Online algorithms for combinatorial problems

Abstract of the Ph.D. Thesis

by

Judit Nagy-György

Supervisor: Péter Hajnal
Associate Professor

Doctoral School in Mathematics and Computer Science
University of Szeged
Bolyai Institute

2009

1 Introduction: competitive analysis

Online algorithms have been investigated for approximately 30 years. One of the main methods to measure the performance of online algorithms is a worst case analysis called competitive analysis. In an online problem the parts of the input sequence (input set equipped with an ordering) appear one by one and the *online algorithm* must produce a sequence of decisions about these parts that will have an impact on the final quality of its overall performance. Each of these decisions must be made based on the already appeared part of the input sequence without any information about the future.

Associated with every input I is a set of feasible outputs, and associated with each feasible output is a positive real representing the cost of the output with respect to I, among which the minimal is denoted by $\text{opt}(I)$. Given any legal input I an algorithm $\mathcal{A}$ computes a feasible output. The cost associated with this output is denoted by $\mathcal{A}(I)$.

An online algorithm $\mathcal{A}$ is called *(strictly) c-competitive* for some $c > 0$ if for all finite input sequences I

$$\mathcal{A}(I) \leq c \cdot \text{opt}(I).$$

The competitive ratio of $\mathcal{A}$ is the smallest c such that $\mathcal{A}$ is c-competitive. An algorithm $\mathcal{A}$ is *weakly c-competitive* if there is a constant a such that for all finite input sequence I,

$$\mathcal{A}(I) \leq c \cdot \text{opt}(I) + a.$$

The weak competitive ratio of $\mathcal{A}$ is the smallest c such that $\mathcal{A}$ is c-competitive.

We call a randomized online algorithm *c-competitive* if there exists a constant a such that for any input sequence I

$$\text{E}[\mathcal{A}(I)] \leq c \cdot \text{opt}(I) + a.$$

The competitive ratio of $\mathcal{A}$ is the smallest c for which $\mathcal{A}$ is c-competitive.

2 Scheduling with machine cost and rejection

The area of scheduling theory has large literature and several models (see in [18]). In one of the most fundamentals and simplests we have a fixed number of machines and the jobs arrive from a list (list model). Each job has a *processing time*. In the "parallel machines case" m machines are given. To schedule a job we have to assign it to a machine. We have to schedule each job and no machine may simultaneously run two jobs. By the load of a machine we mean the sum of processing times of all jobs assigned to it and the *makespan* is the maximum of loads. The cost is the makespan, therefore our goal is to minimize it.

In the online version of the problem the jobs and their processing times are revealed one by one. When a job is revealed the online algorithm has to assign to a machine without any information about the further jobs. Algorithm $\mathcal{LIST}$ is the first algorithm in this model has been developed by Graham [7] in 1966. When the jth job is revealed Algorithm $\mathcal{LIST}$ assigns it to the machine where the actual load is minimal. Graham [7] proved that the competitive ratio of Algorithm $\mathcal{LIST}$ is $2 - 1/m$.

The problem of scheduling with machine cost is defined in [8]. In this model the number of machines is not a given parameter of the problem: the algorithm has to purchase the machines, and the goal is to minimize the cost spent for purchasing the machines plus the makespan. In [8] the problem where each machine has cost 1 is investigated. The jobs arrive one by one and the decision maker has to decide in each step whether to buy new machines and then schedule the job on one of the already purchased machines without any information about the further jobs. Imreh and Noga [8] presented the Algorithm $\mathcal{A}_\rho$, where $\rho = (0 = \rho_1, \rho_2, \dots, \rho_i, \dots)$ is an increasing sequence. They proved that the competitive ratio of Algorithm $\mathcal{A}_\rho$ is $\varphi = (1 + \sqrt{5})/2$ for $\rho = (0, 4, 9, 16, \dots, i^2, \dots)$.

The problem of scheduling with rejection is defined in [3]. In this model, it is possible to reject the jobs. The jobs are characterized by a *processing time* and a *penalty*. In this problem the cost is the sum of the makespan and the penalties of all rejected jobs. Bartal et al. [3] developed algorithm Reject-Total-Penalty(α). It has a parameter α which plays the role of a threshold. The authors proved that the Algorithm $\mathcal{RTP}(\varphi - 1)$ is $(1 + \varphi)$-competitive.

The following results of this section can be found in [14].

We considered a more general model MCR combining the above two approaches. Here the machines are not given to the algorithm in advance but the algorithm must purchase them, and the jobs can be rejected. The cost is the makespan plus the cost of purchasing the machines plus the sum of the penalties of the rejected jobs so our goal is to minimize it. We suppose that each machine has cost 1. We call the total cost of purchasing the machines *machine purchasing cost*.

In the problem the jth job has a processing time p_j and a penalty which is the cost of rejecting it, denoted by w_j. $P_H = \sum_{j \in H} p_j$ and $W_H = \sum_{j \in H} w_j$.

Since we have rules for purchasing the machines and for the rejection and scheduling of the jobs it is a straightforward idea to combine these rules and build algorithms for the complex problem. In the first part we showed the surprising result that the simple combinations of these rules are not constant competitive.

We presented a more sophisticated algorithm. The basic idea is that instead of the original problem we consider a relaxed version, where we replace part of the cost of the schedule (purchasing cost of machines plus the makespan) with a lower bound of it.

Suppose that we accepted a set of jobs, denote the set of their indices by A, furthermore m machines were purchased, and the current makespan is M. Then for the cost of the schedule $M + m \geq M + P_A/M$ is valid. Let l_A denote the greatest processing time that belongs to a job with index in A. We defined the following expression:

$$M_A := \begin{cases} \max\{\sqrt{P_A}, l_A\}, & \text{if } P_A > 1 \\ 1 & \text{otherwise} \end{cases}$$

Now for an arbitrary set A of indices let

$$T_A := \begin{cases} M_A + \dfrac{P_A}{M_A} & \text{if } A \neq \emptyset \\ 0 & \text{if } A = \emptyset \end{cases}$$

We defined the *relaxed problem*: jobs arrive, each job has a processing time and a penalty. We have to find a solution where the total penalty paid for the rejected jobs plus the value T_A for the set A of indices of accepted jobs is minimal. For a set J of indices of jobs the cost of the optimal solution of the relaxed problem is denoted by $\mathrm{ropt}(J)$. The following statement holds.

Corollary 13 *For an arbitrary set J of indices of jobs,* $\mathrm{ropt}(J) \leq \mathrm{opt}(J)$.

To develop algorithm Optcopy we have to examine the structure of the optimal solutions of the relaxed problem. For a set of indices J denote J_k the set of the first k indices of J. Then the following statement is valid.

Lemma 14 *Suppose that A^*_{k-1} is the set which belongs to an optimal solution of the relaxed problem on set J_{k-1}. Then the relaxed problem on set J_k has an optimal solution such that A^*_{k-1} is a subset of the set of the indices of the accepted jobs.*

The relaxed problem can be solved in polynomial time. The algorithm which solves the problem is based on the following structural property.

Lemma 15 *For the jth job we consider the problem* $\mathrm{REL}(j)$ *which is the restricted relaxed problem where it is given that j is the index of the largest accepted job. Order the set of jobs whose indices are not larger than j by the value p_i/w_i into an increasing sequence. Then* $\mathrm{REL}(j)$ *has an optimal solution which is a prefix of this sequence.*

By Lemma 15 we have a polynomial time algorithm giving optimal solutions of the relaxed problems which satisfy Lemma 14. We call this algorithm *Relopt*. Denote the sets of the indices of the accepted jobs from J_k by A^*_k and the set of the indices of rejected jobs by R^*_k. Therefore $A^*_i \subseteq A^*_k$ if $i \leq k$. Then the following statement holds.

Lemma 16 *For the above defined sets, the following inequality is valid:*

$$\sum_{j=1}^{n} W_{R^*_{j-1} \cap A^*_j} \;\leq\; T_{A^*_n}.$$

We defined the class $\mathrm{Optcopy}_\rho$ of algorithms.

Algorithm $\mathcal{OC}_\rho$

At the arrival of a new job perform the following steps.

(i) If it is rejected by Relopt, reject it, otherwise go to step (ii)

(ii) Schedule the job by Algorithm $\mathcal{A}_\rho$, where in the machine purchasing rule only the accepted jobs are taken into account.

Theorem 17 *Algorithm $\mathcal{OC}_\rho$ with the sequence $\rho = (0, 4, 9, 16, \ldots, i^2, \ldots)$ is $(\varphi + 1)$-competitive.*

3 Coloring graphs and hypergraphs

The notion of online graph appeared in [11], followed by that of online hypergraph which is the generalization of online graph defined first in [1]. An *online hypergraph* is a structure $H^{\prec} = (H, \prec)$ where $H = (V, E)$ is a hypergraph and $\prec$ is a linear order of its vertices. Let H_i denote the online hypergraph induced by the $\prec$-first i elements V_i of V. An online hypergraph coloring algorithm colors the i-th vertex of the hypergraph by only looking at the subhypergraph H_i. For an online algorithm $\mathcal{A}$ and an online hypergraph $H^{\prec}$, the cost is the number of colors used by $\mathcal{A}$ to color $H^{\prec}$ which is denoted by $\chi_{\mathcal{A}}(H^{\prec})$. Clearly, $\text{opt}(H^{\prec}) = \chi(H)$. For a hypergraph H, $\chi_{\mathcal{A}}(H)$ denotes the maximum of the $\chi_{\mathcal{A}}(H^{\prec})$ values over all orderings $\prec$. Online graph coloring has been investigated in several papers, one can find many details on the problem in the survey paper [9].

3.1 Results on graphs with forbidden subgraphs

The results of this subsection can be found in [13].

The girth of a graph G denoted $g(G)$ is the length of its shortest cycle and the oddgirth of G denoted $g_o(G)$ is the length of its shortest odd cycle. The distance $\text{dist}(u, v)$ of vertices u and v is the length of the shortest uv path. For a positive integer d, let $N_d(v)$ be the set of vertices with positive distance at most d from vertex v. $N_{d,\text{odd}}(v)$ is the set of vertices with a positive *odd* distance at most d from vertex v. For any $S \subset V(H)$ let us define $N_d(S) = \bigcup_{v \in S} N_d(v) \setminus S$ and $N_{d,\text{odd}}(S) = \bigcup_{v \in S} N_{d,\text{odd}}(v) \setminus S$. Let $N_d^{\prec}(v)$ the set of vertices preceding v with a positive distance at most d from vertex v. $N_{d,\text{odd}}^{\prec}(v)$ is the set of vertices preceding v with a positive odd distance at most d from vertex v.

In [10] Algorithm $\mathcal{B}_n$ is presented which uses less than $2n^{1/2}$ colors to color any online graph on n vertices that induces neither C_3 nor C_5. We generalized Algorithm $\mathcal{B}_n$ for graphs with high girth.

Algorithm $\mathcal{B}_{n,d}$

Consider the input sequence $v_1 \prec \ldots \prec v_n$ of an online graph $G^{\prec}$ with $g(G) > 4d+1$. Initialize by setting $U_i = \emptyset$ for all $i > dn^{1/(d+1)}$.

s-th stage.

(i) If there exists $i \in [dn^{1/(d+1)}]$ such that v_s is no adjacent to any vertex colored i then color v_s by the least such i.

(ii) Otherwise, if there exists $i > dn^{1/(d+1)}$ such that $v_s \in N_d(U_i)$ then then color v_s by the least such i.

(iii) Otherwise, let j be the least integer $i > dn^{1/(d+1)}$ with $U_i = \emptyset$. Set $U_j = N_d^{\prec}(v_s)$ and color v_s with j.

Theorem 21 *Algorithm $\mathcal{B}_{n,d}$ produces a coloring of any graph $G^{\prec}$ on n vertices with girth $g > 4d + 1$ with less than $(d + 1)n^{1/(d+1)}$ colors.*

The second generalization is $\mathcal{BO}_{n,d}$, which uses Algorithm $\mathcal{AA}$ defined by Lovász which colors bipartite graphs on n' vertices with at most $\log_2 n'$ colors (see [9]) as an auxiliary algorithm.

Algorithm $\mathcal{BO}_{n,d}$

Consider the input sequence $v_1 \prec \ldots \prec v_n$ of online graph $G^{\prec}$ with $g_o(G) > 4d+1$. Set $r = (n/(2d\log_2 2d))^{1/2}$. Initialize by setting $S_i = \emptyset$ for all $i \in [r]$ and $U_i = \emptyset$ for all $i > r$. At the s-th stage the algorithm processes the vertex v_s as follows.

(i) If there exists $i \in [r]$ such that the subgraph induced by $S_i \cup \{v_s\}$ is 2-colorable and Algorithm $\mathcal{AA}$ uses at most $2\log_2 2d$ colors to color $S_i \cup \{v_s\}$, then let j be the least such i. Set $S_j = S_j \cup \{v_s\}$ and color v_s (in the subgraph induced by S_j) by Algorithm $\mathcal{AA}$ using colors $2(j-1)\log_2 2d + 1, \ldots, 2j\log_2 2d$.

(ii) Otherwise, if there exists $i > r$ such that $v_s \in N_{d,\mathrm{odd}}(U_i)$ then color v_s with the least such i.

(iii) Otherwise, let j be the least integer $i > r$ such that $U_i = \emptyset$. Set $U_j = N^{\prec}_{d,\mathrm{odd}}(v_s) \cap (\bigcup_{\ell=1}^{r} S_\ell)$ and color v_s with j.

Theorem 23 *Algorithm $\mathcal{BO}_{n,d}$ produces a coloring of any graph $G^{\prec}$ on n vertices having oddgirth greater than $4d+1$ with at most $2(2n\log_2 2d/d)^{1/2}$ colors.*

3.2 Results on hypergraphs

The results of this subsection can be found in [15].

In this part we considered the case of 2-colorable k-uniform hypergraphs with $k \geq 3$. A straightforward online hypergraph coloring algorithm is First Fit ($\mathcal{FF}$). When a vertex arrived $\mathcal{FF}$ assigns to it the least color which does not make a monochromatic edge. We proved the following results.

Theorem 24 *For every online hypergraph coloring algorithm $\mathcal{A}$ there exists a 2-colorable k-uniform hypergraph H on n vertices with $\chi_{\mathcal{A}}(H) \geq \lceil n/(k-1) \rceil$. If H is a k-uniform hypergraph then $\chi_{\mathcal{FF}}(H) \leq \lceil n/(k-1) \rceil$.*

The theorem shows that the competitive ratio of $\mathcal{FF}$ is $\lceil n/(k-1) \rceil/2$ on this class and no better algorithm can be defined for this class. Moreover this theorem also proves (with $k=3$) that contrary to the case of the online graph coloring in the case of hypergraphs no online algorithm with sublinear competitive ratio exists.

We considered the case of 2-colorable hypergraphs with maximal degree k. It is easy to see that any hypergraph with maximal degree k is $k+1$ colorable, since $\mathcal{FF}$ colors them with at most $k+1$ colors.

Theorem 27 *For every online hypergraph coloring algorithm $\mathcal{A}$ and integer $d > 2$ there exists a 2-colorable d-uniform hypergraph H on at most $\frac{(d-1)^k - 1}{d-2}$ vertices with maximal degree k such that $\chi_{\mathcal{A}}(H) \geq k+1$.*

Considering the class of hypergraphs with bounded matching number $\mathcal{FF}$ can achieve the following performance.

Theorem 28 *For any hypergraph H algorithm $\mathcal{FF}$ gives a coloring of H with at most $2 \cdot \nu(H) + 1$ colors.*

Since any two edges of the finite projective planes are intersecting (the matching number is 1) we also obtained that $\mathcal{FF}$ colors the finite projective planes with at most 3 colors. On the other hand, as the following statement shows there exists no online algorithm which can use less colors than $\mathcal{FF}$ in this cases.

Theorem 30 *No online algorithm exists which can color a finite projective plane with less than 3 colors.*

4 The k-server problem

The k-server problem can be formulated as follows. Given a metric space with k mobile servers that occupy distinct points of the space and a sequence of requests (points), each of the requests has to be served, by moving a server from its current position to the requested point. The goal is to minimize the total cost, that is the sum of the distances covered by the k servers; the optimal cost for a given sequence ϱ is denoted $\mathrm{opt}(k, \varrho)$. A k-server algorithm is online if it serves each request immediately when it arrives (without any prior knowledge about the future requests).

4.1 A randomized algorithm on decomposable spaces

The results of this subsection can be found in [12]. These results modify the approach of [4] and [5].

We consider a restriction of the problem, namely we seek for an efficient randomized online algorithm for metric spaces that are "μ-HST spaces" [2] and defined as follows:

Definition 32 *For $\mu \geq 1$, a μ-hierarchically well-separated tree (μ-HST) is a metric space defined on the leaves of a rooted tree T. To each vertex $u \in T$ there is associated a label $\Lambda(u) \geq 0$ such that $\Lambda(u) = 0$ if and only if u is a leaf of T. The labels are such that if a vertex u is a child of a vertex v then $\Lambda(u) \leq \Lambda(v)/\mu$. The distance between two leaves $x, y \in T$ is defined as $\Lambda(\mathrm{lca}(x, y))$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in T.*

In [17], μ-decomposable spaces have been introduced. We considered a special case.

Definition 33 *Let $\mathcal{M}$ be a metric space. We call $\mathcal{M}$* uniformly μ-decomposable *for some $\mu > 1$ if its points can be partitioned into $t \geq 2$* block*s, $B_1, \ldots, B_t$ such that the following conditions both hold:*

1. *whenever $x, y \in \mathcal{M}$ are belonging to different blocks, their distance is exactly Δ, the diameter of $\mathcal{M}$;*

2. *the diameter of each B_i is at most Δ/μ.*

For a given request sequence ϱ we denote its ith member by ϱ_i, and the prefix of ϱ of length i by $\varrho_{\leq i}$.

The set of points where the servers are staying at a given time is a *configuration*. Given a block B_s, a request sequence ϱ and an initial configuration C in B_s, let $\mathcal{A}_s(C, \varrho)$ denote the cost computed by the algorithm $\mathcal{A}$ for the subsequence of ϱ consisting of the requests arriving to B_s. For any number ℓ of servers, let $\mathcal{A}_s(\ell, \varrho)$ stand for $\max_{|C|=\ell} \mathcal{A}_s(C, \varrho)$, where C runs over all the initial configurations in B_s consisting of ℓ servers. Also, let $\mathrm{opt}_s(C, \varrho)$ denote the optimal cost for the subsequence of ϱ consisting of the requests arriving to B_s, starting from configuration C and let $\mathrm{opt}_s(\ell, \varrho) = \min_{|C|=\ell} \mathrm{opt}_s(C, \varrho)$. Thus, if ϱ is nonempty, $\mathrm{opt}_s(0, \varrho)$ is defined to be infinite.

Our algorithm is based on the following notion.

Definition 34 *The* demand *of the block B_s for the request sequence ϱ is*

$$D_s(\varrho) := \min\{\ell \,|\, \mathrm{opt}_s(\ell, \varrho) + \ell\Delta = \min_j\{\mathrm{opt}_s(j, \varrho) + j\Delta\}\},$$

if ϱ is nonempty, otherwise it is 0.

We also introduced a technical notion.

Definition 36 *Suppose $\mathcal{N}$ is a metric space, $\mathcal{A}$ is a randomized online algorithm, f is a real function and $\mu > 0$ is a real number satisfying the following conditions:*

1. *$f(\ell)/\log \ell$ is monotone non-decreasing;*
2. *for any $0 < \ell \leq \mu$ and request sequence ϱ in $\mathcal{N}$,*

$$\mathrm{E}[\mathcal{A}(\ell, \varrho)] \;\leq\; f(\ell) \cdot \mathrm{opt}(\ell, \varrho) + \frac{f(\ell) \cdot \ell \cdot \mathrm{diam}(\mathcal{N})}{\log \ell}.$$

Then we call $\mathcal{A}$ an (f, μ)-efficient algorithm on $\mathcal{N}$.

We proved the following theorem:

Theorem 37 *Suppose $\mathcal{M}$ is a uniformly μ-decomposable space and $\mathcal{A}$ is an (f, μ)-efficient algorithm on each block of $\mathcal{M}$. Then there exists an (f', μ)-efficient algorithm on $\mathcal{M}$, where $f'(x)$ is defined as $c \cdot f(x) \log x$ for some absolute constant $c > 0$.*

This algorithm uses $\mathcal{A}$ as a subroutine and it works in *phase*s. Let $\varrho^{(p)}$ denote the sequence of the pth phase. In this phase algorithm Shell works as follows:

Algorithm $\mathcal{SH}$

Initially we mark the blocks that contain no servers.

When $\varrho_i^{(p)}$, the ith request of this phase arrives to block B_s, we compute the demand $D_s(\varrho_{\leq i}^{(p)})$ and the maximal demand

$$D_s^*(\varrho_i^{(p)}) = \max\{D_s(\varrho_{\leq j}^{(p)}) | j \leq i\}$$

for this block (note that these values do not change in the other blocks).

(i) If $D_s^*(\varrho_i^{(p)})$ is less than the number of servers in B_s at that moment, then the request is served by algorithm $\mathcal{A}$, with respect to the block B_s.

(ii) If $D_s^*(\varrho_i^{(p)})$ becomes equal to the number of servers in B_s at that moment, then the request is served by algorithm $\mathcal{A}$, with respect to the block B_s and we mark the block B_s.

(iii) If $D_s^*(\varrho_i^{(p)})$ is greater than the number of servers in B_s at that moment, we mark the block B_s and perform the following subtask until we have $D_s^*(\varrho_i^{(p)})$ servers in that block or we cannot execute the steps (this happens when all the blocks become marked):

> Choose an unmarked block $B_{s'}$ randomly uniformly, and a server from this block also randomly. We move this chosen server to the block B_s (such a move is called a *jump*), either to the requested point, or, if there is already a server occupying that point, to a randomly chosen unoccupied point of B_s. If the number of servers in $B_{s'}$ becomes $D_{s'}^*(\varrho_i^{(p)})$ via this move, we mark that block. In both B_s and $B_{s'}$ we restart algorithm $\mathcal{A}$ from the current configuration of the block.

If we cannot raise the number of servers in block B_s to $D_s^*(\varrho_i^{(p)})$ by repeating the above steps (all the blocks became marked), then phase $p+1$ is starting and the last request is belonging to this new phase.

Starting from a $c \log k$-competitive algorithm and iterating Theorem 37 we get the following result:

Corollary 43 *There exists a $(c_1 \log k)^h$-competitive randomized online algorithm on any μ-HST of height h (here $\mu \geq k$), where c_1 is a constant. Consequently, when $h < \frac{\log k}{\log c_1 + \log\log k}$, this algorithm is $o(k)$-competitive.*

4.2 Results on the k-server problem with rejection

We investigated a more general model in which the requests can be rejected. The results of this subsection are in [16]. In this problem the ith request is a pair (ϱ_i, w_i) for each i, where ϱ_i is a point and $w_i > 0$ is the *penalty* for the rejection. Each request can be served the same way as before, or optionally it also can be rejected at the penalty given along with the request. The cost of an algorithm is the sum of the distances covered by the k servers plus the sum of the penalties of the rejected requests. We proved the following.

Theorem 44 *There is no weakly c-competitive online algorithm for the k-server problem with rejection on uniform spaces with $c < 2k$.*

We presented a weakly competitive algorithm for the k-server problem with rejection on uniform spaces called Threshold. Algorithm Threshold uses the marking procedure seen before and is picky. Let $t > 0$ be some fixed value.

Algorithm $\mathcal{TH}_t$

To each point a counter is assigned. Initially each counter is set to 0 and all the vertices are unmarked. After each request, the marks are updated, followed by a server movement if necessary, as follows:

(i) Increase the value of the counter of the requested point by the penalty of the request.

(ii) If the value assigned to the requested point is at least t, mark that point. At the moment when $k+1$ points become marked, all the marks (including this new one) are erased and all counters are set to 0.

(iii) If the requested point is already covered by a server, then no servers shall move.

(iv) If the requested point is unmarked and not covered then the request is rejected.

(v) If the requested point is marked and not covered, then the least recently moved server is moved to cover the requested point.

Theorem 45 *Algorithm* $\mathcal{TH}_1$ *is weakly* $(2k+1)$*-competitive on uniform spaces.*

Finally we investigated the randomized version of Algorithm $\mathcal{TH}_t$.
Let $t > 0$ be some fixed value.

Algorithm $\mathcal{RTH}_t$

To each point a counter is assigned. Initially each counter is set to 0 and all the vertices are unmarked. After each request, the marks are updated, followed by a server movement if necessary, as follows:

(i) Increase the value of the counter of the requested point by the penalty of the request.

(ii) If the value assigned to the requested point is at least t, mark that point. At the moment when $k+1$ points become marked, all the marks (including this new one) are erased and all counters are set to 0.

(iii) If the requested point is already covered by a server, then no servers shall move.

(iv) If the requested point is unmarked and not covered then the request is rejected.

(v) If the requested point is marked and not covered, then a server is chosen uniformly at random from among the ones occupying an unmarked vertex and is moved to cover the requested point.

Theorem 46 *Algorithm* $\mathcal{RTH}_2$ *is* $(6\mathrm{H_k}+2)$*-competitive on uniform spaces.*

References

[1] N. Alon, U. Arad, Y. Azar, Independent Sets in Hypergraphs with Applications to Routing Via Fixed Paths, *Proc. of APPROX 99, LNCS 1671* (1999), 16–27.

[2] Y. Bartal, B. Bollobás, M. Mendel, A Ramsey-type theorem for metric spaces and its application for metrical task system and related problems, *Journal of Computer and System Sciences* **72** (2006), 890–921.

[3] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection, *SIAM Journal on Discrete Mathematics* **13** (2000) 64–78.

[4] Y. Bartal, M. Mendel, Randomized k-server algorithms for growth-rate bounded graphs, *Journal of Algorithms* **55** (2005), 192–202.

[5] B. Csaba, S. Lodha, A randomized on-line algorithm for the k-server problem on a line, *Random Structures and Algorithms* **29** (2006), 82–104.

[6] B. Csaba, A. Pluhár, A randomized algorithm for on-line weighted bipartite matching problem, *Journal of Scheduling* **11** (2008), 449–455.

[7] R.L. Graham, Bounds for certain multiprocessor anomalies, *Bell System Technical Journal* **45** (1966), 1563–1581.

[8] Cs. Imreh, J. Noga, Scheduling with Machine Cost, *Proc. APPROX'99, Lecture Notes in Computer Science*, Vol. 1761, Springer, Berlin, 1999, 168–176.

[9] H. A. Kierstead, Coloring Graphs On-line, in: A. Fiat, and G. J. Woeginger (eds.) *Online algorithms: The State of the Art, LNCS 1442*, 1998, 281–305.

[10] H. A. Kierstead, On-line coloring k-colorable graphs, *Israel Journal of Mathematics* **105** (1998), 93–104.

[11] L. Lovász, M. Saks, W. T. Trotter, An on-line graph coloring algorithm with sublinear performance ratio, *Discrete Mathematics* **75** (1989), 319–325.

[12] J. Nagy-György, Randomized algorithm for the k-server problem on decomposable spaces, *Journal of Discrete Algorithms* (2009), doi: 10.1016/j.jda.2009.02.005

[13] J. Nagy-György, Online coloring graphs with high girth and high oddgirth, submitted

[14] J. Nagy-György, Cs. Imreh, Online scheduling with machine cost and rejection, *Discrete Applied Mathematics* **155** (2007), 2546–2554.

[15] J. Nagy-György, Cs. Imreh, Online hypergraph coloring, *Information Processing Letters* **109** (2008), 23–26.

[16] J. Nagy-György, Cs. Imreh, k-server problem with rejection, manuscript

[17] S. S. Seiden, A general decomposition theorem for the k-server problem, *Information and Computation* **174** (2002), 193–202.

[18] J. Sgall, On-line scheduling, in: A. Fiat, G.J. Woeginger (eds.) *Online algorithms: The State of the Art, Lecture Notes of Computer Science*, Vol. 1442, Springer-Verlag Berlin, 1998, 196–231.