Algorithmica (2020) 82:938–965
https://doi.org/10.1007/s00453-019-00625-1

Online Clique Clustering

Marek Chrobak[1] · Christoph Dürr[2] · Aleksander Fabijan[3] · Bengt J. Nilsson[3]

Received: 12 October 2016 / Accepted: 27 August 2019 / Published online: 23 September 2019

Abstract
Clique clustering is the problem of partitioning the vertices of a graph into disjoint clusters, where each cluster forms a clique in the graph, while optimizing some objective function. In online clustering, the input graph is given one vertex at a time, and any vertices that have previously been clustered together are not allowed to be separated. The goal is to maintain a clustering with an objective value close to the optimal solution. For the variant where we want to maximize the number of edges in the clusters, we propose an online algorithm based on the doubling technique. It has an asymptotic competitive ratio at most 15.646 and a strict competitive ratio at most 22.641. We also show that no deterministic algorithm can have an asymptotic competitive ratio better than 6. For the variant where we want to minimize the number of edges between clusters, we show that the deterministic competitive ratio of the problem is $n - \omega(1)$, where n is the number of vertices in the graph.

Keywords Online algorithms · Graphs · Cliques · Competitive analysis · Clustering

M. Chrobak: Research supported by NSF Grants CCF-1536026, CCF-0729071 and CCF-1217314.

This paper builds on preliminary results that appeared in conference publications [7,13].

✉ Bengt J. Nilsson
bengt.nilsson.TS@mau.se

Marek Chrobak
marek@cs.ucr.edu

Christoph Dürr
christoph.durr@lip6.fr

Aleksander Fabijan
aleksander.fabijan@mau.se

[1] Department of Computer Science and Engineering, University of California at Riverside, Riverside, USA

[2] Laboratoire d'informatique de Paris 6, LIP6, CNRS, Sorbonne Université, 75252 Paris, France

[3] Department of Computer Science and Media Technology, Malmö University, 205 06 Malmö, Sweden

1 Introduction

The correlation clustering problem and its different variants have been extensively studied over the past decades; see e.g. [1,5,11]. The instance of correlation clustering consists of a graph whose vertices represent some objects and edges represent their similarity. The objective is to find a partitioning of the graph into disjoint subsets called *clusters* that is optimal, or at least near-optimal, with respect to some objective function. Several objective functions are used in the literature, e.g., maximizing the number of edges within the clusters plus the number of non-edges between clusters (maximizing agreements), or minimizing the number of non-edges inside the clusters plus the number of edges outside them (minimizing disagreements). Unlike more conventional approaches to clustering, typically involving some parameter that controls the number or the size of clusters, correlation clustering is parameter-free—the structure of the computed clustering conforms naturally to the similarity function. Bansal et al. [1] show that both the minimization of disagreement edges and the maximization of agreement edges versions are NP-hard. However, from the point of view of approximation the two versions differ. In the case of maximizing agreements, this problem admits a PTAS, whereas in the case of minimizing disagreements it is APX-hard. Several efficient constant factor approximation algorithms are proposed for minimizing disagreements [1,5,11] and maximizing agreements [5].

Another approach to developing parameter-free clustering models is by imposing restrictions on the structure of clusters. We study the variant, called *clique clustering*, where the clusters are required to form disjoint cliques in the underlying graph $G = (V, E)$. Here, we can maximize the number of edges inside the clusters or minimize the number of edges outside the clusters. These measures give rise to the maximum and minimum clique clustering problems, respectively. The computational complexity and approximability of these problems have attracted attention recently [12,15,18], and they have several applications within the areas of gene expression profiling and rDNA clone classification [2,14,18–20]. In the context of rDNA clone classification, for example, a collection of unknown rDNA clones from some environment (bacteria from gut or fungi from soil) are subjected to a sequence of hybridization experiments with appropriately designed primers (short DNA sequences). The signals from these experiments produce a *fingerprint vector* associated with each clone. Very similar fingerprint vectors typically represent rDNA clones of closely related organisms, so clustering of such fingerprint vectors allows one to estimate the level of diversity of a bacterial or fungal community and help with their taxonomic classification. The use of parameter-free clustering is necessitated by the lack of prior information about the environments from which the samples are taken, and clustering into cliques—rather than correlation clustering—reduces the likelihood of "false positives", namely classifying unrelated organisms into one category.

In this paper, we focus on the online variant of clique clustering, where the input graph G is not known in advance. The vertices of G arrive one at a time. Let v_t denote the vertex that arrives at time t, for $t = 1, 2, \ldots$. When v_t arrives, its edges to all preceding vertices $v_1, \ldots, v_{t-1}$ are revealed as well. In other words, after step t, the subgraph of G induced by $v_1, v_2, \ldots, v_t$ is known, but no other information about

G is available. In fact, we assume that even the number n of vertices is not known upfront.

Our objective is to design an online algorithm, namely one that constructs a clustering incrementally, step by step, based on the information acquired up to the current step. Specifically, when v_t arrives at step t, the algorithm first creates a singleton clique $\{v_t\}$. Then it is allowed to merge any number of cliques (possibly none) in its current partitioning into larger cliques. No other modifications of the clustering are allowed. The merge operation in this online setting is irreversible; once vertices are clustered together, they will remain so, and hence, a bad decision may have significant impact on the final solution. This online model was proposed by Charikar et al. [4].

With only limited information about the input sequence and the restrictions on allowed operations, an online clique clustering algorithm cannot be guaranteed to always compute an optimal solution. This is a common feature of most online problems, where information about the input appears gradually over time, and the online algorithm is required to build its solution incrementally, at all times maintaining a valid solution to the already revealed input sequence. As is common in the area of online algorithms, we will measure the performance of an online algorithm by its *competitive ratio*, which represents the ratio between the optimal solution and the solution produced by the algorithm. We distinguish between the *strict competitive ratio*, which is the worst-case such ratio over all possible inputs, and the *asymptotic competitive ratio*, which is (roughly) the limit of such ratios when the optimum value grows to infinity. We define these concepts formally in Sect. 2. We refer the reader to [3] for more background on online problems and competitive analysis.

We emphasize that we place no limits on the computational power of our algorithms. This approach allows us to focus specifically on the limits posed by the lack of complete information about the input. Similar settings have been studied in previous work on online computation, for example for online medians [8,9,16], minimum-latency tours [6], and several other online optimization problems [10], where algorithms with unlimited computational power were studied.

Our approach to online clustering is closely related to that of Mathieu et al. [17], who studiy online correlation clustering. Also in their version, vertices arrive one at a time and clusters need to be built incrementally. They prove that for minimizing disagreements the optimal competitive ratio is $\Theta(n)$, and that it is achieved by a simple greedy algorithm. For maximizing agreements they show that the greedy algorithm is 2-competitive, that a slightly smaller competitive ratio can be achieved with a more sophisticated algorithm, but that no online algorithm can have ratio smaller than 1.199. Interestingly, while the values of the ratios are significantly different, these results parallel those for clique clustering presented in this paper, see below.

Our results. We investigate the online clique clustering problem and provide upper and lower bounds for the competitive ratios for its maximization and minimization versions, that we denote MaxCC and MinCC, respectively.

Section 3 is devoted to the study of MaxCC. We first observe that the competitive ratio of the natural greedy algorithm is linear in n. We then give a constant competitive algorithm for MaxCC, with asymptotic competitive ratio at most 15.646 and strict competitive ratio at most 22.641. The algorithm is based on the doubling technique

often used in online algorithms. We show that the doubling approach cannot give a competitive ratio smaller than 10.927. We also give a general lower bound, proving that there is no online algorithm for MaxCC with competitive ratio smaller than 6. Both these lower bounds apply also to asymptotic ratios.

In Sect. 4 we study online algorithms for MinCC. We prove that no online algorithm can have a competitive ratio of $n - \omega(1)$. We then show that the competitive ratio of the greedy algorithm is $n - 2$, matching this lower bound.

2 Preliminaries

We begin with some notation and basic definitions of the MaxCC and MinCC clustering problems. They are defined on an input graph $G = (V, E)$, with vertex set V and edge set E. We wish to find a partitioning of the vertices in V into clusters so that each cluster induces a clique in G. In addition, we want to optimize some objective function associated with the clustering. In the MaxCC case this objective function is to maximize the total number of edges inside the clusters, whereas in the MinCC case we want to minimize the number of edges outside the clusters.

We adapt the incremental model that was proposed by Charikar et al. [4] and Mathieu et al. [17] for the online correlation clustering problem. Throughout the paper we will implicitly assume that any graph G has its vertices ordered $v_1, v_2, \ldots, v_n$. These vertices arrive one at a time; that is, at step t vertex v_t and all edges between v_t and the previous vertices $v_1, v_2, \ldots, v_{t-1}$ are revealed. At each step the arriving vertex is placed into a new singleton cluster. The resulting clustering can be then updated using any number of *merge* operations, where $merge(C, C')$ merges two existing clusters C, C' into one, provided that the resulting cluster induces a clique in G. This means that once two vertices are clustered together, they cannot be later separated.

For MaxCC, we define the *profit* of a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ of a given graph $G = (V, E)$ to be the total number of edges in its cliques, that is $\sum_{i=1}^{k} \binom{|C_i|}{2} = \frac{1}{2}\sum_{i=1}^{k} |C_i|(|C_i| - 1)$. Similarly, for MinCC, we define the *cost* of $\mathcal{C}$ to be the total number of edges outside its cliques, that is $|E| - \sum_{i=1}^{k} \binom{|C_i|}{2}$. For a graph G, we denote the optimal profit or cost for MaxCC and MinCC, respectively, by $\mathsf{profit}_{\mathsf{OPT}}(G)$ and $\mathsf{cost}_{\mathsf{OPT}}(G)$.

As mentioned earlier, we will measure the performance of an online algorithm by its *competitive ratio*. This ratio is defined as the worst case ratio between the profit/cost of the online algorithm and the profit/cost of an offline optimal algorithm, one that knows the complete input sequence in advance. More formally, for an online algorithm $\mathcal{S}$, we define $\mathsf{profit}_{\mathcal{S}}(G)$ to be the profit of $\mathcal{S}$ when the input graph is $G = (V, E)$ and, similarly, let $\mathsf{cost}_{\mathcal{S}}(G) \stackrel{\text{def}}{=} |E| - \mathsf{profit}_{\mathcal{S}}(G)$ be the cost of $\mathcal{S}$ on G.

We say that an online algorithm $\mathcal{S}$ is *R-competitive* for MaxCC if there is a constant β such that, for any input graph G, we have

$$R \cdot \mathsf{profit}_{\mathcal{S}}(G) + \beta \geq \mathsf{profit}_{\mathsf{OPT}}(G). \tag{1}$$

Similarly $\mathcal{S}$ is *R-competitive* for MinCC if there is a constant β such that, for any input graph G, we have

$$\mathsf{cost}_{\mathcal{S}}(G) \leq R \cdot \mathsf{cost}_{\mathsf{OPT}}(G) + \beta. \tag{2}$$

The reason for defining the competitive ratio differently for maximization and minimization problems is to have all ratios being at least 1. The smallest R for which an online algorithm $\mathcal{S}$ is R-competitive is called the (asymptotic) *competitive ratio* of $\mathcal{S}$. The smallest R for which $\mathcal{S}$ is R-competitive with $\beta = 0$ is called the *strict competitive ratio* of $\mathcal{S}$. (If it so happens that these minimum values do not exist, in both cases the competitive ratio is actually defined by the corresponding infimum.)

Note that an online algorithm does not know when the last vertex arrives and, as a consequence, in order to be R-competitive, it needs to ensure that the corresponding bound, (1) or (2), is valid after each step. To be more precise, for any given step t, inequalities (1) and (2) need to hold for the graph $G = G_t$ induced by vertices $v_1, v_2, \ldots, v_t$. We stress here that we place no restrictions on the optimal solution (in particular, it does not need to be incremental); that is, for any such G_t, the values of $\mathsf{profit}_{\mathsf{OPT}}(G_t)$ and $\mathsf{cost}_{\mathsf{OPT}}(G_t)$ are simply offline optimal solutions computed for the input graph $G = G_t$.

3 Online Maximum Clique Clustering

In this section, we study online MaxCC, the clique clustering problem where the objective is to maximize the number of edges within the cliques. The main results here are upper and lower bounds for the competitive ratio. For the upper bound, we give an algorithm that uses a doubling technique to achieve a competitive ratio of at most 15.646. For the lower bound, we show that no online algorithm has a competitive ratio smaller than 6. Additional results include a competitive analysis of the greedy algorithm and a lower bound for doubling based algorithms.

3.1 The Greedy Algorithm for Online MaxCC

Greedy, the *greedy* algorithm for MaxCC, merges each input vertex with the largest current cluster that maintains the clique property. This maximizes the increase in profit at this step. If no such merging is possible the vertex remains in its singleton cluster. Greedy algorithms are commonly used as heuristics for a variety of online problems and in some cases they produce near-optimal solutions. We show that for MaxCC the solution of Greedy can be far from optimal.

We start with an observation that examines the ratio of Greedy for small values of n; and we follow it with lower and upper bounds for arbitrary values of n.

Observation 1 *For $n = 1, 2, 3$,* Greedy *always finds an optimal clustering.*

This observation is straightforward; it follows by exhaustive verification of a small number of cases, as shown in Fig. 1.

Theorem 1 *For all $n \geq 2$,* Greedy *has competitive ratio at least $\lfloor n/2 \rfloor$ for* MaxCC.

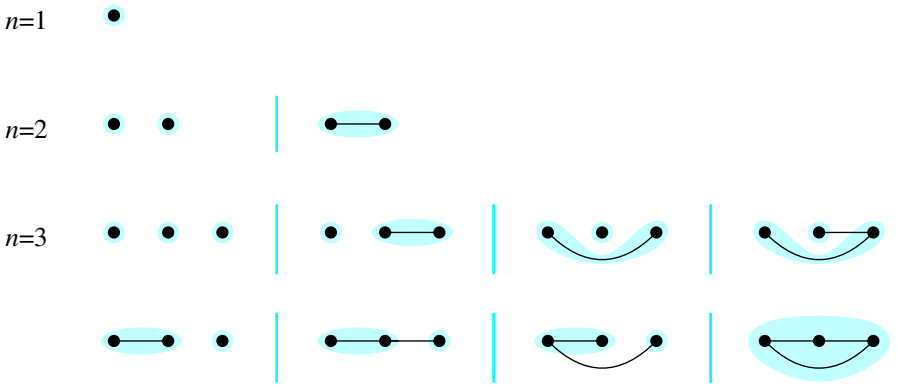

Fig. 1 Greedy finds optimal clusterings for $n = 1, 2, 3$. Vertices are released in order from left to right

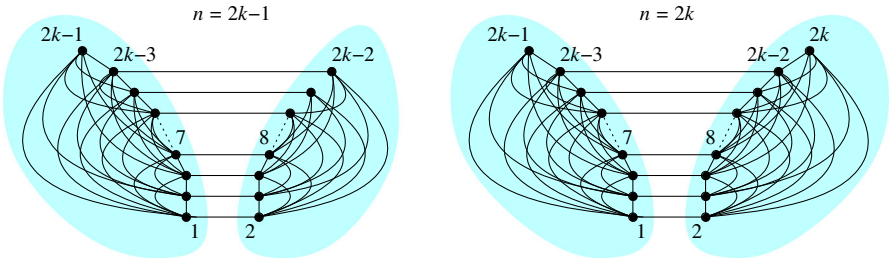

Fig. 2 Illustrating the proof of Theorem 1 for odd and even n

Proof We first give the proof for the strict ratio, and then extend it to the asymptotic ratio. By Observation 1, for $n = 2, 3$, the ratio is 1 and the theorem holds. So we can assume that $n \geq 4$.

Consider an adversary that provides input to the algorithm to make it behave as badly as possible. Our adversary creates an instance with n vertices, numbered from 1 to n. The odd vertices are connected to form a clique, and similarly the even vertices are connected to form a clique. In addition each vertex of the form $2i$, for $i = 1, \ldots, \lfloor (n-1)/2 \rfloor$, is connected to vertex $2i - 1$; see Fig. 2.

GREEDY clusters the vertices as odd/even pairs, leaving the vertex $2k - 1$ as a singleton if $n = 2k - 1$ is odd, and leaving both vertices $2k - 1$ and $2k$ as singletons if $n = 2k$ is even. This generates a clustering of profit $\mathsf{profit}_{\mathsf{GDY}}(G) = k - 1$. An optimal algorithm clusters the odd vertices in one clique of size k and the even vertices in another clique of size $k - 1$ or k, depending on whether n is odd or even. The profit for the optimal solution is $\mathsf{profit}_{\mathsf{OPT}}(G) = (k-1)^2$ if n is odd, and $\mathsf{profit}_{\mathsf{OPT}}(G) = k(k-1)$ if n is even. Hence, the ratio between the optimum and the greedy solution is $k - 1 = (n-1)/2 = \lfloor n/2 \rfloor$ if n is odd, and $k = n/2 = \lfloor n/2 \rfloor$ if n is even; therefore the worst case strict competitive ratio of the greedy algorithm is at least $\lfloor n/2 \rfloor$.

To obtain the same lower bound on the asymptotic ratio, it suffices to notice that, if we follow the above adversary algorithm, then for any $R < \lfloor n/2 \rfloor$ and any constant $\beta > 0$, we can find sufficiently large n for which inequality (1) will be false. □

Next, we look at the upper bound for the greedy algorithm.

Theorem 2 *For all $n \geq 2$,* GREEDY*'s strict competitive ratio for* MAXCC *is at most $\lfloor n/2 \rfloor$.*

Proof By Observation 1, for $n = 2, 3$, the ratio is 1 and the theorem holds. So in the rest of the proof we can assume that $n \geq 4$.

Fix an optimal clustering of G that we denote $\mathsf{OPT}(G)$. Assume this clustering consists of p non-singleton clusters of sizes $c_1, \ldots, c_p$. The profit of $\mathsf{OPT}(G)$ is $\mathsf{profit}_{\mathsf{OPT}}(G) = \frac{1}{2}\sum_{i=1}^{p} c_i(c_i - 1)$. Let $k = \max_i c_i$ be the size of the maximum cluster of $\mathsf{OPT}(G)$.

Case 1: $k \leq \lfloor n/2 \rfloor$. In this case, we can distribute the profit of each cluster of GREEDY equally among the participating vertices; that is, if a vertex belongs to a GREEDY cluster of size c, it will be assigned a profit of $\frac{1}{2}(c-1)$. We refer to this quantity as *charged profit*. We now note that at most one vertex in each cluster of $\mathsf{OPT}(G)$ can be a singleton cluster in GREEDY's clustering, since otherwise GREEDY would cluster any two such vertices together. This gives us that each vertex in a non-singleton cluster of $\mathsf{OPT}(G)$, except possibly for one, has charged profit at least $\frac{1}{2}$. So the total profit charged to the vertices of an $\mathsf{OPT}(G)$ cluster of size c_i is at least $\frac{1}{2}(c_i - 1)$. Therefore the *profit ratio* for this clique of $\mathsf{OPT}(G)$, namely the ratio between its optimal profit and GREEDY's charged profit, is at most

$$\frac{\frac{1}{2}c_i(c_i-1)}{\frac{1}{2}(c_i-1)} = c_i.$$

From this bound and the case assumption, all cliques of $\mathsf{OPT}(G)$ have profit ratio at most $k \leq \lfloor n/2 \rfloor$, so the competitive ratio is also at most $\lfloor n/2 \rfloor$.

Case 2: $k \geq \lfloor n/2 \rfloor + 1$. In this case there is a unique cluster Q in $\mathsf{OPT}(G)$ of size k. The optimum profit is maximized if the graph has one other clique of size $n-k$, so

$$\mathsf{profit}_{\mathsf{OPT}}(G) \;\leq\; \tfrac{1}{2}k(k-1) + \tfrac{1}{2}(n-k)(n-k-1) \;=\; \tfrac{1}{2}\big(n^2 + 2k^2 - 2nk - n\big). \quad (3)$$

We now consider two sub-cases.

Case 2.1: GREEDY's profit is at least k. In this case, using (3) and $k \geq \lfloor n/2 \rfloor + 1 \geq \frac{1}{2}(n+1)$, the competitive ratio is at most

$$\frac{\frac{1}{2}\big(n^2 + 2k^2 - 2nk - n\big)}{k} \;\leq\; \tfrac{1}{2}(n-1) \;\leq\; \lfloor n/2 \rfloor,$$

where the first inequality holds because

$$\big(n^2 + 2k^2 - 2nk - n\big) - k(n-1) = -2\big(k - \tfrac{1}{2}(n-1)\big)(n-k) \leq 0$$

for $\frac{1}{2}(n+1) \leq k \leq n$.

Case 2.2: GREEDY's profit is at most $k-1$. We show that in this case the profit of GREEDY is in fact *equal to* $k-1$, and that GREEDY's clustering has a special form.

To prove this claim, consider those clusters of GREEDY that intersect Q. For $i \geq 1$ and $j \geq 0$, let d_{ij} be the number of these clusters that have i vertices in Q and j outside Q. Note that at most one cluster of GREEDY can be wholly contained in Q, as otherwise GREEDY would merge such clusters. Denote by α the size of this cluster of GREEDY contained in Q (if it exists; if not, let $\alpha = 0$). Let also $\beta = d_{11}$ and

$$\gamma \;=\; \sum_{\substack{i,j\geq 1\\ i+j\geq 3}} i \cdot d_{ij} \;=\; k-\alpha-\beta \;\geq\; 0,$$

where $k = \sum_{i\geq 1, j\geq 0} i \cdot d_{ij}$ counts the number of vertices in Q. The total profit of GREEDY is at least

$$\begin{aligned}
\tfrac{1}{2}\sum_{i\geq 1, j\geq 0}(i+j)(i+j-1)d_{ij} \;&=\; \tfrac{1}{2}\alpha(\alpha-1)+\beta+\tfrac{1}{2}\sum_{\substack{i,j\geq 1\\ i+j\geq 3}}(i+j)(i+j-1)d_{ij}\\
&\geq\; \tfrac{1}{2}\alpha(\alpha-1)+\beta+\tfrac{3}{2}\sum_{\substack{i,j\geq 1\\ i+j\geq 3}} i\cdot d_{ij}\\
&=\; \tfrac{1}{2}\alpha(\alpha-1)+\beta+\tfrac{3}{2}\gamma\\
&=\; k+\tfrac{1}{2}\alpha(\alpha-3)+\tfrac{1}{2}\gamma\\
&\geq\; k-1+\tfrac{1}{2}\gamma.
\end{aligned}$$

The last inequality holds because, for integer values of α, the expression $\alpha(\alpha-3)$ is minimized for $\alpha \in \{1, 2\}$.

Combined with the case assumption that GREEDY's profit is at most $k-1$, we can now conclude that GREEDY's profit is indeed equal to $k-1$ and, in addition, we have that $\gamma = 0$ and $\alpha \in \{1, 2\}$.

So, for $\alpha = 1$, GREEDY's clustering consists of $k-1$ disjoint edges, each with exactly one endpoint in Q, plus a singleton vertex in Q. Thus $n \geq 2k-1$. As $k \geq \lfloor n/2 \rfloor + 1$, this is possible only when $n = 2k-1$. By (3), the optimal profit in this case is at most $(k-1)^2$, so the ratio is at most $k-1 = \lfloor n/2 \rfloor$.

For $\alpha = 2$, GREEDY's clustering consists of $k-1$ edges, of which one is contained in Q and the remaining ones have exactly one endpoint in Q. So $n \geq 2k-2$. If n is odd, this and the bound $k \geq \lfloor n/2 \rfloor + 1$ would force $n = 2k-1$, in which case the argument from the paragraph above applies. On the other hand, if n is even, then these bounds will force $n = 2k-2$. Then, by (3), the optimal profit is k^2-3k+3, so the competitive ratio is at most $(k^2-3k+3)/(k-1) = k-2+1/(k-1) \leq k-1 = \lfloor n/2 \rfloor$, for $k \geq 2$, concluding the proof. □

3.2 A Constant Competitive Algorithm for MaxCC

In this section, we give our competitive online algorithm OCC. Roughly, the algorithm works in phases. In each phase we consider the "batch" of nodes that have not yet been clustered with other nodes, compute an optimal clustering for this batch, and add these new clusters to the algorithm's clustering. The phases are defined so that the profit for consecutive phases increases exponentially.

The overall idea can be thought of as an application of the "doubling" algorithm (see [10], for example), but in our case a subtle modification is required. Unlike other doubling approaches, the phases are not completely independent in our algorithm: the clustering computed in each phase, in addition to the new vertices, needs to include the singleton vertices from earlier phases as well. This is needed, because in our objective function singleton clusters do not bring any profit.

We remark that one could alternatively consider using profit value $k^2/2$ for a clique of size k, which is a very close approximation to our function if k is large. This would lead to a simpler algorithm and much simpler analysis. However, this function is a bad approximation when the clustering involves many small cliques. This is, in fact, the most challenging scenario in the analysis of our algorithm, and instances with this property are also used in the lower bound proof in Sect. 3.4.

3.2.1 Algorithm OCC

We now describe our algorithm. Fix some constant parameter $\gamma > 1$. The algorithm works in phases, starting with phase $j = 0$. At any moment the clustering maintained by the algorithm contains a set U of *singleton* clusters. During phase j, each arriving vertex is added into U. As soon as there is a clustering of U of profit at least γ^j, the algorithm clusters U according to this clustering and adds these new (non-singleton) clusters to its current clustering. The vertices that still form singleton clusters remain in U and then phase $j + 1$ starts.

Note that phase 0 ends as soon as one edge is revealed, since then it is possible for OCC to create a clustering with $\gamma^0 = 1$ edge. The last phase may not be complete; as a result all nodes released in this phase will be clustered as singletons. Observe also that the algorithm never merges non-singleton cliques produced in different phases.

3.2.2 Asymptotic Analysis of OCC

For the purpose of the analysis it is convenient to consider (without loss of generality) only infinite ordered graphs H, whose vertices arrive one at a time in some order $v_1, v_2, \ldots$, and we consider the ratios between the optimum profit and OCC's profit after each step. Furthermore, to make certain that all phases are well-defined, we will assume that the optimum profit for the whole graph H is unbounded. Any finite instance can be converted into an infinite instance with this property by appending to it an infinite sequence of disjoint edges, without decreasing the worst-case profit ratio.

For a given instance (infinite graph) H, define $\mathsf{O}_j(H)$ to be the total profit of the adversary at the end of phase j in the OCC's computation on H. (More precisely, if $v_1, v_2, \ldots, v_t$ are the vertices released in the first j phases, then $\mathsf{O}_j(H)$ is the optimum

offline profit of a clustering of the subgraph H_t of H induced by $v_1, v_2, \ldots, v_t$. See also the last paragraph in Sect. 2.)

Similarly, $\mathsf{S}_j(H)$ denotes the total profit of Algorithm OCC at the end of phase j (including the incremental clustering produced in phase j). During phase 0 the graph is empty, and at the end of phase 0 it consists of only one edge, so $\mathsf{S}_0(H) = \mathsf{O}_0(H) = 1$. For any phase $j > 0$, the profit of OCC is equal to $\mathsf{S}_{j-1}(H)$ throughout the phase, except right after the very last step, when new non-singleton clusters are created. At the same time, the optimum profit can only increase. Thus the maximum ratio in phase j is at most $\mathsf{O}_j(H)/\mathsf{S}_{j-1}(H)$. We can then conclude that, to estimate the competitive ratio of our algorithm OCC, it is sufficient to establish an asymptotic upper bound on numbers R_j, for $j = 1, 2, \ldots$, defined by

$$\mathsf{R}_j \;=\; \max_H \frac{\mathsf{O}_j(H)}{\mathsf{S}_{j-1}(H)}, \tag{4}$$

where the maximum is taken over all infinite ordered graphs H. (While not immediately obvious, the maximum is well-defined. There are infinitely many prefixes of H on which OCC will execute j phases, due to the presence of singleton clusters. However, since these singletons induce an independent set after j phases, only finitely many graphs H need to be considered in this maximum.)

Deriving a recurrence for R_j*'s.* Our objective now is to derive a recurrence relation for the sequence $\mathsf{R}_1, \mathsf{R}_2, \ldots$. The value of R_1 is some constant whose exact value is not important here since we are interested in the asymptotic ratio. (We will, however, estimate R_1 later, when we bound the strict competitive ratio in Sect. 3.2.3.)

So now, fix some $j \geq 2$ and assume that ratios $\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_{j-1}$ are given. We want to bound R_j in terms of $\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_{j-1}$. To this end, let H^* be some infinite graph for which R_j is realized, that is $\mathsf{R}_j = \mathsf{O}_j(H^*)/\mathsf{S}_{j-1}(H^*)$. With H^* fixed, to avoid clutter, we will omit it in our notation, writing $\mathsf{O}_i = \mathsf{O}_i(H^*)$, $\mathsf{S}_i = \mathsf{S}_i(H^*)$, etc., for $i = 1, 2, \ldots, j$. In particular, from the choice of H^*, we have $\mathsf{R}_j = \mathsf{O}_j/\mathsf{S}_{j-1}$.

We claim that, without loss of generality, we can assume that in the computation on H^*, the incremental clusterings of Algorithm OCC in each phase $1, 2, \ldots, j-1$ do not contain any singleton clusters. (The clustering in phase j, however, is allowed to contain singletons.) We will refer to this property as the *No-Singletons Assumption*.

To prove this claim, we modify the ordering of H^* as follows: if there is a phase $i < j$ such that the incremental clustering of U in phase i clusters some vertex v from U as a singleton, then delay the release of v to the beginning of phase $i+1$. Postponing a release of a vertex that was clustered as a singleton in some phase $i < j$ to the beginning of phase $i+1$ does not affect the computation and profit of OCC, because vertices from singleton clusters remain in U, and thus are available for clustering in phase $i+1$. In particular, the value of S_{j-1} will not change. This modification also does not change the value of O_j, because the subgraph of H^* induced by the first j phases is the same, only the ordering of the vertices has been changed. We can thus repeat this process until the No-Singletons Assumption is eventually satisfied. This proves the claim.

After this modification of H^* we still have $\mathsf{O}_j/\mathsf{S}_{j-1} = \mathsf{R}_j$, and of course also $\mathsf{O}_i/\mathsf{S}_{i-1} \le \mathsf{R}_i$ for $i = 1, 2, \ldots, j-1$, from the definition (4) of R_i's. (Note that for $i < j$ the inequality $\mathsf{O}_i/\mathsf{S}_{i-1} \le \mathsf{R}_i$ could be strict, since R_i could be realized by a different graph.)

With the No-Singletons Assumption, the set U is empty at the beginning of each phase $0, 1, \ldots, j$. We can thus divide the vertices of H^* released in phases $0, 1, \ldots, j$ into disjoint *batches*, where batch B_i contains the vertices released in phase i, for $i = 0, 1, \ldots, j$. (At the end of phase i, right before the clustering is updated, we will have $B_i = U$.) For each such i, denote by Δ_i the maximum profit of a clustering of B_i. Then the total profit after i phases is $\mathsf{S}_i = \Delta_0 + \cdots + \Delta_i$, and, by the definition of OCC, we have $\Delta_i \ge \gamma^i$, which implies that $\mathsf{S}_i \ge (\gamma^{i+1} - 1)/(\gamma - 1)$.

For $i = 0, 1, \ldots, j$, let $\bar{B}_i = B_0 \cup \cdots \cup B_i$ be the set of all vertices of H^* released in phases $0, \ldots, i$. Consider the optimal clustering of $\bar{B}_j$. In this clustering, every cluster has some number a of nodes in $\bar{B}_{j-1}$ and some number b of nodes in B_j. For any $a, b \ge 0$, let $k_{a,b}$ be the number of clusters of this form in the optimal clustering of $\bar{B}_j$. Then we have the following bounds, where the sums range over all integers $a, b \ge 0$:

$$\mathsf{O}_j = \sum \binom{a+b}{2} k_{a,b} \tag{5}$$

$$\mathsf{O}_{j-1} \ge \sum \binom{a}{2} k_{a,b} \tag{6}$$

$$\Delta_j \ge \sum \binom{b}{2} k_{a,b} \tag{7}$$

$$\mathsf{S}_{j-1} \ge \tfrac{1}{2} \sum a k_{a,b} \tag{8}$$

Equality (5) is the definition of O_j. Inequality (6) holds because the right hand side represents the profit of the optimal clustering of $\bar{B}_j$ restricted to $\bar{B}_{j-1}$, so it cannot exceed the optimal profit O_{j-1} for $\bar{B}_{j-1}$. Similarly, inequality (7) holds because the right hand side is the profit of the optimal clustering of $\bar{B}_j$ restricted to B_j, while Δ_j is the optimal profit of B_j. The last bound (8) follows from the fact that (as a consequence of the No-Singletons Assumption) our algorithm does not have any singleton clusters in $\bar{B}_{j-1}$. This means that in OCC's clustering of $\bar{B}_{j-1}$ (which has $\sum a k_{a,b}$ vertices) each vertex has an edge included in some cluster, so the number of these edges must be at least $\frac{1}{2} \sum a k_{a,b}$.

We can also bound the algorithm's profit values Δ_i and S_i, for $0 \le i \le j$, from above. (Why we need upper bounds for the algorithm's profit will be seen shortly.) We have $\Delta_0 = 1$ and for each phase $i \ge 1$,

$$\Delta_i \;\le\; \gamma^i + \tfrac{1}{2}\left(\sqrt{8\gamma^i + 1} + 1\right) \;<\; \gamma^i + \sqrt{2}\gamma^{j/2} + 2 - \sqrt{2}. \tag{9}$$

To show (9), suppose that phase i ends at step t (that is, right after v_t is revealed). Consider the optimal partitioning $\mathcal{P}$ of B_i, and let the cluster c containing v_t in $\mathcal{P}$ have size $p+1$. If we remove v_t from this partitioning, we obtain a partitioning $\mathcal{P}'$ of

the batch after step $t-1$, whose profit must be strictly smaller than γ^i. So the profit of $\mathcal{P}$ is smaller than $\gamma^i + p$. In partitioning $\mathcal{P}'$, the cluster $c - \{v_t\}$ has size p. We thus obtain that $\binom{p}{2} < \gamma^i$, because, in the worst case, $\mathcal{P}$ consists only of cluster c. This gives us $p < \frac{1}{2}(\sqrt{8\gamma^i + 1} + 1)$. The second inequality in (9) follows by routine calculation.

From (9), by adding up all profits from phases $0, \ldots, i$, we obtain an upper bound on the total profit of the algorithm:

$$\mathsf{S}_i < \frac{\gamma^{i+1} - 1}{\gamma - 1} + \sqrt{2} \cdot \frac{\gamma^{(i+1)/2} - \gamma^{1/2}}{\gamma^{1/2} - 1} + \big(2 - \sqrt{2}\big)i + 1. \tag{10}$$

Lemma 3.1 *For any pair of non-negative integers a and b, the inequality*

$$\binom{a+b}{2} \le (x+1)\binom{a}{2} + \frac{x+1}{x}\binom{b}{2} + a$$

holds for any $0 < x \le 1$.

Proof Define the function

$$\begin{aligned} F(a,b,x) &= 2x(x+1)\binom{a}{2} + 2(x+1)\binom{b}{2} + 2ax - 2x\binom{a+b}{2} \\ &= a^2x^2 - ax^2 + 2ax + b^2 - b - 2abx \\ &= (b - ax)^2 + ax(2-x) - b, \end{aligned}$$

i.e., $2x$ times the difference between the right hand side and the left hand side of the inequality above. It is sufficient to show that $F(a,b,x)$ is non-negative for integers $a, b \ge 0$ and $0 < x \le 1$.

Consider first the cases when $a \in \{0,1\}$ or $b \in \{0,1\}$. $F(0,b,x) = b(b-1) \ge 0$, for any non-negative integer b and any x. $F(a,0,x) = ax(ax-x+2) \ge ax(ax+1) > 0$, for any positive integer a and $0 < x \le 1$. $F(a,1,x) = x^2a(a-1) \ge 0$, for any positive integer a and any x. $F(1,2,x) = 2 - 2x \ge 0$, for $0 < x \le 1$, and $F(1,b,x) = b^2 - b + 2x - 2bx \ge b^2 - 3b \ge 0$, for any integer $b \ge 3$ and $0 < x \le 1$.

Thus, it only remains to show that $F(a,b,x)$ is non-negative when both $a \ge 2$ and $b \ge 2$. The function $F(a,b,x)$ is quadratic in x and hence has one local minimum at $x_0 = \frac{b-1}{a-1}$, as can be easily verified by differentiating F in x. Therefore, in the case when $a \le b$, $F(a,b,x) \ge F(a,b,1) = (b-a)^2 - (b-a) \ge 0$, for $0 < x \le 1$. In the case when $a > b$, we have that $F(a,b,x) \ge F(a,b,\frac{b-1}{a-1}) = \frac{(a-b)(b-1)}{a-1} > 0$, which completes the proof. □

We now combine all estimates derived above to establish our recurrence. Fix some parameter x, $0 < x < 1$, whose value we will determine later. Using Lemma 3.1, the bounds (5)–(8), and the definition of R_{j-1}, we obtain

$$\mathsf{R}_j\mathsf{S}_{j-1} = \mathsf{O}_j = \sum \binom{a+b}{2} k_{a,b}$$

$$\leq (x+1)\sum\binom{a}{2}k_{a,b} + \frac{x+1}{x}\sum\binom{b}{2}k_{a,b} + \sum ak_{a,b}$$
$$\leq (x+1)\mathsf{O}_{j-1} + \frac{x+1}{x}\Delta_j + 2\mathsf{S}_{j-1}$$
$$\leq (x+1)\mathsf{R}_{j-1}\mathsf{S}_{j-2} + \frac{x+1}{x}\Delta_j + 2\mathsf{S}_{j-1}. \tag{11}$$

(Recall that $j \geq 2$.) Thus R_j satisfies the inequality

$$\mathsf{R}_j \leq \frac{x+1}{x\mathsf{S}_{j-1}}\left[x\mathsf{S}_{j-2}\mathsf{R}_{j-1} + \Delta_j\right] + 2. \tag{12}$$

From inequalities (9) and (10), we have

$$\Delta_i = \gamma^i(1+o(1)) \quad \text{and} \quad \mathsf{S}_i = \frac{\gamma^{i+1}(1+o(1))}{\gamma-1}.$$

for all $i = 0, 1, \ldots, j$. Above, we use the notation $o(1)$ to denote any function that tends to 0 as the phase index i goes to infinity (with x and γ assumed to be some fixed constants, still to be determined). Substituting into inequality (12), we obtain our recurrence for the numbers R_j:

$$\mathsf{R}_j \leq \left(\frac{x+1}{\gamma} + o(1)\right)\cdot \mathsf{R}_{j-1} + \frac{(x+1)(\gamma-1)}{x} + 2 + o(1). \tag{13}$$

Solving recurrence. (13) Define

$$\mathsf{R} = \frac{\gamma(\gamma x + x + \gamma - 1)}{x(\gamma - x - 1)}. \tag{14}$$

Lemma 3.2 *If $x + 1 < \gamma$ then $R_j = R + o(1)$.*

Proof The proof is by routine calculus, so we only provide a sketch. For all $j \geq 1$ let $\rho_j = \mathsf{R}_j - \mathsf{R}$. Then, substituting this into (13) and simplifying, we obtain that the ρ_j's satisfy the recurrence

$$\rho_j \leq \left(\frac{x+1}{\gamma} + o(1)\right)\cdot \rho_{j-1} + o(1). \tag{15}$$

Since $x + 1 < \gamma$, this implies that $\rho_j = o(1)$, and the lemma follows. □

Lemma 3.2 gives us (essentially) a bound of R on the asymptotic competitive ratio of Algorithm OCC, for fixed values of parameters γ (of the algorithm) and x (of the analysis). We can now choose γ and x to make R as small as possible. R is minimized for parameters $x = \frac{1}{2}(5 - \sqrt{13}) \approx 0.697$ and $\gamma = \frac{1}{2}(3 + \sqrt{13}) \approx 3.303$, yielding

$$\mathsf{R} = \tfrac{1}{6}(47 + 13\sqrt{13}) \approx 15.646.$$

Using Lemma 3.2, for each infinite graph H and phase j, we have that $\mathsf{O}_j(H) \leq (\mathsf{R} + o(1))\mathsf{S}_{j-1}(H)$. Since, in fact, $\mathsf{R} < 15.646$, this implies that $\mathsf{O}_j(H) \leq 15.646 \cdot \mathsf{S}_{j-1}(H)$, as long as j is large enough. Thus $\mathsf{O}_j(H) \leq 15.646 \cdot \mathsf{S}_{j-1}(H) + O(1)$ for all phases j. As we discussed earlier, bounding $\mathsf{O}_j(H)$ in terms of $\mathsf{S}_{j-1}(H)$ like this is sufficient to establish a bound on the (asymptotic) competitive ratio of Algorithm OCC. Summarizing, we obtain the following theorem.

Theorem 3 *The asymptotic competitive ratio of Algorithm OCC is at most* 15.646.

3.2.3 Strict Competitive Ratio

In fact, for $\gamma = \frac{1}{2}(3 + \sqrt{13})$, Algorithm OCC has a low strict competitive ratio as well. We show that this ratio is at most 22.641. The argument uses the same value of parameter $x = \frac{1}{2}(5 - \sqrt{13})$, but requires a more refined analysis.

When phase 0 ends, the competitive ratio is 1. For $j \geq 1$, let O'_j be the optimal profit right before phase j ends, that is before the last vertex of phase j is released. (Earlier we used O_j to upper bound this value, but this is a loose bound because it also includes the profit for the last step of phase j.) It remains to show that for phases $j \geq 1$ we have $\mathsf{R}'_j \leq 22.641$, where $\mathsf{R}'_j = \mathsf{O}'_j/\mathsf{S}_{j-1}$.

The outline of the argument is as follows. By exhaustively analyzing the behavior of Algorithm OCC in phase 1, taking into account that $\gamma \approx 3.303 > 3$, we can establish that $\mathsf{R}'_1 = 10$. We will then bound the remaining ratios using a refined version of recurrence (12).

We start by establishing the maximum value of $\mathsf{R}'_1 = \mathsf{O}'_1/\mathsf{S}_0 = \mathsf{O}'_1$. Let t be the last step of phase 1. By the construction of the algorithm, since $\gamma \approx 3.303$, after step $t-1$ the profit of the vertices released in phase 1 is at most 3. We can assume that phase 0 has only two vertices v_1, v_2 connected by an edge. Let H_t be the subgraph of H induced by $v_1, \ldots, v_{t-1}$ and H'_t be its subgraph induced by $v_3, \ldots, v_{t-1}$. Thus R'_1 is equal to the maximum value (over all choices of H) of the optimal profit of H_t under the assumption that the optimal profit of H'_t is at most 3. We now proceed to bound this value.

Denote by $\mathbf{K}_i$ the clique with i vertices. The optimal clustering of H'_t cannot include a $\mathbf{K}_4$, and either

1. H'_t has no $\mathbf{K}_3$, and it has at most three $\mathbf{K}_2$'s, or
2. H'_t has a $\mathbf{K}_3$, with each edge of H'_t having at least one endpoint in this $\mathbf{K}_3$.

In Case 1, H_t cannot contain a $\mathbf{K}_5$. If a clustering of H_t includes a $\mathbf{K}_4$ then this $\mathbf{K}_4$ contains v_1, v_2, and two vertices from phase 1. So, in addition to this $\mathbf{K}_4$ it can at best include two $\mathbf{K}_2$'s, for a total profit of at most 8. In Case 2, if a clustering of H_t includes a $\mathbf{K}_5$, then it cannot include any cluster except this $\mathbf{K}_5$, so its profit is 10. If a clustering of H_t includes a $\mathbf{K}_4$, then this $\mathbf{K}_4$ must contain at least one of v_1 and v_2, and it may include at most one other clique of type $\mathbf{K}_2$. This will give a total profit of at most 7. Summarizing, in each case the profit of H_t is at most 10 giving us $\mathsf{R}'_1 \leq 10$, as claimed.

Table 1 Some initial bounds for S_j and the strict competitive ratio

j	0	1	2	3	4	5	6	7	8
min S_j	1	5	16	53	172	566	1864	6152	20311
max S_j	1	7	23	68	202	623	1972	6352	20679
R'_j	1.00	10.00	13.18	18.63	21.88	**22.64**	21.51	19.92	18.50

Maximum bound value is highlighted in bold

For phases $j \geq 2$, we can tabulate upper bounds for R'_j by explicitly computing the ratios $\mathsf{R}'_j = \mathsf{O}'_j/\mathsf{S}_{j-1}$ using the following modification of recurrence (12),

$$\mathsf{R}'_j \leq \frac{x+1}{x\mathsf{S}_{j-1}}\left[x\mathsf{S}_{j-2}\mathsf{R}'_{j-1} + \Delta_j\right] + 2, \tag{16}$$

where we use the more exact bounds

$$\lceil \gamma^i \rceil \leq \Delta_i \leq \lfloor \gamma^i + \tfrac{1}{2}(\sqrt{8\gamma^i + 1} + 1)\rfloor,$$

obtained by rounding the bounds $\Delta_i \geq \gamma^i$ and (9), which we can do because Δ_i is integral. From the definition of $\mathsf{S}_i \stackrel{\text{def}}{=} 1 + \sum_{i'=1}^{i} \Delta_{i'}$, we compute the first few estimates as shown in Table 1.

To bound the sequence $\{\mathsf{R}'_j\}_{j\geq 9}$ we rewrite recurrence (16) as

$$\mathsf{R}'_j \leq \frac{(x+1)\mathsf{S}_{j-2}}{\mathsf{S}_{j-1}} \cdot \mathsf{R}'_{j-1} + \frac{(x+1)\Delta_j}{x\mathsf{S}_{j-1}} + 2 = \alpha_j \mathsf{R}'_{j-1} + \beta_j,$$

and bound α_j and β_j using (9) and (10). With routine calculations, we can establish the bounds $\alpha_j < \frac{3}{5}$ and $\beta_j < 8$, for $j \geq 8$.

Therefore $\mathsf{R}'_j \leq \hat{\mathsf{R}}_j$, where $\hat{\mathsf{R}}_j$ is

$$\hat{\mathsf{R}}_j = \tfrac{3}{5}\hat{\mathsf{R}}_{j-1} + 8 \leq 20 - a\left(\tfrac{3}{5}\right)^j,$$

for $j \geq 8$ and some positive constant a. The sequence $\{\hat{\mathsf{R}}_j\}_{j\geq 9}$, is thus bounded above by a monotonically growing function of j having limit 20 and hence $\hat{\mathsf{R}}_j \leq 20$ for every $j \geq 9$.

Combining this with the bounds estimated in Table 1, we see that the largest bound on R'_j is 22.641 given for $j = 5$. We can thus conclude that the strict competitive ratio of OCC is at most 22.641.

We can improve on the strict competitive ratio by choosing different values for γ and x that allow the asymptotic competitive ratio to increase slightly. The optimal values can be found empirically (using mathematical software) to be $\gamma = 4.02323428$ and $x = 0.823889$, giving asymptotic competitive ratio 15.902 and strict competitive ratio 20.017.

3.3 A Lower Bound for Algorithm OCC

In this section we will show that, for any choice of γ, the worst-case ratio of Algorithm OCC is at least 10.927.

Denote by B_j the j-th batch, that is the vertices released in phase j. We will use notation S_j for the profit of OCC and O_j for the optimal profit on the sub-instance consisting of the first j batches. To avoid clutter we will omit lower order terms in our calculations. In particular, we focus on j being large enough, treating γ^j as integer, and all estimates for S_j and O_j given below are meant to hold within a factor of $1 \pm o(1)$. (The asymptotic notation is with respect to the phase index j tending to ∞.)

We start with a simpler construction that shows a lower bound of 9; then we will explain how to improve it to 10.927. In the instance we construct, all batches will be disjoint, with the jth batch B_j having $2\gamma^j$ vertices connected by γ^j disjoint edges (that is, a perfect matching). We will refer to these edges as *batch* edges. The edges between any two batches B_i and B_j, for $i < j$, form a complete bipartite graph. These edges will be called *cross* edges; see Fig. 3.

At the end of each phase j, the algorithm will collect all γ^j edges inside B_j. Therefore, by summing up the geometric sequence, right before the end of phase j (before the algorithm adds the new edges from B_j to its clustering), the algorithm's profit is

$$\mathsf{S}_{j-1} = \sum_{i=0}^{j-1} \gamma^i \leq \frac{\gamma^j}{\gamma - 1}.$$

After the first j phases, the adversary's clustering consists of cliques C_p, $p = 0, 1, \ldots, \gamma^j - 1$, where C_p contains the p-th edge (that is, its both endpoints) from each batch B_i for $i = p, p+1, \ldots, j$; see Fig. 3. We claim that the adversary gain after j phases satisfies

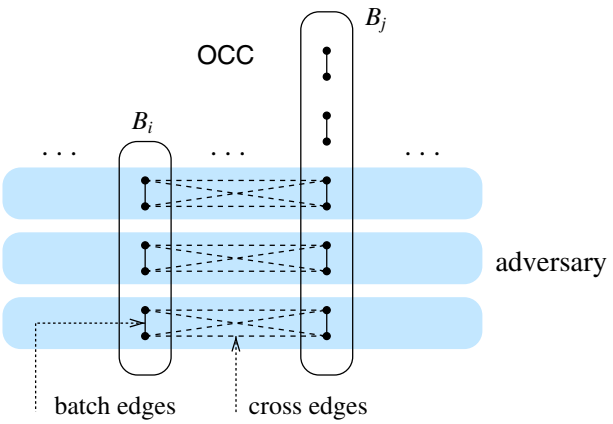

Fig. 3 The lower bound example for Algorithm OCC. The figure shows two batches B_i and B_j, for $i < j$. Batch edges, drawn with solid lines, are collected by Algorithm OCC. Dashed lines show cross edges that are in the adversary's clustering. Shaded regions illustrate the cliques in the adversary's clustering

$$\mathsf{O}_j \geq \mathsf{O}_{j-1} + \gamma^j + 4\sum_{i=0}^{j-1} \gamma^i = \mathsf{O}_{j-1} + \frac{(\gamma+3)\gamma^j}{\gamma-1}. \tag{17}$$

[Recall that all equalities and inequalities in this section are assumed to hold only within a factor of $1 \pm o(1)$.] We now justify this bound. The second term γ^j is simply the number of batch edges in B_j. To see where each term $4\gamma^i$ comes from, consider the p-th batch edge from B_i, for $i < j$. When we add B_j after phase j, the adversary can add the 4 cross edges connecting this edge's endpoints to the endpoints of the pth batch edge in B_j to C_p. Overall, this will add $4\gamma^i$ cross edges between B_i and B_j to the existing adversary's cliques.

From recurrence (17), by simple summation, we get

$$\mathsf{O}_j \geq \frac{(\gamma+3)\gamma^{j+1}}{(\gamma-1)^2}.$$

Dividing it by OCC's profit of at most $\gamma^j/(\gamma-1)$, we obtain that the ratio is at least $\frac{\gamma(\gamma+3)}{\gamma-1}$, which, by routine calculus, is at least 9.

We now outline an argument showing how to improve this lower bound to 10.927. The new construction is almost identical to the previous one, except that we change the very last batch B_j. As before, each batch B_i, for $i < j$, has γ^i disjoint edges. Batch B_j will also have γ^j edges, but they will be grouped into $q = \frac{1}{3}\gamma^j$ disjoint triangles. (So B_j has γ^j vertices.) For $p = 0, 1, \ldots, q-1$, we add the p-th triangle to clique C_p. (If $q > \gamma^{j-1}$, the last $q - \gamma^{j-1}$ triangles will form new cliques.)

This modification will preserve the number of edges in B_j and thus it will not affect the algorithm's profit. But now, for each $i = 0, 1, \ldots, j-1$ and each $p = 0, 1, \ldots, \min(q, \gamma^i) - 1$, we can connect the two vertices in $B_i \cap C_p$ to three vertices in B_j, instead of two. This creates two new cross edges that will be called *extra* edges. It should be intuitively clear that the number of these extra edges is $\Omega(\gamma^j)$, which means that this new construction gives a ratio strictly larger than 9.

Specifically, to estimate the ratio, we will distinguish three cases, depending on the value of γ. Suppose first that $\gamma \geq 3$. Then $q \geq \gamma^{j-1}$, so the number of extra edges is $2\sum_{i=0}^{j-1} \gamma^i = 2\gamma^j/(\gamma-1)$, because each vertex in $B_0 \cup B_1 \cup \cdots \cup B_{j-1}$ is now connected to three vertices in B_j, not two. Thus the new optimal profit is

$$\mathsf{O}'_j = \mathsf{O}_j + \frac{2\gamma^j}{\gamma-1} = \frac{(\gamma^2+5\gamma-2)\gamma^j}{(\gamma-1)^2}.$$

Dividing by OCC's profit, the ratio is at least $\frac{\gamma^2+5\gamma-2}{\gamma-1}$, which is at least 11 for $\gamma \geq 3$.

The second case is when $\sqrt{3} \leq \gamma \leq 3$. Then $\gamma^{j-2} \leq q \leq \gamma^{j-1}$. In this case all vertices in $B_0 \cup B_1 \cup \cdots \cup B_{j-2}$ and $\frac{2}{3}\gamma^j$ vertices in B_{j-1} get an extra edge, so the number of extra edges is $2\gamma^{j-1}/(\gamma-1) + \frac{2}{3}\gamma^j$. Therefore the new adversary profit is

$$\mathsf{O}'_j = \mathsf{O}_j + 2\,\frac{\gamma^{j-1}}{\gamma-1} + \tfrac{2}{3}\gamma^j = \frac{(5\gamma^3+5\gamma^2+8\gamma-6)\gamma^{j-1}}{3(\gamma-1)^2}.$$

We thus have that the ratio is at least $\frac{5\gamma^3+5\gamma^2+8\gamma-6}{3\gamma(\gamma-1)}$. Minimizing this quantity, we obtain that the ratio is at least 10.927.

The last case is when $1 < \gamma \leq \sqrt{3}$. In this case, even using the earlier construction (without any extra edges), we have that the ratio $\mathsf{O}_j/\mathsf{S}_{j-1} = \frac{\gamma(\gamma+3)}{\gamma-1}$ is at least $3 + 6\sqrt{3} \approx 11.2$ (it is minimized for $\gamma = \sqrt{3}$).

3.4 A Lower Bound of 6 for MaxCC

We now prove that any deterministic online algorithm $\mathcal{S}$ for the clique clustering problem has competitive ratio at least 6. We present the proof for the strict competitive ratio and explain later how to extend it to the asymptotic ratio. The lower bound is established by showing, for any constant $R < 6$, an adversary algorithm for constructing an input graph G on which $\mathsf{profit}_{\mathsf{OPT}}(G) \geq R \cdot \mathsf{profit}_{\mathcal{S}}(G)$, that is the optimal profit is at least R times the profit of $\mathcal{S}$.

Skeleton trees. Fix some non-negative integer D. (Later we will make the value of D depend on R.) It is convenient to describe the graph constructed by the adversary in terms of its underlying *skeleton tree* $\mathcal{T}$, which is a rooted binary tree. The root of $\mathcal{T}$ will be denoted by r. For a node $v \in \mathcal{T}$, define the *depth* of v to be the number of edges on the simple path from v to r. The adversary will only use skeleton trees of the following special form: each non-leaf node at depths $0, 1, \ldots, D-1$ has two children, and each non-leaf node at levels at least D has one child. Such a tree $\mathcal{T}$ can be thought of as consisting of its *core subtree*, which is the subtree of $\mathcal{T}$ induced by the nodes of depth up to D, with paths attached to its leaves at level D. The nodes of $\mathcal{T}$ at depth D are the leaves of the core subtree. If v is a leaf of the core subtree of $\mathcal{T}$ then the path extending from v down to a leaf of $\mathcal{T}$ is called a *tentacle*—see Fig. 4. (Thus v belongs both to the core subtree and to the tentacle attached to v.) The length of a tentacle is the number of its edges. The nodes in the tentacles are all considered to be left children of their parents.

Skeleton-tree graphs. The graph represented by a skeleton tree $\mathcal{T}$ will be denoted by $\mathcal{G}$. We differentiate between the *nodes* of $\mathcal{T}$ and the *vertices* of $\mathcal{G}$. The relation between $\mathcal{T}$ and $\mathcal{G}$ is illustrated in Fig. 4. The graph $\mathcal{G}$ is obtained from the tree $\mathcal{T}$ as follows:

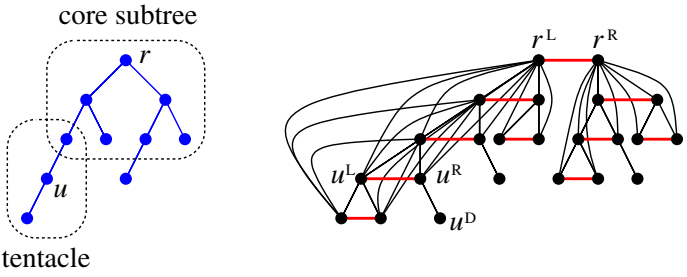

Fig. 4 On the left, an example of a skeleton tree $\mathcal{T}$. The core subtree of $\mathcal{T}$ has depth 2 and two tentacles, one of length 2 and one of length 1. On the right, the corresponding graph $\mathcal{G}$

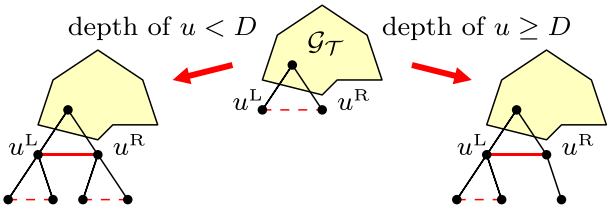

Fig. 5 Adversary moves. Upward edges from new vertices are not shown, to avoid clutter. Dashed lines represent cross edges that are not collected by $\mathcal{S}$, while thick lines represent those that are already collected by $\mathcal{S}$

- For each node $u \in \mathcal{T}$ we create two vertices u^{L} and u^{R} in $\mathcal{G}$, with an edge between them. This edge $(u^{\mathrm{L}}, u^{\mathrm{R}})$ is called the *cross edge* corresponding to u.
- Suppose that $u, v \in \mathcal{T}$. If u is in the left subtree of v then $(u^{\mathrm{L}}, v^{\mathrm{L}})$ and $(u^{\mathrm{R}}, v^{\mathrm{L}})$ are edges of $\mathcal{G}$. If u is in the right subtree of v then $(u^{\mathrm{L}}, v^{\mathrm{R}})$ and $(u^{\mathrm{R}}, v^{\mathrm{R}})$ are edges of $\mathcal{G}$. These edges are called *upward edges*.
- If $u \in \mathcal{T}$ is a node in a tentacle of $\mathcal{T}$ and is not a leaf of $\mathcal{T}$, then $\mathcal{G}$ has a vertex u^{D} with edge $(u^{\mathrm{D}}, u^{\mathrm{R}})$. This edge is called a *whisker*.

The adversary algorithm. The adversary constructs $\mathcal{T}$ and $\mathcal{G}$ gradually, in response to algorithm $\mathcal{S}$'s choices. Initially, $\mathcal{T}$ is a single node r, and thus $\mathcal{G}$ is a single edge $(r^{\mathrm{L}}, r^{\mathrm{R}})$. At this time, $\mathsf{profit}_{\mathcal{S}}(\mathcal{T}) = 0$ and $\mathsf{profit}_{\mathsf{OPT}}(\mathcal{T}) = 1$, so $\mathcal{S}$ is forced to collect this edge (that is, it creates a 2-clique $\{r^{\mathrm{L}}, r^{\mathrm{R}}\}$), since otherwise the adversary can immediately stop with unbounded strict competitive ratio.

In general, the invariant of the construction is that, at each step, the only non-singleton cliques that $\mathcal{S}$ can add to its clustering are cross edges that correspond to the current leaves of $\mathcal{T}$. Suppose that, at some step, $\mathcal{S}$ collects a cross edge $(u^{\mathrm{L}}, u^{\mathrm{R}})$, corresponding to node u of $\mathcal{T}$. ($\mathcal{S}$ may collect more cross edges in one step; if so, the adversary applies its algorithm to each such edge independently.) If u is at depth less than D, the adversary extends $\mathcal{T}$ by adding two children of u. If u is at depth at least D, the adversary only adds the left child of u, thus extending the tentacle ending at u. In terms of $\mathcal{G}$, the first move appends two triangles to u^{L} and u^{R}, with all corresponding upward edges. The second move appends a triangle to u^{L} and a whisker to u^{R} (see Fig. 5). In the case when $\mathcal{S}$ decides not to collect any cross edges at some step, the adversary stops the process.

Thus, the adversary will be building the core binary skeleton tree down to depth D, and from then on, if the game still continues, it will extend the tree with tentacles. Our objective is to prove that, in each step, right after the adversary extends the graph but before $\mathcal{S}$ updates its clustering, we have

$$\mathsf{profit}_{\mathsf{OPT}}(\mathcal{T}) \geq (6 - \epsilon_D) \cdot \mathsf{profit}_{\mathcal{S}}(\mathcal{T}), \tag{18}$$

where $\epsilon_D \to 0$ when $D \to \infty$. This is enough to prove the lower bound of $6 - \epsilon_D$ on the strict ratio. The reason is this: If $\mathcal{S}$ does not collect any edges at some step, the game stops, the ratio is $6 - \epsilon_D$, and we are done. Otherwise, the adversary will stop the game after $2^{D+1} + M$ steps, where M is some large integer. Then the profit of $\mathcal{S}$ is bounded by $2^{D+1} + M$ (the number of steps) plus the number of remaining cross

edges, and there are at most 2^D of those, so $\mathcal{S}$'s profit is at most $2^{D+2} + M$. At that time, $\mathcal{T}$ will have at least M nodes in tentacles and at most 2^D tentacles, so there is at least one tentacle of length $M/2^D$, and this tentacle contributes $\Omega((M/2^D)^2)$ edges to the optimum. Thus for M large enough, the ratio between the optimal profit and the profit of $\mathcal{S}$ will be larger than 6 (or any constant, in fact).

Once we establish (18), the lower bound of 6 will follow, because for any fixed $R < 6$ we can take D to be large enough to achieve a lower bound of $6 - \epsilon_D \geq R$.

Computing the adversary's profit. We now explain how to estimate the adversary's profit for $\mathcal{G}$. To this end, we provide a specific recipe for computing a clique clustering of $\mathcal{G}$. We do not claim that this particular clustering is actually optimal, but it is a lower bound on the optimum profit, and thus it is sufficient for our purpose.

For any node $v \in \mathcal{T}$ that is not a leaf, denote by $\mathcal{P}^{\mathrm{L}}(v)$ the longest path from v to a leaf of $\mathcal{T}$ that goes through the left child of v. If v is a non-leaf in the core tree, and thus has a right child, then $\mathcal{P}^{\mathrm{R}}(v)$ is the longest path from v to a leaf of $\mathcal{T}$ that goes through this right child. In both cases, ties are broken arbitrarily but consistently, for example in favor of the leftmost leaves. If v is in a tentacle (so it does not have the right child), then we let $\mathcal{P}^{\mathrm{R}}(v) = \{v\}$.

Let $\mathcal{P}^{\mathrm{L}}(v) = (v = v_1, v_2, \ldots, v_m)$, where v_m is a leaf of $\mathcal{T}$. Since v is not a leaf, the definition of $\mathcal{T}$ implies that $m \geq 2$. We now define the clique $C^{\mathrm{L}}(v)$ in $\mathcal{G}$ that corresponds to $\mathcal{P}^{\mathrm{L}}(v)$. Intuitively, for each v_i we add to $C^{\mathrm{L}}(v)$ one of the corresponding vertices, v_i^{L} or v_i^{R}, depending on whether v_{i+1} is the left or the right child of v_i. The following formal definition describes the construction of $C^{\mathrm{L}}(v)$ in a top-down fashion:

- $v_1^{\mathrm{L}} \in C^{\mathrm{L}}(v)$.
- Suppose that $1 \leq i \leq m-1$ and that $v_i^{\sigma} \in C^{\mathrm{L}}(v)$, for $\sigma \in \{\mathrm{L}, \mathrm{R}\}$. Then
  - if $i = m-1$, add v_m^{L} and v_m^{R} to $C^{\mathrm{L}}(v)$;
  - otherwise, if v_{i+2} is the left child of v_{i+1}, add v_{i+1}^{L} to $C^{\mathrm{L}}(v)$, and if v_{i+2} is the right child of v_{i+1}, add v_{i+1}^{R} to $C^{\mathrm{L}}(v)$.

We define $C^{\mathrm{R}}(v)$ analogously to $C^{\mathrm{L}}(v)$, but with two differences. First, we use $\mathcal{P}^{\mathrm{R}}(v)$ instead of $\mathcal{P}^{\mathrm{L}}(v)$ and second, if v is in a tentacle then we let $C^{\mathrm{R}}(v) = \{v^{\mathrm{R}}, v^{\mathrm{D}}\}$. In other words, the whiskers form 2-cliques.

Observe that except cliques $C^{\mathrm{R}}(v)$ corresponding to the whiskers (that is, when v is in a tentacle), all cliques $C^{\sigma}(v)$ have cardinality at least 3.

We now define a clique partitioning $\mathcal{C}^*$ of $\mathcal{G}$, as follows: First we include cliques $C^{\mathrm{L}}(r)$ and $C^{\mathrm{R}}(r)$ in $\mathcal{C}^*$. We then proceed recursively: choose any node v such that exactly one of $v^{\mathrm{L}}, v^{\mathrm{R}}$ is already covered by some clique of $\mathcal{C}^*$. If v^{L} is covered but v^{R} is not, then include $C^{\mathrm{R}}(v)$ in $\mathcal{C}^*$. Similarly, if v^{R} is covered but v^{L} is not, then include $C^{\mathrm{L}}(v)$ in $\mathcal{C}^*$.

Analysis. Denote by $\mathcal{T}_v$ the subtree of $\mathcal{T}$ rooted at v. By $\mathcal{G}_v$ we denote the subgraph of $\mathcal{G}$ induced by the vertices that correspond to the nodes in $\mathcal{T}_v$. Each clique in $\mathcal{C}^*$ that intersects $\mathcal{G}_v$ induces a clique in $\mathcal{G}_v$, and the partitioning $\mathcal{C}^*$ induces a partitioning $\mathcal{C}_v^*$ of $\mathcal{G}_v$ into cliques. We will use notation O_v for the profit of partitioning $\mathcal{C}_v^*$. Note that $\mathcal{C}_v^*$ can be obtained with the same top-down process as $\mathcal{C}^*$, but starting from v as the root instead of r (Fig. 6).

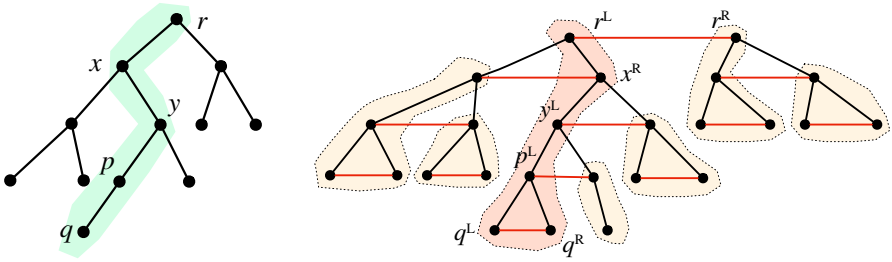

Fig. 6 On the left, an example of a path $\mathcal{P}^{\mathrm{L}}(r) = (r, x, y, p, q)$ in $\mathcal{T}$. In this example, $D = 3$. The corresponding clique $C^{\mathrm{L}}(r)$ is shown on the right (darker shape). The figure on the right also shows the adversary clique partitioning of $\mathcal{G}$. To avoid clutter, upward edges are not shown

We denote algorithm $\mathcal{S}$'s profit (the number of cross edges) within $\mathcal{G}_v$ by S_v . In particular, we have $\mathsf{profit}_{\mathcal{S}}(\mathcal{G}) = \mathsf{S}_r$ and $\mathsf{profit}_{\mathsf{OPT}}(\mathcal{G}) \geq \mathsf{O}_r$. Thus, to show (18), it is sufficient to prove that

$$\mathsf{O}_r \geq (6 - \epsilon_D) \cdot \mathsf{S}_r, \tag{19}$$

where $\epsilon_D \to 0$ when $D \to \infty$.

We will in fact prove an analogue of inequality (19) for all subtrees $\mathcal{T}_v$. To this end, we distinguish between two types of subtrees $\mathcal{T}_v$. If $\mathcal{T}_v$ ends at depth D of $\mathcal{T}$ or less (in other words, if $\mathcal{T}_v$ is inside the core of $\mathcal{T}$), we call $\mathcal{T}_v$ *shallow*. If $\mathcal{T}_v$ ends at depth $D + 1$ or more, we call it *deep*. So deep subtrees are those that contain some tentacles of $\mathcal{T}$.

Lemma 3.3 *If $\mathcal{T}_v$ is shallow, then*

$$O_v \geq 6 \cdot S_v.$$

Proof This can be shown by induction on the depth of $\mathcal{T}_v$. If this depth is 0, that is $\mathcal{T}_v = \{v\}$, then $\mathsf{O}_v = 1$ and $\mathsf{S}_v = 0$, so the ratio is actually infinite. To jump-start the induction we also need to analyze the case when the depth of $\mathcal{T}_v$ is 1. This means that $\mathcal{S}$ collected only edge $(v^{\mathrm{L}}, v^{\mathrm{R}})$ from $\mathcal{T}_v$. When this happened, the adversary generated vertices corresponding to the two children of v in $\mathcal{T}$ and its clustering consists of two triangles. So now $\mathsf{O}_v = 6$ and $\mathsf{S}_v = 1$, and the lemma holds.

Inductively, suppose that the depth of $\mathcal{T}_v$ is at least two, let y, z be the left and right children of v in $\mathcal{T}$, and assume that the lemma holds for $\mathcal{T}_y$ and $\mathcal{T}_z$. Naturally, we have $\mathsf{S}_v = \mathsf{S}_y + \mathsf{S}_z + 1$. Regarding the adversary profit, since the depth of $\mathcal{T}_v$ is at least two, cluster $C^{\mathrm{L}}(v)$ contains exactly one of $y^{\mathrm{L}}, y^{\mathrm{R}}$; say it contains y^{L}. Thus $C^{\mathrm{L}}(v)$ is obtained from $C^{\mathrm{L}}(y)$ by adding v^{L}. By the definition of clustering $\mathcal{C}^*$, the depth of $\mathcal{T}_y$ is at least 1, which means that adding v^{L} will add at least three new edges. By a similar argument, we will also add at least three edges from v^{R}. This implies that $\mathsf{O}_v \geq \mathsf{O}_y + \mathsf{O}_z + 6 \geq 6 \cdot \mathsf{S}_y + 6 \cdot \mathsf{S}_z + 6 \geq 6 \cdot \mathsf{S}_v$, completing the inductive step. □

From Lemma (3.3) we obtain that, in particular, if $\mathcal{T}$ itself is shallow then $\mathsf{O}_r \geq 6 \cdot \mathsf{S}_r$, which is even stronger than inequality (19) that we are in the process of justifying. Thus, for the rest of the proof, we can restrict our attention to skeleton trees $\mathcal{T}$ that are deep.

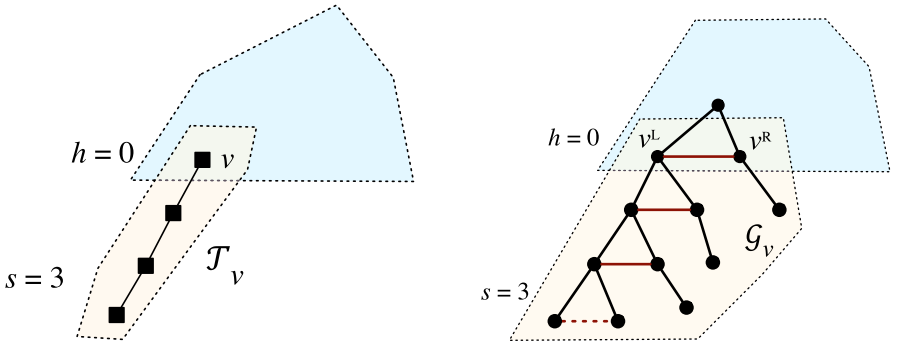

Fig. 7 Illustration of the proof of Lemma 3.4, the base case. Subtree $\mathcal{T}_v$ on the left, the corresponding subgraph $\mathcal{G}_v$ on the right

So next we consider deep subtrees of $\mathcal{T}$. The *core depth* of a deep subtree $\mathcal{T}_v$ is defined as the depth of the part of $\mathcal{T}_v$ within the core subtree of $\mathcal{T}$. (In other words, the core depth of $\mathcal{T}_v$ is equal to D minus the depth of v in $\mathcal{T}$.) If h and s are, respectively, the core depth of $\mathcal{T}_v$ and its maximum tentacle length, then $0 \le h \le D$ and $s \ge 1$. The sum $h + s$ is then simply the depth of $\mathcal{T}$.

Lemma 3.4 *Let $\mathcal{T}_v$ be a deep subtree of core depth $h \ge 0$ and maximum tentacle length $s \ge 1$, then*

$$\mathsf{O}_v + 2(h + s) \ge 6 \cdot \mathsf{S}_v.$$

Before proving the lemma, let us argue first that this lemma is sufficient to establish our lower bound. Indeed, since we are now considering the case when $\mathcal{T}$ is a deep subtree itself, the lemma implies that $\mathsf{O}_r + 2(D+s) \ge 6 \cdot \mathsf{S}_r$, where s is the maximum tentacle length of $\mathcal{T}$. But O_r is at least quadratic in $D + s$. So for large D the ratio $\mathsf{O}_r/\mathsf{S}_r$ approaches 6.

Proof To prove Lemma 3.4, we use induction on h, the core depth of $\mathcal{T}_v$. Consider first the base case, for $h = 0$ (when $\mathcal{T}_v$ is just a tentacle). In his clustering $\mathcal{C}_v^*$, the adversary has one clique of $s + 2$ vertices, namely all x^{L} vertices in the tentacle (there are $s + 1$ of these), plus one z^{R} vertex for the leaf z. He also has s whiskers, so his profit for $\mathcal{T}_v$ is $\binom{s+2}{2} + s = \frac{1}{2}(s^2 + 5s + 2)$. $\mathcal{S}$ collects only s edges, namely all cross edges in $\mathcal{T}_v$ except the last. (See Fig. 7.) Solving the quadratic inequality and using the integrality of s, we get $\mathsf{O}_v + 2s \ge 6s = 6 \cdot \mathsf{S}_v$. Note that this inequality is in fact tight for $s = 1$ and 2.

In the inductive step, consider a deep subtree $\mathcal{T}_v$. Let y and z be the left and right children of v. Without loss of generality, we can assume that $\mathcal{T}_y$ is a deep tree with core depth $h - 1$ and the same maximum tentacle length s as $\mathcal{T}_v$, while $\mathcal{T}_z$ is either shallow (that is, it has no tentacles), or it is a deep tree with maximum tentacle length at most s (Fig. 8).

By the inductive assumption, we have $\mathsf{O}_y + 2(h - 1 + s) \ge 6 \cdot \mathsf{S}_y$. Regarding z, if $\mathcal{T}_z$ is shallow then from Lemma 3.3 we get $\mathsf{O}_z \ge 6 \cdot \mathsf{S}_z$, and if $\mathcal{T}_z$ is deep (necessarily of core depth $h - 1$) then $\mathsf{O}_z + 2(h - 1 + s') \ge 6 \cdot \mathsf{S}_z$, where s' is $\mathcal{T}_z$'s maximum tentacle length, such that $1 \le s' \le s$.

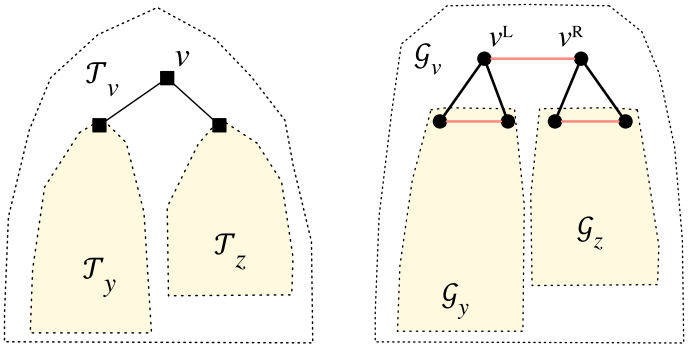

Fig. 8 Illustration of the proof of Lemma 3.4, the inductive step. Subtrees $\mathcal{T}_v, \mathcal{T}_y, \mathcal{T}_z$ on the left, the corresponding subgraphs on the right

Consider first the case when $\mathcal{T}_z$ is shallow. Note that

$$\begin{aligned} \mathsf{S}_v &= \mathsf{S}_y + \mathsf{S}_z + 1 \quad \text{and} \\ \mathsf{O}_v &\geq \mathsf{O}_y + \mathsf{O}_z + h + s + 4 \end{aligned}$$

The first equation is trivial, because the profit of $\mathcal{S}$ in $\mathcal{G}_v$ consists of all cross edges in $\mathcal{G}_y$ and $\mathcal{G}_z$, plus one more cross edge $(v^{\mathrm{L}}, v^{\mathrm{R}})$. The second inequality holds because the adversary clustering $\mathcal{C}_v^*$ is obtained by adding v^{L} to $\mathcal{G}_y$'s cluster with $(h-1)+s+2 = h+s+1$ vertices, and v^{R} can be added to $\mathcal{G}_z$'s cluster that with at least 3 vertices. We get

$$\begin{aligned} \mathsf{O}_v + 2(h+s) &\geq [\mathsf{O}_y + \mathsf{O}_z + h + s + 4] + 2(h+s) \\ &\geq [\mathsf{O}_y + 2(h-1+s)] + \mathsf{O}_z + 6 \\ &\geq 6 \cdot \mathsf{S}_y + 6 \cdot \mathsf{S}_z + 6 \\ &= 6 \cdot \mathsf{S}_v. \end{aligned}$$

The second case is when $\mathcal{T}_z$ is a deep tree (of the same core depth $h-1$ as $\mathcal{T}_y$) with maximum tentacle length s', where $1 \leq s' \leq s$. As before, we have $\mathsf{S}_v = \mathsf{S}_y + \mathsf{S}_z + 1$. The optimum profit satisfies (by a similar argument as before, applied to both $\mathcal{T}_y$ and $\mathcal{T}_z$)

$$\mathsf{O}_v \geq \mathsf{O}_y + \mathsf{O}_z + 2h + s + s' + 2.$$

We obtain (using $s \geq s'$)

$$\begin{aligned} \mathsf{O}_v + 2(h+s) &\geq [\mathsf{O}_y + \mathsf{O}_z + 2h + s + s' + 2] + 2(h+s) \\ &\geq [\mathsf{O}_y + 2(h-1+s)] + [\mathsf{O}_z + 2(h-1+s')] + 6 \\ &\geq 6 \cdot \mathsf{S}_y + 6 \cdot \mathsf{S}_z + 6 \\ &= 6 \cdot \mathsf{S}_v. \end{aligned}$$

This completes the proof of Lemma 3.4. □

The asymptotic ratio. Lemma 3.4 implies (as explained right after the lemma) that there is no deterministic algorithm for the clique clustering problem with strict competitive ratio smaller than 6. We still need to explain how to extend our proof so that it also applies to the asymptotic competitive ratio. This is quite simple: Choose some large constant K. The adversary will create K instances of the above game, playing each one independently. Our construction above uses the fact that at each step the algorithm is forced to collect one of the pending cross edges, otherwise its competitive ratio exceeds ratio R (where R is arbitrarily close to 6). Now, for K sufficiently large, the algorithm is forced to collect cross edges in all except for some finite number of copies of the game, where this number depends on the additive constant in the competitiveness bound.

Summarizing this section, we have just proved the following lower bound.

Theorem 4 *There is no online deterministic algorithm for* MaxCC *clustering with competitive ratio smaller than* 6.

Note: Our construction is very tight, in the following sense. Suppose that $\mathcal{S}$ maintains $\mathcal{T}$ as balanced as possible. Then the ratio is exactly 6 when the depth of $\mathcal{T}$ is 1 or 2. Furthermore, suppose that D is very large and the algorithm constructs $\mathcal{T}$ to have depth D or more, that is, it starts growing tentacles (but still maintaining $\mathcal{T}$ balanced.) Then the ratio is $6 - o(1)$ for tentacle lengths $s = 1$ and $s = 2$. The intuition is that when the adversary plays optimally, he will only allow the online algorithm to collect isolated edges (cliques of size 2). For this reason, we conjecture that 6 is the optimal competitive ratio.

4 Online MinCC Clustering

In this section, we study the clique clustering problem with a different measure of optimality that we call MinCC. For MinCC, we define the *cost* of a clustering $\mathcal{C}$ to be the total number of *non-cluster edges*. Specifically, if the cliques in $\mathcal{C}$ are $C_1, C_2, \ldots, C_k$ then the cost of $\mathcal{C}$ is $|E| - \sum_{i=1}^{k} \binom{|C_i|}{2}$. The objective is to construct a clustering that minimizes this cost.

4.1 A Lower Bound for Online MinCC Clustering

In this section, we present a lower bound for deterministic MinCC clustering.

Theorem 5 *(a) There is no online deterministic algorithm for* MinCC *clustering with competitive ratio $n - \omega(1)$, where n is the number of vertices.*

(b) There is no online deterministic algorithm for MinCC *clustering with strict competitive ratio smaller than $n - 2$.*

Proof (a) Consider an algorithm $\mathcal{S}$ with competitive ratio $R_n = n - \omega(1)$. Thus, according to the definition (2) of the competitive ratio, there is a constant β that

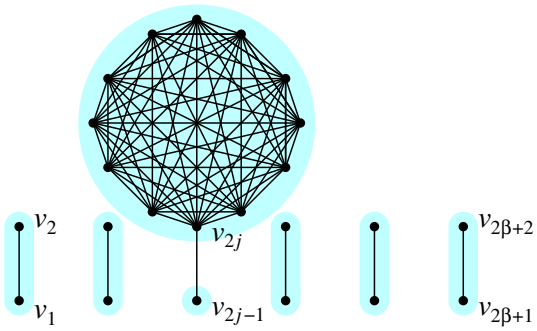

Fig. 9 Illustrating the lower bound proof of Theorem 5. The figure shows the optimal clustering for the graph

satisfies $\mathsf{cost}_{\mathcal{S}}(G) \leq R_n \cdot \mathsf{cost}_{\mathsf{OPT}}(G) + \beta$, where $n = |G|$. We can assume that β is a positive integer.

The adversary first produces a graph of $2\beta + 2$ vertices connected by $\beta + 1$ disjoint edges (v_{2i-1}, v_{2i}), for $i = 1, 2, \ldots, \beta + 1$. At this point, $\mathcal{S}$ must have added at least one pair $\{v_{2j-1}, v_{2j}\}$ to its clustering, because otherwise, since $\mathsf{cost}_{\mathsf{OPT}}(G) = 0$, inequality (2) would be violated. The adversary then chooses some large n and adds $n - 2\beta - 2$ new vertices $v_{2\beta+3}, \ldots, v_n$ that together with v_{2j} form a clique of size $n - 2\beta - 1$; see Fig. 9. All edges from v_{2j} to these new vertices are non-cluster edges for $\mathcal{S}$ and the optimum solution has only one non-cluster edge (v_{2j-1}, v_{2j}). Thus

$$\begin{aligned}\mathsf{cost}_{\mathcal{S}}(G) - \beta &\geq (n - 2\beta - 2) - \beta = n - 3\beta - 2\\ &= (n - 3\beta - 2) \cdot \mathsf{cost}_{\mathsf{OPT}}(G) > R_n \cdot \mathsf{cost}_{\mathsf{OPT}}(G),\end{aligned}$$

giving us a contradiction for sufficiently large n.

(b) The proof of this part is a straightforward modification of the proof for (a): the adversary starts by releasing just one edge (v_1, v_2), and the online algorithm is forced to cluster v_1 and v_2 together, because now $\beta = 0$. Then the adversary adds $n - 2$ vertices that together with v_2 form a clique. This clique will be its only cluster, so $\mathsf{cost}_{\mathsf{OPT}}(G) = 1$. For the online algorithm all edges between v_2 and the other vertices in this clique will be outside its clusters, so $\mathsf{cost}_{\mathcal{S}}(G) \geq n - 2$, proving part (b). □

4.2 The Greedy Algorithm for Online MinCC Clustering

We continue the study of online MinCC clustering, and we prove that Greedy, the greedy algorithm presented in Sect. 3.1, yields a competitive ratio matching the lower bound from the previous section.

Lemma 4.1 *Let (u, v) be a non-cluster edge of* Greedy. *Then OPT (the optimal clustering) has at least one non-cluster edge adjacent to u or v (which might also be (u, v) itself).*

Proof The key observation for this proof is that, for any triplet of vertices x, y, and z, if the graph contains the two edges (x, y) and (x, z) but y and z are not connected by an edge, then in any clustering at least one of the edges (x, y) or (x, z) is a non-cluster edge.

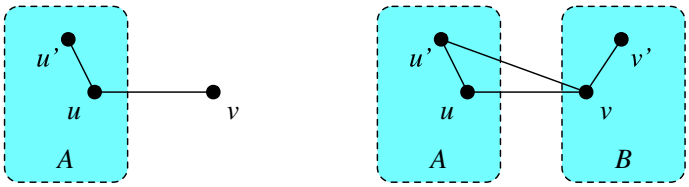

Fig. 10 Illustration of the proof of Lemma 4.1

Without loss of generality suppose vertex v arrives after vertex u. Let A be the cluster of GREEDY containing vertex u right after the step when vertex v arrives. We have that $v \notin A$ by the assumption of the lemma.

We have two possibilities. First, if A contains some vertex u' not connected to v, then the earlier key observation shows that one of the edges (u', u), (u, v) is a non-cluster edge for OPT (see Fig. 10 on the left).

Second, assume that v is connected to all vertices of A. GREEDY had an option of adding v to A but it didn't, so it placed v in some clique B (of size at least 2) that is not merge-able with A, that is, there are vertices $u' \in A$ and $v' \in B$ which are not connected by an edge (see Fig. 10 on the right). Now the earlier key observation shows that one of the edges (u', v), (v, v') is a non-cluster edge of OPT. This completes the proof of the lemma. □

Theorem 6 *The strict competitive ratio of* GREEDY *is* $n - 2$.

Proof To estimate the number of non-cluster edges of GREEDY, we use a charging scheme. Let (u, v) be a non-cluster edge of GREEDY. We charge it to non-cluster edges of OPT as follows.

> *Self charge*: If (u, v) is a non-cluster edge of OPT, we charge 1 to (u, v) itself.
> *Proximate charge*: If (u, v) is a cluster edge in OPT, we split the charge of 1 from (u, v) evenly among all non-cluster edges of OPT incident to u or v.

From Lemma 4.1, the charging scheme is well-defined, that is, all non-cluster edges of GREEDY have been charged fully to non-cluster edges of OPT. It remains to estimate the total charge that any non-cluster edge of OPT may have received. Since the strict competitive ratio is the ratio between the number of non-cluster edges of GREEDY and the number of non-cluster edges of OPT, the maximum charge to any non-cluster edge of OPT is an upper bound for the strict competitive ratio.

Consider a non-cluster edge (x, y) of OPT. Edge (x, y) can receive charges only from itself (self charge) and other edges incident to x or y (proximate charges). Let P be the set of vertices adjacent to both x and y, and let Q be the set of vertices that are adjacent to only one of them, but excluding x and y:

$$P = N(x) \cap N(y) \quad \text{and} \quad Q = N(x) \cup N(y) - P - \{x, y\}.$$

($N(z)$ denotes the neighborhood of a vertex z, the set of vertices adjacent to z.) We have $|P| + |Q| \leq n - 2$.

Edges connecting x or y to Q will be called Q-edges. Trivially, the total charge from Q-edges to (x, y) is at most $|Q|$.

Edges connecting x or y to P will be called P-edges. Consider some $z \in P$. Since x and y are in different clusters of OPT, at least one of P-edges (x, z) or (y, z) must also be a non-cluster edge for OPT. By symmetry, assume that (x, z) is a non-cluster edge for OPT. If (x, z) is a non-cluster edge of GREEDY then (x, z) will absorb its self charge. So (x, z) will not contribute to the charge of (x, y). If (y, z) is a non-cluster edge of GREEDY then either it will be self charged (if it's also a non-cluster edge of OPT) or its proximate charge will be split between at least two edges, namely (x, y) and (x, z). Thus the charge from (y, z) to (x, y) will be at most $\frac{1}{2}$. Therefore the total charge from P-edges to (x, y) is at most $\frac{1}{2}|P|$. We now have some cases.

Case 1: (x, y) is a cluster edge of GREEDY. Then (x, y) does not generate a self charge, so the total charge received by (x, y) is at most $\frac{1}{2}|P| + |Q| \leq |P| + |Q| \leq n - 2$.

Case 2: (x, y) is a non-cluster edge of GREEDY. Then (x, y) contributes a self charge to itself.

Case 2.1: $|P| \geq 2$. Then $\frac{1}{2}|P| \leq |P| - 1$, so the total charge received by (x, y) is at most $\frac{1}{2}|P| + |Q| + 1 \leq (|P| - 1) + |Q| = |P| + |Q| \leq n - 2$.

Case 2.2: At least one Q-edge is a cluster edge of GREEDY. Then the total proximate charge from Q-edges is at most $|Q| - 1$, so the total charge received by (x, y) is at most $\frac{1}{2}|P| + (|Q| - 1) + 1 \leq |P| + |Q| \leq n - 2$.

Case 2.3: $|P| \in \{0, 1\}$ and all Q-edges are non-cluster edges of GREEDY. We claim that this case cannot actually occur. Indeed, if $|P| = 0$ then GREEDY would cluster x and y together. Similarly, if $P = \{z\}$, then GREEDY would cluster x, y and z together. In both cases, we get a contradiction with the assumption of Case 2.

Summarizing, we have shown that each non-cluster edge of OPT receives a total charge of at most $n - 2$, and the theorem follows. □

The proof of Theorem 6 applies in fact to a more general class of algorithms, giving an upper bound of $n - 2$ on the strict competitive ratio of all "non-procrastinating" algorithms, which never leave merge-able clusters in their clusterings (that is clusters C, C' such that $C \cup C'$ forms a clique).

Acknowledgements Open access funding provided by Malmö University.

References

1. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Mach. Learn. **56**(1–3), 89–113 (2004)
2. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. J. Comput. Biol. **6**(3/4), 281–297 (1999)
3. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)

4. Charikar, M., Chekuri, C., Feder, T., Motwani, R.: Incremental clustering and dynamic information retrieval. SIAM J. Comput. **33**(6), 1417–1440 (2004)
5. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. In: Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on, pp. 524–533. IEEE (2003)
6. Chaudhuri, K., Godfrey, B., Rao, S., Talwar, K.: Paths, trees, and minimum latency tours. In: 44th Symposium on Foundations of Computer Science (FOCS 2003), 11–14 October 2003, Cambridge, MA, USA, Proceedings, pp .36–45 (2003)
7. Chrobak, M., Dürr, C., Nilsson, B.J.: Competitive strategies for online clique clustering. In: Proceedings of the 9th International Conference on Algorithms and Complexity (CIAC'15), pp. 101–113 (2015)
8. Chrobak, M., Hurand, M.: Better bounds for incremental medians. Theor. Comput. Sci. **412**(7), 594–601 (2011)
9. Chrobak, M., Kenyon, C., Noga, J., Young, N.E.: Incremental medians via online bidding. Algorithmica **50**(4), 455–478 (2008)
10. Chrobak, M., Kenyon-Mathieu, C.: SIGACT news online algorithms column 10: competitiveness via doubling. SIGACT News **37**(4), 115–126 (2006)
11. Demaine, E.D., Immorlica, N.: Correlation clustering with partial information. In: Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'03), pp. 1–13 (2003)
12. Dessmark, A., Jansson, J., Lingas, A., Lundell, E.-M., Persson, M.: On the approximability of maximum and minimum edge clique partition problems. In: Proceedings of the 12th Computing: The Australasian Theory Symposium (CATS'06), pp. 101–105 (2006)
13. Fabijan, A., Nilsson, B.J., Persson, M.: Competitive online clique clustering. In: Proceedings of the 8th International Conference on Algorithms and Complexity (CIAC'13), pp. 221–233 (2013)
14. Figueroa, A., Borneman, J., Jiang, T.: Clustering binary fingerprint vectors with missing values for DNA array data analysis. J. Comput. Biol. **11**(5), 887–901 (2004)
15. Figueroa, A., Goldstein, A., Jiang, T., Kurowski, M., Lingas, A., Persson, M.: Approximate clustering of incomplete fingerprints. J. Discrete Algorithms **6**(1), 103–108 (2008)
16. Lin, G., Nagarajan, C., Rajaraman, R., Williamson, D.P.: A general approach for incremental approximation and hierarchical clustering. SIAM J. Comput. **39**(8), 3633–3669 (2010)
17. Mathieu, C., Sankur, O., Schudy, W.: Online correlation clustering. In: 27th International Symposium on Theoretical Aspects of Computer Science (STACS'10), pp. 573–584 (2010)
18. Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. Discrete Appl. Math. **144**(1–2), 173–182 (2004)
19. Valinsky, L., Vedova, G.D., Jiang, T., Borneman, J.: Oligonucleotide fingerprinting of rRNA genes for analysis of fungal community composition. Appl. Environ. Microbiol. **68**, 5999–6004 (2002)
20. Valinsky, L., Vedova, G.D., Scupham, R.J., Alvey, S., Figueroa, A., Bei Yin, R., Hartin, J., Chrobak, M., Crowley, D.E., Jiang, T., Borneman, J.: Analysis of bacterial community composition by oligonucleotide fingerprinting of rRNA genes. Appl. Environ. Microbiol. **68**, 243–3250 (2002)