Metrical Service Systems with Multiple Servers

Ashish Chiplunkar · Sundar Vishwanathan

Received: 30 July 2013 / Accepted: 24 May 2014 / Published online: 19 June 2014
© Springer Science+Business Media New York 2014

Abstract The problem of metrical service systems with multiple servers \((k, l)\)-MSSMS, proposed by Feuerstein (LATIN’98: Theoretical Informatics, Third Latin American Symposium, 1998), is to service requests, each of which is an \(l\)-point subset of a metric space, using \(k\) servers in an online manner, minimizing the distance traveled by the servers. We prove that Feuerstein’s deterministic algorithm for \((k, l)\)-MSSMS actually achieves an improved competitive ratio of \(k \left( \frac{k+1}{l} - 1 \right)\) on uniform metrics. In the randomized online setting on uniform metrics, we give an algorithm which achieves a competitive ratio \(O(k^3 \log l)\), beating the deterministic lower bound of \(\left( \frac{k+1}{l} \right) - 1\). We prove that any randomized algorithm for \((k, l)\)-MSSMS on uniform metrics must be \(\Omega(\log kl)\)-competitive. For the offline \((k, l)\)-MSSMS, we give a factor \(l\) pseudo-approximation algorithm using \(kl\) servers on any metric space, and prove a matching hardness result, that a pseudo-approximation using less than \(kl\) servers is unlikely, even on uniform metrics.

Keywords \(k\)-server · Metrical service system · Online · Approximation

1 Introduction

The problem of metrical service systems (MSS) with multiple servers (MSSMS) generalizes two well-known problems—the \(k\)-server problem [2] and MSS [2, 3]. These problems share a common paradigm, that there is an underlying metric space and
requests are to be served by moving servers on the metric space, in such a way that the total distance traveled by the servers is minimized. For a problem in the online setting, the input is revealed to an algorithm piece by piece, and the algorithm must take irrevocable decisions on seeing each piece. In case of the aforementioned problems, each piece in the input is a request, and the irrevocable decisions are the movements of the servers. A (possibly randomized) online algorithm is said to be $c$-competitive if, on every input, it returns a solution whose (expected) cost is at most $c$ times the cost of optimal solution for that input. The book by Borodin and El-Yaniv [4] gives a nice comprehensive introduction to online algorithms and competitive analysis.

The $k$-server Problem

The $k$-server problem of Manasse et al. [2] is, arguably, the most famous among the problems that are naturally posed in the online setting. The following quote by Koutsoupias, in his beautiful survey on the $k$-server problem [5], upholds the importance of this problem.

*The $k$-server problem is perhaps the most influential online problem natural, crisp, with a surprising technical depth that manifests the richness of competitive analysis. The $k$-server conjecture, which was posed more than two decades ago when the problem was first studied within the competitive analysis framework, is still open and has been a major driving force for the development of the area online algorithms.*

In the $k$-server problem, we have $k$ servers occupying points in a metric space. Each request is a point in the metric space. To serve the request, one of the servers has to be moved to the requested point. The $k$-server conjecture, referred to in the quote, states that there is a $k$-competitive deterministic algorithm for the $k$-server problem.

Manasse et al. [2] proved a lower bound of $k$ on the competitive ratio of any deterministic algorithm on any metric space with more than $k$ points. They proved that the competitive ratio is $k$ only for very specific cases, and posed the $k$-server conjecture. The conjecture has been shown to hold for a few metric spaces, for example, the line [6] and tree metric spaces [7]. Fiat et al. [8] were the first to give an algorithm for the $k$-server problem, with competitive ratio bounded by a function of $k$, which was later improved by Bartal and Grove [9,10]. The breakthrough result was due to Koutsoupias and Papadimitriou, who proved that an algorithm, first proposed by Chrobak and Larmore and called the Work Function Algorithm (WFA), is $(2k−1)$-competitive [11]. In case of randomized algorithms, the best known lower bound that holds for every metric space is $\Omega(\log k / \log \log k)$ due to Bartal et al. [12], and there do exist metric spaces with a lower bound more than $H_k$ [13]. No better algorithm than the Work Function Algorithm is known, even with randomization. The randomized $k$-server conjecture states that there exists a randomized algorithm for the $k$-server problem with a competitive ratio of $O(\log k)$ on any metric space. Recent developments, which ingeniously adapt the primal-dual framework to the online setting, have been applied to the $k$-server problem, culminating in an $O(\log k)$-competitive randomized algorithm for star metrics [14], and an $O(poly\log(k)poly\log(n))$-competitive randomized algorithm on metric spaces with $n$ points [15].

The Generalized Server Problem

This is a generalization of the $k$-server problem, in which the metric space is a disjoint union of $k$ metric spaces, mutually separated by an
infinite distance. A server is located at one point in each of the subspaces. A request is a set of $k$ points, one from each subspace. The request is to be served by moving some server to the requested point which lies in its subspace. An interesting problem, called the Weighted Server problem [16], is a particular case of the Generalized Server problem. This problem is same as the $k$-server problem, except that the servers have different weights, and the cost of moving a server is equal to the product of its weight and the distance covered. We can thus think of this as the Generalized Server problem, where the metric spaces are scaled copies of one another. Fiat and Ricklin [16] were the first to study the Weighted Server problem. They gave a deterministic algorithm with a competitive ratio of $2^{\Omega(k)}$ for uniform metric spaces. They proved that for every metric space, there exist weights so that no deterministic algorithm can have a competitive ratio of less than $(k + 1)!/2$. Chrobak and Sgall [17] studied the weighted 2-server problem on uniform spaces, and proved that the Work Function Algorithm achieves the best possible competitive ratio of 5. They proved that, in contrast with the $k$-server problem, there does not exist a memoryless randomized algorithm with a finite competitive ratio, even for the weighted 2-server problem. More recently, Sitters [18] proved that the Work Function Algorithm, in fact, is competitive for the generalized 2-server problem.

Metrical Service System

The term Metrical Service System (MSS) was coined by Chrobak and Larmore [19] for the following problem. We have a single server in an underlying metric space. Each request is a set of $l$ points from the metric space, where $l$ is called the width, and is a parameter to the problem. To serve a request, the server has to be dispatched to one of the requested points.

Finding shortest paths is a fundamental problem in the offline setting. In the online setting it is posed as the problem of Layered Graph Traversal (LGT). This problem, introduced by Papadimitriou and Yannakakis [20], was a precursor of MSS. Fiat et al. [21] proved that MSS and LGT are, in fact, equivalent problems. That is, there is a $c$-competitive algorithm for MSS if and only if there is a $c$-competitive algorithm for LGT. They also proved that there exists a metric space on which the competitive ratio of any deterministic algorithm for the MSS problem is $\Omega(2^l)$. Further, they gave an $O(9^l)$-competitive algorithm. They proved that $l/2$ is a lower bound on the competitive ratio of any randomized algorithm for LGT. Ramesh [22] gave a better deterministic algorithm for LGT, which achieves a competitive ratio of $l^32^l$, and a randomized $l^{13}$-competitive algorithm. He proved that there exists a metric space on which any randomized algorithm must have competitive ratio $\Omega(l^2/\log^{1+\varepsilon} l)$, for any $\varepsilon > 0$. Burley [23] proved that a variant of the Work Function Algorithm is $O(l \cdot 2^l)$-competitive for the MSS (and hence, the LGT) problem. For the uniform metric space, Chrobak and Larmore [19] proved a lower bound of $l$ on the competitive ratio of any deterministic algorithm, and also gave an algorithm achieving this bound. It is easily seen that the lower bound holds for any metric space with at least $l + 1$ points.

Metrical Service System with Multiple Servers

In a natural generalization of both the $k$-server problem and metrical service system, we have $k$ servers on an underlying
metric space, and each request is a set of \( l \) points from the metric space. To serve a request, one of the \( k \) servers has to move to one of the \( l \) requested points. We call this problem MSS with Multiple Servers, with parameters \( k \) and \( l \) ((\( k, l \))-MSSMS). It is easy to see that this problem is, in fact, a further generalization of the Generalized Server problem. Feuerstein [1] studied this problem for uniform metric spaces, and called it the Uniform Service System with parameters \( k \) and \( l \) (USS(\( k, l \))). He proved a lower bound of \( \left( \frac{k+l}{k+1} \right) - 1 \) on the competitive ratio of any deterministic algorithm for this problem. In fact, this proof holds for any (not necessarily uniform) metric space with at least \( k+l \) points. Feuerstein also gave an algorithm, and proved that its competitive ratio is \( k \cdot \min\left(\frac{k^{l+1} - k}{k-1}, \sum_{i=0}^{k-2} l^i + l^k\right) \). He concluded the paper with the following comment.

An interesting subject of future research is to extend USS(\( k, l \)) to non-uniform metric spaces. This would extend both the work in this paper and the work by Chrobak and Larmore [3] on Metrical Service Systems, where only one server is considered.

Our Results In Sect. 2, we present a simple analysis of Feuerstein’s algorithm, which improves the bound on its competitive ratio proved in [1], to \( k \cdot \left( \left( \frac{k+l}{l} \right) - 1 \right) \). In Sect. 3, we give \( \mathcal{O}(k^3 \log l) \)-competitive randomized algorithm on uniform metric spaces, which beats the deterministic lower bound by an exponential factor. In Sect. 4, we consider the offline \( (k, l) \)-MSSMS problem on arbitrary metric spaces, and we give a pseudo-approximation by a factor of \( l \) using \( kl \) servers. We complement this by proving that it is NP-hard to find a pseudo-approximate solution within any reasonable approximation factor, using less than \( k(l-1) \) servers.

2 Uniform Metric Spaces: The Hitting Set Algorithm

In this section, we analyze the algorithm for \( (k, l) \)-MSSMS on uniform metric spaces given by Feuerstein (Sect. 2 of [1]), which he calls the Hitting Set algorithm. Feuerstein proved that the competitive ratio of this algorithm is at most \( k \cdot \min\left(\frac{k^{l+1} - k}{k-1}, \sum_{i=0}^{k-2} l^i + l^k\right) \), whereas we prove an asymptotically better bound,\(^2\) of \( k \cdot \left( \left( \frac{k+l}{l} \right) - 1 \right) \).

The Hitting Set (HS) algorithm can be described as follows. HS divides the request sequence into phases, the first phase starting with the first request of the sequence. We say that a request produces a fault whenever the requested set of points is disjoint from the set of points occupied by the servers. Each time a request produces a fault, the algorithm behaves as follows. First, it computes a minimum cardinality set \( H \) of points that intersects all the requests that produced a fault during the current phase. If \( |H| \leq k \) then any \( |H| \) servers are made to occupy all points in \( H \). Other-

\(^1\) Feuerstein used ‘\( w \)’ for the width parameter, while we use ‘\( l \)’.

\(^2\) For instance, when \( k = \Theta(l) \), Feuerstein’s bound is \( \Omega(l^l) \), whereas ours is \( \mathcal{O}(c^l) \) for some constant \( c \).
wise, if $|H| > k$, then the phase terminates and a new phase begins with the current request.

**Theorem 1** HS is an $k \cdot \left(\binom{k+l}{l} - 1\right)$-competitive algorithm for $(k, l)$-MSSMS.

**Proof** Feuerstein observed that the adversary must incur a cost of at least one per phase. He then proved that at most $\min\left(k l + 1 - k^2, \sum_{i=0}^{k-2} l^i + l^k\right)$ requests can produce faults, in any phase. We improve this upper bound to $\binom{k+l}{l} - 1$, and our claim follows, since the algorithm pays at most $k$, for every request that produces a fault. Our proof uses the following result from extremal combinatorics, due to Lovász [24].

**Fact 1 (Skew Bollobás Theorem)** Let $A_1, \ldots, A_r$ and $B_1, \ldots, B_r$ be sets such that $|A_i| = l$ and $|B_i| = k$ for all $i$. Suppose $A_i \cap B_i = \emptyset$ for all $i$, and $A_j \cap B_i \neq \emptyset$ for all $i, j$ with $j < i$. Then $r \leq \binom{k+l}{k}$.

Let $A_1, \ldots, A_{r-1}$ be the requests that produced a fault in a given phase, and let $A_r$ be the first request in the next phase, which must also have produced a fault. Let $B_i$ be the set of points occupied by the servers when the request $A_i$ was given. Clearly, $|A_i| = l$ and $|B_i| = k$ for all $i$. Since $A_i$ produced a fault, we have for every $i$ with $1 \leq i \leq r$, $A_i \cap B_i = \emptyset$. On the other hand, since no request between $A_{i-1}$ and $A_i$ produced a fault, the hitting set chosen to serve $A_{i-1}$ must be $B_i$ itself. By the definition of the algorithm, for any $i, j$ with $1 \leq j < i \leq r$ we have $A_j \cap B_i \neq \emptyset$. Applying the skew Bollobás theorem, we have $r \leq \binom{k+l}{k}$. Thus, the number of requests in the phase, that produced a fault, is $r - 1 \leq \binom{k+l}{k} - 1$, as required.

### 3 Uniform Metric Spaces: Randomized Bounds

In this section, we give a randomized version of the Hitting Set algorithm from Sect. 2, which we call the Randomized Hitting Set (RHS) algorithm, and prove that its competitive ratio is $O(k^3 \log l)$. The algorithm is as follows.

RHS divides the sequence of requests in phases, just like the Hitting Set algorithm. Each time a request produces a fault, the algorithm behaves as follows. First, it computes $s$, the minimum cardinality of a set of points that intersects all the requests in the current phase given so far. If $s \leq k$, it chooses a set $H$ uniformly at random from the collection of all the hitting sets of size $s$, and then any $s$ servers are made to occupy all points in $H$. Otherwise, if $s > k$, the current phase ends and a new phase begins with the current request.

It is easily seen that the adversary must incur a cost of at least one per phase. We prove that the expected cost of RHS is $O(k^3 \log l)$ per phase. Note that the value of $s$ increases from 1 to $k$ as the algorithm progresses in a phase. We divide the phase into $k$ sub-phases, where the $i$th sub-phase is the part of the phase when the value of $s$ is equal to $i$. We require the following combinatorial lemma.

**Lemma 1** Let $s$ be the size of the smallest hitting set of an $l$-uniform set system $S$. Then the number of minimum hitting sets of $S$ is at most $l^s$. 

© Springer
Proof We prove the claim by induction on \( s \). For \( s = 1 \), the claim is obvious. For a general \( s \), let \( \mathcal{H} \) be the collection of all minimum hitting sets of \( \mathcal{S} \). Let \( A' \) be an arbitrary set in \( \mathcal{S} \). For each \( e \in A' \), let \( \mathcal{S}_e = \{ A \in \mathcal{S} \mid e \notin A \} \) and \( \mathcal{H}_e = \{ B \setminus \{ e \} \mid B \in \mathcal{H}, e \in B \} \). If \( \mathcal{H}_e \neq \emptyset \), then each set in \( \mathcal{H}_e \) hits each set in \( \mathcal{S}_e \), so the size of smallest hitting set of \( \mathcal{S}_e \) is at most \( s - 1 \). Further, if \( \mathcal{S}_e \) has a hitting set \( H \) of size less than \( s - 1 \), then \( H \cup \{ e \} \) would be a hitting set of \( \mathcal{S} \) of size less than \( s \). This is a contradiction. Thus, the size of the smallest hitting set of \( \mathcal{S}_e \) is exactly \( s - 1 \). Therefore, by induction, we have \( |\mathcal{H}_e| \leq l^{s-1} \), for all \( e \in A' \). Now each set in \( \mathcal{H} \) intersects \( A' \) in some \( e \), and therefore, contributes to \( \mathcal{H}_e \), for some \( e \). Thus, \( |\mathcal{H}| \leq \sum_{e \in A'} |\mathcal{H}_e| \leq l \cdot l^{s-1} = l^s \). \( \square \)

Theorem 2 \( \mathcal{RHS} \) is an \( \mathcal{O}(k^3 \log l) \) competitive algorithm for \((k,l)\)-MSSMS.

Proof First, observe that the boundaries between sub-phases are completely determined by the requests, and are independent of the algorithm. Consider the \( s' \)th sub-phase of any phase. Let \( A_1, \ldots, A_r \) be the sets requested in this sub-phase. Let \( \mathcal{S}_0 \) be the collection of sets requested in the current phase before the \( s' \)th sub-phase started (that is, before \( A_1 \) was requested), and let \( \mathcal{S}_i = \mathcal{S}_0 \cup \{ A_1, \ldots, A_i \} \). For \( i \geq 1 \), let \( \mathcal{H}_i \) be the collection of all minimum hitting sets of \( \mathcal{S}_i \), and \( h_i = |\mathcal{H}_i| \). Thus, each \( \mathcal{H}_i \) is an \( s \)-uniform set system, and \( \mathcal{H}_1 \supseteq \cdots \supseteq \mathcal{H}_r \neq \emptyset \).

Let \( T \) be the set of all \( i \) such that the algorithm faulted on \( A_i \), and let \( H_i \) be the hitting set chosen uniformly at random from \( \mathcal{H}_i \), to serve \( A_i \). Clearly, \( 1 \in T \), otherwise, the \( s' \)th sub-phase would not have started with \( A_1 \). We need an upper bound on \( \mathbb{E}[|T|] = \sum_{i=1}^{r} \text{Pr}[i \in T] \).

For \( i \geq 2 \) and \( j < i \), say that the event \( E_{ji} \) has occurred if \( i, j \in T \), but no \( i' \) between \( j \) and \( i \) is in \( T \). Given that \( j \in T \), this event occurs exactly when \( H_j \), chosen uniformly at random from \( \mathcal{H}_j \), turns out to be in \( \mathcal{H}_{i-1} \), but not in \( \mathcal{H}_i \). Thus, we have \( \text{Pr}[E_{ji} \mid j \in T] = (h_{i-1} - h_i)/h_j \). Note that for a fixed \( i \), the events \( E_{ji} \) are pairwise disjoint, and \( i \in T \) if and only if \( E_{ji} \) occurs for some \( j < i \). Hence,

\[
\text{Pr}[i \in T] = \sum_{j=1}^{i-1} \text{Pr}[E_{ji}] = \sum_{j=1}^{i-1} \text{Pr}[E_{ji} \mid j \in T] \text{Pr}[j \in T] \\
= \sum_{j=1}^{i-1} \frac{h_{i-1} - h_i}{h_j} \times \text{Pr}[j \in T] = (h_{i-1} - h_i) \sum_{j=1}^{i-1} \frac{\text{Pr}[j \in T]}{h_j}
\]

We will now inductively prove that for \( i \geq 2 \), \( \text{Pr}[i \in T] = (h_{i-1} - h_i)/h_{i-1} \). This is evident for \( i = 2 \), since \( \text{Pr}[1 \in T] = 1 \). For an arbitrary \( i > 2 \), we have

\[
\text{Pr}[i \in T] = (h_{i-1} - h_i) \sum_{j=1}^{i-1} \frac{\text{Pr}[j \in T]}{h_j} = (h_{i-1} - h_i) \left[ \frac{1}{h_1} + \sum_{j=2}^{i-1} \frac{h_j - h_j}{h_{j-1}h_j} \right] \\
= (h_{i-1} - h_i) \left[ \frac{1}{h_1} + \sum_{j=2}^{i-1} \left( \frac{1}{h_j} - \frac{1}{h_{j-1}} \right) \right] = \frac{h_{i-1} - h_i}{h_{i-1}}
\]
The expected size of $T$ is given by

$$\sum_{i=1}^{r} \Pr[i \in T] = 1 + \sum_{i=2}^{r} \frac{h_{i-1} - h_i}{h_{i-1}} = \sum_{j=1}^{h_1} \frac{1}{j}.$$ 

Now, $h_1 = |\mathcal{H}_1|$ and $\mathcal{H}_1$ is the collection of all minimum hitting sets of the $l$-uniform set system $S_1$, with minimum hitting set size $s$. By Lemma 1, $h_1 \leq l^s$. Hence, $\mathbb{E}[|T|]$ is $O(s \log l)$. The cost incurred for every request which produced a fault is at most $s$, and hence, the total cost is $O(s^2 \log l)$ in the $s$'th sub-phase. Summing over $s$ from 1 to $k$, we infer that the cost incurred in an entire phase is $O(k^3 \log l)$. ⊓⊔

While we have a $O(k^3 \log l)$-competitive algorithm, we also have the following lower bound on the competitive ratio of any randomized online algorithm for $(k, l)$-MSSMS on uniform metric spaces.

**Theorem 3** The competitive ratio of any randomized online algorithm for $(k, l)$-MSSMS against an oblivious adversary\(^3\) is $\Omega(\log kl)$.

We use a form of Yao’s principle [25] to prove the above theorem. The required form of this principle is as follows [26, 27].

**Fact 2 (Yao’s principle)** Let $\mathcal{A}$ denote the set of deterministic online algorithms for an online minimization problem. If $P$ is a distribution over input sequences such that for some real number $c \geq 1$, $\inf_{A \in \mathcal{A}} \mathbb{E}_{\rho \sim P}[A(\rho)] \geq c \cdot \mathbb{E}_{\rho \sim P}[OPT(\rho)]$, then $c$ is a lower bound on the competitive ratio of any randomized algorithm against an oblivious adversary.

Here $OPT(\rho)$ denotes the optimum cost of serving the sequence $\rho$.

**Proof (Theorem 3)** Let $\mathcal{M}$ be the uniform metric space with $k + l$ points, and let $B_0$ be the set of points occupied by the servers initially. Let $S$ be the collection of all sets of points $B$ such that $|B| = k$ and $|B \setminus B_0| = 1$. The distribution $P$, required to use Fact 2, is given by the following random process. We generate sets of size $l$ uniformly at random and request them, until the complements of all sets in $S$ have been picked. We do not request the last such set picked, say $A$. Note that $\bar{A} \in S$, and $A$ does not appear in the input. Let $\rho$ denote this random input.

The adversary can serve the request sequence $\rho$, simply by covering the points in $\bar{A}$. Since the metric space has only $k + l$ points, $\bar{A}$ intersects with all the sets that appear in $\rho$. Further, since $\bar{A} \in S$, $|\bar{A} \setminus B_0| = 1$. Thus $OPT(\rho) = 1$.

The algorithm incurs a cost exactly when the request is the complement of the set of positions occupied by its servers. This happens with probability $1/\binom{k+l}{i}$, and hence the expected cost incurred by the algorithm for any request is $1/\binom{k+l}{i}$. Let $X_i$ be the random variable representing the cost on the $i$th request, and $Y$ be the random variable representing the number of requests in $\rho$. Then $\mathbb{E}_{\rho \sim P}[X_i | Y \geq i] = 1/\binom{k+l}{i}$ and $\mathbb{E}_{\rho \sim P}[X_i | Y < i] = 0$. Thus, $\mathbb{E}_{\rho \sim P}[X_i] = 1/\binom{k+l}{i} \cdot \Pr_{\rho \sim P}[Y \geq i]$, and therefore, the

\(^3\) An oblivious adversary is an adversary who does not have access to the random bits used by the algorithm.
expected total cost is given by \( \sum_{i=1}^{\infty} \mathbb{E}_{\rho \sim P} [X_i] = (\sum_{i=1}^{\infty} \Pr_{\rho \sim P} [Y \geq i]) / (k+l) = \mathbb{E}_{\rho \sim P} [Y] / (k+l) \). By the coupon collector argument, the expected number of requests in \( \rho \) is \( (\frac{k+l}{i}) \cdot H(\mathcal{S}) - 1 = (\frac{k+l}{i}) \cdot H(\mathcal{S}) - 1 \). Thus the expected cost incurred by the algorithm is \( \left( (\frac{k+l}{i}) \cdot H(\mathcal{S}) - 1 \right) / (k+l) = \Omega(\log kl) \). \( \square \)

4 The Offline Problem

We elaborate on the offline \((k, l)\)-MSSMS problem in this section. Before that, we briefly describe the offline algorithms for the \(k\)-server problem and MSS.

The problem of finding the optimal solution to an instance of \(k\)-server problem can be reduced to the problem of finding a min-cost flow with integral capacities on a suitably constructed directed graph [6], and hence, the offline \(k\)-server problem can be solved in polynomial time. Note that for any instance of min-cost flow with integral capacities, there exists a solution in which all the flows are integral, and which is no worse than any fractional solution. We will use this fact later. The offline MSS problem can be translated to finding the shortest source-to-sink path in a suitably constructed directed graph of size linear in the size of the instance. Thus, MSS too can be solved in polynomial time. In fact, \((k, l)\)-MSSMS can be solved in polynomial time for any constant \(k\). However, the problem is NP-hard for any fixed \(l \geq 2\), when \(k\) is allowed to vary.

In the subsequent subsections, we give a natural Integer Linear Program (ILP) for the offline \((k, l)\)-MSSMS problem. Although it has an unbounded integrality gap, we use it to design a pseudo-approximation algorithm which uses \(kl\) servers, and incurs a cost of at most \(l\) times the optimum. We then prove a lower bound which suggests that nothing better than our pseudo-approximation can be achieved in polynomial time. This result has implications in the online setting also. Feuerstein [1] gave a polynomial time online algorithm for MSSMS on uniform spaces which uses \(kl\) servers, and achieves a competitive ratio of \(kl\) against an adversary using \(k\) servers, where \(l\), as before, is the size of each request set. Our result implies that it is unlikely that a polynomial time competitive algorithm using less than \(kl\) servers exists.

4.1 ILP Formulation and Pseudo-approximation Algorithm

For the metric space \((M, d)\), let \(S = \{s^1, \ldots, s^k\}\) be the set of initial positions of the \(k\) servers and let the request sequence be \(\rho = (R_1, \ldots, R_m)\) where \(R_i \subseteq M\) and \(R_i = \{r_i^1, \ldots, r_i^l\}\). A natural integer linear program is as follows. For each \(1 \leq i < i' \leq m\) and \(1 \leq j, j' \leq l\), we have a variable \(f(i, j, i', j')\), which is 1 if some server was present at \(r_i^j\) to serve \(R_i\), and that server was next made to shift to \(r_{i'}^j\) in order to serve \(R_{i'}\); and 0 otherwise. For each \(1 \leq k' \leq k\), \(1 \leq i \leq m\), \(1 \leq j \leq l\), we have a variable \(g(k', i, j)\) which is 1 if the \(k'\)th server was first shifted to \(r_i^j\), to serve \(R_i\), and 0 otherwise. The objective and the constraints are as given in Fig. 1.

To relax the ILP to an LP, we replace the constraints \(f(i, j, i', j') \in \{0, 1\}\) and \(g(k', i, j) \in \{0, 1\}\) by the constraints \(0 \leq f(i, j, i', j') \leq 1\) and \(0 \leq g(k', i, j) \leq 1\) respectively. Unfortunately, this relaxation has an unbounded integrality gap.

\( \copyright \) Springer
Minimize $\sum_{k',i,j} d(s^{k'}, r_{i}^{j}) g(k', i, j) + \sum_{i,j,i',j'} d(r_{i}^{j}, r_{i'}^{j'}) f(i, j, i', j')$

$\sum_{i,j} g(k', i, j) \leq 1$ for each $1 \leq k' \leq k$

$\sum_{k'} g(k', i, j) + \sum_{i' < i,j'} f(i', j', i, j) \geq \sum_{i'' > i,j''} f(i, j, i'', j'')$ for each $1 \leq i \leq m$, $1 \leq j \leq l$

$\sum_{k',j} g(k', i, j) + \sum_{i' < i,j'} f(i', j', i, j) \geq 1$ for each $1 \leq i \leq m$

$f(i, j, i', j') \in \{0, 1\}$ for all $i, j, i', j'$

$g(k', i, j) \in \{0, 1\}$ for all $k', i, j$

Fig. 1 An ILP formulation of the offline $(k, l)$-MSSMS problem

Claim The LP relaxation of the ILP given in Fig. 1 has unbounded integrality gap when $k, l > 1$.

Proof Consider the uniform metric space on a set of points $M_0 \sqcup M$, where $M_0$ is the set of $k$ points occupied by the servers initially, and $|M| = kl$. Let $R_1, \ldots, R_{\binom{l}{k}}$ be an arbitrary ordering of the $l$-size subsets of $M$. The request sequence is constructed by repeating this ordering $m$ times. This request sequence can be fractionally served as follows. Initially, a $1/l$ fraction of servers shifts to each of the $kl$ points in $M$, and this costs $k$. Having done this, all requests can now be served at zero cost. Thus, the optimum of the LP relaxation is at most $k$.

However, an integer solution must have cost at least $(k - 1)(l - 1)$ on the sequence $R_1, \ldots, R_{\binom{l}{k}}$, starting with any server configuration. This is justified as follows. Given an initial server configuration, let $U$ be the set of points visited to serve the sequence $R_1, \ldots, R_{\binom{l}{k}}$, including the points occupied initially. Then the cost paid is at least $|U| - k$, and $U$ is a hitting set of $R_1, \ldots, R_{\binom{l}{k}}$. Since the sequence $R_1, \ldots, R_{\binom{l}{k}}$ contains all $l$-subsets of a $kl$-sized set, any hitting set of this sequence must have size at least $kl - l + 1$. Thus, $|U| \geq kl - l + 1$, and the cost of serving the sequence is at least $kl - l + 1 - k = (k - 1)(l - 1)$.

Therefore, the total cost of the ILP is at least $m(k - 1)(l - 1)$. Since $m$ can be chosen to be arbitrarily large, the ILP has infinite integrality gap when both $k$ and $l$ are more than 1.

We can, however, round the solution of the LP relaxation to get a pseudo-approximation algorithm for MSSMS.

Theorem 4 There is a polynomial time algorithm, which computes a feasible solution to a given instance of $(k, l)$-MSSMS using $kl$ servers instead of $k$, and such that the cost of the solution is at most $l$ times the cost of the optimum (fractional) solution of the LP relaxation of the ILP.

Proof Given an instance of $(k, l)$-MSSMS with $S$, the set of initial server positions, let the vector $(f^*, g^*)$ be $l$ times the optimum (fractional) solution of the LP relaxation. Then we have $\sum_{k',j} g^*(k', i, j) + \sum_{i' < i,j'} f^*(i', j', i, j) \geq l$ for each $i$. Thus, for each $i$, there exists some $j_i$ such that $\sum_{k'} g^*(k', i, ji) + \sum_{i' < i,j'} f^*(i', j', i, ji) \geq 1$.

 Springer
Let \( r^*_i = r^*_j \). The algorithm solves the LP relaxation and finds \( r^*_1, \ldots, r^*_m \). It then treats \( r^*_1, \ldots, r^*_m \) as an instance of the \( kl \)-server problem, and assuming \( l \) servers to be initially located at each point in \( S \), computes the optimum solution. Note that this is a feasible solution to the given instance of \((k, l)\)-MSSMS, except that it uses \( kl \) servers instead of \( k \). To analyze its cost, observe that the vector \((f^*, g^*)\) gives a (fractional) solution to the instance \( r^*_1, \ldots, r^*_m \) of the \( kl \)-server problem. Hence the cost of the optimum solution to this instance is no more than the cost of \((f^*, g^*)\), which is \( l \) times the cost of the optimum fractional solution to the LP relaxation. Thus, the solution returned by the algorithm has cost at most \( l \) times that of the optimum of the LP relaxation.

4.2 Hardness of Pseudo-approximation

The hardness result, that we prove next, essentially implies that nothing better than what the pseudo-approximation algorithm does, can be achieved in polynomial time. The proof involves a reduction from the problem of finding an approximate vertex cover of a given uniform hypergraph. The following results are known about the hardness of approximating hypergraph vertex cover.

**Fact 3** (Khot and Regev [28]) Assuming the Unique Games Conjecture, (UGC) [29], it is NP-hard to find a factor \( l - \epsilon \) approximation of the minimum vertex cover of a given \( l \)-uniform hypergraph, for all \( l \geq 2 \) and \( \epsilon > 0 \).

**Fact 4** (Dinur et al. [30]; Dinur and Safra [31]) It is NP-hard to find a factor \( l - 1 - \epsilon \) (resp. \( 1.36 - \epsilon \)) approximation of the minimum vertex cover of a given \( l \)-uniform hypergraph, for all \( l \geq 3 \) (resp. \( l = 2 \)) and \( \epsilon > 0 \).

**Theorem 5** Assuming the Unique Games Conjecture, it is NP-hard to pseudo-approximate the \((k, l)\)-MSSMS problem on the uniform metric space with \( n \) points, within any factor polynomial in \( n \), using at most \( k(l - \epsilon) \) servers, for any fixed \( l \geq 2 \) and \( \epsilon > 0 \).

**Proof** Suppose there is a polynomial time algorithm \( A \) which, on a metric space with \( n \) points, can pseudo-approximate the solution of an instance of \((k, l)\)-MSSMS using \( K = k(l - \epsilon) \) servers, within a factor \( f(n) = \mathcal{O}(np) \) for some constant \( p \) independent of \( k \). We use this algorithm to construct a polynomial time algorithm \( B \) which, when given a \( l \)-uniform hypergraph having a vertex cover of size \( k \), outputs a vertex cover of the hypergraph of size \( K \). By running algorithm \( B \) for all \( k \) and picking the best solution, we get a factor \( l - \epsilon \) approximation algorithm for the minimum vertex cover problem on \( l \)-uniform hypergraphs. This contradicts Fact 3.

Algorithm \( B \) does the following. Suppose it is given a hypergraph \( H = (V, E) \) where \(|V| = n, E = \{E_1, \ldots, E_m\}, E_i \subseteq V \) \(|E_i| = l\), and an integer \( k \), with the promise that \( H \) has a vertex cover of size \( k \). \( B \) takes the uniform metric space on the set \( V \sqcup W \) where \( W = \{w_1, \ldots, w_k\} \). Then it constructs the instance of \((k, l)\)-MSSMS, where one server is initially placed at each of the \( w_i \)'s and the request sequence is \( E_1, \ldots, E_m \) repeated more than \( kf(n + k) \) (but polynomially many) times. We will
call each repetition of the sequence $E_1, \ldots, E_m$ a phase. $B$ then uses algorithm $A$ to find a pseudo-approximate solution of this instance, using $K$ servers.

Since the hypergraph has a vertex cover $V' \subseteq V$ of size $k$, the requests can be served by shifting the servers to the points in $V'$ once for all, at a cost $k$. Therefore, the cost of the solution returned by algorithm $A$ is at most $kf(n+k)$, which is less than the number of phases. Thus, there exists a phase in which $A$ incurs zero cost, which means that $A$ does not shift servers during that phase at all. Since all the hyperedges are requested in each phase, the set of points occupied by the $K$ servers in this phase must be a vertex cover of $H$.

Having obtained a solution from algorithm $A$, $B$ simply searches for a phase in which $A$ incurred zero cost, and returns the set of $K$ points occupied by the servers during this phase, as an approximate vertex cover of $H$. It is easy to see that $B$ runs in polynomial time. □

Using the same reduction as above, and Fact 4 instead of Fact 3, we obtain the following hardness result, which does not rely on the validity of the UGC.

**Theorem 6** It is NP-hard to pseudo-approximate the $(k,l)$-MSSMS problem on the uniform metric space with $n$ points, within any factor polynomial in $n$, using at most $k(l - 1 - \varepsilon)$ (resp. $(1.36 - \varepsilon)k$) servers, for any fixed $l \geq 3$ (resp. $l = 2$) and $\varepsilon > 0$.

Theorem 5 implies that unless the UGC is false, there does not exist a competitive online algorithm for $(k,l)$-MSSMS, which uses $k(l - \varepsilon)$ servers and runs in polynomial time. This, in particular, implies the optimality of Feuerstein’s polynomial time algorithm (Theorem 8 of [1]) using $kl$ servers.

## 5 Open Problems

We conclude with a number of interesting problems left open. The most important problem among these is the following.

**Problem 1** Design an $f(k,l)$-competitive deterministic / randomized algorithm for $(k,l)$-MSSMS on arbitrary metric spaces, for some function $f$.

For $(k,l)$-MSSMS on arbitrary metric spaces, we do not have an online algorithm with competitive ratio determined by $k$ and $l$ alone. We believe that such an algorithm exists, and the Work Function Algorithm is the prime candidate for the deterministic case. However, it is not known whether the Work Function Algorithm is competitive, even for the Generalized 3-server problem. We know that if such a deterministic algorithm exists, it must perform a super-polynomial amount of computation on the input. It would be interesting even to find a constant factor competitive algorithm for $(2,2)$-MSSMS on special metric spaces, such as the line.

It is possible to prove a lower bound of $\frac{k+2l-1}{k} - \frac{k+l-1}{k}$ on the competitive ratio of any deterministic algorithm for $(k,l)$-MSSMS, even when the metric space has only two distances [32]. For arbitrary metric spaces, it is also possible to establish a lower bound which is exponential in $k$ for any fixed $l$, exactly on the lines of the lower bound proof by [16] for the weighted server problem. These lower bounds are
improvements to the bound of $\binom{k+l}{k} - 1$ in [1]. Both lower bounds are polynomials in $l$ for a fixed $k$. However, we believe that the competitive ratio is considerably larger, possibly exponential in $l$ as well as $k$. We think that such a bound can be proved by a construction which combines ideas in the $(k+1)/2$ lower bound proof for weighted server problem [16], and those in the $\Omega(2^l)$ lower bound proof for MSS [21]. In the randomized case, we believe that a lower bound of $\Omega(poly(l) \cdot \log k)$ should hold.

**Problem 2** Prove an exponential (resp $\Omega(poly(l) \cdot \log k)$) lower bound on the competitive ratio of any deterministic (resp. randomized) algorithm for $(k, l)$-MSSMS.

For deterministic algorithms on uniform metric spaces, our upper and lower bounds on the competitive ratio differ by the factor $k$. We believe that the upper bound can be improved by carefully choosing the hitting sets in the Hitting Set algorithm described in Sect. 2. Similarly, there is a significant gap between the bounds in case of randomized algorithms. In the upper bound (resp. lower bound) results, we have made the conservative assumption that whenever the algorithm (resp. adversary) faults, it can potentially shift all the $k$ servers, and hence the cost incurred could be $k$, while the adversary (resp. algorithm) shifts at most one server per fault. This assumption introduces a gap of a factor of $k$ between the bounds.

**Problem 3** For $(k, l)$-MSSMS on uniform metric spaces, close the gap between the lower bound of $\binom{k+l}{l} - 1$ and the upper bound of $k \cdot \left(\binom{k+l}{l} - 1\right)$ in the deterministic case, and the lower bound of $\Omega(\log kl)$ and the upper bound of $O(k^3 \log l)$ in the randomized case.

**References**

1. Feuerstein, E.: Uniform service systems with $k$ servers. LATIN ’98: Theoretical Informatics, Third Latin American Symposium. Lecture Notes in Computer Science, vol. 1380, pp. 23–32. Springer, Berlin (1998)
2. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for on-line problems. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, pp. 322–333. ACM, Victoria (1988)
3. Chrobak, M., Larmore, L.: The server problem and on-line games. On-Line Algorithms: Proceedings of a DIMACS Workshop. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 7, pp. 11–64. American Mathematical Society, New York (1992)
4. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, New York (1998)
5. Koutsoupias, E.: The $k$-server problem. Comput. Sci. Rev. 3(2), 105–118 (2009)
6. Chrobak, M., Karloff, H.J., Payne, T.H., Vishwanathan, S.: New results on server problems. SIAM J. Discret. Math. 4(2), 172–181 (1991)
7. Chrobak, M., Larmore, L.L.: An optimal on-line algorithm for $k$-servers on trees. SIAM J. Comput. 20(1), 144–148 (1991)
8. Fiat, A., Rabani, Y., Ravid, Y.: Competitive $k$-server algorithms (extended abstract). In: 31st Annual Symposium on Foundations of Computer Science, pp. 454–463. IEEE Computer Society (1990)
9. Grove, E.F.: The harmonic online $k$-server algorithm is competitive. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, pp. 260–266. ACM, Seattle (1991)
10. Bartal, Y., Grove, E.: The harmonic $k$-server algorithm is competitive. J. ACM 47(1), 1–15 (2000)
11. Koutsoupias, E., Papadimitriou, C.H.: On the $k$-server conjecture. J. ACM 42(5), 971–983 (1995)
12. Bartal, Y., Bollobás, B., Mendel, M.: A ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In: 42nd Annual Symposium on Foundations of Computer Science, pp. 396–405. IEEE Computer Society (2001)
13. Karlin, A.R., Manasse, M.S., McGeoch, L.A., Owicki, S.S.: Competitive randomized algorithms for nonuniform problems. Algorithmica 11(6), 542–571 (1994)
14. Bansal, N., Buchbinder, N., Naor, J.: A primal-dual randomized algorithm for weighted paging. In: 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 507–517. IEEE Computer Society (2007)
15. Bansal, N., Buchbinder, N., Madry, A., Naor, J.: A polylogarithmic-competitive algorithm for the k-server problem. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, pp. 267–276. IEEE (2011)
16. Fiat, A., Ricklin, M.: Competitive algorithms for the weighted server problem. Theor. Comput. Sci. 130(1), 85–99 (1994)
17. Chrobak, M., Sgall, J.: The weighted 2-server problem. Theor. Comput. Sci. 324(2–3), 289–312 (2004)
18. Sitters, R.: The generalized work function algorithm is competitive for the generalized 2-server problem. CoRR abs/1110.6600 (2011)
19. Chrobak, M., Larmore, L.L.: Metrical Service Systems: Deterministic Strategies. Technical Report (1993)
20. Papadimitriou, C.H., Yannakakis, M.: Shortest paths without a map. Theor. Comput. Sci. 84(1), 127–150 (1991)
21. Fiat, A., Foster, D.P., Karloff, H.J., Rabani, Y., Vaidya, S.: Competitive algorithms for layered graph traversal. SIAM J. Comput. 28(2), 447–462 (1998)
22. Ramesh, H.: On traversing layered graphs on-line. J. Algorithms 18(3), 480–512 (1995)
23. Burley, W.R.: Traversing layered graphs using the work function algorithm. J. Algorithms 20(3), 479–511 (1996)
24. Lovász, L.: Flats in matroids and geometric graphs. Combinatorial Surveys. Proceedings of Sixth British Combinatorial Conference, Royal Holloway College, Egham, pp. 45–86. Academic Press, London (1977)
25. Yao, A.C.C.: Probabilistic computations: toward a unified measure of complexity (extended abstract). In: 18th Annual Symposium on Foundations of Computer Science, pp. 222–227. IEEE Computer Society (1977)
26. Borodin, A., El-Yaniv, R.: On randomization in on-line computation. Inf. Comput. 150(2), 244–267 (1999)
27. Stougie, L., Vestjens, A.P.A.: Randomized algorithms for on-line scheduling problems: how low can’t you go? Oper. Res. Lett. 30(2), 89–96 (2002)
28. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2 - \varepsilon$. J. Comput. Syst. Sci. 74(3), 335–349 (2008)
29. Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings on 34th Annual ACM Symposium on Theory of Computing, pp. 767–775. ACM, Montreal (2002)
30. Dinur, I., Guruswami, V., Khot, S., Regev, O.: A new multilayered PCP and the hardness of hypergraph vertex cover. SIAM J. Comput. 34(5), 1129–1146 (2005)
31. Dinur, I., Safra, S.: The importance of being biased. In: Proceedings on 34th Annual ACM Symposium on Theory of Computing, pp. 33–42. ACM, Montreal (2002)
32. Chiplunkar, A., Vishwanathan, S.: Metrical service systems with multiple servers. CoRR abs/1206.5392 (2012)