An Improved Analysis for a Greedy Remote-Clique Algorithm Using Factor-Revealing LPs*

Benjamin Birnbaum
Department of Computer Science and Engineering
University of Washington, Seattle
Box 352350
Seattle, WA 98105-2350
birnbaum@cs.washington.edu

Kenneth J. Goldman
Department of Computer Science and Engineering
Washington University in St. Louis
Campus Box 1045
One Brookings Drive
St. Louis, Missouri 63130-4899
kjg@cse.wustl.edu

Abstract

Given a positive integer k and a complete graph with non-negative edge weights satisfying the triangle inequality, the *remote-clique* problem is to find a subset of k vertices having a maximum-weight induced subgraph. A greedy algorithm for the problem has been shown to have an approximation ratio of 4, but this analysis was not shown to be tight. In this paper, we use the technique of *factor-revealing linear programs* to show that the greedy algorithm actually achieves an approximation ratio of 2, which is tight.

Keywords: approximation algorithms, dispersion, factor-revealing linear programs, remote-clique

1 Introduction

Motivation

As computing system reliability becomes increasingly important, techniques are being developed to ensure that essential services can survive in spite of a limited number of arbitrary faults, including communication failures, crash failures, software errors, and malicious attacks. These techniques require replication of data and processing so that when failures occur in some replicas, the others can continue to correctly execute operations to completion. As an example, the CLBFT algorithm [2], uses a group of $3f + 1$ replicated data servers to guarantee correct performance when up to f

*This research was supported in part by the National Science Foundation under grant CISE-EI-0305954 and was performed while the first author was at Washington University in St. Louis. A preliminary version of this paper appears in *RANDOM-APPROX '06*, volume 4110 of *Lecture Notes in Computer Science*, pp. 49–60, 2006.

of the replica servers exhibit faulty behavior. However, such algorithms are only beneficial if the faults are *independent*. Otherwise, a single failure type (such as a power outage affecting all servers in a region) could induce faults in multiple server replicas, exceeding the fault-tolerance bound f. Therefore, in choosing a set of k replicas from a large group of potential servers, we desire a subset that is as diverse as possible with respect to properties such as system architecture, software implementation, physical location, and the network administration.

This diversification problem can be modeled with a graph. For each potential server, create a vector of attributes representing its properties (architecture, software, location, etc.) and define the distance between two servers to be the *Hamming distance* [9] between the vectors (the number of corresponding attributes that differ). For each server, create a vertex in a complete graph. Let the edge weight between two vertices be the distance between their corresponding servers. Finding a subset of servers that is most diverse now reduces to the problem of finding a subset of k vertices having an induced subgraph with maximum average edge weight (or equivalently maximum total edge weight).

This example provides one of several motivations for a general class of graph optimization problems called *dispersion* problems, which involve finding subsets of vertices that are in some way as distant from each other as possible. Other motivations for dispersion problems include choosing locations for facilities that interfere with each other and providing good starting solutions for some TSP heuristics. (Chandra and Halldórsson [3] discuss these and other motivations.) Maximizing the average weight of the induced subgraph is the measure of dispersion addressed here, but it is possible to measure dispersion in a number of other ways, including the minimum weight edge and the minimum weight spanning tree of the induced subgraph. As in [3], we call the problem of maximizing the average weight the *remote-clique* problem, but it has also been called *maxisum dispersion* [13] and *max-avg facility dispersion* [17].

Related Work

Approximation algorithms for dispersion problems have been widely studied (see, for example, [1, 3, 4, 6, 8, 10, 17, 18, 19, 20]). Ravi, Rosenkrantz, and Tayi [17] provide a simple greedy algorithm for *remote-clique* with running time $O(n^2)$ and prove that it achieves an approximation ratio of 4. They provide an instance of the problem in which the optimal solution has twice the weight of the algorithm's solution, but they leave open the question of whether a tighter bound for the algorithm can be shown. It is this algorithm that we prove has an approximation ratio of 2. Hassin, Rubinstein, and Tamir [10] provide a more complicated algorithm with a running time of $O(n^2 + k^2 \log k)$ that they show has an approximation ratio of 2. However, a tight approximation ratio for the faster and simpler greedy algorithm has not been proved until this paper.

The *remote-clique* problem is similar to the *dense k-subgraph* problem, which, given an unweighted graph, is to find a subset of k vertices having the most edges in the induced subgraph. Feige, Peleg, and Kortsarz [5] provide an algorithm for this problem that achieves an approximation ratio slightly better than $O(n^{1/3})$.

Our Result

We provide an algorithm parameterized by an integer d called d-Greedy Augment. When $d = 1$, the algorithm is nearly the same as the algorithm analyzed in [17], and when $d = k$, the algorithm amounts to examining all subsets of k vertices and choosing the one with maximum edge weight.

(Clearly, this will return an optimal solution, but it does not run in polynomial time if k is not a constant.) We will show that d-GREEDY AUGMENT has an approximation ratio of $(2k-2)/(k+d-2)$ and has a running time of $O(kdn^d)$. Therefore if $d = 1$, our algorithm guarantees an approximation ratio of 2 and has a running time of $O(kn)$. This algorithm, then, is the fastest known (and easiest to implement) 2-approximation for the *remote-clique* problem.

Our Technique

To prove an approximation ratio of $(2k-2)/(k+d-2)$, we use the technique of *factor-revealing linear programs*. This technique was used in [7, 14, 15] before it was explicitly named and used in [11, 12, 16]. It is a simple generalization of a method often used to provide bounds for approximation algorithms. Consider a maximization (resp., minimization) problem P. A typical analysis of an approximation algorithm ALG for P proceeds by using the behavior of ALG and the structure of P to generate a number of inequalities. These inequalities are then combined to provide a bound on the ratio of the value of the solution obtained by ALG to that of an optimal solution. Often, this can be done by simple manipulations, but not always. A more general way of obtaining a bound is to view the process as an optimization problem Q in its own right, in which an adversary tries to minimize (resp., maximize) the value of ALG's solution to P subject to the constraints given by the generated inequalities. The optimal solution to Q is then a bound on the performance of ALG. If Q can be formulated as a linear program, then this is a *factor-revealing LP*, which can then be solved using duality.

This technique is applicable to many problems, but in most cases it does not seem to be the easiest way to provide a bound. However, there are some algorithms, including the one in [11] and the greedy algorithm examined here, in which it is the only known technique to provide a tight bound. Further discussion of this technique and its advantages is provided in [11, 12].

Organization

In Section 2, we establish notation, state the problem formally, and prove some combinatorial identities that will be used later in the paper. We describe the greedy algorithm and analyze its running time in Section 3. In Section 4, we show that our algorithm achieves an approximation ratio of $(2k-2)/(k+d-2)$, and in Section 5, we show that this ratio is tight. We conclude in Section 6 with a discussion and directions for future research.

2 Preliminaries

Notation

We define the following notation that will be used throughout this paper:

- For any natural number n, let $[n] = \{1, \ldots, n\}$.
- For any integers n and j such that $n \geq 1$ and $j \leq n$, let

$$\binom{n}{j} = \begin{cases} \frac{n!}{j!(n-j)!} & \text{if } j \geq 0 \\ 0 & \text{otherwise} \end{cases} .$$

There will be several times in this paper when we will write $\binom{n}{j}$ when j could be less than 0, so it is important to keep in mind the full definition of $\binom{n}{j}$.

- For any set S and integer j, let $\binom{S}{j} = \{S' \subseteq S \ : \ |S'| = j\}$. If $j < 0$ or $j > |S|$, then $\binom{S}{j} = \emptyset$.

Problem Definition

Let $G = (V, E)$ be a complete graph with the weight of edge $\{v_1, v_2\} \in E$ given by $w(v_1, v_2)$. The edge weights are nonnegative and satisfy the triangle inequality: for all distinct $v_1, v_2, v_3 \in V$, $w(v_1, v_2) + w(v_2, v_3) \geq w(v_1, v_3)$. For a given integer parameter k, such that $1 \leq k \leq |V|$, the *remote-clique* problem is to find a subset $V' \subseteq V$ such that $|V'| = k$ and the average edge weight in V',

$$\frac{2}{k(k-1)} \sum_{\{v_1, v_2\} \in \binom{V'}{2}} w(v_1, v_2) \ ,$$

is maximized. This problem can be shown to be NP-hard by a simple reduction from *clique*.

Combinatorial Identities

The following combinatorial identities will be used in the analysis of our algorithm.

Lemma 1. *Let n be a positive integer and let i and j be two positive integers such that $j \leq i \leq n$. If f is some function defined on the domain $\binom{[n]}{j}$, then*

$$\sum_{I \in \binom{[n]}{i}} \sum_{J \in \binom{I}{j}} f(J) = \binom{n-j}{i-j} \sum_{J \in \binom{[n]}{j}} f(J) \ .$$

Proof. To prove this, we count the number of times that a given $J \in \binom{[n]}{j}$ appears in the summation on the left. Each J appears once for each $I \in \binom{[n]}{i}$ containing J. The number of such sets I is $\binom{n-j}{i-j}$, since for a given J, we can choose $i - j$ elements from the $n - j$ elements in $[n] - J$ to form such an I. □

Lemma 2. *For all integers j, d, and s, such that $0 \leq j \leq d \leq s$,*

$$\binom{s}{d}\binom{d}{j} = \binom{s}{j}\binom{s-j}{d-j} \ .$$

Proof. The lemma follows immediately from the definition of the binomial coefficient. □

3 The Algorithm

In this section, we describe our algorithm d-Greedy Augment and analyze its running time. The algorithm maintains a set of vertices T, initially empty. At each step, it augments T with the set of d vertices that add the most weight to T. When $|T| = k$, d-Greedy Augment returns the set

T. (Throughout this paper, we assume for simplicity that d divides k evenly.) To be more precise, we define the following notation. For any subset of vertices $V' \subseteq V$ that is disjoint from T, let

$$\text{aug}_T(V') = \sum_{v' \in V'} \sum_{v \in T} w(v', v) + \sum_{\{v'_1, v'_2\} \in \binom{V'}{2}} w(v'_1, v'_2) \ .$$

In other words, $\text{aug}_T(V')$ is the amount of edge weight that the vertices in V' would contribute to T if T is augmented by V'. At each step in the algorithm, d-Greedy Augment chooses a set of d vertices V' that maximizes $\text{aug}_T(V')$ and adds them to T. For the first step, this means that d-Greedy Augment chooses a group of d vertices with the greatest edge weights.[1]

An implementation of d-Greedy Augment is listed below as Algorithm 1. The running time is dominated by the outer while loop, which runs k/d times. Each iteration of the while loop performs $O(n^d)$ iterations through the candidate subsets V', and each of these iterations takes $O(d^2)$ time to compute the sum to add to $aug(V')$. Therefore, the overall running time of this algorithm is $O(kdn^d)$.

Algorithm 1 d-Greedy Augment

```
{Initialize aug array with total weight of each clique of size d.}
for all V' ⊆ V such that |V'| = d do
   aug[V'] ← Σ_{{v'_1,v'_2}∈C(V',2)} w(v'_1, v'_2)
end for
T ← ∅
while |T| < k do
   MaxWeight ← −∞
   {Choose the d-clique with maximum contribution.}
   for all V' ⊆ V − T such that |V'| = d do
      if aug[V'] > MaxWeight then
         MaxWeightSet ← V'
         MaxWeight ← aug[V']
      end if
   end for
   T ← T ∪ MaxWeightSet
   {Update all clique contributions to include edges incident to newly added vertices in T.}
   for all V' ⊆ V − T such that |V'| = d do
      aug[V'] ← aug[V'] + Σ_{v∈MaxWeightSet} Σ_{v'∈V'} w(v, v')
   end for
end while
return T
```

Because the running time of d-Greedy Augment is exponential in d, it only runs in polynomial time for constant values of d. Furthermore, *remote-clique* can only be shown to be NP-hard for non-constant values of k. (For constant values of k, we can just examine every subset of k vertices in

[1]Note that if $d = 1$, then during the first step $\text{aug}_T(V') = 0$ for all $V' \subseteq V$ such that $|V'| = d$. Thus for $d = 1$, d-Greedy Augment just starts with an arbitrary vertex. This is, in fact, the only difference between d-Greedy Augment for $d = 1$ and the algorithm analyzed in [17]; the algorithm in [17] initializes T to two vertices that are endpoints of a maximum weight edge.

polynomial time and choose the maximum weight subset). Therefore, increasing the value of d does not affect the approximation ratio asymptotically. However, we include the analysis for all values of d both for completeness and also because it could have some practical benefit for small values of k. For example, if $k = 4$, then the naive exact algorithm would take quartic time, whereas if we were willing to spend quadratic time, we could run d-Greedy Augment for $d = 2$ to guarantee finding a solution at least $\frac{2}{3}$ the value of the optimal solution.

4 Upper Bound

In this section, we prove that d-Greedy Augment achieves an approximation ratio of $(2k-2)/(k+d-2)$. For an instance of the *remote-clique* problem, let OPT be the average edge weight in an optimal solution. To prove our approximation ratio, we will show that at each augmenting step, there exists a group of d vertices V' for which $\text{aug}_T(V')$ is sufficiently high. The following lemma, which is the main result of this section, states this fact. Part of it is proved here, and part of it is proved by the sub-cases given by Lemmas 5, 6, 7, 8, and 9.

Lemma 3. *Before each augmenting step in the algorithm, there exists a group of d vertices $V' \subseteq V$ such that V' is disjoint from T and $\text{aug}_T(V') \geq \frac{1}{2}d(|T| + d - 1)OPT$.*

Proof. We defer the proof of the lemma when $|T| > 0$ to the remainder of this section. When $|T| = 0$, then the statement of the lemma is that there exists at least one group of vertices $V' \subseteq V$ of size d and total weight at least $\binom{d}{2}OPT$. Let S be the set vertices in an optimal solution. We prove that such a group of vertices V' must be found as a subset of S. By the optimality of S,

$$\sum_{\{v_1,v_2\}\in\binom{S}{2}} w(v_1, v_2) = \binom{k}{2}OPT = \frac{\binom{k}{d}\binom{d}{2}}{\binom{k-2}{d-2}}OPT \ ,$$

where the second equality follows by Lemma 2. Thus,

$$\binom{k-2}{d-2} \sum_{\{v_1,v_2\}\in\binom{S}{2}} w(v_1, v_2) = \binom{k}{d}\binom{d}{2}OPT \ .$$

By Lemma 1, we have

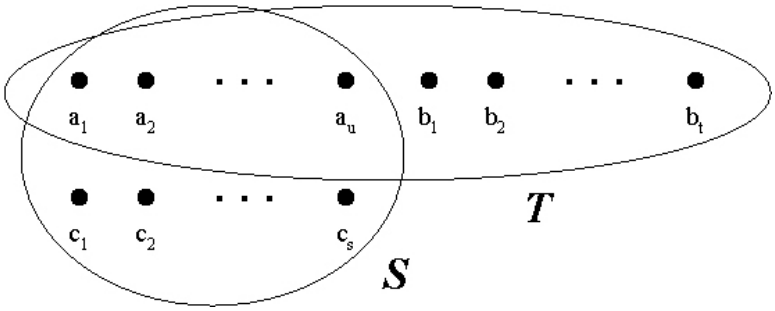
$$\sum_{V'\in\binom{S}{d}} \sum_{\{v_1,v_2\}\in\binom{V'}{2}} w(v_1, v_2) = \binom{k}{d}\binom{d}{2}OPT \ ,$$

which, by an averaging argument, implies the lemma for $|T| = 0$. □

Before proving Lemma 3 for $|T| > 0$, we show that it implies our main result, that d-Greedy Augment achieves an approximation ratio of $(2k-2)/(k+d-2)$. This result is given in the following theorem.

Theorem 4. *The average weight of the edges in the solution returned by d-Greedy Augment is at least $(k+d-2)/(2k-2) \cdot OPT$. (Hence, d-Greedy Augment is a 2-approximation.)*

Figure 1: An intermediate state of d-Greedy Augment.



Proof. Since d-Greedy Augment adds d vertices to T during each augmenting step, we have that $|T| = (i-1)d$ before the i^{th} augmenting step. Thus by Lemma 3, before the i^{th} augmenting step there exists a set of d vertices V' disjoint from T such that $\text{aug}_T(V') \geq \frac{1}{2}d(di-1)OPT$. Therefore, the weight of the edges added by d-Greedy Augment on the i^{th} augmenting step is at least $\frac{1}{2}d(di-1)OPT$, since d-Greedy Augment chooses the set of d vertices that maximizes $\text{aug}_T(V')$. Thus after j steps, the weight of the edges in T is at least

$$\sum_{i=1}^{j} \frac{1}{2}d(di-1)OPT = \frac{1}{4}dj(dj+d-2)OPT \ .$$

Since the algorithm terminates after k/d steps, the final weight of the edges in T is at least $\frac{1}{4}k(k+d-2)OPT$. Since there are $k(k-1)/2$ edges in this set of vertices, we conclude that the average weight of the edges in the solution returned by d-Greedy Augment is at least $(k+d-2)/(2k-2)\cdot OPT$. □

It remains to prove Lemma 3 for $|T| > 0$. We begin with some notation. Consider an intermediate state of d-Greedy Augment. The set of vertices chosen so far is T, and let S be the set of vertices in an optimal solution. Let $u = |S \cap T|$, $t = |T - S|$, and $s = |S - T|$. Arbitrarily label the vertices in $S \cap T$ as $a_1, a_2, \ldots, a_u$, the vertices in $T - S$ as $b_1, b_2, \ldots, b_t$, and the vertices in $S - T$ as $c_1, c_2, \ldots, c_s$. Note that one of u or t may be equal to zero, but since we are in an intermediate state of d-Greedy Augment, we know that $d \leq |T| \leq |S| - d$, and hence $u + t \geq d$ and $s \geq t + d$. This situation is illustrated in Figure 1.

We break the proof of Lemma 3 into five cases based on the values of u and t and state the proof for each case as its own lemma. For each of these cases, we show that a group of d vertices satisfying the condition of Lemma 3 can be found in the set $S - T$. We start with the case when S and T are disjoint, i.e., when $u = 0$ and $t \geq 1$. This case permits a direct analysis without the use of linear programming. We note that this case was proved for the special case of $d = 1$ in [17].

Lemma 5. *Lemma 3 holds when $u = 0$ and $t \geq 1$. Specifically, there exists a set of indices $L \in \binom{[s]}{d}$ such that*

$$\sum_{\ell \in L} \sum_{j \in [t]} w(b_j, c_\ell) + \sum_{\{\ell, m\} \in \binom{L}{2}} w(c_\ell, c_m) \geq \frac{1}{2}d(d+t-1)OPT \ .^2$$

[2]Note that if $d = 1$, then the second sum in this inequality is empty. In general, there will be a number of formulas

Proof. The intuition behind our proof is that by optimality, the edges in S must have high weight on average. Hence, by the triangle inequality, edges adjacent to those in S must also have high weight on average. We can show use this to show that there is at least one set of d vertices in S that adds enough weight to T.

Specifically, the triangle inequality implies

$$w(b_j, c_\ell) + w(b_j, c_m) \geq w(c_\ell, c_m) \qquad j \in [t], \{\ell, m\} \in \binom{[s]}{2} \ .$$

Summing over all j, this becomes

$$\sum_{j\in[t]} w(b_j, c_\ell) + \sum_{j\in[t]} w(b_j, c_m) \geq tw(c_\ell, c_m) \qquad \{\ell, m\} \in \binom{[s]}{2} \ .$$

Now, summing over all $\{\ell, m\}$ yields

$$\sum_{\{\ell,m\}\in\binom{[s]}{2}} \left(\sum_{j\in[t]} w(b_j, c_\ell) + \sum_{j\in[t]} w(b_j, c_m) \right) \geq t \sum_{\{\ell,m\}\in\binom{[s]}{2}} w(c_\ell, c_m) = t\binom{s}{2} OPT \ , \tag{1}$$

where the equality follows from the optimality of S. By applying Lemma 1 to the left-hand side of (1), we can rewrite it as

$$(s-1) \sum_{\ell\in[s]} \sum_{j\in[t]} w(b_j, c_\ell) \geq t\binom{s}{2} OPT \ ,$$

which can be simplified to

$$\sum_{\ell\in[s]} \sum_{j\in[t]} w(b_j, c_\ell) \geq \frac{st}{2} OPT \ .$$

From this fact, along with the optimality of S, it follows that

$$\begin{aligned} \binom{s-1}{d-1} \sum_{\ell\in[s]} \sum_{j\in[t]} w(b_j, c_\ell) + \binom{s-2}{d-2} \sum_{\{\ell,m\}\in\binom{[s]}{2}} w(c_\ell, c_m) &\geq \left(\binom{s-1}{d-1} \frac{st}{2} + \binom{s-2}{d-2} \binom{s}{2} \right) OPT \\ &= \frac{1}{2} \binom{s}{d} d(d+t-1) OPT \ , \end{aligned} \tag{2}$$

where the equality is obtained from some simplification using Lemma 2.[3] We can now apply Lemma 1 to the left-hand side of this inequality to obtain

$$\sum_{L\in\binom{[s]}{d}} \left(\sum_{\ell\in L} \sum_{j\in[t]} w(b_j, c_\ell) + \sum_{\{\ell,m\}\in\binom{L}{2}} w(c_\ell, c_m) \right) \geq \frac{1}{2} \binom{s}{d} d(d+t-1) OPT \ .$$

But this implies that there must exist at least one $L \in \binom{[s]}{d}$ such that

$$\sum_{\ell\in L} \sum_{j\in[t]} w(b_j, c_\ell) + \sum_{\{\ell,m\}\in\binom{L}{2}} w(c_\ell, c_m) \geq \frac{1}{2} d(d+t-1) OPT \ .$$

□

in this section that simplify significantly if $d = 1$. However, we use the general form for compactness, and unless otherwise noted, all statements hold for any $d \geq 1$.

[3] Again, note the difference in the formula if $d = 1$. In this case, $\binom{s-2}{d-2} = 0$.

Now that we have proved Lemma 3 for the case when T is disjoint from S, we turn to proving Lemma 3 when some number of vertices in T are also in S. Intuitively, the algorithm should do no worse when T is not disjoint from S. If d-Greedy Augment has already found some of the optimal solution, then that should not hurt its performance. However, the inequalities obtained when following an approach similar to the one used in Lemma 5 are more complicated. We use factor-revealing linear programs to deal with this complexity. We start with the most general non-disjoint case that we examine, when $u \geq 2$ and $t \geq 1$. (Our application of the triangle inequality leads to certain boundary cases that arise for other values of u and t, which we handle separately.)

Lemma 6. *Lemma 3 holds when $u \geq 2$ and $t \geq 1$. Specifically, there exists a set of indices $L \in \binom{[s]}{d}$ such that*

$$\sum_{\ell \in L} \left(\sum_{h \in [u]} w(a_h, c_\ell) + \sum_{j \in [t]} w(b_j, c_\ell) \right) + \sum_{\{\ell,m\} \in \binom{L}{2}} w(c_\ell, c_m) \geq \frac{1}{2} d(d+t+u-1) OPT \ .$$

Proof. It is sufficient to show that

$$\binom{s-1}{d-1} \sum_{\ell \in [s]} \left(\sum_{h \in [u]} w(a_h, c_\ell) + \sum_{j \in [t]} w(b_j, c_\ell) \right) + \binom{s-2}{d-2} \sum_{\{\ell,m\} \in \binom{[s]}{2}} w(c_\ell, c_m) \geq \frac{1}{2} \binom{s}{d} d(d+t+u-1) OPT \ , \quad (3)$$

since we can then proceed to prove this lemma as we proved Lemma 5 from inequality (2). To prove inequality (3), we again use the triangle inequality and the optimality of S to show that edges between T and $S - T$ have a high enough average weight. However, because S and T now overlap, we have the following more complicated set of inequalities, broken into three groups based on three types of triangles between T and $S - T$:

$$w(a_h, c_\ell) + w(a_h, c_m) - w(c_\ell, c_m) \geq 0 \qquad h \in [u], \{\ell, m\} \in \binom{[s]}{2} \quad (4)$$

$$w(b_j, c_\ell) + w(b_j, c_m) - w(c_\ell, c_m) \geq 0 \qquad j \in [t], \{\ell, m\} \in \binom{[s]}{2} \quad (5)$$

$$w(a_h, c_\ell) + w(a_i, c_\ell) - w(a_h, a_i) \geq 0 \qquad \{h, i\} \in \binom{[u]}{2}, \ell \in [s] \ . \quad (6)$$

By the optimality of S, we have

$$\sum_{\{h,i\} \in \binom{[u]}{2}} w(a_h, a_i) + \sum_{\{\ell,m\} \in \binom{[s]}{2}} w(c_\ell, c_m) + \sum_{h \in [u]} \sum_{\ell \in [s]} w(a_h, c_\ell) \geq \binom{s+u}{2} OPT \ . \quad (7)$$

At this point in the proof of Lemma 5, it was possible to combine the inequalities expressing the triangle inequality and the optimality of S to yield (2) and finish the proof. In this lemma, however, the inequalities have a much more complicated form because of the overlap of S and T, and we do not know of a simple way to combine them to yield (3). Instead, our approach is to write the inequalities as a linear program and use duality as a systematic way of determining how to combine them. Specifically, consider an adversary trying to minimize

$$\binom{s-1}{d-1} \sum_{\ell \in [s]} \left(\sum_{h \in [u]} w(a_h, c_\ell) + \sum_{j \in [t]} w(b_j, c_\ell) \right) + \binom{s-2}{d-2} \sum_{\{\ell,m\} \in \binom{[s]}{2}} w(c_\ell, c_m)$$

subject to the constraints given by (4), (5), (6), and (7). If we can show that the optimal value of this factor-revealing linear program (where the variables are the weights of the edges) is at least $\frac{1}{2}\binom{s}{d}d(d+t+u-1)OPT$, then we will have proven (3). Since the value of any feasible dual solution is a lower bound for the optimal value of the primal, we can prove (3) by finding a feasible dual solution with value $\frac{1}{2}\binom{s}{d}d(d+t+u-1)OPT$. The dual linear program is

maximize

$$\binom{s+u}{2}OPT \cdot z$$

subject to

$$-\sum_{\ell\in[s]} y_{\{h,i\},\ell} + z \leq 0 \qquad \{h,i\} \in \binom{[u]}{2} \tag{8}$$

$$-\sum_{h\in[u]} w_{h,\{\ell,m\}} - \sum_{j\in[t]} x_{j,\{\ell,m\}} + z \leq \binom{s-2}{d-2} \qquad \{\ell,m\} \in \binom{[s]}{2} \tag{9}$$

$$\sum_{m\in[s]-\{\ell\}} w_{h,\{\ell,m\}} + \sum_{i\in[u]-\{h\}} y_{\{h,i\},\ell} + z \leq \binom{s-1}{d-1} \qquad h \in [u], \ell \in [s] \tag{10}$$

$$\sum_{m\in[s]-\{\ell\}} x_{j,\{\ell,m\}} \leq \binom{s-1}{d-1} \qquad j \in [t], \ell \in [s] \quad , \tag{11}$$

where $w_{h,\{\ell,m\}}$ corresponds to (4), $x_{j,\{\ell,m\}}$ corresponds to (5), $y_{\{h,i\},\ell}$ corresponds to (6), and z corresponds to (7). Our task is now to find a dual solution of value $\frac{1}{2}\binom{s}{d}d(d+t+u-1)OPT$. Note that if the objective function of the dual is set to this value, then z is completely determined. Using the value this gives for z, we assume that the $w_{h,\{\ell,m\}}$'s, $x_{j,\{\ell,m\}}$'s, and $y_{\{h,i\},\ell}$'s are the same for all h, i, j, ℓ, and m and solve the system of equations given when inequalities (8), (10), and (11) are set to equality. Doing so yields the following dual solution.

$$w'_{h,\{\ell,m\}} = \frac{s-d-t+1}{(u+s)(s-1)}\binom{s-1}{d-1} \qquad h \in [u], \{\ell,m\} \in \binom{[s]}{2}$$

$$x'_{j,\{\ell,m\}} = \frac{1}{s-1}\binom{s-1}{d-1} \qquad j \in [t], \{\ell,m\} \in \binom{[s]}{2}$$

$$y'_{\{h,i\},\ell} = \frac{d+t+u-1}{(u+s)(u+s-1)}\binom{s-1}{d-1} \qquad \{h,i\} \in \binom{[u]}{2}, \ell \in [s]$$

$$z' = \frac{d(d+t+u-1)}{(u+s)(u+s-1)}\binom{s}{d}$$

Since we found this solution by solving inequalities (8), (10), and (11) for equality, it is clear that these constraints are satisfied by this solution. Thus we must only verify constraint (9). Some straightforward (but tedious) manipulation shows that if $d = 1$, then the left-hand side is equal to

$$-\frac{su(u+t)}{(u+s)(u+s-1)(s-1)}$$

which is no greater than 0 since $s \geq 2$ when $t \geq 1$. Similarly, if $d > 1$, then the left-hand side can be written as

$$\left(1 - \frac{su(d+t+u-1)}{(u+s)(u+s-1)(d-1)}\right)\binom{s-2}{d-2}$$

which is clearly no greater than $\binom{s-2}{d-2}$. Since we have a feasible dual solution of value $\frac{1}{2}\binom{s}{d}d(d+t+u-1)OPT$, we have shown that inequality (3) holds, thus proving the lemma. □

We have now proved the most general (and hardest) case of Lemma 3. It remains only to prove three boundary cases. The next case we consider is when $u = 1$ and $t \geq 1$. We need to consider it separately because here we do not have constraint (6) (or dual variables of the form $y_{\{h,i\},\ell}$). Because the proof is so similar to the proof of Lemma 6, we merely state the primal and dual factor-revealing linear program and give a feasible solution to the dual that provides a sufficient lower bound for the optimal value of the primal. The technique to generate and solve these linear programs is nearly the same as in Lemma 6.

Lemma 7. *Lemma 3 holds when $u = 1$ and $t \geq 1$. Specifically, there exists a set of indices $L \in \binom{[s]}{d}$ such that*

$$\sum_{\ell \in L}\left(w(a_1, c_\ell) + \sum_{j \in [t]} w(b_j, c_\ell)\right) + \sum_{\{\ell,m\} \in \binom{L}{2}} w(c_\ell, c_m) \geq \frac{1}{2}d(d+t)OPT \ .$$

Proof. As before, it is sufficient to prove that

$$\binom{s-1}{d-1}\sum_{\ell \in [s]}\left(w(a_1, c_\ell) + \sum_{j \in [t]} w(b_j, c_\ell)\right) + \binom{s-2}{d-2}\sum_{\{\ell,m\} \in \binom{[s]}{2}} w(c_\ell, c_m) \geq \frac{1}{2}\binom{s}{d}d(d+t)OPT \ .$$

Again, we prove this by lower-bounding the optimal solution to the following linear program, in which the constraints are derived by the triangle inequality and the optimality of S.

minimize

$$\binom{s-1}{d-1}\sum_{\ell \in [s]}\left(w(a_1, c_\ell) + \sum_{j \in [t]} w(b_j, c_\ell)\right) + \binom{s-2}{d-2}\sum_{\{\ell,m\} \in \binom{[s]}{2}} w(c_\ell, c_m)$$

subject to

$$w(a_1, c_\ell) + w(a_1, c_m) - w(c_\ell, c_m) \geq 0 \qquad \{\ell, m\} \in \binom{[s]}{2}$$

$$w(b_j, c_\ell) + w(b_j, c_m) - w(c_\ell, c_m) \geq 0 \qquad j \in [t], \{\ell, m\} \in \binom{[s]}{2}$$

$$\sum_{\{\ell,m\} \in \binom{[s]}{2}} w(c_\ell, c_m) + \sum_{\ell \in [s]} w(a_1, c_\ell) \geq \binom{s+1}{2}OPT$$

The dual of this linear program is

maximize

$$\binom{s+1}{2} OPT \cdot z$$

subject to

$$\begin{aligned}
-x_{\{\ell,m\}} - \sum_{j\in[t]} y_{j,\{\ell,m\}} + z &\le \binom{s-2}{d-2} && \{\ell,m\} \in \binom{[s]}{2} \\
\sum_{m\in[s]-\{\ell\}} x_{\{\ell,m\}} + z &\le \binom{s-1}{d-1} && \ell \in [s] \\
\sum_{m\in[s]-\{\ell\}} y_{j,\{\ell,m\}} &\le \binom{s-1}{d-1} && j \in [t], \ell \in [s] \quad .
\end{aligned}$$

We conclude the proof of this lemma by providing the following dual solution, which can easily be shown to be feasible and to have the value $\frac{1}{2}\binom{s}{d} d(d+t)OPT$.

$$\begin{aligned}
x'_{\{\ell,m\}} &= \frac{s-d-t+1}{(s+1)(s-1)} \binom{s-1}{d-1} && \{\ell,m\} \in \binom{[s]}{2} \\
y'_{j,\{\ell,m\}} &= \frac{1}{s-1} \binom{s-1}{d-1} && j \in [t], \{\ell,m\} \in \binom{[s]}{2} \\
z' &= \frac{d(d+t)}{s(s+1)} \binom{s}{d}
\end{aligned}$$

□

The next case we consider is when $u \ge 2$ and $t = 0$. Although the factor-revealing LP is slightly different, the proof for this case is very similar to the proofs of Lemmas 6 and 7.

Lemma 8. *Lemma 3 holds when $u \ge 2$ and $t = 0$. Specifically, there exists a set of indices $L \in \binom{[s]}{d}$ such that*

$$\sum_{\ell\in L} \sum_{h\in[u]} w(a_h, c_\ell) + \sum_{\{\ell,m\}\in\binom{L}{2}} w(c_\ell, c_m) \ge \frac{1}{2} d(d+u-1)OPT \quad .$$

Proof. Again, it is sufficient to prove that

$$\binom{s-1}{d-1} \sum_{\ell\in[s]} \sum_{h\in[u]} w(a_h, c_\ell) + \binom{s-2}{d-2} \sum_{\{\ell,m\}\in\binom{[s]}{2}} w(c_\ell, c_m) \ge \frac{1}{2} \binom{s}{d} d(d+u-1)OPT \quad ,$$

which we do by finding a lower bound to the optimal solution of the following linear program:

minimize

$$\binom{s-1}{d-1} \sum_{\ell\in[s]} \sum_{h\in[u]} w(a_h, c_\ell) + \binom{s-2}{d-2} \sum_{\{\ell,m\}\in\binom{[s]}{2}} w(c_\ell, c_m)$$

subject to

$$w(a_h, c_\ell) + w(a_h, c_m) - w(c_\ell, c_m) \geq 0 \qquad h \in [u], \{\ell, m\} \in \binom{[s]}{2}$$

$$w(a_h, c_\ell) + w(a_i, c_\ell) - w(a_h, a_i) \geq 0 \qquad \{h, i\} \in \binom{[u]}{2}, \ell \in [s]$$

$$\sum_{\{h,i\} \in \binom{[u]}{2}} w(a_h, a_i) + \sum_{\{\ell,m\} \in \binom{[s]}{2}} w(c_\ell, c_m) + \sum_{h \in [u]} \sum_{\ell \in [s]} w(a_h, c_\ell) \geq \binom{s+u}{2} OPT \ .$$

The dual of this linear program is

maximize

$$\binom{s+u}{2} OPT \cdot z$$

subject to

$$-\sum_{\ell \in [s]} y_{\{h,i\},\ell} + z \leq 0 \qquad \{h, i\} \in \binom{[u]}{2}$$

$$-\sum_{h \in [u]} x_{h,\{\ell,m\}} + z \leq \binom{s-2}{d-2} \qquad \{\ell, m\} \in \binom{[s]}{2}$$

$$\sum_{m \in [s]-\{\ell\}} x_{h,\{\ell,m\}} + \sum_{i \in [u]-\{h\}} y_{\{h,i\},\ell} + z \leq \binom{s-1}{d-1} \qquad h \in [u], \ell \in [s] \ .$$

The following dual solution is feasible and has value $\frac{1}{2}\binom{s}{d} d(d+u-1)OPT$, thus concluding the proof of the lemma:

$$x'_{h,\{\ell,m\}} = \frac{s-d+1}{(u+s)(s-1)} \binom{s-1}{d-1} \qquad h \in [u], \{\ell, m\} \in \binom{[s]}{2}$$

$$y'_{\{h,i\},\ell} = \frac{d+u-1}{(u+s)(u+s-1)} \binom{s-1}{d-1} \qquad \{h, i\} \in \binom{[u]}{2}, \ell \in [s]$$

$$z' = \frac{d(d+u-1)}{(u+s)(u+s-1)} \binom{s}{d} \ .$$

□

The final case yet to be covered is when $u = 1$ and $t = 0$. For these conditions to hold, we must have $d = 1$, since $u + t \geq d$. The proof of this case follows from a simple contradiction argument.

Lemma 9. *Lemma 3 holds when $u = 1$ and $t = 0$ (and hence $d = 1$). Specifically, there exists an $\ell \in [s]$ such that $w(a_1, c_\ell) \geq \frac{1}{2} OPT$.*

Proof. Suppose by way of contradiction that no such ℓ exists. Then by the triangle inequality, $w(c_\ell, c_m) < OPT$ for all $\{\ell, m\} \in \binom{[s]}{2}$. But this contradicts the optimality of S, since it implies that every edge in S has weight strictly less than OPT. Thus we conclude that the set $S - T$ does indeed contain a vertex c_ℓ satisfying the statement of the lemma. □

Lemmas 5, 6, 7, 8, and 9 together imply Lemma 3, which in turn implies Theorem 4, stating that d-Greedy Augment has approximation ratio $(2k-2)/(k+d-2)$.

5 Lower Bound

The following theorem states that the approximation ratio $(2k-2)/(k+d-2)$ for d-Greedy Augment is tight.

Theorem 10. *There exist infinitely many* remote-clique *problem instances in which the ratio of the average edge weight in an optimal solution to the average edge weight in the solution returned by* d-Greedy Augment *is* $(2k-2)/(k+d-2)$.

Proof. Consider the following problem instance $(G=(V,E),k)$, in which $|V|=2k$ and V is partitioned into k/d subsets of d vertices called $V_1,\ldots,V_{k/d}$ and one group of k vertices called O. The edge weights are determined as follows:

$$w(v_1,v_2)=\begin{cases}2 & \text{if } v_1,v_2\in V_i \text{ for some } i \text{ or } v_1,v_2\in O\\ 1 & \text{otherwise}\end{cases} .$$

This construction is illustrated in Figure 2. It is clear that the edges in this problem instance satisfy the triangle inequality, since every edge has weight either 1 or 2. Also, it is clear that d-Greedy Augment could (if ties were broken badly) return the solution $T=\bigcup_i V_i$, whereas the optimal solution is O. If this happens, then the total edge weight in T is

$$\frac{k}{d}\binom{d}{2}\cdot 2+\left(\binom{k}{2}-\frac{k}{d}\binom{d}{2}\right)\cdot 1=\frac{1}{2}k(k+d-2) ,$$

and since the total edge weight in O is $2\binom{k}{2}$, the ratio of the performance of an optimal algorithm to d-Greedy Augment is $(2k-2)/(k+d-2)$. To see that there are an infinite number of such graphs, note that without affecting the relative performance of d-Greedy Augment, we can add arbitrarily many vertices to this construction in which every edge incident to these new vertices has weight 1. □

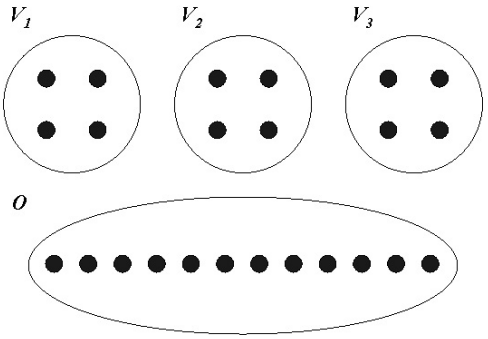
Note that this construction provides some intuition for the importance of the triangle inequality in the *remote-clique* problem. In particular, although it would clearly hurt the performance of the algorithm to make the weight of the heavy edges in this construction greater than 2, doing so is impossible without violating the triangle inequality.

6 Discussion

In this paper, we have used the technique of factor-revealing linear programs to prove that d-Greedy Augment achieves an approximation ratio of $(2k-2)/(k+d-2)$. This improves the best-known approximation ratio of Greedy Augment from 4 to 2 (asymptotically). Since we have shown that there are infinitely many problem instances in which d-Greedy Augment returns a solution no better than $(k+d-2)/(2k-2)\cdot OPT$, we have completely characterized the worst-case performance of d-Greedy Augment.

Currently, there are no hardness of approximation results for *remote-clique*. Therefore, a clear direction for future research is to either improve the approximation factor of 2 or provide a hardness of approximation result.

Figure 2: An example (for $k = 12$ and $d = 4$) showing that the approximation ratio of $(2k-2)/(k+d-2)$ is a tight bound on the worst-case performance of d-Greedy Augment. Edges that are contained within the same circle have weight 2, and all other edges have weight 1.



References

[1] C. Baur and S. P. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30(3):451–470, 2001.

[2] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI '99: Proceedings of the third symposium on Operating systems design and implementation*, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.

[3] Barun Chandra and Magnús M. Halldórsson. Approximation algorithms for dispersion problems. *J. Algorithms*, 38(2):438–465, 2001.

[4] A. Czygrinow. Maximum dispersion problem in dense graphs. *Operations Research Letters*, 27(5):223–227, 2000.

[5] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[6] Sándor P. Fekete and Henk Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38(3):501–511, 2003.

[7] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998.

[8] Magnús M. Halldórsson, Kazuo Iwano, Naoki Katoh, and Takeshi Tokuyama. Finding subsets maximizing minimum structures. *SIAM Journal on Discrete Mathematics*, 12(3):342–359, 1999.

[9] Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

[10] Refael Hassin, Shlomi Rubinstein, and Arie Tamir. Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21(3):133–137, 1997.

[11] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.

[12] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 731–740, New York, NY, USA, 2002. ACM Press.

[13] M.J. Kuby. Programming models for facility dispersion: the p-dispersion and maxisum dispersion problems. *Geographical Analysis*, 19(4):315–329, 1987.

[14] Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. A greedy facility location algorithm analyzed using dual fitting. In Michel X. Goemans, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *RANDOM-APPROX*, volume 2129 of *Lecture Notes in Computer Science*, pages 127–137. Springer, 2001.

[15] Robert J. McEliese, Eugene R. Rodemich, Howard Rumsey Jr., and Lloyd R. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Transactions on Information Theory*, 23(2):157–166, 1977.

[16] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized on-line matching. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, Washington, DC, USA, 2005. IEEE Computer Society.

[17] S. S. Ravi, D. J. Rosencrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

[18] Daniel J. Rosenkrantz, Giri K. Tayi, and S. S. Ravi. Facility dispersion problems under capacity and cost constraints. *Journal of Combinatorial Optimization*, 4(1):7–33, 2000.

[19] Daniel J. Rosenkrantz, Giri K. Tayi, and S. S. Ravi. Obtaining online approximation algorithms for facility dispersion from offline algorithms. *Networks*, 47(4):206–217, 2006.

[20] Arie Tamir. Obnoxious facility location on graphs. *SIAM Journal on Discrete Mathematics*, 4(4):550–567, 1991.