Lower Bounds for On-line Graph Problems with Application to On-line Circuit and Optical Routing *

Yair Bartal † Amos Fiat ‡ Stefano Leonardi §

Abstract

We present lower bounds on the competitive ratio of randomized algorithms for a wide class of on-line graph optimization problems and we apply such results to on-line virtual circuit and optical routing problems.

Lund and Yannakakis [LY93a] give inapproximability results for the problem of finding the largest vertex induced subgraph satisfying any non-trivial, hereditary, property π. E.g., independent set, planar, acyclic, bipartite, etc. We consider the on-line version of this family of problems, where some graph G is fixed and some subgraph H is presented on-line, vertex by vertex. The on-line algorithm must choose a subset of the vertices of H, choosing or rejecting a vertex when it is presented, whose vertex induced subgraph satisfies property π. Furthermore, we study the on-line version of graph coloring whose off-line version has also been shown to be inapproximable [LY93b], on-line max edge-disjoint paths and on-line path coloring problems.

Irrespective of the time complexity, we show an $\Omega(n^{\epsilon})$ lower bound on the competitive ratio of randomized on-line algorithms for any of these problems.

*A preliminary version of this work appears in the Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 1996.

†International Computer Science institute, Berkeley. Research supported in part by the Rothchild Postdoctoral fellowship. e-mail: yairb@icsi.berkeley.edu

‡Department of Computer Science, Tel Aviv University, Tel Aviv. Research supported in part by two grants from the Israel Academy of Sciences. e-mail: fiat@math.tau.ac.il

§International Computer Science Institute (Berkeley) & Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza". This work is partly supported by EU ESPRIT Long Term Research Project ALCOM-IT under contract n 20244, and by Italian Ministry of Scientific Research Project 40% "Algoritmi, Modelli di Calcolo e Strutture Informative". e-mail: leon@dis.uniroma1.it

As a consequence, we obtain an $\Omega(n^\epsilon)$ lower bound on the competitive ratio of randomized on-line algorithms for virtual circuit routing on general networks, in contrast to the known results for some specific networks. Moreover, this lower bound holds even if the use of *preemption* is allowed. Similar lower bounds are obtained for on-line optical routing as well.

1 Introduction

1.1 On-line graph problems

On-line graph optimization problems have been previously considered in several works: the problem of obtaining a maximal on-line matching in a bipartite graph is considered in [KVV90], the on-line coloring of a three colorable graph is studied in [V90], the problem of coloring an inductive graph is studied in [I90], and randomized lower bounds for on-line graph coloring are given in [HS92].

In all the previous papers, vertices and/or edges arrive on-line, and the on-line algorithm has to make some relevant decision when this happens. The adversary has total freedom to decide what graph will be presented, and in what order the edges/vertices will arrive.

In this paper, we consider a different model, where the graph presented by the adversary is a subgraph of some (larger) graph that is *known in advance* to the on-line algorithm. Upper bounds that hold for the models of [KVV90, V90, I90, HS92] clearly also hold for the model where the presented graph is a subgraph of some known graph. Lower bounds do not. Lower bounds for the unknown graph model can be trivially translated into our model simply by considering a much larger graph that contains all possible subgraphs. However, the large graph is now exponentially larger than the subgraph, which implies that the lower bound is exponentially smaller.

The motivation for this model it is not only theoretical. It allows to capture problems such as on-line routing in networks. We effectively translate routing problems on graph problems on a graph related to the original network. It is reasonable to assume that the on-line routing algorithm does not know anything about future communication requests, it is not reasonable to assume that it does not know anything about the routing network itself.

We consider a large set of on-line graph optimization problems. Some of the problems are benefit problems (*e.g.*, find the largest subgraph satisfying a property), others are cost problems (*e.g.*, color a graph with as few colors as possible).

The graph optimization problems we consider are listed below, for each such problem we first give the off-line version, and then give the on-line variant. The performance of an algorithm for an on-line problem is measured by its *competitive ratio* [ST85]. The competitive ratio for benefit problems is defined as the worst case ratio between the optimal off-line benefit and the on-line benefit. For cost problems the competitive ratio is defined as the worst case ratio between the on-line cost and the optimal off-line cost.

Off-line Induced Subgraph: Let π be a graph property that is *non-trivial* (e.g. true for infinitely many graphs and false for infinitely many graphs) and *hereditary* (e.g., if a graph G satisfies it then also any vertex-induced subgraph of G satisfies it). Given a graph $G = (V, E)$, find the subset of V of maximum cardinality whose vertex-induced subgraph satisfies π.

Such problems include complete graph, independent set, k-colorable, planar, outerplanar, bipartite, complete bipartite, acyclic, degree-constrained, interval, circular-arc, circle graph, chordal, perfect, etc.

Lund and Yannakakis [LY93a] give inapproximability results for this family of problems. Note that in on-line computation, time complexity is not considered so that our results do not follow from [LY93a].

On-line Induced Subgraph: The graph $G = (V, E)$ is known to the on-line algorithm. Vertices $v \in V$ are presented one by one. The adversary may choose to stop the sequence at any time. The on-line algorithm has to decide if to accept v or not. The induced graph of the accepted vertices must satisfy property π. The benefit of the on-line algorithm is the cardinality of the set of accepted vertices.

We define the *On-line Independent Set* problem as the on-line induced subgraph problem for the hereditary property $\pi =$ *independent set*.

Off-line Graph Coloring: Given a graph $G = (V, E)$, color its vertices with as few colors as possible so that no two adjacent nodes receive the same color.

Lund and Yannakakis [LY93b] also prove inapproximability results for minimum graph coloring.

On-line Graph Coloring: The graph $G = (V, E)$ is known to the algorithm. At every step a new vertex $v \in V$ is presented to the on-line algorithm, the vertex is immediately assigned a color. The adversary may choose to stop the sequence at any time. The coloring must be valid. The cost of the on-line algorithm is given by the number of colors used.

The on-line graph coloring problem has been studied extensively, and deterministic and randomized algorithms have been proposed, but only in the setting where the graph is unknown [I90, V90, HS92].

Off-line Edge-disjoint Paths: The instance is a graph $G = (V, E)$ and a collection $\mathcal{C} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs of vertices. We say that a set of pairs $\{(s_{i_j}, t_{i_j})\}_{j=1}^{m}$ is *consistent* if there exist edge disjoint paths $\{p_{i_j}\}_{j=1}^{m}$, where p_{i_ℓ} connects s_{i_ℓ} to t_{i_ℓ}. The problem is to find a maximum cardinality consistent subset of $\mathcal{C}$.

On-line Edge-disjoint Paths: The graph G is known to the algorithm, at every step the adversary presents a pair of vertices to the on-line algorithm. The on-line algorithm may accept or reject this pair, but the set of all accepted pairs must be consistent. The benefit of the on-line algorithm is here defined as the cardinality of the set of accepted pairs.

A variation of this problem is the Edge-disjoint Fixed Paths problem, where an entire path rather than just its endpoints is given. The goal is to accept a maximal set of pairwise non-overlapping paths.

Off-line Path Coloring: Given a graph $G = (V, E)$, and a collection of $\mathcal{C} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of k pairs of vertices in G, what is the minimum number of consistent sets of pairs so that their union gives $\mathcal{C}$? This is equivalent to minimizing the number of colors required to color paths between the k endpoints (s_i, t_i) so that no edge is contained in two paths colored with the same color.

On-line Path Coloring: The graph $G = (V, E)$ is known to the on-line algorithm, at every step the adversary presents a pair of vertices. The on-line algorithm chooses a color for the two vertices. The set of pairs accepted with the same color has to be consistent. The on-line cost is the number of colors used.

For all of the problems above we prove a lower bound of $\Omega(n^\epsilon)$ on the competitive ratio of any on-line algorithm, for some $\epsilon > 0$. This is fairly easy to show for deterministic algorithms, our main contribution is showing that this holds even for randomized on-line algorithms against an oblivious adversary (cf. [BBKTW90]).

Our lower bounds for the on-line independent set and the on-line edge-disjoint paths problems also hold when in the context of benefit problems the use of *preemption* is allowed to the randomized algorithm. This means that the on-line algorithm is allowed to discard a previously chosen element (a vertex or a pair) but cannot later change its mind again.

We complement these lower bounds results by presenting an algorithm for the on-line induced subgraph problem that achieves a competitive ratio of $O(\frac{n}{\sqrt{\alpha}})$ given a black-box $O(\frac{n}{\alpha})$-approximation algorithm for the induced subgraph problem, and an $O(\frac{n}{\sqrt{\alpha}})$-competitive algorithm for graph coloring if an $O(\frac{n}{\alpha})$-approximation algorithm for independent set is available.

1.2 Virtual circuit routing problems

The benefit or throughput version of the virtual circuit routing problem is where communication requests require the assignment of a specified bandwidth on a path that connects a transmitter to a receiver, for a given duration and offering a given benefit. Each call can be either accepted and scheduled observing the bandwidth constraints on all links, or rejected. Following [GG92] we call this problem the *Call Control* problem.

[GGKMY93] considered the call control problem when the algorithm is allowed to preempt previously accepted calls. They give deterministic preemptive algorithms with a competitive ratio of $O(\log n)$ for a line network.

Non-preemptive call control was first considered in [AAP93]. They gave throughput competitive algorithms, $O(\log n)$ competitive, for arbitrary networks, (not requiring preemption), if every call request bandwidth was for no more than a logarithmic fraction of the minimal link capacity. They also prove an $\Omega(n)$ lower bound on the competitive ratio of deterministic on-line algorithms on general networks if calls may request arbitrary bandwidth.

[ABFR94] considered randomized non-preemptive algorithms for call control, on tree networks, getting a competitive ratio of $O(\log n)$ when all edge capacities are 1, all bandwidth requests are 1, all durations are infinite, and all benefits are 1. [ABFR94] also show that problem parameters such as variable bandwidth, call benefit and call duration can be dealt with by a randomized algorithm at the expense of logarithmic factors in the competitive ratio.

The call control problem when all links have capacity 1, all calls require bandwidth 1, duration is infinite, and all call benefits are 1, is thus the central problem to be considered. This problem is equivalent to the edge-disjoint paths graph problem stated above. Thus, any lower bound on the competitive ratio for edge-disjoint paths is a lower bound for the call control problem. Similarly, an upper bound on the competitive ratio for edge-disjoint paths gives an upper bound on the more general problem using the techniques of [ABFR94].

[AGLR94] give a randomized call control algorithm for the mesh, with a competitive ratio of $O(\log^2 n)$. More recently, [KT95] gave an $O(\log n)$ competitive randomized call control algorithm for the mesh and extended this result to more general "densely embedded nearly Euclidean" planar graphs.

[KS95] recently claimed an $O(\log n)$ randomized upper bound for the on-line call control problem when the degree of every node in the network is $\Omega(\log n)$.

We show an $\Omega(n^\epsilon)$ lower bound on the competitive ratio of any randomized on-line algorithm for general networks even if call preemption is allowed. We give specific networks where this occurs, including a network of degree $\Omega(\log n)$, contradicting the claims of [KS95].

1.3 Routing problems in optical networks

In optical networks the bandwidth available on any link is split into many channels, each at a different wavelength. Each transmission is associated with exactly one wavelength that must be available on all links forming a path from the transmitter to the receiver.

The routing problem on optical networks differs considerably depending on the network model considered. There are a number of models, differing in the nature of network switches.

The basic classification is into

- Non-reconfigurable networks – also called passive or *switchless* networks,– where a wavelength originating from a transmitting node will always follow a fixed pattern. There may be "switches" that mix and match incoming wavelengths to outgoing links, but these cannot be changed during the operation of the network.
- Reconfigurable networks, that use two different types of switches:
  - Elementary switches, that can direct signals coming along one of the input lines to one or more output links, but cannot differentiate between wavelengths when performing the switch.
  - Generalized switches, that are capable of splitting the incoming stream based upon the wavelength.

In all models, the key restriction is that if more than one data item is transmitted along an edge on the same wavelength (color), then they are all unusable.

Optical routing was considered in [RU94, ABC$^+$94, BH93, P92]. Most of the work focuses on the question of how many wavelengths are required to accommodate a given communication traffic.

We consider a number of on-line optical routing problems on both *reconfigurable* optical networks, where all vertices are generalized switches, and *switchless* optical networks. We consider benefit problems, where calls can be accepted or rejected, analogous to the call control problem, and coloring problems whose goal is to minimize the number of wavelengths (colors) used.

On-line Routing in Reconfigurable Optical Networks — Benefit Version: Calls are presented one by one, calls can be either accepted or rejected. Every call can be routed using the generalized switches along a path to be chosen. All calls have to be transmitted along a single wavelength and thus no two calls can share an edge. This problem is now identical to on-line edge-disjoint paths.

On-line Routing in Reconfigurable Optical Networks – Coloring Version: Calls are presented one by one, all calls have to be accepted, assigned a wavelength and a path linking source to destination. The goal is to minimize the number of wavelengths used as to preserve the rule that no two calls on *the same wavelength* can share an edge. The problem is identical to on-line path coloring.

In the switchless cases described below, the routing decisions are very restricted. What does have to be decided is the wavelength to be assigned to any call so that calls do not interfere with one another. Networks are acyclic in order to prevent self-interference of transmissions, and the optical beam is directed in one of the two directions.

Off-line Routing in Switchless Optical Networks — Benefit Version: Consider a directed graph $D = (V, E)$ and a collection $\mathcal{C} = \{(s_i, t_i), \ldots, (s_k, t_k)\}$ of k pairs of vertices. Denote by $R(v)$, the set of vertices of V reachable from v in D via a directed path. We say two pairs (s_i, t_i), (s_j, t_j) of terminal vertices are *non-interfering* if $t_j \notin R(s_i)$ and $t_i \notin R(s_j)$. The problem on a single wavelength consists of finding a maximum cardinality pairwise non-interfering subset of $\mathcal{C}$.

On-line Routing in Switchless Optical Networks – Benefit Version: The graph D is known to the algorithm. At every step a pair is communicated to the on-line algorithm that can either accept or reject it, the set of accepted pairs should be pairwise non-interfering. The benefit of the on-line algorithm is the cardinality of the set of accepted calls.

Off-line Routing in Switchless Optical Networks – Coloring Version: Given a directed graph $D = (V, E)$, and a collection $\mathcal{C} = \{(s_i, t_i), \ldots, (s_k, t_k)\}$ of pairs of vertices, what is the minimum number of pairwise non-interfering sets of pairs so that their union gives C? This can be seen as to minimize the number of colors necessary to color the pairs in $\mathcal{C}$, where two pairs with the same color must be non-interfering.

On-line Routing in Switchless Optical Networks — Coloring Version: The on-line algorithm knows some graph D. At each step a new pair of vertices is chosen by the adversary and assigned a color by the on-line algorithm. The on-line algorithm assigns a color so that the pair is non-interfering with all pairs previously assigned with the same color. The on-line cost is the number of colors used.

We show for all these on-line optical routing problems an $\Omega(n^\epsilon)$ lower bound on the competitive ratio of randomized algorithms.

The results for reconfigurable networks are directly derived from the analogous results for on-line edge disjoint paths and on-line path coloring.

The results for switchless networks are derived by showing that independent set is a subproblem of routing in switchless optical networks — benefit version, and that graph coloring is a subproblem of routing in optical switchless networks — coloring version.

1.4 Structure of the paper

In Section 2 we give formal definitions of competitive analysis and explain the use of Yao's Lemma ([Y77]) for lower bounds for on-line randomized algorithms.

Section 3 deals with the on-line induced subgraph problem. In Section 3.1 we present the lower bound for randomized non-preemptive algorithms for the on-line induced subgraph problem. In Section 3.2 we present the lower bound for randomized *preemptive* algorithms for on-line independent set. Finally, Section 3.3 gives upper bounds for the on-line induced subgraph problem.

We then show in Section 4 how the lower bounds for on-line independent set can be transformed into lower bounds for on-line edge-disjoint paths (and thus for call control). Section 4.1 shows the transformation for non-preemptive algorithms, while Sections 4.2 and 4.3 show transformations for preemptive algorithms.

Section 5 deals with the on-line graph coloring problem. The lower bound for the problem is presented in Section 5.1 and upper bounds are presented in Section 5.2.

The lower bound for on-line graph coloring is then transformed into a lower bounds for on-line path coloring in Section 6.

Section 7 deals with optical routing. The section contains all the lower bound results that follow from the results in the previous sections.

Finally, in Section 8 we present the lower bounds for the on-line edge-disjoint fixed paths problem.

2 Competitive analysis of randomized on-line algorithms

We consider problems that consist of a set of request sequences, and a set of possible corresponding answers. Given a specific request sequence an algorithm for the problem must provide an answer in response to each of the requests in the sequence. If the algorithm is on-line it must base its answer for a particular request only upon the requests already requested prior to the current request. A randomized algorithm may use coin tosses to decide upon its answers. An objective function is defined that maps request/answer sequences into the non-negative reals.

Let A be an algorithm and let σ be a request sequence. The value of the objective function associated with σ and the answers produced by A in response to σ is denoted by $A(\sigma)$.

Problems are divided into *cost* problems and *benefit* problems.

Cost Problems.

In the case of cost problems $A(\sigma)$ denotes the *cost* of algorithm A, and the goal of the algorithm is to minimize its cost. If A is randomized the goal is to minimize its expected cost over its coin tosses, $E(A(\sigma))$.

Let ON be an on-line algorithm (possibly randomized) and OPT be an optimal algorithm for some problem.

We say ON is ρ-competitive [ST85] for a cost problem (against an oblivious adversary (cf. [BBKTW90])) if there exists a constant a so that for every sequence σ such that $\mathrm{OPT}(\sigma) \geq a$,

$$E(\mathrm{ON}(\sigma)) \quad \leq \quad \rho \cdot \mathrm{OPT}(\sigma).$$

The value ρ is called the competitive ratio of the algorithm.

Benefit Problems.

In the case of benefit problems $A(\sigma)$ denotes the *benefit* of algorithm A, and the goal of the algorithm is to maximize its benefit. If A is randomized the goal is to maximize its expected benefit over its coin tosses, $E(A(\sigma))$.

Let ON be an on-line algorithm (possibly randomized) and OPT be an optimal algorithm for some problem.

We say ON is ρ-competitive for a benefit problem (against an oblivious adversary) if there exists a constant a so that for every sequence σ such that $\mathrm{OPT}(\sigma) \geq a$,

$$\mathrm{OPT}(\sigma) \quad \leq \quad \rho \cdot E(\mathrm{ON}(\sigma)).$$

The value ρ is called the competitive ratio of the algorithm.

2.1 Lower bounds using the minimax principle

Lower bounds for on-line randomized algorithms can be proved using Yao's Lemma based on the *von Neumann minimax principle* (see [Y77]) applied to on-line algorithms.

Yao's Lemma can be stated in this context in the following way: Let $f(\mathrm{ON}, \sigma)$ be some function of the algorithm ON and the sequence σ to the reals. Let $\mathcal{D}$ be the set of deterministic on-line algorithms, $\mathcal{R}$ be the set of randomized on-line algorithms, $\mathcal{S}$ be the set of all possible request sequences, and $\mathcal{P}$ be the set of all probability distributions on the request sequences.

$$\inf_{\mathrm{ON} \in \mathcal{R}} \sup_{\sigma \in \mathcal{S}} \mathrm{E}(f(\mathrm{ON}, \sigma)) = \sup_{\sigma \in \mathcal{P}} \inf_{\mathrm{ON} \in \mathcal{D}} \mathrm{E}(f(\mathrm{ON}, \sigma)),$$

where the expectation on the left-hand side is over the coin tosses of the algorithm while the expectation on the right-hand side is over the probability distribution on the request sequences.

The following corollaries can be derived from Yao's Lemma:

Corollary 1 *Given a cost problem. The following three statements are equivalent:*

1. *ρ is a lower bound on the competitive ratio of randomized on-line algorithms.*
2. *There exists a probability distribution over request sequences σ such that $\mathrm{OPT}(\sigma) \geq a$ and for any deterministic on-line algorithm*

$$\mathrm{E}(\frac{\mathrm{ON}(\sigma)}{\mathrm{OPT}(\sigma)}) \geq \rho.$$

3. *There exists a probability distribution over request sequences σ such that $\mathrm{OPT}(\sigma) \geq a$ and for any deterministic on-line algorithm*

$$\frac{\mathrm{E}(\mathrm{ON}(\sigma))}{\mathrm{E}(\mathrm{OPT}(\sigma))} \geq \rho.$$

Statement 2 is derived by setting $f(\mathrm{ON}, \sigma) = \mathrm{ON}(\sigma)/\mathrm{OPT}(\sigma)$, and statement 3 is derived by setting $f(\mathrm{ON}, \sigma) = \mathrm{ON}(\sigma) - \rho \cdot \mathrm{OPT}(\sigma)$.

Similarly, for benefit problems we get the following corollary:

Corollary 2 *Given a benefit problem. The following three statements are equivalent:*

1. *c is a lower bound on the competitive ratio of randomized on-line algorithms.*
2. *There exists a probability distribution over request sequences σ such that $\mathrm{OPT}(\sigma) \geq a$ and for any deterministic on-line algorithm*

$$\mathrm{E}(\frac{\mathrm{OPT}(\sigma)}{\mathrm{ON}(\sigma)}) \geq \rho.$$

3. *There exists a probability distribution over request sequences σ such that* $\mathrm{OPT}(\sigma) \geq a$ *and for any deterministic on-line algorithm*

$$\frac{\mathrm{E}(\mathrm{OPT}(\sigma))}{\mathrm{E}(\mathrm{ON}(\sigma))} \geq \rho.$$

Statement 2 is derived by setting $f(\mathrm{ON}, \sigma) = \mathrm{OPT}(\sigma)/\mathrm{ON}(\sigma)$, and statement 3 is derived by setting $f(\mathrm{ON}, \sigma) = \mathrm{OPT}(\sigma) - \rho \cdot \mathrm{ON}(\sigma)$.

Note that the expectation in the above corollaries is over the probability distribution on the request sequences.

As a digression, note that although in general the expectation of a quotient is different than the the quotient of expectations the above discussion in fact implies that under worst case probability distributions (which may be different) the two achieve the same value.

In this paper we give lower bounds, concentrating on the case $a = 0$. However, they can be extended to deal with arbitrary constant a.

These lower bounds are based on statement 3 in the above corollaries.

3 The on-line induced subgraph problem

In this section we give lower and upper bounds on the competitive ratio of randomized algorithms for the on-line induced subgraph problem.

We start with some properties concerning induced subgraph problems that will be used later in our proofs.

A graph property π is *hereditary* if when satisfied for a given graph G, it is also satisfied for any vertex induced subgraph of G. Property π is *non-trivial* if it is satisfied for infinitely many graphs and not satisfied for infinitely many graphs.

Given a graph $G = (V, E)$ and a non-trivial hereditary property π, the induced subgraph problem consists of finding the maximum cardinality induced a subgraph satisfying the property π. Any graph H that does not satisfy a hereditary property π is called a *forbidden graph* for π. Observe that any graph G that has H as an induced subgraph does not satisfy π.

For any hereditary property there exists a set of "minimal" forbidden graphs, i.e. forbidden graphs for which every induced proper subgraph satisfies the property. For instance, a K_5 (a clique of 5 vertices) is a minimal forbidden graph for planarity, a K_2 is a minimal forbidden graph for independent set.

Given a non-trivial hereditary property π, there is a complementary property π^c that is satisfied for a graph G if and only if π is satisfied for the complemented graph G^c (the graph that contains an edge iff G does not contain that edge). The complementary property is also a non-trivial hereditary property. Ramsey's theorem implies that any non-trivial hereditary property is satisfied by all independent sets or by all cliques [LY93a]. Thus, without loss of generality, we thereafter restrict attention to hereditary non-trivial graph properties that are satisfied for all independent sets.

The graph instances for our lower bounds are based on an application of the following variant of Lemma 9 in [LY93a]:

Lemma 3 [Lund,Yannakakis] *Given any graph H, there exist constants M and α, such that for all $n > M$, there exists a graph G on n vertices such that any induced subgraph of G on at least $l = \alpha \log n$ vertices contains H as a induced subgraph.*

Observe that if the graph H of the lemma is a K_c, then G can be a K_n and $l = c$.

Let the graph H of Lemma 3 be a forbidden graph for a non-trivial, hereditary, property π. We base our lower bound constructions on a graph $G = (V, E)$ of n vertices, whose existence is guaranteed by Lemma 3 applied with the specific graph H.

On the basis of this graph G, we build the instance $\tilde{G}$, for the lower bound for the on-line induced subgraph problem.

Recall that in the on-line version of the induced subgraph problem we assume that the graph $\tilde{G}$ from which the adversary draws vertices, is known to the on-line algorithm.

The graph $\tilde{G}$ is built recursively. The vertex set of $\tilde{G}$ is V. In fact, $\tilde{G}$ is simply the graph G with some edges removed. We remove edges so that the "clever" adversary can choose vertices carefully while avoiding including the forbidden graph H as an induced subgraph of the solution, whereas the on-line algorithm cannot do so well.

We next need to define sequences of requests to the vertices in $\tilde{G}$. Recall that at every step a new vertex is presented to the on-line algorithm that must decide whether to choose to include it or choose to reject it.

The lower bounds are based on Yao's Lemma (see Section 2). We define recursively a probability distribution on request sequences in $\tilde{G}$ so that for the ratio between the expected benefit of an optimal off-line algorithm and the expected benefit of any deterministic on-line algorithm is $\Omega(n^{\epsilon})$.

Section 3.1 describes the graph construction, the definition of the probability distribution, and the analysis of the ratio in the case of non-preemptive algorithms. For the case of the independent set graph property we give a similar analysis in Section 3.2 also in the case of preemptive algorithms. In Section 3.3 we give upper bounds for the on-line induced subgraph problem.

3.1 Lower bound for non-preemptive randomized algorithms for on-line induced subgraph

Let $G = (V, E)$ be a graph of n vertices, such that every subgraph of size at least $l = \alpha \log n$ is a forbidden graph for property π. As mentioned before, the existence of such a graph for arbitrarily large n is given by Lemma 3. For clarity of exposition, we assume that n is a power of 3.

We consider a sequence of $\log_3 n$ partitions of the vertices of $V = \{v_1, v_2, \ldots, v_n\}$. The level 0 partition,

$$P_0 = \{V_0^1 = \{v_1\}, V_0^2 = \{v_2\}, \ldots, V_0^n = \{v_n\}\},$$

consists of n disjoint singleton sets of vertices from V. The level i partition, $0 < i \leq \log_3 n$,

$$P_i = \{V_i^1, V_i^2, \ldots, V_i^{n/3^i}\},$$

is a partition of the vertices of V where

$$V_i^j = \cup_{k=3(j-1)+1,\ldots,3(j-1)+3} V_{i-1}^k.$$

Note that $V_{\log_3 n}^1 = V$.

We now describe the construction of $\tilde{G}$ recursively. The vertex set of $\tilde{G}$ is V. The graph $\tilde{G}$ can be viewed as the graph G with some edges removed.

For all i, $0 \leq i \leq \log_3 n$ and all $V_i^j \in P_i$ we define graphs $\tilde{G}_i(V_i^j)$ with vertex set equal to V_i^j. The final graph is $\tilde{G} = \tilde{G}_{\log_3 n}(V)$. The graph $\tilde{G}_0(V_0^j)$ is simply the single vertex v_j and has no edges.

The graph $\tilde{G}_i(V_i^j)$, $i > 0$, $j = 1, \ldots, \frac{n}{3^i}$, is constructed from three graphs $\tilde{G}_{i-1}^k = \tilde{G}_{i-1}(V_{i-1}^{3(j-1)+k})$, for $k = 1, 2, 3$.

The set of edges of $\tilde{G}_i(V_i^j)$ (see Figure 1) is recursively defined as the union of the set of edges in $\tilde{G}_{i-1}^1, \tilde{G}_{i-1}^2, \tilde{G}_{i-1}^3$, and two new sets of edges:

- E^{12} — edges (u, w), $u \in \tilde{G}_{i-1}^1$, $w \in \tilde{G}_{i-1}^2$, if and only if $(u, w) \in E$;
- E^{13} — edges (u, w), $u \in \tilde{G}_{i-1}^1$, $w \in \tilde{G}_{i-1}^3$, if and only if $(u, w) \in E$;

Graph $\tilde{G}_i(V_i^j)$ has $n(i) = 3^i$ vertices.

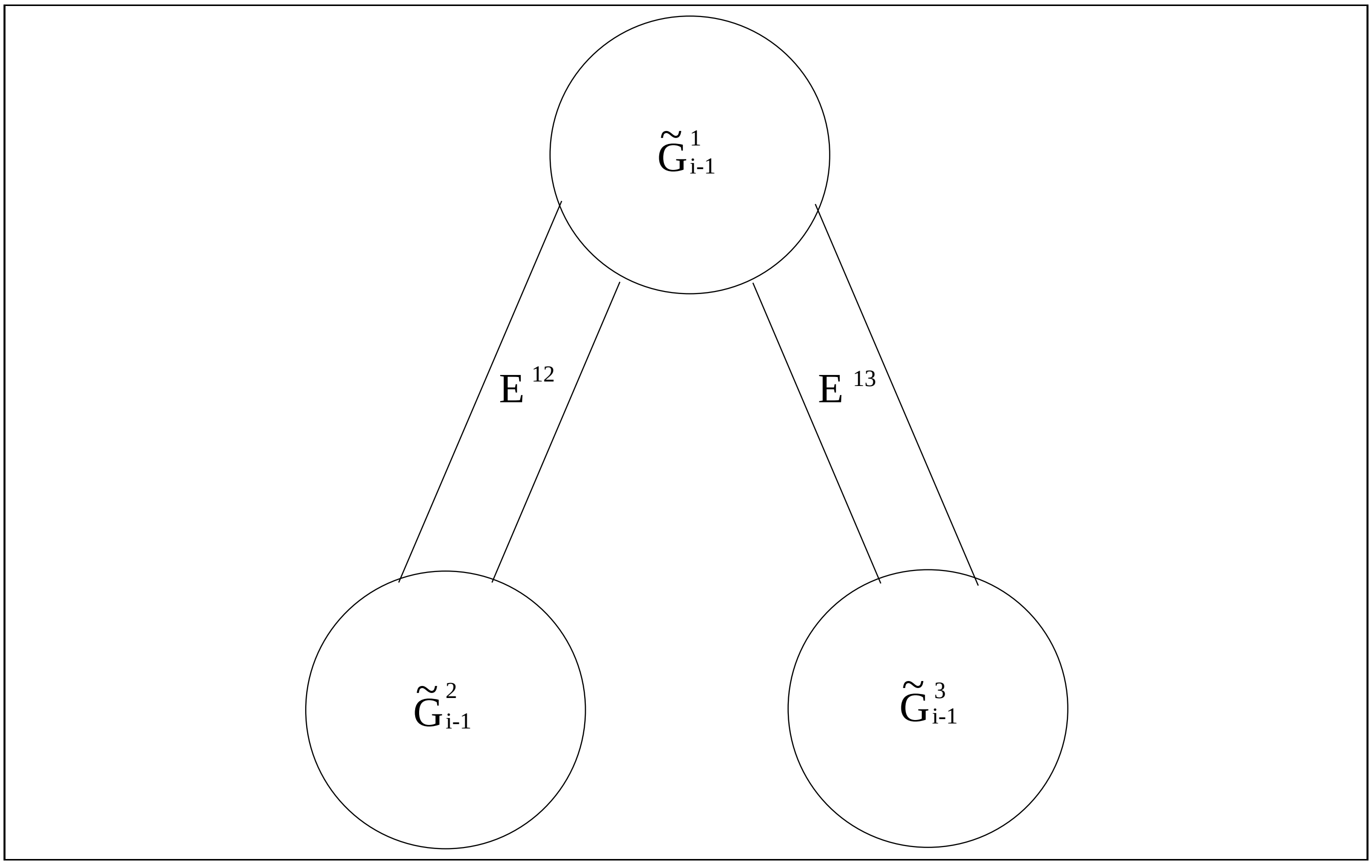

Figure 1: The construction of graph $\tilde{G}_i$.

We say that two graphs of level $i-1$ connected by edges are *adjacent*. (Observe that in the case that $G = K_n$ a vertex of a graph is connected to all the vertices of an adjacent graph).

We say that a graph, $\tilde{G}_i(V_i^j)$, *contains an induced subgraph of G of size s*, if the following condition holds: There exist s different vertices $\{u_1, u_2, \ldots, u_s\} \subseteq V$, such that for all $1 \leq l, k \leq s$, $(u_l, u_k) \in E$ if and only if (u_l, u_k) is an edge of $\tilde{G}_i(V_i^j)$.

The reason for the edges E^{12}, E^{13} is to ensure that if two adjacent graphs of level $i-1$ contain two induced subgraphs of G of sizes s^1 and s^2, then $\tilde{G}_i$ contains an induced subgraph of G of size $s^1 + s^2$.

From this point on we omit the sets V_i^j from the notation and the appropriate sets should be evident from the context.

We will now define the probability distribution on the request sequences of $\tilde{G}$. We will prove that for any on-line non-preemptive algorithm ON a lower bound on the ratio ρ between the expected optimal off-line benefit E(OPT) and the expected on-line benefit E(ON).

The probability distribution will be defined recursively. The probability distribution on the input sequences for a graph $\tilde{G}_0$ consists of deterministically choosing the single vertex of the graph. The probability distribution on the input sequences for a graph $\tilde{G}_i$ is recursively defined by a two step process:

1. The probability distribution on the input sequences for $\tilde{G}^1_{i-1}$.
2. Toss a fair coin:
   - If heads, the probability distribution on the input sequences for $\tilde{G}^2_{i-1}$ and $\tilde{G}^3_{i-1}$.
   - If tails, end the sequence.

Lemma 4 *The expected off-line optimal benefit for any input sequence from the distribution for $\tilde{G}_i$ is* $\mathrm{E}(\mathrm{OPT}(i)) = \left(\frac{3}{2}\right)^i$.

Proof. We will prove by induction on i that the optimal solution for any input sequence from the distribution for $\tilde{G}_i$ consists of an independent set of expected size $\left(\frac{3}{2}\right)^i$. Since the property π is satisfied for any independent set, it follows that $\mathrm{E}(\mathrm{OPT}(i)) = \left(\frac{3}{2}\right)^i$. The claim is clearly true for $i = 0$. Assume that the claim holds for the distribution on a graph of level $i-1$ with $\mathrm{E}(\mathrm{OPT}(i-1))= \left(\frac{3}{2}\right)^{i-1}$.

Consider the recursive definition of the probability distribution on input sequences for $\tilde{G}_i$. Then with probability 1/2 the probability distribution on input sequences for $\tilde{G}^1_{i-1}$ is given, and therefore the optimal solution consists of exactly the optimal solution for $\tilde{G}^1_{i-1}$. With probability 1/2 the probability distributions on input sequences for $\tilde{G}^2_{i-1}$ and $\tilde{G}^3_{i-1}$ are given. Since $\tilde{G}^2_{i-1}$ and $\tilde{G}^3_{i-1}$ are non-adjacent, the union of their optimal solutions forms an independent set. Therefore the optimal solution consists of the union of the optimal solutions for $\tilde{G}^2_{i-1}$ and $\tilde{G}^3_{i-1}$. The expected size of the optimal solution is thus

$$\mathrm{E}(\mathrm{OPT}(i)) \;=\; \frac{1}{2}\left(\frac{3}{2}\right)^{i-1} + \frac{1}{2}\cdot 2\left(\frac{3}{2}\right)^{i-1} \;=\; \left(\frac{3}{2}\right)^{i}. \qquad \blacksquare$$

Fix some level i, and consider the on-line solution over the distribution on the input sequences for the graph $\tilde{G}_i$. This solution is some subset of vertices of $\tilde{G}_i$ that induces a subgraph satisfying property π. Let s_i be the random variable (depending on the particular given sequence out of the distribution) equal to the size of the maximum size induced subgraph of G contained in the on-line solution. Similarly, let s^1_{i-1}, s^2_{i-1}, s^3_{i-1} be the analogous values for the on-line solutions over the subsequences of vertices in the three graphs $\tilde{G}^1_{i-1}, \tilde{G}^2_{i-1}, \tilde{G}^3_{i-1}$ of level $i-1$, respectively. The following claim follows from the graph definition.

Claim 5 $\max(s^1_{i-1} + s^2_{i-1}, s^1_{i-1} + s^3_{i-1}) \leq s_i$.

Lemma 6 *The expected benefit of any non-preemptive deterministic on-line algorithm for the distribution over the input sequences for* $\tilde{G}_i$ *is* $\mathrm{E}(\mathrm{ON}(i)) \leq s_i$.

Proof. Consider a non-preemptive algorithm ON. The on-line expected benefit is given by the sum of the expected benefit on $\tilde{G}^1_{i-1}$ plus, with probability $1/2$, the expected benefits on $\tilde{G}^2_{i-1}$ and $\tilde{G}^3_{i-1}$. Moreover, by Claim 5, the sum of the values s_{i-1} of two adjacent graphs is at most s_i. Hence, by induction,

$$\begin{aligned}
\mathrm{E}(\mathrm{ON}(i)) &\leq s^1_{i-1} + \frac{1}{2}(s^2_{i-1} + s^3_{i-1}) \\
&\leq \frac{1}{2}(s^1_{i-1} + s^2_{i-1}) + \frac{1}{2}(s^1_{i-1} + s^3_{i-1}) \\
&\leq s_i.
\end{aligned}$$

■

Theorem 7 *Any non-preemptive randomized algorithm for the on-line induced subgraph problem has competitive ratio* $\Omega(n^{1-\log_3 2 - o(1)})$. *For a hereditary property having a* K_c *as forbidden graph the competitive ratio is* $\Omega(n^{1-\log_3 2})$.

Proof.

By Lemma 4 and Lemma 6 the ratio $\rho(i)$ between the expected optimal benefit and the expected on-line benefit for the probability distribution on the input sequences for a graph $\tilde{G}_i$, $1 \leq i \leq \log_3 n$, is:

$$\begin{aligned}
\rho(i) &= \frac{\mathrm{E}(\mathrm{OPT}(i))}{\mathrm{E}(\mathrm{ON}(i))} \geq \frac{\left(\frac{3}{2}\right)^i}{s_i} = \frac{n(i)}{2^{\log_3 n(i)} s_i} \\
&= \frac{n(i)}{n(i)^{\log_3 2} s_i} = \frac{n(i)^{1-\log_3 2}}{s_i}.
\end{aligned}$$

Recall that s_i is the size of an induced subgraph in G, satisfying property π. It follows from Lemma 3 that $s_i < l = \alpha \log n$. Therefore, for $i = \log_3 n$, we obtain the $\Omega(n^{1-\log_3 2 - o(1)})$ lower bound on the competitive ratio. If the forbidden graph is a K_c, then $l = c$, we get an $\Omega(n^{1-\log_3 2})$ lower bound. ■

3.2 Lower bound for preemptive algorithms for on-line independent set

Foe the on-line independent set problem, we give the lower bound proof in the case that the randomized algorithm is also preemptive. I.e., we allow for the on-line algorithm that vertices included in the solution may be excluded later. However, once a vertex has been excluded, it cannot be re-inserted in the future.

The proof technique is essentially the same as in the last section. The difference is in the basic recursive construction of the graph $\tilde{G}$. Notice that in the construction given in the previous section a preemptive algorithm can actually achieve the optimal benefit. However it turns out that a somewhat different basis for the recursive construction can eliminate the power of preemption.

For clarity of exposition, we assume that n is a power of 4. We consider a sequence of $\log_4 n$ partitions of the vertices of $V = \{v_1, v_2, \ldots, v_n\}$. The level 0 partition,

$$P_0 = \{V_0^1 = \{v_1\}, V_0^2 = \{v_2\}, \ldots, V_0^n = \{v_n\}\},$$

consists of n disjoint singleton sets of vertices from V. The level i partition, $0 < i \leq \log_4 n$,

$$P_i = \{V_i^1, V_i^2, \ldots, V_i^{n/4^i}\},$$

is a partition of the vertices of V where

$$V_i^j = \cup_{k=4(j-1)+1,\ldots,4(j-1)+4} V_{i-1}^k.$$

Note that $V_{\log_4 n}^1 = V$.

We now describe the construction of $\tilde{G}$ recursively. The vertex set of $\tilde{G}$ is V.

For all i, $0 \leq i \leq \log_4 n$ and all $V_i^j \in P_i$ we define graphs $\tilde{G}_i(V_i^j)$ with vertex set equal to V_i^j. The final graph is $\tilde{G} = \tilde{G}_{\log_4 n}(V)$. The graph $\tilde{G}_0(V_0^j)$ is simply the single vertex v_j and has no edges.

The graph $\tilde{G}_i(V_i^j)$, $i > 0$, $j = 1, \ldots, \frac{n}{4^i}$, is constructed from four graphs $\tilde{G}_{i-1}^k = \tilde{G}_{i-1}(V_{i-1}^{4(j-1)+k})$, for $k = 1, 2, 3, 4$.

The set of edges of $\tilde{G}_i(V_i^j)$ (see Figure 2) is recursively defined as the union of the set of edges in $\tilde{G}_{i-1}^1, \tilde{G}_{i-1}^2, \tilde{G}_{i-1}^3, \tilde{G}_{i-1}^4$, and three new sets of edges:

- E^{12} — all edges (u, w), $u \in \tilde{G}_{i-1}^1$, $w \in \tilde{G}_{i-1}^2$.
- E^{13} — all edges (u, w), $u \in \tilde{G}_{i-1}^1$, $w \in \tilde{G}_{i-1}^3$.

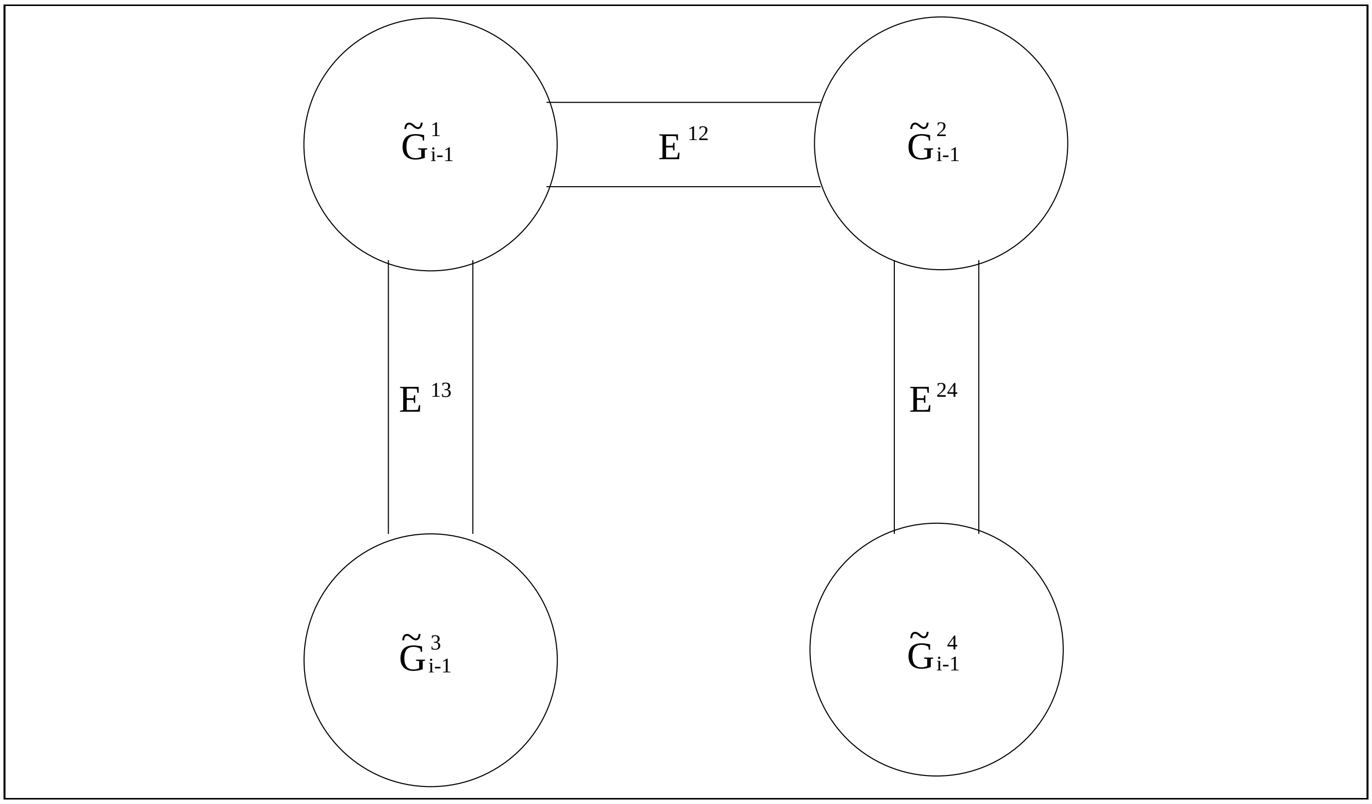

Figure 2: The construction of graph $\tilde{G}_i$ for preemptive algorithms.

- E^{24} — all edges (u, w), $u \in \tilde{G}^2_{i-1}$, $w \in \tilde{G}^4_{i-1}$.

Graph $\tilde{G}_i(V_i^j)$ has $n(i) = 4^i$ vertices.

As in the previous section, we say that two graphs of level $i-1$ connected by edges are *adjacent*.

The reason for the edges E^{12}, E^{13}, and E^{24} is to ensure that any independent set in $\tilde{G}_i$ does not contain vertices in two adjacent graphs of level $i-1$.

We now turn to define the probability distribution on the request sequences of $\tilde{G}$. The optimal benefit on every sequence generated by our distribution will be the same. We will prove that for any on-line preemptive algorithm ON a lower bound on the ratio ρ between the optimal off-line benefit OPT and the expected on-line benefit E(ON).

The probability distribution will be defined recursively. The probability distribution on the input sequences for a graph $\tilde{G}_0$ consists of deterministically choosing the single vertex of the graph. The probability distribution on the input sequences for a graph $\tilde{G}_i$ is defined by a two step process:

1. The probability distribution on the input sequences for $\tilde{G}^1_{i-1}$ and $\tilde{G}^2_{i-1}$.

2. Toss a fair coin:
   - If heads, the probability distribution on the input sequences for $\tilde{G}^3_{i-1}$.
   - If tails, the probability distribution on the input sequences for $\tilde{G}^4_{i-1}$.

Lemma 8 *The off-line optimal benefit for any input sequence from the distribution for $\tilde{G}_i$ is* $\mathrm{OPT}(i) = 2^i$.

Proof. We will prove by induction on i that the optimal solution for any input sequence from the distribution for $\tilde{G}_i$ consists of an independent set of size 2^i. The claim is clearly true for $i = 0$. Assume that the claim holds for the distribution on a graph of level $i-1$ with $\mathrm{OPT}(i-1) = 2^{i-1}$. Since any input sequence from the distribution for $\tilde{G}_i$ always contains two sequences for two non-adjacent graphs of level $i-1$, we get that the union of the optimal solutions for the two non-adjacent graphs forms an independent set of size 2^i. ∎

Lemma 9 *The expected benefit of any preemptive deterministic on-line independent set algorithm for the distribution over the input sequences for $\tilde{G}_i$ is* $\mathrm{E}(\mathrm{ON}(i)) \leq \left(\frac{3}{2}\right)^i$.

Proof. Let ON be a preemptive algorithm for on-line independent set. The proof is by induction on i. The claim is clearly true for $i = 0$. We will assume that it holds for $i-1$ and prove that it also holds for i. After the probability distributions for $\tilde{G}^1_{i-1}$ and $\tilde{G}^2_{i-1}$ are presented (i.e., after step 1 of the probability distribution for $\tilde{G}_i$) the algorithm solution may be either an independent set in $\tilde{G}^1_{i-1}$ or an independent set in $\tilde{G}^2_{i-1}$. Then in step 2 of the probability distribution for $\tilde{G}_i$ exactly one of the probability distributions for $\tilde{G}^3_{i-1}$ and $\tilde{G}^4_{i-1}$ is presented with probability 1/2. Therefore, if the algorithm preempts all vertices formerly accepted from $\tilde{G}^1_{i-1}$ and $\tilde{G}^2_{i-1}$ then its expected benefit is at most the expected benefit on the probability distribution for the new graph (either $\tilde{G}^3_{i-1}$ or $\tilde{G}^4_{i-1}$) presented in step 2, and thus by induction, at most $\left(\frac{3}{2}\right)^{i-1}$. Otherwise the algorithm accepts at least one vertex of either $\tilde{G}^1_{i-1}$ or $\tilde{G}^2_{i-1}$. Assume w.l.o.g that the algorithm accepts a vertex of $\tilde{G}^1_{i-1}$. Then the algorithm can increase the size of its solution only with probability $\frac{1}{2}$, in the case that the graph presented in step 2 is non-adjacent with $\tilde{G}^1_{i-1}$. Thus the expected benefit is at most

$$\begin{aligned}\mathrm{E}(\mathrm{ON}(i)) &\leq \left(\frac{3}{2}\right)^{i-1} + \frac{1}{2}\left(\frac{3}{2}\right)^{i-1} \\ &= \left(\frac{3}{2}\right)^{i}.\end{aligned}$$

∎

Theorem 10 *Any preemptive randomized algorithm for the on-line independent set problem has competitive ratio* $\Omega(n^{1-\log_4 3})$.

Proof.

By Lemma 8 and Lemma 9 the ratio $\rho(i)$ between the optimal benefit and the expected on-line benefit for the probability distribution on the input sequences for a graph $\tilde{G}_i$, $1 \leq i \leq \log_4 n$, is:

$$\begin{aligned}
\rho(i) &= \frac{\text{OPT}(i)}{E(\text{ON}(i))} \geq \frac{2^i}{\left(\frac{3}{2}\right)^i} \geq \frac{4^i}{3^i} \\
&= \frac{n(i)}{3^{\log_4 n(i)}} = \frac{n(i)}{n(i)^{\log_4 3}} = n(i)^{1-\log_4 3}.
\end{aligned}$$

■

3.3 Upper Bounds for the on-line induced subgraph problem

In this section we give upper bounds for the on-line induced subgraph problem. The upper bounds are based on algorithms that compute an approximate solution for the off-line maximum induced subgraph problem.

We present a randomized algorithm based on the Classify and Randomly Select method of [ABFR94].

Let $G = (V, E)$ be the graph that we assume is known to the on-line algorithm. The algorithm requires a preprocessing phase in which a collection of k repetitive solutions $I_1, I_2, \ldots, I_k$ of the induced subgraph problem for the graph G are computed, as follows.

Let $G_1 = G$, and $G_{i+1} = G_i / I_i$ be the result of the operation that removes from G_i all vertices in I_i and all the incident edges.

I_i is the set of vertices forming the solution of the induced subgraph problem returned by applying the $\frac{n}{\alpha}$-approximation algorithm to the graph G_i. The process of computing solutions is continued until $|I_{k+1}| \leq \sqrt{\alpha}$. Let $\overline{I} = V - \{I_1 \cup I_2 \cup \ldots \cup I_k\}$ be the set of vertices of G_{k+1}.

Finally, the algorithm performs the following random choice:

Toss a fair coin:

1. If heads choose at random with equal probability one of the sets $I_1, I_2, \ldots, I_k$, and then accept only presented vertices that belong to that set.
2. If tails then greedily accept vertices from $\overline{I}$.

Theorem 11 *Given a $\frac{n}{\alpha}$-approximation algorithm for the induced subgraph problem, there exists* *an $O(\frac{n}{\sqrt{\alpha}})$-competitive randomized algorithm for the on-line version of the problem. If the original algorithm is polynomial, so is the on-line algorithm.*

Proof. First observe that k is at most $\frac{n}{\sqrt{\alpha}}$. Next, observe that the optimal solution over the set $\overline{I}$ is at most $\frac{n}{\sqrt{\alpha}}$, as the solution produced by the $\frac{n}{\alpha}$-approximation algorithm applied to the graph induced by $\overline{I}$ has size less than $\sqrt{\alpha}$.

Let $\mathrm{OPT}(I)$ be the size of the optimal solution in the subgraph of I induced by the vertices in I presented. Then

$$\mathrm{OPT} \leq \sum_{i=1}^{k} \mathrm{OPT}(I_i) + \mathrm{OPT}(\overline{I}).$$

The expected number of vertices accepted by the on-line algorithm is

$$\begin{aligned} \mathrm{ON} &\geq \tfrac{1}{2}(\tfrac{1}{k}\textstyle\sum_{i=1}^{k}\mathrm{OPT}(I_i)+1) \\ &\geq \tfrac{\sqrt{\alpha}}{2n}(\textstyle\sum_{i=1}^{k}\mathrm{OPT}(I_i)+\mathrm{OPT}(\overline{I})) \geq \tfrac{\sqrt{\alpha}}{2n}\mathrm{OPT}. \end{aligned}$$

■

The current best known polynomial time algorithms have an approximation ratio of $O(\frac{n}{\log n})$ for the induced subgraph problem [Hal94], and $O(\frac{n}{(\log n)^2})$ for independent set [BH92]. In particular, using the optimal (non-polynomial time) algorithm for the problem (e.g. $\alpha = n$) we obtain:

Corollary 12 *There exists an $O(\sqrt{n})$-competitive randomized algorithm for the on-line induced subgraph problem.*

4 On-line edge-disjoint paths

In this section we present $\Omega(n^\epsilon)$ lower bounds on the competitive ratio of preemptive and non-preemptive randomized algorithms for on-line edge-disjoint paths. As a consequence (see Section 1.2), we obtain an $\Omega(n^\epsilon)$ lower bound for on-line call-control. For this reason, in the following we refer to pairs of vertices as calls.

The lower bounds are derived from the lower bounds for the on-line induced subgraph problem presented in Section 3 when the hereditary property is independent set. Recall that the forbidden graph of Lemma 3 for independent set can be a K_2, and thus the proofs of section 3 can be carried out with the graph G of n vertices being a K_n.

We say two calls are *consistent* if they can be connected with two edge-disjoint paths, otherwise they are *inconsistent*.

The lower bounds for on-line edge-disjoint paths we derive are obtained by embedding the graph $\tilde{G}$ and an input sequence on which the lower bound for on-line independent set is obtained within a network W with the following two properties:

1. Every vertex u in $\tilde{G}$ corresponds to one call $(s(u), t(u))$ in W.
2. Two vertices u, v in $\tilde{G}$ are not adjacent if and only if the two corresponding calls $(s(u), t(u))$, $(s(v), t(v))$ in W are consistent.

The network constructions we will present are based on the *Brick Wall* network. A brick wall means a portion of the hexagonal lattice as indicated in Figure 3. A brick wall of height h and width w consists of h horizontal paths and w edges between consecutive pairs of them.

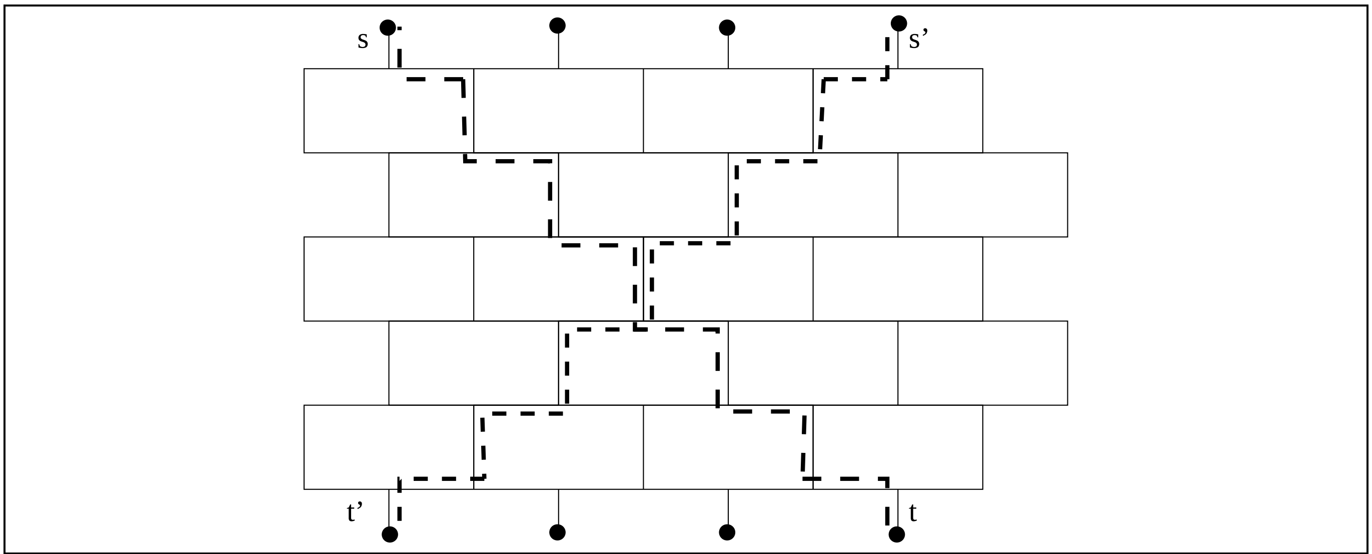

Figure 3: The brick wall network.

If we restrict attention to calls from top vertices to bottom vertices of the brick wall, we can easily observe in Figure 3 the following property.

Lemma 13 *Let $c = (s, t)$ and $c' = (s', t')$ be two top-to-bottom calls in W with $s < s'$. c and c' are consistent if and only if $t < t'$.*

The following lemma gives a bound on the maximum number of top-bottom calls that can be routed in a brick wall network.

Lemma 14 *Let h be the height of the brick wall and let w be its width. Any set of up to $\ell = \min(h, w)$ pairwise consistent top-to-bottom calls can be routed in a brick-wall network of height h and width w.*

Proof. Consider a set of pairwise consistent top-to-bottom calls. A feasible routing of the set of calls is as follows. Consider the pairs from left to right. Assign at each pair the leftmost available path that does not intersect a path assigned to any previous considered pair. Figure 4 gives an example for a set of 3 calls in a brick wall of height 3. ■

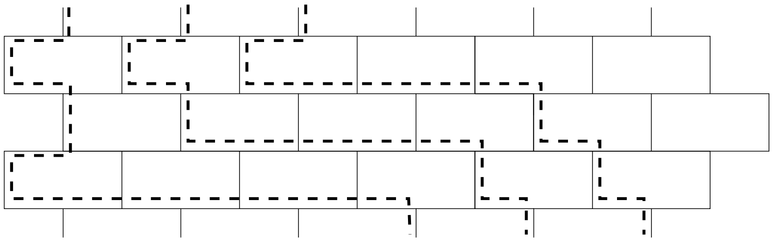

Figure 4: Three consistent calls in a brick wall of height 3.

In Section 4.1 we present a lower bound on the competitive ratio of non-preemptive randomized algorithms for on-line edge-disjoint paths, obtained by embedding the on-line independent set lower bound for non-preemptive algorithms of Section 3.2 in a network of $O(n^{1+\log_2 3})$ vertices.

We present two randomized lower bounds for preemptive on-line edge-disjoint paths in Section 4.2. The first lower bound is obtained by embedding the on-line independent set lower bound for preemptive algorithms of Section 3.2 in a network of $O(n^{\frac{3}{2}})$ vertices. This result is then improved using a more complicated embedding in a network of $O(n^{\log_4 6})$ vertices.

4.1 A lower bound for non-preemptive on-line edge-disjoint paths

In this section we devise an $\Omega(n^{\epsilon})$ lower bound on the competitive ratio of non-preemptive randomized algorithms for on-line edge disjoint paths by embedding in a brick wall W the graph $\tilde{G}$ used in the lower bound for non-preemptive randomized algorithms for on-line independent set presented in Section 3.1.

Let n be the number of vertices in $\tilde{G}$. The number of vertices at both top and bottom rows of W (indicated with dots in Figure 3) is n.

We associate every vertex $u \in \tilde{G}$ with a top-to-bottom call $(s(u), t(u))$ in W. The mapping will be such that two vertices are adjacent in $\tilde{G}$ iff the two calls are inconsistent. Each of the top and bottom vertices will appear in exactly one call.

The largest independent set in $\tilde{G}$ has size $2^{\log_3 n} = n^{\log_3 2}$. By setting the height of W to $n^{\log_3 2}$, we can ensure, by Lemma 14, that the optimal solution can be consistently routed. The width of

W is n. Thus the number of vertex of W is $N = O(n^{1+\log_3 2})$.

Recall the construction of the graph $\tilde{G}$ for the lower bound for the non-preemptive on-line independent set problem. The construction can be based on a K_n (graph G of Lemma 3). A graph $\tilde{G}_i$ of level i is formed by 3 graphs of level $i-1$. Every vertex of a level $i-1$ graph is adjacent in $\tilde{G}$ to all vertices of an adjacent level $i-1$ graph.

Our embedding is itself recursive. For $\tilde{G}_0$, the call associated with the single vertex u is the single top-bottom call.

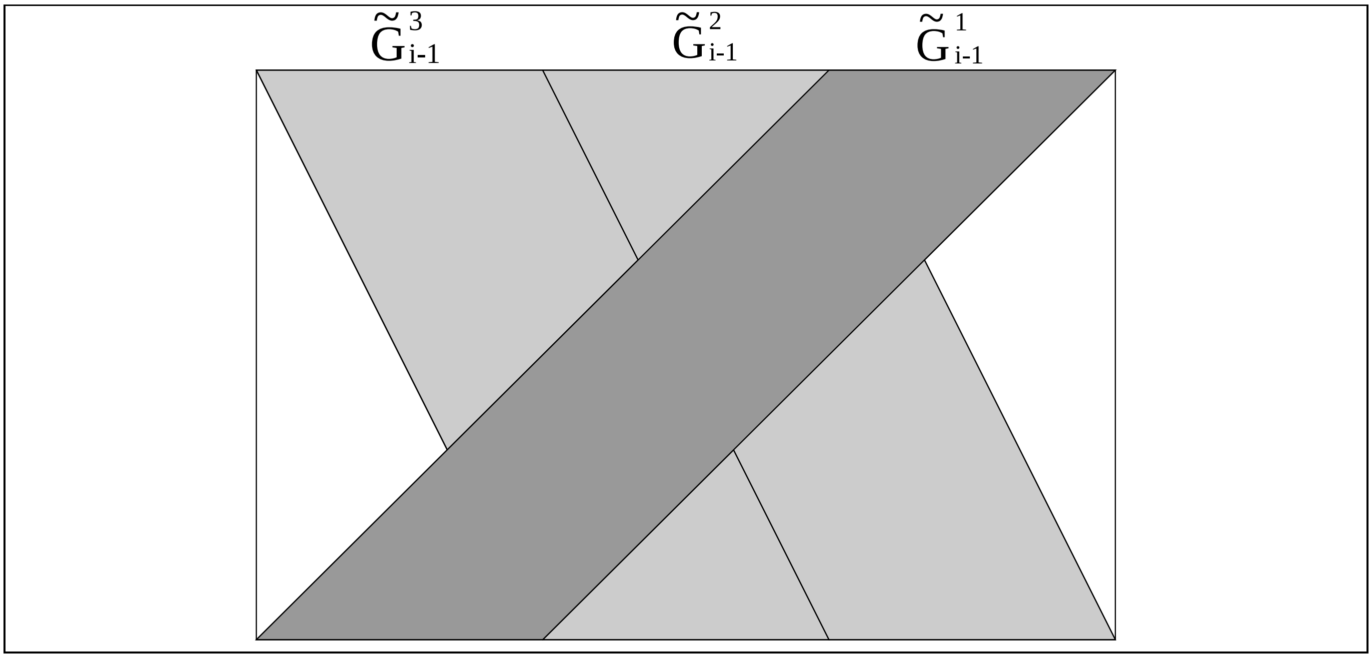

Figure 5: The mapping of the sequence in the graph $\tilde{G}_i$ on the brick wall.

Figure 5 shows the embedding at level i. For each of the three graphs of level $i-1$, $\tilde{G}^1_{i-1}$, $\tilde{G}^2_{i-1}$ and $\tilde{G}^3_{i-1}$ that make up $\tilde{G}_i$, a slice of the brick wall is reserved. Every vertex v in G_{i-1} is inductively mapped to a top-bottom call c in the inductive brick wall embedding of G_{i-1}. The vertex v in G_i is mapped to the translation of the call c under the slice mapping, *i.e.*, the top vertices of the brick-wall embedding of G_{i-1} are mapped (in left to right order) to the top vertices of the slice associated with G_{i-1} in the brick wall embedding of G_i, the bottom vertices of the brick-wall embedding of G_{i-1} are mapped (in left to right order) to the bottom vertices of the slice associated with G_{i-1} in the brick wall embedding of G_i.

We stress that the above defined *slices* of the brick wall are only used for clarity of exposition. However note that even if the endpoints of a call are in a particular slice the routing of the call is *not* restricted to that slice. This remark also applies to the brick-wall based constructions in the next sections.

Correctness follows by noting that the slices associated with adjacent graphs intersect, while the slices associated with non-adjacent graphs are disjoint. Finally, every top or bottom vertex is associated only with a single call.

Recall that W has $N = O(n^{1+\log_3 2})$ vertices, from Theorem 7 we get the following:

Theorem 15 *Any non-preemptive randomized algorithm for the on-line edge-disjoint paths problem (and hence for call control) on general networks of N vertices has competitive ratio $\Omega(N^{\frac{1-\log_3 2}{1+\log_3 2}})$.*

The brick wall network has degree 3. However, we prove the following theorem that extends any lower bound for on-line edge-disjoint paths to networks of minimum degree δ.

Theorem 16 *Given a $\Omega(f(N))$ lower bound on the competitive ratio of preemptive or non-preemptive algorithms for on-line edge-disjoint paths (and thus for on-line call control) on networks of N vertices, there exists a $\Omega(f(\frac{N}{\delta}))$ lower bound for the problem on network of minimum degree δ.*

Proof. We transform the network H of N vertices on which the lower bound is achieved into a network H' of minimum degree δ as follows: To every vertex corresponds a clique of size δ in the new network, to every edge of H corresponds a single edge between two arbitrary vertices in the two associated cliques.

Let us consider the input sequence σ for the $\Omega(f(n))$ lower bound in H. We transform σ into an input sequence σ' for H' by replacing every call in σ formed by two vertices, with a call formed by two arbitrary vertices in the two associated cliques. Under this transformation, any solution for σ' in H' corresponds to a solution for σ in H with equal size. Since H' has more vertices than H by a factor δ, the $\Omega(f(N))$ lower bound in the network H with input sequence σ yields an $\Omega(f(\frac{N}{\delta}))$ lower bound in H' with input sequence σ'. ∎

We derive the following corollary from Theorem 15 and 16 thus contradicting the claim for an $O(\log n)$-competitive randomized algorithm for call control in networks of degree $\Omega(\log n)$ appeared in [KS95].

Corollary 17 *Any non-preemptive randomized algorithm for on-line call control on general networks of N vertices and minimum degree δ has competitive ratio $\Omega(\left(\frac{N}{\delta}\right)^{\frac{1-\log_3 2}{1+\log_3 2}})$.*

4.2 Lower bounds for preemptive on-line edge-disjoint paths

We present two randomized lower bounds for preemptive on-line edge disjoint paths.

The first lower bound is obtained by embedding the graph $\tilde{G}$ and a sequence for the preemptive randomized lower bound for on-line independent set in a brick wall. The embedding is very similar to the embedding for the non-preemptive lower bound of the previous section.

Let n be the number of vertices in $\tilde{G}$. The number of vertices at both top and bottom rows of the brick wall W is n. The largest independent set in the graph $\tilde{G}$ of n vertices of the preemptive randomized lower bound for on-line independent set has size $\sqrt{n}$. By setting the height of W to $\sqrt{n}$, we can ensure, by Lemma 14, that the optimal solution can be consistently routed. Thus, the number of vertices of the network W is $N = O(n^{\frac{3}{2}})$.

We now associate every vertex $u \in \tilde{G}$ with a top-to-bottom call $(s(u), t(u))$ in W. The mapping will be such that two vertices are adjacent in $\tilde{G}$ iff the two calls are inconsistent.

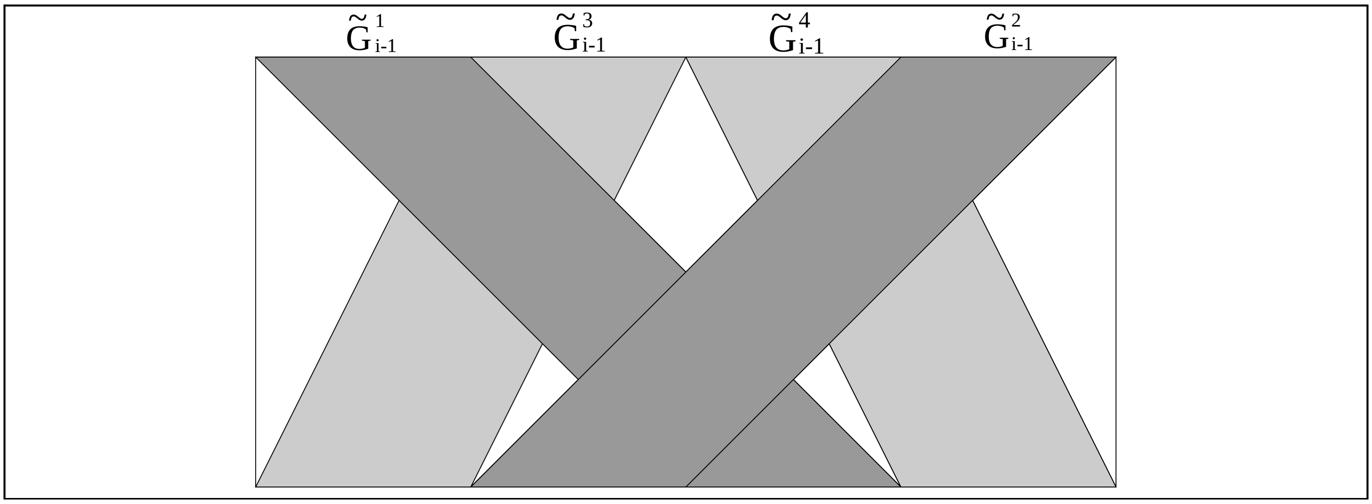

Figure 6: The mapping of the sequence in the graph $\tilde{G}_i$ on the brick wall.

Figure 6 shows how such embedding can be done at the i-th level. For each of the four graphs of level $i-1$, $\tilde{G}^1_{i-1}$, $\tilde{G}^2_{i-1}$, $\tilde{G}^3_{i-1}$ and $\tilde{G}^4_{i-1}$, that make up $\tilde{G}_i$, a slice of the brick wall is reserved. The mapping of vertices to calls observes the same rules as for the embedding in the non-preemptive lower bound of the previous subsection.

Observe in Figure 6 that the slices associated with adjacent graphs of Figure 2 intersect, while the slices associated with non-adjacent graphs are disjoint.

Since W has $N = n^{\frac{3}{2}}$ vertices, from Theorem 10 we get the following theorem:

Theorem 18 *Any preemptive randomized algorithm for the on-line edge-disjoint paths problem (and hence for call control) on general networks of N vertices has competitive ratio $\Omega(N^{\frac{2}{3}(1-\log_4 3)})$.*

In the next Section we will present a better lower bound for preemptive randomized algorithms.

4.3 Better results for on-line edge-disjoint paths

In this section we present an improved randomized lower bound for preemptive on-line edge-disjoint paths.

As for the lower bounds for on-line edge-disjoint paths described in the previous subsections, the lower bound is obtained by embedding the lower bound for on-line independent set (Section 3) on a particular network. The transformation associates a call with every vertex in the instance of the on-line independent set lower bound.

Let $n(i)$ denote the number of vertices in the i-level graph of the on-line independent set lower bound. We describe a network construction with $N(i) = O(n(i)^{\log_4 6})$ nodes.

The construction is built recursively. The network is associated with a set of calls only some of which can be routed consistently. The construction of a certain level i network is composed of $(i-1)$-level networks.

The network H_i has 4^i input nodes and 3^i output nodes.

H_0 is the network having a single vertex, representing both 1 input and 1 output of the network.

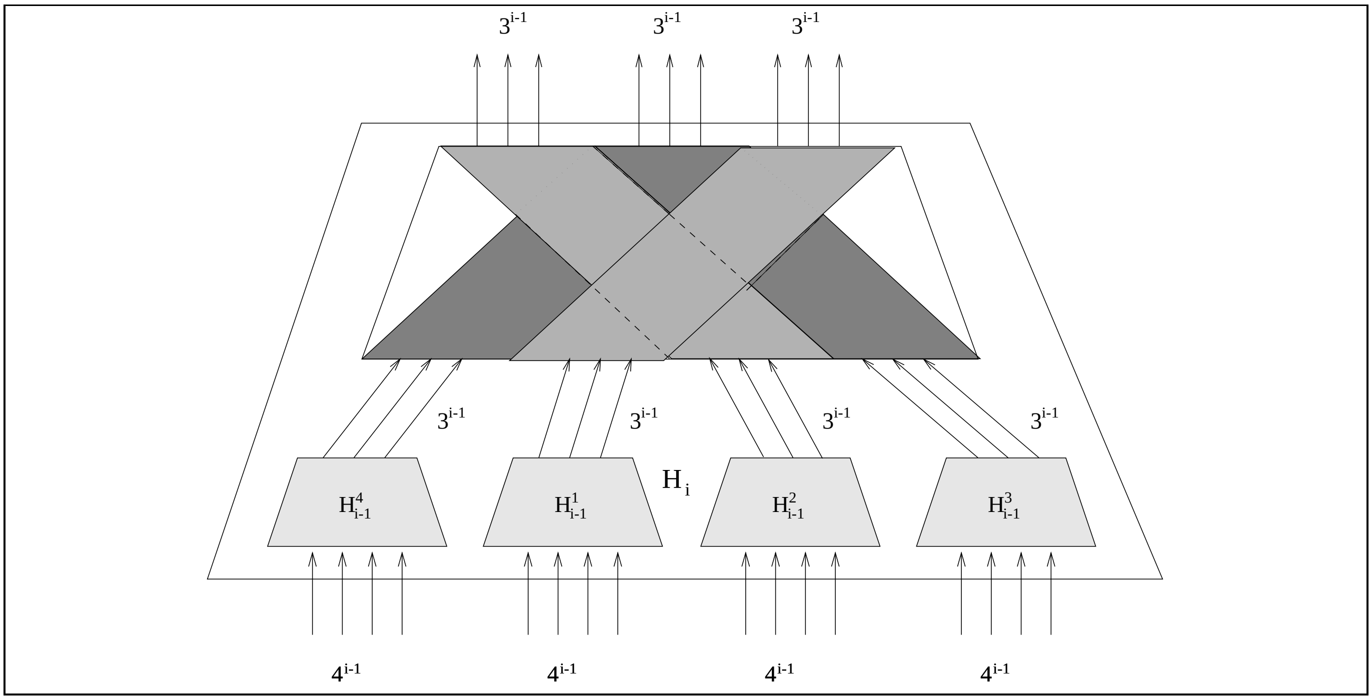

Figure 7: The construction of the network H_i.

The network H_i (Figure 7) is recursively composed of four instances of $(i-1)$-level networks: $H^1_{i-1}, H^2_{i-1}, H^3_{i-1}, H^4_{i-1}$, and one Brick-Wall network of width $4 \cdot 3^{i-1}$ and height 2^i.

The input of H_i is composed by the union of the sets of inputs for the four $(i-1)$-level networks.

The outputs of $H_{i-1}^1, H_{i-1}^2, H_{i-1}^3$ and H_{i-1}^4 are directed into appropriate four slices of the brick wall network, *while maintaining the respective order of the outputs within a slice.* The slices are chosen to intersect in a way so that the calls associated with H_{i-1}^1 intersect with those associated with H_{i-1}^2 and H_{i-1}^3 and the calls associated with H_{i-1}^2 intersect with H_{i-1}^4.

We now describe in detail the transformation from the instance of the preemptive independent set lower bound to the network described above.

Let $\tilde{G}_i$ denote the i-level graph of the independent set lower bound. Recall that the graph $\tilde{G}_i$ is recursively composed of 4 $(i-1)$-level graphs $\tilde{G}_{i-1}^1, \tilde{G}_{i-1}^2, \tilde{G}_{i-1}^3, \tilde{G}_{i-1}^4$.

With each vertex v of $\tilde{G}_i$ we associate a corresponding call in the network H_i recursively.

If $v \in \tilde{G}_{i-1}^j$ then v is recursively associated with a call in H_{i-1}^j $(1 \leq j \leq 4)$.

For $i = 0$, the single vertex of $\tilde{G}_0$ is associated with the degenerate call (u, u) where u is the single vertex of H_0.

Let $v \in \tilde{G}_{i-1}^j$. Consider the associated call $c_{i-1}(v) = (s_{i-1}(v), t_{i-1}(v))$ in H_{i-1}^j. We define the associated call in H_i as the call $c_i(v) = (s_i(v), t_i(v))$ where $s_i(v) = s_{i-1}(v)$ and $t_i(v)$ is the output in the slice of the brick wall contained in H_i corresponding to the input on which $t_{i-1}(v)$ is directed.

Lemma 19 *1. Two nodes v_1, v_2 of $\tilde{G}_i$ are adjacent iff $c_i(v_1), c_i(v_2)$ are inconsistent in H_i.*

2. Given an independent set T in $\tilde{G}_i$ then all calls in the set $C(T) = \{c_i(v); v \in T\}$ can be routed consistently in H_i.

Proof. The proof is by induction on i. For $i = 0$ the claims are trivial.

Assume the claim holds for $i - 1$.

First we prove the first part of the lemma.

Let us consider first the case in which two nodes v_1 and v_2 are in the same $(i-1)$-level graph. The corresponding calls $c_{i-1}(v_1)$ and $c_{i-1}(v_2)$ in the associated $(i-1)$-level network enter the brick wall of H_i in the same slice and the order between $t_{i-1}(v_1)$ and $t_{i-1}(v_2)$ is maintained between $t_i(v_1)$ and $t_i(v_2)$. Therefore, $c_i(v_1)$ and $c_i(v_2)$ are consistent iff $c_{i-1}(v_1)$ and $c_{i-1}(v_2)$ are consistent, and thus the claim holds for the inductive hypothesis.

We are left to consider pairs of nodes in different $(i-1)$-level graphs.

Let v_1 and v_2 be nodes from two different $(i-1)$-level graphs, with the exception of $\tilde{G}_{i-1}^3$ and $\tilde{G}_{i-1}^4$ that are never presented together in the preemptive on-line independent set lower bound.

Therefore $c_{i-1}(v_1)$ and $c_{i-1}(v_2)$ enter the brick wall of network H_i in two different slices. It follows that the two calls $c_i(v_1)$ and $c_i(v_2)$ are inconsistent iff the two corresponding slices are intersecting and thus iff two graphs are non-adjacent. This ends the proof of the first part of the lemma.

Now we turn to prove the second part of the lemma. We will prove the claim by induction on the level i of the network construction. For level i the independent set T is of size at most 2^i. For $i = 0$ there is only one call and the claim is obvious.

Consider $i > 0$. The vertices of T can be partitioned into two subsets T_1 and T_2, belonging to two non-adjacent level $(i-1)$ graphs (each of size at most 2^{i-1}). Notice that the two non-adjacent graphs are either $\tilde{G}^1_{i-1}$ and $\tilde{G}^4_{i-1}$ or $\tilde{G}^2_{i-1}$ and $\tilde{G}^3_{i-1}$.

Therefore, w.l.o.g we can assume that T_1 belongs to $\tilde{G}^1_{i-1}$ and T_2 belongs to $\tilde{G}^4_{i-1}$, and thus for every vertex $v \in T_1$, $c_{i-1}(v)$ belongs to H^1_{i-1} and for every vertex $v \in T_2$, $c_{i-1}(v)$ belongs to H^4_{i-1}.

By induction, the set of calls $\{c_{i-1}(v); v \in T_1\}$ can be routed consistently through H^1_{i-1}, and the set of calls $\{c_{i-1}(v); v \in T_2\}$ can be routed consistently through H^4_{i-1}.

The output vertices of these calls in H^1_{i-1} and H^4_{i-1} are all distinct, and so are the input vertices in the brick wall of H_i into which these calls are directed. Since the slices of the brick wall associated with H^1_{i-1} and H^4_{i-1} are non-intersecting, and since for each slice the order of the corresponding outputs in the brick wall of H_i is maintained, we have that every pair of calls in $C(T)$ can be routed consistently through the brick wall of H_i.

Since the number of such calls is at most 2^i, and thus at most the height of the brick wall of H_i, all calls in $C(T)$ can be routed consistently.

The case in which T_1 belongs to $\tilde{G}^2_{i-1}$ and T_2 belongs to $\tilde{G}^3_{i-1}$ is similarly proved. ∎

The next lemma gives a bound on the number of vertices in a i-level network.

Claim 20 *The number of vertices/edges in H_i is $N(i) = O(6^i)$.*

Proof. Let α be some constant such that a brick wall of height h and width w has at most $\alpha h w$ vertices/edges. We show by induction on i that $N(i) \leq 4\alpha \cdot 6^i$. For $i = 0$ the claim is clearly satisfied. Assume that $N(i-1) \leq \alpha \cdot 6^{i-1}$. Since H_i is composed by 4 H_{i-1} networks and a brick wall of width $4 \cdot 3^{i-1}$ and height 2^i, we obtain:

$$N(i) \leq 4\alpha \cdot 3^{i-1} \cdot 2^i + 4 \cdot 4\alpha \cdot 6^{i-1} \leq 4\alpha \cdot 6^i.$$

∎

Using the network construction described we derive from Theorem 10 the following:

Theorem 21 *Any randomized preemptive algorithm for on-line edge-disjoint paths (and thus for call control) on general networks of N vertices has competitive ratio $\Omega(N^{\frac{1-\log_4 3}{\log_4 6}})$.*

From Theorems 21 and 16 we thus derive the following corollary:

Corollary 22 *Any preemptive randomized algorithm for on-line call control on general networks of N vertices and minimum degree δ has competitive ratio $\Omega(\left(\frac{N}{\delta}\right)^{\frac{1-\log_4 3}{\log_4 6}})$.*

5 The on-line graph coloring problem

In this section we will consider the on-line graph coloring problem. We present a lower bound of $\Omega(n^{\epsilon})$ on the competitive ratio of randomized on-line algorithms. We recall that in our model the algorithm knows the graph $G = (V, E)$ from which the instance of the problem is drawn.

Vertices of V are presented over time. Each time a new vertex is communicated, the algorithm must answer with a color for the vertex. The goal is to color the set of vertices presented with as few colors as possible so that any two adjacent vertices have different colors.

In Section 5.1 we present an $\Omega(n^{\epsilon})$ lower bound for the on-line graph coloring problem, and in Section 5.2 we give upper bounds for the problem.

5.1 Lower Bounds for on-line graph coloring

The structure of the graph from which the input sequence is chosen is recursive. The basic recursive construction of the graph is equal to the recursive construction of the graph $\tilde{G}$ for the preemptive randomized lower bound for the on-line independent set problem of Section 3.2 described in Figure 2.

As in Sections 3.1 and 3.2 define a probability distribution on the request sequences of $\tilde{G}$. For any on-line non-preemptive algorithm ON we give a lower bound on the ratio ρ between the expected optimal off-line benefit E(OPT) and the expected on-line benefit E(ON).

The probability distribution is defined recursively. The distribution for a graph G_0 of level 0 consists of giving deterministically the single vertex of the graph.

The probability distribution on the input sequences for a graph G_i is given by the following two step process:

1. The probability distribution on the input sequences for $\tilde{G}^3_{i-1}$ and $\tilde{G}^4_{i-1}$.

2. Toss a fair coin:

    - If tails, end the sequence.
    - If heads, the probability distribution on the input sequences for G^1_{i-1} and G^2_{i-1}.

First we compute the expected size of the off-line optimal solution.

Lemma 23 *The expected off-line optimal number of colors for any input sequence from the distribution for $\tilde{G}_i$ is* $\mathrm{E}[\mathrm{OPT}(i)] = \left(\frac{3}{2}\right)^i$.

Proof. The claim is proved by induction. It is clearly true for $i = 0$. Assume that the chromatic number for a graph $\tilde{G}_{i-1}$ of level $i-1$ is $\mathrm{E}[\mathrm{OPT}(i-1)] = \left(\frac{3}{2}\right)^{i-1}$.

With probability 1/2 the probability distribution ends after the first step of the sequence. Therefore the set of presented vertices from $\tilde{G}_i$ is formed by the union of the input sequences for the two graphs $\tilde{G}^3_{i-1}$ and $\tilde{G}^4_{i-1}$. Since the two graphs are non-adjacent, the same set of colors can be used for both. Therefore the expected number of colors is $\left(\frac{3}{2}\right)^{i-1}$.

With probability 1/2, the set of presented vertices from $\tilde{G}_i$ is formed by the union of the input sequences for the four graphs $\tilde{G}^1_{i-1}$, $\tilde{G}^2_{i-1}$, $\tilde{G}^3_{i-1}$ and $\tilde{G}^4_{i-1}$. The set of colors used for $\tilde{G}^3_{i-1}$ must be disjoint from the set of colors used for $\tilde{G}^1_{i-1}$. Therefore the expected number of colors necessary for the vertices in the input sequence for $\tilde{G}_i$ is at least $2 \cdot \left(\frac{3}{2}\right)^{i-1}$. This number of colors is also sufficient to color the whole sequence of presented vertices. In fact, the colors used for $\tilde{G}^1_{i-1}$ can also be used for the graph $\tilde{G}^4_{i-1}$, and the colors used for $\tilde{G}^3_{i-1}$ can also be used for the graph $\tilde{G}^2_{i-1}$.

The number of colors of the off-line optimal solution is thus given by:

$$\mathrm{E}[\mathrm{OPT}(i)] = \frac{1}{2}\left(\frac{3}{2}\right)^{i-1} + \frac{1}{2}\cdot 2\left(\frac{3}{2}\right)^{i-1} = \left(\frac{3}{2}\right)^{i}.$$

■

Let us turn the attention to the on-line expected number of colors.

Lemma 24 *The expected number of colors of any deterministic on-line algorithm for the distribution over the input sequences for $\tilde{G}_i$ is* $\mathrm{E}[\mathrm{ON}(i)] \geq 2^i$.

Proof. We prove the claim by induction. The claim is true for $i = 0$. Let $\mathrm{ON}^j(i-1)$ be the number of colors used by a deterministic on-line algorithm for the graph $\tilde{G}^j_{i-1}$ of level $i-1$. By induction, the expected number of colors used for a graph $\tilde{G}^j_{i-1}$ of level $i-1$ is $\mathrm{E}[\mathrm{ON}^j(i-1)] \geq 2^{i-1}$. Let c_{34} be a random variable indicating the number of colors used by ON for both vertices in $\tilde{G}^3_{i-1}$ and in $\tilde{G}^4_{i-1}$ during the first step of the input sequence.

If the outcome of the coin toss is heads, then the expected number of colors used by the on-line algorithm is $\mathrm{E}[\mathrm{ON}^3(i-1)+\mathrm{ON}^4(i-1)-c_{34}]$.

If the outcome of the coin toss is tails, then the algorithm must assign colors to the presented vertices of $\tilde{G}^1(i-1)$ and $\tilde{G}^2(i-1)$. The colors used in the first step for vertices in $\tilde{G}^4_{i-1}$ can be used in the second step for vertices in $\tilde{G}^1_{i-1}$ if they have not been used for $\tilde{G}^3_{i-1}$. On the other side, the colors used in the first step for vertices in $\tilde{G}^3_{i-1}$ can be used in the second step for vertices in $\tilde{G}^2_{i-1}$ if they have not been used for $\tilde{G}^4_{i-1}$. Therefore, at least $\mathrm{ON}^1(i-1)-(\mathrm{ON}^4(i-1))-c_{34})$ new colors have to be used for presented vertices in $\tilde{G}^1_{i-1}$, and at least $\mathrm{ON}^2(i-1)-(\mathrm{ON}^3(i-1))-c_{34})$ new colors have to be used for presented vertices in $\tilde{G}^3_{i-1}$.

The claim now follows from the following:

$$\begin{aligned}
\mathrm{E}[\mathrm{ON}(i)] &\geq \frac{1}{2}\mathrm{E}[\mathrm{ON}^3(i-1)+\mathrm{ON}^4(i-1)-c_{34}] \\
&+ \frac{1}{2}\mathrm{E}[\mathrm{ON}^3(i-1)+\mathrm{ON}^4(i-1)-c_{34}] \\
&+ \frac{1}{2}\mathrm{E}[(\mathrm{ON}^1(i-1)-(\mathrm{ON}^4(i-1))-c_{34})+(\mathrm{ON}^2(i-1)-(\mathrm{ON}^3(i-1))-c_{34})] \\
&= \frac{1}{2}\mathrm{E}[\mathrm{ON}^1(i-1)+\mathrm{ON}^2(i-1)+\mathrm{ON}^3(i-1)+\mathrm{ON}^4(i-1)] \\
&\geq 2^i.
\end{aligned}$$

■

Theorem 25 *Any randomized algorithm for the on-line graph coloring problem in a graph of n vertices has competitive ratio $\Omega(n^{1-\log_4 3})$.*

Proof. The number of vertices of the graph $\tilde{G}_i$ is $n(i)=4^i$. By Lemma 23 and 24, the expected competitive ratio $\rho(i)$ on the input sequence from the distribution for a graph $\tilde{G}_i$ is:

$$\rho(i) \geq \frac{\mathrm{E}[\mathrm{ON}(i)]}{\mathrm{E}[\mathrm{OPT}(i)]} \geq \frac{(2)^i}{\left(\frac{3}{2}\right)^i} = \frac{n(i)}{3^{\log_4 n(i)}} = n(i)^{1-\log_4 3}.$$

■

5.2 Upper Bounds for on-line graph coloring

Using similar techniques as in Section 3.3 we show that off-line algorithms for maximum independent set can also be used to obtain deterministic on-line graph coloring algorithms.

Let $G=(V,E)$ be the graph that we assume is known to the on-line algorithm. The algorithm requires a preprocessing phase similar to the algorithm presented in Section 3.3 for on-line induced subgraph. A collection of k repetitive solutions $I_1, I_2, \ldots, I_k$ of the maximum independent set problem for the graph G are computed, as follows.

Let $G_1 = G$, and $G_{i+1} = G_i/I_i$ be the result of the operation that removes from G_i all vertices in I_i and all the incident edges.

I_i is the set of vertices forming the solution of the independent set problem returned by applying the $\frac{n}{\alpha}$-approximate algorithm to the graph G_i. The process of computing solutions is continued until $|I_{k+1}| \leq \sqrt{\alpha}$. Let $\overline{I} = V - \{I_1 \cup I_2 \cup \ldots \cup I_k\}$ be the set of vertices of G_{k+1}.

The algorithm colors every presented vertex v as follows:

1. If there exist $i \leq k$ such that $v \in I_i$: assign color i to vertex v.
2. Otherwise, if $v \in \overline{I}$: assign the first color $j > k$ not used for any previously presented vertex $u \in \overline{I}$, $(u, v) \in E$.

Theorem 26 *Given a $\frac{n}{\alpha}$-approximation (deterministic) algorithm for maximum independent set, there exists a (deterministic) $O(\frac{n}{\sqrt{\alpha}})$-competitive algorithm for on-line minimum graph coloring. If the original algorithm is polynomial and deterministic, so is the on-line algorithm.*

Proof. First note that k is at most $\frac{n}{\sqrt{\alpha}}$. Next, observe that the maximum size of an independent set in $\overline{I}$ is at most $\frac{n}{\sqrt{\alpha}}$, since the size of the solution of the $\frac{n}{\alpha}$-approximate algorithm applied to the graph induced by $\overline{I}$ is less than $\sqrt{\alpha}$. Let $\overline{I}'$ be the subset of $\overline{I}$ presented in the sequence.

If the sequence is not empty, the number of colors of the optimal solution is at least the size of $\overline{I}'$ divided by the size of a maximum independent set in $\overline{I}'$. Therefore

$$\mathrm{OPT} \ \geq \ \max\{1, \frac{\sqrt{\alpha}}{n}|\overline{I}'|\}.$$

The number of colors used by the on-line algorithm is bounded by

$$\mathrm{ON} \ \leq \ k + |\overline{I}'| \leq 2\frac{n}{\sqrt{\alpha}}\mathrm{OPT},$$

thus proving the theorem. ∎

Using the optimal (non-polynomial time) algorithm for maximum independent set (e.g. $\alpha = n$) we obtain:

Corollary 27 *There exists a deterministic $O(\sqrt{n})$-competitive algorithm for the on-line graph coloring.*

6 Lower bound for on-line path coloring

In this section we derive a $\Omega(n^{\epsilon})$ lower bound on the competitive ratio of randomized algorithms for on-line path coloring. At every step the adversary presents a pair of vertices in a network. The algorithm must choose a color to be assigned to the pair of vertices. The goal is to use as few colors as possible so that all the pairs assigned with the same color can be connected with mutually edge-disjoint paths. As in the Section 39 for the on-line edge-disjoint path problem, we refer in the following to pairs as calls.

The lower bound for on-line path coloring is obtained by embedding the graph $\tilde{G}$ and the input sequence on which the lower bound for on-line graph coloring is obtained in a brick wall W. The structure of the graph $\tilde{G}$ of n vertices for on-line graph coloring is the recursive structure described in Figure 2 of Section 3. The embedding of the recursive structure of $\tilde{G}$ in a brick wall W is also described in Figure 6 of Section 4.2.

The embedding associates a top-to-bottom call in the brick wall at each vertex of the graph, so that each vertex is associated with one and only one call, and two vertices are adjacent if and only if the two corresponding calls are intersecting. Therefore, any input sequence in the graph $\tilde{G}$ for graph coloring can be directly translated in an input sequence for path coloring in the brick wall W.

The maximum size of an independent set in a graph $\tilde{G}$ of level i with 4^i vertices is 2^i. Therefore we can embed the on-line graph coloring lower bound in a brick wall W of width n and height $\sqrt{n}$.

Thus, we obtain the following lower bound:

Theorem 28 *Any randomized algorithm for on-line path coloring on general networks of N vertices has competitive ratio $\Omega(N^{\frac{2}{3}(1-\log_4 3)})$.*

Proof. The theorem follows from theorem 25 and by observing that an input sequence for graph coloring in the graph $\tilde{G}$ of n vertices can be embedded in a brick wall W of $N = n^{\frac{3}{2}}$ vertices. ∎

7 Lower bounds for optical networks

As pointed out in the introduction, a lower bound for on-line edge-disjoint path is also a lower bound for on-line routing in reconfigurable networks – benefit version, and a lower bound for on-line path coloring is also a lower bound for on-line routing in reconfigurable networks – coloring version. Therefore, from Theorems 15, 21 and 28, we derive the following three corollaries.

Corollary 29 *Any non-preemptive randomized algorithm for on-line routing in reconfigurable optical networks – benefit version on general networks of N vertices has competitive ratio $\Omega(N^{\frac{1-\log_3 2}{1+\log_3 2}})$.*

Corollary 30 *Any preemptive randomized algorithm for on-line routing in reconfigurable optical networks – benefit version on general networks of N vertices has competitive ratio $\Omega(N^{\frac{1-\log_4 3}{\log_4 6}})$.*

Corollary 31 *Any randomized algorithm for on-line routing in reconfigurable optical networks — coloring version on general networks of N vertices has competitive ratio $\Omega(N^{\frac{2}{3}(1-\log_4 3)})$.*

In what follows we present a simple result (Lemma 35) that allows to reduce maximum independent set to routing in switchless optical networks – benefit version and minimum graph coloring to routing in switchless optical networks – coloring version. This reduction is from a graph of n vertices to a network of $O(n)$ vertices. Therefore, from Theorems 7, 10 and 25 we respectively derive the following three corollaries.

Corollary 32 *Any non-preemptive randomized algorithm for on-line routing in switchless optical networks – benefit version has competitive ratio $\Omega(n^{1-\log_3 2})$.*

Corollary 33 *Any preemptive randomized algorithm for on-line routing in switchless optical networks – benefit version has competitive ratio $\Omega(n^{1-\log_4 3})$.*

Corollary 34 *Any randomized algorithm for on-line routing in switchless optical networks — coloring version, has competitive ratio $\Omega(n^{1-\log_4 3})$.*

The required reductions are derived from the following lemma.

Lemma 35 *For any undirected graph $G = (V, E)$ there exists a directed network $D' = (V', E')$ and a one-to-one mapping from vertices v in G to calls $(s(v), t(v))$ in D', so that two vertices $v_i, v_j \in V$ are adjacent if and only if the two corresponding calls $(s(v_i), t(v_i))$, $(s(v_j), t(v_j))$ are interfering.*

Proof. Let n be the number of vertices of $V = \{v_1, \ldots, v_n\}$. D' is a bipartite graph whose set of vertices V' is formed by the union of two sets of n vertices, $S = \{s_1, \ldots, s_n\}$ and $T = \{t_1, \ldots, t_n\}$. The one-to-one mapping is defined by $s(v_i) = s_i$ and $t(v_i) = t_i$, $i = 1, \ldots, n$. The set of directed edges of D' is $E' = \{(s(v), t(v)) | v \in V\} \cup \{(s(v_i), t(v_j)) | (v_i, v_j) \in E\}$.

For any two vertices $v_i, v_j \in V$ consider the two calls $(s(v_i), t(v_i))$ and $(s(v_j), t(v_j))$ in D'. If v_i and v_j are adjacent in E, then the bipartite graph D' contains the two edges $(s(v_i), t(v_j))$ and $(s(v_j), t(v_i))$. Therefore, $t(v_j) \in R(s(v_i))$ and $t(v_i) \in R(s(v_j))$, implying that the two calls are interfering. For the other direction, if v_i and v_j are non-adjacent then $t(v_j) \notin R(s(v_i))$ and $t(v_j) \notin R(s(v_i))$. Since the graph D' is bipartite, with all edges directed from S to T, follows that $u_2 \notin R(v_1)$ and $u_1 \notin R(v_2)$, implying that the two calls are non-interfering. ∎

8 Lower bounds for on-line edge-disjoint fixed paths

In this section we present tight lower bounds for on-line edge-disjoint fixed paths, an $\Omega(\sqrt{n})$ lower bound for *meshes* and an $\Omega(n)$ lower bound for *general networks*. We use the terminology of call control.

We make use of Yao's Lemma (see Section 2), and establish a lower bound for any deterministic algorithm on a given probability distribution. We first describe the kind of probability distribution, embedded in next subsections on different topologies.

Every call c_i is specified by a path p_i connecting endpoints s_i and t_i. Calls c_i and c_j are said *inconsistent* if p_i and p_j intersect on at least one edge ($p_i \cap p_j \neq \emptyset$), *consistent* otherwise ($p_i \cap p_j = \emptyset$).

The structure of the sequence on which the lower bounds are given has the form described below.

The sequence is formed of $2k$ calls presented in k steps according to the following probability distribution:

1. At step j, $j = 1, \dots, k$, two inconsistent calls c_j^1 and c_j^2 are presented.
2. Toss a fair coin.
   - If heads, all the calls presented later will be consistent with c_j^1 and inconsistent with c_j^2.
   - If tails, all the calls presented later will be consistent with c_j^2 and inconsistent with c_j^1.

Lemma 36 *Any preemptive randomized algorithm for on-line edge-disjoint fixed paths has competitive ratio $\Omega(k)$ on a probability distribution of k steps as described above.*

Proof. The optimal benefit does not depend over the coin tosses. There is a maximal set of k consistent calls, one for every step of the sequence. The claim follows by proving an $O(1)$ upper bound on the expected on-line benefit.

If a deterministic algorithm has accepted j calls, the probability that any single future call is consistent with all the accepted ones is no more than $\frac{1}{2^j}$, independently from call preemption. The expected benefit of any deterministic algorithm is thus $\mathrm{ON} \leq \sum_{j=0}^{k-1} \frac{1}{2^j} \leq 2$. ∎

What remains to show is how a sequence of the type described above can be constructed for specific networks. The obvious question is how large can k be with respect to the size of the network.

In the two following subsections we deal with meshes and general networks.

8.1 Lower bound on meshes

Consider two dimensional $\sqrt{n} \times \sqrt{n}$ mesh. Let the set of nodes be $\{(u,v) | 1 \leq u \leq \sqrt{n}, 1 \leq v \leq \sqrt{n}\}$. Every vertex is denoted with a pair, the first item is the row index and the second item is the column index.

For our construction we use a network (see Figure 8 where the row index increases from top to bottom, the column index from left to right) derived from a two dimensional mesh via the two following operations:

1. Remove all vertices (u,v) with $v > u$, and the adjacent edges.
2. Insert the edges $\{((u,v),(u-1,v+1)) | 3 \leq u \leq \sqrt{n}, 1 \leq v \leq \sqrt{n}-1\}$.

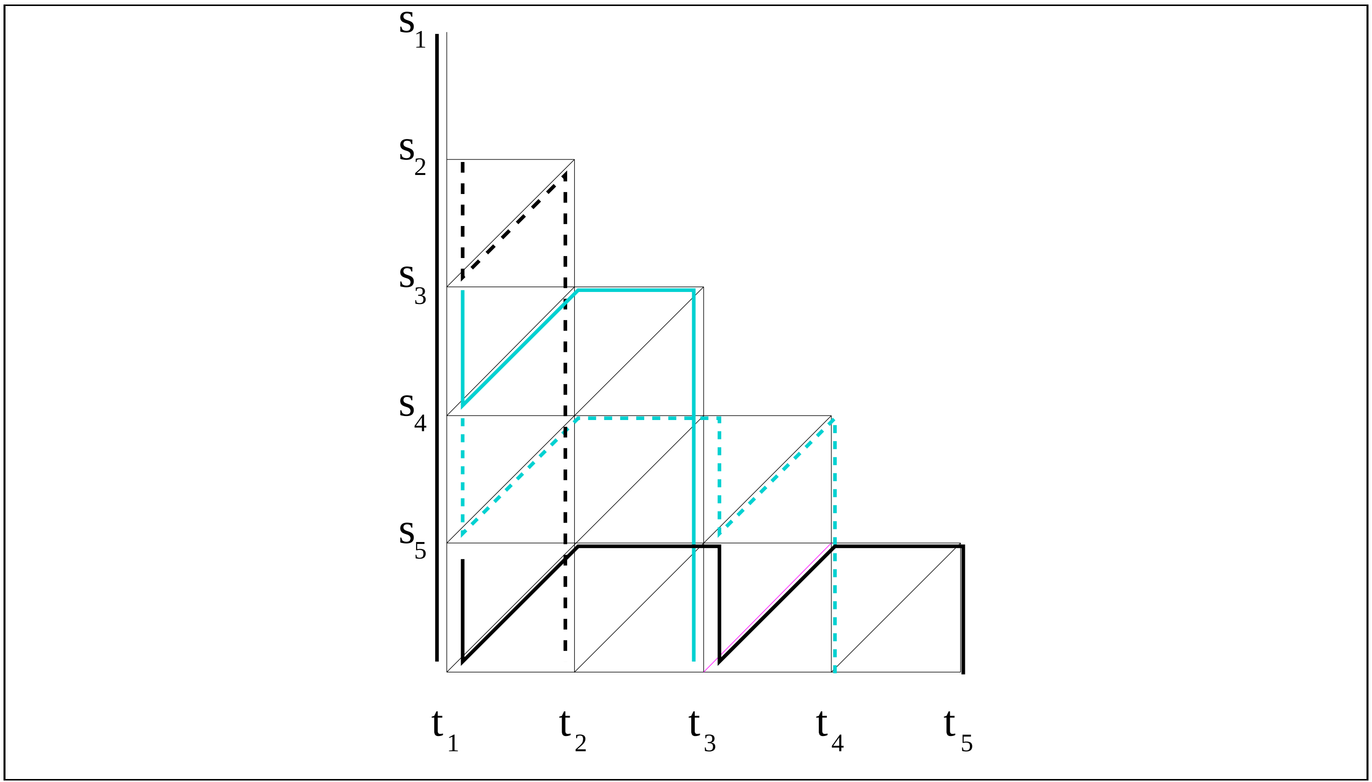

Figure 8: The network for the lower bound on meshes

We remark that for our purpose such network can always be embedded in a mesh of size $2\sqrt{n} \times 2\sqrt{n}$.

We prove that any instance drawn from the probability distribution for the $\Omega(k)$ lower bound of the previous section can be implemented on this network with $k = \frac{\sqrt{n}-1}{2}$. This is done by presenting a sequence of $\sqrt{n}-1$ calls, the path of each call is chosen in order to make it consistent or inconsistent with any other previous call in the sequence, as required from the outcome of the coin tosses.

Call c_i in the sequence is defined by a pair of vertices (s_i, t_i) and by a path p_i that connects s_i to t_i. Let $C_i = \{j : j < i, p_i \cap p_j = \emptyset\}$ be the set of previous calls in the input sequence from the probability distribution that are consistent with c_i and let $I_i = \{j : j < i, p_i \cap p_j \neq \emptyset\}$ be the set of previous calls that are inconsistent with c_i.

Call c_i of the sequence, $i = 1, \ldots, \sqrt{k}-1$, is defined as follows:

- $s_i = (i, 1)$;
- $t_i = (\sqrt{n}, i)$;
- p_i is formed by the following sets of edges:
    1. $E_i^1 = \{((i,j),(i,j+1)) | j \in C_i\}$;
    2. $E_i^2 = \{((i,j),(i+1,j))$ and $((i+1,j),(i,j+1)) | j \in I_i\}$;
    3. $E_i^3 = \{((j,i),(j+1,i)) | i \leq j \leq \sqrt{n}-1\}$

Observe that the set of edges forming p_i, i.e., $E_i^1 \cup E_i^2 \cup E_i^3$, is actually a path from s_i to t_i. For $j = 1, \ldots, i-1$, vertex (i,j) is connected to vertex $(i, j+1)$ either directly if $j \in C_i$, or through the two-edge path formed by $((i,j),(i+1,j))$ and $((i+1,j),(i,j+1))$ if $j \in I_i$. For $j = i, \ldots, \sqrt{n}-1$, vertex (j,i) is directly connected to vertex $(j+1,i)$. Moreover, the first vertex of the path is $s_i = (i,1)$ and the last vertex is $(\sqrt{n}, i)$.

Figure 8 shows a network with a sequence of 5 calls and the relative paths, where c_1 and c_3 are inconsistent and c_2 and c_4 consistent with respect to following calls.

Lemma 37 *Let c_j and c_i, $j < i$, be two calls in the sequence. p_i and p_j do not intersect if and only if $j \in C_i$.*

Proof.

First we show that if $j \in I_i$ then c_i and c_j intersect in one edge, namely $((i,j),(i+1,j))$. In fact, this edge always belongs to p_j, while it belongs to p_i only if c_j is inconsistent with c_i.

Next, we show that if $j \in C_i$ then p_i and p_j do not intersect in any edge. For any h, all edges in $E_h^1 \cup E_h^2$ are either between two vertices in row h or between a vertex in row h and a vertex in row $h+1$. Hence, the set $E_i^1 \cup E_i^2$ do not intersect with the set $E_j^1 \cup E_j^2$.

Since all edges in p_j have column at most j, and all edges in E_i^3 lie on the column $i > j$, the two sets do not intersect.

It remains to show that the edges in E_j^3 do not intersect with the edges in $E_i^1 \cup E_i^2$. This follows since the edges in E_j^3 lie in column j whereas since $j \in C_i$ no edge in $E_i^1 \cup E_i^2$ lies on column j. ■

We get the following theorem.

Theorem 38 *Any preemptive randomized algorithm for on-line edge-disjoint with fixed paths on meshes of n nodes has competitive ratio $\Omega(\sqrt{n})$.*

8.2 Lower bound on general networks

We prove an $\Omega(n)$ lower bound on the competitive ratio of randomized algorithms for the on-line edge-disjoint fixed paths problem in general networks by showing that the probability distribution of Theorem 36 can be constructed for a sequence of $\Omega(n)$ steps, in a complete graph with $O(n)$ vertices.

Let n be a *prime number* and let $V \cup V'$ with $V = \{v(0), \ldots, v(n-1)\}$ and $V' = \{v'(0), \ldots, v'(n-1)\}$ be the set of vertices of a complete graph G.

We denote:

$$n_h^l = (h \cdot l) \bmod n,$$

with $h = 1, \ldots, \lfloor \frac{n-1}{2} \rfloor$ and $l = 1, \ldots, n$.

The two following lemmas follow from simple algebraic properties of prime numbers.

Lemma 39 *Let (h, l) and (h', l') be two pairs of integers with $h, h' \in \{1, \ldots, \frac{n-1}{2}\}$, $h \neq h'$ and $l, l' \in \{1, \ldots, n-1\}$. Then, $\{n_h^l, n_h^{l+1}\} \neq \{n_{h'}^{l'}, n_{h'}^{l'+1}\}$.*

Lemma 40 *For any integer $h = 1, \ldots, \frac{n-1}{2}$ it holds*

$$\{n_h^l | l = 0, \ldots, n-1\} = \{0, 1, \ldots, n-1\}.$$

Let c_i be the i-th call of the sequence, and let $C_i = \{j : j < i, p_i \cap p_j = \emptyset\}$ be the set of previous calls that are consistent with c_i and let $I_i = \{j : j < i, p_i \cap p_j \neq \emptyset\}$ be the set of previous calls that are inconsistent with c_i.

With every pair of calls c_i, c_j, $i = 1, \ldots, j-1$, $j = 1, \ldots, k$, two values s_{ij} and s'_{ij} are associated, $s_{ij} \in \{0, \ldots, n-2\}$ and $s'_{ij} \in \{0. \ldots, n-1\}$.

The idea behind the construction presented in the sequel of this section is the following. The generic path p_i associated with call c_i crosses all vertices of set V in the sequence $v(n_i^0), \ldots, v(n_i^{n-1})$ (by Lemma 40 such a sequence spans the set V). Consider two calls c_i and c_j, $i < j$, with associated values s_{ij} and s'_{ij}. By Lemma 40, there exist two integer values q and r such that $n_i^q = n_j^r = s_{ij}$. In our construction, path p_i always contains edges $(v(s_{ij}), v'(s'_{ij}))$ and $(v'(s'_{ij}), v(n_i^{q+1}))$. If $c_i \in C_j$ then path p_j must not share any edge with p_i. In this case vertex $v(s_{ij})$ is followed by $v(n_j^{r+1})$ in p_j. Otherwise, if $c_i \in I_j$ then path p_j contains edges $(v(s_{ij}), v'(s'_{ij}))$ and $(v'(s'_{ij}), v(n_j^{r+1}))$, the first of which is common with p_i.

To ensure that two consistent calls do not share any edge we use Lemma 39 that states that two integers between 0 and $n-1$ are adjacent only in one sequence; furthermore the association of values s_{ij} and s'_{ij} to any pair of calls c_i, c_j is to guarantee that any edge between a vertex in V and a vertex in V', included in some path p_i, and dedicated to the possible intersection with path p_j, does not not appear in any other path.

Assign values s_{ij} and s'_{ij} to any pair of calls c_i, c_j, $i < j$, in order of increasing j and for a fixed j in order of increasing i. Let $n_i^q = n_j^r = s_{ij}$.

Consider a pair x, y, $x < y \leq j$ for which the values s_{xy} and s'_{xy} have been assigned prior to the assignment for i, j. Let l and m be integers such that $n_x^l = n_y^m = s_{xy}$. The following are the conditions on the assignment for i, j:

1. Assign value s_{ij} different from: s_{xy} with $y = i$, $y = j$ or $x = i$.
2. Assign value s'_{ij} different from: s'_{xy} if $\{s_{xy}, n_x^{l+1}, n_y^{m+1}\} \cap \{s_{ij}, n_i^{q+1}, n_j^{r+1}\} \neq \emptyset$ for any previously assigned pair xy.

Lemma 41 *There exists a sequence of $k = \Omega(n)$ calls such that for every pair c_i, c_j, $i < j$, values s_{ij} and s'_{ij} can be assigned obeying Conditions 1 and 2.*

Proof. We prove that there exists a value $k = \Omega(n)$ such that, in the worst case, at least one possibility to assign values s_{ij} and s'_{ij} is left for any pair of calls c_i, c_j with $i < j \leq k$.

Denote by x, y any pair considered before i, j in the designed order. Then either $x < y < j$ or $y = j$ and $x < i$.

Following Condition 1, we can bound the number of pairs such that $y = i$, $y = j$ or $x = i$ by $(i-1) + (i-1) + (j-i) \leq 2(k-2)$. Therefore, if $n > 2(k-2)$ at least one possibility is left for assigning s_{ij}.

Following Condition 2, we would like to bound the number of pairs x, y for which one of the three values in $\{s_{ij}, n_i^{q+1}, n_j^{r+1}\}$ appears in $\{s_{xy}, n_x^{l+1}, n_y^{m+1}\}$.

For fixed $h < k$, each of the values in $\{s_{ij}, n_i^{q+1}, n_j^{r+1}\}$ is equal to n_h^d for some integer d. By Condition 1 there could be either one y for which $s_{hy} = n_h^d$ or one x such that $s_{xh} = n_h^d$. Similarly, there could be either one y for which $s_{hy} = n_h^{d-1}$ or one x such that $s_{xh} = n_h^{d-1}$. Thus we get that h can be involved in at most 6 pairs for which the above restriction holds.

Hence, if $n > 6(k-1)$ there is still a possibility left for assigning s'_{ij}.

The above observations show that by setting $k = n/6$ then values s_{ij} and s'_{ij} can be assigned to each pair of calls c_i, c_j. ∎

In the following we formally define call c_i, $i = 1, \ldots, k$. The endpoints s_i, t_i of call c_i are:

- $s_i = v(n_i^0)$;
- $t_i = v(n_i^{n-1})$;

Path p_i associated with call c_i contains the following set of edges:

1. $(v(n_i^l), v(n_i^{l+1}))$, for any $l \in \{1, \ldots, n-1\}$ such that:
   (a) For all $j < i$, $n_i^l \neq s_{ji}$ and for all $j > i$, $n_i^l \neq s_{ij}$.
   (b) There exists $j < i$ such that $n_i^l = s_{ji}$ and $j \in C_i$.
2. $(v(n_i^l), v'(s'_{ij}))$ and $(v'(s'_{ij}), v(n_i^{l+1}))$, for any $l \in \{1, \ldots, n-1\}$ such that there exists $j > i$ for which $s_{ij} = n_i^l$;
3. $(v(n_i^l), v'(s'_{ji}))$ and $(v'(s'_{ji}), v(n_i^{l+1}))$, for any $l \in \{1, \ldots, n-1\}$ such that there exists $j < i$ for which $s_{ji} = n_i^l$ and $j \in I_i$.

The validity of the above definition follows from Condition 1 in the assignment procedure that ensures that n_i^l can be equal to only one value of the form s_{ij} or s_{ji}.

Lemma 42 *Let c_i, c_j, with $i < j$ be two calls in the sequence. p_i and p_j do not intersect if and only if $i \in C_j$.*

Proof. First we prove that for any $i \in I_j$, the paths p_i and p_j intersect. Let $n_j^p = n_i^m = s_{ij}$. Then the edge $(v(s_{ij}), v'(s'_{ij}))$ is in p_i by rule 2 and in p_j by rule 3.

We now turn to prove that if $c_i \in C_j$ then the two paths p_i and p_j do not intersect. Lemma 39 ensures that the edges between two vertices in V (rule 1) cannot be shared by two paths.

Edges in p_i between vertices in V and vertices in V' are either of the form $(v(s_{ih}), v'(s'_{ih}))$ or $(v'(s'_{ih}), v(n_i^{d+1}))$ for some $h > i$ and d such that $s_{ih} = n_i^d$ (by rule 2), or $(v(s_{hi}), v'(s'_{hi}))$ or $(v'(s'_{hi}), v(n_i^{d+1}))$, for some $h < i$ and d such that $s_{hi} = n_i^d$ (by rule 3). Thus one end-point of each such edge is of the form s'_{xy} and the other one in $\{s_{xy}, n_x^{l+1}, n_y^{m+1}\}$ for a pair in which one of x, y is equal to i and $n_x^l = n_y^m = s_{xy}$. A similar statement holds for the path p_j. By Conditions 2 in the assignment procedure for s'_{ij}, such edges cannot appear in more than one path and hence the claim follows. ∎

We conclude with the following theorem:

Theorem 43 *Any preemptive randomized algorithm for on-line edge-disjoint fixed paths in a network of n nodes has competitive ratio $\Omega(n)$.*

Acknowledgments

The authors would like to thank Yossi Azar, Anna Karlin and Alberto Marchetti-Spaccamela.

References

[ABC+94] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswani, B. Schieber and M. Sudan. Efficient Routing and Scheduling Algorithms for Optical Networks. In *Proc. of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 412–423, 1994.

[AAP93] B. Awerbuch, Y. Azar and S. Plotkin. Throughput Competitive On-line Routing. In *Proc. of the 34th Annual Symposium on Foundations of Computer Science*, November 1993.

[ABFR94] B. Awerbuch, Y. Bartal, A. Fiat and A. Rosén. Competitive Non-preemptive Call Control. In *Proc. of 5th ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[AGLR94] B. Awerbuch, R. Gawlick, F.T. Leighton and Y. Rabani. On-line Admission Control and Circuit Routing for High Performance Computing and Communication. In *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, 1994.

[BBKTW90] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos and A. Widgerson. On the Power of Randomization in On-line Algorithms. *Proc of the 22nd Annual ACM Symposium on Theory of Computing*, 1990.

[BH92] R.B. Boppana and M.M. Halldorson. Approximating maximum independent set by excluding subgraphs. *BIT* 32(2), pp.180-196, 1992.

[BH93] R.A. Barry and P.A. Humblet. On the Number of Wavelengths and Switches in All Optical Networks. *IEEE Trans. Comm.*, 1993.

[GG92] J. Garay and I.S. Gopal. Call Preemption in Communications Networks, *Proc. INFOCOM 92*, Florence, Italy, pp. 1043-1050, 1992.

[GGKMY93] J. Garay, I.S. Gopal, S. Kutten, Y. Mansour and M. Yung. Efficient On-line Call Control Algorithms. In *Proc. 2nd Israel Symposium on Theory of Computing and Systems*, pages 285-293, June 1993.

[Hal94] M.M. Halldórsson. Approximation via partitioning. Manuscript, 1994.

[HS92] M.M. Halldórsson and M. Szegedy. Lower Bounds for On-Line Graph Coloring. In *Proc. of the 3rd Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 211-216, January 1992.

[I90] S. Irani. Coloring Inductive Graphs On-Line. In *Proc. of the 31st Ann. IEEE Symp. on Foundations of Computer Science*, pages 470-479, October 1990.

[KS95] V. Kapoulas and P. Spirakis. Randomized Competitive Algorithms for Admission Control in General Networks. In *Proc. of the 15th ACM Principle of Distributed Computing*, pp.253, August 1995.

[KVV90] R.M. Karp, U.V. Vazirani and V.V. Vazirani. An Optimal Algorithm for On-Line Bipartite Matching. In *Proc. of the 22nd Ann. ACM Symp. on Theory of Computing*, pages 352-358, May 1990.

[KT95] J. Kleinberg and E. Tardos. Disjoint Paths in Densely Embedded Graphs. *Proc of the 36-th IEEE Annual Symposium on Foundations of Computer Science*, 1995.

[LY93a] C. Lund and M. Yannakakis. The Approximation of Maximum Subgraph Problems. In *Proc. of the 20th International Colloquium on Automata, Languages and Programming*, pp. 40–51, July 1993.

[LY93b] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. In *Proc. of the 25th ACM Symp. on Theory of Computing*, 1993.

[P92] R.K. Pankay. *Architectures for Linear Light-wave Networks.* PhD thesis, MIT, 1992.

[RU94] P. Raghavan and U. Upfal. Efficient Routing in All-Optical Networks. In *Proc. of the 26th Annual Symposium on Theory of Computing*, pages 133–143, 1994.

[ST85] D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of ACM* 28, 1985.

[V90] S. Vishwanathan. Randomized On-line Graph Coloring, In*Proc. of the 31st IEEE Annual Symposium on Foundations of Computer Science*, pp. 464–469, 1990.

[Y77] A.C. Yao. Probabilistic Computations: Towards a Unified Measure of Complexity. In *Proc. of the 17th Annual Symposium on Foundations of Computer Science*, pp. 222-227, 1977.