A PTAS for Minimum Clique Partition in Unit Disk Graphs

Imran A. Pirwani* Mohammad R. Salavatipour†

Abstract

We consider the problem of partitioning the set of vertices of a given unit disk graph (UDG) into a minimum number of cliques. The problem is NP-hard and various constant factor approximations are known, with the current best ratio of 3. Our main result is a polynomial time approximation scheme (PTAS) for this problem on UDG. In fact, we present a robust algorithm that given a graph G (not necessarily UDG) with edge-lengths, it either (i) computes a clique partition or (ii) gives a certificate that the graph is not a UDG; for the case (i) that it computes a clique partition, we show that it is guaranteed to be within $(1+\varepsilon)$ ratio of the optimum if the input is UDG; however if the input is not a UDG it either computes a clique partition as in case (i) with no guarantee on the quality of the clique partition or detects that it is not a UDG. Noting that recognition of UDG's is NP-hard even if we are given edge lengths, our PTAS is a robust algorithm. Our main technical contribution involves showing the property of *separability* of an optimal clique partition; that there exists an optimal clique partition where the convex hulls of the cliques are pairwise non-overlapping. Our algorithm can be transformed into an $O\left(\frac{\log^* n}{\varepsilon^{O(1)}}\right)$ time distributed polynomial-time approximation scheme (PTAS).

Finally, we consider a weighted version of the clique partition problem on vertex weighted UDGs; the weight of a clique is the weight of a heaviest vertex in it, and the weight of a clique partition is the sum of the weights of the cliques in it. This formulation generalizes the classical clique partition problem. We note some key distinctions between the weighted and the unweighted versions, where ideas developed for the unweighted case do not help. Yet we show that the problem admits a $(2+\varepsilon)$-approximation algorithm for the weighted version of the problem where the graph is expressed in standard form, for example, as an *adjacency matrix*. This improves on the best known algorithm which constructs an 8-approximation for the unweighted case for UDGs expressed in standard form.

1 Introduction

Consider a large wireless network deployed over a large geographic terrain. Nodes in this network are equipped with a small computation device, and a small band radio that can be used to communicate with "nearby" nodes, forming a network. Each device is also equipped with a limited battery supply which powers all its functions. These networks are becoming cheaper to deploy and easier to maintain while their utility is increasing simultaneously, due to advances in micro-processor, battery, and signal processing technologies. It is becoming feasible to deploy nodes in larger numbers due to the reduction in manufacturing cost and their robustness. Hence, such networks are becoming ubiquitous [28]. As a result of the increased demand and utility, numerous algorithmic challenges on such networks arise. These challenges include orderly communication with the goal of reducing cacophony, thrifty use of limited battery supply without compromising the primary task, and fast multi-hop routing, to name a few. While there are numerous kinds of wireless networks, *homogeneous* networks, where all nodes in the network have identical capabilities, are amongst the commonly deployed.

*Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: pirwani@cs.ualberta.ca. Supported by Alberta Ingenuity.

†Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: mreza@cs.ualberta.ca. Supported by NSERC and Alberta Ingenuity.

One of the standard network models for homogeneous networks is the *unit disk graph* (UDG). A graph $G = (V, E)$ is a UDG if there is a mapping $f : V \mapsto \mathbb{R}^2$ such that $\|f(u) - f(v)\|_2 \leq 1 \Leftrightarrow \{u, v\} \in E$; $f(u)$[1] models the position of the node u while the unit disk centered at $f(u)$ models the range of radio communication. Two nodes u and v are said to be able to directly communicate if they lie in the unit disks placed at each others' centers. There is a vast collection of literature on algorithmic problems studied on UDGs [22, 9, 11, 1, 18, 2, 12]. For a recent survey see [3].

Clustering of a set of points or nodes, is an important sub-routine in many algorithmic and practical applications and there exist various kinds of clusterings depending upon the application. Typically, the objective in clustering is to minimize the number of "groups" such that each "group" (cluster) satisfies a set of criteria. Mutual *proximity* of nodes in a cluster is one such criterion, while nodes in a cluster forming a *clique* in the underlying network is an extreme form of mutual proximity. In this paper, we study a classical optimization problem related to clustering, called the *minimum clique partition* problem on this important graph class.

Minimum clique partition on unit disk graphs (MCP): Given a *unit disk graph*, $G = (V, E)$, partition V into a smallest number of cliques.

While the problem is of independent theoretical interest, MCP has proved to be quite useful for other problems. For example, the authors of [25] shows how to use a small-sized clique partition of a UDG to construct a large collection of disjoint (almost) dominating sets. They [26] also show how to obtain a good *quality realization* of UDGs, and an important ingredient in their technique was to construct a small-sized clique partition of the graph. It is shown [21] how to use a small-sized clique partition to obtain *sparse spanners* with bounded dilation, which also permit *guaranteed geographic routing* on a related class of graphs. Recently [23] employed MCP to obtain an $O(\log^* n)$ time distributed algorithm which is an $O(\log n)$-approximation for the *facility location problem* on UDGs without geometry; they also give an $O(1)$ time distributed $O(1)$-approximation to the facility location problem on UDGs with geometry also using MCP.

Related Work. On general graphs, the clique-partition problem is equivalent to the minimum graph coloring on the complement graph which is not approximable within $n^{1-\varepsilon}$, for any $\varepsilon > 0$, unless P=NP [31]. Halldórsson gave an $O(n\frac{(\log\log n)^2}{(\log n)^3})$-approximation algorithm for the coloring problem [14]. A related problem called the *minimum clique cover* problem is shown to be equivalent to MCP under ratio-preserving reduction [30]. MCP has also been studied for cubic graphs and its subclasses. The problem is shown to be MaxSNP-hard for cubic graphs, whereas it is shown to be NP-complete for the case of planar cubic graphs [6]; they also give a 5/4-approximation algorithm for graphs with maximum degree at most 3.

MCP is NP-hard even on a special class of UDGs, called *unit coin graphs*, where the interiors of the associated disks are pairwise disjoint [7]. Good approximations, however, are possible on UDGs. A 6-approximation can be easily obtained using the fact that the maximum independent size in the neighborhood of any vertex has size at most 6. The best known approximation is due to [7] who give a 3-approximation via a partitioning the vertices into *co-comparability* graphs, and solving the problem exactly on them. They also give a 3/2-approximation algorithm for coin graphs. MCP has also been studied on input UDGs expressed in standard form, that is, without a realization. For UDGs without the use of geometry or even access to edge-lengths [26] give an 8-approximation algorithm.

Our Results and Techniques. In this paper we present a *robust* PTAS for MCP on a given UDG. For ease of exposition, first we prove this (in Section 2) for the case that the UDG is given with a *realization*, $f(.)$. The *holy-grail* is a PTAS for the case when the UDG is expressed in *standard form*, say, as an adjacency matrix. However, falling short of proving this, we show (in Section 3) how to get a

[1] $f(.)$ is called a *realization* of UDG, G. Note that G may not come with a realization.

PTAS for the case when the input UDG is expressed in standard form along with associated edge-lengths corresponding to some realization. The algorithm is robust in the sense that it either (i) computes a clique partition of the input graph or (ii) gives a certificate that the input graph is not a UDG. If the input is indeed a UDG then the algorithm returns a clique partition (case (i)) which is a $(1+\varepsilon)$-approximation (for a given $\epsilon > 0$). However, if the input is not a UDG, the algorithm either computes a clique partition but with no guarantee on the quality of the solution or returns that it is not a UDG. Therefore, this algorithm should also be seen as a *robust PTAS*. The generation of a polynomial-sized certificate which proves why the input graph is not a UDG should be seen in the context of the negative result of [2] which says that even if edge lengths are given, UDG recognition is NP-hard. We show (in Section 4) how this algorithm can be modified to run in $O(\frac{\log^* n}{\varepsilon^{O(1)}})$ rounds of distributed computation in the $\mathcal{LOCAL}$ model [24].

At the heart of our PTAS is a subroutine which computes an optimal MCP for graphs of bounded diameter. It can be easily shown that the number of cliques in an optimum solution for such graphs has constant size. However, the number of points in every clique can be very large. Therefore, it is non-trivial to enumerate all the possible solutions in polynomial time. To overcome this difficulty, we first show the existence of an optimum solution in which the convex-hulls of the cliques are non-overlapping. This implies that for every pair of cliques in such a partition, there is a straight line separating the two. Using this property, one can show that even though the number of points in every clique in an optimum solution might be large, there is a collection of constant number of disjoint polygons each of which contains one of the cliques and each has a constant number of corners. Therefore, for each clique, we have a bounding polygon of constant size. These polygons are precisely the convex regions obtained using the separating lines.

Finally, (in Section 5) we explore a weighted version of MCP where we are given a vertex weighted UDG. In this formulation, the weight of a clique is the weight of a heaviest vertex in it, and the weight of a clique partition is the sum of the weights of the cliques in it. We note some key distinctions between the weighted and the unweighted versions of the problem and show that the ideas that help in obtaining a PTAS do not help in the weighted case. Yet, surprisingly, we show that the problem admits a $(2+\varepsilon)$-approximation algorithm for the weighted case *using only adjacency*. This result should be contrasted with the unweighted case where it is not clear as to how to remove the dependence on the use of edge-lengths, which was crucially exploited in deriving a PTAS.

Throughout the paper, we use OPT to denote an optimum clique partition and opt to denote the size (or, in Section 5, weight) of an optimum clique partition. We also use n and m to denote the number of points (i.e. nodes of $G = (V, E)$) and the number of edges, respectively.

2 A PTAS for UDGs Given With a Geometric Realization

In this section we present a PTAS for a given UDG, G, expressed with a realization on the plane. The overall structure of the algorithm is as follows. Using a randomly shifted grid whose cell size is $k \times k$ (for a constant $k = k(\varepsilon)$) we first partition the plane. It can be shown that for large enough values of k, the probability that any fixed clique in an optimum clique partition is cut by this grid (and therefore belongs to two or more different cells) is small. Thus, if we compute an optimal clique partition in each cell and return the union of these cliques, then we get a solution whose size is at most $(1+\epsilon)$opt. See Algorithm 1 for details.

Theorem 1. *MinCP1 returns a clique partition of size at most $(1+\varepsilon)$opt w.h.p.*

We begin with a simple observation.

Observation 2. *The diameter of the convex hull of every clique is at most* 1.

Algorithm 1 MinCP1(G, ε)

1: Let $k = \lceil \frac{16}{\varepsilon} \rceil$. Place a grid whose squares have size $k \times k$, on the plane. Call it $\mathcal{G}_{0,0}$.
2: Pick $(a, b) \in [0, k) \times [0, k)$ uniformly at random.
3: Shift $\mathcal{G}_{0,0}$ by (a, b) to get $\mathcal{G}_{a,b}$ which is a grid shifted a units to the right and b units above. $\mathcal{G}_{a,b}$ induces a random partition of V into points in $k \times k$ regions.
4: **for all** $k \times k$ regions of $\mathcal{G}_{a,b}$ **do**
5: Obtain an optimal partition for point-set P in the $k \times k$ square.
6: Let $\mathcal{C}_{a,b} = C_1 \cup C_2 \cup \ldots \cup C_i$ be the union of clique partitions obtained for the points in each $k \times k$ square.
7: Repeat "Step 2"−"Step 6" $\lceil \log n \rceil$ times and return the smallest $\mathcal{C}_{a,b}$ obtained over the $\lceil \log n \rceil$ independent trials.

We will show in the next subsection how to perform "Step 5" of the algorithm MinCP1 efficiently. Assuming this, let us prove Theorem 1. For a random shift $\mathcal{G}_{a,b}$ and a clique C, we say that $\mathcal{G}_{a,b}$ "cuts" C if some line of $\mathcal{G}_{a,b}$ crosses an edge of C. Next, we bound the probability that C is cut by $\mathcal{G}_{a,b}$.

$$\mathbf{Pr}\left[\substack{C \text{ is cut}\\ \text{by } \mathcal{G}_{a,b}}\right] \leq \mathbf{Pr}\left[\substack{\text{a vertical line of } \mathcal{G}_{a,b}\\ \text{crosses an edge of } C}\right] + \mathbf{Pr}\left[\substack{\text{a horizontal line of } \mathcal{G}_{a,b}\\ \text{crosses an edge of } C}\right] \leq \frac{1}{k} + \frac{1}{k} = \frac{2}{k}$$

Next, we compute the expected number of cliques in an optimal partition that are "cut" by $\mathcal{G}_{a,b}$

$$\mathbf{E}\left[\substack{\text{number of cliques } O_i \in \text{OPT}\\ \text{that are cut by } \mathcal{G}_{a,b}}\right] = \sum_{O_i \in \text{OPT}} \mathbf{Pr}\left[\substack{O_i \text{ is cut}\\ \text{by } \mathcal{G}_{a,b}}\right] \leq \frac{2}{k} \cdot \text{opt}$$

So, by Markov's inequality, we have that, $\mathbf{Pr}\left[\substack{\text{more than } \frac{4}{k}\text{opt}\\ \text{cliques are cut by } \mathcal{G}_{a,b}}\right] \leq \frac{1}{2}$

Consider an optimal solution and for a $k \times k$ grid cell S, let $\mathcal{O}_s$ be the cliques in the optimum solution that intersect S. Note that each clique of $\mathcal{O}_s$ might intersect up to 4 grid cells. Since we compute an optimal solution for the grid cell S, the number of cliques we compute for S is no more than that of $\mathcal{O}_s$. However, for each of the cliques in $\mathcal{O}_s$ that are cut by the grid lines we may generate up to four cliques in the final solution (one in each of the at most 4 grid cells that intersects that clique). So for each of the at most $\frac{4}{k} \cdot$ opt cliques that are cut by $\mathcal{G}_{a,b}$ we have at most four cliques in our final solution. Thus, in one iteration of steps 2 to 6, with probability at least 1/2, our algorithm constructs a clique partition whose size is at most opt plus an excess of at most $4 \times \frac{4}{k} \cdot$ opt. Since we repeat this process for $\lceil \log n \rceil$ independent random trials, we get that with probability at least $1 - \frac{1}{n}$:

$$|\mathcal{C}| \leq \text{opt} + \frac{16}{k} \cdot \text{opt} = \left(1 + \frac{16}{k}\right) \cdot \text{opt} \leq \left(1 + \frac{16\varepsilon}{16}\right) \cdot \text{opt} = (1 + \varepsilon) \cdot \text{opt}.$$

Also, note that the random shifting strategy has a standard deterministic analogue [15, 16, 8].

Corollary 3. *There is a deterministic PTAS for Minimum Clique-Partition on UDGs.*

2.1 Optimal Clique Partition of a UDG in a $k \times k$ Square

In this section, we describe details of an algorithm which computes an optimal clique partition of a UDG whose vertices lie in a square of size $k \times k$. In fact, the algorithm works for a slightly more general setting in which we have a given upper bound of ℓ for the size of an optimum clique partition of the input UDG; the algorithm solves the problem optimally in time $n^{O(\ell^2)}$. For a $k \times k$ square, we can show that $\ell \in O(k^2)$, thus the running time will be polynomial in n for constant k. Before describing the algorithm, we reveal a key structure of an optimal solution in a region whose diameter is bounded above by a constant.

2.1.1 Structural Properties and the Separation Theorem

Our goal here is to show that for any UDG there is an optimum solution in which the convex hulls of the cliques in the clique partition are pairwise non-overlapping. We remark that this discovery may be of independent interest; we refer to this property as *separability* and it plays a crucial role in reducing the combinatorial complexity of clique partitions in a small region. It is worthwhile to note that unlike optimization problems such as *maximum (weighted) independent set* and *minimum dominating set*, both of which share the property that one can "guess" only a small-sized subset of points, one of which corresponds to an optimal solution, the combinatorial complexity of any single clique in an optimal solution can be quite high. Therefore, it is unclear as to how to "guess" even few cliques, each of which may have a large set of points. This phenomenon of separability of an optimal partition, coupled with the fact (which we show) that the size of an optimal partition in a small region is small, allows us to circumvent this difficulty.

Lemma 4. *Any set of points P in a $k \times k$ square has a clique partition of size $O(k^2)$.*

Proof. Place a grid whose cells have size $1/2 \times 1/2$. This grid induces a vertex partition where each block in the partition consists of the points that share a common grid cell (and therefore form a clique). $\square$

Lemma 5. *If edges $\{a, b\}$ and $\{x, y\}$ of a UDG cross in the plane, then of the four edges corresponding to the four sides of the quadrilateral defined by the four end-points, two edges exist in G that share an end-point.*

Proof. Suppose that p is the crossing point of ab and xy. Without loss of generality, assume that $|ap| \leq \frac{1}{2}$ and that $|xp| \leq \frac{1}{2}$ and also that $|xp| \leq |ap|$. Then by triangle inequality we have $|ax| \leq |ap| + |xp| \leq 1$ so $|ax|$ is an edge. Also, again by triangle inequality, $|xb| \leq |pb| + |xp| \leq |pb| + |ap| \leq 1$, so $|xb|$ is an edge too. $\square$

Call a pair of convex polygons A and B on the plane *overlapping* if $A \cap B$ has a non-zero area. The proof of the following technical lemma appears in Appendix A.

Lemma 6. *For any clique-partition $\mathcal{O} = \{O_1, O_2, \ldots, O_\ell\}$ there is another clique partition of size at most ℓ such that the convex hulls of the cliques are pairwise non-overlapping.*

An immediate consequence of the above lemma is the following separation theorem which permits enumeration of small-sized clique partitions, one of which is a partition of optimal size,

Theorem 7. [Separation Theorem] *For any clique partition in which the convex hulls of the cliques are pairwise disjoint and for each pair of cliques, C_i, C_j, there is a straight line l_{ij} that separates C_i, C_j such that all vertices of C_i are on one side of l_{ij}, and all the vertices of C_j are on the other side of l_{ij}. (see Figure 1).*

Because there are $O(k^2)$ cliques in an optimal clique partition of a $k \times k$ region, there exist $O(k^4)$ straight-line separators between all pairs of cliques. Two lines that *separate* the same pair of cliques are said to be in the same equivalence class. In this sense, the number of equivalence classes of separator lines that determine an optimum clique partition is $O(k^4)$. If we can find one separator line from each equivalence class in poly-time then an exhaustive search for all such structures can be done in time $n^{O(k^4)}$, which is polynomial in n (for a fixed k).

In order to reduce the search space for separator lines, one can find a characterization of the separator lines with some extra properties. Let C_i, C_j be a pair of cliques each having at least two points. Let L_{ij} be the (infinite) set of distinct separator lines. Since C_i and C_j are convex, there exists at least one line in L_{ij} that goes through two points of C_i (or C_j) (see Figure 1(b)). Therefore:

Lemma 8. *Given two cliques C_i and C_j in a clique partition (with pairwise non-overlapping parts) there is a separator line l_{ij} that goes through two vertices of one of them, say $u, v \in C_i$ such that all the vertices of C_j are on one side of this line and all the vertices of C_i are on the other side or on the line.*

2.1.2 Obtaining an Optimal Partition via the Separation Theorem

We show that for any UDG with a given bound ℓ on the size of the optimum clique partition, we can find an optimum clique partition in time $n^{O(\ell^2)}$. Since for a $k \times k$ square we have $\ell \in O(k^2)$, the result follows for the sub-problem in a $k \times k$ square. In the rest of this subsection we assume that ℓ is the given upper bound on the size of an optimum clique partition of the input UDG. Before we describe how to exploit the separation theorem to obtain a polynomial time exact algorithm, consider Figure 1.

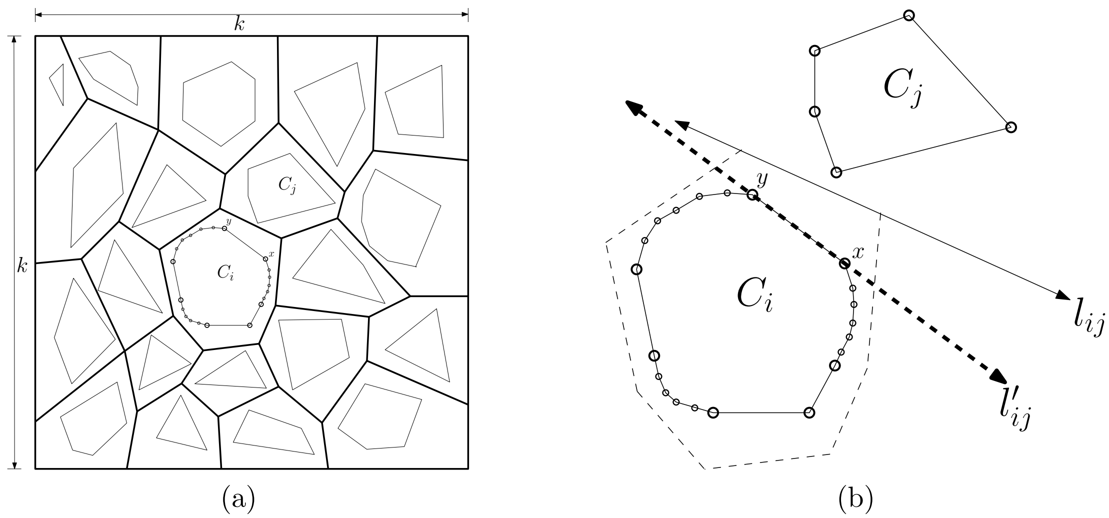


Figure 1: (a) An optimal clique partition of UDG points in a bounded region; each light convex shape corresponds to a clique in the clique partition. The heavy line-segments represent segments of the corresponding separators. (b) A close-up view of C_i and C_j. A separator line, l_{ij} is shown which separates C_i and C_j, corresponding to the segment in (a). Note that l'_{ij} is also a separator for C_i and C_j and l'_{ij} is passing through points x and y in C_i.

Figure 1(a) shows an optimal clique partition whose existence is a consequence of the separation theorem. Since there are $O(\ell)$ cliques in an optimal partition, there are $O(\ell^2)$ distinct pairs of cliques in the partition and we can assume that their convex hulls are pairwise non-overlapping. So, there are $O(\ell^2)$ distinct straight lines, each of which separate a pair of cliques in our (separable) optimal solution. For every clique C_i, the separator lines l_{ij} (for all values of j) define a convex region that contains clique C_i. So once we guess this set of $O(\ell^2)$ lines, these convex regions define the cliques. We will try all possible (non-equivalent) sets of $O(\ell^2)$ separator lines and check if each of the convex regions indeed defines a clique. We describe the details below.

For $\alpha = 1, 2, 3, \ldots, \ell$, the algorithm tries to build a clique partition $C_1, C_2, \ldots, C_\alpha$; one of the guessed value of α is the correct value of opt. Note that for a clique in the optimum clique partition, there are $\alpha - 1$ separator lines separating this clique from the others. These lines will involve at most $2\alpha - 2$ points of the clique. For each clique in the clique partition of size α, we guess (by enumerating all possibilities) whether it has at most $2\alpha - 2$ points (and so it is called small) or at least $2\alpha - 1$ points (and so it is called large). For every small clique, we also guess (i.e. enumerate) all the points that will belong to it. There are $O(n^{2\alpha})$ such guesses for each small clique and since there are at most α such cliques, thus there are a total of at most $n^{O(\alpha^2)}$ such guesses for small cliques. We now describe how to partition the remaining points into large cliques. For each large clique C_i we guess a representative point r_i; the idea is that none of the separator lines for C_i is going through r_i. We use r_i to determine which side of separator line is the side that contains clique C_i. There are at most $n^{O(\alpha)}$ guesses for the representatives. For every pair of large cliques C_i and C_j we also guess two points that are distinct from their representatives such that the separator line between C_i and C_j goes through them. We call this line l_{ij} and let $\mathcal{L}$ be the collection of all the guessed lines. There are a total of $O(\alpha^2)$ pairs of large cliques and for each a total of $O(n^2)$ guesses for the separator line l_{ij}. So, there are a total of at most $n^{O(\alpha^2)}$ guesses for $\mathcal{L}$ and r_i's (combined). For each such line l_{ij} and point r_i, the side of the line that contains r_i is considered positive for r_i. Now, consider the convex region defined by the positive sides of lines $l_{i1}, l_{i2}, \ldots$. All the points in this convex region will be added to the set C_i. Together with the small cliques computed earlier, we obtain sets $C_1, \ldots, C_\alpha$. At the end we check if each C_i forms a clique or not and if their union contains all the points. See Algorithm 2 for details.

Algorithm 2 CP1(P)

$\mathcal{O} \leftarrow P$
for all values of $\alpha \leq \ell$ as the guess value for optimum clique-partition **do**
  for all $1 \leq i \leq \alpha$ guess a number $1 \leq s_i \leq 2\alpha - 1$. If $s_i \leq 2\alpha - 2$ then guess a set of s_i points (distinct from others) and place them in C_i; C_i will be a small clique. Otherwise, guess a representative point r_i for C_i. **do**
    for all pairs i, j which have representatives r_i and r_j guess two points distinct from r_i, r_j and let l_{ij} be the line that goes through those two points **do**
      Let R_i be the convex region containing r_i defined by $l_{i1}, l_{i2}, \ldots$. Define C_i to be the set of points that belong to R_i.
      if each of $C_1, \ldots, C_\alpha$ forms a clique and $\alpha < |\mathcal{O}|$ **then**
        $\mathcal{O} \leftarrow \mathcal{C}$
return $\mathcal{O}$

Running Time: With n being the number of points in P, there are at most $n^{O(\alpha^2)}$ guesses for small cliques; for each such guess we guess the representatives and there are at most $n^{O(\alpha)}$ many guesses. Then we guess the collection $\mathcal{L}$, there are $n^{O(\alpha^2)}$ such guesses. So, overall, for any value of α, there are $n^{O(\alpha^2)}$ configurations that we consider and for each we check whether $C_1, \ldots, C_\alpha$ is really a clique partition. Thus, the total running time is in $n^{O(\ell^2)}$. The following is an immediate consequence,

Theorem 9. *Given a set P of n points representing the vertices of a UDG together with an upper bound ℓ for the size of an optimum clique partition, algorithm CP1(P) computes an optimal clique partition in time $n^{O(\ell^2)}$.*

Since for the case of a $k \times k$ grid we have $\ell \in O(k^2)$ the running time in this case will be $n^{O(k^4)}$.

3 A Robust PTAS for UDG Expressed with Edge-lengths

In this section, we weaken our assumption on having access to geometric embedding of the nodes; instead, we assume that all the edge-lengths are known exactly with respect to some feasible, yet unknown, realization of the UDG.

Theorem 10. *Given a graph G with associated (rational) edge-lengths and $\varepsilon > 0$, there is a polynomial time algorithm which either computes a clique partition of G or gives a certificate that G is not a UDG. If G is a UDG, the size of the clique partition computed is a $(1+\varepsilon)$-approximation of the optimum clique partition (but there is no guarantee on the size of the clique partition if the input graph is not UDG).*

Our algorithm will borrow heavily from ideas developed in earlier sections where we show how to compute a minimum clique partition in a neighborhood of bounded radius. The high level idea of the algorithm is as follows. Similar to the previous algorithm, we first decompose the graph into bounded diameter regions and show that if we can compute the optimum clique partition of each region then the union of these clique partitions is within $(1+\varepsilon)$ fraction of the optimum. For this purpose, in place of the grid shifting strategy, we will employ a ball growing technique that will give us bounded size regions. This is inspired by [22, 19], to obtain local PTAS for weighted independent set, and minimum dominating set for UDGs without the use of geometry. We then show that we can either compute a clique partition for each subgraph induced by a ball, or give a certificate that the subgraph is not UDG. If the subgraph is a UDG, then our clique partition is optimal but if it is not a UDG there is no guarantee (it may give a solution with no guarantee on the quality or detect that it is not a UDG).

Define the ball of radius r (in the number of hops) around v as $B_r(v) = \{u : d(u,v) \leq r\}$, where by $d(u,v)$ we mean the number of edges on a shortest path from u to v. So, $B_r(v)$ can be computed using a breadth-first search (BFS) tree rooted at v. We describe our decomposition algorithm which partitions the graph into bounded diameter subgraphs below (see Algorithm 3). We will describe a procedure, called OPT-CP which, given a graph induced by the vertices of $B_r(v)$ and a parameter $\ell = \text{poly}(r)$, runs in time $|B_r(v)|^{O(\ell^2)} \leq n^{O(\ell^2)}$ and either produces a certificate that $B_r(v)$ is not a UDG or computes a clique partition of $B_r(v)$; this clique partition is optimum if $B_r(v)$ is a UDG. We only call this procedure for "small" values of r.

Algorithm 3 MinCP2(G, ε)

1: $\mathcal{C} \leftarrow \emptyset$; $\beta \leftarrow \lceil c_0 \frac{1}{\varepsilon} \log \frac{1}{\varepsilon} \rceil$; $\ell \leftarrow c_1 \beta^2$.
  {In the above c_0 is the constant in Lemma 14, and c_1 is the constant in inequality (1).}
2: **while** $V \neq \emptyset$ **do**
3:   Pick an arbitrary vertex $v \in V$
4:   $r \leftarrow 0$
    {Here we use $C_r(v)$ to denote a clique partition of $B_r(v)$ computed by calling OPT-CP}
5:   **while** $|C_{r+2}(v)| > (1+\varepsilon) \cdot |C_r(v)|$ **do**
6:     $r \leftarrow r+1$
7:     **if** $(r > \beta)$ or (OPT-CP$(B_r(v))$ returns "not a UDG") **then**
8:       **return** "G is not a UDG" and produce $B_r(v)$ as the certificate
9:   $\mathcal{C} \leftarrow \mathcal{C} \cup C_{r+2}(v)$
10:   $V \leftarrow V \setminus B_{r+2}(v)$
11: **return** $\mathcal{C}$ as our clique partition

It is clear that if the algorithm returns $\mathcal{C}$ on "Step 11", it is a clique partition of V. Let us assume that each ball $B_r(v)$ we consider induces a UDG and that the procedure OPT-CP returns an optimal clique partition $C_r(v)$ for ball $B_r(v)$. We first show that in this case $|\mathcal{C}| \leq (1+\varepsilon)\text{opt}$. Then we will show that for any iteration of the outer "while–loop", "Step 5" of MinCP2 is executed in time polynomial in n, by relying on edge-lengths instead of coordinates in the Euclidean plane.

For an iteration i of the outer "while–loop", let v_i be the vertex chosen in "Step 3" and let r_i^* be the value of r for which the "while-loop" on "Step 5" terminates, that is, $|C_{r_i^*+2}(v_i)| \leq (1+\varepsilon) \cdot |C_{r_i^*}(v_i)|$. Let k be the maximum number of iterations of the outer "while-loop".

Lemma 11. *For every $i \neq j$, every pair of vertices $v \in B_{r_i^*}(v_i)$ and $u \in B_{r_j^*}(v_j)$ are non-adjacent.*

Proof. Without loss of generality, assume that $i < j$. Therefore, every vertex in $B_{r_j^*}(v_j)$ is at a level larger than $r_i^* + 2$ of the BFS tree rooted at v_i (otherwise it would have been part of the ball $B_{r_i^*+2}(v_i)$ and was removed from V). Note that in a BFS tree rooted at v_i, there cannot be an edge between a level

r and r' with $r' \geq r+2$. Thus there cannot be an edge between a node in $v \in B_{r_i^*}(v_i)$ (which has level at most r_i^*) and a node $u \in B_{r_j^*}(v_j)$ (which would been at a level at least r_i^*+3 in the BFS tree rooted at v_i). ☐

Lemma 12. *opt* $\geq \sum_{i=1}^{k} |C_{r_i^*}(v_i)|$

Proof. Note that $B_{r_i^*}(v_i)$ is obtained by constructing a BFS tree rooted at vertex v_i up to some depth r_i^*. Using the previous lemma there is no edge between any two nodes $v \in B_{r_i^*}(v_i)$ and $u \in B_{r_j^*}(v_j)$. So, no single clique in an optimum solution can contain vertices from distinct $B_{r_i^*}(v_i)$ and $B_{r_j^*}(v_j)$. Consider the subset of cliques in an optimal clique partition of G that intersect $B_{r_i^*}(v_i)$ and call this subset OPT_i. The argument above shows that OPT_i is disjoint from OPT_j. Also, each OPT_i contains all the vertices in $B_{r_i^*}(v_i)$. Since $C_{r_i^*}(v_i)$ is an optimal clique partition for $B_{r_i^*}(v_i)$, $|\mathrm{OPT}_i| \geq |C_{r_i^*}(v_i)|$. The lemma immediately follows by observing that OPT_i and OPT_j are disjoint. ☐

Next, we relate the cost of the clique partition that we obtain with an optimal clique partition and show that the clique partition obtained by the algorithm MinCP2(G, ε) is within a factor $(1+\varepsilon)$ of an optimal clique partition of G.

Lemma 13. *If* $|C_{r_i^*+2}(v_i)| \leq (1+\varepsilon)\cdot |C_{r_i^*}(v_i)|$, *then* $\left|\bigcup_{i=1}^{k} C_{r_i^*+2}(v_i)\right| \leq (1+\varepsilon)\cdot$ *opt*

Proof.

$$\left|\bigcup_{i=1}^{k} C_{r_i^*+2}(v_i)\right| = \sum_{i=1}^{k} \left|C_{r_i^*+2}(v_i)\right| \leq (1+\varepsilon)\cdot \sum_{i=1}^{k} \left|C_{r_i^*}(v_i)\right| \leq (1+\varepsilon)\cdot \mathrm{opt},$$

where the last inequality uses the previous lemma. ☐

Finally, we show that the inner "while-loop" terminates in $\tilde{O}(\frac{1}{\varepsilon})$, so $r_i^* \in \tilde{O}(\frac{1}{\varepsilon})$. Obviously, the "while-loop" on "Step 5" terminates eventually, so r_i^* exists. By definition of r_i^*, for all smaller values of $r < r_i^*$: $|C_r(v_i)| > (1+\varepsilon)\cdot |C_{r-2}(v_i)|$. Since diameter of $B_r(v_i)$ is $O(r)$, if $B_r(v)$ is a UDG, there is a realization of it in which all the points fit into a $r \times r$ grid. Thus, $|C_r(v_i)| \in O(r^2)$. So for some $\alpha \in O(1)$:

$$\alpha\cdot r^2 > |C_r(v_i)| > (1+\varepsilon)\cdot |C_{r-2}(v_i)| > \ldots > (1+\varepsilon)^{\frac{r}{2}}\cdot |C_0(v_i)| = O(\left(\sqrt{1+\varepsilon}\right)^r),$$

when r is even (for odd values of r we obtain $|C_r(v_i)| > (1+\varepsilon)^{\frac{r-1}{2}}\cdot |C_1(v_i)| \geq O(\left(\sqrt{1+\varepsilon}\right)^{r-1})$. Therefore we have:

Lemma 14. *There is a constant $c_0 > 0$ such that for each i: $r_i^* \leq c_0/\varepsilon\cdot \log 1/\varepsilon$.*

In the next subsection, we show that the algorithm OPT-CP, given $B_r(v)$ and an upper bound ℓ on $|C_r(v)|$, either computes a clique partition or declares that the graph is not UDG; the size of the partition is optimal if $B_r(v)$ is a UDG. The algorithm runs in time $n^{O(\ell^2)}$. By the above arguments, if $B_r(v)$ is a UDG then, there is a constant $c_1 > 0$ such that:

$$|C_r(v)| = O(r_i^{*2}) \leq c_1 \cdot \frac{c_0^2}{\varepsilon^2} \log^2 \frac{1}{\varepsilon}. \tag{1}$$

Thus we can pass $\ell = \lceil c_1 \frac{c_0^2}{\varepsilon^2} \log^2 \frac{1}{\varepsilon} \rceil$ as a parameter to any invocation of OPT-CP as an upper bound, where c_1 is the constant in $O(r_i^{*2})$. So, the running time of the algorithm is at most $n^{\tilde{O}(1/\varepsilon^4)}$.

3.1 An Optimal Clique Partition for $B_r(v)$

Here we present the algorithm OPT-CP that given $B_r(v)$ (henceforth referred to as G') and an upper bound ℓ on the size of an optimal solution for G', either computes a clique partition of it or detects that it is not a UDG; if G' is a UDG then the partition is optimal. The algorithm runs in time $n^{O(\ell^2)}$. Since, by Lemma 14, ℓ is a constant in each call to this algorithm, the running time of OPT-CP is polynomial in n. Our algorithm is based on the separation theorem proved earlier and is similar to the algorithm CP1 presented in the previous section. Even though we do not have a realization of the nodes on the plane, assuming that G' is a UDG, we know a realization exists. We use node/point to refer to a vertex of G' and/or its corresponding point on the plane for some realization of G'. We will use the following technical lemma whose proof appears in Appendix A.

Lemma 15. *Suppose we have four mutually adjacent nodes p, a, b, r and their pairwise distances with respect to some realization on the Euclidean plane. Then there is a poly-time procedure that can decide if p and r are on the same side of the line that goes through a and b or are on different sides.*

Let us assume that G' is a UDG and has an optimum clique partition of size $\alpha \leq \ell$. As in Section 2, these cliques fall in two categories: small, and large. Here, we focus only on finding the large cliques since it is easy to guess all the small ones exactly as discussed in Section 2. Suppose for each pair $C_i, C_j \in$ OPT of large cliques, we guess their respective representatives, r_i and r_j. Further, suppose that we also guess a separating line l_{ij} correctly which goes through points u_{ij} and v_{ij}. For a given point p that is adjacent to r_i or r_j we wish to efficiently test if p is on the the same side of line l_{ij} as r_i (the *positive* side), or on r_j's side (the *negative* side), using only edge-lengths. Without loss of generality, assume that both u_{ij} and v_{ij} belong to clique C_i. For every node p different from the representatives:

- If p is adjacent to r_i (and also to u_{ij} and v_{ij}) and not to r_j then it is on the positive side of l_{ij} for C_i. If p is adjacent to r_j and not to r_i (or not to u_{ij} or v_{ij}) then it is on the positive side of l_{ij} for C_j (and so on the negative side for C_i).
- Suppose p is adjacent to all of r_i, u_{ij}, v_{ij}, r_j. Observe that we also have the edges $r_i u_{ij}$ and $r_i v_{ij}$. Given the edge-lengths of all the six edges among the four vertices r_i, u_{ij}, v_{ij}, p using Lemma 15 we can decide if in a realization of these four points, the line going through u_{ij}, v_{ij} separates the two points p and r_i or not. If p and r_i are on the same side, we say p is on the positive side of l_{ij} for C_i. Else, it is on the positive side of l_{ij} for C_j.

For each C_i and all the lines l_{ij}, consider the set of nodes that are on the positive side of all these lines with respect to C_i; we place these nodes in C_i. After obtaining the large and the small cliques, we obtain sets $C_1, \ldots, C_\alpha$. At the end we check if each C_i forms a clique and if their union covers all the points. As before, the number of guesses for representatives is $n^{O(\alpha)}$ and the number of guesses for the separator lines is $n^{O(\alpha^2)}$. So there are a total of $n^{O(\alpha^2)}$ configurations that we consider.

It is easy to see that if G' is a UDG then one of the set of guesses is a correct one, allowing us to obtain an optimum clique partition. If G' is not a UDG, we may still find a clique partition of G'. However, if we fail to obtain a clique partition in our search then it is a certificate that G' is not a UDG.

4 $O(\log^* n)$-round Distributed PTAS for UDGs with Edge-Lengths

In this section, we give details of a distributed PTAS for MCP which runs in $O(\frac{\log^* n}{\varepsilon^{O(1)}})$ rounds of distributed computation under the $\mathcal{LOCAL}$ model of computation [24]. The model of computation that we employ assumes a *synchronous* system where *communication* between neighboring nodes takes place in synchronous rounds using messages of unbounded size [24]. So, in a single round of communication, any node acquires the subgraph (information pertaining to the set of nodes, edges, the states of local variables, etc.) within its immediate neighborhood. So, after k rounds of communication, any node acquires complete knowledge about its k-neighborhood.

Observe that in Algorithm MinCP2, the radius r of any ball $B_v(r)$ is bounded above by $\tilde{O}(1/\varepsilon)$, while the center, v, is an arbitrary vertex. Since the radius of any ball is "small", the maximum number of rounds of distributed computation that the sequential algorithm needs before terminating the "while-loop" is also "small". Therefore, for any pair of balls $B_u(r_i)$ and $B_v(r_j)$, such that $d(u,v) \in \omega(1/\varepsilon)$, one should be able to run part of the sequential algorithm in parallel, as they surely are independent of each other. We borrow some ideas from [19] and find regions that are far apart such that we can run the sequential algorithm in those regions in parallel. See Algorithm 4 for details.

Algorithm 4 Distr-MCP-UDG(G, ε)

```
β ← ⌈c_0 (1/ε) log (1/ε)⌉; ℓ ← c_1 β^2; all vertices are unmarked.
{c_0 is the constant in Lemma 14 and c_1 is the constant inequality (1).}
Construct a maximal subset, V_c ⊂ V, such that for any pair u, v ∈ V_c, d(u,v) > β. Construct a graph
  G_c = (V_c, E_c), where E_c = {{u,v} : u, v ∈ V_c, d_G(u,v) ≤ 4β}. We call V_c, the set of leaders.
Proper color G_c using Δ(G_c) + 1 colors, where Δ(G_c) is the maximum degree of G_c.
Every v ∈ V \ V_c, "assigns" itself to a nearest leader u ∈ V_c, with ties broken arbitrarily, and colors
  itself the same color as the leader.
for i = 1 to Δ(G_c) + 1 do
  For each leader j with color i let G_i^j be the subgraph induced by the vertices assigned to leader j.
  for all G_i^j in parallel do
    Consider a fixed ordering on the unmarked vertices of G_i^j;
    Run the sequential ball growing algorithm on the next (in this ordering) unmarked vertex v ∈ G_i^j,
      we compute B_r(v); Note that B_r(v) might contain vertices of different colors (from outside G_i^j).
    Compute (using the sequential algorithm) the optimal clique-partition of B_r(v) and "mark" all
      those vertices
```

It should be pointed out that adapting the algorithm of [19] for *maximum independent set* and *minimum dominating set* to our setting is not trivial. The reason is that MCP is a *partition* of the entire vertex set and partitioning just a subset well enough will not do. Specifically [19] chooses a subset of vertices upon which their ball-growing algorithm is run; it suffices for their purposes to dispense with the remaining subset of vertices that were not picked by their ball-growing algorithm. If we had followed a similar scheme then we would surely get a good clique partition on a subset of vertices; however, it is unclear as to how to obtain a good partition of the remaining subset in terms of the optimal size for the original problem instance over the entire vertex set. As a means to circumvent this issue, we first construct a "crude" partition of the vertex set, instead of just a subset of vertices as done in [19].

4.1 Analysis

We now show that the algorithm constructs a $(1+\varepsilon)$-approximation to MCP on UDGs given only rational edge-lengths in $O(\log^* n)$ rounds of distributed computation under the $\mathcal{LOCAL}$ model; we first show correctness of the algorithm, followed by bounding the number of communication rounds.

Correctness: We prove that our algorithm is correct by showing that any execution of Distr-MCP-UDG can be turned into a sequential execution of MinCP2. As stated, every vertex has the same color as its leader; let a leader vertex be its own leader. We first show that the distance of every vertex to its leader is small.

Lemma 16. *For any vertex $v \notin V_c$, there is a vertex $u \in V_c$ such that $d_G(u, v) \leq \beta$.*

Proof. Suppose not. So there is a v whose distance to every $u \in V_c$ is more than β. But then $V'_c = V_c \cup \{v\}$ has the property that for all $x, y \in V'_c$, $d_G(x, y) > \beta$, contradicting the maximality of V_c. □

Next we show that for any pair of vertices u, v of the same color but with different leaders, the minimum distance between them is large enough so that a pair of balls of radius at most β over them will be disjoint, where β is defined in MinCP2 and Distr-MCP-UDG.

Lemma 17. *Consider two leaders x, y of the same color, say i, and any two vertices $u \in G_i^x$ and $v \in G_i^y$ (note that we might have $u = x$ or $v = y$). Then for all values of r considered in the ball growing algorithm, $B_r(u)$ and $B_r(v)$ are disjoint.*

Proof. Since x and y have the same color $d(x, y) > 4\beta$. By Lemma 16, any vertex in either of G_i^x or G_i^y is at a distance of at most β from the respective leader; so, $d_G(u, v) > 2\beta$. The lemma follows easily by noting the fact that $r \leq \beta$ in the ball growing algorithm. □

We are now ready to prove the correctness of Distr-MCP-UDG by showing an equivalence between any execution of it to some execution of MinCP2.

Lemma 18. *Any execution of Distr-MCP-UDG from "Step 5" to "Step 10" can be converted to a valid execution of MinCP2.*

Proof. Consider an arbitrary execution of Distr-MCP-UDG. Suppose that $V_1, V_2, V_3, \ldots$ is a sequence of disjoint sets of the vertices of V such that we run the ball growing algorithm in parallel (during Distr-MCP-UDG) on vertices of V_1 (and thus we compute an optimal clique partition on each vertex of V_1 in parallel) then we do this for vertices in V_2, and so on. Note that the vertices in V_i all have the same color and each has a different leader. Consider an arbitrary ordering π_i of the vertices in each V_i and suppose that we run MinCP2 algorithm on vertices of V_1 based on ordering π_1, then on vertices of V_2 based on ordering π_2, and so on. Since the vertices in each V_i have distinct leaders, by Lemma 17, the balls grown around them are disjoint. It should be easy to see that the balls grown by algorithm MinCP2 is exactly the same as the ones computed by Distr-MCP-UDG. □

The following result follows immediately as a corollary to Lemma 18.

Corollary 19. *Given an $\varepsilon > 0$, Distr-MCP-UDG constructs a clique partition of the input graph G with associated edge-lengths, or produces a certificate that G is not a UDG. If G is a UDG then the size of the partition is within $(1 + \varepsilon)$ of the optimum clique partition.*

Running Time: We now show that the algorithm runs in $O(\frac{\log^* n}{\varepsilon^{O(1)}})$ distributed rounds under the $\mathcal{LOCAL}$ model of computation.

Lemma 20. *"Step 2" requires $O(\beta \cdot \log^* n)$ rounds of communication.*

Proof. Observe that the result of "Step 2" is identical to constructing a *maximal independent set* (MIS) in G^β. Note that G^β is also a UDG where the new unit is β. As a result, G^β is a subclass of *growth-bounded graphs* [17] where all the distances are scaled by β; computation of MIS on G^β takes $O(\beta \cdot \log^* n)$ rounds [29] while the construction of G^β takes β rounds. Hence, the number of rounds needed by "Step 2" can be bounded by $O(\beta \cdot \log^* n)$. □

It is easy to see that constructing G_c requires at most 4β communication rounds. Next, we show that the maximum degree of G_c, $\Delta(G_c)$ is bounded by a constant.

Lemma 21. $\Delta(G_c) \in O(1)$

Proof. Let v be a vertex of G_c having maximum degree. Note that all its neighboring vertices in G_c lie in a disk of radius at most 4β. Also note that due to "Step 2" the minimum distance between any pair of vertices in G_c is more than β. As a result, any disk of diameter β contains at most 1 vertex of G_c. Using standard packing arguments of the underlying space, a crude upper bound on the number of vertices of G_c in a disk of radius at most 4β is 256 vertices; this also upper bounds the degree of v. $\square$

Next, we bound the number of rounds needed for "Step 3"

Lemma 22. *"Step 3" requires $O(\beta \cdot \log^* n)$ rounds of communication.*

Proof. For graphs whose maximum degree is Δ, a $\Delta + 1$ proper coloring requires $O(\Delta + \log^* n)$ rounds [20, 4]. Since $\Delta(G_c) \in O(1)$ (Lemma 21), and the fact that distances in G_c are scaled by a factor of 4β as compared to the distances in G, a $\Delta(G_c) + 1$ proper coloring of G_c can be obtained in $O(\beta \cdot \log^* n)$ rounds. $\square$

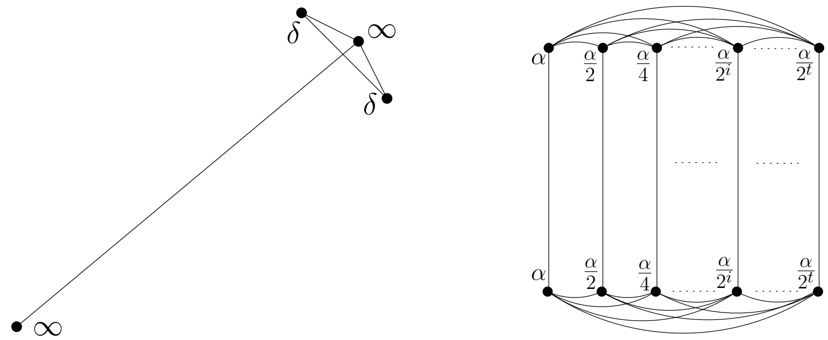
"Step 4" requires at most β rounds of communication; according to Lemma 16, for every $v \notin V_c$, there is some $u \in V_c$ that is at a distance at most β from it. The identity and color of such a vertex can be obtained in β rounds. We can now bound the number of rounds that Distr-MCP-UDG requires. First, note that for any iteration, i, of "Step 7", only knowledge of a subgraph up to radius β is required, and any node can obtain knowledge of the subgraph up to radius β from it in β rounds of communication. So, for any vertex in G_i^j obtains knowledge about the "marked/unmarked" status of all the vertices in G_i^j in β rounds of communication. Since the diameter of each G_i^j is at most 2β, the number of balls to grow in "Step 9." is at most $O(\beta^2)$. Therefore:

Theorem 23. *Distr-MCP-UDG requires $O(\beta \cdot \log^* n)$ rounds of communication under the $\mathcal{LOCAL}$ model of computation.*

5 $(2+\varepsilon)$-Approximation for Weighted Clique Partition using only Adjacency

In this section we consider a generalization of the minimum clique partition on UDGs, which we call *minimum weighted clique partition*. Given a node-weighted graph $G(V, E)$ with vertex weight $\mathbf{wt}(v)$, the weight of a clique C is defined as the weight of the heaviest vertex in it. For a clique partition $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$, the weight of $\mathcal{C}$ is defined as sum of the weights of the cliques in $\mathcal{C}$, i.e. $\mathbf{wt}(\mathcal{C}) = \mathbf{wt}\left(\bigcup_{i=1}^t C_i\right) = \sum_{i=1}^t \mathbf{wt}(C_i)$. The problem is, given G in standard form, say, as an adjacency matrix, construct a clique partition $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ while minimizing $\mathbf{wt}(\mathcal{C})$. We refer to this as the *minimum weighted clique partition* (MWCP) problem on UDGs. The weighted version of the problem as it is defined above has also been studied in different contexts. See [10, 5, 13] for study of weighted clique-partition on interval graphs and circular arc graphs.

Observe that MWCP distinguishes itself from MCP in two important ways: (i) The *separability* property which was crucially used earlier to devise a PTAS does not hold in the weighted case, and (ii) the number of cliques in an optimal solution for a UDG in a region of bounded radius is not bounded by the size of the region anymore, i.e. it is easy to construct examples of weighted UDGs in a bounded region where an optimal weighted clique partition contains an unbounded (in terms of region size) number of cliques. In addition, examples where two cliques in an optimal solution are not separable, that is, their convex hulls overlap, is easy to construct. (See the examples given in Figure 2.) To the best of our knowledge, MWCP has not been investigated before on UDGs. We, however, note that a simple modification to the algorithm by [26] also yields a factor-8 approximation to the weighted case, a generalization which they do not consider.



(a) (b)

Figure 2: (a) A vertex weighted UDG. The vertex weights are shown near the vertices. The optimal clique partition has weight $\infty + \delta$ whereas the lightest clique partition where the cliques are separable has weight $\infty + 2\cdot\delta$. (b) A UDG for which an optimal clique partition contains t cliques. The weight is less than $2\cdot\alpha$.

Here, we give an algorithm which runs in time $O(n^{\text{poly}(1/\varepsilon)})$ for a given $\varepsilon > 0$ and computes a $(2+\varepsilon)$-approximation to MWCP for UDGs expressed in standard form, for example, as an adjacency matrix. In fact, the algorithm is robust in that it either produces a clique partition or produces a polynomial-sized certificate proving that the input is not a UDG. For the case when the input is a UDG, the algorithm returns a clique partition and it is guaranteed to be a $(2+\varepsilon)$-approximation; but if the input is not UDG there is no guarantee on the quality of the clique partition (if it computes one).

Theorem 24. *Given a graph G expressed in standard form, and $\varepsilon > 0$, there is a polynomial time algorithm which either computes a clique partition of G or gives a certificate that G is not a UDG. If G is a UDG, the weight of the clique partition computed is a $(2+\varepsilon)$-approximation of the minimum weighted clique partition (but there is no guarantee on the weight of the clique partition if the input graph is not UDG).*

Our algorithm will borrow heavily from ideas developed in earlier sections and in [26]. The high level idea of the algorithm is as follows. Similar to the algorithm in Section 3, we first decompose the graph into bounded diameter regions and show that if we can compute a $(2+\varepsilon)$-approximate clique partition of each region then the union of these clique partitions is within $(2+\varepsilon)$ fraction of opt. We will employ a similar ball growing technique (as in Section 3) that will give us bounded size regions. This is also inspired by the strategy that is used in the work of Nieberg et al. [22], to obtain *local* PTAS for weighted independent set for UDGs using only adjacency. We then show that we can either compute a clique partition or give a certificate that the subgraph is not a UDG. If the subgraph is a UDG, then our clique partition is within a factor $(2+\varepsilon)$ of the optimal. For the case of bounded size region, although the optimum solution may have a large number of cliques, we can show that there is a clique partition with small number of cliques whose cost is within $(1+\varepsilon)$-factor of the optimum solution. First we describe the main algorithm. Then in Subsection 5.1 we show that for each subgraph $B_r(v)$ (of bounded diameter) there is a near optimal clique partition with $\tilde{O}(r^2)$ cliques. Then in Subsection 5.2 we show how to find such a near optimal clique partition.

Let us denote the weight of the optimum clique partition of G by opt. As before, let $B_r(v) = \{u : d(u,v) \leq r\}$, called the ball of (unweighted) distance r around v, be the set of vertices that are at most r hops from v in G. Our decomposition algorithm described below (see Algorithm 5) is similar to Algorithm 3 and partitions the graph into bounded diameter subgraphs below. The procedure CP, given a graph induced by the vertices of $B_r(v)$ and a parameter $\ell = \text{poly}(r)$, runs in time $n^{O(\ell^2)}$) and either gives a certificate that $B_r(v)$ is not a UDG or computes a clique partition of $B_r(v)$; this clique partition is within a factor $(2+\varepsilon)$ of the optimum if $B_r(v)$ is a UDG. We only call this procedure for constant values of r. In the following, let $0 < \gamma \leq \frac{\sqrt{9+4\varepsilon}-3}{2}$ be a rational number. See Algorithm 5 for details.

Algorithm 5 MinCP(G, γ)

1: $\mathcal{C} \leftarrow \emptyset$; $\beta \leftarrow \lceil c_0 \frac{1}{\gamma} \log \frac{1}{\gamma} \rceil$; $\ell \leftarrow c_1 \beta^2$.
   {In the above c_0 is the constant implicit in Lemma 28, and c_1 is the constant in inequality (2).}
2: **while** $V \neq \emptyset$ **do**
3:     $v \leftarrow \arg\max_u \{\mathbf{wt}\,(u)\}$
4:     $r \leftarrow 0$
       {Here we use $C_r(v)$ to denote a factor-$(2+\gamma)$ clique partition of $B_r(v)$ computed by calling CP}
5:     **while** $\mathbf{wt}\,(C_{r+2}(v)) > (1+\gamma) \cdot \mathbf{wt}\,(C_r(v))$ **do**
6:         $r \leftarrow r+1$
7:         **if** $(r > \beta)$ or (CP$(B_r(v), \ell)$ returns "not a UDG") **then**
8:             **return** "G is not a UDG" and produce $B_r(v)$ as the certificate
9:     $\mathcal{C} \leftarrow \mathcal{C} \cup C_{r+2}(v)$
10:     $V \leftarrow V \setminus B_{r+2}(v)$
11: **return** $\mathcal{C}$ as our clique partition

It is clear that if the algorithm returns $\mathcal{C}$ on "Step 11", it is a clique partition of V. Let us assume that each ball $B_r(v)$ we consider induces a UDG and that the procedure CP returns a $(2+\gamma)$-approximation, $C_r(v)$, for ball $B_r(v)$. We first show that in this case $\mathbf{wt}\,(\mathcal{C}) \leq (2+\varepsilon)$opt. Then we will show that for any iteration of the outer loop, "Step 5" of MinCP is executed in time polynomial in n while relying only on adjacency.

For an iteration i of the outer for loop, let v_i be the vertex chosen in "Step 3." and let r_i^* be the value of r for which the "while-loop" on "Step 5" terminates, that is, $\mathbf{wt}\left(C_{r_i^*+2}(v_i)\right) \leq (1+\gamma) \cdot \mathbf{wt}\left(C_{r_i^*}(v_i)\right)$. Let k is the maximum number of iterations of the outer "while-loop". The proof of the following Lemma is identical to the proof of Lemma 11.

Lemma 25. *Every two vertices $v \in B_{r_i^*}(v_i)$ and $u \in B_{r_j^*}(v_j)$ are non-adjacent.*

Lemma 26. $(2+\gamma) \cdot$ *opt* $\geq$ ***wt*** $\left(\bigcup_{i=1}^{k} C_{r_i^*}(v_i)\right)$

Proof. Note that $B_{r_i^*}(v_i)$ is obtained by constructing a BFS tree rooted at vertex v_i up to some depth r_i^*. Since the algorithm removes a super-set, $B_{r_i^*+2}(v_i)$, which has two more levels of the BFS tree, using the previous lemma there is no edge between any two nodes $v \in B_{r_i^*}(v_i)$ and $u \in B_{r_j^*}(v_j)$ for any pair $i \neq j$. So, no single clique in an optimum solution can contain vertices from distinct $B_{r_i^*}(v_i)$ and $B_{r_j^*}(v_j)$. Consider the subset of cliques in an optimal clique partition of G that intersect $B_{r_i^*}(v_i)$ and call this subset OPT_i. The argument above shows that OPT_i is disjoint from OPT_j. Also, each OPT_i contains all the vertices in $B_{r_i^*}(v_i)$. Since $C_{r_i^*}(v_i)$ is a factor-$(2+\gamma)$ approximation for $B_{r_i^*}(v_i)$, $(2+\gamma) \cdot \mathbf{wt}\,(\mathrm{OPT}_i) \geq \mathbf{wt}\left(C_{r_i^*}(v_i)\right)$. The lemma immediately follows by observing that OPT_i and OPT_j are disjoint. ☐

As a corollary, we can relate the cost of the clique partition that we obtain to an optimal clique partition:

Corollary 27. *If* ***wt*** $\left(C_{r_i^*+2}(v_i)\right) \leq (1+\gamma) \cdot$ ***wt*** $\left(C_{r_i^*}(v_i)\right)$, *then* ***wt*** $\left(\bigcup_{i=1}^{k} C_{r_i^*+2}(v_i)\right) \leq (2+\gamma)(1+\gamma) \cdot$ *opt.*

Proof.

$$\mathbf{wt}\left(\bigcup_{i=1}^{k} C_{r_i^*+2}(v_i)\right) = \sum_{i=1}^{k}\mathbf{wt}\left(C_{r_i^*+2}(v_i)\right) \le (1+\gamma)\cdot\sum_{i=1}^{k}\mathbf{wt}\left(C_{r_i^*}(v_i)\right) \le (2+\gamma)(1+\gamma)\cdot \mathrm{opt},$$

where the last inequality uses the previous lemma. ◻

Next, we show that the inner "while-loop" terminates in $\tilde{O}(\frac{1}{\gamma})$, that is each r_i^* is bounded by $\tilde{O}(\frac{1}{\gamma})$. This is similar to the proof of Lemma 14. Since the while loop terminates, r_i^* exists and by definition of r_i^*, it must be the case that for all smaller values of $r < r_i^*$, $\mathbf{wt}\left(C_r(v_i)\right) > (1+\gamma)\cdot\mathbf{wt}\left(C_{r-2}(v_i)\right)$. Because the diameter of $B_r(v_i)$ is $O(r)$, if $B_r(v)$ is a UDG, there is a realization of it in which all the points fit into a $r \times r$ grid. Also, since v_i is a heaviest vertex in the (residual) graph, there is a clique partition whose weight is at most $\alpha\cdot\mathbf{wt}\left(v_i\right)\cdot r^2$. Therefore, $\mathbf{wt}\left(C_r(v_i)\right) < \alpha\cdot\mathbf{wt}\left(v_i\right)\cdot r^2$, for some constant α. So:

$$\alpha\cdot\mathbf{wt}\left(v_i\right)\cdot r^2 > \mathbf{wt}\left(C_r(v_i)\right) > (1+\gamma)\cdot\mathbf{wt}\left(C_{r-2}(v_i)\right) > \ldots > (1+\gamma)^{\frac{r}{2}}\cdot\mathbf{wt}\left(C_0(v_i)\right) = \mathbf{wt}\left(v_i\right)\cdot\left(\sqrt{1+\gamma}\right)^r,$$

which implies $\alpha\cdot r^2 > \left(\sqrt{1+\gamma}\right)^r$, for the case that r is even. If r is odd we obtain $\alpha\cdot r^2 > \left(\sqrt{1+\gamma}\right)^{r-1}$. Thus, the following lemma easily follows:

Lemma 28. *There is a constant $c_0 > 0$ such that for each i: $r_i^* \le c_0/\gamma\cdot\log 1/\gamma$.*

In Subsection 5.2, we show the algorithm CP that given $B_r(v)$ and an upper bound ℓ on $|C_r(v)|$, either computes a clique partition (which is within a factor $2+\gamma$ of opt if $B_r(v)$ is a UDG) or detects that the graph is not UDG; the algorithm runs in time $n^{O(\ell^2)}$. By the above arguments, if $B_r(v)$ is a unit disk graph then there is a constant $c_1 > 0$ such that:

$$|C_r(v)| = O(r_i^{*2}) \le c_1\cdot\frac{c_0}{\gamma^2}\log^2\frac{1}{\gamma} \tag{2}$$

Thus we can pass $\ell = \lceil c_1\frac{1}{\gamma^2}\log^2\frac{1}{\gamma}\rceil$ as a parameter to any invocation of CP as an upper bound, where c_1 is the constant in $O(\frac{1}{\gamma^2}\log^2\frac{1}{\gamma})$. So, the running time of the algorithm will be at most $n^{\tilde{O}(1/\gamma^4)}$.

5.1 Existence of a Small Clique Partition of $B_r(v)$ having Near-optimal Weight

Before describing algorithm CP, we prove that even though (unlike the case of unweighted clique partition) an optimal weighted clique partition in a small region may contain an a large number of cliques, there exists a clique partition whose weight is within a factor $(1+\frac{\gamma}{2})$ of the minimum weight and it contains few cliques (where by "few" we mean ℓ as in Algorithm 5). The existence of such a small partition allows us to enumerate all possible number of cliques in the same manner we did in algorithm of subsection 3.1, permitting a $(2+\gamma)$-approximation for the problem instance in a ball of small radius. In the following, let $r \in \tilde{O}(\frac{1}{\gamma})$ and we focus on the subproblem that lies in some $B_r(v)$. Recall that any ball of radius r can be partitioned into $O(r^2)$ cliques (Lemma 4). We begin by proving a simple lemma which states that for any clique partition $\mathcal{C}$, if the set of vertices can be be covered by another clique partition $\mathcal{C}'$ containing x cliques then the sum of the weights of the x cliques in $\mathcal{C}'$ is not significantly more than the weight of the heaviest clique in $\mathcal{C}$.

Lemma 29. *For any collection of disjoint cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ having weights such that $\boldsymbol{wt}(C_1) \ge \boldsymbol{wt}(C_2) \ge \ldots \ge \boldsymbol{wt}(C_t)$ suppose the vertices of $\mathcal{C}$ can be partitioned into x cliques $\mathcal{C}' = \{C_1', C_2', \ldots, C_x'\}$. Then $\boldsymbol{wt}(\mathcal{C}') = \boldsymbol{wt}(\bigcup_{l=1}^{x} C_l') = \sum_{l=1}^{x}\boldsymbol{wt}(C_l') \le x\cdot\boldsymbol{wt}(C_1)$*

Proof. Without loss of generality, let $\mathbf{wt}\left(C_1'\right) \ge \mathbf{wt}\left(C_2'\right) \ge \ldots \ge \mathbf{wt}\left(C_x'\right)$. Since $\mathcal{C}'$ partitions vertices in $\mathcal{C}$, $\mathbf{wt}\left(C_1'\right) = \mathbf{wt}\left(C_1\right)$. Since $|\mathcal{C}'| = x$, $\mathbf{wt}\left(\mathcal{C}'\right) = \sum_{l=1}^{x}\mathbf{wt}\left(C_l'\right) \le x\cdot\mathbf{wt}\left(C_1'\right) = x\cdot\mathbf{wt}\left(C_1\right)$. ◻

Next, we show that any optimal clique partition of a ball of radius r has the property that for any clique, the sum of the weights of the lighter cliques is not significantly more than its weight.

Lemma 30. *Let $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ be an optimal clique partition and let $\boldsymbol{wt}(C_1) \geq \boldsymbol{wt}(C_2) \geq \ldots \geq \boldsymbol{wt}(C_t)$. Suppose there is another clique partition $\mathcal{C}' = \{C'_1, \ldots, C'_x\}$ of the vertices of $\mathcal{C}$. Then, for every $1 \leq i < t$: $(x-1)\cdot \boldsymbol{wt}(C_i) \geq \sum_{l=i+1}^{t} \boldsymbol{wt}(C_l)$.*

Proof. By way of contradiction, suppose there exists an index $1 \leq j < t$ such that $(x-1)\cdot \mathbf{wt}\left(C_j\right) < \sum_{l=j+1}^{t} \mathbf{wt}\left(C_l\right)$. Because $\bigcup_{l=1}^{t} C_l$ can be covered by $\mathcal{C}'$, so can $\bigcup_{l=j}^{t} C_l$. Let the $2 \leq x' \leq x$ be the smallest index such that $\mathcal{C}'_j = \{C'_1, C'_2, \ldots, C'_{x'}\}$ covers $\bigcup_{l=j}^{t} C_l$. On the other hand, $(x'-1)\cdot \mathbf{wt}\left(C_j\right) \leq (x-1)\cdot \mathbf{wt}\left(C_j\right) < \sum_{l=j+1}^{t} \mathbf{wt}\left(C_l\right)$, which implies

$$\text{opt} = \sum_{l=1}^{t} \mathbf{wt}\left(C_l\right) > \sum_{l=1}^{j} \mathbf{wt}\left(C_l\right) + (x'-1)\cdot \mathbf{wt}\left(C_j\right) = \sum_{l=1}^{j-1} \mathbf{wt}\left(C_l\right) + x' \mathbf{wt}\left(C_j\right). \tag{3}$$

By Lemma 29, $\mathbf{wt}\left(\mathcal{C}'_j\right) \leq x'\cdot \mathbf{wt}\left(C_j\right)$. This, combined with inequality (3) implies $\text{opt} > \sum_{l=1}^{j-1} \mathbf{wt}\left(C_l\right) + \sum_{l=1}^{x'} \mathbf{wt}\left(C'_l\right)$. Therefore the cliques in $\mathcal{C}'' = \{C_1, C_2, \ldots, C_{j-1}, C'_1, C'_2, \ldots, C'_{x'}\}$ cover all the nodes of cliques in $\mathcal{C}$ and has cost smaller than opt. If a vertex belongs to two or more cliques in $\mathcal{C}''$ we remove it from all but one of them to obtain a clique partition with cost no more than cost of $\mathcal{C}''$ which is smaller than opt. This contradiction completes the proof. $\square$

We now are ready to prove the main result of this section which states that for any optimal weighted clique partition of a ball of radius r, there exists another clique partition whose weight is arbitrarily close to the weight of the optimal partition, but has $O(r^2)$ cliques in it. Since the radius of the ball within which the subproblem lies is small, $r \in \tilde{O}(\frac{1}{\gamma})$, this means that if we were to enumerate all the clique partitions of the subproblem up to $O(r^2)$, we will see one whose weight is arbitrarily close to the weight of an optimal clique. Choosing a lightest one from amongst all such cliques guarantees that we will choose a one whose weight is arbitrarily close to the optimal weight.

Lemma 31. *Let $\gamma > 0$ and $r \in \tilde{O}(1/\gamma)$ be two constants. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ be an optimal weighted clique partition of $B_r(v)$ and let $\mathcal{C}' = \{C'_1, \ldots, C'_x\}$ be another clique partition of vertices of $\mathcal{C}$ with $x \in O(r^2)$. Let $\boldsymbol{wt}(C_1) \geq \boldsymbol{wt}(C_2) \geq \ldots \geq \boldsymbol{wt}(C_t)$. Then, there is a partition of vertices of $\mathcal{C}$ into at most $j + x$ cliques for some constant $j = j(\gamma)$, with cost at most $(1 + \frac{\gamma}{2})$opt.*

Proof. Without loss of generality, we assume that both x and t are at least two (as if $B_r(v)$ is a clique we are done). Consider an arbitrary value of $j \leq t$. Since $\bigcup_{l=1}^{t} C_l$ can be covered by x cliques in $\mathcal{C}'$, there is an index x' $(2 \leq x' \leq x)$ such that $\bigcup_{l=j}^{t} C_l$ can be covered by $\mathcal{C}'_j = \{C'_1, C'_2, \ldots, C'_{x'}\}$. By applying Lemma 30 repeatedly:

$$\text{opt} \geq \sum_{l=1}^{j} \mathbf{wt}\left(C_l\right) \geq \frac{1}{x'-1}\left(\sum_{l=2}^{j} \mathbf{wt}\left(C_l\right)\right) + \sum_{l=2}^{j} \mathbf{wt}\left(C_l\right) \geq \ldots \geq \left(\frac{x'}{x'-1}\right)^{j-1} \cdot \mathbf{wt}\left(C_j\right)$$

$$\Rightarrow \text{opt}\frac{(x'-1)^{j-1}}{x'^{j-2}} \geq x'\cdot \mathbf{wt}\left(C_j\right) \tag{4}$$

Using inequality (4):

$$\text{opt} + \frac{\text{opt}\cdot (x'-1)^{j-1}}{x'^{j-2}} \geq \sum_{l=1}^{j-1} \mathbf{wt}\left(C_l\right) + x'\cdot \mathbf{wt}\left(C_j\right) \geq \sum_{l=1}^{j-1} \mathbf{wt}\left(C_l\right) + \sum_{l=1}^{x'} \mathbf{wt}\left(C'_l\right), \tag{5}$$

where the second inequality follows by applying Lemma 29. Let $\mathcal{C}'' = \{C_1, \ldots, C_{j-1}, C'_1, \ldots, C'_{x'}\}$. Thus, the cliques in $\mathcal{C}''$ cover all the vertices of $\mathcal{C}$ and has total cost at most $\left(1 + \frac{(x'-1)^{j-1}}{x'^{j-2}}\right)$ opt by inequality (5). If a vertex belongs to two or more cliques in $\mathcal{C}''$ we remove it from all but one of them arbitrarily to obtain a clique partition of size $j - 1 + x'$ and whose total cost is upper bounded by $\left(1 + \frac{(x'-1)^{j-1}}{x'^{j-2}}\right)$ opt. Note that, $\frac{(x'-1)^{j-1}}{x'^{j-2}} = (x'-1)\left(\frac{x'-1}{x'}\right)^{j-2}$ and $0 < \frac{x'-1}{x'} < 1$ (because $x' \geq 2$). Since $r \in \tilde{O}(\frac{1}{\gamma})$ and $x' \leq x \in O(r^2)$, for an appropriate choice of $j = j(\gamma)$, $(x'-1)\left(\frac{x'-1}{x'}\right)^{j-2} < \gamma/2$. Thus we obtain a clique partition with $j + x - 1$ cliques and cost at most $(1 + \gamma/2)\cdot$ opt. This proves the lemma. $\square$

5.2 $(2 + \gamma)$-Approximation for MWCP in $B_r(v)$

In this subsection we show how to compute a $(2+\gamma)$-approximate MWCP of the graph $B_r(v)$ for any given γ. For an edge ordering $L = (e_1, e_2, \ldots, e_m)$ of a graph G with m edges, let $G_L[i]$ denote the edge induced subgraph with edge-set $\{e_i, e_{i+1}, \ldots, e_m\}$. For each e_i, let $N_L[i]$ denote the common neighborhood of the end-points of e_i in $G_L[i]$. An edge ordering $L = (e_1, e_2, \ldots, e_m)$ is a CNEEO if for every e_i in L, $N_L(i)$ induces a co-bipartite graph in G. It is known [27] that every UDG graph admits a co-bipartite edge elimination ordering (CNEEO). In the following, let G_v denote $B_r(v)$. We state a key lemma of [26].

Lemma 32. *[26] Let C be a clique in G_v, and let L be a CNEEO of G_v. Then, there is an i, $1 \leq i \leq m$, such that $N_L[i]$ contains C.*

Assume that G_v can be partitioned into $\alpha \leq \ell = \tilde{O}(1/\gamma^2)$ cliques, $\mathcal{O} = \{O_1, O_2, \ldots, O_\alpha\}$, such that $\mathbf{wt}(\mathcal{O}) \leq (1 + \frac{\gamma}{2})\cdot \mathbf{wt}(\mathrm{OPT}_v)$, where OPT_v is an optimal weighted clique partition of G_v. Note that by Lemma 31 this is true for subgraph $B_r(v)$. Suppose that we are given the upper bound ℓ; we will try all possible values of α. Without loss of generality, let $\mathbf{wt}(O_1) \geq \mathbf{wt}(O_2) \geq \ldots \geq \mathbf{wt}(O_\alpha)$. Observe that, without loss of generality, we can assume O_i is a maximal clique in $\bigcup_{j=i}^{\alpha} O_j$. The implication of the above lemma is that even though we do not know O_1, hence we do not know $\mathcal{O}$, we do know that for every CNEEO L of G_v, there is an e_i such that $N_L[i]$ can be partitioned into at most two cliques that fully cover O_1. Since O_1 is a heaviest clique, the two cliques that cover the subgraph $N_L[i]$ pay a cost of at most $2\cdot \mathbf{wt}(O_1)$. This suggests an algorithm that guesses an edge sequence $(f_1, f_2, \ldots, f_\alpha)$ of G_v. Then, the algorithm computes L, a CNEEO of G_v. The algorithm's first guess is "good" if f_1 is an edge in O_1 that occurs first in L. Suppose that this is the case and suppose that f_1 has rank i in L. Then, O_1 is contained in $N_L[i]$, and we cover $N_L[i]$ with at most two cliques. Call these C'_1 and C''_1 and $\mathbf{wt}(C'_1) + \mathbf{wt}(C''_1) \leq 2\cdot \mathbf{wt}(O_1)$. So, when we remove $N_L[i]$ from G_v, we get a UDG which can be partitioned into at most $\alpha - 1$ cliques, namely, $\mathcal{O}' = \{O_2, \ldots, O_\alpha\}$. We then again construct a CNEEO, L', of $G'_v = G_v \setminus N_L[i]$. Just like before, our guess f_2 is "good" if f_2 is an edge in O_2 and occurs first in L'. Let i' be the rank of f_2 in L', we see that $N_{L'}[i']$ fully contains O_2, and we again cover it with at most 2 cliques. Next, delete $N_{L'}[i']$ from G'_v to get a graph which can be partitioned into $\alpha - 2$ cliques, and so on. See Algorithm 6 for details.

Note that while Lemma 32 allows us to cover any clique with at most 2 cliques, it does not help us in finding the clique exactly. In the algorithm, note that if at any point, the algorithm is unable to construct a CNEEO, we can declare that the graph G_v is not a UDG. Also, if for all invocations of the algorithm by an external algorithm that guesses the value of opt we are unable to find a clique partition, then again we can declare that G_v is not a UDG.

6 Concluding Remarks

Recall that the weakest assumption that we needed to obtain a PTAS for unweighted clique partition problem was that all the edge lengths are given. This information was crucially used in obtaining a *robust*

Algorithm 6 CP(G_v, ℓ)

$\mathcal{C} \leftarrow V$; min $\leftarrow \mathbf{wt}(\mathcal{C})$;
for all $\alpha \leq \ell$ **do**
  for all α-edge sequence $(f_1, f_2, \ldots, f_\alpha)$ of G_v **do**
    $G_0 \leftarrow G_v$
    for $j = 1$ to α **do**
      Compute a CNEEO L of G_{j-1}
      $i \leftarrow$ rank of f_j in L
      Partition $N_L[i]$ into two cliques C'_j and C''_j
      $G_j \leftarrow G_{j-1} \setminus N_L[i]$
    if $G_\alpha = \emptyset$ and $\mathbf{wt}\left(\bigcup_{j=1}^{\alpha}\{C'_j, C''_j\}\right) <$ min **then**
      $\mathcal{C} \leftarrow \bigcup_{j=1}^{\alpha}\{C'_j, C''_j\}$; min $\leftarrow \mathbf{wt}(\mathcal{C})$;
return $\mathcal{C}$

PTAS. In the case of weighted clique partition, we gave a $(2+\varepsilon)$-approximation algorithm without the use of edge-lengths (using only the adjacency information). It will be interesting to see if a PTAS exists for the unweighted case but with reliance only on adjacency.

It is also unclear if a PTAS is possible even with the use of geometry in the weighted case. Recall that the PTAS given in Sections 3 crucially uses the idea of separability of an optimal clique partition. However, in the weighted case, even though a near optimal clique partition in a small region has few cliques, there are examples where any separable partition pays a cost at least factor-2 to that of a near optimal partition. We give an example in Figure 3.

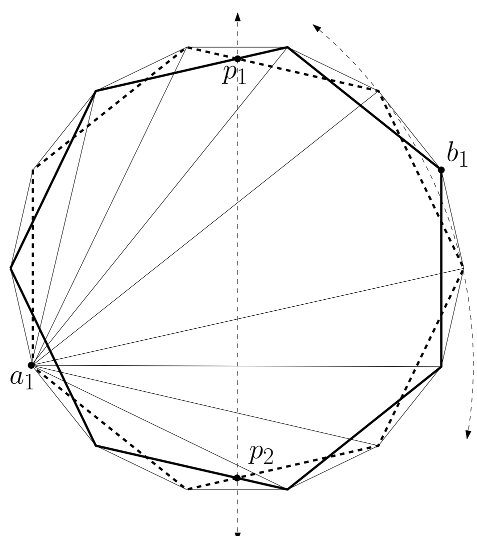


Figure 3: An example showing a gap of at least factor-2 for the weighted case between a partition in which the convex hulls overlap versus a one that is separable.

In the example shown in Figure 3 two cliques of optimal weight are shown: one of them, A, whose vertices are the vertices of the k-gon shown in dashed-heavy lines, and the other, B, whose vertices are the vertices of the k-gon shown in solid-heavy lines. The example is that for $k = 7$. The vertices of A are labeled $a_1, a_2, \ldots, a_k$ in a counter-clockwise fashion. The vertices of B are labeled such that b_i is diametrically opposite to a_i. The distance between a_i and b_i is more than 1 while the distance between a_i and b_j, $i \neq j$ is at most 1. So, there is an edge between a_i to every a_l and to every $b_j, j \neq i$. This is also the case for b_i. In the figure, the edges incident to a_1 are shown by solid-light lines. Also, the dashed arc shows part of the unit disk boundary that is centered at a_1 – note that it does not include b_1. Let the weights of vertices in A be k and the weights of vertices in B be 1. Clearly, opt $\leq k+1$. However, any separable clique partition pays a cost of at least $2k$: if vertices in A must all belong to a common clique, then every vertex in B must belong to a distinct clique in a separable clique partition. Also, note that as-per separability, a line going through $\{p_1, p_2\}$ separates two cliques having weight $2k$ also.

Acknowledgments

We thank Sriram Pemmaraju, Lorna Stewart, and Zoya Svitkina for some very helpful discussions.

References

[1] C. Ambühl, T. Erlebach, M. Mihalák, and M. Nunkesser. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2006.

[2] J. Aspnes, D. K. Goldenberg, and Y. R. Yang. On the computational complexity of sensor network localization. In *ALGOSENSORS '04: First International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pages 32–44, Turku, Finland, 2004. Springer-Verlag.

[3] B. Balasundaram and S. Butenko. Optimization problems in unit-disk graphs. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2832–2844. Springer, 2009.

[4] L. Barenboim and M. Elkin. Distributed $(\Delta+1)$-coloring in linear (in Δ) time. In *STOC*, 2009.

[5] L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti. Latency constrained aggregation in sensor networks. In Yossi Azar and Thomas Erlebach, editors, *ESA*, volume 4168 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 2006.

[6] M. R. Cerioli, L. Faria, T. O. Ferreira, C. A. J. Martinhon, F. Protti, and B. Reed. Partition into cliques for cubic graphs: Planar case, complexity and approximation. *Discrete Applied Mathematics*, 156(12):2270–2278, 2008.

[7] M.R. Cerioli, L. Faria, T.O. Ferreira, and F. Protti. On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. *Electronic Notes in Discrete Mathematics*, 18:73–79, 2004.

[8] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric graphs. In *SODA*, pages 671–679, 2001.

[9] T. Erlebach and E. J. Leeuwen. Approximating geometric coverage problems. In Shang-Teng Huang, editor, *SODA*, pages 1267–1276. SIAM, 2008.

[10] G. Finke, V. Jost, M. Queyranne, and A. Sebö. Batch processing with interval graph compatibilities between tasks. *Discrete Applied Mathematics*, 156(5):556 – 568, 2008.

[11] S. Funke and N. Milosavljević. Network sketching or: "How much geometry hides in connectivity?– Part II". In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 958–967. SIAM, 2007.

[12] J. Gao and L. Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. In *STOC*, pages 483–492. ACM, 2003.

[13] D. Gijswijt, V. Jost, and M. Queyranne. Clique partitioning of interval graphs with submodular costs on the cliques. *Operations Research*, 41(3):275–287, jul 2007.

[14] M.M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Inf. Process. Lett.*, 45(1):19–23, 1993.

[15] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

[16] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Nc-approximation schemes for np- and pspace-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

[17] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computationon growth-bounded graphs. In Pierre Fraigniaud, editor, *DISC*, volume 3724 of *Lecture Notes in Computer Science*, pages 273–287. Springer, 2005.

[18] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the locality of bounded growth. In Marcos Kawazoe Aguilera and James Aspnes, editors, *PODC*, pages 60–68. ACM, 2005.

[19] F. Kuhn, T. Nieberg, T. Moscibroda, and R. Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *DIALM-POMC*, pages 97–103, 2005.

[20] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In Eric Ruppert and Dahlia Malkhi, editors, *PODC*, pages 7–15. ACM, 2006.

[21] K. M. Lillis, S. V. Pemmaraju, and I. A. Pirwani. Topology control and geographic routing in realistic wireless networks. In Evangelos Kranakis and Jaroslav Opatrny, editors, *ADHOC-NOW*, volume 4686 of *Lecture Notes in Computer Science*, pages 15–31. Springer, 2007.

[22] T. Nieberg, J. Hurink, and W. Kern. Approximation schemes for wireless networks. *ACM Transactions on Algorithms*, 4(4):1–17, 2008.

[23] S. Pandit and S. Pemmaraju. Finding facilities fast. In *ICDCN*, 2009.

[24] D. Peleg. *Distributed computing: a locality-sensitive approach.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[25] S. V. Pemmaraju and I. A. Pirwani. Energy conservation via domatic partitions. In Sergio Palazzo, Marco Conti, and Raghupathy Sivakumar, editors, *MobiHoc*, pages 143–154. ACM, 2006.

[26] S. V. Pemmaraju and I. A. Pirwani. Good quality virtual realization of unit ball graphs. In Lars Arge, Michael Hoffmann, and Emo Welzl, editors, *ESA*, volume 4698 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2007.

[27] V. Raghavan and J. Spinrad. Robust algorithms for restricted domains. *J. Algorithms*, 48(1):160–172, 2003.

[28] K. Römer and F. Mattern. The design space of wireless sensor networks. *Wireless Communications, IEEE [see also IEEE Personal Communications]*, 11(6):54–61, Dec. 2004.

[29] J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *PODC*, pages 35–44. ACM, 2008.

[30] H. Simon. On approximate solutions for combinatorial optimization problems. *SIAM J. Discrete Math.*, 3(2):294–310, 1990.

[31] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

A Missing Proofs

This section contains proofs of Lemma 6 from Section 2 and Lemma 15 from Section 3.

Proof of Lemma 6. We prove this by way of defining an appropriate *potential function*, Ψ, over the convex hulls of the cliques in $\mathcal{O}$. The value of Ψ for $\mathcal{O}$ will be the sum of the *perimeters* of the convex hulls of the cliques. We will show that a clique partition of a fixed size having the lowest potential, $\Psi(.)$, has the property that the convex regions of the cliques in the partition are pairwise non-overlapping. First, we define some key terms that are local to this proof.

For a clique C, define the minimum enclosing convex polygon containing C as $\mathbf{conv}\,(C)$. For an m-vertex polygon P, let $\mathbf{perim}\,(P) = \langle p_1, p_2, \ldots, p_m, p_1\rangle$, be the vertices encountered during a walk on the boundary of the polygon in, say, counter-clockwise direction; we call $\mathbf{perim}\,(P)$, the perimeter of P. Define the *length of the perimeter* as: $|\mathbf{perim}\,(P)| = \left(\sum_{i=1}^{m-1} |p_i p_{i+1}|\right) + |p_m p_1|$. If P is a single point, then we say that $|\mathbf{perim}\,(P)| = 0$ and if P is a single edge then $|\mathbf{perim}\,(P)|$ is twice the length of that edge. For any collection of cliques, $\mathcal{C} = \{C_1, C_2, \ldots, C_\ell\}$, define the "potential" of $\mathcal{C}$ as,

$$\Psi(\mathcal{C}) = \sum_{i=1}^{\ell} |\mathbf{perim}\,(\mathbf{conv}\,(C_i))|$$

Consider a clique partition $\mathcal{C}$ having the lowest potential from amongst all clique partitions of size $|\mathcal{C}|$. Suppose there exist two cliques, say C_1 and C_2 such that their convex regions overlap each others'. Let $A = \mathbf{conv}\,(C_1)$, and $B = \mathbf{conv}\,(C_2)$. We will construct a new clique partition, $\mathcal{C}'$ by replacing C_1 and C_2 with $\tilde{C}_1$ and $\tilde{C}_2$ which together contain all the vertices of C_1 and C_2 and the value of $\Psi(\mathcal{C}')$ is strictly smaller than that of $\Psi(\mathcal{C})$, contradicting our assumption.

Let L denote the polygon defined by tracing the *outer-boundary* of $A \cup B$. Partition $A \cup B$ into $A \setminus B$, $B \setminus A$, and $A \cap B$. Let $A \setminus B = \{X_1, X_2, \ldots, X_\alpha\}$, denote the set of connected regions in counter-clockwise order. Similarly, let $B \setminus A = \{Y_1, Y_2, \ldots, Y_\beta\}$ (again in counter-clockwise order). We call each region X_i (and also Y_j) a *petal* (see Figure 4(a)).

Without loss of generality, we assume that no three points are collinear. An immediate consequence of this is that between every two petals X_i, X_{i+1} there is a petal Y_i; so the number of X petals is equal to number of Y petals $\alpha = \beta$. We first prove that $\alpha \geq 2$. By way of contradiction, suppose that $\alpha = 1$. In this case, $\mathbf{conv}\,(A)$ and $\mathbf{conv}\,(B)$ intersect at exactly two points, p_1 and p_2. Let $\tilde{C}_1$ be the set of points in $C_1 \cup C_2$ that are to one side of the line $p_1 p_2$, and $\tilde{C}_2$ be the set of points to the other side of this line (see Figure 4(b)).

Clearly, $\tilde{C}_1$ and $\tilde{C}_2$ are cliques. So we get another clique partition $\mathcal{C}' = (\mathcal{C} \setminus \{C_1, C_2\}) \cup \{\tilde{C}_1, \tilde{C}_2\}$ of the same size as $\mathcal{C}$. We show that $\Psi(\mathcal{C}') < \Psi(\mathcal{C})$, which contradicts our assumption of the minimality of $\Psi(\mathcal{C})$. Observe that it is sufficient to show that $\Psi(\{\tilde{C}_1, \tilde{C}_2\}) < \Psi(\{C_1, C_2\})$. Let I be the convex polygon which is the boundary of region $A \cap B$ and recall that L is the boundary of the region $A \cup B$. Therefore: $\Psi(\{C_1, C_2\}) = |\mathbf{perim}\,(A)| + |\mathbf{perim}\,(B)| = |\mathbf{perim}\,(L)| + |\mathbf{perim}\,(I)|$. Let A' be the

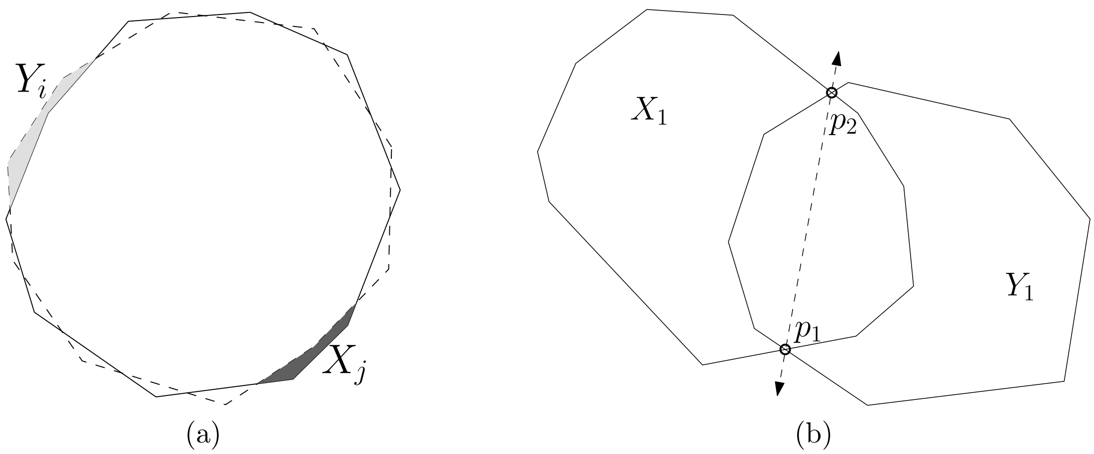


Figure 4: (a) Two overlapping polygons A and B are shown (A with solid line and B with dashed lines) together with two of the petals, one from each. (b) Two overlapping polygons A and B for the case of $\alpha = 1$ petals each.

convex polygon obtained from the segment of A in counter-clockwise order from p_2 until p_1 plus the edge p_1p_2 (see Figure 4(b)). Similarly, define B' to be the polygon obtained by taking the segment of B in counter-clockwise order from p_1 until p_2, followed by edge p_1p_2. It is easy to see that all the points of $\tilde{C}_1$ reside in A' and all the points of C_2 reside in B'. Therefore, the convex hull of the points in $\tilde{C}_1$ will be a convex polygon that is within A' and the convex hull of $\tilde{C}_2$ is within B'. Using the fact that the perimeter of a convex polygon P inside another convex polygon Q is no more than the perimeter of Q, $\Psi(\{\tilde{C}_1, \tilde{C}_2\}) \leq |\mathbf{perim}\,(A')\,| + |\mathbf{perim}\,(B')\,| = |\mathbf{perim}\,(L)\,| + 2 \cdot |p_1p_2|$. Thus, $\Psi(\{C_1, C_2\}) - \Psi(\{\tilde{C}_1, \tilde{C}_2\}) \geq |\mathbf{perim}\,(L)\,| + |\mathbf{perim}\,(I)\,| - (|\mathbf{perim}\,(L)\,| + 2 \cdot |p_1p_2|)$. Next, we show that $|\mathbf{perim}\,(I)\,| > 2 \cdot |p_1p_2|$. Note that, by definition and under the assumptions of this lemma, the area of I is non-zero. Since A and B are both convex polygons, I is also convex, and $\overline{p_1p_2}$ is a chord in I. So $\overline{p_1p_2}$ divides $\mathbf{perim}\,(I)$ into two chains. It is easy to see that the length of one of the chains is strictly greater than $|p_1p_2|$ while the length of the other chain is at least $|p_1p_2|$.

Henceforth, we assume that $\alpha \geq 2$. We assume that the collection of petals appear in the order $X_1, Y_1, X_2, Y_2, \ldots$ going counter-clockwise, and that $X_i = X_{i+\alpha}$ We say two petals X_i and Y_j are *incompatible* if and only if there is a vertex $x_i \in C_1$ contained in X_i, and $y_j \in C_2$ contained in Y_j, such that $\{x_i, y_j\} \notin E$. In this case, the vertices of X_i and Y_j cannot be part of a clique together. We prove the lemma via several claims.

Claim 33. *We cannot have two pairs of incompatible petals, $X_i, Y_{i'}$ and $X_j, Y_{j'}$ where $i \leq i' < j \leq j'$ (see Figure 5).*

Proof. Suppose such two pairs of incompatible petals exist. Then there are points $x_i \in X_i$, $y_{i'} \in Y_{i'}$, $x_j \in X_j$, and $y_{j'} \in Y_{j'}$ where $x_iy_{i'} \notin E$ and $x_jy_{j'} \notin E$. But given that we have $x_ix_j \in E$ and $y_{i'}y_{j'} \in E$ (because $x_i, x_j \in C_1$ and $y_{i'}, y_{j'} \in C_2$), this contradicts Lemma 5. ☐

Claim 34. *Every petal X_i is incompatible with some petal Y_j.*

Proof. By way of contradiction, suppose a petal X_i is not incompatible with any petal Y_j. It means that the points in X_i union with clique C_2 forms a clique. In this case, let $\tilde{C}_1$ be the clique obtained from C_1 by removing the points in X_i, and let $\tilde{C}_2$ be the clique obtained from C_2 by adding the points in X_i. We show that in this case, $|\mathbf{perim}\left(\mathbf{conv}\left(\tilde{C}_1\right)\right)| + |\mathbf{perim}\left(\mathbf{conv}\left(\tilde{C}_2\right)\right)| < |\mathbf{perim}\,(A)\,| + |\mathbf{perim}\,(B)\,|$, contradicting our assumption.

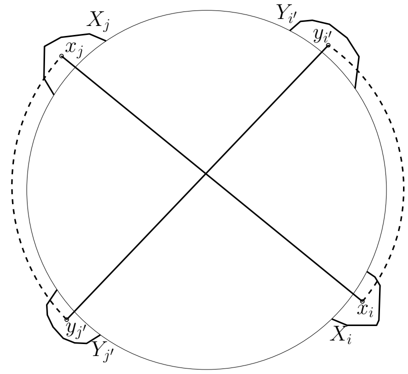


Figure 5: Configuration for the proof of Claim 33. The dashed arcs between a pair of petals represent incompatibility between the corresponding pair. The solid straight lines between a pair of vertices between two petals represents an edge. The large central circle is a schematic representing $A \cap B$ around which, petals are arranged.

Assume that a and b are the intersection points of A and B at petal X_i (see Figure 6). We use $P_{ab}(C_2)$ to denote the length of the segment from a to b (in clock-wise order) on the convex hull of C_2. Similar notation is used for other segments of C_1 and/or C_2. Then:

$$|\textbf{perim}\left(\textbf{conv}\left(\tilde{C}_1\right)\right)| \leq |\textbf{perim}\,(A)\,| - |P_{aq}(C_1)| - |P_{qq'}(C_1)| - |P_{q'b}(C_1)| + |P_{ab}(C_2)| \tag{6}$$

and

$$|\textbf{perim}\left(\textbf{conv}\left(\tilde{C}_2\right)\right)| \leq |\textbf{perim}\,(B)\,| - |P_{pa}(C_2)| - |P_{ab}(C_2)| - |P_{bp'}(C_2)| + |pq| + |P_{qq'}(C_1)| + |q'p'| \tag{7}$$

Therefore, using inequalities (6) and (7) we have:

$$\begin{aligned}|\textbf{perim}\left(\textbf{conv}\left(\tilde{C}_1\right)\right)| + |\textbf{perim}\left(\textbf{conv}\left(\tilde{C}_2\right)\right)| \;&\leq\; |\textbf{perim}\,(A)\,| + |\textbf{perim}\,(B)\,| + |pq| + |q'p'| \\ &\quad -|P_{pa}(C_2)| - |P_{aq}(C_1)| - |P_{q'b}(C_1)| - |P_{bp'}(C_2)|.\end{aligned}$$

By triangular inequality and our assumption that no three points are collinear: $|pq| + |q'p'| < |P_{pa}(C_2)| + |P_{aq}(C_1)| + |P_{q'b}(C_1)| + |P_{bp'}(C_2)|$. This implies that:

$$|\textbf{perim}\left(\textbf{conv}\left(\tilde{C}_1\right)\right)| + |\textbf{perim}\left(\textbf{conv}\left(\tilde{C}_1\right)\right)| < |\textbf{perim}\,(A)\,| + |\textbf{perim}\,(B)\,|,$$

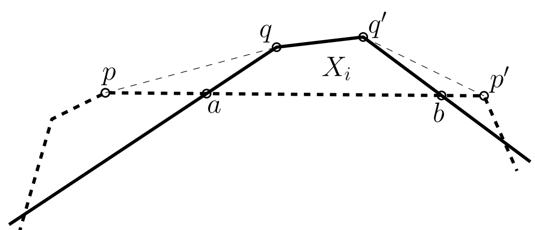
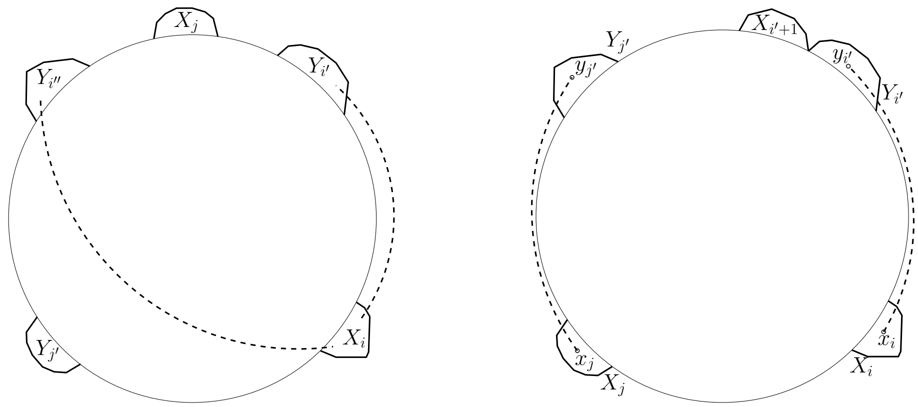


Figure 6: How to merge a petal that is compatible with every other petal; here the solid lines are part of polygon A (the convex hull of C_1) and the thick dashed lines are part of polygon B (the convex hull of C_2).

which contradicts the assumption that we started with a clique partition with the smallest total perimeter. □

Claim 35. *Every petal X_i (or Y_j) is incompatible with at most one other petal.*



(a) (b)

Figure 7: (a) Configuration for the proof of Claim 35; (b) Configuration for the proof of Claim 36. In both (a) and (b), the dashed arcs between a pair of petals represents incompatibility between the pair.

Proof. By way of contradiction, suppose there is a petal, say X_i, that is incompatible with two petals say $Y_{i'}$ and $Y_{i''}$ with $i' < i''$ (see Figure 7(a)). We will show that in this case, every petal X_j (where $i' < j \leq i''$) is contradicting Claim 34 by showing that that X_j cannot be incompatible with any petal. Suppose that X_j is incompatible with $Y_{j'}$. $Y_{j'}$ is either not between X_i and $Y_{i'}$ in counter-clockwise order (i.e. $j' \notin \{i, \ldots, i'\}$) or it is not between $Y_{i''}$ and X_i (i.e. $j' \notin \{i'', \ldots, i\}$). Without loss of generality assume $Y_{j'}$ is not between X_i and $Y_{i'}$ (see Figure 7(a)). But then the two pairs $X_i, Y_{i'}$ and $X_j, Y_{j'}$ contradict Claim 33. □

Claims 34 and 35 imply that every petal is incompatible with exactly one other petal.

Claim 36. *We cannot have two incompatible pairs $X_i, Y_{i'}$ and $X_j, Y_{j'}$ where $i \leq i' < j' < j$ (see Figure 7(b)).*

Proof. By way of contradiction, suppose such two pairs exist. Consider a petal that is between $Y_{i'}$ and $Y_{j'}$, say $X_{i'+1}$ and suppose it is incompatible with some petal Y_ℓ. If Y_ℓ is not between X_i and $Y_{i'}$ (in counter-clockwise order) then the two pairs $X_i, Y_{i'}$ and $X_{i'+1}, Y_\ell$ contradict Claim 33. Similarly, if Y_ℓ is not between $Y_{j'}$ and X_j (in counter-clockwise order) then the two pairs $X_j, Y_{j'}$ and $X_{i'+1}, Y_\ell$ contradict Claim 33. □

Claim 37. *We cannot have two incompatible pairs $X_i, Y_{i'}$ and $Y_i, X_{i'+1}$ with $i < i'$ (see Figure 8(a)).*

Proof. By way of contradiction, suppose we have such two pairs. Let p_1 be the intersecting points of A and B common to X_i and Y_i and p_2 be the intersecting point of A and B common to $Y_{i'}$ and $X_{i'+1}$. Consider the line that runs between p_1 and p_2. This line partitions the points in $C_1 \cup C_2$ into two parts. We claim that each of these two parts forms a clique and the sum of perimeters of the convex hulls of these two cliques is smaller than that of C_1 and C_2, contradicting the assumption that we started with a clique partition with minimum $\Psi(.)$ value.

To prove that, we show that every two petals between Y_i and $Y_{i'}$ and including these two (in counter-clockwise order) are compatible and so they all form a clique. By the same argument (and symmetry) all the petals between $X_{i'+1}$ and X_i and including these two form a clique. Consider two petals, say X_l

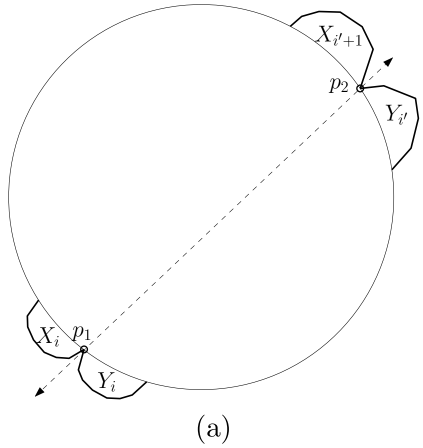


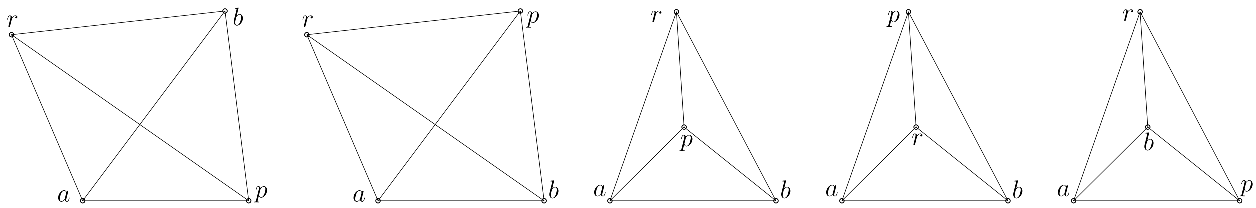
Figure 8: (a) Configuration for the proof of Claim 37; (b) The configuration for the last paragraph proof of Lemma 6.

and Y_j where $i < l \leq i'$ and $i < j < i'$; without loss of generality assume that $j < l$ (see Figure 8(b)). If X_l and Y_j are incompatible, then the two pairs X_l, Y_j and $X_{i'+1}, Y_i$ contradict Claim 36.

So, the collection of petals between Y_i and $Y_{i'}$ forms a clique, and the collection of petals between $X_{i'+1}$ and X_i also forms a clique. So, let all points of $C_1 \cup C_2$ to one side of line p_1p_2 form $\tilde{C}_1$ and all points of $C_1 \cup C_2$ to the other side of this line form $\tilde{C}_2$. Then: $\mathcal{C}' = (\mathcal{C} \setminus \{C_1, C_2\}) \cup \{\tilde{C}_1, \tilde{C}_2\}$. We show that $\Psi(\{\tilde{C}_1, \tilde{C}_2\}) < \Psi(\{C_1, C_2\})$ to get the desired contradiction. (this is similar to proof of case $\alpha = 1$, which we prove earlier). Let I be the convex polygon on the boundary of region $A \cap B$. By definition, $\Psi(\{C_1, C_2\}) = |\mathbf{perim}(A)| + |\mathbf{perim}(B)|$. But this sum can be decomposed exactly into the the length of the perimeter of I and the length of the perimeter of L. Therefore, $\Psi(\{C_1, C_2\}) = |\mathbf{perim}(I)| + |\mathbf{perim}(L)|$. Recall, that L is the outer-face of $A \cup B$. Also, it can be easily seen that $\Psi(\{\tilde{C}_1, \tilde{C}_2\}) \leq |\mathbf{perim}(L)| + 2 \cdot |p_1p_2|$ (here again we are implicitly using the fact that if we have two nested convex polygon, the perimeter of the one inside is no more than the perimeter of the one that is outside). So, $\Psi(\{C_1, C_2\}) - \Psi(\{\tilde{C}_1, \tilde{C}_2\}) \geq |\mathbf{perim}(L)| + |\mathbf{perim}(I)| - (|\mathbf{perim}(L)| + 2 \cdot |p_1p_2|)$. It is easy to see that $|\mathbf{perim}(I)| > 2 \cdot |p_1p_2|$ by observing that p_1p_2 is a chord (or an edge) of this convex polygon and the area of I (that is $A \cap B$) is non-zero. □

Now we derive the final contradiction by showing that if A and B are overlapping (as assumed) then one of the Claims 33- 37 must be violated. Consider an incompatible pair, say $X_i, Y_{i'}$. Without loss of generality, assume $i \leq i'$. By Claims 34 and 35, $X_{i'+1}$ is incompatible with exactly one petal and by Claim 36, this petal (which is incompatible with $X_{i'+1}$) must be between X_i and $Y_{i'}$; let's call it Y_j. By Claim 37 $j \neq i$, so we can assume $i < j < i'$. Now consider Y_i. It must be incompatible with exactly one petal, say X_ℓ. If X_ℓ is not in between Y_j and $X_{i'+1}$ (in counter-clockwise), then the two pairs X_ℓ, Y_i and $Y_j, X_{i'+1}$ will contradict one of Claims 33 or 36. So we must have $j < \ell \leq i'$ (See Figure 8(b)). But in this case, the two pairs $X_i, Y_{i'}$ and X_ℓ, Y_i contradict Claim 33. □

Proof of Lemma 15. First, we describe how to detect if the quadrilateral on these four points is convex or concave. If the quadrilateral is concave, then one of the points will be inside the triangle formed by the other three. There are three possible cases: r is inside, p is inside, or one of a or b is inside (see Figure 9(c)-(e)). There are four triangles each of which is over three of these four points. The quadrilateral is concave if the sum of the areas of three of these triangles is equal to the area of the fourth triangle. Equivalently, it is convex if sum of areas of two of the triangles is equal to the sum of areas of the other two. Given a triangle with edge lengths x, y, z, using Heron's formula, the area of the triangle is equal to $\sqrt{2(x^2y^2 + y^2z^2 + z^2x^2) - (x^4 + y^4 + z^4))/4}$. So the area of a triangle is of the form $\sqrt{A}$ where A is a polynomial in terms of lengths of the edges of the triangle. Suppose that the areas of the four triangles over these four points are $\sqrt{A_1}$, $\sqrt{A_2}$, $\sqrt{A_3}$, and $\sqrt{A_4}$. We need to check if the sum of two is equal to the sum of the other two and we would like to do this without computing the square roots of numbers. For instance, suppose we want to verify $\sqrt{A_1}+\sqrt{A_2}=\sqrt{A_3}+\sqrt{A_4}$. For this to hold, we must have $A_1+A_2+2\sqrt{A_1A_2}=A_3+A_4+2\sqrt{A_3A_4}$. Verifying this is equivalent to verifying $D+\sqrt{A_1A_2}=\sqrt{A_3A_4}$ where $D=\frac{1}{2}(A_1+A_2-A_3-A_4)$. Taking the square of both sides, we need to have $D^2+A_1A_2+2D\sqrt{A_1A_2}=A_3A_4$, which is the same as $\frac{1}{4}(A_3A_4-D^2-A_1A_2)^2=A_1A_2D^2$. Thus by comparing two polynomials of edge-lengths (and without computing square roots) we can check if the quadrilateral is convex or concave.



(a) (b) (c) (d) (e)

Figure 9: The five non-isomorphic configurations needed to consider for a quadrilateral on four points in Lemma 15

Suppose the quadrilateral is convex. If r and p are on two opposite corners (see Figure 9(a)), then r and p are on different sides. In this case $|rp|+|ab|>|ra|+|bp|$ and $|rp|+|ab|>|rb|+|ap|$. If rp is one of its sides (see Figure 9(b)), then $|rp|+|ab|$ is not the largest of the above three pairs of sums.

Now suppose that the quadrilateral is concave. The only case in which r and p are on two sides of line ab is when one of a or b is inside the triangle obtained by the other three (see Figure 9(e)). In this case, the area of the largest triangle is the one that does not contain a or b. Thus, if we compute the square of the areas of the four triangle, we can detect this case too. $\square$