Independent Sets in Hypergraphs with Applications to Routing Via Fixed Paths

Noga Alon[1], Uri Arad[2], and Yossi Azar[3]

[1] Department of Mathematics and Computer Science, Tel-Aviv University.
noga@math.tau.ac.il †
[2] Dept. of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel.
uria@math.tau.ac.il
[3] Dept. of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel.
azar@math.tau.ac.il ‡

Abstract. The problem of finding a large independent set in a hypergraph by an online algorithm is considered. We provide bounds for the best possible performance ratio of deterministic vs. randomized and non-preemptive vs. preemptive algorithms. Applying these results we prove bounds for the performance of online algorithms for routing problems via fixed paths over networks.

1 Introduction

The problem of finding the maximum independent set in a graph is a fundamental problem in Graph Theory and Theoretical Computer Science. It is well known that the problem is NP-hard ([18]), and that even the task of finding a rough approximation for the size of the maximum independent set is NP-hard ([4]). The intensive study of this problem includes the design and analysis of approximation algorithms ([10], [3]) and the investigation of online algorithms. The performance ratio of such an algorithm is the (worst case) ratio between the size of the maximum independent set, and the size (or expected size, when dealing with a randomized algorithm) of the independent set found by the algorithm.

In the online version of the maximum independent set problem the input graph is not known in advance, and the vertices arrive online. Here each vertex arrives with its incident edges towards previously presented vertices and the algorithm has to make an online decision if to add the vertex to the independent set. The adversary has the freedom to build the graph in any way he chooses.

The online algorithms can be deterministic or randomized. In addition, they can be non-preemptive or preemptive, where a preemptive algorithm may discard previously selected vertices (but may not pick a vertex that has already been discarded). This results in four basic online variants of the problem.

† Supported in part by the Israel Science Foundation, and by a USA-Israel BSF grant.
‡ Research supported in part by the Israel Science Foundation and by the US-Israel Binational Science Foundation (BSF).

Here we extend the study of the online independent set problem from the domain of graphs to that of hypergraphs. We consider the case of k-uniform hypergraphs, where the hypergraph is not known in advance, and vertices are presented along with their edges. The first part of the paper contains lower and upper bounds for the performance ratio (usually called the competitive ratio) of online algorithms for these problems.

Besides being interesting in their own rights, the results on the performance of online algorithms for the hypergraph maximum independent set problem have nice applications in obtaining lower bounds for the performance of online algorithms for routing over networks via fixed paths. These applications are obtained by an on-line reduction, a notion that differs from the usual reduction and works in the online setting.

In the online routing problems considered here a network graph is given in advance and the algorithm is presented with requests for calls over given paths in the network. We refer to the throughput version of the problem in which each call is accompanied by a required bandwidth, and must be either allocated this bandwidth, or rejected. The goal is to maximize the weighted number of calls (that is, the total bandwidth) accepted by the network. Routing received a lot of attention recently with various results. We explore the relation between the hypergraph independent set problem and the routing problem, obtaining lower bounds for the performance of online algorithms for both. This relation also captures randomized and preemptive algorithms.

1.1 Independent sets in graphs and hypergraphs

The offline version of the problem is defined as follows. Given a hypergraph $G = (V, E)$, find a maximum subset of V such that the vertex induced subgraph on it does not contain any edge.

In the *online* version of the problem, vertices are presented one by one along with edges which connect them to previously presented vertices. The online algorithm must decide for each vertex, as it is presented, whether to accept it or not. The accepted set must induce an independent set at all times. The goal is to maximize the size of the selected set.

We consider deterministic or randomized algorithms. Our discussion will allow both preemptive and non-preemptive algorithms. In the preemptive version of the problem, the algorithm may discard previously selected vertices. However, a vertex which has been discarded, at any time, can not be returned to the set.

Deterministic, non-preemptive algorithms for the online graph independent set problem have been considered before. A well known folklore result states that any deterministic algorithm has a competitive ratio $\Omega(n)$ when the graph is not known in advance. Here we provide tight bounds for the competitive ratio of deterministic non-preemptive, randomized non-preemptive and deterministic preemptive algorithms for graphs as well as for hypergraphs. We also obtain upper and lower bounds for the randomized preemptive case. Note that our upper bound for the randomized preemptive case for hypergraphs is obtained using a polynomial time algorithm, and its performance bound matches the

bound of the best known polynomial time approximation off-line algorithm that can be obtained using the methods of [9] (see also [17], [3]). To the best of our knowledge, this is the first online algorithm for the hypergraph independent set problem that achieves sub-linear competitive ratio. Note that by the result of [16] following [4], one cannot hope to obtain a much better bound even by an off-line polynomial time algorithm, unless NP have polynomial time randomized algorithms.

It is interesting to note that our polynomial time algorithm does not rely on the special properties of independent sets in uniform hypergraphs, but on the fact that being an independent set is a hereditary property. A property of subsets of a universe U is hereditary, if for every subset $A \subset U$ that satisfies it, every subset of A has the property as well. Hence, the same algorithm and upper bound hold for any hereditary property. In particular the upper bound holds for independent set in arbitrary hypergraphs, which are not necessarily uniform.

A related version of the online independent set problem deals with the model in which a graph (or a hypergraph) is known in advance to the algorithm, and a subset of vertices of it is presented by the adversary in an online manner. The goal here is also that of finding a large independent set, and the performance is measured by comparing it to the maximum size of an independent set in the induced subgraph on the presented vertices. It is quite easy to show that an $\Omega(n^\epsilon)$ lower bound holds for deterministic algorithms when the graph is known in advance where $\epsilon < 1$ is some fixed positive number. Bartal, Fiat and Leonardi [8] showed that an $\Omega(n^\epsilon)$ lower bound still holds when randomization and preemption are allowed.

1.2 Routing via fixed paths

Our results for the online hypergraph independent set problem can be applied in the study of the problem of (virtual circuit) routing over networks. Here the network graph is known in advance, and each edge has a known capacity. The algorithm is presented with requests for calls over the network with a certain required bandwidth. The algorithm either allocates this bandwidth on a path or rejects the call. The goal is to maximize the throughput (total bandwidth) of the accepted calls. Clearly, one may allow to use randomization and preemption. In the preemptive case accepted calls may be preempted, but preempted or rejected calls cannot be re-accepted. Obviously, calls preempted by an algorithm are not counted for the value of the throughput of this algorithm.

Two different versions for routing via fixed paths can be considered. In the first, the algorithm is presented with a request consisting of a source and a destination node, and must assign a route with the required bandwidth over the network to accept the call, while in the second version, each request includes a path to be routed over the network, and the algorithm may only decide to accept or reject the call.

There are numerous results for the virtual circuit routing problem for both versions (for surveys see [11, 15]). The competitive ratio of any deterministic (non-preemptive) algorithm has been shown to have an $\Omega(n)$ lower bound when

the bandwidth request could be as large as the capacity. On the other hand, an $O(\log n)$-competitive deterministic routing algorithm has been shown for general networks when all bandwidth requirements are bounded by the network capacity over $\log n$ [2].

A lot of research has been invested to overcome the small capacity requirements for special networks such as lines, tress, meshes [13, 14, 7, 5, 6, 12, 1]. However, the problem of deciding whether randomized or preemptive algorithms can achieve poly-logarithmic bound for large bandwidth requests over general networks remained open. A major step has been taken by [8] that showed an $\Omega(n^{\epsilon})$ lower bound for randomized preemptive online routing algorithms on general networks. Their lower bound holds for requests of maximal bandwidth, i.e. unit bandwidth for each request in a unit capacity network. The lower bound was proved by a reduction from the online maximum independent set problem in a known graph to the problem of routing calls over a network. The reduction does not extend for unit bandwidth and capacity k networks. In fact, it is still a major open problem to show a lower bound even for capacity 2.

Interestingly, we show a reduction between the independent set problem with an unknown graph and the fixed paths routing problem. Our reduction does extend for the case of capacity k. Specifically, we show a reduction from the independent set problem in a k uniform hypergraph to the fixed paths routing problem in a network of capacity $k-1$. This enables us to obtain lower bounds for the latter problem by using our lower bounds for the performance of online algorithms for the hypergraph independent set problem. The reduction holds also for randomized and preemptive algorithms.

Our result covers the gap between the known results for unit bandwidth and logarithmic bandwidth by giving a lower bound that approaches the known results as the bandwidth grows from 1 to $\log n$.

1.3 The presented results

We show the following,

- For the Independent Set problem in k-uniform hypergraphs with n vertices,
  - A $\Theta(\frac{n}{k})$ tight lower bound for the competitive ratio of deterministic, deterministic preemptive or randomized non-preemptive algorithms.
  - An $\Omega(\frac{n^{1/2}}{k})$ lower bound for the competitive ratio of randomized preemptive algorithms.
  - An $O(\frac{n}{\log n})$ upper bound for the competitive ratio of randomized preemptive algorithms.
- For the fixed paths routing problem over a network of N vertices with capacity $k-1$,
  - An $\Omega(\frac{N^{1/k}}{k})$ lower bound for the competitive ratio of deterministic, deterministic preemptive or randomized non-preemptive algorithm.
  - An $\Omega(\frac{N^{1/(2k)}}{k})$ lower bound for the competitive ratio of randomized preemptive algorithms.

2 Independent sets in k-uniform hypergraphs

As mentioned in the introduction, the algorithmic problem discussed in this section is the following. Given a k-uniform hypergraph $G = (V, E)$, with $V = \{v_1, v_2, \ldots, v_n\}$ and $E \subseteq 2^V$ ($\forall e \in E$, $|e| = k$), find a subset $V' \subseteq V$ of maximum cardinality such that for all $v_{i_1}, v_{i_2}, \ldots, v_{i_k} \in V'$: $(v_{i_1}, v_{i_2}, \ldots, v_{i_k}) \notin E$.

In the online version, the vertices are presented one by one, along with the edges which connect them to previously presented vertices.

2.1 A tight lower bound for online deterministic or randomized algorithms

Since G is a k-uniform hypergraph, any set of $k-1$ vertices forms an independent set. Therefore an upper bound of $\frac{n}{k-1} = O(\frac{n}{k})$ is trivially achievable. We now prove an $\Omega(\frac{n}{k})$ lower bound.

Theorem 1. *Any deterministic or randomized non-preemptive algorithm for the hypergraph independent set problem in a k-uniform hypergraph on n vertices has a competitive ratio $\Omega(\frac{n}{k})$.*

Proof. We use the online version of Yao's lemma by evaluating the performance of deterministic algorithms on a probability distribution on the inputs. Define the following probability distribution on the input sequences:

- Vertices are presented in pairs.
- One vertex of each pair will be selected randomly and marked as a "good" vertex, the other vertex will be marked as "bad".
- A set of k vertices containing a vertex from the current pair is an edge iff it contains at least one "bad" vertex from a previous pair.

Clearly, once the online algorithm picked one "bad" vertex, it can no longer pick more than $k-2$ additional vertices. Note that, crucially, the two vertices in each pair are indistinguishable when they are presented. Therefore, whenever the online algorithm picks a vertex, the probability it is "bad" is $\frac{1}{2}$, regardless of the history. The expected number of vertices the algorithm picked until the first "bad" vertex is picked, is 2. Hence the expected size of the independent set it finds is at most $2 + (k-2) = k$. The offline algorithm, on the other hand, can always pick all "good" vertices, yielding a competitive ratio of $\Omega(\frac{n}{k})$.

2.2 A tight lower bound for online deterministic preemptive algorithms

Theorem 2. *Any deterministic preemptive algorithm for the hypergraph independent set problem for k-uniform hypergraphs on n vertices has a competitive ratio $\Omega(\frac{n}{k})$.*

Proof. We define the following input sequence:

- Vertices are presented in steps. In each step there are $2k-2$ vertices such that any subset of k of them is an edge.
- At most $k-1$ vertices from each step will be selected as "bad" vertices, all the other vertices will be marked as "good".
- A set of k vertices that contains vertices from the current step and previous steps is an edge if it contains at least one "bad" vertex from a previous step.

The deterministic algorithm may choose at most $k-1$ vertices from each step. The adversary will mark them as "bad" and all other vertices (at least $k-1$) as "good". Therefore all the vertices which may be selected by the online algorithm are "bad", and may be replaced, by preemption, only by other "bad" vertices. By the construction of the sequence the online algorithm may hold a maximum of $k-1$ vertices at any time without having an edge (at most $k-1$ from one step or at most $k-1$ from several steps). However, the optimal algorithm will collect all "good" vertices, thus creating an independent set of at least $\frac{n}{2}$ vertices.

2.3 A lower bound for online randomized preemptive algorithms

We prove a lower bound of $\Omega(\frac{\sqrt{n}}{k})$ for the competitive ratio of any randomized preemptive on-line algorithm. We make use of Yao's lemma to establish a lower bound for any deterministic algorithm on a given probability distribution, thus yielding a lower bound for the randomized case.

Theorem 3. *Any randomized preemptive on-line algorithm for the online independent set problem for k-uniform hypergraphs on n vertices has competitive ratio $\Omega(\frac{\sqrt{n}}{k})$.*

Proof. Define the following probability distribution on the input sequences. Each sequence will be constructed of vertices, presented in steps. Each step consists of l vertices, with a total of n vertices in all steps. Each step will be generated according to the following distribution:

- At step j, l vertices are presented such that any subset of k of them is an edge.
- One vertex chosen uniformly at random will be marked as a "good" vertex, while all others will be marked as "bad".
- A set of k vertices that contains vertices from the current step and previous steps is an edge iff it contains at least one "bad" vertex from a previous step.

For the proof, we reveal at the end of each step, which is the "good" vertex, thus giving the algorithm the opportunity to immediately discard all "bad" vertices, at the beginning of the next step. Note that all the vertices in each step look indistinguishable given all the history since they participate in exactly the same edges. Thus, there is no way for the algorithm to distinguish between the "good" and the "bad" vertices before the step ends. Therefore, at the end of each step, the algorithm may hold any number of "good" vertices from previous steps, plus a set of at most $k-1$ additional vertices. Some of these additional

vertices may be "bad" vertices from previous steps, and some may belong to the current step. The probability of the algorithm to select each "good" vertex in a step is at most $\frac{k-1}{l}$, regardless of previous selections. The expected benefit of the algorithm is thus:

$$E(\mathrm{ON}) \leq \frac{n}{l} \cdot \frac{k-1}{l} + k - 1 \leq \frac{nk}{l^2} + k$$

On the other hand, the optimum algorithm OPT may pick all the "good" vertices, giving a benefit of at least $\frac{n}{l}$. Choosing, optimally, $l = \sqrt{n}$ we get a competitive ratio of $\Omega(\frac{\sqrt{n}}{k})$.

2.4 A sublinear upper bound

Here we present a randomized, preemptive algorithm for the independent set problem in an arbitrary (not necessarily uniform) hypergraph and show that its competitive ratio is $O(n/\log n)$. The algorithm also runs in polynomial time.

Given an input sequence of n vertices, the algorithm divides the sequence into groups of y vertices each. Each of these groups will be called a phase. At the beginning of each phase we uniformly select at random β distinct vertices of that phase. During the phase we pick all selected vertices, as long as they induce an independent set. If they do not induce an independent set, then the phase fails, and we drop all the vertices but one. If the phase succeeds we stop. Otherwise, we start another phase and replace the one remaining vertex with the first selected vertex of the next phase. We assume that the portion of the maximal independent set size, x, is known in advance (i.e. the set contains n/x vertices). Later we use an appropriate weighted version of classify and randomly select to relieve this restriction.

Claim. For each $4 \leq x \leq \log n$ define $\beta = \frac{\log n}{4 \log x}$ and $y = 4\beta x$. Then our algorithm picks an independent set of size β with high probability, or there is no independent set of size $\frac{n}{x}$ in the graph.

Proof. We assume that there is an independent set in the graph, consisting of at least $\frac{n}{x}$ vertices. We distinguish between phases with a lot of vertices from the set, and those with few vertices from the independent set. Phases with more than $\frac{y}{2x}$ vertices are good phases. There are at least $\frac{n}{2xy}$ good phases, otherwise the total number of vertices in the independent set is less than $\frac{n}{2xy} \cdot y + \frac{n}{y} \cdot \frac{y}{2x} = \frac{n}{x}$ in contradiction to the size of the independent set. From each good set we select β vertices at random. Since $y = 4\beta x$, each of these vertices has a conditional probability greater than $\frac{y/2x - \beta}{y} = \frac{1}{4x}$ of being a vertex from the independent set, given that all the previously selected vertices are from the independent set. Therefore the probability of failure is less than $1 - (\frac{1}{4x})^\beta$, for each good phase. Since we have $\frac{n}{2xy}$ good phases, the total probability of failure is bounded by $\left(1 - (\frac{1}{4x})^\beta\right)^{\frac{n}{2xy}}$. As $\beta = \frac{\log n}{4 \log x}$ and $y = 4\beta x$, we get that the probability of failure

is less than

$$\left(1-(\frac{1}{4x})^{\beta}\right)^{\frac{n}{2xy}} \leq \left(1-\frac{1}{n^{1/2}}\right)^{\frac{n}{2xy}} \leq e^{\frac{-n^{1/2}}{2xy}} \leq e^{-n^{0.49}}$$

Theorem 4. *There exists a randomized preemptive algorithm which achieves a competitive ratio of $O(\frac{n}{\log n})$ for the independent set problem in an arbitrary hypergraph on n vertices.*

Proof. We use classify and randomly select. Divide the range of x from 4 to $\log n$ into classes by powers of two, and assign a probability to each such class. For the class of $2^{i-1} \leq x < 2^i$ we assign probability proportional to $\frac{i}{2^i}$.
Using the above algorithm for the chosen class we get an algorithm for which:

- If $\mathrm{OPT}(\sigma) \leq \frac{n}{\log n}$, then $\mathrm{ON}(\sigma) \geq 1$ and the competitive ratio is at most $\frac{n}{\log n}$.
- If $\mathrm{OPT}(\sigma) = \frac{n}{x} > \frac{n}{\log n}$, then $E(\mathrm{ON}(\sigma)) \geq \Omega(\frac{\log x}{x}) \cdot \Theta(\frac{\log n}{\log x}) = \Omega(\frac{\log n}{x})$, and again the competitive ratio is at most $O(\frac{n}{\log n})$.

Note that if the length of the sequence is unknown we may use a technique similar to the standard doubling techniques by selecting an initial value for n and then squaring its value if the sequence turns out to be too long. Squaring ensures that only a small portion of the independent set will be processed using a wrong value of n, while having the wrong value for n (by at most a power of 2) will only add a constant factor to the competitive ratio. To avoid the sequence from ending just as we update the value of n, we use a simple boundary smoothing technique, such as randomly selecting a multiplicative factor between 1 and 2, and multiplying the updated value by this factor.

3 Routing via fixed paths in constant capacity networks

We next show a reduction from the independent set problem for k-uniform hypergraphs to routing with fixed paths over a $k-1$ bandwidth network. The reduction step translates vertices into paths over the given graph, while making sure that any hyperedge results in an inconsistent set of calls. The reduction yields lower bounds for the routing problem.

Note that while the hypergraph was unknown to the algorithm in the independent set problem, the network structure is known in advance in the case of routing. The process of adding a new (unknown) vertex of the hypergraph while revealing the edges which connect it to previously presented vertices, corresponds to the process of presenting a new path, to be allocated over the network.

A vertex v is called the *completing vertex* of an edge e, if all the other vertices of e were presented before v, and thus the edge e was revealed at the appearance of v.

3.1 The reduction step

Let $G = (V, E)$ be a k-uniform hypergraph, with $V = \{c_1, c_2, \ldots, c_n\}$, and assume the vertices are presented in this order. We construct a graph $G' = (V', E')$, where each edge has capacity $k-1$ and a set of paths $P = \{p_1, p_2, \ldots, p_n\}$, such that

1. Each vertex $c_i \in V$ corresponds to the path p_i.
2. For every set of paths $p_{i_1}, p_{i_2}, \ldots, p_{i_k}$, there exists an edge $e \in E'$ such that $e \in p_{i_j}, \forall j$ if and only if $(c_{i_1}, c_{i_2}, \ldots, c_{i_k}) \in E$.

Note that the reduction we present is an on-line reduction, and not a standard reduction. In the on-line reduction the network, and the paths are built as the algorithm advances, without knowing what are the actions of the algorithm, and how the sequence will continue, while in a regular "offline" reduction the whole sequence in known in advance. Moreover, in a standard reduction, any input sequence might result in a completely different image, even if the sequences have a common prefix. In an on-line reduction, on the other hand, we must allow any prefix to be completed in every possible way without restricting it by the reduction process itself.

G' consists of $2n + 2\binom{n}{k}$ vertices, and $2n\binom{n}{k} + \binom{n}{k}^2$ edges. First we construct a graph with $\binom{n}{k}$ independent edges. Each edge has a unique label $(i_1, i_2, \ldots, i_k)$ where $i_j < i_{j+1}\, \forall j$, i_k is called the last coordinate in the edge label. We refer to these edges as *restricting* edges. We assign a left and a right vertex to each restricting edge. The right end of each edge is connected to the left end of every other restricting edge, we refer to these edges as *connecting* edges. We then add two sets of vertices s_i and t_i $i = 1 \ldots n$ with s_i connected to all left ends, and t_i connected to all the right ends (see Figure 1).

For each vertex c_i we assign the following path p_i; starting from s_i, we pass through all edges containing i in their label not as their last coordinate, and through all the edges labeled $(i_1, i_2, \ldots, i)$ with i as the last coordinate of the label if and only if $(c_{i_1}, c_{i_2}, \ldots, c_i) \in E$. Note that c_i is the completing vertex of the edge $(c_{i_1}, c_{i_2}, \ldots, c_i)$. Finally we connect the last edge to the vertex t_i. Starting from s_i, the path will enter each edge through its left vertex, and leave through its right vertex.

To complete the reduction we prove the following lemma,

Lemma 1. *In the resulting graph, for every set of paths $p_{i_1}, p_{i_2}, \ldots, p_{i_k}$, there exists an edge $e \in E'$ such that $e \in p_{i_j}, \forall j$ if and only if $(c_{i_1}, c_{i_2}, \ldots, c_{i_k}) \in E$.*

Proof. We first show that each connecting edge is used by at most $k-1$ different paths. Each path passing through a connecting edge must pass through the restricting edges connected to it. A path p_i uses a certain restricting edge only if its index is one of the coordinates in the label of that edge, but two restricting edges may share at most $k-1$ coordinates. Thus no connecting edge is used more than $k-1$ times.

Therefore, we limit our attention to restricting edges. Consider the paths going through the restricting edge $e \in E'$ whose label is $(i_1, i_2, \ldots, i_k)$. All the

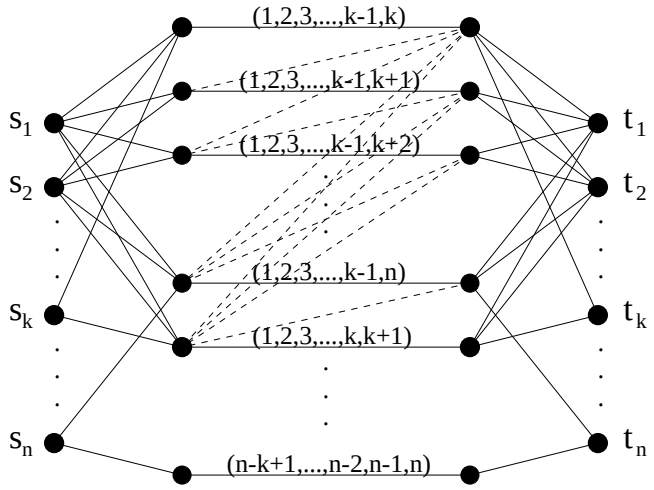

Fig. 1. The structure of G'. Labeled edges are the restricting edges, dashed lines represent connecting edges.

paths $p_{i_1}, p_{i_2}, \ldots, p_{i_{k-1}}$ pass through the edge, as their index is not the last coordinate in the edge label. Therefore, all edges are used to their maximum capacity by the paths corresponding to the first $k-1$ coordinates in their label. According to our construction, the last path p_{i_k} goes through this edge if and only if $(c_{i_1}, c_{i_2}, \ldots, c_{i_k}) \in E$, thus creating an inconsistency.

3.2 The resulting lower bounds

Theorem 5. *The following lower bounds hold for the online routing problem with fixed paths, over a network with N vertices and constant capacity $k-1$.*

- *Any deterministic, deterministic preemptive or randomized non-preemptive on-line algorithm has competitive ratio $\Omega(\frac{N^{1/k}}{k})$.*
- *Any randomized preemptive on-line algorithm has competitive ratio $\Omega(\frac{N^{1/(2k)}}{k})$.*

Proof. By the above lemma, any online algorithm for the fixed paths routing problem over a network with capacity $k-1$, is also an algorithm for the independent set problem over a k-uniform hypergraph. Each path selected matches a vertex in the hypergraph, and vice-versa. Moreover, any independent set in the hypergraph defines a set of consistent paths in the network, and any set of consistent paths defines an independent set. Therefore, any algorithm(online or offline) which achieves a value of $A(\sigma)$ for the network routing problem, may be used to build an independent set of the same size in the hypergraph. Thus, any lower bound on the competitive ratio for the independent set problem for a

k-uniform hypergraph with n vertices, is also a lower bound for the competitive ratio for the routing problem on a $k-1$ capacity network with $N = \Theta(n^k)$ vertices.

Using the lower bounds found for the k-uniform hypergraph problem, we get the following set of lower bounds:

- The competitive ratio of any deterministic, deterministic preemptive or randomized non-preemptive algorithm is $\Omega(\frac{n}{k}) = \Omega(\frac{N^{1/k}}{k})$.
- The competitive ratio of any randomized preemptive on-line algorithm, is $\Omega(\frac{n^{1/2}}{k}) = \Omega(\frac{N^{1/(2k)}}{k})$.

Note that the lower bound becomes smaller than $\log N$ for $k = \Theta(\log N)$. This conforms with the $O(\log N)$ competitive algorithm of [2] for $k = \Theta(\log n)$.

References

[1] R. Adler and Y. Azar, *Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms*, Proc. 10th ACM-SIAM Symp. on Discrete Algorithms, 1999, pp. 1–10.

[2] B. Awerbuch, Y. Azar, and S. Plotkin, *Throughput-competitive online routing*, 34th IEEE Symposium on Foundations of Computer Science, 1993, pp. 32–40.

[3] N. Alon and N. Kahale, *Approximating the independence number via the θ-function*, Math. Programming 80 (1998), 253-264.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and intractability of approximation problems*, Proc. of the 33rd IEEE FOCS, IEEE (1992), pages 14–23.

[5] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén, *Competitive non-preemptive call control*, Proc. of 5th ACM-SIAM Symposium on Discrete Algorithms, 1994, pp. 312–320.

[6] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani, *On-line admission control and circuit routing for high performance computation and communication*, Proc. 35th IEEE Symp. on Found. of Comp. Science, 1994, pp. 412–423.

[7] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, and B. Schieber, *Bandwidth allocation with preemption*, Proc. 27th ACM Symp. on Theory of Computing, 1995, pp. 616–625.

[8] Y. Bartal, A. Fiat, and S. Leonardi, *Lower bounds for on-line graph problems with application to on-line circuit and optical routing*, Proc. 28th ACM Symp. on Theory of Computing, 1996, pp. 531–540.

[9] B. Berger and J. Rompel, *A better performance guarantee for approximate graph coloring*, Algorithmica 5(1990),459–466.

[10] R. Boppana and M. M. Halldorsson, *Approximating maximum independent sets by excluding subgraphs*, BIT 32 (1992), 180–196.

[11] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, 1998.

[12] R. Canetti and S. Irani, *Bounding the power of preemption in randomized scheduling*, Proc. 27th ACM Symp. on Theory of Computing, 1995, pp. 606–615.

[13] J.A. Garay and I.S. Gopal, *Call preemption in communication networks*, Proceedings of INFOCOM '92 (Florence, Italy), vol. 44, 1992, pp. 1043–1050.

[14] J. Garay, I. Gopal, S. Kutten, Y. Mansour, and M. Yung, *Efficient on-line call control algorithms*, Journal of Algorithms **23** (1997), 180–194.

[15] S. Leonardi, *On-line network routing*, Online Algorithms - The State of the Art (A. Fiat and G. Woeginger, eds.), Springer, 1998, pp. 242–267.

[16] J. Håstad, *Clique is hard to approximate within* $n^{1-\epsilon}$, Proc. 37^{th} IEEE FOCS, IEEE (1996), 627 – 636.

[17] T. Hofmeister and H. Lefmann, *Approximating maximum independent sets in uniform hypergraphs*, Proc. of the 23^{rd} International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 1450, Springer Verlag (1998), 562-570.

[18] R. Karp, *Reducibility among combinatorial problems*, Plenum Press, New York, 1972, Miller and Thatcher (eds).