Better Bounds on Online Unit Clustering[☆]

Martin R. Ehmsen[a], Kim S. Larsen[a,1,*]

[a]*Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark*

Abstract

Unit Clustering is the problem of dividing a set of points from a metric space into a minimal number of subsets such that the points in each subset are enclosable by a unit ball. We continue the work, recently initiated by Chan and Zarrabi-Zadeh, on determining the competitive ratio of the online version of this problem. For the one-dimensional case, we develop a deterministic algorithm, improving the best known upper bound of 7/4 by Epstein and van Stee to 5/3. This narrows the gap to the best known lower bound of 8/5 to only 1/15. Our algorithm automatically leads to improvements in all higher dimensions as well. Finally, we strengthen the deterministic lower bound in two dimensions and higher from 2 to 13/6.

Keywords: Clustering, Online algorithms, Competitive analysis

1. Introduction

Unit Clustering is the problem of dividing a set of points from a metric space into a minimal number of subsets such that the points in each subset are enclosable by a unit ball. The subsets are also referred to as *clusters*. Clustering algorithms have applications in for instance information retrieval, data mining, and facility location. However, our primary focus is to come closer to a full understanding of one of the most fundamental clusterings problems which has inspired additional work on many variants of the model [4].

In the online version, we must treat a sequence of points one at a time, i.e., a point must be treated without knowing the remaining sequence of points to come, or even the length of this future sequence. When treating a point, we always have the option of opening a new cluster for the point, i.e., increasing the number of subsets by one new subset containing only the most recent point. We may also have the option of including the new point in an existing cluster, provided that the new point together with all other points already assigned to that cluster are still enclosable in a unit ball. Each point must be assigned to exactly one cluster and this decision is irrevocable, i.e., we cannot at a later stage move a point from one cluster to another.

[☆]Supported in part by the Danish Natural Science Research Council. A preliminary abstract appeared in the Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory, Lecture Notes in Computer Science, Springer, 2010.

[*]Corresponding author.

Email addresses: `ehmsen@imada.sdu.dk` (Martin R. Ehmsen), `kslarsen@imada.sdu.dk` (Kim S. Larsen)

[1]Part of this work was carried out while this author was visiting the University of California, Irvine and the University of Waterloo.

Note that this problem is different from online covering [3]. In online covering, when a cluster is opened, a unit diameter is placed in a fixed location. In online clustering, points are assigned to a cluster, but the exact location of the cluster is not fixed. Another way to view this is that clusters open with size zero and then gradually have their sizes increased as points are assigned to the clusters.

To measure the quality of an online algorithm for this problem, we follow previous work on the same topic and use competitive analysis [6, 8, 7], which is defined as follows. Let I denote an input sequence and let A(I) denote the cost of processing I using the algorithm A. We are interested in the ratio $\mathrm{A(I)}/\,\mathrm{OPT(I)}$ for a given online algorithm A, where OPT(I) denotes the cost of an optimal algorithm that knows the entire sequence in advance, i.e., OPT(I) is a lower bound on the cost of any algorithm, and therefore $\mathrm{A(I)}/\,\mathrm{OPT(I)}$ is an upper bound on how poorly A can perform in comparison with any other algorithm.

Since we want to provide worst-case guarantees, we work on limiting the worst ratio taken over all input sequences. In order to avoid getting unrealistic results due to start-up costs for short sequences, one usually allows for an additive constant α, which is fixed and independent of I, and define an algorithm A to be *c-competitive* if for all input sequences I, $\mathrm{A(I)} \leq c \cdot \mathrm{OPT(I)} + \alpha$. The infimum over all such c is referred to as the *competitive ratio* of A and this number exactly characterizes its worst-case performance with regards to competitive analysis. If an algorithm can be proven to be c-competitive using $\alpha = 0$, then it is said to be strictly c-competitive. All results proved in this paper are strict in that sense.

In unit clustering, the input sequence I is a sequence of points and the cost of an algorithm is the number of clusters it opens. For the randomized results that we reference, the ratio is defined using the expected cost E[A(I)] instead of the deterministic cost A(I), and the results are with respect to an *oblivious adversary* [1]. This means that the input sequence can be defined to be worst possible for A based on knowing the definition of A (its code), but not the result of running the algorithm (the random bits it uses).

We develop a deterministic algorithm for the one-dimensional Online Unit Clustering problem. The work on this problem was initiated by Chan and Zarrabi-Zadeh [2] and the currently best known bounds are by Epstein and van Stee [5]. An overview of previous results, our results, and lower bounds for 1-dimensional unit cluster is given in Table 1.

Thus, by this new upper bound of $\frac{5}{3}$, we narrow the gap to the best known lower bound of $\frac{8}{5}$ to only $\frac{1}{15}$.

In higher dimensions, using an idea from [2], our 1-dimensional result improves the upper bound of $\frac{7}{8}2^d$ from [5] to $\frac{5}{6}2^d$ in d dimensions. As in the previous work, this is with respect to the L_∞ norm. Thus, the unit balls are really squares, cubes, etc. In one dimension, a 2-competitive algorithm is almost immediate. There are more than one easy formulation of such an algorithm. The greedy algorithm which only opens a new cluster when forced to do so is one of them. Thus, given a c-competitive algorithm for the 1-dimensional case, one can use that algorithm in the first dimension and combine that with an additional factor 2 for each additional dimension. Thus, in dimension $d \geq 2$, a $c2^{d-1}$-competitive algorithm can be designed from the 1-dimensional case, and we obtain an algorithm with competitive ratio $\frac{5}{3}2^{d-1} = \frac{5}{6}2^d$.

Finally, we strengthen the deterministic lower bound in dimension 2 and higher from two, obtained in [5], to $\frac{13}{6}$. The lower bound proof is carried out in dimension 2. The lower bound holds in higher dimensions by simply giving the corresponding points in a 2-dimensional subspace.

Another contribution of this paper is the proof technique we develop to establish the improved upper bound. First, among the possibly many ways of clustering optimally, we select a special one

	Paper	Result	Type
Upper Bounds	[2]	2	deterministic
	[2]	$\frac{15}{8} = 1.875$	randomized
	[9]	$\frac{11}{6} \approx 1.833$	randomized
	[5]	$\frac{7}{4} = 1.75$	deterministic
	This	$\frac{5}{3} \approx 1.667$	deterministic
Lower Bounds	[5]	$\frac{8}{5} = 1.6$	deterministic
	[5]	$\frac{3}{2} = 1.5$	randomized
	[2]	$\frac{3}{2} = 1.5$	deterministic
	[2]	$\frac{4}{3} \approx 1.333$	randomized

Table 1: New and previous results for 1-dimensional unit clustering.

and work up against that. Next, using a "state machine"-like technique, we analyze constant-sized collections of clusters based on some "input conditions" capturing the essentials of the behavior to the left of the collection of clusters and establishing some "output conditions" summarizing the effect of the analyzed collection together with the result to the left of the collection. Then, by establishing invariants that each local analysis must fulfill, a complete analysis then follows directly from the components. We believe that this proof technique could prove useful in establishing improved upper bounds not just for this problem, but for clustering problems in general.

2. The Algorithm

In this section, we start the development of a new algorithm for online unit clustering. We structure the definition of the algorithm in a number of steps, in order to organize the analysis in a way that makes it reasonable to verify. In this section, we define the three most fundamental parts of the overall algorithm. Then we establish properties of the algorithm and complete the analysis. From the analysis it becomes evident that the local properties which are required to obtain the overall result hold in almost all cases. We identify the cases where the properties cannot be established by the algorithm up to this point, and then devise an algorithmic refinement, improving the behavior of the algorithm in those situations. A final cases analysis then completes the treatment.

We structure our algorithm around components consisting of 2–4 clusters that we refer to as a *groups* with various characteristics. We then define behavior inside and outside these groups, i.e., we define the actions taken for new points that falls inside and outside these groups.

We start with terminology and a few definitions of groups and other components.

We are working on the line, and define a *cluster* to be an interval with a maximal length of one. For a cluster C, let r_C denote the right endpoint of C and let l_C denote the left endpoint of C. For two points p_1 and p_2, we let $d(p_1, p_2)$ denote the distance between the two points.

We say that a cluster C can *cover* a point p, if $\max(d(p, r_C), d(p, l_C)) \leq 1$, i.e., if assigning p to C does not violate the restriction on the length of a cluster being at most one unit; see Figure 1. The distance $d(C, D)$ between two clusters C and D is defined as the distance between the two

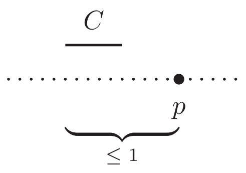


Figure 1: C can cover p.

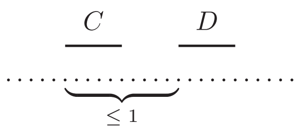


Figure 2: C can reach D.

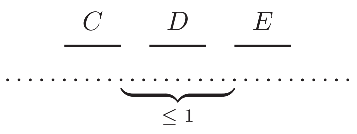


Figure 3: D is underutilized.

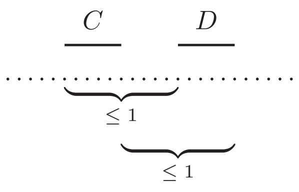


Figure 4: A close pair.

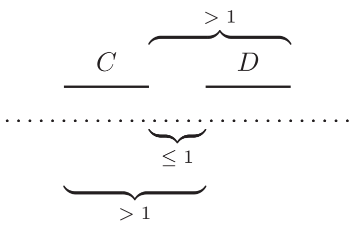


Figure 5: A far pair.

nearest points of C and D, i.e., $d(C, D) = \min(d(r_C, l_D), d(r_D, l_C))$. Similarly, the distance $d(C, p)$ between a cluster C and a point p is defined as $d(C, p) = \min(d(r_C, p), d(l_C, p))$, in other words treating the point p as a cluster of length zero. Note that if, at some point during processing the request sequence, C can cover a point p, then that has been the case since C was created.

Very often we discuss situations where we open a new cluster for a point p. If we do not give that cluster some other name, we let P refer to the cluster opened for p.

A cluster C is said to be able to *reach* another cluster D, if C can cover all points on the line between C and D; see Figure 2. As above, we extend this to points, thinking of these as clusters of length zero. Note that this relation is not symmetric.

If, for some cluster D, there exist a cluster C to the left and a cluster E to the right of D, then D is said to be *underutilized* if $d(r_C, l_E) \leq 1$; see Figure 3. Two clusters C and D are said to be a *close pair* if they can reach each other; see Figure 4. Two clusters C and D are said to be a *far pair* if neither one can reach the other, but $d(r_C, l_D) \leq 1$; see Figure 5.

We now start discussing groups. In the algorithm to be presented, we will ensure that groups satisfy certain specific criteria. However, satisfying these criteria does not define a group. For instance, we do not want overlapping groups, i.e., groups that share clusters. Thus, the definition of a group is a labelling issue, in the sense that when the situation is right, we may decide that a collection of clusters form a group.

We say that three consecutive clusters C, D, and E, none of which belong to another group, form a *potential regular group* (PRG) if $d(C, D) \leq 1$, $d(D, E) \leq 1$, $d(C, E) > 1$, at least one of C and E cannot reach D, and D can reach at least one of C and E; see Figure 6.

We use RG for regular group, i.e., a PGR that we decided to label an RG. An RG initially consists of three clusters. However, as the algorithm progresses, a fourth cluster might be needed at some point. On the other hand, the algorithm we develop ensures that a fifth cluster is never needed. We denote C and E *side* clusters in an RG or a PRG. D and the possible fourth cluster are denoted *middle* clusters. We denote an RG with three clusters a *regular 3-group* (using R_3 as a shorthand) and a regular group with four clusters a *regular 4-group* (using R_4 as a shorthand). A side cluster in an RG or a PRG that cannot reach the middle cluster is called a *far cluster*, e.g., if

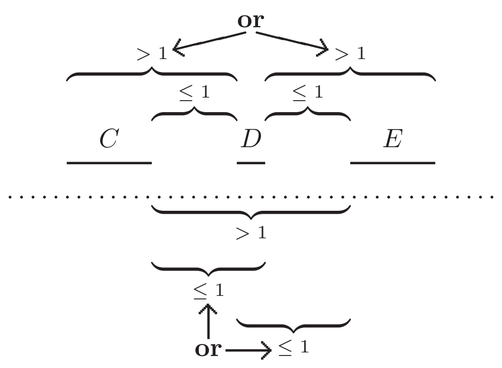


Figure 6: A potential regular group (PRG).

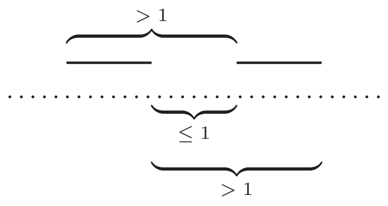


Figure 7: A tight group.

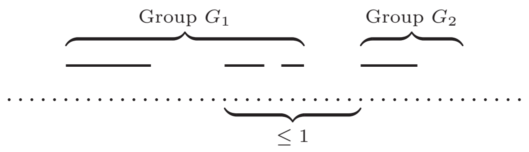


Figure 8: A close regular group G_1.

$d(l_C, l_D) > 1$ in the above, then C is denoted a far cluster. If C is the far cluster in a (potential) RG with clusters C, D, and E, we introduce the shorthand notation $\overline{C}DE$. If, in addition, D and E are known to be contained in a unit interval, we write the R_3 as $\overline{C}(DE)_1$, and as $\overline{C}(DE)_2$ if this is not the case. We use similar notation for the case where the far cluster is to the right in the group.

Some R_3s have undesirable properties and we refer to an R_3 where the middle cluster forms a far pair with one side cluster and a close pair with the other as a *bad* RG. R_3s that are not bad are referred to as *good*.

We will also need a variant that we refer to as a *tight* group which is really just a far pair with a different label; see Figure 7.

A tight group initially consists of two clusters denoted T_2 (or $(CD)_T$ when we need to name the clusters). However, as the algorithm progresses, a third cluster might be needed at some point. A tight group with three clusters is denoted T_3 (or $(CDE)_T$ when we need to name the clusters). The two initial clusters in a tight group are denoted side clusters and the possible third cluster is denoted a middle cluster.

Finally, an RG G_1 is said to be *close* to another group G_2, if a middle cluster from G_1 can reach a cluster in G_2; see Figure 8. Note that if we would replace the two clusters of G_1 that are inside a unit interval by a single cluster that covers all of their points, we would get a far pair (tight group).

We are now ready to define the initial version of the algorithm. Obviously, if a new point falls in an already opened cluster (between its left and right endpoints), then that cluster covers the point. If the new point p falls inside a regular group, then Algorithm 2 handles the point, if p falls inside a tight group, then Algorithm 3 handles the point, and otherwise we let Algorithm 1 handle the point.

Algorithm 1 Main

```
Require: $p$ falls outside any group
 1. if $p$ can be covered by a group cluster then
 2.     Cover $p$ by that cluster, avoiding the creation of close RGs if possible and using an under-
        utilized cluster if possible
 3. else if covering $p$ with some cluster creates a new good PRG then
 4.     Cover $p$ by the rightmost possible cluster that allows a good RG to be created and create
        the leftmost possible new good RG
 5. else if opening a new cluster $P$ for $p$ creates a new good PRG then
 6.     if any new good PRG would be close to another group then
 7.         Cover $p$ with the cluster of the PRG that is contained in a unit interval with $p$, and
            create a $T_2$
 8.     else
 9.         Open a new cluster for $p$ and create the leftmost possible new good RG that is not
            close to another group
10. else if $p$ can be covered by a cluster then
11.     Cover $p$ by a cluster, avoiding the creation of a close pair if possible
12. else
13.     Open a new cluster for $p$
```

Algorithm 2 Regular Group (RG)

```
Require: $p$ falls inside a regular group
 1. if a side cluster can cover $p$ without underutilizing a middle cluster then
 2.     Cover $p$ by that side cluster
 3. else if $p$ can be covered by a middle cluster which will still afterwards be able to reach a side
    cluster then
 4.     Cover $p$ by that middle cluster
 5. else
 6.     Open a new middle cluster for $p$
```

Algorithm 3 Tight Group (T_2 or T_3)

```
Require: $p$ falls inside a tight group
 1. if $p$ can be covered by a side cluster then
 2.     Cover $p$ by that side cluster
 3. else if there is a middle cluster then
 4.     Cover $p$ by the middle cluster
 5. else
 6.     Open a new middle cluster for $p$
```

3. Analysis

In the analysis, our general strategy is to analyze constructions at the time of creation. We are free to analyze the constructions at any time as long as the number of clusters in the constructions remain constant (opening new clusters generally lead to new constructions, just as opening a new cluster in an R_3 results in an R_4). We simply choose the time of creation out of convenience. For instance, when a group is created, the conditions in Figure 6 are fulfilled. We use that information to deduce what can happen with possible future points that might come in that area. Later, for example, the left endpoints of C and D and the right endpoints of D and E, may no longer be more than one unit apart, but this is unproblematic since, in all cases, we analyze the full future behavior of the group.

We first establish some basic properties of the algorithm.

Lemma 1. After each point has been processed, there exist no good PRG.

Proof It follows by induction on the number of points given. First recall that none of the clusters of a PRG are contained in other groups (by definition). If the action taken by the algorithm creates a PRG, then either that group is created, another RG is created which overlaps with it, or a T_2 is created. □

Lemma 2. If two clusters are contained in a unit interval, then they are both part of the same RG.

Proof Assume for the sake of contradiction that C_1 and C_2 are two clusters contained in a unit interval, and that they are not part of the same RG. Assume without loss of generality that C_1 was opened before C_2. Consider the point p on which C_2 was opened. Thus, C_1 can cover p and p must be handled by Algorithm 1.

By Algorithm 1, C_2 can either be opened in Line 9 or in Line 13. We consider both possibilities and show that both lead to a contradiction, which will prove the lemma.

First, assume C_2 was opened in Line 9. Since, by assumption, C_1 and C_2 are not part of the same RG, there must exist two other clusters, say C_3 and C_4, such that a new cluster for p can create a good RG together with C_3 and C_4.

It follows that if opening C_2 for p can create a good RG $C_2C_3C_4$, then if C_1 covers p, then $C_1C_3C_4$ would also be a good RG. Hence, by Line 4 in Algorithm 1, C_1 would cover p and we reach a contradiction.

Next, assume C_2 was opened in Line 13 in Algorithm 1. Since, C_1 can cover p, we immediately reach a contradiction, since p would have been covered in Line 11. □

Lemma 3. Algorithm 1 is well-defined.

Proof The only part which is not obviously well-defined is Line 7 in Algorithm 1. However, from the definition of a close RG, it has two clusters contained in a unit interval. By Lemma 2, the new cluster considered in Line 5 must be one of them in this case. Thus, the other one of them (which already exists) can cover p instead and create a tight group. □

We now establish a number of properties regarding how groups are created and how various constructs can interact.

Lemma 4. If C and D are two clusters, neither of which are part of any group, then C and D are not a close pair.

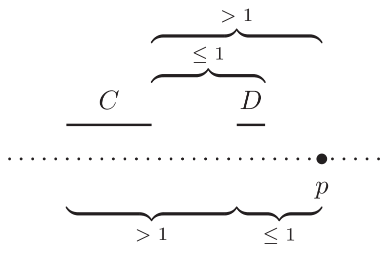


Figure 9: The possible creation of a far pair for $p = r_D$.

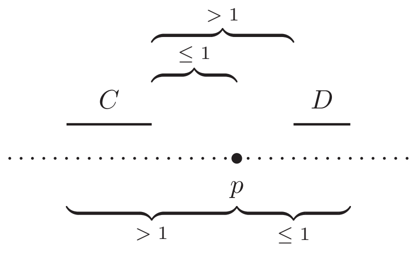


Figure 10: The possible creation of a far pair for $p = l_D$.

Proof Assume for the sake of contradiction that C and D are a close pair, and consider the point p on which they became a close pair. Neither C nor D can be opened on p, since then C and D would be contained in a unit interval and by Lemma 2, they would be part of the same RG. Hence, either C or D covers p, and since they are not contained in a unit interval before p was given, p must be a point between C and D. It follows that both C and D can cover p, and since neither C nor D are part of a group after p has been handled, p must be handled by Line 11 in Algorithm 1. Since C and D were not a close pair before p was given and since they both can cover p, one of C or D can cover p without them becoming a close pair. Hence, we reach a contradiction by Line 11. □

Lemma 5. If C and D are two clusters, neither of which are part of any group, then C and D are not a far pair.

Proof Assume for the sake of contradiction that C and D are a far pair, and that they are not part of any group.

Consider the point p on which C and D became a far pair. Either C or D covers p. Without loss of generality assume it is D. Then $p = r_D$ (Figure 9) or $p = l_D$ (Figure 10).

In both cases, the inequalities follow from the facts that D could cover p, that C and D did not form a far pair before p was given, and that C and D would become a far pair if D covered p. If one opened a new cluster for p, the three clusters would be a good PRG. Hence, by Algorithm 1, D must be part of a group after p has been processed (either D covers p in Line 4 or in Line 7, or a new cluster is opened for p in Line 9), a contradiction. □

Lemma 6. A PRG cannot be created by a cluster covering a point outside this PRG.

Proof This follows almost directly from Lemma 4. Assume for the sake of contradiction that a PRG is created by a cluster covering an outside point p. Since the three clusters did not form a PRG before p was given, it follows from the definition of an RG that they lacked a far cluster. Consider such a scenario; see Figure 11. It can be seen that two of them are a close pair, which contradicts Lemma 4. □

Lemma 7. During the execution of the algorithm, all RGs remain good.

Proof By Algorithm 1, no RG is created bad. In order for a good RG to become bad, it must cover a point p that falls inside of the group: covering an outside point cannot make a pair of the RG close; it can make a pair far, but if the RG is then bad, there was no far cluster in it before, a

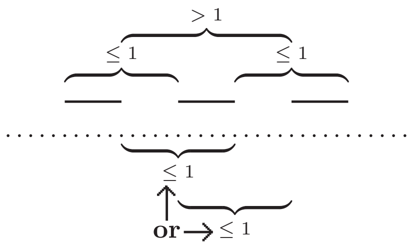


Figure 11: Almost a PRG, except for the lack of a far cluster.

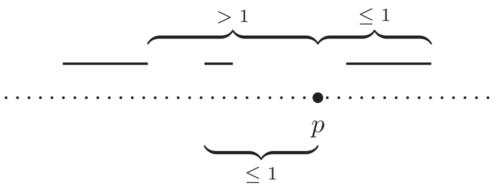


Figure 12: If the middle cluster covers p, the regular group becomes bad.

contradiction. In order for the middle cluster D to become a far pair with one side cluster and a close pair with the other side cluster, it has to be D that covers p (see Figure 12), since if a side cluster covers p, the reachability relationships do not change, unless possibly if it is a far cluster, but then it could only make a bad RG good. Thus, assume to the contrary that D covers p and makes the group bad.

Since the group becomes bad, the side cluster on the side where p comes forms a close pair with D after D covers p, and so it can reach the middle cluster after D covers p. Thus, that side cluster could also have covered p. Doing so would not underutilize D, so by Line 2 in Algorithm 2, the side cluster covers p, which is a contradiction. □

Lemma 8. When the fourth cluster is opened on a point p in an RG S_1MS_2, where, without loss of generality, p is between S_1 and M, then S_1 cannot cover p, $d(p, S_2) > 1$, S_2 cannot reach M and M cannot reach S_1 (see Figure 13 for the case where S_1 is on the left).

Proof

If S_1 could cover p, but that would underutilize M, then M would cover p instead (Line 4 in Algorithm 2) since then it would still be able to reach S_2. Hence, if a fourth cluster is opened on p, S_1 cannot cover p and M could not cover p while still being able to reach a side cluster, implying that $d(p, S2) > 1$ and that M currently cannot reach S_1. Since by Lemma 7 the group was good before

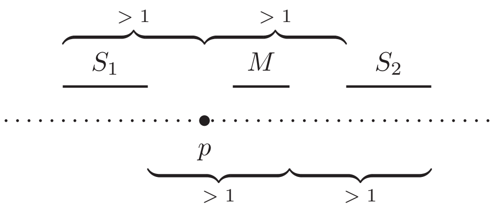


Figure 13: Inequalities for the fourth cluster to be opened.

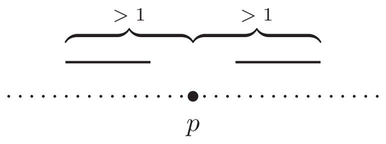


Figure 14: Inequalities for the third cluster to be opened in a T_2.

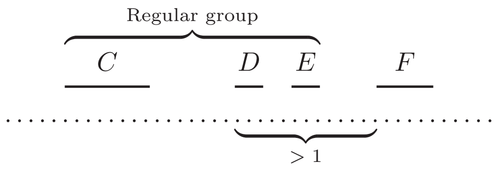


Figure 15: D cannot reach F.

p was given, this implies that S_2 cannot reach M. This follows because then S_2 and M would be a close pair and S_1 and M are already a far pair, which would mean that the group was bad. □

Corollary 1. If a side cluster in an RG is underutilized, then the group only contains three clusters.

Proof Follows directly from Lemma 8. □

Corollary 2. An R_4 never opens a fifth cluster.

Proof Just before becoming a 4-group, M could not reach S_1, so it could reach S_2; see Figure 13. Thus, after opening P, M can reach one side cluster, and P can reach the other. From the group inequalities, $d(S_1, S_2) \leq 2$, so any future point inside the group will be covered by M or P. □

Lemma 9. When the third cluster is opened on a point p in a T_2, neither of the side clusters can cover p; see Figure 14.

Proof Follows directly from Algorithm 3. □

Corollary 3. An additional cluster is never opened in a T_3.

Proof By definition of a tight group, the distance between the two side clusters is at most one. Hence, at most one middle cluster is needed in a tight group; see Line 4 in Algorithm 3. □

Lemma 10. The middle cluster in an R_3 cannot reach a non-group cluster; see Figure 15.

Proof We show that the inequality holds when the RG is created. It then follows from Line 2 in Algorithm 1 that F must be part of a group for the distance $d(l_D, l_F)$ to become less than or equal to one.

Consider the request p for which the group was created.

Assume, for the sake of contradiction, that D can reach F. It follows from Lemma 2 that the group must have been created by opening either D or E. Since all RGs are good, we have the scenario shown in Figure 16.

Let X be the cluster among D and E that existed already before the request to p. It follows that X could reach F. By Lemma 4, X and F were not a close pair before the request to p. Hence,

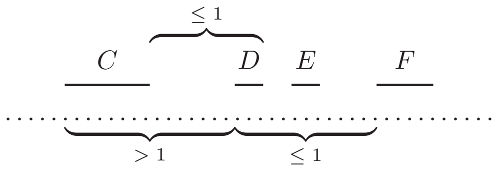


Figure 16: The scenario under the assumption that D can reach F.

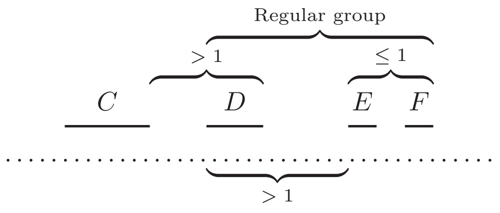


Figure 17: D cannot reach C

if F could cover p, this would not make X and F a close pair. On the other hand, if F could not cover p (if $X = E$, for instance), then covering p by X does not make X and F a close pair. Hence, there is always an existing cluster (X or F) which can cover p and create a good PRG. Therefore, according to Line 4 in Algorithm 1, one of them would. So no new cluster would be opened for p, a contradiction. □

Lemma 11. If $\overline{D}(EF)_1$ or $(FE)_1\overline{D}$ is created, then D can never later reach a cluster, C, not part of any group; see Figure 17.

Proof We show that D cannot reach C when the RG is created. It then follows from Line 2 in Algorithm 1 that C must be part of a group for this to become possible.

Assume for the sake of contradiction that D could reach C. Just before the RG is created, C cannot reach D by Lemma 4. Now, observe that an RG with two clusters contained in a unit interval must have been created when one of the two clusters E or F was opened (Lemma 2). Consider the request p for which the group was created, and assume without loss of generality that it was created when F was opened for the new point p. At that time, E could cover p, and it follows from the definition of an RG that if E indeed did that, then CDE would be an PRG; see Figure 18. The group would be good since C cannot reach D. Hence, by Line 4 in Algorithm 1, E would cover p, F would not have been opened, and we have reached a contradiction. □

Lemma 12. If an RG G_1 is close to another RG G_2, then G_1 and G_2 are not R_4s.

Proof Observe that by Line 7 in Algorithm 1, no RGs are created close to another group. Additionally, an R_4 cannot become close to another group since, when the fourth cluster was opened, neither side cluster could reach its nearest middle cluster; see Lemma 8. Hence, for G_1 to become close to G_2, it follows that G_2 must have covered a point p such that G_1 became close to it while they were still both R_3s. However, by Line 2 in Algorithm 1, when a new point is covered by a group, if there is any choice, then close groups are not created. Thus, since G_1 could also cover p, then if it did so, G_2 must have become close to G_1; see Figure 19.

Since $d(C,H) \leq 4$, there is no need to open more clusters. We argue that indeed the algorithm does not. Assume without loss of generality that F covers p. Then the conditions of Figure 13 can

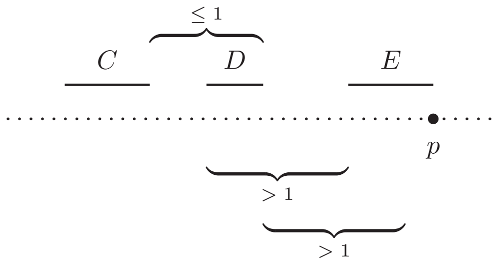


Figure 18: An alternative good RG if E covers p.

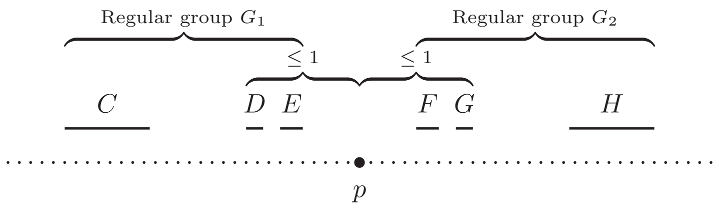


Figure 19: Two possible close groups.

never hold for G_1, since the fact that E will always be able to reach D means that it can never become the S_2 of Figure 13. Thus, G_1 can never become an R_4. For G_2, E will cover points between E and F, since it is underutilized, until F possibly becomes underutilized too. In any case, E and F will cover all points between D and G, and F will always be able to reach G. Therefore the conditions in Figure 13 will also not hold for G_2. □

Lemma 13. When a $(CD)_T$ is created in Line 7 in Algorithm 1 in order to avoid creating an R_3 which would be close to another group G_2 (with cluster E at the side of the tight group), let D be the cluster among C and D which is closest to G_2. Then there exists a point r covered by D which cannot be reached by E, while $d(r, C) > 1$; see Figure 20.

Proof If E cannot reach D when $(CD)_T$ is created, then let $r = r_D$ (r may stop being an endpoint of D later, but it will still satisfy the inequalities). Now, if E can reach D, then D must have been opened before the group G_2 was created (otherwise E would have covered at least the right endpoint of D by Line 2 in Algorithm 1). Consider the situation when G_2 was created. By Lemma 2, D and E could not be contained in a unit interval, and it follows that there exists a point r covered by D at a distance greater than one from the right endpoint of E. At some later point in time, a point p

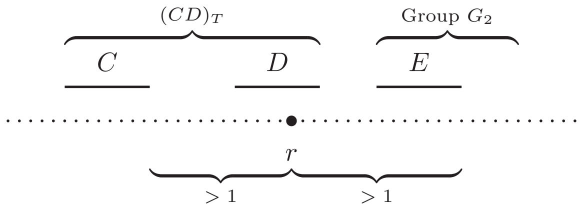


Figure 20: Location of a point r when a tight group is created.

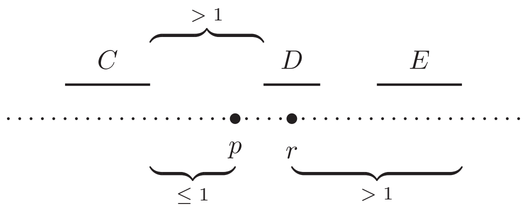


Figure 21: Location of a point r when a tight group is created.

arrives which is covered by D and $(CD)_T$ is created; see Figure 21. Note that this point p cannot be at the other side of D, because then CD would already be a tight group.

Hence, $d(C, r) > 1$, and the lemma follows. □

Lemma 14. The middle of three consecutive clusters not part of any group cannot be underutilized.

Proof This follows directly from Lemma 2 and Lemma 5: the two outermost clusters of the three would form a far pair in this case. □

3.1. Overall Proof Structure

The basic idea of the overall proof is to divide the clustering produced by our algorithm into chunks, and show that if we process this division from left to right, then the algorithm is $\frac{5}{3}$-competitive after each chunk. In order to keep track of our status in this process, we introduce the notation $\left\{ {a \atop b} \right\}$ which means that after possibly having discarded a number of chunks where the ratio of clusters used by the online algorithm to clusters used by OPT is at least as good as $\frac{5}{3}$, we are currently in a situation where the online algorithm has used a clusters and OPT has used b clusters.

We define an ordering on pairs in the following way:

$$\left\{ {a \atop b} \right\} \succeq \left\{ {c \atop d} \right\} \iff \forall x, y : \frac{x+c}{y+d} \leq \frac{5}{3} \Rightarrow \frac{x+a}{y+b} \leq \frac{5}{3}.$$

where the latter is equivalent to $5(b-d) \geq 3(a-c)$.

From this we also define the relations $\preceq$, $\sim$, $\prec$, and $\succ$ in the straight forward way, and addition of pairs as

$$\left\{ {a \atop b} \right\} \oplus \left\{ {c \atop d} \right\} = \left\{ {a+c \atop b+d} \right\}.$$

Note that $\left\{ {a+5 \atop b+3} \right\} \sim \left\{ {a \atop b} \right\}$ and that $\left\{ {1 \atop 1} \right\} \succ \left\{ {3 \atop 2} \right\}$, for example, expresses that in the $\left\{ {1 \atop 1} \right\}$ scenario, we are in a better situation with regards to obtaining the $\frac{5}{3}$ bound than if we were in the $\left\{ {3 \atop 2} \right\}$ situation.

For convenience, for the reader to be able to refer back, we list the relationships commonly used: $\left\{ {0 \atop 1} \right\} \sim \left\{ {5 \atop 4} \right\} \succ \left\{ {2 \atop 2} \right\} \succ \left\{ {4 \atop 3} \right\} \succ \left\{ {1 \atop 1} \right\} \succ \left\{ {3 \atop 2} \right\} \succ \left\{ {0 \atop 0} \right\} \succ \left\{ {2 \atop 1} \right\} \succ \left\{ {1 \atop 0} \right\}$.

Observe that we can produce an OPT-clustering of a request sequence (offline, that is), by processing the points from left to right, and cluster points greedily: only open new clusters when the previously opened cluster cannot cover the next point. Consider the situation after we have produced an OPT-clustering in this manner (from left to right).

We will define chunks starting from the left. Suppose that some chunks have already been defined. Let a be the number of clusters used by our algorithm so far, and let b be the number of OPT-clusters used in the processing of the clusters from left to right. Consider the rightmost

OPT-cluster that covers at least one point of the previous chunk and the next cluster that was opened by our algorithm after the previously defined chunk. We identify three states for the OPT cluster and associate a letter with each of these states. The OPT cluster might not be able to cover points from the next cluster opened by our algorithm. We use the letter N to denote this state. The remaining two states represent different situations where the OPT cluster can cover points from the next cluster opened by our algorithm. We use the letter A to denote the state where the OPT cluster cannot cover all of the next cluster opened by our algorithm. Finally, we use the letter S to denote the state where the OPT cluster partially used for the last cluster opened by our algorithm may be able to cover the entire next cluster opened by our algorithm. The letters describe advantages OPT might have when continuing. Thus, A stand for "advantage", N for "no advantage", and S for "super advantage".

We show that if we are in state N, then $\left\{ {a \atop b} \right\} \succeq \left\{ {0 \atop 0} \right\}$, if we are in state A, then $\left\{ {a \atop b} \right\} \succeq \left\{ {3 \atop 2} \right\}$, and if we are in state S, then $\left\{ {a \atop b} \right\} \succeq \left\{ {2 \atop 2} \right\}$. With a slight abuse of notation we also use N, A, and S to denote $\left\{ {0 \atop 0} \right\}$, $\left\{ {3 \atop 2} \right\}$, and $\left\{ {2 \atop 2} \right\}$, respectively. Observe that $S \succ A \succ N$.

Since we process from left to right, we start in state N, since there are no clusters to the left of the leftmost point that can help OPT. If we can show that the above is an invariant after each decision our algorithm makes, then we have shown that our algorithm is $\frac{5}{3}$-competitive.

In order to make the analysis easier to carry through, we usually only require OPT to serve points that are endpoints of a cluster opened by our algorithm. When we deviate from this, we explicitly discuss such a point and argue why we still have the necessary properties. Also, there are a few situations where the last OPT-cluster may be able to reach more than just the next online cluster. We also make this clear when it occurs.

We now describe how we divide the clustering into chunks. First, observe that we only need to consider sets of consecutive points where the distance between each two consecutive points is at most one, since no cluster opened by the algorithm or by OPT can be shared between two such sets of points. In each set, each group is a chunk and anything between two consecutive groups or at the end of a set is a chunk. For each chunk and starting state we analyze the chunk (in a worst case manner) and identify the end state and the number of new clusters used by our algorithm and OPT. If, for example, we are in state S before we process a chunk, the online algorithm uses four clusters to process the chunk, the OPT-clustering only needs to open two new clusters for the chunk, and the end state is A, then we are then faced with the inequality $S \oplus \left\{ {4 \atop 2} \right\} \succeq A$, which is true since $S \oplus \left\{ {4 \atop 2} \right\} \sim \left\{ {2 \atop 2} \right\} \oplus \left\{ {4 \atop 2} \right\} \sim \left\{ {6 \atop 4} \right\} \sim \left\{ {1 \atop 1} \right\} \succ \left\{ {3 \atop 2} \right\} \sim A$.

Based on the properties of the algorithm established in the above, we now proceed to analyze groups and sequences of non-group clusters, based on the state classification N, A, and S.

3.2. Groups

For groups, the analysis can be completely captured in a collection of figures. We now describe how such figures should be read. Confer with the first figure (Figure 22) below as an example. Above the dotted line, we show the configuration that we are analyzing. We do not know the placement of OPT's clusters, but we know that we can assume a division of starting states according to the classification described above. Thus, each possible OPT starting state is analyzed separately, using one line on each. When this is not obvious from the distances given in the online configuration, we give a reference to a lemma on top of an OPT cluster explaining why that cluster cannot reach the next point, or why a given end state can be argued. Similarly, we may include references in the figure captions to lemmas justifying the distances displayed in the configuration.

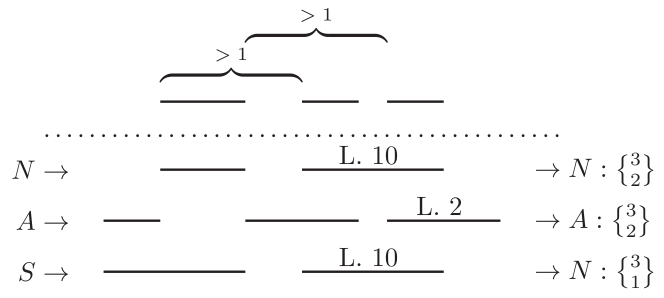

> 1
> 1
$N \rightarrow$
L. 10
$\rightarrow N : \left\{ {3 \atop 2} \right\}$
$A \rightarrow$
L. 2
$\rightarrow A : \left\{ {3 \atop 2} \right\}$
$S \rightarrow$
L. 10
$\rightarrow N : \left\{ {3 \atop 1} \right\}$

Figure 22: R$_3$ Case 1: Far cluster to the left.

All the way to the right on an OPT line, we list a pair denoting how many new online and OPT clusters, respectively, were used on that line. As an example, in Figure 22, in the second OPT line (starting state A), the first OPT cluster in the line has already been counted because we analyze from left to right and the cluster shares a point with whatever online configuration is placed to the left of the current. Thus, three new online and two new OPT clusters are used. In all cases, we must verify (and have done so) that the starting state plus the pair of clusters that are used sum up to at least the end state, e.g., $A \oplus \left\{ {3 \atop 2} \right\} \succeq A$.

We analyze the groups at the time of creation (for R$_3$s and T$_2$s), at the time when the fourth cluster is opened (for R$_4$s), and at the time when the third cluster is opened (for T$_3$s). In addition, we analyze the situation where an RG is close to another group separately; see Lemma 12. Hence, in the analysis of RGs and tight groups, we can assume they are not close to another group.

Note from the captions that this is an exhaustive case analysis.

Before we continue, notice that the N-in and S-in cases in Figure 22 are very similar. This is very often the situation, though not always, as can be seen from Figure 23. We use this observation to eliminate most of the N-in cases from here onwards, only displaying them when not similar to the S-in case. The following lemma formally allows us to do this:

Lemma 15. Assume an S-in case leads to T-out, where T is any of N, A, or S, at a cost of $\left\{ {x \atop y} \right\}$, $S \oplus \left\{ {x \atop y} \right\} \succeq T$, and the first OPT-cluster in the corresponding N-in case covers no more than the first cluster in the S-in case (the cluster that started in the previous chunk), then the N-case also leads to T-out, the costs are $\left\{ {x \atop y+1} \right\}$, and $N \oplus \left\{ {x \atop y+1} \right\} \succeq T$.

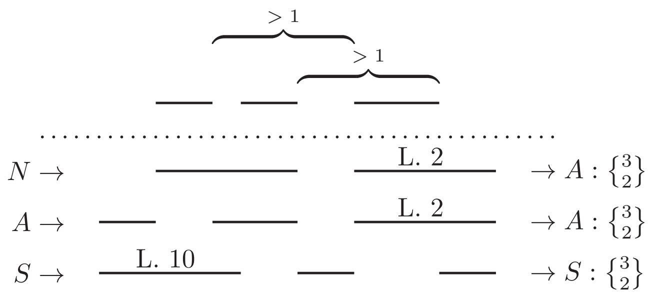
Proof If the first OPT-cluster in the N-in case covers no more, it of course covers exactly the same, and we must end in the same state. Finally,

$$N \oplus \begin{Bmatrix} x \\ y+1 \end{Bmatrix} \sim \begin{Bmatrix} x \\ y+1 \end{Bmatrix} \sim \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \oplus \begin{Bmatrix} x \\ y \end{Bmatrix} \succ \begin{Bmatrix} 2 \\ 2 \end{Bmatrix} \oplus \begin{Bmatrix} x \\ y \end{Bmatrix} \sim S \oplus \begin{Bmatrix} x \\ y \end{Bmatrix}.$$

□

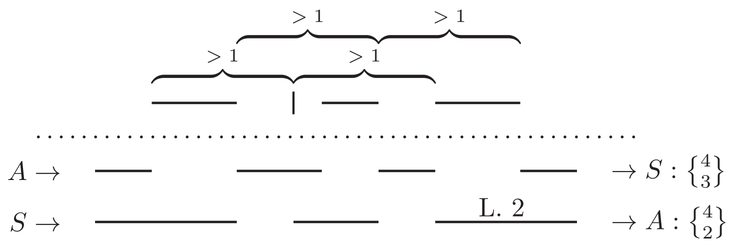
For the R$_4$ Case (Figure 24), observe that the used constraints are symmetric. Thus, the analysis of the case where the fourth cluster is to the right of the other middle cluster is identical to the above.

This concludes the analysis of all group cases.



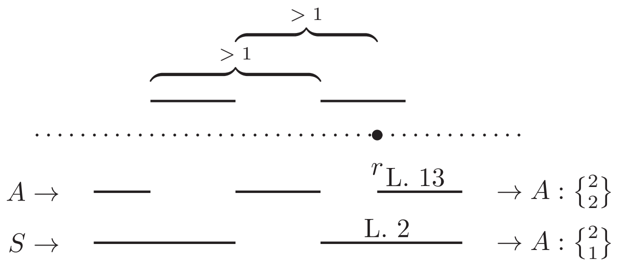
> 1
> 1
$N \to$
L. 2
$\to A : \left\{{3 \atop 2}\right\}$
$A \to$
L. 2
$\to A : \left\{{3 \atop 2}\right\}$
$S \to$
L. 10
$\to S : \left\{{3 \atop 2}\right\}$

Figure 23: R_3 Case 2: Far cluster to the right.



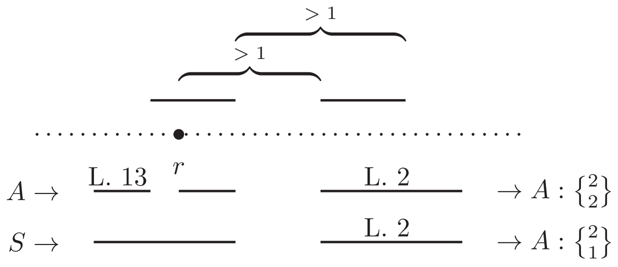
> 1
> 1
> 1
> 1
$A \to$
$\to S : \left\{{4 \atop 3}\right\}$
$S \to$
L. 2
$\to A : \left\{{4 \atop 2}\right\}$

Figure 24: R_4 Case (L. 8).



> 1
> 1
r
$A \to$
L. 13
$\to A : \left\{{2 \atop 2}\right\}$
$S \to$
L. 2
$\to A : \left\{{2 \atop 1}\right\}$

Figure 25: T_2 Case 1: Group to the right (L. 13).



> 1
> 1
r
$A \to$
L. 13
L. 2
$\to A : \left\{{2 \atop 2}\right\}$
$S \to$
L. 2
$\to A : \left\{{2 \atop 1}\right\}$

Figure 26: T_2 Case 2: Group to the left (L. 13).

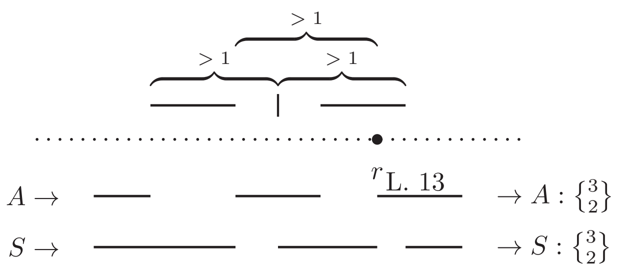

> 1
> 1
> 1
r
L. 13
$A \rightarrow$
$\rightarrow A : \left\{ {3 \atop 2} \right\}$
$S \rightarrow$
$\rightarrow S : \left\{ {3 \atop 2} \right\}$

Figure 27: T_3 Case 1: Group to the right (L. 13, L. 9).

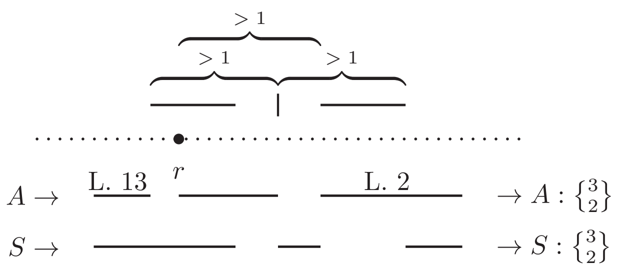

> 1
> 1
> 1
L. 13
r
L. 2
$A \rightarrow$
$\rightarrow A : \left\{ {3 \atop 2} \right\}$
$S \rightarrow$
$\rightarrow S : \left\{ {3 \atop 2} \right\}$

Figure 28: T_3 Case 2: Group to the left (L. 13, L. 9).

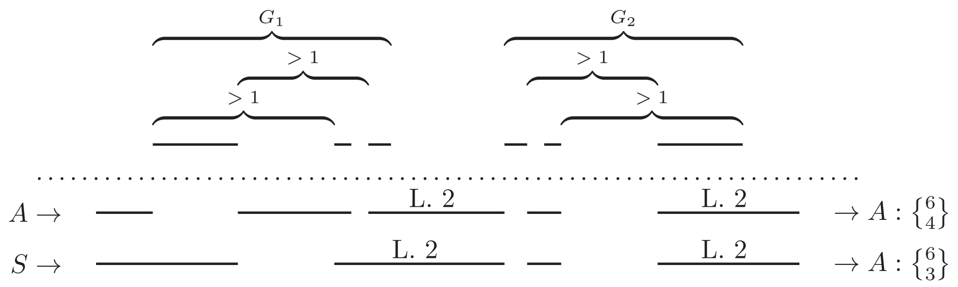

G_1
G_2
> 1
> 1
> 1
> 1
L. 2
L. 2
$A \rightarrow$
$\rightarrow A : \left\{ {6 \atop 4} \right\}$
L. 2
L. 2
$S \rightarrow$
$\rightarrow A : \left\{ {6 \atop 3} \right\}$

Figure 29: Close Group: Just before it became close; see Figure 19.

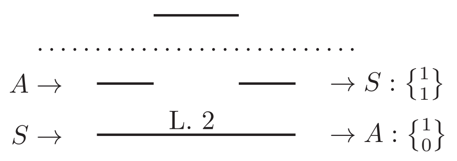

$A \to$
$\to S : \left\{{1 \atop 1}\right\}$
$S \to$
L. 2
$\to A : \left\{{1 \atop 0}\right\}$

Figure 30: Single Cluster Case.

Group to the right	$\left\{{a \atop b}\right\}$	$A \oplus \left\{{1 \atop 1}\right\} \oplus \left\{{a \atop b}\right\}$	Out state
$DE\overline{F}$ (Figure 23)	$\left\{{3 \atop 2}\right\}$	$\left\{{7 \atop 5}\right\} \sim \left\{{2 \atop 2}\right\}$	S
R_4 (Figure 24)	$\left\{{4 \atop 2}\right\}$	$\left\{{8 \atop 5}\right\} \sim \left\{{3 \atop 2}\right\}$	A
T_2 (Figure 25)	$\left\{{2 \atop 1}\right\}$	$\left\{{6 \atop 4}\right\} \sim \left\{{1 \atop 1}\right\}$	A
T_3 (Figure 27)	$\left\{{3 \atop 2}\right\}$	$\left\{{7 \atop 5}\right\} \sim \left\{{2 \atop 2}\right\}$	S

Table 2: Analysis of A-in/S-out case for a single cluster.

3.3. Outside Groups

We now analyze a maximal sequence of clusters, none of which are a part of any group. This can be an isolated sequence, a sequence with a group in one end, or a sequence in between two groups. We divide the analysis up into different cases depending on the length of the sequence. In the analysis of sequences of length at least three, we repeatedly apply Lemma 14 for each cluster except for the first and last in the sequence.

3.3.1. A Single Cluster

Consider the single cluster case (Figure 30). If this is not the end of the sequence, but continues with a group to the right, then there is a potential problem with the first line, since $A \oplus \left\{{1 \atop 1}\right\} \not\succeq S$. If the group G to the right is of the form $\overline{D}EF$, then Lemma 11 implies that OPT starts G in state A instead. This leaves the cases $DE\overline{F}$, R_4, T_2, and T_3. See Table 2. In total, it works for all cases.

3.3.2. Two Clusters

For a maximal sequence of two clusters outside groups, we analyze the case under the worst-case assumption that both clusters are underutilized; see Figure 31.

First, observe that for the N-in/S-out case, there are no problems.

Now, consider the A-in/A-out case. This is a harder case, the treatment of which is deferred to Section 4.

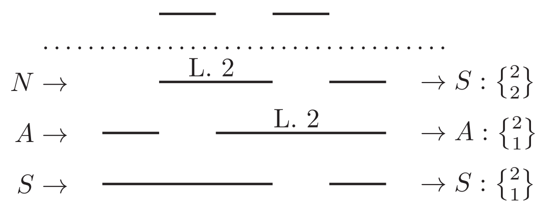
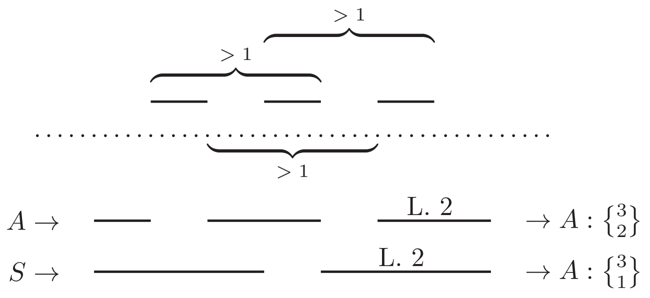
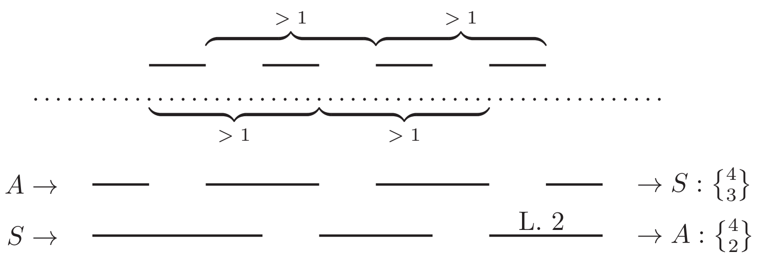

$N \to$
L. 2
$\to S : \left\{{2 \atop 2}\right\}$
$A \to$
L. 2
$\to A : \left\{{2 \atop 1}\right\}$
$S \to$
$\to S : \left\{{2 \atop 1}\right\}$

Figure 31: Two Cluster Case: Both clusters underutilized.



> 1
> 1
> 1
$A \to$
L. 2
$\to A : \left\{ {3 \atop 2} \right\}$
$S \to$
L. 2
$\to A : \left\{ {3 \atop 1} \right\}$

Figure 32: Three Cluster Case (L. 2, L. 14).



> 1
> 1
> 1
> 1
$A \to$
$\to S : \left\{ {4 \atop 3} \right\}$
$S \to$
L. 2
$\to A : \left\{ {4 \atop 2} \right\}$

Figure 33: Four Cluster Case (L. 2, L. 14).

Finally, consider the S-in/S-out case. Since we are in the two cluster case, there is a group G just to the right; otherwise S-out becomes N-out, which is fine. By Lemma 11, similar to the single cluster case, S-out is only possible if G does not have the far cluster to the left. Otherwise, S-out becomes A-out, which is fine. Now consider the case of S-out and G not having the far cluster to the left. This leaves the remaining cases: R_3, R_4, T_2, and T_3. For those groups, consider the case with S as starting state for the two clusters. We need to consider the value of $S \oplus \left\{ {2 \atop 1} \right\} \oplus \left\{ {a \atop b} \right\}$, but since $S \oplus \left\{ {2 \atop 1} \right\} \sim A \oplus \left\{ {1 \atop 1} \right\}$, we get exactly the same results as in Table 2.

3.3.3. Three Clusters

The only problematic case is S-in/A-out; see Figure 32. This is a harder case, the treatment of which is deferred to Section 4.

3.3.4. Four Clusters

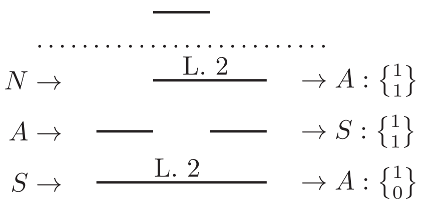
This case works directly; see Figure 33.

3.3.5. More Clusters

Now consider a maximal sequence of five or more clusters outside groups.

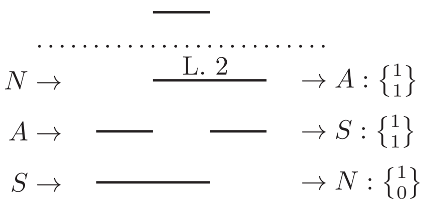
By Lemma 14, only the first and last cluster in such a sequence can be underutilized. Hence, for the first and last cluster, we get Figure 34 and for any cluster in between we get Figure 35.

For a middle cluster it can be observed that an N-in state is converted to an A-out state, an A-in state is converted to an S-out state, and an S-in state is converted to an N-out state. The number of middle clusters in the sequence can be written as $3a + b$, where a and b are integers, $b \in \{0, 1, 2\}$, and $3a + b \geq 1$.



L. 2
L. 2
$N \rightarrow$
$A \rightarrow$
$S \rightarrow$
$\rightarrow A : \left\{ {1 \atop 1} \right\}$
$\rightarrow S : \left\{ {1 \atop 1} \right\}$
$\rightarrow A : \left\{ {1 \atop 0} \right\}$

Figure 34: In and out cases.



L. 2
$N \rightarrow$
$A \rightarrow$
$S \rightarrow$
$\rightarrow A : \left\{ {1 \atop 1} \right\}$
$\rightarrow S : \left\{ {1 \atop 1} \right\}$
$\rightarrow N : \left\{ {1 \atop 0} \right\}$

Figure 35: A middle cluster.

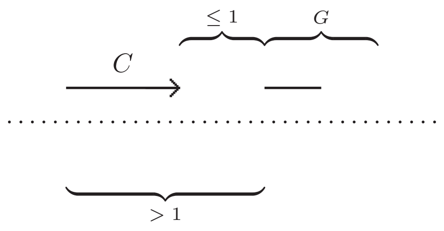


Figure 36: A binding cluster.

Since this structure repeats itself cyclicly for every three middle clusters and OPT uses two clusters for each cycle, we must verify the equations $N \oplus \left\{ {1 \atop 1} \right\} \oplus a \cdot \left\{ {3 \atop 2} \right\} \oplus \left\{ {1 \atop 1} \right\} \succeq S$ for any integer a, as well as the similar equations where the starting and ending states N and S are replaced by A and N, as well as S and S, for the case $b = 0$. However, if such an inequality holds for some a, then the inequality also holds for $a + 1$ since $\left\{ {3 \atop 2} \right\} \succeq \left\{ {5 \atop 3} \right\}$. Thus, it is sufficient that we verify these equations for $a = 0$. We must also verify the equations for $b = 1$ and $b = 2$, and again, it is sufficient to consider $a = 0$.

In Section 3.3.1 through Section 3.3.4, we have handled most of these cases, and have already shown that the inequalities hold for $a = 0$, except for the following two cases: $b = 0$, A-in, and $b = 1$, S-in, which both correspond to the case handled in the next section (we need some additional definitions to deal with them). Hence, for those two cases, it does not follow immediately that the situation is resolved for larger a. We establish this by verifying these two cases as described above, but for $a = 1$. The first case ($a = 1$, $b = 0$, A-in) results in $A \oplus \left\{ {1 \atop 1} \right\} \oplus \left\{ {3 \atop 2} \right\} \oplus \left\{ {1 \atop 0} \right\} \succeq A$, which is true, and the second case ($a = 1$, $b = 1$, S-in) results in $S \oplus \left\{ {1 \atop 0} \right\} \oplus \left\{ {3 \atop 2} \right\} \oplus \left\{ {1 \atop 1} \right\} \oplus \left\{ {1 \atop 0} \right\} \succeq A$, which is also true.

4. Handling Remaining Cases

4.1. Preliminaries

Consider a regular group G and a cluster C to the left (this definition is intentionally not symmetric!) of G which is not part of any group. We say that C is a *binding cluster* if C cannot reach any cluster from G, but $d(C, G) \leq 1$; see Figure 36, We say that C *binds* to G. In the figures, we use an arrow to indicate a binding cluster and let the arrow point to the group the cluster binds to.

There are two remaining cases from the first part of the paper. The first is the case with two clusters outside groups with A-in/A-out and groups to both sides; see Figure 31. The second is the case with three clusters outside groups with S-in/A-out and groups to both sides; see Figure 32.

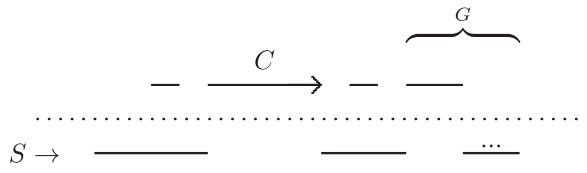


Figure 37: Analysis of a dangerous configuration.

G	$\left\{ {a \atop b} \right\}$		Out state
R$_3$, far left (Figure 22)	$\left\{ {3 \atop 2} \right\}$	$\sim$	A
R$_3$, far right (Figure 23)	$\left\{ {3 \atop 2} \right\}$	$\sim$	A
R$_4$ (Figure 24)	$\left\{ {4 \atop 3} \right\}$	$\not\succeq$	S
T$_2$, group right (Figure 25)	$\left\{ {2 \atop 2} \right\}$	$\succ$	A
T$_3$, group right (Figure 27)	$\left\{ {3 \atop 2} \right\}$	$\sim$	A
Close group (Figure 29)	$\left\{ {6 \atop 4} \right\}$	$\succ$	A

Table 3: A group together with a dangerous configuration.

Observe that both cases contain a binding cluster (the left cluster in the first case and the middle cluster in the second case).

Consider the three clusters outside groups with the S-in/A-out case; see Figure 37. We coin this a *dangerous configuration* since such a cluster (the middle one) with one or two underutilized clusters on its sides represent the hardest scenario between groups.

Adding the $\left\{ {3 \atop 1} \right\}$ result of such a dangerous configuration to $S = \left\{ {2 \atop 2} \right\}$, we get $\left\{ {5 \atop 3} \right\} \sim \left\{ {0 \atop 0} \right\}$. Considering each of the possibilities for the group G in Table 3, we see that only a dangerous configuration with a regular 4-group to the right can lead to a ratio larger than $\frac{5}{3}$ for our algorithm.

We can establish the following regarding the creation of binding clusters:

Lemma 16. If there exists a cluster between a binding cluster C and the regular group to which it binds at the time of this binding, then the binding cluster can only be created by C covering a point to the right of C.

Proof Let C denote the binding cluster and let D denote the cluster closest to C in the regular group $G = DEF$ to which C binds. Assume that there exists a cluster I between C and D when the binding cluster configuration is created by serving a point p. First, assume that the dangerous configuration is created by the creation of the group G; see Figure 38.

It follows from Lemma 2 that D could not reach C before p was given, since otherwise D and I would have been contained in a unit interval. By Lemma 6, the group G cannot be created by

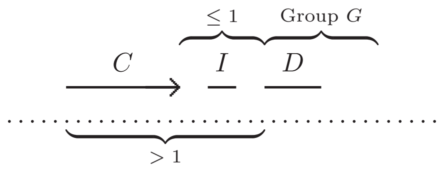


Figure 38: Location of cluster I.

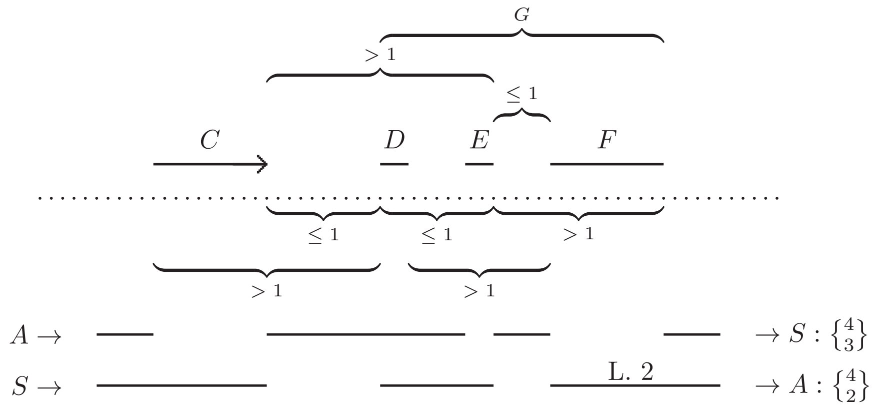


Figure 39: An R-ERG.

D covering a point to the left of it. Hence, $d(C, D)$ was at most one before p was given. It follows that C and D were a far pair before p was given, which is a contradiction with Lemma 5.

Now, observe that C cannot just have been opened for p, since by the definition of a binding cluster it must have positive size.

Finally, the binding cluster configuration cannot be created by C covering a point (p) to the left of C, since then just before p was given, C and I would have been contained in a unit interval, which is not possible by Lemma 2. □

4.2. Additional Group Types

To handle the binding cluster cases, we have to introduce additional group types.

The Group Type R-ERG

A cluster C, which is not part of any group, and an RG DEF, related as shown in Figure 39, is denoted a *right-oriented extended* regular group (its far cluster is to the right), R-ERG. Observe that at most three clusters (including D and E) are needed between C and F in order to cover the entire space between C and F. In addition, the clusters D and E are located in such a way that at most one additional cluster is needed. In Algorithm 4, we describe a behavior of an R-ERG that ensures that at most one additional cluster is needed between C and F, and the argument is given in Lemma 17.

Lemma 17. Algorithm 4 is correct.

Proof We must prove that the behavior after a possibly third cluster is opened between C and F is possible, i.e., when a cluster is supposed to cover a point, we must argue that the point is indeed reachable from that cluster.

In serving p, a new cluster P is opened if covering with D would mean that none of the following two conditions hold: 1) D can reach C, or 2) $d(l_D, l_F) \leq 2$.

Thus, one of these two conditions still hold after opening P. We remark here, referring to Figure 39, that the R-ERG is created in Algorithm 5 on the condition that D and E are contained in a unit interval. Together with the group condition that $d(r_E, l_F) \leq 1$, this implies that $d(l_D, l_F) \leq 2$.

Algorithm 4 Right-oriented extended regular group, R-ERG

```
Require: $p$ is the next point given and $p$ falls in an R-ERG with side clusters $C$ and $F$ and middle
    clusters $D$ and $E$
 1. if $C$ can cover $p$ then
 2.     Cover $p$ with $C$
 3. else if $F$ can cover $p$ and doing so would not underutilize $E$ then
 4.     Cover $p$ with $F$
 5. else if $E$ can cover $p$ and still reach $F$ then
 6.     Cover $p$ with $E$
 7. else if $D$ can cover $p$ and after doing so it could still reach $C$
        or have $d(l_D, F) \leq 2$ then
 8.     Cover $p$ with $D$
 9. else
10.     Open a new cluster $P$ for $p$ and divide the responsibility for space of length at most three
        between $C$ and $F$ as follows: $C$ and $F$ do not cover points in that space and $E$ covers
        points at a distance at most one from $F$. The leftmost cluster of $P$ and $D$ covers points at
        a distance up to one from $C$ and the remaining cluster covers all other points.
```

After that, if one of the two conditions holds, then only covering by D can change that, and Algorithm 4 brings us to exactly this case, if covering by D would negate both conditions.

If D can still reach C, then D could have covered a point to the left of D, so P must have been opened between D and E. Clearly, D and E can cover future points as specified, and since P has length zero, it can cover any future points in the gap of length at most one between where D and E are supposed to cover.

If $d(l_D, l_F) \leq 2$, then D could have covered any point to its right that E did not cover, so p is between C and D. Since $d(C, F) \leq 3$ and $d(p, F) > 2$ (since D did not cover it), P can reach C. Since the second condition holds and D could have covered p, D can cover all the points in the space of length at most one between P and E. □

The Group Type L-ERG

We will also need to form a special group when G is an R_3 of the form $\overline{D}(EF)_1$, and C to the left of D becomes a binding cluster. This can be due to D covering a point to its left or C covering a point to its right. In both cases, we group C together with G, creating a *left-oriented extended* regular group (its far cluster is to the left), L-ERG. see Figure 40.

In Figure 40, we have shown the situation when D covers p. For this case, at the time of creation, all cases work out directly. Also observe that the only way the S-in (and N-in) cases can result in A-out is if G was close with another group (otherwise these two cases would have resulted in N-out). Hence, the above also works even if G was close with another group before p was given. If instead it is C that covers p, OPT cannot do any better. Since C was not a binding cluster before, we still have the points q_C and q_D such that $d(q_C, q_D) > 1$, which is important for the A-in case, and since D is a far cluster, D cannot reach E, which establishes the rest of OPT's clusters for both cases.

Now, we must specify the future behavior of this construction, and show that all cases work out for any number of future given points and clusters opened.

Before proceeding, we identify three points q_C, q_D, and q_E, where q_C is the right endpoint of

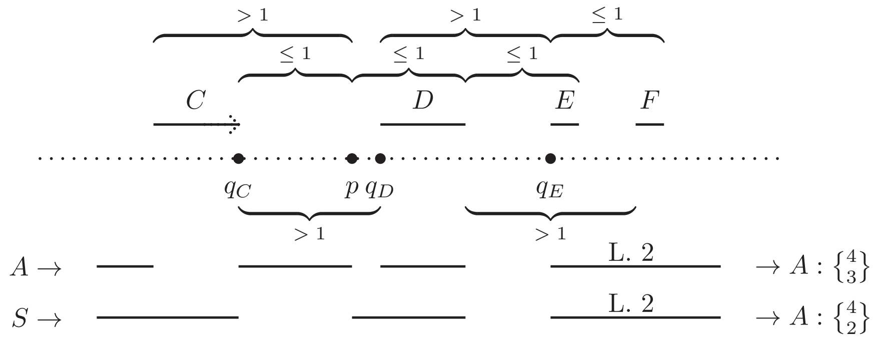


Figure 40: Left-oriented extended regular group, L-ERG. C becomes a binding cluster only after D covers p.

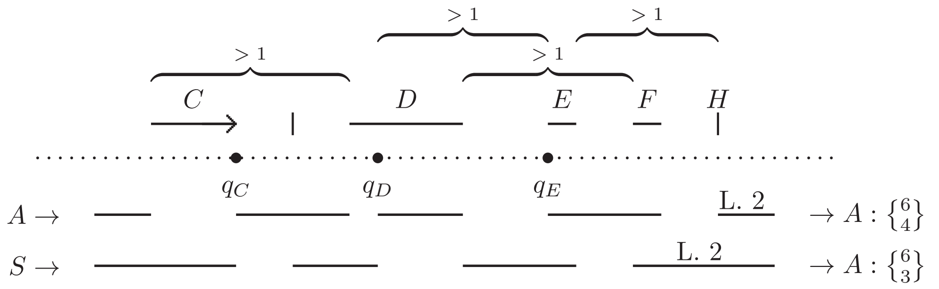


Figure 41: L-ERG with an additional cluster between C and D and just to the right of F.

C, q_D is the left endpoint of D before D covered p, and q_E is the left endpoint of E. As shown in the figure, the distance between each pair of the three points is greater than one.

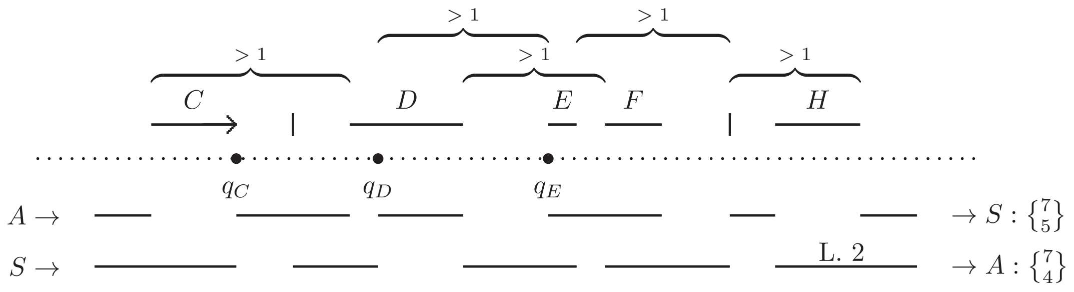
We now describe how future points are handled by this configuration: C and D cover points if possible (however, D is not allowed to underutilize E), E covers points as long as it can still reach D, and F covers points within distance two of D, if possible. An additional cluster X might be needed between C and D. However, since D just covered p, after this action, $d(C, D) \leq 1$, so we need at most one cluster between C and D. If X is opened, it will cover any future points between C and D. Also, an additional cluster H might be needed just to the right of F (for a point which F could cover, but if it did, it would have covered a point at a distance greater than two from D). However, H is only opened on a point p if p cannot be covered by any other cluster (part of a group or not). If H is opened, then it covers points if possible without underutilizing F, i.e., it does not cover points within a distance of two from D. Eventually, this might lead to a final cluster Y having to be opened between F and H. In addition, the behavior of F is updated if H is opened so it covers a point if it can afterwards still reach H or has not covered a point at a distance greater than two from D. This might lead to an extra cluster Y being opened between E and F.

We now analyze the configuration when these additional clusters are opened. If H is opened, it must be at a distance more than one from l_E, so OPT cannot cover using the cluster covering E and F, so its cost increases as much as our algorithm's cost.

Observe that all the states are A-out since if a cluster just to the right of F could cover the point on which H is going to be opened, then it does so. All cases give sufficiently high values.

Now consider the configuration when both X and H are opened; see Figure 41.

Again, all cases lead to sufficiently good values. Now, consider the case where X is opened,



> 1
> 1
> 1
> 1
> 1
C
D
E
F
H
q_C
q_D
q_E
$A \to$
$S \to$
L. 2
$\to S : \left\{ {7 \atop 5} \right\}$
$\to A : \left\{ {7 \atop 4} \right\}$

Figure 42: L-ERG with an additional cluster between F and H.

but where H is not opened. This saves a cluster for the algorithm, and we should consider the impact on OPT. Still referring to Figure 41, OPT cannot place its clusters better for the S-in (and N-in) cases, since it must cover F. For the A-in case, OPT could save the cluster that should have covered H, but then it has no advantage when considering the next configuration of clusters. Thus, the result is $\left\{ {5 \atop 3} \right\}$, since both the algorithm and OPT save a cluster and it becomes an N-out case.

Finally, we need to consider the situation when Y is opened either to the left or right of F, both with and without the opening of X. Assume without loss of generality that Y is opened between F and H. For the case where X is opened, see Figure 42. All cases work out. If X is not opened, then, since H cannot reach Y or E, both the algorithm and OPT must use two additional clusters compared with Figure 40, and $\left\{ {2 \atop 2} \right\}$ can provide even an S-out.

4.3. An Algorithm for Binding Clusters

We now alter our algorithm to avoid problematic binding clusters. The change is the following: whenever Algorithm 1 would create a binding cluster, we let Algorithm 5 handle the point instead. Algorithm 5 says how to handle that particular point and serves to organize the subsequent case analysis where we account for correctness and ratios, and, in each case, explain how further points are handled.

In short, the algorithm can be formulated as follows: if the binding cluster is caused by a new regular group, we can create something else which is not problematic. If the binding cluster is caused by a regular group covering a point, we open a new cluster. Otherwise, we create the binding cluster.

4.4. Analysis

As a consequence of Lemma 16, in the following case analysis, we can assume that there is no cluster between the binding cluster and the regular group to the right in all cases, except when the binding cluster is created by that cluster covering a point.

In the following, we analyze each action taken in Algorithm 5. We show that the algorithm is well-defined, analyze the configurations created, and describe how future points are handled (if nothing is mentioned, future points are handled by Algorithm 1).

Recall that the only remaining cases from the main analysis are exactly cases with a binding cluster, and also recall, from the beginning of this section, that this only represents a problem in connection with an R_4.

Finally, observe that the RGs suggested by Algorithm 1 are good PRGs by definition.

Algorithm 5 Binding cluster

```
Require: $p$ is the next point given and Algorithm 1 would have created a binding cluster
 1. Let $C$ and $G = DEF$ be the binding cluster and group that Algorithm 1 would have created
 2. Let $B$ be the cluster immediately to the left of $C$, if it exists
 3. if the binding cluster is caused by the creation of $G$ then
 4.     if $D$ can reach $C$ then
 5.         Apply Algorithm 1, i.e., create $G$ and the dangerous configuration
 6.     else
 7.         Let $D'$ be the cluster that becomes $D$ if we let it cover $p$
 8.         Open a new cluster $P$ for $p$ and create the group $CD'P$
 9. else if $G$ is $\overline{D}(EF)_1$ then
10.     Apply Algorithm 1, i.e., cover by $D$ if possible and otherwise by $C$ and create an L-ERG
11. else if $G$ was already a group and would create this situation by covering $p$ then
12.     Open a new cluster $P$ for $p$
13. else if $G$ is $(DE)_1\overline{F}$ then
14.     Apply Algorithm 1, i.e., cover $p$ with $C$, and create an R-ERG
15. else
16.     Open a new cluster $P$ for $p$, and if $P$ is between $C$ and $D$, then as a first priority, $P$ covers
        future points if it can do so and still reach $D$
```

Line 5 in Algorithm 5: Binding Cluster

In this case, we create the new regular group and the binding cluster. By the preference in Algorithm 1 to create the leftmost possible regular group, CDE is not a PRG. All conditions for CDE being a PRG follow from DEF being a PRG and C being a binding cluster, with the exception of $d(C, E) > 1$. Thus, CDE not being a PRG implies that D is underutilized. It follows from Corollary 1 that the group cannot be an R_4 and cannot develop into an R_4 at some later point.

Line 7 in Algorithm 5: D covered p

We need to argue why D must have covered p. We are in the case where the binding cluster is caused by the creation of G, so one of the group clusters covered p. By the algorithm, D cannot reach C, and by definition of a binding cluster, C cannot reach D. Hence, D and C is a far pair. By Lemma 5, no far pairs existed before p was given, and it follows that D' must have covered p. By Lemma 6, p must fall between D' and E.

Line 8 in Algorithm 5: New Cluster and Group

First, consider well-definedness. The situation is shown in Figure 43. By Lemma 5, C and D' are not a far pair. Since C would bind to DEF, C and D' are also not a close pair. Hence, $CD'P$ is a good RG.

Again, we need to argue that by creating the alternative group, no binding clusters are created. First consider just to the right of the new group. Since E and F were considered for regular group formation, they were not binding clusters before p was given and therefore cannot be binding clusters after processing p, because they did not change. Now consider just to the left of C. The cluster B cannot be a binding cluster to $CD'P$. This follows since C can reach B, since otherwise B and C would be a far pair, so if B were a binding cluster, then BCD' would be a good PRG,

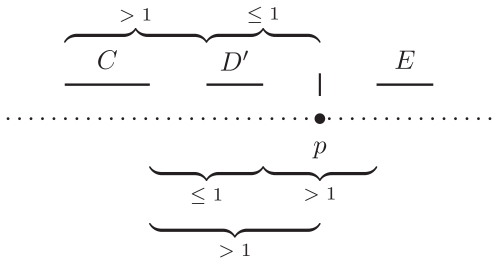


Figure 43: The situation in Line 8 in Algorithm 5.

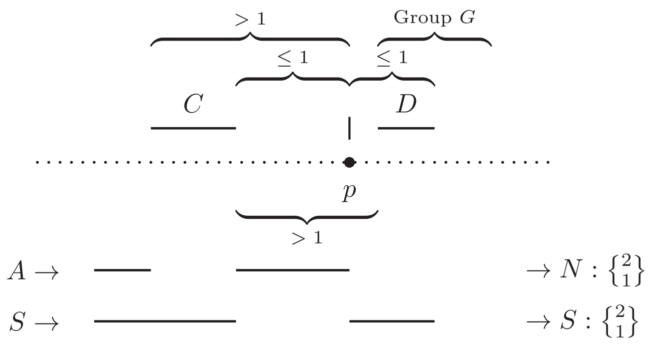


Figure 44: Open P for p.

contradicting Lemma 1. Note that BCD' could not be bad, because B cannot reach C, but C can reach B.

Line 10 in Algorithm 5: Creating an L-ERG

This case includes the close group case; see Figure 29. Either D or C covers p, and we then create an L-ERG. This construction has been completely analyzed in Section 4.2.

Line 12 in Algorithm 5: Opening a New Cluster

If G is not of the form $G \neq \overline{D}(EF)_1$ everything works out, i.e., when a new cluster is opened for p; see Figure 44. The only case that does not work out directly is the S-in/S-out case. However, consider that case together with all possibilities for G. As described above, $G \neq \overline{D}(EF)_1$, but we do need to analyze the case $G = \overline{D}(EF)_2$; see Figure 45.

Now consider the S-in/S-out case from above together with all possibilities for G and observe that $S \oplus \left\{ {2 \atop 1} \right\} \sim \left\{ {4 \atop 3} \right\}$; see Table 4. All cases work out.

Line 14 in Algorithm 5: Creating an R-ERG

First, consider well-definedness. The situation is shown in Figure 39. The inequalities follow from the fact that G is a regular group with two clusters contained in a unit interval, that C is a binding cluster, and that E cannot reach C (Lemma 10). Clearly, C together with G fulfill the requirements for an R-ERG.

As observed earlier, at most one additional cluster is needed in an R-ERG. By the way an R-ERG handles points, the additional cluster can either be opened just to the left or just to the

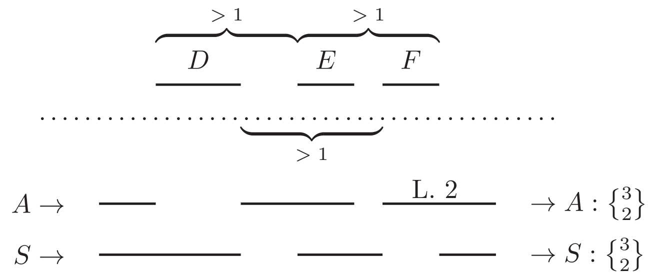


Figure 45: R_3: Far cluster to the left and not two clusters in a unit interval, $\overline{D}(EF)_2$.

G	$\left\{ {a \atop b} \right\}$	$\left\{ {4 \atop 3} \right\} \oplus \left\{ {a \atop b} \right\}$	Out state
R_3, far right (Figure 23)	$\left\{ {3 \atop 2} \right\}$	$\left\{ {7 \atop 5} \right\} \sim \left\{ {2 \atop 2} \right\}$	S
R_4 (Figure 24)	$\left\{ {4 \atop 2} \right\}$	$\left\{ {8 \atop 5} \right\} \sim \left\{ {3 \atop 2} \right\}$	A
$\overline{D}(EF)_2$ (Figure 45)	$\left\{ {3 \atop 2} \right\}$	$\left\{ {7 \atop 5} \right\} \sim \left\{ {2 \atop 2} \right\}$	S

Table 4: Analysis of two clusters contained in a unit interval.

right of D. The additional cluster is only opened if covering the new point with D would result in both D not being able to reach C and D covering a point at a distance greater than two from F, or if D cannot cover it. Hence, if the additional cluster is opened just to the left of D, then D cannot reach C and the distance between the new cluster and E is greater than one; see Figure 46.

On the other hand, if the additional cluster is opened just to the right of D, then D cannot reach E and the distance between C and the new cluster is greater than one. Thus, we get the same result as in Figure 46.

The R-ERG is analyzed in Section 4.2.

Line 16 in Algorithm 5: Opening a New Cluster

We open a new cluster for p; see Figure 47.

The only case that works out directly is the A-in/N-out case.

In order to analyze all the possibilities for G below, we also need to consider the case where G is an R_3 with the far cluster to the right, but the two other clusters not contained in a unit interval; see Figure 48.

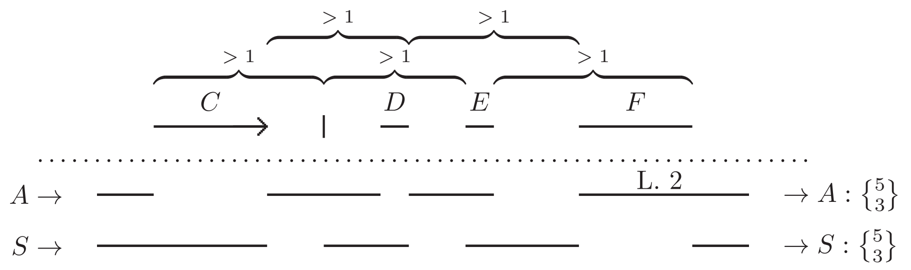


Figure 46: R-ERG: 5 clusters, new cluster to the left of D.

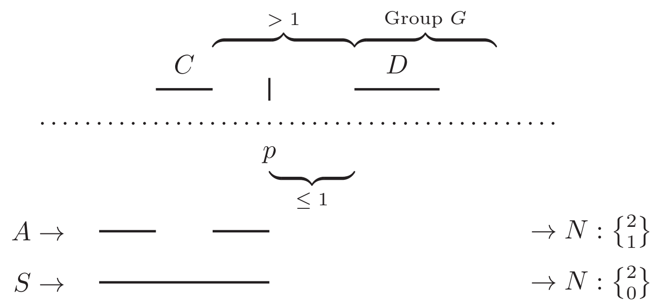


Figure 47: Two clusters contained in a unit interval, p to the right of C.

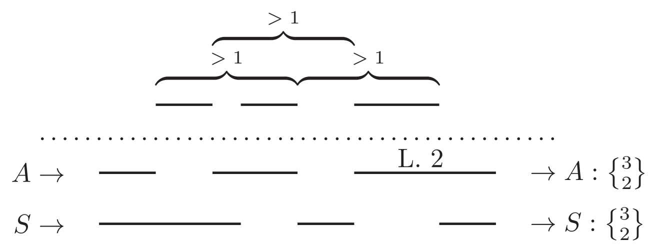


Figure 48: R$_3$ case: Far cluster to the right and not two clusters in a unit interval.

Consider the N-in/N-out case. We now analyze this case together with all possibilities for G. By the algorithm, G cannot be a $(DE)_1\overline{F}$ or a $\overline{D}(EF)_1$ (including the close groups case in Figure 29), Hence, the only remaining cases are R$_3$s where no clusters are contained in a unit interval, and R$_4$s; see Table 5.

All cases work out.

Now, consider the S-in/S-out case. As above, we analyze it together with all possibilities for G. First observe that $S \oplus \left\{{2 \atop 0}\right\} = \left\{{4 \atop 2}\right\}$. Now, consider Table 6.

Again all cases work out.

We now describe how future points are handled: If the new cluster opened for p above can cover a point and still reach D, then it does so. Other points are handled by the regular algorithm. No binding clusters can be created again at this location, since, by Lemma 16, it would have to be C covering a point within a distance one of D, but P would cover such a point.

Next, consider if p is to the left of C; see Figure 49.

Observe that if the cluster just to the left of C could cover p, then it would do so. Hence, the A-in case cannot reach p. The analysis of the N-in/N-out and S-in/N-out cases are identical to the above.

G	$\left\{{a \atop b}\right\}$	$\left\{{2 \atop 1}\right\} \oplus \left\{{a \atop b}\right\}$	Out state
$(DE)_1\overline{F}$ (Figure 48)	$\left\{{3 \atop 3}\right\}$	$\left\{{5 \atop 4}\right\} \sim \left\{{0 \atop 1}\right\}$	S
$(DE)_2\overline{F}$ (Figure 45)	$\left\{{3 \atop 3}\right\}$	$\left\{{5 \atop 4}\right\} \sim \left\{{0 \atop 1}\right\}$	S
R$_4$ (Figure 24)	$\left\{{4 \atop 3}\right\}$	$\left\{{6 \atop 4}\right\} \sim \left\{{1 \atop 1}\right\}$	A

Table 5: Analysis of N-in/N-out.

G	$\left\{\begin{smallmatrix} a \\ b \end{smallmatrix}\right\}$	$\left\{\begin{smallmatrix} 4 \\ 2 \end{smallmatrix}\right\} \oplus \left\{\begin{smallmatrix} a \\ b \end{smallmatrix}\right\}$	Out state
$(DE)_2\overline{F}$ (Figure 48)	$\left\{\begin{smallmatrix} 3 \\ 3 \end{smallmatrix}\right\}$	$\left\{\begin{smallmatrix} 7 \\ 5 \end{smallmatrix}\right\} \sim \left\{\begin{smallmatrix} 2 \\ 2 \end{smallmatrix}\right\}$	S
$\overline{D}(EF)_2$ (Figure 45)	$\left\{\begin{smallmatrix} 3 \\ 3 \end{smallmatrix}\right\}$	$\left\{\begin{smallmatrix} 7 \\ 5 \end{smallmatrix}\right\} \sim \left\{\begin{smallmatrix} 2 \\ 2 \end{smallmatrix}\right\}$	S
R_4 (Figure 24)	$\left\{\begin{smallmatrix} 4 \\ 3 \end{smallmatrix}\right\}$	$\left\{\begin{smallmatrix} 8 \\ 5 \end{smallmatrix}\right\} \sim \left\{\begin{smallmatrix} 3 \\ 2 \end{smallmatrix}\right\}$	A

Table 6: Analysis of S-in/N-out.

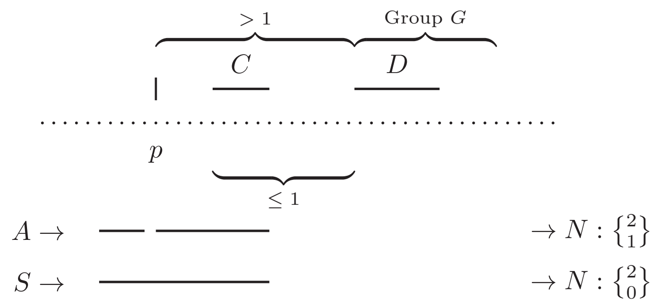

> 1
Group G
C
D
p
≤ 1
$A \rightarrow$
$\rightarrow N : \left\{\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}\right\}$
$S \rightarrow$
$\rightarrow N : \left\{\begin{smallmatrix} 2 \\ 0 \end{smallmatrix}\right\}$

Figure 49: Two clusters contained in a unit interval, p to the left of C.

Finally, we need to consider the situation when there exists a cluster between C and D when C is about to become a binding cluster. It follows from Lemma 16 that p must be to the right of C, and hence the cluster between C and D covers p. The analysis is identical to the analysis above.

This concludes the proof, and we have established the following result:

Theorem 1. There exists a 5/3-competitive deterministic algorithm for one-dimensional online unit clustering.

5. Lower Bound for the Algorithm

The following request sequence shows that the upper bound of $\frac{5}{3}$ established above has a matching lower bound for our algorithm, i.e., the analysis of this algorithm is tight.

$p_1 = 1.25$**:** A new cluster A is opened (Line 13 in Algorithm 1).

$p_2 = 1.75$**:** p_2 is covered by A (Line 11 in Algorithm 1).

$p_3 = 0.5$**:** A new cluster B is opened (Line 13 in Algorithm 1).

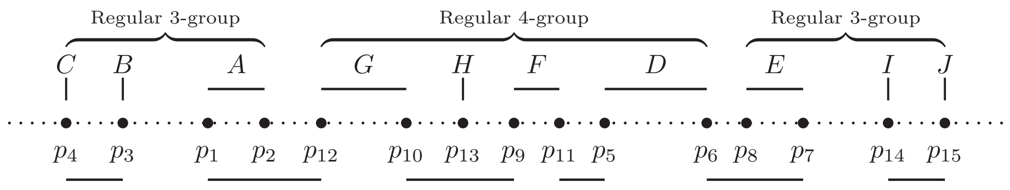


Figure 50: Lower bound for the algorithm.

$p_4 = 0$: A new cluster C is opened and a regular 3-group is created (Line 9 in Algorithm 1).

$p_5 = 4.5$: A new cluster D is opened (Line 13 in Algorithm 1).

$p_6 = 5.5$: p_6 is covered by D (Line 11 in Algorithm 1).

$p_7 = 6.25$: A new cluster E is opened (Line 13 in Algorithm 1).

$p_8 = 5.75$: p_8 is covered by E (Line 11 in Algorithm 1).

$p_9 = 3.75$: A new cluster F is opened (Line 13 in Algorithm 1). Observe that DEF is not a PRG since D cannot reach E or F.

$p_{10} = 2.75$: A new cluster G is opened and an R_3 is created (Line 9 in Algorithm 1).

$p_{11} = 4$: p_{11} is covered by F (Line 4 in Algorithm 2).

$p_{12} = 2$: p_{12} is covered by G (Line 2 in Algorithm 1).

$p_{13} = 3.25$: A new cluster H is opened (Line 6 in Algorithm 2).

$p_{14} = 7$: A new cluster I is opened (Line 13 in Algorithm 1).

$p_{15} = 7.5$: A new cluster J is opened and an R_3 is created (Line 9 in Algorithm 1).

6. Two-dimensional Lower Bound

In this section, we establish the 2-dimensional lower bound which all higher dimensions inherit, as explained fully in the introduction.

Theorem 2. No deterministic on-line algorithm can have a competitive ratio less than $\frac{13}{6}$ in two dimensions.

Proof Observe that it is enough to consider on-line algorithms that never produce clusterings where a cluster is contained in another cluster. Consider any such deterministic on-line algorithm A, and assume by contradiction that it has a competitive ratio less than $\frac{13}{6}$.

First, four points arrive on the corners of a unit square. The algorithm A can either assign them all to one cluster or assign them to two clusters. Otherwise, since there exists a feasible solution using a single cluster, it has ratio of at least three and the input stops.

Points 1–4 in one cluster

If A assigns all points to a single cluster, then four additional points arrive (see the first eight points of Fig. 51). The algorithm A must group at least one of the new pairs in one cluster, since it is possible to group all existing points with only two clusters. Otherwise, A would not stay below the $\frac{13}{6}$-bound.

Points 1–4 in one cluster, points 5–8 in two clusters

If A groups the four points (points 5–8) in exactly two clusters, then eight additional points arrive (see the first 16 points of Fig. 51). Since it is possible to serve all these points with only four clusters, A must use at most five clusters to group the new points.

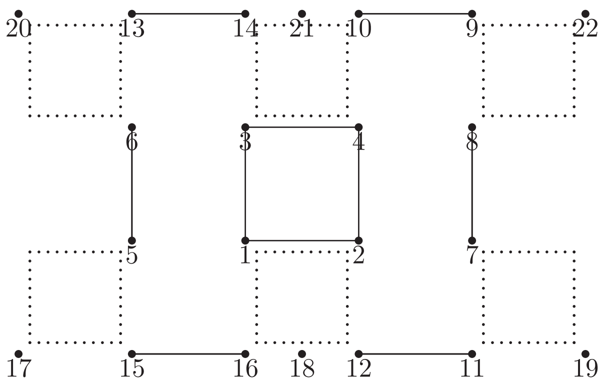


Figure 51: Lower bound.

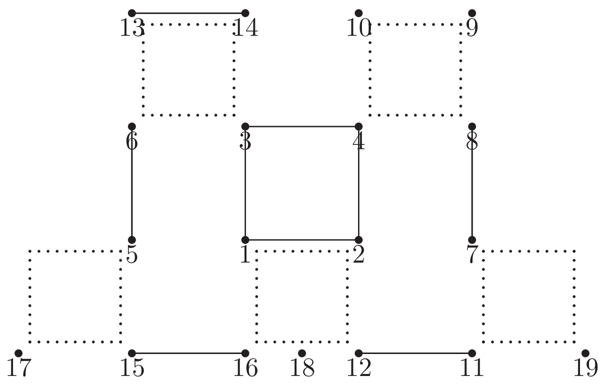


Figure 52: Lower bound.

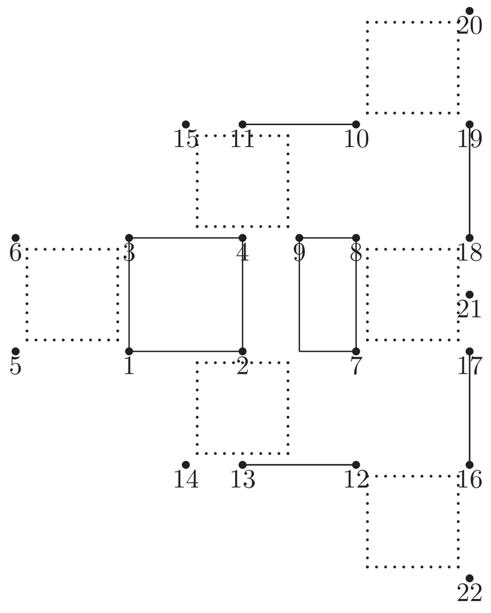


Figure 53: Lower bound.

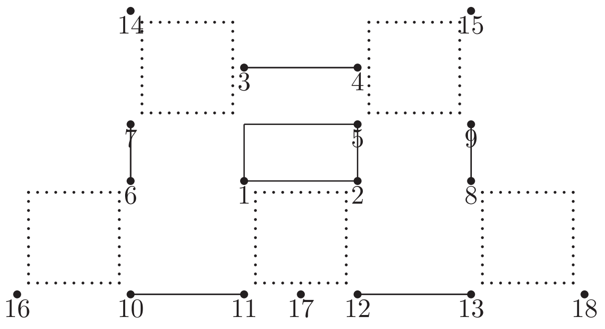


Figure 54: Lower bound.

Points 1–4 in one cluster, points 5–8 in two clusters, points 9–16 in four clusters

If A groups all eight points using four clusters, then six additional points arrive; see Fig. 51. The algorithm A must open six clusters for them and has now used 13 clusters while an optimal solution requires only six clusters, which is a contradiction.

Points 1–4 in one cluster, points 5–8 in two clusters, points 9–16 in five clusters

If A groups the eight points using five clusters, then three additional points arrive; see Fig. 52. The algorithm A must open three clusters for them and has now used 11 clusters while an optimal solution requires only five clusters, which is a contradiction.

Points 1–4 in one cluster, points 5–8 in three clusters

If A groups the four points (points 5–8) using three clusters, then the following sequences of points arrive; see Fig. 53.

First, one additional point arrives (point 9). Since the nine points can be grouped using two clusters A must assign the new point to an already open cluster.

Next, four points arrive (points 10–13). Since the 13 points can be grouped using three clusters A must group the new points into two clusters.

Now, six additional points arrive (points 14–19). Since the 18 points can be grouped using five clusters A must group the new points using four clusters.

Finally, three points arrive (points 20–22). The algorithm A must open three new clusters for them and has now used 13 clusters while an optimal solution only requires six clusters, which is a contradiction.

Points 1-4 in two clusters

If A groups the first four points given using two clusters, then the following sequence of points arrive; see Fig. 54.

First, one additional point arrives (point 5). Since the five points can be grouped using one cluster, A must assign the new point to an already open cluster.

Next, four points arrive (points 6–9). Since, the nine points can be grouped using two clusters A must group the new points using two clusters.

Now, six additional points arrive (points 10–15). Since the 15 points can be grouped using four clusters, A must group the new points using four clusters.

Finally, three points arrive (points 16–18). The algorithm A must open three new clusters for them and has now used 11 clusters while an optimal solution only requires five clusters, which is a contradiction.

In all cases, we reach a contradiction and hence A has a competitive ratio of at least $\frac{13}{6}$. $\square$

7. Open Problems

To us, the most intriguing open problem is the exact competitive ratio of the 1-dimensional unit clustering problem. If the correct answer is not either $\frac{8}{5}$ or $\frac{5}{3}$, it may be quite hard to find the final answer since the only ratios between these two values with enumerators and denominators smaller than 20 are $\frac{13}{8}$ and $\frac{18}{11}$. This indicates that in order to improve either the upper or lower bound to something strictly in between $\frac{8}{5}$ and $\frac{5}{3}$, one would have to consider much larger constructions, with all the combinatorial difficulties that follow.

As discussed in the introduction, it would be interesting to apply the proof technique developed in this paper to clustering problems in general and see if that could lead to upper bound improvements.

Two other interesting open problems are mentioned in [5]. The first question is whether or not the competitive ratio grows with the dimension of the problem, and the second is whether or not randomization can give strictly better results.

Acknowledgment

The authors are greatly indebted to an anonymous referee who made a valuable and enormous effort to improve the presentation of this paper. The authors would also like to thank Leah Epstein for pointing to the lower bound for dimension two as a promising candidate for an improved result.

References

[1] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[2] Timothy M. Chan and Hamid Zarrabi-Zadeh. A randomized algorithm for online unit clustering. *Theory of Computing Systems*, 45(3):486–496, 2009.

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

[4] Leah Epstein, Asaf Levin, and Rob van Stee. Online unit clustering: Variations on a theme. *Theoretical Computer Science*, 407:85–96, 2008.

[5] Leah Epstein and Rob van Stee. On the online unit clustering problem. In *Proceedings of the 5th International Workshop on Approximation and Online Algorithms*, pages 193–206, 2007.

[6] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.

[7] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[8] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[9] Hamid Zarrabi-Zadeh and Timothy M. Chan. An improved algorithm for online unit clustering. *Algorithmica*, 54(4):490–500, 2009.