How to Allocate Network Centers

Judit Bar-Ilan [*] Guy Kortzars [†] David Peleg [‡]

November 23, 1992

Abstract

This paper deals with the issue of allocating and utilizing centers in a distributed network, in its various forms. The paper discusses the significant parameters of center allocation, defines the resulting optimization problems, and proposes several approximation algorithms for selecting centers and for distributing the users among them. We concentrate mainly on *balanced* versions of the problem, i.e., in which it is required that the assignment of clients to centers be as balanced as possible. The main results are constant ratio approximation algorithms for the balanced κ-centers and balanced κ-weighted centers problems, and logarithmic ratio approximation algorithms for the ρ-dominating set and the k-tolerant set problems.

[*]School of Library and Information, The Hebrew University, Jerusalem 9xxxx, Israel. This work was carried out while the author was with the Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science.

[†]Department of Applied Mathematics, The Weizmann Institute, Rehovot, 76100, Israel.

[‡]Department of Applied Mathematics, The Weizmann Institute, Rehovot 76100, Israel. Supported in part by an Allon Fellowship, by a Walter and Elise Haas Career Development Award and by a Bantrell Fellowship.

1 Introduction

The problem of allocating and utilizing centers in a communication network is a major issue in distributed network design. Among the various applications requiring the use of centers are distributed databases, routing, distributed data structures, etc. ([HR88, ML77, MK83, BG87, Pel90]). Using a collection of centers offers a convenient intermediate approach between the fully centralized and the fully distributed solutions, and provides a reasonable balance between the need for fault-tolerance and economical considerations. Unfortunately, all but the simplest center allocation problems are NP-hard, and therefore are considered unlikely to be tractable. In this paper, we discuss the significant parameters of center allocation, define the resulting optimization problems, and propose several approximation algorithms for selecting centers and for distributing the users among them.

In all variations of the center allocation problem considered in this paper, our goal is composed of two parts. The first is to select a collection of centers $C = \{\theta_1, \ldots, \theta_\kappa\}$, where $C \subset V$ and V is the set of the nodes of the network. The second is to assign each of the remaining nodes in the network to one of the centers. We denote by $\varphi(v)$ the center that is assigned to the node v, and we say that $\varphi(v)$ serves v. The pair (C, φ) determining the center assignment is referred to as the assignment pair.

Let $S_\varphi(\theta)$ denote the set of vertices assigned to the center θ, i.e.,

$$S_\varphi(\theta) = \{v \mid \varphi(v) = \theta\}$$

A center assignment can thus be characterized by the collection of pairs

$$\mathcal{S} = \{(\theta_1, S_\varphi(\theta_1)), \ldots, (\theta_\kappa, S_\varphi(\theta_\kappa))\}$$

such that $\theta_i \in C$, $S_\varphi(\theta_i) \subseteq V$ for every $1 \le i \le \kappa$ and $\bigcup_i S_\varphi(\theta_i) = V$. We refer to the collection $\mathcal{S}$ as an *assignment tuple*. We sometimes refer to $\mathcal{S}$ as a partition, when we wish to ignore the centers and consider only the collection of subsets $S_\varphi(\theta_i)$. We shall use the two representations, (C, φ) and $\mathcal{S}$, interchangeably.

As accessibility is a major concern in distributed systems, the distance between clients and their respective centers is an important design parameter in center allocation problems. Consequently, given a collection of centers, one obvious assignment choice would be to select, for every client v, the center nearest to it. A potential problem with this choice is that it may conflict with another significant concern, namely, workload balancing. The *nearest center* assignment might result in all (or most) sites using a single center (or a small subset of centers). This is undesirable, since it overloads the chosen centers and may create bottleneck problems. A possible way to overcome this situation is to bound the maximum number of sites assigned to any particular center θ by $|S_\varphi(\theta)| \le L$, for some bound L. (Naturally L has to be large enough, i.e., $L \ge n/\kappa$). This paper concentrates on center problems that insist on this maximal load requirement, referred to as *balanced* center problems.

It turns out that the two parts of the center allocation problems are not of equal difficulty. Once the centers are selected, it is possible to assign the clients optimally using standard flow techniques, as indicated in Section 2. We should therefore focus mainly on the harder part of the problem, involving the selection of the centers. Our aim is to select the "best" set of centers and the best assignment of nodes to the centers in a given network. There are two natural ways to approach this problem:

1. Fixing the number of centers, κ, and trying to minimize $\max\{(dist(v, \varphi(v))\}$.
2. Fixing a bound, ρ, on the maximal distance between a node and its center, and minimizing the number of centers.

These problems are sometimes referred to as the *κ-centers* and the *ρ-dominating set* problems respectively. In both cases we consider the balanced versions of these problems, i.e., solutions in which no center is assigned more than L clients. Both these problems are NP-hard (for unbounded κ) (see [GJ79]), even without the load constraint. Therefore, we direct our efforts toward attempting to approximate the optimum solution.

In the unbalanced case, i.e., with no constraint on the load, the two dual forms of the problem were given approximation algorithms before. The κ-centers problem was treated in [HS86], and given a polynomial time algorithm with approximation ratio 2. The ρ-dominating set problem can be formulated as a special case of the set cover problem of [Lov75], for which the greedy algorithm described therein provides an approximation ratio of $\log|V| + 1$.

In Section 5 we present an approximation algorithm for the balanced ρ- dominating set problem with approximation ratio $\lceil \ln|V| \rceil$.

The κ-centers problem becomes harder with the introduction of the balancing requirement, and it is necessary to develop stronger techniques than the ones used in [HS86] in order to overcome the difficulties resulting from the need to take the load constraint into account. In Section 3, we present our first main result, which is an approximation algorithm $Select\_Centers(G, L, \kappa)$ for the balanced κ-center problem that achieves a constant approximation ratio. Let us now outline the strategy on which our solution is based. Our starting point is the elegant approximation technique of Hochbaum and Shmoys [HS86] for the unbalanced κ center problem. We start by choosing an initial set of centers using the algorithm of [HS86]. After the initial centers are chosen, we assign the clients to these centers in two phases using flow techniques. This initial assignment, φ_I, does not necessarily obey the load constraints. Now the centers are partitioned into two sets, namely, the "light" centers (those that have fewer than L clients), and the "heavy" ones (those that have more). For the light centers, this assignment is final; we prove that the specific choice of the initial assignment φ_I guarantees that this does not harm the solution. For the heavy centers, however, some rebalancing is necessary. Each connected component of the heavy centers with their clients is treated separately. In each component, we construct a spanning tree, and apply a "tree-contraction" algorithm whose task is to balance the loads on the centers. This algorithm processes the spanning tree from the leaves up, and moves clients along the edges of the tree, spreading them among the selected centers and other nodes in their neighborhood, as necessary.

In Section 4 we consider the weighted variant of the problem, in which each node v has a nonnegative weight $\omega(v)$, and the feasibility constraint is that the selected collection of centers satisfies $\sum_{v \in C} \omega(v) \leq \kappa$. Again, an approximation algorithm for the unbalanced version of this problem (with ratio 3) is given in [HS86], however, we see no immediate way of modifying it for the balanced problem. The algorithm $Weighted\_Centers(G, L, \kappa)$ proposed for this problem in the current paper is based on a technique for converting solutions with approximation ratio α for the unweighted problem into solutions with approximation ratio $2\alpha + 1$ for the weighted problem. (The technique applies only for a specific type of solutions for the unweighted problem, referred to as *minimum cardinality* solutions, but fortunately the solutions generated by our unweighted algorithm $Select\_Centers(G, L, \kappa)$ fall into this category.)

Finally, we also consider the issue of fault tolerance. One common strategy for handling this problem is based on assigning multiple centers for each client. This approach handles the problem of center crashes, but does not attempt to handle the problem of communication faults. We would like to exploit redundancy in order to enhance data *availability* in the face of communication failures, including possible network partitions. Towards this goal, we propose the concept of *k-tolerant* sets: Let $A \subseteq V$ be the set of potential servers, and $B \subseteq V$ the set of potential customers. A *k-tolerant A-set for B* (or simply a k-tolerant set) is a subset $C \subseteq A$, such that for every $v \in B$, either $v \in C$ (this is possible only if $v \in A \cap B$) or there are k vertex-disjoint paths from v to C (in particular, to k distinct vertices in C). A solution to the *k-tolerant set* problem is such a set $C \subseteq A$ of minimal size. Note that when $A = V$, such a set exists in every graph, regardless of its connectivity. For example, Figure 1 depicts a 3-tolerant center set in a 1-connected graph, where $A = B = V$. An approximation

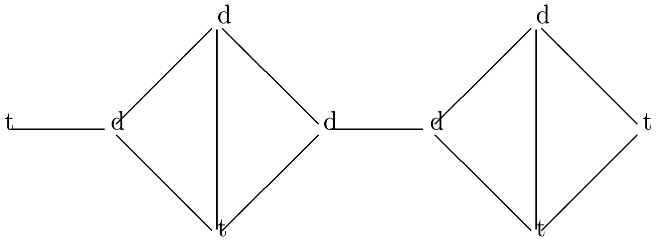

Figure 1: A 1-connected graph and a 3-tolerant center set for it (darkened vertices denote centers).

algorithm $Tolerant\_Centers(G, k)$ for the k-tolerant set problem is described in Section 6.

2 Preliminaries

2.1 The problem

The network is described by a connected undirected graph $G = (V, E)$, $|V| = n$, with a weight $c_{(u,v)}$ for every edge $(u, v) \in E$, representing the length of the edge. The vertices represent the sites of the network (or the processors located at these sites) and the edges represent bidirectional communication channels between these sites.

Let us define some concepts concerning graphs. For two vertices u, w in G, let $dist_G(u, w)$ denote the length of a shortest path in G between those vertices, where the length of a path is the sum of its edge weights. (In the above notation, we sometimes omit the reference to G where no confusion arises.) The *neighborhood* of a vertex $v \in V$ is defined as $\Gamma(v) = \{w \mid (w, v) \in E\}$. Let (C, φ) be a given center assignment, and let

$$\mathcal{S} = \{(\theta_1, S_\varphi(\theta_1)), \ldots, (\theta_\kappa, S_\varphi(\theta_\kappa))\}$$

be the induced assignment tuple. In all our assignments, $\theta \in \mathcal{S}_\varphi(\theta)$; that is a center always serves itself. For $A \subseteq C$, define

$$S_\varphi(A) = \bigcup_{\theta \in A} S_\varphi(\theta)$$

.

The following definitions formalize our measures for the quality of this assignment. We first define the appropriate *radius* measure. Let

$$\mathcal{R}(S_\varphi(\theta_i)) = \max_{v \in S_\varphi(\theta_i)} \{dist(v, \theta_i)\},$$

and let $\mathcal{R}(\mathcal{S}) = \max_i \mathcal{R}(S_\varphi(\theta_i))$. Next, the *load* of a center θ_i is denoted

$$\mathcal{L}(\theta_i) = |S_\varphi(\theta_i)|,$$

and the maximal load of the assignment tuple $\mathcal{S}$ is $\mathcal{L}(\mathcal{S}) = \max_i\{\mathcal{L}(\theta_i)\}$. We say that a pair (C, φ) is *ρ-dominating* if the induced assignment tuple $\mathcal{S}$ satisfies $\mathcal{R}(\mathcal{S}) \leq \rho$, and *$L$-balanced* if $\mathcal{S}$ satisfies $\mathcal{L}(\mathcal{S}) \leq L$.

Next let us define the basic notion of an approximation algorithm. This is a polynomial time algorithm for an optimization problem, with some performance guarantee on the quality of the produced solutions. The *approximation ratio* of an approximation algorithm for a minimization problem is the maximal ratio between the solution obtained by the algorithm and the optimal solution, where the maximum is taken over all input instances to the problem. A similar definition applies to maximization problems.

2.2 Assigning clients to fixed centers

This subsection describes how to handle situations in which the centers are already fixed, but we are given control over the assignment, φ, of centers to sites. As mentioned earlier, if one relies solely on distance considerations, then the obvious choice is the *nearest* assignment, $\mathcal{N}$. This assignment satisfies

$$\mathcal{N}(v) \in \{\theta_i \mid dist(\theta_i, v) \leq dist(\theta_j, v),\ \forall 1 \leq j \leq |C|\},$$

and is straightforward to compute. However, we are interested in balancing the work load as well, that is, we want to minimize the radius of the assignment tuple, subject to the constraint $\mathcal{L}(\mathcal{S}) \leq L$. Producing a balanced assignment is a variant of a partitioning problem described in [BG87], Sect. 5.4.3, as the *concentrator location* problem, and solved via linear programming or the flow methods of [Ber85, BG89]. In that problem the minimized function is $\sum_{v \in V} dist(v, \varphi(v))$. Our solution for assigning centers to clients is a slightly more involved variation of that solution.

The procedure $Assign(G, C, L)$, depicted in figure 2, solves the assignment problem. It finds the minimal radius for which an assignment is possible using standard flow techniques (cf. [Eve79]). We define the integral flow function $f_G : E \mapsto Z^+$, where $f_G(e)$ is the flow assigned to edge e. (We omit the definitions of flow functions and constraints they satisfy; these definitions can be found in, e.g. [Eve79]). The assignment is constructed based on the maximum flow in the appropriate flow-graph, G_b.

The flow graph is generated by Procedure $Flow\_graph(C, U, \rho, m_1, m_2, m_3)$, presented in Figure 3. This procedure is also used as a component in our later algorithms.

1. Let $\bar{d} = (d_1, ..., d_q)$ be the list of distances between the vertices of $V \setminus C$ and the centers in C, in increasing order.

2. Let the variable ρ run a binary search on $\bar{d}$:

   (a) Call $G_\rho \leftarrow Flow\_graph(C, V \setminus C, \rho, L-1, 1, 1)$.

   (b) Compute the maximum integral flow function f_{G_ρ} in G_ρ.

   (a) Let $b = \min\{d \mid \text{the maximum flow in } G_d \text{ is } |V \setminus C|\}$.

   (b) For every v in $V \setminus C$ set $\varphi(v)$ to be the center θ_i such that $f_{G_b}(\theta_i, v) > 0$.

Figure 2: Procedure $Assign(G, C, L)$

1. Construct a bipartite graph $G' = (C, U, E')$, with $E' = \{(\theta, v) \mid \theta \in C,\ v \in U,\ dist(\theta, v) \leq \rho\}$.

2. Add two new vertices s and t. Connect s to every node in C. Connect every node in U to t.

3. Define capacities γ by setting $\gamma(s, \theta) = m_1$, $\gamma(\theta, v) = m_2$ and $\gamma(v, t) = m_3$.

4. Output the resulting flow-graph.

Figure 3: Procedure $Flow\_graph(C, U, \rho, m_1, m_2, m_3)$

The relationship between flow in the flow graphs G_ρ constructed by Procedure $Assign(G, C, L)$ and the center assignment is established by the following lemma.

Lemma 2.1 There exists a feasible assignment of the clients to the centers in C with radius d, iff the maximum flow in G_d is $|V \setminus C|$.

Proof: Let φ be a balanced center assignment with radius d. No center is assigned more than $L - 1$ clients (plus itself). If $\varphi(v) = \theta$, then there is an edge in G_d between v and θ. On each such edge it is possible to push one unit of flow, resulting in a total flow of at least $|V \setminus C|$, while obeying the capacity constraint for the edges of type (s, θ).

Conversely, if there exists a flow of size $|V \setminus C|$, then an assignment with radius d can be constructed simply as described in procedure $Assign(G, C, L)$. ∎

Corollary 2.2 Procedure $Assign(G, C, L)$ returns an assignment with the minimal feasible radius. ∎

Since procedure $Assign(G, C, L)$ requires polynomial time in n, we have shown:

Proposition 2.3 Given a graph G, a collection of centers $C \subseteq V$, $\kappa = |C|$, and a bound L, such that $L \geq \frac{n}{\kappa}$, there is a polynomial time algorithm for computing an assignment $\varphi : V \setminus C \mapsto C$ with an induced assignment tuple $\mathcal{S}$ satisfying $\mathcal{L}(\mathcal{S}) \leq L$ and minimal radius $\mathcal{R}(\mathcal{S})$. ∎

Using similar techniques it is possible to assign several centers to each client. Such an approach may be useful for fault tolerance purposes, for instance.

Definition 2.4 The *balanced t-assignment* problem is defined as follows:
Input: Graph $G(V, E)$, a collection of centers $C \subseteq V$, integers $L, t \geq 1$.
Goal: A t-assignment $\varphi : V \setminus C \mapsto C^t$ assigning t centers to each vertex, and minimizing the radius $\mathcal{R}(\mathcal{S})$ of the induced assignment tuple $\mathcal{S}$, subject to the constraint $\mathcal{L}(\mathcal{S}) \leq L$.

The algorithm for solving this problem is a modification of the previous one. Procedure $Flow\_graph(C, V \setminus C, \rho, L-1, 1, t)$ is called with different values of the parameter ρ. It is easy to see that there exists a feasible assignment iff the maximum flow is $t \cdot |V \setminus C|$, therefore the minimal ρ for which the maximum flow equals $t \cdot |V \setminus C|$ is the minimum radius.

Proposition 2.5 Given a graph G, a collection of centers $C \subseteq V$, $\kappa = |C|$, and a bound L, such that $L \geq \frac{tn}{\kappa}$, there is a polynomial time algorithm for computing a t-assignment $\varphi : V \setminus C \mapsto C^t$ with an induced assignment tuple $\mathcal{S}$ satisfying $\mathcal{L}(\mathcal{S}) \leq L$ and minimal radius $\mathcal{R}(\mathcal{S})$. ∎

3 The balanced κ-center selection problem

In this section we turn to the selection problem, and present an approximate solution to the problem in which the number of centers κ, and a load constraint L, are given, and we want to optimize the radius. Let us first give a formal definition of the problem and introduce some basic notation.

Definition 3.1 The *L-balanced κ-centers* problem is defined as follows:
Input: A complete weighted graph $G = (V, E)$, edge weights $c_{(u,v)}$, integers L, $\kappa \geq 1$.
Goal: Select an L-balanced center assignment (C, φ), such that $|C| = \kappa$, minimizing the radius $\mathcal{R}(\mathcal{S})$ of the induced assignment tuple $\mathcal{S}$.

Let us denote the optimal radius by $\mathcal{R}_{opt}(G, L, \kappa)$.

We assume that the edge weights satisfy the triangle inequality. If the graph does not obey this assumption, we can modify it into such a complete graph by setting $c_{(u,v)} = dist_G(u, v)$

for each edge (u, v). Without loss of generality, we label the edges so that $c_{e_1} \leq c_{e_2} \leq \ldots \leq c_{e_{\frac{n(n-1)}{2}}}$, and denote $c_j = c_{e_j}$ for all $j \geq 1$.

Following [HS86], for an integer $\mu \geq 1$, we define the bottleneck graph, $G_\mu = (V, E_\mu)$ to be an edge subgraph of G, where $E_\mu = \{e_j \in E \mid j \leq \mu\}$. We also define the *t-closure* graph $(G')^t$, for any edge subgraph G' of G, as the unweighted graph in which two nodes are connected iff there is a path of at most t edges between them in the original graph G'.

3.1 The algorithm

We now give an approximation algorithm $Select\_Centers(G, L, \kappa)$ to the problem, achieving a constant approximation ratio. Let us start with an overview of the algorithm. The algorithm considers the bottleneck graphs, G_μ, in increasing order of μ. For each such graph, it chooses a maximal independent set, C, in G_μ^2. As explained in [HS86] (see Claim 3.2 below), this set C indicates whether there exists a feasible solution in G_μ. If there is no feasible solution in G_μ, the algorithm turns to the next bottleneck graph. Otherwise, it calls procedure *Allocate*, whose task is to select the centers. After this procedure is applied to all graphs G_μ, the solution with the minimal radius (among all the solutions produced by the algorithm) is taken as our final solution for the balanced κ-center problem.

Procedure *Allocate* attempts to assign up to $L-1$ clients from $V \setminus C$ to each center, using a flow computation. This process may leave some nodes of $V \setminus C$ unassigned. Next, each of these remaining nodes gets assigned, again using a flow computation, but this time ignoring the balance constraints. The goal of the first flow phase is to ensure that the assignment obtained by the two flow phases is "as balanced as possible", in the sense that if a center $\theta \in C$ has fewer than $L-1$ clients, while some nodes are left unassigned, there is no way to improve the situation by moving clients to θ, while still restricting ourselves to G_μ. We refer to the assignment obtained after both flow phases as the *initial* assignment, denoted φ_I.

Next, Procedure $Partition(C, \varphi_I)$ partitions the centers into two sets $\mathcal{E}$ and $\mathcal{F}$, according to the initial assignment φ_I. The set $\mathcal{E}$ essentially consists of the light (or *empty*) centers, i.e., those having fewer than $L-1$ clients (apart from themselves), and the set $\mathcal{F}$ contains the heavy (or *full*) centers, i.e., those having more than $L-1$ clients. Actually, the more delicate part of the procedure involves specifying the classification of those centers with precisely $L-1$ clients. Notice (see Figure 6) that $\mathcal{E}$ consists of the set $\mathcal{E}_0$ of centers having at most $L-2$ clients, plus all the centers that can potentially transfer clients to the ones in $\mathcal{E}_0$ (along augmenting paths). We show (see Claim 3.4) that the first flow phase guarantees that $\mathcal{E}$ contains no center, θ with *more* than $L-1$ clients, since otherwise we could transfer clients

If $|V| > L \cdot \kappa$ then return "failure".

For $\mu = 1$ **to** $|E|$ **do**

1. Choose a maximal independent set in G_μ^2. Let this set be $C = \{\theta_1, \theta_2, \ldots, \theta_\ell\}$.
2. **If** $\ell > \kappa$ **then** go to **Next**.
3. Let $\bar{G}_1, \ldots, \bar{G}_m$ be the connected components of G_μ^2.
4. Let $\kappa_i = \lceil \frac{|\bar{V}_j|}{L} \rceil$, for $1 \le i \le m$.
5. **If** $\sum_i \kappa_i > \kappa$ **then** go to **Next**.
6. Let $C_i \subseteq C$ be the centers in connected component $\bar{G}_i$.
7. **For** $i = 1, \ldots, m$:

   Call $(C_\mu^i, \varphi_\mu^i) \leftarrow Allocate(\bar{G}_i, L, C_i)$.
8. Let φ_μ be the union of the partial assignments φ_μ^i, and $C_\mu = \bigcup_i C_\mu^i$.

Next:

End-for

Select the assignment (C_μ, φ_μ) with the minimal radius $\mathcal{R}(\mathcal{S})$, for $1 \le \mu \le |E|$.

Figure 4: Algorithm $Select\_Centers(G, L, \kappa)$

from θ to other centers in $\mathcal{E}$ and increase the flow in $G_{F_1}(C)$, the flow graph with the load constraint.

For the nodes being assigned to centers in $\mathcal{E}$, this assignment is final. For the remaining nodes the algorithm appoints a minimal number of additional centers (if $|C| < \kappa$, and the load constraint is not satisfied). This is done by creating an auxiliary "neighborhood graph" connecting the centers, constructing a spanning forest of the centers of $\mathcal{F}$ in this graph, and applying a tree-contraction procedure, $Tree\_Contract$, to each tree.

The main algorithm, $Select\_Centers(G, L, \kappa)$, is presented in Figure 4. This algorithm calls Procedure *Allocate*, presented in Figure 5. Procedure *Allocate*, in turn calls three other procedures: Procedure $Flow\_graph$ (which was described in Figure 3), Procedure $Partition(C, \varphi_I)$ (given in Figure 6), and Procedure $Tree\_Contract(\tilde{G})$ (given in Figure 7).

1. Call $G_{F_1} \leftarrow Flow\_graph(C, V \setminus C, 2c_\mu, L-1, 1, 1)$.
2. Compute the maximum integral flow f_{F_1} in the graph G_{F_1}.
3. Define the partial assignment, φ_1 as follows: θ_j serves itself and those nodes $v \in V \setminus C$ that in the flow f_{F_1} receive flow from θ_j.
4. Let V' be the set of nodes assigned to centers by φ_1.
5. Call $G_{F_2} \leftarrow Flow\_graph(C, V \setminus (V' \cup C), 2c_\mu, \infty, 1, 1)$.
6. Compute the maximum f_{F_2} flow in the graph G_{F_2}.
7. Assign each node a center according to the flow f_{F_2}. Denote this assignment by φ_2, and let $\varphi_I = \varphi_1 \cup \varphi_2$.
8. Call $(\mathcal{E}, \mathcal{F}) \leftarrow Partition(C, \varphi_I)$.
9. For every center $\theta \in C$, let $Bin(\theta) = \mathcal{S}_{\varphi_I}(\theta)$.
10. Define $\tilde{G} = (\mathcal{F}, \tilde{E})$, where $\tilde{E} = \{(\theta, \theta') \mid \theta, \theta' \in \mathcal{F},\ \exists u, v \in V \text{ s.t. } v \in Bin(\theta) \text{ and } u \in Bin(\theta') \text{ and there is an edge between } u \text{ and } v \text{ in } G_\mu^2\}$.
11. Let $\tilde{G}_1, \ldots \tilde{G}_l$ be the connected components of $\tilde{G}$.
12. Call $(\tilde{C}^j, \varphi_F^j) \leftarrow Tree\_Contract(\tilde{G}_j)$ for every connected component.
13. Let $\varphi(v) = \varphi_I(v)$ if $\varphi_I(v) \in \mathcal{E}$, and let $\varphi(v) = \varphi_F^j(v)$ if $\varphi_I(v) \in \mathcal{F} \cap \tilde{G}_j$.
14. Let $C_F = \mathcal{E} \cup \bigcup_j \tilde{C}^j$.
15. Output (C_F, φ).

Figure 5: Procedure $Allocate(G, L, C)$

1. Let $\mathcal{E}_0$ be the set of centers having at most $L-2$ nodes assigned to them by φ_I.

2. Set

$$\begin{aligned}\mathcal{E}_{j+1} &= \mathcal{E}_j \cup \{\theta \in C \mid \exists v \in V \setminus C,\ \exists \theta' \in \mathcal{E}_j,\\ &\qquad dist(\theta, v) \leq 2 \cdot c_\mu,\ dist(\theta', v) \leq 2 \cdot c_\mu,\ \varphi_I(v) = \theta\}.\end{aligned}$$

3. Let $\mathcal{E}$ be the largest set $\mathcal{E}_j$ obtained in this process, and let $\mathcal{F} = C \setminus \mathcal{E}$.

4. Return $(\mathcal{E}, \mathcal{F})$.

Figure 6: Procedure $Partition(C, \varphi_I)$

3.2 Correctness and analysis

In step 2 of the algorithm $Select\_Centers(G, L, \kappa)$, we skip the current μ, when the size of the maximal independent set is greater than κ. This action is justified by the following claim.

Lemma 3.2 [**HS86**] If the maximal independent set, C, selected in step 1 of algorithm $Select\_Centers(G, L, \kappa)$ has size $|C| > \kappa$, then there is no solution to the κ-center problem (even without the balance constraint) with radius $\leq c_\mu$.

Proof: Assume there is a solution (C', φ') to the unbalanced κ center problem with radius $\leq c_\mu$. This means that the graph G_μ can be covered with κ stars, where the stars are formed around the centers of C' according to the assignment φ' in this solution. These κ stars turn into κ cliques is G_μ^2. At most one node from each clique can appear in any maximal independent set. Therefore the size of any maximal independent set in G_μ^2 is at most κ, but C is an independent set in G_μ^2 with size greater than κ ; a contradiction. ∎

Having an independent set of size κ is sufficient for producing a feasible solution for the κ-center problem with no balance constraint. In the balanced problem, some additional constraints must be satisfied. These constraints appear in lines 3-5 in Algorithm $Select\_Centers(G, L, \kappa)$. Their necessity is justified by the next lemma:

Lemma 3.3 Let $\bar{G}_1, \ldots, \bar{G}_m$ be the connected components of G_μ^2, where $\bar{G}_j = (\bar{V}_j, \bar{E}_j)$, and let $\kappa_j = \lceil \frac{|\bar{V}_j|}{L} \rceil$. If $\sum \kappa_j > \kappa$, then there is no feasible solution for the balanced problem with radius $\leq c_\mu$.

Proof: Assume that there exists a feasible solution (C', φ') for the balanced κ-center problem with radius $b \leq c_\mu$. Then the graph G_μ can be covered with κ stars, where each star has

1. Let $\tilde{C} = \{\theta_1, \ldots, \theta_m\}$ be the nodes of $\tilde{G}$.

2. For every $1 \le j \le m$, let $Carry\_in(\theta_j) \leftarrow \emptyset$

3. Construct a spanning tree in $\tilde{G}$.

4. **Repeat**

   (a) Pick an arbitrary leaf θ of the tree.
   Let $|Bin(\theta)| = iL + \epsilon$, where $0 \le \epsilon < L$.

   (b) Pick $(i-1)$ additional centers from the nodes in $Bin(\theta)$,
   and add them to $\tilde{C}$.

   (c) Distribute $i(L-1)$ nodes among the i centers, assigning L nodes to each center (including itself), and assigning first the nodes in $Carry\_in(\theta)$.

   (d) Let $Carry\_out(\theta)$ be the set of ϵ unassigned nodes of $Bin(\theta)$.

   (e) Let θ' be the parent of θ in the tree.

   (f) Set $Carry\_in(\theta') \leftarrow Carry\_in(\theta') \cup Carry\_out(\theta)$,
   and set $Bin(\theta') \leftarrow Bin(\theta') \cup Carry\_out(\theta)$.

   (g) Remove θ from the tree.

5. **until** the tree consists of a single node.

6. For the last remaining node θ, execute steps (a) to (d).
   If $|Carry\_out(\theta)| > 0$, pick an additional center from this set and assign all the other members of $Carry\_out(\theta)$ to it.

7. Return the resulting assignment $(\tilde{C}, \tilde{\varphi})$ for the nodes of $Bin(\theta)$, $\theta \in \tilde{G}$.

Figure 7: Procedure $Tree\_Contract(\tilde{G})$

at most L nodes. Notice that G_μ and G_μ^2 induce the same partition of V into connected components, namely $\bar{V}_1, \ldots, \bar{V}_m$. A center can serve only nodes in its connected component. Therefore, the connected component $\bar{V}_j$ must have at least κ_j centers. Since (C', φ') is a feasible solution, $\sum_{j=1}^{m} \kappa_j \leq \kappa$. ∎

Let b be the radius of the optimal solution, let $X_{opt} = \{x_1, \ldots x_\kappa\}$ be the set of centers chosen in an optimal solution, and let φ_{opt} be an optimal assignment associated with these centers. There exists an integer μ_b, such that $b = c_{\mu_b}$, since G is a complete graph, and the edge weights obey the triangle inequality. Let us analyze the solutions produced by our algorithm for $\mu = \mu_b$.

We first show that for G_μ, $\mu = \mu_b$, the initial assignment can be modified into feasible assignment, even if the assignment of all the nodes in $S_{\varphi_I}(\mathcal{E})$ remains unchanged. The definition of the set $\mathcal{E}$ appears in Figure 6.

Claim 3.4 Each center in $\mathcal{E}$ has at most $L-1$ clients.

Proof: Assume that there exists a center $\theta \in \mathcal{E}$ with more than $L-1$ clients assigned to it by φ_I. Then there exists a $j \neq 0$, such that θ belongs to $\mathcal{E}_j$ but does not belong to any $\mathcal{E}_i$ for $i < j$. If this is the case then there exists a $v \in V \setminus C$ and $\theta' \in \mathcal{E}_{j-1}$ such that v can be transferred to θ'. By asequence of such transfers $\mathcal{E}_0$ will be reached. Each center in $\mathcal{E}_0$ has less than $L-1$ clients, therefore this new client can be added without disobeying the load constraint. The flow corresponding to this assignment is a legal flow in G_{F_1} and is strictly greater than the maximum flow in this graph, a contradiction. ∎

Lemma 3.5 The partial solution $(\mathcal{E}, \varphi_I\mid_{S_{\varphi_I}(\mathcal{E})})$ can be extended to a total feasible solution.

Proof: Let $\mathcal{E} = \{\theta_1, \ldots, \theta_k\}$, and let

$$X_{\mathcal{E}} = \{x \in X_{opt} \mid \varphi_{opt}(\theta) = x,\ \theta \in \mathcal{E}\}.$$

The set $X_{\mathcal{E}}$ is the set of "optimal centers" that in the optimal assignment serve the centers in $\mathcal{E}$. We proceed by proving the following claims.

Claim 3.6 $\varphi_{opt}(\theta) \neq \varphi_{opt}(\theta')$ for every $\theta, \theta' \in \mathcal{E}$.

Proof: In G_μ, each $x \in X_{\mathcal{E}}$ is the center of a star with radius $\leq b$ whose leaves are the nodes of $S_{\varphi_{opt}}(x) \setminus x$. In G_μ^2, each such star turns into a clique. The set $\mathcal{E}$ is an independent set in G_μ^2, therefore it contains at most one node from each set $S_{\varphi_{opt}}(x)$. ∎

Corollary 3.7 $|X_{\mathcal{E}}| = |\mathcal{E}|$. ∎

Claim 3.8 $S_{\varphi_I}(\mathcal{F}) \cap S_{\varphi_{opt}}(X_{\mathcal{E}}) = \emptyset$.

Proof: The proof is by contradiction. Suppose there exists some $v \in S_{\varphi_I}(\mathcal{F}) \cap S_{\varphi_{opt}}(X_{\mathcal{E}})$, and let $\varphi_{opt}(v) = x$, $x \in X_{\mathcal{E}}$. By the definition of $X_{\mathcal{E}}$, there exists a center $\theta \in \mathcal{E}$, such that $\varphi_{opt}(\theta) = x$, therefore $dist(v, \theta) \leq dist(\theta, x) + dist(x, v) \leq 2c_\mu$. But $\varphi_I(v) \in \mathcal{F}$, therefore by the definition of $\mathcal{E}$, $dist(\theta, v) > 2c_\mu$ for every $\theta \in \mathcal{E}$; a contradiction. ∎

Therefore, $S_{\varphi_{opt}}(X_{\mathcal{E}}) \subseteq V \setminus S_{\varphi_I}(\mathcal{F}) = S_{\varphi_I}(\mathcal{E})$. Every node served by $X_{\mathcal{E}}$ in the optimal assignment is served by a center in $\mathcal{E}$ under the assignment φ_I. Therefore we can dedicate the centers in $\mathcal{E}$ to serve the set $S_{\varphi_I}(\mathcal{E})$, and the assignment $\varphi_{I|_{S_{\varphi_I}(\mathcal{E})}}$ can be extended to a total feasible assignment (e.g. $\varphi_{opt}\,|_{S_{\varphi_I}(\mathcal{F})}$). This completes the proof of Lemma 3.5. ∎

We took care of the nodes assigned to centers in $\mathcal{E}$. The centers in $\mathcal{F}$ might be overloaded, so we have to appoint new centers. Let $Bin(\theta) = \mathcal{S}_\varphi(\theta)$ and let $\tilde{G} = (\mathcal{F}, \tilde{E})$, where $\tilde{E} = \{(\theta, \theta') \mid \theta, \theta' \in \mathcal{F},\ \exists u, v \in V \text{ s.t. } v \in Bin(\theta) \text{ and } u \in Bin(\theta') \text{ and there is an edge between } u \text{ and } v \text{ in } G_\mu^2\}$. Next we show, that a feasible solution can be produced, even if we consider each connected component $\tilde{G}_j$ of $\tilde{G}$ separately. That is, letting

$$Bin(\tilde{G}_j) = \bigcup_{\theta \in \tilde{G}_j} Bin(\theta),$$

there exists a feasible solution where the nodes of $Bin(\tilde{G}_j)$ are served only by nodes from $Bin(\tilde{G}_j)$. This claim justifies line 12 of Procedure *Allocate*.

Claim 3.9 Let x be an optimal center, such that $x \in Bin(\tilde{G}_j)$. Then in the optimal assignment, x does not serve nodes in $Bin(\tilde{G}_i)$, $i \neq j$.

Proof: Since $x \in Bin(\tilde{G}_j)$, its distance from every $u \in Bin(\tilde{G}_i)$, $i \neq j$, is at least $b+1$, but the radius of the optimal solution is only b. If there existed a $u \in Bin(\tilde{G}_i)$, $i \neq j$ such that $dist(u, x)$ in the graph G_μ is $\leq b$, then there would be an edge between u and x in the graph G_μ^2, and then $u \in Bin(\tilde{G}_j)$, a contradiction. TZIUR!!! ∎

Claim 3.10 Let $x \in X_{opt}$ be an optimal center, and let $v \in Bin(\tilde{G}_m)$, such that $\varphi_{opt}(v) = x$. Assume that $\varphi_I(x) \in \mathcal{E}$. Then there does not exist a v', such that $v' \in Bin(\tilde{G}_i)$, $i \neq m$, and $\varphi_{opt}(v') = x$.

Proof: Assume that there exists such a v'. The center x serves v in an optimal assignment, therefore $dist(x, v) \leq b$. Similarly $dist(x, v') \leq b$. But then, by the definition of G_μ^2, there exists an edge between v and v', therefore $\varphi_I(v)$ and $\varphi_I(v')$ must be in the same connected component of $\tilde{G}$; a contradiction. ∎

We have already set up a one-to-one correspondence between the centers in $X_{\mathcal{E}}$ and the centers in $\mathcal{E}$. No center in $X_{\mathcal{E}}$ belongs to $\cup Bin(\tilde{G}_i)$. The last two claims show that an $x \in X \setminus X_{\mathcal{E}}$ serves clients in at most one connected component $\tilde{G}_i$:

Claim 3.11 There exists a partition of $X_{opt} \setminus X_{\mathcal{E}}$ into l subsets $X^1, \ldots, X^l$, corresponding to $\tilde{G}_1, \ldots, \tilde{G}_l$ respectively, such that a center $x \in X^j$ does not serve nodes in $Bin(\tilde{G}_i)$ for $i \neq j$. ∎

Recall that each node of $\tilde{G}$ corresponds to a center and its clients that are assigned to it by φ_I. Two nodes in $\tilde{G}$ are connected if they are "near". We have already seen that we can consider each connected component of $\tilde{G}$ separately. In each component a spanning tree is constructed. Since some of the centers are overloaded, we have to appoint additional centers. We do this proceeding along the spanning tree from the leafs upwards. Assume that there are $iL + \epsilon$, $0 \leq \epsilon < L$ nodes in $Bin(\theta)$, that is $iL + \epsilon$ nodes are assigned to θ by φ_I. One center, θ, was already appointed by φ_I. We arbitrarily choose $i - 1$ new centers from $Bin(\theta)$ and assign each of them $L - 1$ clients. The number of nodes not taken care of at this stage of the algorithm is ϵ. Let θ' be the parent of θ in the spanning tree. The ϵ nodes from $Bin(\theta')$ are added to the set $Carry\_in(\theta')$ (this set is initially empty), and $Bin(\theta')$ is set to $Bin(\theta') \cup Carry\_in(\theta')$. The leaf node, θ, is pruned. Once all the children of θ' are pruned, we appoint additional centers in $Bin(\theta')$ in the same manner as for θ, while making sure that the nodes in $Carry\_in(\theta')$ get assigned to centers at this stage and are not forwarded to the parent of θ' (see Lemma 3.13). This process is continued until there are no more nodes in the tree.

During the execution of the tree contraction procedure, new centers are chosen. Let $\mathcal{H}$ be the set of the new centers, and let $\mathcal{T} = \mathcal{F} \cup \mathcal{H}$. The set $\mathcal{T}$ can also be partitioned into l subsets $T^1, \ldots, T^l$, corresponding to $\tilde{G}_1, \ldots, \tilde{G}_l$ respectively, such that a $\theta \in T^j$ serves only nodes in $Bin(\tilde{G}_j)$, since each connected component was treated separately.

Lemma 3.12 For every $1 \leq j \leq l$, $|T^j| \leq |X^j|$.

Proof: The tree contraction procedure, $Tree\_Contract$, assigns $|T^j| = \lceil \frac{|Bin(\tilde{G}_j)|}{L} \rceil$ centers to serve $Bin(\tilde{G}_j)$. The centers in X^j are dedicated to $Bin(\tilde{G}_j)$, but they might serve additional nodes, that in our solution are served by centers in $\mathcal{E}$. Therefore, the centers in X^j must serve at least $|Bin(\tilde{G}_j)|$ clients. These centers obey the balance constraint, thus $|X^j| \geq \lceil \frac{|Bin(\tilde{G}_j)|}{L} \rceil$. ∎

It remains to analyze the quality of the approximation. For the nodes in $S_{\varphi_I}(\mathcal{E})$, we assigned centers at distance $\leq 2b$. Now consider the nodes of $S_{\varphi_I}(\mathcal{F})$. The final assignment is always determined at the leaves of the tree. Let $Bin_{init}(\theta)$ be the initial value of $Bin(\theta)$, and let $Bin_{final}(\theta)$ be the value of $Bin(\theta)$ when it becomes a leaf of the current tree. Let $v \in Bin_{final}(\theta)$. Recall that

$$Bin_{final}(\theta) = Bin_{init}(\theta) \cup Carry\_in(\theta).$$

If $v \in Carry\_in(\theta)$, then v was moved to $Bin_{final}(\theta)$ from a descendant of θ in the original spanning tree. In the assignment process, we first assign centers to the nodes in the set $Carry\_in(\theta)$.

Lemma 3.13 All nodes in $Carry\_in(\theta)$ are assigned centers from $Bin_{final}(\theta)$.

Proof: Let

$$|Carry\_in(\theta)| = kL + \epsilon.$$

Then

$$|Bin_{final}(\theta)| \geq (k+1)L + \epsilon,$$

since $|Bin_{init}(\theta)| \geq L$. Therefore we choose at least $(k+1)$ centers from the nodes of $Bin_{final}(\theta)$. ∎

¿From this lemma it follows, that a node v is moved at most once during the tree contraction process. If $\varphi_I(v) = \theta$, then $\varphi_F(v)$ belongs either to $Bin_{final}(\theta)$ or to $Bin_{final}(\theta')$, where θ' is the parent of θ in the spanning tree. In $Bin_{final}(\theta')$ there are nodes from $Bin_{init}(\theta')$, and possibly also nodes v such that $\varphi_I(v)$ is θ'', where θ'' is a child of θ'.

Let $\varphi_F(v) = c$ and $\varphi_I(v) = \theta$. There are several possibilities:

1. $\varphi_F(v) \in Bin_{init}\theta$. In this case:

$$\begin{aligned} dist(v,c) &\leq dist(v,\theta) + dist(\theta,c) \\ &\leq 2b + 2b = 4b \end{aligned}$$

since the radius of $Bin_{init}(\theta)$ is $2b$.

2. $\varphi_F(v) \in Carry\_in(\theta)$. This means that c was initially assigned to one of the children σ of θ. Recall that in the graph $\tilde{G}$ two centers σ and θ are connected if there is a node $u \in Bin(\sigma)$ and a node $v \in Bin(\theta)$ such that there is an edge between u and v in $\tilde{G^2_{\mu_b}}$ - that is $dist(u,v) \leq 2b$.

TZIUR!!!

$$\begin{aligned} dist(v,c) &\leq dist(c,\sigma) + dist(\sigma,u) + dist(u,w) + \\ &\quad dist(w,\theta) + dist(\theta,v) \leq 10b \end{aligned}$$

¿From the above we see that if σ and θ are neighbors in $\tilde{G}$ then $dist(\sigma,\theta) \leq 6b$.

3. $\varphi_F(v) \in Bin_{init}(\theta')$.

   TZIUR!!!

$$\begin{aligned} dist(v,c) &\leq dist(v,\theta) + dist(\theta,\theta') + dist(\theta',c) \\ &\leq 10b \end{aligned}$$

4. $\varphi_F(v) = c$ and $c \in Carry\_in(\theta') \setminus Bin_{init}(\theta)$ (that is in φ_I the node c was assigned to a center θ'' that is a sibling of θ in the spanning tree).

   TZIUR!!!

$$\begin{aligned} dist(v,c) &\leq dist(v,\theta) + dist(\theta,\theta') + dist(\theta',cnt'') + \\ & \quad dist(\theta'',c) \leq 2b + 6b + 6b + 2b = 16b \end{aligned}$$

All the computations are polynomial in n. Therefore we have shown

Proposition 3.14 There exists a polynomial time approximation algorithm for the L-balanced κ-center problem with approximation ratio 16. ∎

3.3 Improvements

Further improvement in the approximation ratio can be achieved by modifying the algorithm, and making a more careful choice of centers and sets $Carry\_in$ during the tree contraction phase, reducing the approximation ratio to 10. To achieve this, observe the following. First, the clients of centers belonging to $\mathcal{E}$ are at most at distance $2b$ from their centers. Secondly, if v is assigned to a center c in its original bin, then $dist(v,c) \leq 4b$.

Let θ' be the parent of θ in the spanning tree. The nodes θ and θ' are connected in $\tilde{G}$. Therefore there exists $w(\theta) \in Bin_{init}(\theta)$ and $v(\theta) \in Bin_{init}(\theta')$, such that $dist(w,v) \leq 2b$. Thus

$$\begin{aligned} dist(w,\theta') &\leq dist(w,v) + dist(v,\theta') \\ &\leq 2b + 2b = 4b \end{aligned}$$

We call the vertex $w(\theta)$ *special* and the vertex $v(\theta)$ *nearset*. If $Carry\_out(\theta) \neq \emptyset$ then we include $w(\theta)$ in $Carry\_out(\theta)$.

Let $\theta_1, \theta_2, \ldots, \theta_l$ be the children of θ in the spanning tree. Assume that the sets $Carry\_out(\theta_i)$ are already defined, and now the algorithm takes care of the set $Bin_{final}(\theta)$. At this point choose the vertices that will belong to $Carry\_out(\theta)$ (the size of the set is already known, the set must include $w(\theta)$ and all the other vertices must belong to $Bin_{init}(\theta)$). Now, let us look at the vertices $v(\theta_1), \ldots, v(\theta_l)$. Notice that it is possible that $v(\theta_i) = v(\theta_j)$, for $i \neq j$. Let

$$package(v) = \bigcup_{j \in A(v)} Carry\_out(\theta_j) \cup \{v\},$$

where $A(v)$ is the set of centers θ_j for which v is the *nearest* node of $w(\theta_j)$. Let

$$|package(v)| = kL + \epsilon.$$

Notice that $|Carry\_out(\theta)| < L$ for any center θ. Therefore if $|package(v)| = kL + \epsilon$ then are at least k distinct special nodes. Appoint any k of the *special* nodes belonging to v as centers and distribute kL nodes of $package(v)$ between them. Let u be such a node, and c such a center, where $u \in Carry\_out(\theta_j)$ and $c = w(\theta_m)$. Then

$$\begin{aligned} dist(u,c) &\leq dist(u,\theta_j) + dist(w,\theta_j) + dist(w(\theta_j),v) + dist(v,w(\theta_m)) \\ &\leq 2b + 2b + 2b + 2b = 8b \end{aligned}$$

The "leftovers" from the sets $package(v)$ will be served by a subset of the *nearest* nodes. In each set $package(v)$ there is a distinct *nearest* node, v. There are less than L nodes of $package(v)$ that do not get served by *special* nodes. Let there be m "packages" that belong to θ, then there are m special nodes and less than mL nodes that are "leftovers". Therefore all these nodes can be served by a subset of the special nodes. In this case:

$$dist(u,c) \leq dist(u,\theta_j) + dist(\theta_j, w(\theta_j)) + dist(w(\theta_j),v) + dist(v,\theta) + dist(\theta,v') \leq 10b,$$

where v' is a *nearest* node (not necessarily of u's set).

The rest of the nodes belong to $Bin_{init}(\theta)$ and are either in $Carry\_out(\theta)$ or get served by centers belonging to $Bin_{init}(\theta)$. Hence we get:

Proposition 3.15 There exists a polynomial time approximation algorithm for the L-balanced κ-center problem with approximation ratio 10. ∎

If $\kappa = O(\log n)$, then there exists a different approximation scheme achieving an approximation ratio of 4. The algorithm achieving this approximation ratio is depicted in Figure 8.

As in the previous case, let us analyze the algorithm for the bottleneck graph G_μ associated with $c_\mu = \mathcal{R}_{opt}(G, L, \kappa)$. Let us denote this radius by b. Let $X_{opt} = \{x_1, ..., x_\kappa\}$ be

For $\mu = 1$ **to** $|E|$ **do:**

1. Choose a maximal independent set $C = \{\theta_1, ..., \theta_{k_0}\}$ in G_μ^2.
2. **If** $k_0 > \kappa$ **then** skip to the next μ.
3. Cycle through all possible vectors $\bar{s} = (s_1, ..., s_{k_0})$ such that $\forall i \; s_i \geq 0$, and $\sum s_i = \kappa$.
4. **For** each such vector $\bar{s}$ **do:**

   Choose a set C' consisting of s_i arbitrarily chosen points in every neighborhood $\Gamma_{G_\mu}(\theta_i)$ in G_μ /* notice $|C'| = \kappa$ */

   If not successful then skip to next $\bar{s}$

   Call $G_R \leftarrow Flow\_graph(C', V \setminus C', 4 \cdot c_\mu, L - 1, 1, 1)$

   Compute the maximum flow in G_R.
   If the maximum flow is $|V \setminus C'|$ **then** go to **Success**
5. **End-for**

End-for
Success: Run $Assign(G_\mu, C', L)$

Figure 8: Algorithm $Log\_Centers(G, L, \kappa)$

an optimal solution. Let $C = \{\theta_1, ..., \theta_{k_0}\}$ be the set chosen in step 1 of our algorithm. For each x_i, there exists a θ_j, such that θ_j is a neighbor of x_i in G_b^2 (otherwise the set $C \cup \{x_i\}$ is an independent set and c is not maximal). An x_i can be a neighbor of several θ_j's, but let us arbitrarily associate one chosen θ_j with each x_i. Let s'_i denote the number of different vertices x_j that are associated with θ_i in this way. The vector $\bar{s}' = \{s'_1, ..., s'_{k_0}\}$ satisfies $\forall i \ s'_i \geq 0$ and $\sum s'_i = \kappa$, and thus corresponds to one of the vectors considered by the algorithm. In examining this vector, the algorithm placed s'_i arbitrary centers, $\{v^i_1, ...v^i_{s'_i}\}$, in θ_i's neighborhood in G_b. Let $x_{i_1}, ..., x_{i_{s'_i}}$ be the optimal centers associated with θ_i. Let us arbitrarily match the x_{i_j}'s with the v^i_j in θ_i's neighborhood. Each such optimal center, x_{i_j} is at distance at most $2b$ from θ_i, and each new center, v^i_j, is at most at distance b from θ_i, therefore $dist(x_{i_j}, v^i_j) \leq 3b$. Since X_{opt} is an optimal solution to the problem with radius b, our solution has radius $4b$.

As for the complexity of the algorithm, note that the outer loop is carried out polynomially many times in κ and in n. There are $\binom{2\kappa}{\kappa} \approx \frac{2^{2\kappa}}{\sqrt{\kappa}}$ possible $\bar{s}$-vectors for each trial. Since $\kappa = O(\log n)$, this is polynomial in n, as are all the other steps of the algorithm.

Proposition 3.16 There exists a polynomial time approximation algorithm for the L-balanced κ-center problem with approximation ratio 4, under the additional assumption that $\kappa = O(\log n)$. ∎

4 The balanced weighted centers problem

In this section we consider the weighted version of the L-balanced κ-centers problem. It is assumed that every node v has a weight $\omega(v)$, where $\omega(v)$ is a positive real number, and we look for a solution with minimal radius, in which the sum of the weights of the centers is at most κ (notice that κ is a real number now). For $U \subseteq V$, define $\omega(U) = \sum_{u \in U} \omega(u)$.

Definition 4.1 The *L-balanced, κ-weighted centers problem* is defined as follows:
Input: Graph $G = (V, E)$ with weights $\omega(v)$ on the nodes and $c_{(u,v)}$ on the edges, an integer L, and a real $\kappa > 0$.
Goal: Select an L-balanced center assignment (C, φ), such that $\omega(C) \leq \kappa$, minimizing the radius $\mathcal{R}(\mathcal{S})$ of the induced assignment tuple $\mathcal{S}$.

Let us denote the optimal radius by $\mathcal{R}_{wopt}(G, L, \omega, \kappa)$.

As a basis for the approximation we shall make use of an initial solution for the *unweighted* problem, that enjoys the *minimum cardinality* property defined below.

For $\mu = 1$ **to** $|E|$ **do**

1. Choose a maximal independent set in G_μ^2. Let this set be $C = \{\theta_1, \theta_2, \ldots, \theta_\ell\}$.
2. Call Procedure $(\hat{C}_i, \hat{\varphi}_i) \leftarrow Allocate(G_i, L, C_i)$ for each connected component G_i of G_μ^2.
3. Let $(\hat{C}, \varphi) = \cup(\hat{C}_i, \hat{\varphi}_i)$.
4. Construct $\hat{G} = (\hat{V}, \hat{U}, \hat{E})$ as follows: let $\hat{V} = \hat{C} = \{\theta_1, \ldots, \theta_m\}$, and let $\hat{U} = V$. Let $(\theta_i, v) \in \hat{E}$ iff $dist(\theta_i, v) \leq 11c_\mu$. The weight of the edge (θ_i, v) is $\omega(v)$.
5. Compute a minimum weight perfect matching in $\hat{G}$.
6. If there is no perfect matching in the graph or the weight of the matching is greater than κ, then go to the next μ.
7. Otherwise let $C' = \{v_1, \ldots v_m\}$, where θ_i was matched to v_i in the minimum weight perfect matching.
8. Call $G_F \leftarrow Flow\_graph(C', V, 21 \cdot c_\mu, L-1, 1, 1)$.
9. Compute the maximum integral flow f_F in G_F.
10. Assign each node a center according to the flow f_F.

End-for

Return the assignment with the smallest radius.

Figure 9: Algorithm $Weighted\_Centers(G, L, \kappa)$

Definition 4.2 Consider an instance of the L-balanced κ-weighted centers problem. Let (C, φ) be a solution to the problem without the weight restriction. Then (C, φ) is a *minimum cardinality solution* for the problem if for every optimal solution (X_{opt}, φ_{opt}) for the weighted problem, and for every $A \subseteq C$, the subset of centers of $X \subseteq X_{opt}$ serving the clients of A satisfies $|X| \geq |A|$.

The minimum cardinality solution (C, φ) is said to have approximation ratio t if its radius is at most $t \cdot \mathcal{R}_{wopt}(G, L, \omega, \kappa)$.

The approximation algorithm $Weighted\_Centers(G, L, \kappa)$ is given in Figure 9. The idea is to start with applying Procedure *Allocate* and generate a minimum cardinality solution, and then use Hall's theorem to derive the ratio bound. When finding a solution for the unweighted problem, there is no bound on the number of centers. The analysis is given by the following lemmas.

Lemma 4.3 The call to Procedure *Allocate* in line 2 of algorithm $Weighted\_Centers(G, L, \kappa)$ returns a minimum cardinality solution (ignoring the node weight constraint) with approximation ratio 10.

Proof: Let $A \subseteq \hat{C}$, and partition A into two sets, $A_{\mathcal{E}} = A \cap \mathcal{E}$, and $A_{\mathcal{F}} = A \setminus A_{\mathcal{E}}$. Let $X \subseteq X_{opt}$ be the set of optimal centers for the weighted problem that serve the clients of A. In order to prove that the solution provided by the procedure is a minimum cardinality one, we shall have to show that $|A| \leq |X|$. Define $X_{A_{\mathcal{E}}} \subseteq X$ to be the set of optimal centers serving $A_{\mathcal{E}}$. Let $X_{A_{\mathcal{F}}} = X \setminus X_{A_{\mathcal{E}}}$. A proof along the lines of Claim 3.6 shows that $|X_{A_{\mathcal{E}}}| = |A_{\mathcal{E}}|$. It is easy to see that the clients of $A_{\mathcal{F}}$ are not served by $X_{A_{\mathcal{E}}}$ using arguments that appear in Claim 3.8. Next we claim:

Claim 4.4 $|X_{A_{\mathcal{F}}}| \geq |A_{\mathcal{F}}|$.

Proof: The set $\mathcal{F}$ (together with its clients) is partitioned into connected components. Our solution is such that in each connected component at most one center has fewer than L clients. Let $\{\theta_{i+1}, \ldots, \theta_k\}$ be the centers in one of the connected components of $\mathcal{F}$. Then these centers serve at least $(k-i)L + 1$ clients, and therefore there are at least this many optimal centers associated with them. An optimal center having clients in one connected component of $\mathcal{F}$ cannot have clients from another component (see Claim 3.11). ▮

Recall that $X = X_{A_{\mathcal{E}}} \cup X_{A_{\mathcal{F}}}$. We have shown that $|X_{A_{\mathcal{E}}}| = |A_{\mathcal{E}}|$ and $|X_{A_{\mathcal{F}}}| \geq |A_{\mathcal{F}}|$, and $X_{A_{\mathcal{E}}} \cap X_{A_{\mathcal{F}}} = \emptyset$, therefore $|X| \geq |A|$. This completes the proof of Lemma 4.3. ▮

Procedure *Allocate* returns a solution with approximation ratio 10, therefore this minimal cardinality solution (that does not necessarily obey the weight constraint) is such that its radius is at most 10 times larger than the ratio of the optimal solution. It remains to be seen how our solution meets the weight constraint.

Lemma 4.5 Algorithm $Weighted\_Centers(G, L, \kappa)$ is a polynomial time algorithm for the L-balanced κ-weighted centers problem with approximation ratio 21.

Proof: In the bipartite graph, $\hat{G}$, every center of the minimum cardinality solution is connected to every vertex at distance at most $11b$, where the weight of an edge is the weight of the node the center is connected to. The algorithm looks for a minimum weight perfect matching in $\hat{G}$.

We need to argue that for the optimal b such a perfect matching exists, and moreover, that $\omega(C') \leq \kappa$. It suffices to show that there exist a perfect matching between $\hat{V} = \hat{C}$ and a subset of size m of $\hat{X}$, where $\hat{X}$ is a subset of the centers in some optimal solution X to the problem (i.e., a solution with radius $\mathcal{R}_{wopt}(G, L, \omega, \kappa)$ and weight $\omega(X) \leq \kappa$). If this is the case, then the weight of the minimum weight perfect matching is at most the weight of the optimal solution. By Hall's theorem there exists such a perfect matching iff for every subset M of $\hat{V}$, the number of neighbors of M in $\hat{X} \subseteq \hat{U}$ is at least the number of nodes in M.

Let M' be the set of clients of M in the solution (C, φ), and let X' be the set of centers serving these clients in the optimal solution. Notice that $dist(M, X') \leq 11b$, since the distance of θ to its clients is at most $10b$, and the distance of the client to its optimal center is at most b. Therefore the set X' is a subset of the set of neighbors of M in the graph $\hat{G}$. Finally, by the definition of a minimum cardinality solution, $|X'| \geq |M|$. This proves the existence of a perfect matching as desired.

Let $\{(\theta_1, v_1), \ldots, (\theta_m, v_m)\}$ be the selected minimum weight perfect matching. The centers of the L-balanced κ-weighted centers problem will be $C' = \{v_1, \ldots, v_m\}$. We assign them clients using Procedure *Flow_graph*, as described in Section 2. There exists an L-balanced assignment with radius $21b$, since v_i is at most at distance $11b$ from θ_i, and θ_i's clients are at most at distance 10 from θ_i. ▮

To summarize:

Proposition 4.6 There exists a polynomial time approximation algorithm for the L-balanced, κ-weighted centers problem with approximation ratio 21. ▮

5 Algorithms for ρ-dominating sets

Let us now consider the dual problem, where given ρ, the maximal allowable distance of a node from the set of centers to be chosen, and L, a bound on the number of clients assigned to a center, we want to minimize the number of centers.

Definition 5.1 The *L-balanced ρ-dominating set* problem is defined as follows:
Input: Graph $G(V, E)$, integers $L, \rho \geq 1$.
Goal: Select an L-balanced, ρ-dominating assignment (C, φ), such that C is of minimal size.

Recall that in the ρ-dominating set problem, a bound ρ is given on the maximal distance between a node and its center and the aim of the algorithm is to minimize the number of centers. We denote the optimal solution to this problem by $\mathcal{C}(G, L, \rho)$. We now give an approximation algorithm with ratio $\lceil \ln n \rceil$.

In order to approximate the problem, we use an iterative algorithm that is *greedy* in the following sense. We start with an empty set C. At each iteration we examine the possibility of adding to C any vertex v not in C, such that $C \cup \{v\}$ can serve a maximum number

1. Call $G(C,L) \leftarrow Flow\_graph(C, V \setminus C, \rho, L-1, 1, 1)$.
2. Compute the maximum integral flow function f_{G_ρ} in G_ρ.
3. Set $\varphi(v)$ to be the center θ such that $f_{G_\rho}(\theta, v) > 0$.
   Set $\varphi(\theta) = \theta$ for any center $\theta \in C$.

Figure 10: Algorithm $Comp\_Min$ for computing a minimal uniform function for C and L.

clients (but possibly not all of them), when imposing the restriction that no server serves more than L clients. Before stating the main algorithm we introduce the following auxiliary definitions and lemmas.

An assignment is *feasible* if it meets the distance and load constraints. The assignment we consider may be *partial assignment*, i.e., not every node of the graph is assigned a center. Given a feasible (possibly partial) assignment φ into a set C, denote by $U(\varphi)$ the set of unassigned clients, namely,

$$U(\varphi) = V \setminus Dom(\varphi),$$

where $Dom(\varphi)$ denotes the domain of φ. For any feasible assignment φ define the *excess* of φ as

$$X(C, \varphi) = |U(\varphi)|.$$

In particular, for any complete feasible function φ, $X(C, \varphi) = 0$. Let $\mathcal{P}$ denote the set of feasible functions for given C and L. Denote

$$X(C) = \min_{\varphi \in \mathcal{P}} \{X(C, \varphi)\}.$$

Note that $X(\emptyset) = |V|$. We call an assignment function φ such that $X(C) = X(C, \varphi)$, a *minimal function* for C and L. In Fig. 10 we describe procedure $Comp\_Min$ that computes a minimal function for a given set C and integer L. (We allow a center to serve only $L-1$ clients, since we always assume that a center serves itself.)

Fact 5.2 For a given graph G integers L and ρ and the set of centers C, let φ' be an arbitrary feasible (possibly partial) assignment. Let $F(C)$ be the maximal flow obtained by procedure $Comp\_Min$. Then

1. $X(C, \varphi') \geq |V| - F(C)$.
2. $X(C) = |V| - F(C)$.

1. $C \leftarrow \emptyset$
2. **While** $X(C) > 0$ **do**
   (a) Choose a vertex $\theta \in V \setminus C$ such that $X(C \cup \{\theta\})$ is minimal, using Procedure $Comp\_Min$.
   (b) $C \leftarrow C \cup \{\theta\}$.

   End-While

Figure 11: Algorithm $Dominating\_Set(G, L, \rho)$

Proof: In computing $F(C)$ we have augmented the flow as much as possible, considering the constraint that a server does not serve more than $L - 1$ customers (plus itself). Assume that part 1 does not hold, namely,

$$X(C, \varphi') < |V| - F(C).$$

Then, $F(C) < |V| - X(C, \varphi')$. Let us define a new flow function as follows. For every center $\theta \in C$ send a unit of flow to every node it serves in the assignment φ' (excluding itself). In this flow function every node served by the centers receives a unit of flow, therefore the total amount of flow obtained by this process is $|V| - X(C, \varphi')$ and it exceeds the maximal flow $F(C)$. This leads to a contradiction. Part 2 of the claim follows directly from part 1. ∎

We now describe the main algorithm as follows. The algorithm operates sequentially in a greedy fashion, adding at each iteration a new vertex θ to the set of centers, in such a way that it minimizes the new excess, $X(C \cup \{\theta\})$. The formal description is given in Figure 11.

5.1 Analysis

Clearly, when the process terminates, C is a complete feasible function. (Note that such a function always exists if $L \geq 1$ since every center may serve itself).

Let us denote by C_i the set C at the beginning of the i'th iteration, where $C_0 = \emptyset$. Consider the situation at the beginning of some iteration i, and let $C = C_i$. Let C^* be an optimal choice of centers for the given instance and let φ^* be a corresponding optimal assignment. Denote

$$T_I = C \cap C^* ; \quad T_O = (V \setminus C) \cap C^*.$$

For any feasible function φ, denote by $hit(\varphi, \varphi^*)$ the number of vertices that were assigned to the same center in C^* and C, namely,

$$hit(\varphi, \varphi^*) = |\{v \in V \mid \varphi(v) = \varphi^*(v)\}|.$$

Let OPT denote the set of minimal assignments for C and L,

$$OPT = \{\varphi \mid X(C, \varphi) = X(C)\}.$$

Let $\hat{\varphi}$ be a function in OPT, for which $hit(\hat{\varphi}, \varphi^*)$ is maximal (among the functions in OPT), i.e.,

$$hit(\hat{\varphi}, \varphi^*) = \max\{hit(\varphi, \varphi^*) \mid \varphi \in OPT\}.$$

Lemma 5.3 For any vertex $v \in U(\hat{\varphi})$, $\varphi^*(v) \in T_O$.

Proof: Assume the contrary, namely, there is a vertex $v \in U(\hat{\varphi})$, such that

$$\varphi^*(v) = \theta; \quad \theta \in T_I. \tag{1}$$

Clearly, $|S_{\hat{\varphi}}(\theta)| = L$, for otherwise we can assign v to θ, reducing $X(C)$. Since v is assigned to θ in φ^*, and φ^* is feasible, there is a vertex $w \in S_{\hat{\varphi}}(\theta) \setminus S_{\varphi^*}(\theta)$. Let φ' be an assignment identical to $\hat{\varphi}$, except that $\varphi'(v) = \varphi^*(v)$, and $\varphi'(w)$ is undefined (namely, we take w out of $S_{\varphi'}(\theta)$). Clearly, this assignment is feasible. Also note that $v \in U(\hat{\varphi})$ implies that $X(C, \varphi') = X(C, \varphi) = X(C)$. Thus, $\varphi' \in OPT$. Note, however, that by the way φ' is defined, $hit(\varphi', \varphi^*) = hit(\hat{\varphi}, \varphi^*) + 1$, contradicting the assumption that $\hat{\varphi}$ maximizes $hit(\varphi, \varphi^*)$ among the functions in OPT. ▮

We now wish to show, that there is a vertex in T_O whose addition to C reduces the excess $X(C)$ by a fraction of $\frac{1}{T_O}$.

Lemma 5.4 There exists a vertex $\theta \in T_O$ such that

$$|S_{\varphi^*}(\theta) \cap U(\hat{\varphi})| + 1 \geq \frac{|U(\hat{\varphi})|}{|T_O|}.$$

Proof: Note that

$$U(\hat{\varphi}) = T_O \cup \bigcup_{\theta \in T_O} \left(S_{\varphi^*}(\theta) \cap U(\hat{\varphi})\right).$$

Since $S_{\varphi^*}(\theta) \cap S_{\varphi^*}(\theta') = \emptyset$ for every $\theta \neq \theta'$,

$$|U(\hat{\varphi})| = \sum_{\theta \in T_O} (|S_{\varphi^*}(\theta) \cap U(\hat{\varphi})| + 1).$$

It follows from the pigeonhole principle, that at least one of the terms in the summation is of size $|U(\hat{\varphi})|/|T_O|$ or more, thus proving the desired claim. ▮

Let θ' be the vertex whose existence is asserted by Lemma 5.4. Denote $C' = C \cup \{\theta'\}$.

Lemma 5.5 $X(C') \leq X(C) - |S_{\varphi^*}(\theta') \cap U(\hat{\varphi})| - 1$.

Proof: Define φ' to be an assignment equivalent to $\hat{\varphi}$ except that if $\varphi^*(v) = \theta'$ and $v \in U(\hat{\varphi})$, then set $\varphi'(v) = \theta'$. Clearly, every vertex that is now in $S_{\varphi'}(\theta')$, decreases $X(C)$ by 1 (this follows from the fact that every such vertex was unassigned in $\hat{\varphi}$). Furthermore, the feasibility of φ^* assures that the vertex θ' serves no more than L customers in φ', and thus φ' is feasible. Finally, note that θ itself is extracted from $U(\varphi')$. Therefore $X(C') \leq X(C', \varphi') \leq X(C, \hat{\varphi}) - |S_{\varphi^*}(\theta') \cap U(\hat{\varphi})| - 1$, and the desired claim follows. ▮

Lemma 5.5 enables us to estimate the progress made by the algorithm in each iteration i, in terms of reducing the excess $X(C_i)$.

Lemma 5.6 $X(C_{i+1}) \leq X(C_i) \cdot \left(1 - \frac{1}{|C^*|}\right)$.

Proof: It follows from Lemmas 5.5 and 5.4 that, for θ' and C' as defined above,

$$X(C') \leq X(C) - |S_{\varphi^*}(\theta') \cap U(\hat{\varphi})| - 1 \leq X(C) - \frac{|U(\hat{\varphi})|}{|T_O|} = X(C)\left(1 - \frac{1}{|T_O|}\right) \leq X(C)\left(1 - \frac{1}{|C^*|}\right).$$

The choice of C_{i+1} now implies that

$$X(C_{i+1}) \leq X(C') \leq X(C_i) \cdot \left(1 - \frac{1}{|C^*|}\right). \quad ▮$$

Corollary 5.7 $X(C_i) \leq X(C_0) \cdot \left(1 - \frac{1}{|C^*|}\right)^i = |V| \cdot \left(1 - \frac{1}{|C^*|}\right)^i$.

Since the function $(1 - \frac{1}{x})^x, x \geq 0$ is an increasing function that converges to $\frac{1}{e}$ as x tends to ∞, we conclude that after $k = |C^*| \cdot \lceil \ln n \rceil$ iterations, $X(C_i) < 1$, and hence $X(C_i) = 0$. Since we add a single vertex to the set in each iteration we end up with at most $|C^*|\lceil \ln n \rceil$ centers, therefore

Theorem 5.8 Algorithm $Dominating\_Set(G, L, \rho)$ is a $\lceil \ln n \rceil$ approximation algorithm for the $\rho-$dominating set problem. ▮

6 k-tolerant sets

Finally, we give an approximation algorithm with ratio $k(\log n + 1)$ for finding a k-tolerant set of centers.

Definition 6.1 The *k-tolerant center set* problem is defined as follows:
Input: Given a graph G=(V,E), $A, B \subseteq V$, and an integer $k \geq 1$.
Goal: A minimal size k-tolerant center set C .

Recall that we defined the *k-tolerant* sets problem as follows: Let $A \subseteq V$ be the set of potential servers, and $B \subseteq V$ the set of potential customers. A *k-tolerant A-set for B* (or simply a k-tolerant set) is a subset $C \subseteq A$, such that for every $v \in B$, either $v \in C$ (this is possible only if $v \in A \cap B$) or there are k vertex-disjoint paths from v to C (in particular, to k distinct vertices in C). A solution to the *k-tolerant set* problem is such a set $C \subseteq A$ of minimal size.

Again, we can show that the associated decision problem problem is NP-complete (say, by a reduction from SAT). We give an approximation algorithm with ratio $k \log n$. This algorithm generalizes the algorithm of [Lov75]. The result holds also for the interesting subcase of the algorithm where $A = B = V$.

Given a set of centers C and a vertex v, let $f(v, C)$ denote the number of vertex-disjoint paths from v to vertices of C in G. Using standard flow techniques, it is easy to see that:

Lemma 6.2 There is a polynomial time procedure $F(v, C)$ for computing $f(v, C)$.

Proof: The procedure $F(v, C)$ operates as follows. Let v be the source. Connect all the vertices of C to a sink v_s, assign a capacity of 1 to all edges of G (plus the new edges to the sink), and capacity 1 to all the nodes. On the resulting flow-graph, compute the maximum flow from v to v_s. This flow equals $f(v, C)$. ∎

Algorithm $Tolerant\_Centers(G, k)$ presented in Figure 12, computes a solution C. The algorithm first checks whether there is a feasible solution to the problem, that is, for every $v \in B \setminus A$, there must exist k vertex disjoint paths from v to different nodes in A. This can be tested using Procedure $F(v, A)$. If this condition holds, then $C = A$ is a feasible solution, and we can try to minimize the number of centers.

For the analysis we need the following fact (for derivation see [Lov75]).

Fact 6.3 [Lov75] Let t_j, a_j, $1 \leq j \leq n$ be integers such that $a_j = \sum_{1 \leq i \leq j} i \cdot t_i$ for every $1 \leq j \leq n$. Then

$$\sum_{1 \leq i \leq n} t_i = \frac{a_n}{n} + \sum_{1 \leq i \leq n-1} \frac{a_i}{i(i+1)}.$$

∎

Let τ denote the size of the optimal set C, and let $t = |C|$ for the set C selected by the algorithm.

Lemma 6.4 $t \leq \tau k(\log n + 1)$.

Proof:

For every $v \in B \setminus A$ compute $f(v, A)$.
If $\exists v$ s.t. $f(v, A) < k$ then return ('No solution')
Else
$C \leftarrow \emptyset$
$T \leftarrow B$
While $T \neq \emptyset$ **do:**
  For every $v \in A \setminus C$
  let $G(v, T, C) = \{w \in T \mid f(w, C \cup \{v\}) = f(w, C) + 1\}$
  and let

$$g(v, T, C) = \begin{cases} |G(v, T, C)|, & \text{if } v \notin B, \\ |G(v, T, C)| + 1, & \text{otherwise.} \end{cases}$$

  Let v_0 be the vertex maximizing $g(v, T, C)$.
  $C \leftarrow C \cup \{v_0\}$
  $T \leftarrow T \setminus \{w \mid f(w, C) = k\} \setminus \{v_0\}$.
End-while

Figure 12: Algorithm $Tolerant\_Centers(G, k)$

Consider the sequence of sets (T_i, C_i) generated during the iteration i of the algorithm, and let $v_0^i = C_i \setminus C_{i-1}$, be the vertex added to C in iteration i. For each such pair (T_{i-1}, C_{i-1}), let g_i denote the number of paths added to vertices in T_{i-1} by the choice of v_0^i in iteration i, i.e $g_i = g(v_0^i, T_{i-1}, C_{i-1})$. Note that g_i is a non-increasing sequence starting with n or less. Let $(\hat{T}_i, \hat{C}_i)$ (for $i = n, \ldots, 1$) denote the first pair (T_j, C_j), for which

$$g(v_0^{j+1}, T_j, C_j) \leq i,$$

and let $a_i = |\hat{T}_i|$. Let t_i, for $1 \leq i \leq n$, denote the number of steps of the algorithm between $\hat{T}_i$ and $\hat{T}_{i-1}$. Then

$$k \cdot a_j \geq \sum_{1 \leq i \leq j} i \cdot t_i \qquad \text{for every } 1 \leq j \leq n.$$

The inequality follows from the fact that $k \cdot a_j$ is an upper bound on the number of times we have to hit the set $\hat{T}_j$ before the algorithm is over, while the sum on the other side of the inequality is a lower bound on the number of hits the algorithm scores from this phase until the end of the algorithm. Notice, that if a $v \in T$ is chosen as a center, it is removed from T without having to hit it k times.

By Fact 6.3,

$$t = \sum_{1 \leq i \leq n} t_i \leq \frac{k \cdot a_n}{n} + \sum_{1 \leq i \leq n-1} \frac{k \cdot a_i}{i(i+1)}. \tag{2}$$

Let τ_i denote the size of the minimal set C_i that solves the problem for the vertices of $\hat{T}_i$. Then $\tau \geq \tau_i \geq a_i/i$, since at most i vertices can be removed from $\hat{T}_i$ by adding a single new vertex to the set of centers. By (2) we get

$$t \leq \sum_{1 \leq i \leq n} \frac{k\tau}{i} \leq k\tau(\log n + 1).$$

▮

Proposition 6.5 There exists a polynomial time approximation algorithm for the k-tolerant set problem with approximation ratio $k(\log n + 1)$. ▮

Acknowledgments

We would like to thank Madan Gopal and David Shmoys for helpful discussions.

References

[Ber85] D. Bertsekas. A unified framework for primal-dual methods in minimum cost network flow problems. *Math. Prog.*, 32:125–145, 1985.

[BG87] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[BG89] A. Bouloutas and P.M. Gopal. Some graph partitioning problems and algorithms related to routing in large computer networks. In *Proc. 9th IEEE Conf. on Distributed Computing Systems*, pages 362–370, Newport Beach, CA, 1989.

[Eve79] S. Even. *Graph Algorithms.* Computer Science Press, 1979.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H Freeman and Company, 1979.

[HR88] A.R. Hevner and A. Rao. Distributed data allocation strategies. In *Advances in Computers, Vol. 27*, pages 121–155. Academic Press, 1988.

[HS86] D.S Hochbaum and D. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33:533–550, 1986.

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[MK83] K. Murthy and J. Kam. An approximation algorithm to the file allocation problem in computer networks. In *Proc. 2nd ACM Symp. on Principles of Database Systems*, pages 258–266. ACM, 1983.

[ML77] H.L. Morgan and K.D. Levin. Optimal program and data locations in computer networks. *Comm. of the ACM*, 20:315–322, 1977.

[Pel90] D. Peleg. Distributed data structures: A complexity oriented view. In *4th Int. Workshop on Distributed Algorithms*, September 1990.