Online unit clustering: variations on a theme

Leah Epstein* Asaf Levin† Rob van Stee‡

April 17, 2008

Abstract

Online unit clustering is a clustering problem where classification of points is done in an online fashion, but the exact location of clusters can be modified dynamically. We study several variants and generalizations of the online unit clustering problem, which are inspired by variants of packing and scheduling problems in the literature.

1 Introduction

Clustering problems involve a partition of a set of points into groups, which are often called clusters. The goal is typically the optimization of a given objective function. Clustering problems are fundamental and have numerous applications. Such applications include the usage of clustering for computer related purposes, such as information retrieval and data mining, but also various applications in other fields, such as medical diagnosis and facility location.

In the online scenario which we study, points are presented one by one to the algorithm, and must be assigned to clusters upon arrival. An assignment of a point to a cluster becomes fixed at this time, and cannot be changed later. We measure the performance of an online algorithm $\mathcal{A}$ by comparing it to an optimal offline algorithm OPT using the competitive ratio, which is defined as $\sup_\sigma \frac{\mathcal{A}(\sigma)}{\text{OPT}(\sigma)}$. Here σ is the input, i.e., a sequence of request points, and $\text{ALG}(\sigma)$ denotes the cost of an algorithm ALG for this input, which is the number of clusters in the basic problem, and is a function of the solution in a more general setting. For an algorithm ALG, if σ is clear from the context, we drop σ from the notation and use ALG to denote the cost of the algorithm ALG. For randomized algorithms, we replace $\mathcal{A}(\sigma)$ with $E(\mathcal{A}(\sigma))$, and define the competitive ratio as $\sup_\sigma \frac{E(\mathcal{A}(\sigma))}{\text{OPT}(\sigma)}$. An algorithm with competitive ratio of at most $\mathcal{R}$ is called $\mathcal{R}$-competitive.

A study of online partitioning of points into clusters was presented by Charikar et al. [6]. They considered the so called *online unit covering problem*. In this problem, a set of n points needs to be covered by balls of unit radius, and the goal is to minimize the number of balls used. They gave an upper bound of $O(2^d d \log d)$ and a lower bound of $\Omega(\log d/ \log\log\log d)$ on the competitive ratio of deterministic online algorithms in d dimensions. This problem is fully online in the sense that points arrive one by one, each point needs to be assigned to a ball upon arrival, and if it is assigned to a new ball, the exact location of this

*Department of Mathematics, University of Haifa, 31905 Haifa, Israel. lea@math.haifa.ac.il.

†Department of Statistics, The Hebrew University, Jerusalem, Israel. levinas@mscc.huji.ac.il.

‡Max-Planck-Institut für Informatik, Saarbrücken, Germany. vanstee@mpi-inf.mpg.de. Research supported by the German Research Foundation (DFG). Work performed while this author was at the University of Karlsruhe, Germany.

ball is fixed at this time. The tight bounds on the competitive ratio for $d = 1$ and $d = 2$ are respectively 2 and 4.

Chan and Zarrabi-Zadeh [5] introduced the *unit clustering problem*. In this problem the input and goals are very similar to unit covering. This is an online problem as well, but it is more flexible in the sense that it does not require to fix the exact position of each ball in advance. The algorithm needs to make sure that a set of points which is assigned to one ball (cluster) can always be covered by a single ball. The goal is still to minimize the total number of balls used. Therefore, the algorithm may terminate having clusters that still have more than one option for their location. In an offline scenario, unit covering and unit clustering are the same problem. However, in the online model, an algorithm now has the option of shifting a cluster after a new point arrives, as long as this cluster still covers all the points that are assigned to it. In [5, 12], the two-dimensional problem is considered in the L_∞ norm rather than the L_2 norm. Thus "balls" are actually squares or cubes. In this paper, we focus on the case $d = 1$, for which the two metrics are identical.

Note that online clustering is an online graph coloring problem. If we see the clusters as colors, and the points are seen as vertices, then an edge between two points occurs if they are too far apart to be colored using the same color. The resulting graph for the one-dimensional problem is the complement of a unit interval graph (alternatively, the problem can be seen as a clique partitioning problem in unit interval graphs). See [18] for a survey on online graph coloring. Note that online coloring is a difficult problem that does not admit a constant competitive ratio already for trees [15, 20]. There is a small number of classes that admit constant competitive algorithms, one of which is interval graphs [19].

For the one-dimensional case, it was shown in [5] that several naïve algorithms all have a competitive ratio of 2. Some of these algorithms are actually designed to solve the unit covering problem and thus cannot be expected to overcome this bound (due to the lower bound of [6]). Chan and Zarrabi-Zadeh [5] also showed that any randomized algorithm for unit covering has a competitive ratio of at least 2. Thus randomization by itself without additional relaxation of the problem would not allow to overcome the lower bound of 2. To demonstrate the difference between unit covering and unit clustering, i.e., the role of shifting clusters, they presented a randomized algorithm with a competitive ratio of $15/8 = 1.875$ (later improved by the same authors to 1.833 in [23]). Finally, they showed a lower bound of $4/3$ on the competitive ratio of any randomized algorithm. The deterministic lower bound that is implied by their work is $3/2 = 1.5$. A multi-dimensional extension of their algorithm, that they design, results in a $11/3$-competitive algorithm for two dimensions, and a $2^d \cdot 11/12$-competitive algorithm for general d.

Epstein and van Stee [12] improved these results by presenting a relatively simple *deterministic* algorithm which attains a competitive ratio of $7/4 = 1.75$. Using the construction presented by Chan and Zarrabi-Zadeh [5], this implies an upper bound of $2^d \cdot 7/8$ in d dimensions. Moreover, they improve the randomized lower bound to $3/2 = 1.5$ and show a deterministic lower bound of $8/5 = 1.6$. Finally, they give a deterministic lower bound of 2 and a randomized lower bound of $11/6 \approx 1.8333$ in two dimensions. The deterministic lower bound holds for the L_2 norm as well.

In the current paper, we study several variants and generalizations of this problem. These are presented below together with our results. For most versions, we give matching upper and lower bounds on the best possible performance of an online algorithm. In all versions except the one with resource augmentation, the maximum possible length of a cluster is still 1 as before.

We study the following problems.

1. **Clustering with rejection.** An input point has a non-negative value associated with it, which is called its rejection penalty. For each point that is not assigned to a cluster, its penalty must be paid. Problems with rejection have application in customer service, where the rejection penalty represents the compensation to be paid to a disappointed customer, or if a customer cannot be refused, it is

the cost for servicing this customer in an alternative way (such as out-sourcing). We note that the clustering with rejection problem is the online coloring with rejection problem when restricted to the complement of a unit interval graph. A previous study of the online coloring with rejection of other graph classes appears in [11]. Many other combinatorial optimization and online problems were studied in this scenario, see e.g. [13, 4, 7, 8]. We design an algorithm of competitive ratio 3 for this problem and prove a matching lower bound.

2. **Max clustering.** Every input point has a weight. Points are to be assigned to clusters, so that every cluster would not exceed the length of 1. The cost of a cluster is the maximum weight of any point assigned to it. The goal is to minimize the total cost of the clusters. Max coloring of graphs was introduced by Pemmaraju, Raman and Varadarajan [22] and studied in an online environment in [10].

   We show that the GRID algorithm has competitive ratio 2 for this problem and prove a matching lower bound.

3. **Clustering with cardinality constraints.** In this variant we are given a parameter k, where each cluster can serve at most k points that can all be covered by one interval of length 1. This model assumes that the service provided by the cluster is limited to a given number of clients.

   A large amount of work on capacitated variants of combinatorial optimization and online problems exists in the literature [21, 14, 3, 9].

   We design algorithms of competitive ratio $\frac{3}{2}$ for $k = 2$ and 2 for $k \geq 3$. We prove matching lower bounds for $k = 2$ and $k \geq 4$ and a lower bound of 1.75 for $k = 3$.

4. **Clustering with resource augmentation.** Resource augmentation, or extra resource analysis is a generalization of competitive analysis, where the online algorithm may use resources that are not available to an optimal offline algorithm to which the online algorithm is compared [17]. We study a resource augmented variant of clustering where the online algorithm uses clusters of length at most b, where $b > 1$ is a given parameter, whereas the clusters of the offline algorithm are still of length at most 1. We show a tight bound of 1 for any $b \geq 2$, a lower bound of $3/2$ for any cluster size in $(1, 2)$, and an algorithm of competitive ratio exactly $5/3$ for $b \geq 3/2$.

5. **Clustering with temporary request points.** In this variant, requests are not permanent but arrive and leave over time. The duration of a request point is unknown until the time it leaves. Each point needs to be covered at all times of its duration. The momentary cost of an algorithm at any point in time is determined by the number of clusters that are serving a nonempty subset of request points. The cost of an algorithm is its maximum momentary cost over time.

   Previous work on online problems with temporary requests can be found in [16, 1, 2].

   We show that the GRID algorithm has competitive ratio 2 for this problem and prove a matching lower bound.

Note that in this paper we consider only the (absolute) competitive ratio and not the asymptotic competitive ratio. This is motivated by the fact that in all the variants that we consider one can repeat the input sequence multiple times in disjoint parts of the real line. These disjoint parts cannot be assigned to the same sets of clusters, and therefore the cost of the solution is the sum of all costs (of the different parts).

We conclude this paper in Section 7 by noting that most of our results apply also to the similar variants of the online covering problem.

2 Clustering with rejection

In this variant of the problem, each point p has two attributes. In addition to its location on the real line, a point has a non-negative weight w_p (which is interpreted as its rejection penalty) associated with it. Each arriving point must be either assigned to a cluster upon arrival (i.e., accepted) or rejected. The set of points assigned to one cluster must lie within an interval of length 1.

A rejected point does not need to be assigned to a cluster; instead of the assignment, the algorithm pays a penalty for not serving the point. Thus the cost of an algorithm $\mathcal{A}$ is the sum of rejection penalties of points rejected by $\mathcal{A}$ plus the number of clusters used for the accepted points.

Clustering with rejection is a generalization of standard unit clustering, i.e., unit clustering is the special case of clustering with rejection where all rejection penalties are infinite.

The following algorithm GRID ([5]) is used as a building block in this section.
For every integer $-\infty < k < \infty$, GRID considers points arriving in the interval $I_k = (k, k+1]$ separately and independently from other points. Upon arrival of the first point in I_k, a new cluster is opened in the interval $[k, k+1]$ and all future points in this interval are assigned to this cluster.

We prove a tight bound of 3 for this problem. We begin with a description of an algorithm which is based on GRID.

Denote the subsequence (of the input sequence) of points which belong to I_k by P_k. As long as the total weight of points in P_k does not exceed $\frac{1}{2}$, all such points are rejected. Let p_k be the first point which causes the total weight of points in P_k that arrived so far to be at least $\frac{1}{2}$. Upon arrival of p_k, a new cluster is opened in the interval $[k, k+1]$ and all future points in P_k are assigned to it. We call this algorithm REJECTIVE GRID (RGRID).

Theorem 1 *The competitive ratio of* RGRID *is 3 and this is best possible.*

Proof We start with the proof of the upper bound. Consider an optimal offline algorithm OPT. We analyze each interval of the form $(k, k+1]$ separately, thus we assign shares of the cost of OPT to such intervals so that the sum of shares of the cost, that are assigned to the union of all intervals, is exactly OPT. The cost of every point rejected by OPT is simply assigned to the unique interval it belongs to. Without loss of generality, we assume that OPT always uses clusters of size 1. For every cluster of OPT, this cluster then contains points of exactly two such intervals. We thus assign its cost in equal shares to both these intervals.

We will prove that the cost of RGRID on every interval is at most three times the cost of OPT that is assigned to it. Consider an interval $I_k = (k, k+1]$ that contains at least one point (the algorithm pays a total of zero on an interval with no points, and thus the cost for this interval clearly does not exceed three times the cost of OPT that was assigned to this interval).

If the total weight of points in I_k, which we denote by r_k, is less than $\frac{1}{2}$, the algorithm does not open a cluster for this interval and pays exactly r_k. On the other hand, OPT either covers some of these points by at least one cluster, or rejects all these points. In the first case, at least one cluster of OPT overlaps with I_k, so a share of at least $\frac{1}{2}$ from the cost of OPT was assigned to I_k. In the second case, a share of cost of at least r_k was assigned to this cluster. In both cases the assigned cost is no smaller than the cost of RGRID.

Finally, if $r_k \geq \frac{1}{2}$, the cost of RGRID on I_k is no larger than $\frac{3}{2}$. This cost results from the rejection penalties of all points arriving before p_k, which is less than $\frac{1}{2}$, and the cost of one cluster, which is an additional 1. Similarly to the previous case, OPT either has at least one cluster overlapping with I_k, or rejects all points of P_k. In the first case a cost of at least $\frac{1}{2}$ is assigned to this interval and in the second case, a cost of $r_k \geq \frac{1}{2}$ is assigned to it. The ratio of the cost of RGRID on I_k, and the share of cost of OPT that is assigned to I_k, is no larger than 3.

We next prove a lower bound of 3 on the competitive ratio of any algorithm. Let N be a large enough integer. Consider the following sequence. The first phase consists of the points $\frac{i}{16N}$ for $i = 1, 2, \ldots$, each one of these points has a weight of $\frac{1}{N}$ (the denominator $16N$ is chosen for simplicity). These points are presented one by one until a cluster is opened. The point for which a cluster is opened is the last point of this phase. If no cluster is opened, the first phase stops after $4N$ points are given, in this case no further points will be defined and the sequence stops. Otherwise, let i' be the index of the last point presented. The next phase consists of multiple instances of the point $\frac{i'-1}{16N} - 1$, where each such instance has a penalty of $\frac{1}{N}$. Note that the distance between these points and the point for which a cluster was opened is $1 + \frac{1}{16N}$, thus the points of the second phase cannot be assigned to the same cluster. Such points are presented for $i = 0, \ldots$, until a new cluster is opened or until $4N$ points are presented. The sequence terminates here in both cases.

Consider first the case where $4N$ points were presented in the first phase and no cluster was opened. All points of the first phase lie in an interval of length $\frac{1}{4}$, thus they can fit in one cluster and $\text{OPT} = 1$. The total rejection penalty paid by the algorithm is 4, which results in a competitive ratio of 4.

Next, we consider the case that $4N$ points were presented in the second phase, but no additional cluster was opened. Note that the interval $[\frac{i'-1}{16N} - 1, \frac{i'-1}{16N}]$ contains all points but the last point of phase 1, and thus OPT can open one cluster and reject just one point. We get $\text{OPT} \leq 1 + \frac{1}{N}$. However, the algorithm pays at least 1 for the first phase and 4 for the rejection penalties of the second phase, which gives a total of at least 5. This case results in a competitive ratio of more than 4.

Consider now the case where clusters were opened in both phases. Let i'' denote the index of the point for which a cluster was opened in the second phase. The cost of the algorithm is $\frac{i'+i''}{N} + 2$. As we saw above, we have $\text{OPT} \leq 1 + \frac{1}{N}$. Another possible offline solution would be to reject all points, and get the cost $\frac{i'+i''+2}{N}$. Thus $\text{OPT} \leq \min\{\frac{i'+i''+2}{N}, \frac{N+1}{N}\}$. If $i' + i'' + 3 \leq N$, the cost of the algorithm is at least $\frac{i'+i''}{N} + 2 \geq 3(\frac{i'+i''+2}{N}) \geq 3\text{OPT}$. Otherwise, $i' + i'' \geq N - 2$, the cost of the algorithm is at least $\frac{i'+i''}{N} + 2 \geq 3 - \frac{2}{N} \geq \frac{3N-2}{N+1}\text{OPT}$. Since N can be chosen to be arbitrarily large, we obtain a lower bound of 3 on the competitive ratio. □

3 Max clustering

In this variant of the problem, each point p has a non-negative weight w_p associated with it. Each arriving point must be assigned to a cluster upon arrival. The set of points assigned to one cluster must lie within an interval of length 1. The cost of a cluster is the largest weight of any point assigned to this cluster. The cost of an algorithm is the sum of costs of the clusters defined by the algorithm.

This problem is the generalization of standard unit clustering, since unit clustering is the special case of max clustering where all weights are equal.

We prove a tight bound of 2 for this problem. The upper bound is achieved by simply applying GRID for this problem.

Theorem 2 *The competitive ratio of* GRID *is 2 and this is best possible.*

Proof We start with the proof of the upper bound. Consider an optimal offline algorithm OPT. We analyze each interval of the form $(k, k+1]$ separately, thus we assign shares of the cost of OPT to such intervals, so that the sum of shares of cost that are assigned to the union of all intervals is exactly OPT. Without loss of generality, we assume that OPT always uses clusters of size 1. For every cluster of OPT of cost w, this cluster then contains points of exactly two such intervals. We therefore assign its cost in equal shares to both these intervals, i.e., a cost of $\frac{w}{2}$ to each one of them.

We will prove that the cost of GRID on every interval is at most twice the cost of OPT that is assigned to it. Consider an interval $I_k = (k, k+1]$ that contains at least one point (the algorithm pays a total of zero on an interval with no points, and thus the cost for this interval clearly does not exceed twice the cost of OPT that was assigned to this interval).

Consider an interval I_k for which the cluster in GRID has weight a. Thus I_k contains a request point of weight a. This point is covered by some cluster of OPT which has weight at least a. Thus a cost of at least $\frac{a}{2}$ was assigned to this interval. Therefore, the ratio of the cost of RGRID on I_k and the cost assigned to I_k is no larger than 2.

We next prove a lower bound of 2 on the competitive ratio of any algorithm. Let M be a large enough integer, and let $N = M^2$. Consider the following sequence. The first request point is 0, and has weight 1. Clearly the algorithm must open a cluster for this point. Additional points are presented until the algorithm opens an additional cluster or until all these points are presented. The points are $\frac{i}{N}$ for $i = 0, \ldots, N$, where the point $\frac{i}{N}$ has weight $1 + \frac{iM}{N}$. If no additional cluster was opened, a last request for the point $1 + \frac{1}{N}$ with weight $M + 1$ arrives.

If the last point arrived, it means that the algorithm must open a cluster for this point, since its distance from the very first point is larger than 1. Thus the cost of the algorithm for the first cluster is the weight of the point in position 1, which is $M + 1$, and the cost of the second cluster is $M + 1$ as well. An optimal algorithm would assign all points but the first one to a common cluster, having a cost of $M + 1$ for this cluster, and the first point can be assigned to an additional cluster, which will have the cost 1. This gives a competitive ratio of at least $\frac{2(M+1)}{M+2} = 2 - \frac{2}{M+2}$.

If the last point did not arrive, it means that the sequence stopped right after a second cluster was opened. Let $i' \geq 1$ denote the index of the last request point that was presented. An optimal algorithm would use a single cluster of weight $1 + \frac{i'M}{N}$ for all requests. The algorithm uses two clusters, where the first cluster contains all points but the last one, and thus costs $1 + \frac{(i'-1)M}{N}$, and the second cluster costs $1 + \frac{i'M}{N}$. We get a competitive ratio of at least $\frac{2+(2i'-1)M/N}{1+i'M/N} = \frac{2M+2i'-1}{M+i'} \geq 2 - \frac{1}{M}$.

Since M can be taken to be arbitrarily large, this results in a lower bound of 2 on the competitive ratio of any algorithm. □

4 Clustering with cardinality constraints

In this section we consider the unit clustering problem, where a parameter k limits the number of points that can be assigned to one cluster. Clearly, the case $k = 1$ is trivial. A cluster can contain a set S of points if it is contained in an interval of length 1, and on top of that, $|S| \leq k$.

The next proposition resolves the case $k = 2$. For this case we can apply a greedy algorithm that inserts a point into an existing cluster if possible, and otherwise opens a new cluster for it. Note that this approach is based on a greedy algorithm for finding a maximum cardinality matching.

Proposition 1 *The competitive ratio of the greedy algorithm for $k = 2$ is $\frac{3}{2}$, and this is best possible.*

Proof For the upper bound, we show the relation to maximum matchings. Let m be the cardinality of a maximum matching on the graph of request points (when two points share an edge if the distance between them is at most 1). Let n be the number of request points. We have OPT $= n - m$, since an optimal algorithm is one that maximizes the number of clusters that cover two points. Since for each edge of the maximum matching implied by OPT at least one endpoint was assigned to a cluster with two points by the algorithm (by the greedy assignment rule), we get that at least m points are in such clusters. Thus the cost of the algorithm is at most $n - \frac{m}{2}$. Using $n \geq 2m$ we get $\frac{n-m/2}{n-m} \leq 1 + \frac{m/2}{n-m} \leq \frac{3}{2}$.

For the lower bound, consider the two points 1 and 2. If the algorithm assigns them to two clusters the sequence stops. Clearly $\text{OPT} = 1$, which gives a competitive ratio of 2. Otherwise, two additional points 0 and 3 are presented. The algorithm opens two new clusters, whereas $\text{OPT} = 2$, this gives a competitive ratio of $\frac{3}{2}$. □

We next consider the case $k = 3$.

Theorem 3 *Any algorithm for $k = 3$ has competitive ratio of at least $\frac{7}{4} = 1.75$.*

Proof The first three points are in positions $2, 2.5, 3$. These three points must be assigned by the online algorithm to one cluster that we denote by A (otherwise, the input sequence stops and the online algorithm paid at least twice the cost of the optimal offline solution). Note that by the cardinality constraint no further point can be assigned to A. In this case, we say that A is *full*. The next point is in position 3.5 and it must be assigned to a new cluster that we denote by B. The fifth point is in position 4.5, and it can be assigned to B or to a new cluster C.

- Assume that the fifth point is assigned to cluster B. The location of B is then fixed. The sixth point is in position 5, and it cannot be assigned to A or B, and hence we must open a new cluster denoted as C for this point. The seventh point is in position 4, and it can be assigned to either B or C or to a new cluster D.
  - Assume that the seventh point is assigned to cluster B. B is now full. The next point is in position 4.4. This point cannot be assigned to cluster B due to the cardinality constraint, and hence it must be assigned to either C or to a new cluster D.
    * Assume that point 8 is assigned to cluster C. In this case, the next points are at positions $1.7, 2.8, 3.9, 5.5$. None of these points can be assigned to existing clusters, thus there are now seven clusters. The points can be served using only four clusters that contain three points each: $[1.7, 2.5], [2.8, 3.5], [3.9, 4.4], [4.5, 5.5]$. Therefore, the competitive ratio in this case is at least $7/4$.
    * Assume that point 8 is assigned to cluster D. The next two points are at 3.3 and 2.1, two new clusters are opened for them. Two additional points then appear at 1.1 and 2.2, and at least one additional cluster must be opened for them, giving seven clusters. The points can be served using only four clusters: $[1.1, 2.1], [2.2, 3], [3.3, 4], [4.4, 5]$. Therefore, the competitive ratio in this case is at least $7/4$.
  - Assume that the seventh point is assigned to cluster C. The position of C is then fixed. The next points appear at positions $1.1, 2.2, 3.4, 5.5$ and must be assigned to four new clusters. The points can be served using only four clusters: $[1.1, 2.0], [2.2, 3], [3.4, 4], [4.5, 5.5]$. Therefore, the competitive ratio in this case is at least $7/4$.
  - Assume that the seventh point is assigned to cluster D. Now, two points appear at positions 2.9 and 1.8. Neither one can be assigned to an existing cluster, so there are now six clusters. The points can be served using only three clusters: $[1.8, 2.5], [2.9, 3.5], [4, 5]$. Therefore, the competitive ratio in this case is at least 2.
- Assume that the fifth point is assigned to cluster C. The sixth point is in position 1.1, and it must be assigned to a new cluster D. The seventh point is in position 0.1 and it can be either assigned to cluster D or to a new cluster E.

- Assume that the seventh point is assigned to cluster D. The next points appear at positions $0, 1.2, 2.3$ and must be assigned to three new clusters, since A is full and the location of D is fixed. The points can be served using four clusters: $[0, 0.1], [1.1, 2], [2.3, 3], [3.5, 4.5]$. Therefore, the competitive ratio in this case is at least $7/4$.
- Assume that the seventh point is assigned to cluster E. The eighth point is in position 4. The input so far can be served using three clusters: $[0.1, 1.1], [2, 3], [3.5, 4.5]$. Therefore, the online algorithm cannot use a new cluster for the eighth point. Since the distance to the sixth and seventh point is too large, the online algorithm must assign the eighth point to B or C.
  * Assume that the eighth point is assigned to cluster B. The next points are at positions $3.3, 3.4$. At most one of these can be assigned to B, the other one must be assigned to a new cluster. Finally there is a point at position 2.2, it must also be assigned to a new cluster. The points can be served using four clusters: $[0.1, 1.1], [2, 2.5], [3, 3.4], [3.5, 4.5]$. Therefore, the competitive ratio is again at least $7/4$.
  * Assume that the 8-th point is assigned to cluster C. The final two points appear at positions 5.5 and 2.3 and must be assigned to new clusters: A is full and C cannot serve both 5.5 and 4. The points can be served using four clusters: $[0.1, 1.1], [2, 2.5], [3, 4], [4.5, 5.5]$. Therefore, the competitive ratio in this case is at least $7/4$.

We conclude that in all cases the competitive ratio of the online algorithm is at least $7/4$. □

Finally, we consider the case $k \geq 4$. For this case we can show a tight bound of 2. The algorithm CONSTRAINED GRID (CGRID) acts as follows. CGRID applies GRID in order to partition the request points into mega-clusters. Each mega-cluster is partitioned in an online fashion into clusters consisting of at most k points. All these clusters are defined in the exact same interval as the mega-cluster. Thus, there is at most one active cluster for each mega-cluster at each time. A new point is assigned to a mega-cluster and then to an active cluster of this mega-cluster. If as a result the active cluster has k points, it is closed. If a point is assigned to a mega-cluster which has no active cluster, such an active cluster is opened.

Theorem 4 CGRID *has a competitive ratio of 2, which is best possible for any $k \geq 4$.*

Proof Consider the cost OPT' of an optimal solution OPT' to the problem P' where every cluster must be contained in an interval of the form $(k, k+1]$. We can show that $\text{OPT}' \leq 2\text{OPT}$ as follows. Given a cluster of OPT, $[x, y]$ where $y \leq x + 1$. We can assume without loss of generality that x and y are request points, otherwise we can reduce the length of the cluster so that it fulfills this property. Let $z = \lceil x \rceil$. If $z \geq y$, we are done, since the interval is already contained in an interval of the form $(k, k+1]$. Otherwise, let z' be the leftmost request point in $(x, y]$ that is larger than z, since the input consists of a finite number of points, and since y is a request point and $y > z$, the point z' must exist. We split this cluster into the two parts $[x, z]$ and $[z', y]$. We show that our algorithm provides an optimal solution to P'. Since clusters of OPT' are always contained in an interval of the form $(j, j+1]$, given a set of points J_k in the interval $(k, k+1]$, $\left\lceil \frac{|J_k|}{k} \right\rceil$ clusters of OPT' are required for this set, and this is exactly the number of clusters that the algorithm uses. Thus the competitive ratio of CGRID is at most 2.

To prove the lower bound, we define the following sequence. It starts with k requests, $k-2$ of the point 1 and two of the point 2. At this time $\text{OPT} = 1$ and thus if at least two clusters are opened we are done. If a single cluster is opened, this cluster cannot be used any further. Next, two points arrive which are $\frac{4}{3}$ and $\frac{5}{3}$. If two additional clusters are opened, the point 3 is requested. We have $\text{OPT} = 2$ (by assigning the $k-2$ points at 1, and the two points at $\frac{4}{3}$ and $\frac{5}{3}$ to one cluster, and the other three points to another cluster). The new point is too far from any cluster that can still receive points and thus the algorithm uses four clusters. Otherwise, a single new cluster is opened. Two new points are presented; $\frac{8}{3}$ and $\frac{1}{3}$. These points require two new clusters. However, an optimal solution would be to assign all k points in the interval $[\frac{1}{3}, \frac{4}{3}]$ to one cluster, and the remaining four points in $[\frac{5}{3}, \frac{8}{3}]$ to another cluster. The competitive ratio is again 2. $\square$

5 Clustering with resource augmentation

In this variant of the problem, the online algorithm uses clusters of maximum length b which is larger than the length of clusters used by an optimal offline algorithm which is used for comparison. Thus, each arriving point must be assigned to a cluster upon arrival. The set of points assigned to one cluster by an online algorithm must lie within an interval of length b. The cost of an algorithm is the number of the clusters defined by the algorithm. An offline algorithm can assign a set of points S to one cluster if the maximum distance between any two points is S is at most 1.

5.1 Two initial results

The typical question in problems with resource augmentation is whether it is possible to reach a competitive ratio of 1, or an even smaller competitive ratio. We show that the former is impossible for $b < 2$ and the latter is never possible.

Proposition 2 *For any $b > 1$, the competitive ratio of any algorithm is at least 1. For any $1 < b < 2$, the competitive ratio of any algorithm is at least $\frac{3}{2}$.*

Proof An input which consists of a single point proves the first claim. The second claim follows from the lower bound proof in Proposition 1. The first case is the same. In the second case, two new clusters must be opened if $b < 2$. $\square$

We define the following algorithm CENTER, which is based on an algorithm suggested in [5] for the standard unit covering problem. For every new request point, it is assigned to an existing cluster if possible. Otherwise, for a request at x, a cluster $[x-1, x+1]$ is opened.

Proposition 3 *The competitive ratio of* CENTER *for $b \geq 2$ is 1.*

Proof We assign each cluster opened by CENTER to a cluster used by an optimal offline algorithm OPT. The assignment is done so that at most one cluster of CENTER is assigned to each cluster of OPT, and thus the competitive ratio follows.

Given a cluster of CENTER, $A = [a-1, a+1]$, the point a is a request point. Thus OPT must have a cluster O which contains it. We assign A to O. Note that O is contained in A. We next show that no other clusters of CENTER are assigned to O. Assume by contradiction that cluster $B = [b-1, b+1]$ of CENTER is assigned to O. Then b is a request point. Without loss of generality, assume that B is opened after A. Then the point b does not belong to the interval $[a-1, a+1]$, and thus b does not belong to O, contradiction. $\square$

5.2 Simple algorithms

Before we design an algorithm, we show that simple generalizations of previously known algorithms that simply use longer clusters do not have a competitive ratio which is smaller than 2.

For a given value $1 < b < 2$ we define the algorithm LONG GREEDY (LGREEDY) as follows. A new point p fits into a cluster C if p can be assigned to C, so that its length does not exceed b. LGREEDY assigns an arriving point to a cluster into which it fits, if such a cluster exists (ties are broken arbitrarily). If no such cluster exists, a new cluster is opened for the new point.

Proposition 4 *The competitive ratio of* LGREEDY *for any fixed value of* b*, such that* $1 < b < 2$*, is at least 2.*

Proof Consider the following input. Let M be a large enough integer. For $i = 1, ..., M$ the points $2i$, $2i + b$ are requested (the points arrive from left to right). Next, the points $2i + 1 + \frac{b}{2}$ are requested for $i = 0, \ldots, M$. These points require $M + 1$ additional clusters. Therefore, LGREEDY uses $2M + 1$ clusters. An optimal offline solution opens the clusters $[2i + 1, 2i + 2]$ for $i = 0, \ldots, M$, thus $\text{OPT} = M + 1$. As M grows, the competitive ratio tends to 2. □

We next define the algorithm LONG GRID (LGRID) as follows. Create a grid which consists of all points of the form $i \cdot b$ for all (possibly negative, or zero) integer values of i. If a new point p does not fit into a previously existing cluster, an interval of the form $(j \cdot b, (j+1)b]$ is determined, such that the point p belongs to it, and a cluster is opened at $[j \cdot b, (j + 1)b]$.

Proposition 5 *The competitive ratio of* LGRID *for any fixed value of* b*, such that* $1 < b < 2$*, is at least 2.*

Proof Consider the following input. Let M be a large enough integer. For $i = 0, ..., M - 1$ the points $2ib - 1/2$ and $2ib + 1/2$ are requested. LGRID opens a cluster for each point, so these points require $2M$ clusters. An optimal offline solution opens the clusters $[2ib - 1/2, 2ib + 1/2]$ for $i = 0, \ldots, M - 1$, thus $\text{OPT} = M$. □

Another generalization of a known algorithm is LONG CENTER (LCENTER). If a new point p does not fit into a previously existing cluster, this algorithm opens a cluster of length b centered at p.

Proposition 6 *The competitive ratio of* LCENTER *for any fixed value of* b*, such that* $1 < b < 2$*, is at least 2.*

Proof Consider the following input. Let M be a large enough integer. For $i = 1, ..., 2M$ the point i is requested. LCENTER opens a cluster centered at each point, so the only integer point contained in each cluster is its center point. Therefore, these points require $2M$ clusters. An optimal offline solution opens the clusters $[2i - 1, 2i]$ for $i = 1, \ldots, M$, thus $\text{OPT} = M$. □

Since previously known algorithms do not succeed to make use of the extended length of clusters, we design an algorithm for $b \in [3/2, 1)$ which fixes some clusters similarly to the algorithm of [12]. Unlike that algorithm, the algorithm makes use of extending clusters, but only in particular cases, and otherwise it is based on Greedy with clusters of length 1.

5.3 An algorithm with resource augmentation for $b \in [\frac{3}{2}, 2)$

The main idea of this algorithm is simple: we take advantage of the resource augmentation by not having to create new clusters between two clusters that are relatively close together (Step 1) and we do our best to avoid the situation where three clusters intersect a common interval of length 1 (Step 2).

We first discuss a general property of algorithms of this type. An algorithm is called *thrifty* if it never opens a new cluster for a request point which fits in an existing cluster without extending its length beyond 1.

Lemma 1 *For a thrifty algorithm, there can be no interval of length 1 which completely contains two clusters.*

Proof Assume that two clusters that are defined by a thrifty algorithm are contained in an interval of length 1. Let A and B be two such consecutive clusters (i.e., such that there is no cluster between them).

Without loss of generality, denote by A the cluster that is defined earlier by the algorithm. Let b be the first request point in B. We consider the time at which b is assigned to a cluster. Since the point b fits in A without extending its length above 1, a thrifty algorithm cannot create B at this time, which leads to a contradiction. □

The algorithm is defined as follows. A cluster is called *single* unless it has been *joined* with another cluster in Step 1 or in Step 2. Let p be the new arriving point.

1. If p appears between two existing single clusters A and B, and the minimum distance between two points from A and B is at most 1, and p cannot be assigned to either cluster while keeping the lengths at most 1, we extend both clusters to the point that is in the middle of the gap between them. Now p is contained in (at least) one of the clusters. Assign p to a cluster it is contained in. We now call A and B *joined* clusters.

2. If p appears between two existing single clusters A and B, and p can be assigned to both of them while keeping the lengths at most 1, there are three cases.

   (a) If there exist two additional clusters C and D that are at most 1 away from p, join A and B at point p. Assign p arbitrarily to A or B.

   (b) If there exists one additional cluster C such that $d(p, C) \leq 1$, assign p to the cluster among A and B which is closer to C.

   (c) Else, assign p arbitrarily to A or B.

3. If p appears between a single cluster A and a joined cluster B, B was joined in Step 2, $d(p, q) \leq 1$ for all request points $q \in A$ and $d(p, q) \leq 1$ for all points $q \in B$, then we assign p to B unless this brings B within a distance of 1 of another cluster C; in that case, assign p to A.

4. If p appears between two joined clusters and can be assigned to both of them while keeping their lengths at most $3/2$, assign p arbitrarily to one of them.

5. If p can be assigned to only one existing cluster while keeping its length at most 1, do so.

6. If p is not assigned to a cluster by the previous rules, open a new cluster for p.

For an illustration, see Figure 1. A pair of clusters that is joined in Step 1 is called a *long pair*, other joined pairs are called *short pairs*. It can be seen that our algorithm is thrifty. Thus, it follows from Lemma 1 that if Case 2(a) occurs, clusters C and D must indeed be at different sides of p. Note that by the definition of the algorithm, clusters never overlap.

Note that Lemma 1 holds even if there are joined clusters nearby. Specifically, the lemma shows that for two single clusters A and B that both contain only one request point, we have $d(A, B) > 1$.

5.4 Analysis

A pair of clusters are called *consecutive* if there is no cluster that is located between them. In the following, we will repeatedly discuss sets of consecutive clusters $C_1, C_2, \ldots$. In such cases, denote the leftmost request point contained in C_i by ℓ_i and the rightmost request point by r_i. We now consider a fixed optimal offline algorithm. We call the clusters used by this algorithm "optimal clusters". The clusters used by our algorithm

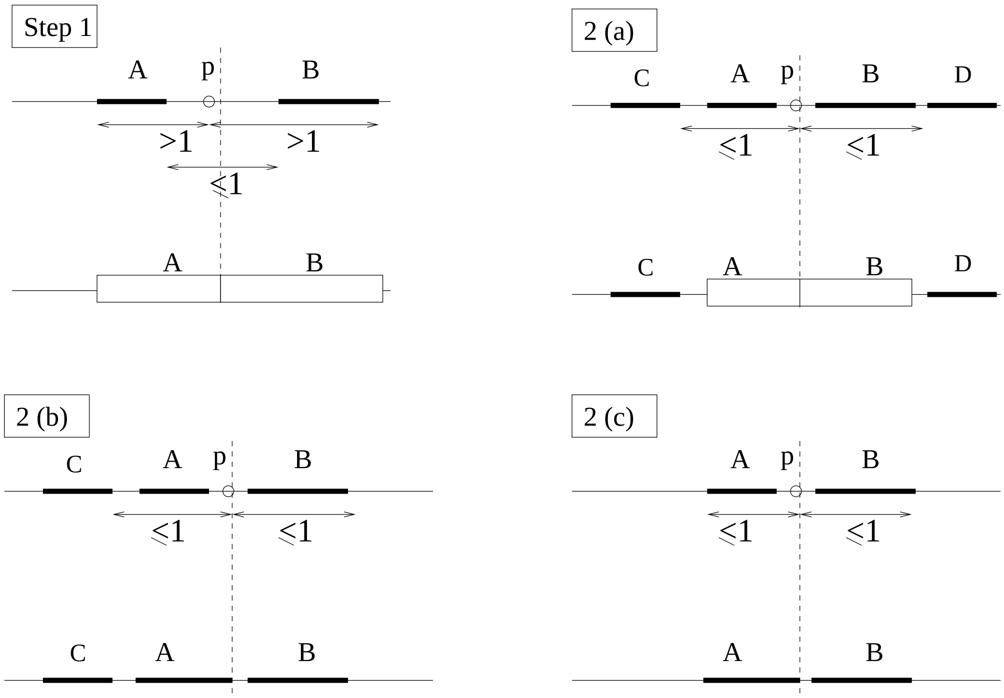

Figure 1: Creation of a joined pair in Step 1 and 2(a), and other assignments

are called "online clusters". We say that an optimal cluster *connects* two online clusters if it intersects both of them.

As noted in [5], it is trivial to provide an optimal solution for a given input offline: starting from the left, repeatedly define a cluster of length 1 that has as its left endpoint the leftmost unserved point. It can be seen that in this solution, no two clusters overlap (not even at their endpoints). We will compare our algorithm to this solution.

Lemma 2 *There can be no interval of length 1 which intersects with three different online single clusters.*

Proof Suppose there is such an interval which contains requests from the single clusters C_1, C_2 and C_3 (from left to right). Note that these three clusters are consecutive clusters, since otherwise, if there is a cluster C_4 between C_1 and C_2 or between C_2 and C_3, then C_2 and C_4 are fully contained in an interval of length 1 in contradiction to Lemma 1.

The assumption implies $d(r_1, \ell_3) \leq 1$. Let q be the oldest request point in C_2. There are two cases. If q is newer than r_1 and ℓ_3, C_1 and C_3 would have been joined together when q arrived in Step 1 or Step 2, or q would have been assigned to one of them in Step 2 or 5.

Otherwise, without loss of generality, let r_1 be newer than ℓ_3 (and q). When r_1 arrives, it could be assigned to C_2, since r_1 is less than 1 away from the furthest point in C_2. If our algorithm does not do this, it must be because there was a second possible cluster to assign r_1 to (Step 2). However, in this case, C_1 and C_2 end up joined (Step 2(a)) or r_1 gets assigned to C_2 because C_3 is less than 1 away from r_1 (Step 2(b)). □

Definition 1 *A* group *of online clusters is a maximal set of consecutive clusters such that each two successive clusters are 'connected' by an optimal cluster.*

That is, if $C_1, \ldots, C_m$ (numbered from left to right) are consecutive online clusters that form a group, there is an optimal cluster which contains both r_i and ℓ_{i+1} for $i = 1, \ldots, m-1$. (These optimal clusters are not necessarily all distinct.)

If there is more than one group, for each group we have that the leftmost point of the leftmost online cluster is not to the right of the leftmost point of the leftmost optimal cluster by the way we construct our optimal solution. Two clusters that are joined together are not necessarily in the same group.

Lemma 3 *For $m \geq 3$, at least $m-1$ optimal clusters are needed to serve all the request points in m consecutive single clusters that are in the same group.*

Proof If at most $m-2$ optimal clusters serve the requests in m consecutive single clusters, then there is either an optimal cluster which serves *all* requests of at least two single clusters (impossible by Lemma 1) or, if there is no such cluster, an optimal cluster that serves some requests from at least three online clusters by the pigeonhole principle. This is impossible by Lemma 2. □

Lemma 4 *It requires at least three optimal clusters to serve all requests from a long pair, and at least two optimal clusters to serve all requests from a short pair. A long pair has at least one optimal cluster that is fully contained in the union of the pair of online clusters.*

Proof In Step 1, p is more than one away from the furthest endpoints of both A and B, which are both request points. This gives three points, each one of which must be in a different optimal cluster, which implies that at least three optimal clusters are required to serve all the points in these two clusters. The cluster that serves p is completely contained within the interval spanned by A and B.

In Step 2(a), the clusters A and B are not contained in an interval of length 1 by Lemma 1. Since their endpoints are request points, the lemma follows. □

Lemma 5 *Consider a cluster J in a short pair, that is joined to a cluster on its left. The first cluster on its right, say C, already existed when J was joined. Just before J was joined, J and C were not contained in an interval of length 1.*

Proof When J was joined in Step 2, there was a cluster C' next to it that J does not get joined to. C' is at most 1 away from the point p that caused J to be joined. Therefore, a future request point p' between C' and J could be assigned to J, since it is less than 1 away from p which is the left endpoint of J. Since our algorithm is thrifty, it does not open a new cluster for p'. Therefore $C' = C$. The second statement follows from Lemma 1. □

Lemma 6 *There can be no optimal cluster X which serves requests from two single clusters C, E and a joined cluster J, unless J was joined in Step 1.*

Proof Assume by contradiction that X exists. By Lemma 1, these are three consecutive clusters.

We first prove that J is either to the left or to the right of the clusters C and E. Since online clusters do not overlap, if this claim does not hold, then the cluster to which J is joined, J', as well as J, are between C and E. Since the distance between C and E is at most 1, we conclude that J and J' are contained in an interval of length 1, which contradicts Lemma 1.

Without loss of generality, we assume that the order of the three clusters from left to right is J, C, E. J was joined to a cluster J' in Step 2, so C must have existed when J was joined by Lemma 5.

While E could have also existed already at this point, it could not yet have been within distance 1 of J, since otherwise we would have three single clusters all intersecting an interval of length 1, contradicting Lemma 2. Any request point p' that appears between J and C can be assigned to either J or C without increasing the length of a cluster above 1. This holds for J since C is of distance at most 1 from point p which is the left endpoint of J. This holds for C as the distance from J to E (or to the future left endpoint of E, up to which C would never be extended) is at most 1. Therefore, the conditions of Step 3 hold, and hence point p' is assigned in Step 3 (since C remains single throughout the process considered here). The point p' is not assigned to J if this brings J within 1 of E.

Note that if a point p' appears between C and E, it can be assigned to C without increasing the length of C above 1. Therefore such a point p' is not assigned in Step 1. Such a point p' is not assigned to a cluster in Step 2(a) since otherwise C and E would not be single. We have that p' is of distance of at most 1 from J, so if p' can be assigned to E, it is assigned to C in Step 2(b). Otherwise, p' is assigned to C in Step 5.

Consider the leftmost point p'' in E. p'' was not in E at the time when J was still single due to Lemma 2 (no matter whether E existed at that time or not). By the above argument, since p'' is a new point between C and E, it must be inserted to C and not to E. □

Lemma 7 *Consider $m \geq 2$ consecutive clusters, where the first and the last cluster are part of a short pair, and the other $m-2$ clusters are single. If all these clusters are in one group, it takes at least $m-1$ optimal clusters to serve all the requests of these clusters. If $m=2$, two optimal clusters are needed.*

Proof If $m = 2$, then if the two clusters are joined together as a short pair, we are done by Lemma 4. If they are not joined together, let the left pair be A and B and the right pair be D and E. The names are from left to right and we are interested in the optimal cost to serve the requests in B and D. Suppose A and B were joined first. When they were joined, since this happened in Step 2, D already existed, but was not contained in an interval of length 1 together with B by Lemma 5. Since all their endpoints were request points before B was joined, this proves the lemma.

If $m = 3$, then just before D is joined to E (using the same notations for the joined clusters), the single cluster C between B and D already exists by Lemma 5, and we can apply Lemma 6 to B and the single clusters C and D.

Now consider $m > 3$. Denote the single clusters by $C_1, \ldots, C_{m-2}$. Again let A and B be the short pair that was joined first. We know by Lemma 1 (if $m = 4$) and Lemma 3 that at least $m-3$ optimal clusters are needed to serve the $m-2$ single clusters. Since the single clusters are all in one group, there are in fact $m-3$ optimal clusters which serve requests from two single clusters, because this is the number of gaps between the single clusters, and no optimal cluster can serve requests from three single clusters by Lemma 2. If any of these $m-3$ clusters also serves a request from B or D, then since no two optimal clusters overlap, it must be one of the outermost optimal clusters, contradicting Lemma 6. This proves the existence of two additional optimal clusters, proving the lemma. □

Theorem 5 *This algorithm has a competitive ratio of 5/3.*

Proof We first show an upper bound of $5/3$. We partition the real line into intervals. The endpoints of the intervals are shared endpoints of joined pairs. If there are two consecutive clusters that are not in the same group, we also define an endpoint between them if there was not one already. There are two half-bounded intervals on either side, and each group may begin and end with a single cluster.

We consider the competitive ratio of our algorithm on each of these intervals separately. Note that as defined, each interval is entirely contained within one group. If there are no joined pairs within an interval, we are done by Lemmas 1 and 3.

Next we consider intervals that have joined pairs at both ends. For long pairs we assign to both clusters of the pair 3/2 optimal clusters for the calculations, using Lemma 4.

For our analysis, it is irrelevant where exactly the optimal clusters are that serve the requests of joined pairs. This leaves us with only a few cases, depending on the types of the pairs that form the endpoints of the current interval, and how many single clusters are between them.

First of all, if there are short pairs at both ends, we are done immediately by Lemma 7. If there is a long pair at at least one end, then some requests from *two* single clusters at that end might be served by the same optimal cluster. For additional single clusters after that, we have Lemma 3. Regarding an end with a short pair, we know that when J (the half of the short pair which is inside the current interval) was joined, the single cluster C immediately next to it already existed, and J and C were not contained in an interval of length 1 by Lemma 5. Moreover, there is no optimal cluster which serves requests from J, C and a third single cluster by Lemma 6. Therefore, for the purposes of this analysis, J acts like a single interval.

Overall, we find the following results. The column Cost indicates upper bounds on the cost.

Left pair	Right pair	Number of single clusters	Cost	Optimal cost	Proof
short	short	at most 1	3	2	Lemma 7
		$m \geq 2$	$m+2$	$m+1$	Lemma 7
short	long	0	2	3/2	Assignment
		1 or 2	4	$5/2$	Lemma 5
		$m \geq 3$	$m+2$	$m+1/2$	Lemmas 3, 6
long	long	0 or 1	3	2	Note 1
		2 or 3	5	3	Note 2
		$m \geq 4$	$m+2$	m	Note 3

Note 1: we assign $3/2$ from both ends to this interval, but we may count (at most) one optimal cluster twice, thus we assign at least 2 in total.

Note 2: Now there is no double counting by Lemma 1.

Note 3: The m single clusters require at least $m-1$ optimal clusters by Lemma 3. Each long pair contributes an additional $1/2$ cluster that does not serve points of single clusters (Lemma 4).

Finally we consider intervals that do not have a joined pair at both ends, but do not contain only single clusters. If an interval contains only one (joined) cluster, this just adds 1 to the online and optimal cost of the interval that contains the cluster with which it is joined, improving the competitive ratio on that interval.

If an interval contains more than one cluster, then w.l.o.g. let the rightmost cluster J be joined. Consider the offline cost to serve all requests in the group up to and including J. If J is part of a long pair, we find a ratio of $2/1.5 = 4/3$ if there is only one single cluster, and the ratio decreases due to Lemmas 1 and 3 if there are more. If J is in a short pair, the ratio is $3/2$ for one or two single clusters and decreases for more in the same way. This completes the proof of the upper bound.

We now show a matching lower bound for this algorithm. Let i run from 1 to M for some large value M. First we give requests at the points $6i, 6i+1, 6i+2, 6i+3$ for $i = 1, \ldots, M$. The algorithm creates $2M$ clusters. Then we give requests at the points $6i + 3/2$ for $i = 1, \ldots, M$, this causes Step 1 to be executed M times. No new clusters are created in this phase.

We then give requests at points $6i + 4, 6i + 5$ for $i = 1, \ldots, M$. The algorithm creates M additional clusters. Finally we give requests at points $6i + 10/3, 6i + 17/3$ for $i = 1, \ldots, M$. This generates $2M$ additional clusters for a total of $5M$ clusters.

It is easy to see that all request points can be served by the set of clusters $[i, i+1]$ for all odd i between 5 and $6M + 5$. This is a set of $6M/2 + 1 = 3M + 1$ clusters. Thus for large M, the ratio tends to $5/3$. □

6 Clustering with temporary points

In this variant of the problem, points arrive and depart online. Every event is either an arrival or a departure of a point. At every time, a cluster can serve points that belong to an interval of length at most 1. The points that need to be taken into account at every time are those that already arrived and did not depart yet. The momentary cost of an algorithm (at a given time) is the number of clusters that are used to cover at least one point at this time. The cost of an algorithm is its maximum momentary cost over time. Each arriving point must be assigned to a cluster upon arrival and remains assigned to this cluster until its departure.

We can show a tight bound of 2 for this problem. The algorithm we use is GRID, where the algorithm closes clusters which do not have points assigned to them due to departure of points.

Theorem 6 GRID *has competitive ratio 2 for clustering with temporary points, and this is best possible.*

Proof To prove the upper bound, we show in the sequel that at every time, the momentary cost of GRID is at most twice the momentary cost of OPT. This would imply a ratio of at most 2 between the costs of the two algorithms. Consider any time t during processing, and let the set of points existing at this time (live points) be J_t. Let x be the number of intervals of the form $(a, a+1]$ that contain at least one live point. At this time, GRID has x open clusters, its momentary cost is x. However, an optimal algorithm can serve by each one of its clusters points from at most two clusters, thus the momentary cost of OPT is at least $\frac{x}{2}$, as we claimed.

To prove a lower bound, we construct the sequence in phases. In each phase a set of three points arrives, and then one point departs. Specifically, in phase i, the three points $4i, 4i+1, 4i+2$ arrive. These points are too far from any previous point and thus new clusters must be opened for them. The algorithm must use exactly two different clusters, A and B for the points $4i$ and $4i+2$. If the point $4i+1$ is in the same cluster as $4i$, then the point $4i$ departs, and otherwise $4i+2$ departs. An optimal algorithm uses one cluster for the point that departs and another cluster for the points that remain.

For any $i = 1, \ldots$, the momentary cost of OPT after the arrival of the points of phase i, and before the departure of one point of this phase, is $i+1$, since it opens two clusters for the new points of this phase. However, after the departure of one point, the momentary cost of OPT becomes i.

The algorithm will use at least $2i$ clusters after i phases, since the points of phase i enforce it to use at least two new clusters, no matter if a point of this phase has departed already. Thus the lower bound will follow from applying M phases for an arbitrarily large M, which gives a ratio of at least $\frac{2M}{M+1} \to 2$. □

7 Concluding remarks

In this paper we study variants of the online clustering problem. For most of these variants we present tight bounds on the competitive ratio of any online algorithms. We note that one can study these variants of the online covering problem as well. However, most of our results hold also for that model as discussed in the following. The lower bounds (for all our variants) clearly hold also for the version of the online covering problem. As for the upper bounds, we note that the algorithms based on GRID, or on CENTER fix the position of the cluster once it is opened. Therefore, we conclude that the following upper bounds hold: the upper bound of 3 for online covering with rejection (using the analysis of RGRID), the upper bound of 2 for online max covering (using the analysis of GRID for the online max clustering problem), the upper bound of 2 for online cardinality constrained covering problem (using the analysis of CGRID), an upper bound of 1 for the online covering with resource augmentation of $b \geq 2$ (using the analysis of CENTER), and an upper bound of 2 for online covering with temporary points (using the analysis of GRID for the online clustering with temporary points). For most of these variants this gives tight bounds as well.

References

[1] James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. *Journal of the ACM*, 44:486–504, 1997.

[2] Yossi Azar and Leah Epstein. On-line load balancing of temporary tasks on identical machines. *SIAM Journal on Discrete Mathematics*, 18(2):347–352, 2004.

[3] Luitpold Babel, Bo Chen, Hans Kellerer, and Vladimir Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1-3):238–251, 2004.

[4] Yair Bartal, Stefano Leonardi, Alberto Marchetti-Spaccamela, Jiri Sgall, and Leen Stougie. Multiprocessor scheduling with rejection. *SIAM Journal on Discrete Mathematics*, 13:64–78, 2000.

[5] Timothy M. Chan and Hamid Zarrabi-Zadeh. A randomized algorithm for onine unit clustering. In *Proc. 4th Workshop on Approximation and Online Algorithms (WAOA 2006)*, volume 4368 of *Lecture Notes in Comput. Sci.*, pages 121–131. Springer, 2006.

[6] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

[7] György Dósa and Yong He. Bin packing problems with rejection penalties and their dual problems. *Information and Computation*, 204(5):795–815, 2006.

[8] Leah Epstein. Bin packing with rejection revisited. In *Proc. of the 4th International Workshop on Approximation and Online Algorithms (WAOA 2006)*, volume 4368 of *Lecture Notes in Comput. Sci.*, pages 146–159. Springer, 2006. Also in Algorithmica, to appear.

[9] Leah Epstein. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics*, 20(4):1015–1030, 2006.

[10] Leah Epstein and Asaf Levin. On the max coloring problem. In *Proc. 5th Workshop on Approximation and Online Algorithms (WAOA 2007)*, volume 4927 of *Lecture Notes in Comput. Sci.*, pages 142–155. Springer, 2008.

[11] Leah Epstein, Asaf Levin, and Gerhard J. Woeginger. Graph coloring with rejection. In *Proc. of the 14th Annual European Symposium on Algorithms (ESA2006)*, volume 4168 of *Lecture Notes in Comput. Sci.*, pages 364–375. Springer, 2006.

[12] Leah Epstein and Rob van Stee. On the online unit clustering problem. In *Proc. 5th Workshop on Approximation and Online Algorithms (WAOA 2007)*, volume 4927 of *Lecture Notes in Comput. Sci.*, pages 193–206. Springer, 2008.

[13] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[14] Sudipto Guha, Refael Hassin, Samir Khuller, and Einat Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257–270, 2003.

[15] András Gyárfás and Jenö Lehel. On-line and first-fit colorings of graphs. *J. Graph Theory*, 12:217–227, 1988.

[16] Edward G. Coffman Jr., Michael R. Garey, and David S. Johnson. Dynamic bin packing. *SIAM Journal on Computing*, 12:227–258, 1983.

[17] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.

[18] Hal A. Kierstead. Coloring graphs on-line. In Amos Fiat and Gergard J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 281–305. Springer, 1998.

[19] Hal A. Kierstead and William T. Trotter. An extremal problem in recursive combinatorics. *Congr. Numer.*, 33:143–153, 1981.

[20] László Lovász, Michael E. Saks, and W. T. Trotter. An on-line graph coloring algorithm with sublinear performance ratio. *Discrete Math.*, 75:319–325, 1989.

[21] Mohammad Mahdian, Yingyu Ye, and Jiawei Zhang. A 2-approximation algorithm for the soft-capacitated facility location problem. In *Proc. of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2003)*, volume 2764 of *Lecture Notes in Comput. Sci.*, pages 129–140. Springer, 2003.

[22] Sriram V. Pemmaraju, Rajiv Raman, and Kasturi R. Varadarajan. Buffer minimization using max-coloring. In *Proc. of 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 562–571, 2004.

[23] Hamid Zarrabi-Zadeh and Timothy M. Chan. An improved algorithm for online unit clustering. In *Proc. of the 13th Annual International Computing and Combinatorics Conference (COCOON 2007)*, volume 4598 of *Lecture Notes in Comput. Sci.*, pages 383–393. Springer, 2007.