized algorithms, however, the situation is more subtle (see [2]).

it can be easily verified that, even in the milder definition involving the additive constant, the competitive ratio is $\Omega(n)$.

We are interested in the question to what extent the known supergraph helps in reducing the amount of advice needed to obtain some given solution quality. We use the model of advice from [3]: at the beginning, the algorithm can access the result of any function of the actual input that returns a binary string of length $s(n)$. Based on this advice, the algorithm then proceeds as a standard online algorithm. Note that this approach is similar to [19] and [20], however, in our case there are $2^{s(n)}$ partial solutions constructed, and the largest one is chosen as output.

We show that in the class of sparse graphs, the knowledge of the supergraph does not help substantially: even in the unknown graph model, advice of constant size is sufficient to obtain constant competitive ratio on graphs with $O(n)$ edges. The situation is, however, different in the class of bipartite graphs. With the known supergraph, a competitive ratio of 2 can be achieved with one bit just by specifying the bipartition with the majority of presented vertices. On the other hand, we show that in the unknown graph model the competitive ratio of any deterministic algorithm with $s(n)$ bits of advice is at least $\Omega(\log(n/s(n))/\log\log(n/s(n)))$. The bounds are complemented with the corresponding opposite bounds.

2 Sparse Graphs

First let us observe the simple fact that if there is a way to construct a proper vertex coloring (i.e. a coloring of vertices where each edge has the endpoints colored by different colors) in the respective online model, then the $s(n)$ bits of advice can specify the largest of the first $2^{s(n)}$ color classes.

Theorem 1. *Let $\mathcal{G}$ be a class of graphs such that each n-vertex graph $G \in \mathcal{G}$ is online colorable (in the respective model) in such a way that the union of the first $k(n)$ color classes contains at least $\alpha(n)$ vertices. Then there is an online MIS algorithm using $\lceil \log k(n) \rceil$ bits of advice with competitive ratio $n\,k(n)/\alpha(n)$.*

In the known supergraph model the presented graph is a subgraph of a known graph. Hence, if the supergraph is k-colorable, we immediately have results for the advice complexity; e.g., 2 bits of advice are sufficient to reach competitive ratio of 4 if the known supergraph is planar (since planar graphs are 4-colorable). There are numerous results on (online) vertex coloring that translate to the advice bounds in a similar way, see e.g., [8, 24, 26] and references therein.

Although sparse graphs (even trees) are not online colorable with constant number of colors, the first fit (FF) algorithm where a new vertex is assigned the smallest color different from the (assigned) colors of its neighbors, is good enough to apply Theorem 1 on sparse graphs in the unknown graph model.

Lemma 2. *Let G be a n-vertex graph with at most cn edges. The* FF *algorithm produces in the unknown graph model a coloring of G such that the union of color classes $1, 2, \ldots, 2c$ contains at least $n/2$ vertices.*

Proof. Let us call an edge e a *forcing edge* if there is a vertex v such that

- when v arrives, it is assigned color larger than $2c$, and
- e is an edge incident to v, leading to a vertex that has arrived before

Let S be the set of vertices given a color larger then $2c$ when FF is run on a graph with at most cn edges. As the sets of forcing edges of different vertices are disjoint and each vertex from S has at least $2c$ forcing edges, it must hold $2c|S| \leq cn$, hence $|S| \leq n/2$. This means that the number of vertices of color at most $2c$ is at least $n/2$. □

Corollary 3. *Online MIS can be solved in the unknown graph model on graphs with at most cn edges using advice of* $1 + \lceil \log c \rceil$ *bits and achieving competitive ratio* $4c$.

Hence, for c independent of n, with $O(\log c)$ bits of advice, it is possible to reach a competitive ratio of $O(c)$ on graphs with at most cn edges. However, improving the ratio to $o(c)$ requires $\Omega(n)$ bits:

Theorem 4. *Let* A *be an algorithm solving online MIS in the unknown graph model on graphs with at most cn edges, and with* $s(n)$ *bits of advice. Then the worst-case competitive ratio of* A *is at least* $\frac{c}{2c\frac{s(n)}{n} + \frac{2c}{n}(1+2\log c)+1}$.

3 Lower Bound for Bipartite Graphs

Recall that in the known supergraph model, one bit of advice is sufficient to achieve competitive ratio 2 on bipartite graphs. The main result of this section is the following theorem:

Theorem 5. *Consider any subadditive function* $s(n) < n$, *and any algorithm* A *solving online MIS in the unknown graph model on bipartite graphs with advice* $s(n)$. *Then the worst-case competitive ratio of* A *over n-vertex bipartite graphs is* $\Omega\left(\frac{\log(n/s(n))}{\log\log(n/s(n))}\right)$.

Hence, not only it is not possible to achieve a constant competitive ratio using constant advice, but $\Omega(n)$ bits of advice are needed to do so.

To prove the theorem we construct a class of bipartite graphs $\{G^{k,a}\}$, such that the number of vertices of $G^{k,a}$ is $n = a(k+1)2^{k+1}$, and the size of maximum independent set of $G^{k,a}$ is $a(k+1)2^k$. We consider a particular set of input instances $\mathcal{C}^{k,a}$ based on a graph $G^{k,a}$ that differ by the order in which the vertices are presented to the algorithm.

Consider an arbitrary algorithm A with at most a bits of advice, and competitive ratio r. Since the advice partitions the set of input instances into 2^a classes such that within each class the instances are processed deterministically, it follows that there is a deterministic algorithm $\mathtt{A}_{det}$, and a subset of $C \subseteq \mathcal{C}^{k,a}$ of size $|C| \geq |\mathcal{C}^{k,a}|/2^a$ such that $\mathtt{A}_{det}$ achieves competitive ratio r over instances from C. The main idea of the proof is to consider an arbitrary deterministic

algorithm $\mathtt{A}_{det}$, and any subset C of size at least $|\mathcal{C}^{k,a}|/2^a$, and to show that $\mathtt{A}_{det}$ constructs an independent set on size $O(a2^k \log k)$ on some instance from C.

The theorem the follows by chosing k to be the largest integer such that $\phi(k) \leq \sqrt{\frac{n}{s(n)+1}}$, where $\phi(x) := (x+1)2^{x+1}$, and letting $a = \left\lceil \frac{n}{\phi(k)} \right\rceil$. The graph $G^{k,a}$ then has $a\phi(k) \geq n$ vertices, and the competitive ratio is $\Omega\left(\frac{\log(\phi(k))}{\log\log(\phi(k))}\right)$. Due to space constraints, we omit the computations.

The lower bound graph $G^{k,a}$ (we shall omit the superscripts from now on) is a bipartite graph with partitions L and R, consisting of a copies of graph G_ϵ (ϵ denoting the empty string) $G_\epsilon(1), G_\epsilon(2), \ldots G_\epsilon(a)$. Each G_ϵ is constructed hierarchically from components G_α for $\alpha \in \{0,1\}^{*k}$ as follows (see Fig. 1):

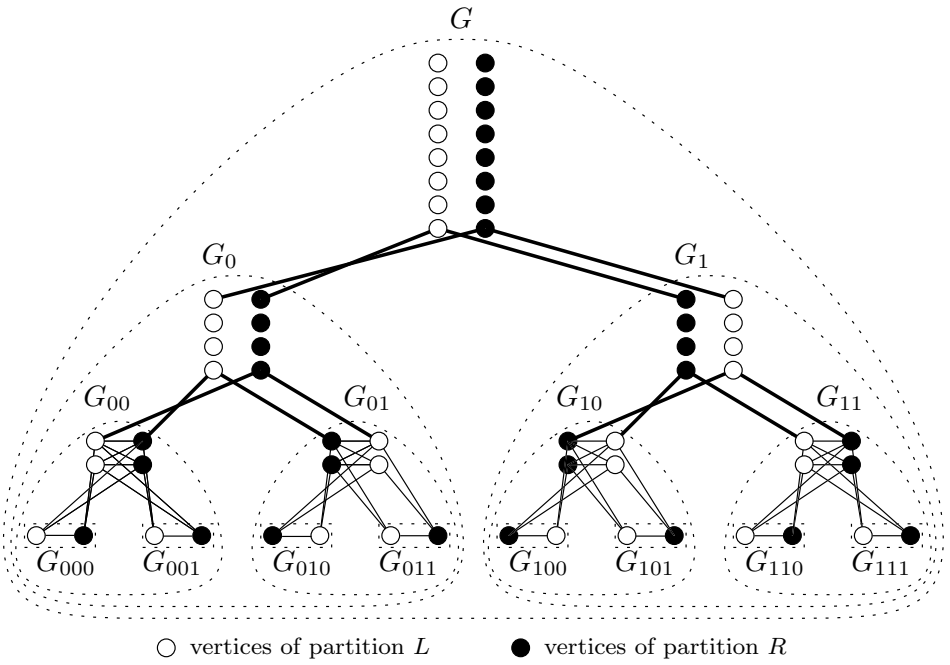

Fig. 1. The basic element G_ϵ of the lower bound graph. All edges are shown between vertices of level 1 and 0. At higher levels only schematic connections to subtrees are shown, the edges to all vertices of the opposite partition in the whole subtree are still present. The left vertex/column arrives before the corresponding right vertex/column.

- Level 0: G_α where $\alpha \in \{0,1\}^k$ consists of a single edge
- Level i: G_α where $\alpha \in \{0,1\}^{k-i}$ is obtained by
  - taking two level $i-1$ graphs $G_{\alpha 0}$ and $G_{\alpha 1}$
  - adding two sets of 2^i vertices $L_\alpha = \{l_j\}_{j=1}^{2^i} \subset L$ and $R_\alpha = \{r_j\}_{j=1}^{2^i} \subset R$
  - connecting l_j to all vertices from R in $G_{\alpha 0} \bigcup G_{\alpha 1} \bigcup R_\alpha$
  - connecting r_j to all vertices from L in $G_{\alpha 0} \bigcup G_{\alpha 1} \bigcup L_\alpha$

The construction is repeated for k levels. Note that at each level exactly 2^{k+1} vertices are added, resulting in $(k+1)2^{k+1}$ vertices in G_ϵ and $n = a(k+1)2^{k+1}$. Each graph G_ϵ can be represented by a complete binary tree H_ϵ of depth k, where the leaves correspond to G_α with $|\alpha| = k$, and each internal vertex $v_\alpha \in H_\epsilon$ represents the union of L_α and R_α. To distinguish between the vertices of the graph and vertices of the tree, we shall call the latter *nodes*. H_α will denote the subtree rooted at v_α. Let $H = \bigcup_{i=1}^{a} H_\epsilon(i)$. The *weight* $w(v)$ of each node $v \in H$ is the size of the maximum independent set of the set of vertices associated with v. Hence $w(v) = 2^i$ for nodes at level i. Clearly, $w(v)$ is an upper bound on how much the nodes of v can contribute to the independent set produced by the algorithm.

The instances $\mathcal{C}^{k,a}$ are defined by the order in which the vertices are presented to the algorithm. For each node $v \in H$, the associated vertices of G arrive as a block, not interleaved with vertices associated with other nodes of H. For each leaf v in H, there are two possible orders for the vertices associated with v to arrive. We assign label $l(v) \in \{0, 1\}$, based on whether the first vertex of v to arrive is from L or from R, respectively. Taken over all leaves, this represents 2^{a2^k} possible configurations. The label of an internal node $v_\alpha \in H$ is equal to the label of the leftmost leaf in H_α (i.e., the first vertex to arrive in H_α). The order of arrival of the vertices associated with an internal node v_α of H is fixed: First to arrive are the vertices of the same partition as was the first to arrive in $v_{\alpha 0}$. The nodes also arrive in fixed order, with the vertices of $H_\epsilon(i)$ arriving in postorder, and before the vertices of $H_\epsilon(i+1)$. The set $\mathcal{C}^{k,a}$ is hence fully described by a configuration of the leaf vertices. Since it is irrelevant which partition is L and which is R and the graphs $G_\epsilon(i)$ are disjoint, we have $|\mathcal{C}^{k,a}| = 2^{a(2^k-1)}$.

Let us now consider an arbitrary deterministic algorithm $\mathtt{A}_{det}$, and an arbitrary subset $C \subseteq \mathcal{C}^{k,a}$ of of size $|C| \geq |\mathcal{C}^{k,a}|/2^a = 2^{a(2^k-2)}$. We show that within C there exists an instance on which $\mathtt{A}_{det}$ selects an independent set of size $O(a2^k \log(k))$. We prove this by presenting Algorithm 1 constructing the configuration for which we can prove the desired bound.

In order to proceed with the arguments, we need to introduce the following notation:

- Node v_α is *dead* at time t if the algorithm has already accepted vertices from different partitions in G_α. A node which is not dead is *alive*.
- Node v_α (or subtree H_α) is *empty* at time t if the algorithm has not selected any vertex from G_α into the independent set. Otherwise v_α (H_α) is *full*.

Suppose first that $\mathtt{A}_{det}$ selects exactly one vertex from each leaf node of H, and C initially contains all instances $\mathcal{C}^{k,a}$. Consider now a leaf node v_α and its closest ancestor v_β such that $v_\alpha \in H_{\beta 1}$. Since $\mathtt{A}_{det}$ selects a vertex from every leaf, $v_{\beta 0}$ is full and in half of the instances the vertices selected in v_α and $v_{\beta 0}$ are in different partitions, making v_β and all its ancestors dead. Applying this argument over all $a(2^k - 1)$ such leaf nodes (there are only a leafs which have no such v_β as they are the leftmost ones in their respective H_ϵ) yields a configuration in which $\mathtt{A}_{det}$ has selected only $a2^k$ vertices.

In general,the situation is complicated by the fact that $\mathtt{A}_{det}$ may decide, in order to avoid killing a low-level node, not to select any vertex from a given node. Moreover, since the initial C is an arbitrary subset of $\mathcal{C}^{k,a}$, the fraction of instances that kill the parent after selecting some vertex from a given node may not be one half.

The finding of a bad instance can be viewed as a game between the algorithm and an adversary that selects the orientation of each presented node[5]. On one hand, the adversary tries to kill as many nodes as possible, as only alive nodes contribute to the independent set, on the other hand it must keep the working set of configurations sufficiently large. The adversary handles this by using a *threshold* t, whose distance (measured on logarithmic scale) from $1/2$ is directly proportional to the size of the node that would be killed: If the fraction of instances consistent with the orientation that kills the node is at least t, the adversary sets that orientation. Otherwise, it lets the node survive in order to retain enough instances in its working set. The analysis uses an accounting scheme, where the algorithm starts with a *credit* a, and gets credit when a vertex is killed, but must pay *cost* to ensure a vertex survives; at the end the account must be non-negative and the weight of the alive nodes determines the size of the independent set.

We shall now present the above ideas more formally. Let I denote the vertices accepted by the algorithm into the independent set. The next lemma shows that the algorithm cannot win too much by not selecting vertices from a presented node. In fact, for the analysis it is sufficient to consider the "decision" nodes W' that are alive and have both children full.

Lemma 6. *Let W denote the union of the sets of full alive nodes and full leaves and let W' be those nodes from W whose both children are full. Let $w(S)$ denote the sum of weights of the nodes in S. Then it holds $3w(W')+2^{k+1} \geq w(W) \geq |I|$.*

Proof. Consider a node $v \in W \setminus W'$. We charge the weight of v to its closest ancestor $u \in W'$. The nodes v form two paths (one in each child subtree of v), since branching at a node u' would mean that u' is a closer ancestor to v than u. As the weights double in each layer, the weight of each of these paths sums up to at most $w(u)$. This way, each node of W' is charged at most three times its weight.

Let W'' be the set of nodes from $W \setminus W'$ which do not have an ancestor in W'. We have $3w(W') + w(W'') > w(W)$. We show by induction on height h that for each $v \in W''$ of level h, $w(W'' \bigcap H_v) \leq 2^{h+1}$. In the base case $h = 0$ we have a single leaf of weight 2, hence the statement trivially follows. Consider now an internal node v at level h. There are two cases: either v is alive or it is dead. In the case v is alive, from the fact that $v \notin W'$ it follows that one of its children is empty. Hence $w(W'' \bigcap H_v) = w(v) + w(W'' \bigcap H_{v'})$,

[5] For a leaf node, this means selecting the order in which the vertices arrive. For an internal node, this means flipping the orientation of one child's subtree. Note that since the partitions L and R are symmetric, this can be done without affecting the algorithm's behavior.

Algorithm 1. Finding the bad instance

1: Let C be the set of configurations for which a given deterministic algorithm applies.
2: **for** all nodes $v_\alpha \in H_\epsilon(i)$ in the traversal order specified above **do**
3: **if** v_α is not a leaf **then**
4: Let C_0 and C_1 be the partitioning of C into sets of configurations consistent with
$l(v_{\alpha 0}) = l(v_{\alpha 1})$ and $l(v_{\alpha 0}) \neq l(v_{\alpha 1})$, respectively.
5: **if** v_α is dead, or ($v_{\alpha 0}$ or $v_{\alpha 1}$ is empty) **then**
6: Set C to be the larger of C_0 and C_1.
7: **else** ▷ hence v_α is alive and both of its children are full.
8: Let C_x be the configurations in which the algorithm has selected in G_α vertices from both partitions.
9: Let j be the level of the highest-level live ancestor of v_α.
10: Set $t(j)$ satisfying $\log(t_j) = -1 - 1/2^{k-j}$.
11: **if** $|C_x|/|C| > t(j)$ **then** ▷ note that C_x might be empty
12: Set $C \leftarrow C_x$. ▷ this kills v_α and all its ancestors
13: **else**
14: Set $C \leftarrow C_{1-x}$. ▷ v_α survives, but C has not decreased much
15: **end if**
16: **end if**
17: **end if**
18: Deliver the vertices of $G_\alpha(i)$ and observe algorithm's action.
19: **end for**
20: Output the configuration consistent with what has happened so far.

where v' is the full child of v'. By induction hypothesis $w(W'' \bigcap H_{v'}) \leq 2^h$. As $w(v) = 2^h$, the induction step follows. In the case of v being dead, let v' and v'' be its two children. Applying the induction hypothesis yields $w(W'' \bigcap H(v)) = w(W'' \bigcap H(v')) + w(W'' \bigcap H(v'')) \leq 2^h + 2^h = 2^{h+1}$.

Finally, note that in a dead internal node, the algorithm cannot accept any vertices into the independent set. An alive internal node v consists of two partitions, each containing $w(v)$ vertices, hence in v the algorithm can accept at most $w(v)$ vertices. Only in leaves can the algorithm accept vertices even if the node becomes dead – but such leaves are included in W. This yields $w(W) \geq |I|$. □

Let p (pass), d (dead) and l (live) denote the number of times the lines 6, 12 and 14 have been executed, respectively. Let $D = \{v \in H$: line 12 was applied when processing $v\}$ and let $L = \{v \in H$:line 14 was applied when processing $v\}$. Note that L is actually the set of decision nodes W' from Lemma 6, since a node v which remained alive after processing will never be killed later. This is because only v's children can kill v, and those have been processed before v. Let j_v for $v \in (D \cup L)$ be the level j from line 9 when processing v. For a node $v \in L$, let us define $cost(v) = \log(1 - t(j_v)) + 1$. Similarly, for a node $v \in D$, define $credit(v) = -\log(t(j_v)) - 1 = 2^{j-k}$. Analogously, for a set S, define $cost(S) = \sum_{v \in S} cost(v)$ and $credit(S) = \sum_{v \in S} credit(v)$. The next lemma bounds the overall cost that the algorithm may pay:

Lemma 7. $cost(L) < a(k+1)$

Let us now find an upper bound on the size of the independent set accepted by $\mathtt{A}_{det}$. From Lemma 6 we know that it is sufficient to bind $w(L)$. Hence, the problem now becomes "*Maximize* $w(L)$ *while satisfying* $cost(L) < a(k+1)$".

Note that if a node $v \in L$ has an alive parent u which is not in L, the set $L' = L \cup \{u\} \setminus \{v\}$ has the same cost but higher weight. Hence, it is sufficient to consider sets L in which $\forall u, v$ such that u is an ancestor of v, the whole path from v to u is in L. Therefore, L induces in H a set of r rooted trees $\{T_i\}_{i=1}^{r}$. Let v_i is the root of T_i. Then, for every node $u \in T_i$, the highest-level live ancestor of u at the moment u is processed is either v_i or its ancestor, hence $j(u) \geq level(v_i)$ and therefore $cost(T_i) \geq |T_i| cost(v_i)$. As T_i is a subtree of a binary tree and the weights of nodes halve as the levels decrease, we obtain $w(T_i) \leq \log(|T_i|+1)w(v_i)$ Since $cost(.)$ is a concave function, $w(L)$ is maximized when $r = a$ and all v_i are at the top level k, with $|T_i| = (k+1)/cost(v_i) = (k+1)/(\log(3)-1)$. Using $w(v_i) = 2^k$ gives $w(L) \leq a \log\left(\frac{k+1}{\log(3)-1} - 1\right) 2^k \in O(a2^k \log(k).)$ Applying Lemma 6 yields $|I| \in O(a2^k \log(k))$, and Theorem 5 follows.

4 Upper Bound for Bipartite Graphs

While Theorem 5 indicates that quite a lot of advice is needed to approach a competitive ratio one, it seems rather weak when relatively little advice is available. In this section we show that it is not weak at all, and very little advice is sufficient to bring the competitive ratio to $O(\log n)$.

For a fixed input graph G with bipartitions L and R, and a fixed arrival order, let us observe the connected components C_i formed by the vertices presented so far. Each of them is a connected bipartite graph, and the algorithm knows its bipartition. However, it does not know which partition of C_i corresponds to L. Let each C_i have a distinguished partition P_{C_i} maintained by the algorithm. Let us assign to each vertex v, and each respective component, a meta-level $m(v)$ using Algorithm 2.

Consider a component C of meta-level l and let v be the first vertex of meta-level l in C, i.e., the vertex that formed C as a component of meta-level l on line 10 of Algorithm 2. By construction the algorithm can, when creating a new component C, always choose the P_C in a consistent way. Let $ms(i)$ denote the minimal size of a component of meta-level i. By construction, we have recurrences $ms(1) = 1$ and $ms(i+1) \geq 2ms(i)+1$, yielding $ms(i) \geq 2^i - 1$. This immediately gives us:

Lemma 8. $\forall v \in G : ms(v) \leq \log(n+1)$

Let l be a fixed meta-level and let $\{C_1, C_2, \ldots, C_r\}$ be the set of connected components of meta-level l. Let V_i be the set of vertices of meta-level l in C_i (note that V_i might be disconnected) and let $V(l) = \bigcup_{i=1}^{r} V_i$. Let $I(l) = \bigcup_{i=1}^{r} V_i \bigcap P_{C_i}$ and $\bar{I}(l) = \bigcup_{i=1}^{r} V_i \bigcap \overline{P_{C_i}}$, where $\bar{P}$ denotes the opposite partition to P. Since the components C_i are disjoint, both $I(l)$ and $\bar{I}(l)$ are independent sets, even

Algorithm 2. Algorithm Meta-levels

if a newly arriving vertex v is singleton **then**
  Set $m(v) \leftarrow 1$, and $m(\{v\}) \leftarrow 1$
else if v is connected to a single connected component C **then**
  Set $m(v) \leftarrow m(C)$, and $m(C \bigcup \{v\}) \leftarrow m(C)$
else
  Let v be connected to components $C_1, C_2, \ldots, C_s$, ordered by meta-levels in non-increasing order.
  if $m(C_1) > m(C_2)$ **then**
    Set $m(v) \leftarrow m(C_1)$ and $m(\{v\} \bigcup C_1 \bigcup C_2 \bigcup \ldots C_s) \leftarrow m(C_1)$.
  else ▷ $m(C_1) = m(C_2)$
    Set $m(v) \leftarrow m(C_1) + 1$ and $m(\{v\} \bigcup C_1 \bigcup C_2 \bigcup \ldots C_s) \leftarrow m(C_1) + 1$.
  end if
end if

though P_{C_i}'s might belong to different partitions of G. Knowing l, a single bit of advice telling which of $I(l)$ and $\bar{I}(l)$ is bigger is sufficient for the algorithm to select an independent set of size $V(l)/2$.

Let m be the meta-level with the maximal number of vertices. From Lemma 8 we have $V(l) \geq n/\log(n+1)$. Therefore, the algorithm that uses $1 + \log\log n$ bits of advice to identify the meta-level m with the largest $V(m)$, and the larger of $I(m)$ and $\bar{I}(m)$, selects an independent set of size $O(n/\log n)$, yielding.

Theorem 9. *$O(\log\log n)$ bits of advice[6] are sufficient to achieve competitive ratio of $O(\log n)$ for the online independent set in bipartite graphs.*

Consider now the case that a bits of advice are available, with $a \geq \log\log n$. In such case, the algorithm can use this advice to learn the orientation of meta-components and improve the competitive ratio.

Theorem 10. *There is an algorithm that using $a \geq \log\log n$ bits of advice achieves competitive ratio $O(\log(n/a))$ for the online independent set problem in bipartite graphs.*

5 Conclusion

We were interested in the relation of the competitiveness of online MIS in unknown graph and known supergraph models in terms of advice complexity. Without any advice, the competitive ratio is $\Omega(n)$ in both models, even restricted to sparse bipartite graphs. We showed that in sparse graphs, constant advice is sufficient in both models to achieve a constant competitive ratio. In bipartite graphs, however, the models differ significantly since 1 bit of advice is sufficient to achieve competitive ratio 2 in known supergraph model, whereas $\Omega(n)$ bits are needed to achieve a constant competitive ratio in the unknown graph model.

[6] Since n is not known, a self-delimited encoding will be used, at a cost of small constant factor increase in the number of bits used.

The use of the advice as a general way to measure the relevant information about unknown input is an attractive alternative to ad-hoc solutions. Although not practical in its general form (e.g., no computability constraints), it may help in characterizing the key structural properties that affect the performance of online algorithms.

Acknowledgement. The authors acknowledge the support of the VEGA agency under the grants 1/0671/11, and 2/0136/12. The research originated at the "Mountains and Algorithms" workshop organized by Juraj Hromkovič in August, 2011.

References

1. Bartal, Y., Fiat, A., Leonardi, S.: Lower bounds for on-line graph problems with application to on-line circuit and optical routing. In: STOC 1996, pp. 531–540. ACM (1996)
2. Bartal, Y., Fiat, A., Leonardi, S.: Lower bounds for on-line graph problems with application to on-line circuit and optical routing. SIAM J. Comput. 36(2), 354–393 (2006)
3. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R.: On the advice complexity of the k-server problem. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part I. LNCS, vol. 6755, pp. 207–218. Springer, Heidelberg (2011)
4. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R., Mömke, T.: On the advice complexity of online problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 331–340. Springer, Heidelberg (2009)
5. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge Univ. Press (1998)
6. Caragiannis, I., Fishkin, A.V., Kaklamanis, C., Papaioannou, E.: Randomized on-line algorithms and lower bounds for computing large independent sets in disk graphs. Discrete Applied Mathematics 155(2), 119–136 (2007)
7. Christodoulou, G., Zissimopoulos, V.: On-line maximum independent set in chordal graphs. Journal of Foundations of Computing and Decision Sciences 30(4), 283–296 (2005)
8. Cieślik, I.: On-line Graph Coloring. PhD thesis, Jagiellonian University, Krakow, Poland (2004)
9. Demange, M., Paradon, X., Paschos, V.T.: On-line maximum-order induced hereditary subgraph problems. In: Jeffery, K., Hlaváč, V., Wiedermann, J. (eds.) SOFSEM 2000. LNCS, vol. 1963, pp. 327–335. Springer, Heidelberg (2000)
10. Dereniowski, D., Pelc, A.: Drawing maps with advice. J. Par. Distrib. Comput. 72(2), 132–143 (2012)
11. Dobrev, S., Královič, R., Pardubská, D.: Measuring the problem-relevant information in input. ITA 43(3), 585–613 (2009)
12. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. Theor. Comput. Sci. 412(24), 2642–2656 (2011)
13. Escoffier, B., Paschos, V.T.: On-line models and algorithms for max independent set. RAIRO - Operations Research 40(2), 129–142 (2006)
14. Fraigniaud, P., Gavoille, C., Ilcinkas, D., Pelc, A.: Distributed computing with advice: information sensitivity of graph coloring. Distributed Computing 21(6), 395–403 (2009)

15. Fraigniaud, P., Ilcinkas, D., Pelc, A.: Tree exploration with advice. Inf. Comput. 206(11), 1276–1287 (2008)
16. Fraigniaud, P., Ilcinkas, D., Pelc, A.: Communication algorithms with advice. J. Comput. Syst. Sci. 76(3-4), 222–232 (2010)
17. Fraigniaud, P., Korman, A., Lebhar, E.: Local mst computation with short advice. Theory Comput. Syst. 47(4), 920–933 (2010)
18. Fusco, E.G., Pelc, A.: Trade-offs between the size of advice and broadcasting time in trees. Algorithmica 60(4), 719–734 (2011)
19. Halldórsson, M.M.: Online coloring known graphs. In: SODA 1999, pp. 917–918. ACM/SIAM (1999)
20. Halldórsson, M.M., Iwama, K., Miyazaki, S., Taketomi, S.: Online independent sets. Theor. Comput. Sci. 289(2), 953–962 (2002)
21. Håstad, J.: Clique is hard to approximate within $n^{1-\varepsilon}$. Acta Mathematica 182, 105–142 (1999)
22. Hromkovič, J., Královič, R., Královič, R.: Information complexity of online problems. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 24–36. Springer, Heidelberg (2010)
23. Ilcinkas, D., Kowalski, D.R., Pelc, A.: Fast radio broadcasting with advice. Theor. Comput. Sci. 411(14-15), 1544–1557 (2010)
24. Kubale, M.: Graph colorings. Contemporary Mathematics, vol. 352. AMS (2004)
25. Lipton, R.J., Tomkins, A.: Online interval scheduling. In: SODA 1994, pp. 302–311 (1994)
26. Randerath, B., Schiermeyer, I.: Vertex colouring and forbidden subgraphs - a survey. Graphs and Combinatorics 20(1), 1–40 (2004)