

# Modeling and Practical Evaluation of a Service Location Problem in Large Scale Networks

Olivier Beaumont, Nicolas Bonichon, Hubert Larchevêque  
 Université de Bordeaux, Laboratoire Bordelais de Recherche en Informatique  
 INRIA Bordeaux Sud-Ouest

**Abstract**—We consider a generalization of a classical optimization problem related to server and replica location problems in networks. More precisely, we suppose that a set of users distributed over a network wish to have access to a particular service proposed by a set of providers. The aim is then to distinguish a set of service providers able to offer a sufficient amount of resources in order to satisfy the requests of the clients. Moreover, a quality of service following some requirements in terms of latencies is desirable. A smart repartition of the servers in the network may also ensure good fault tolerance properties. We model this problem as a variant of Bin Packing, namely Bin Packing under Distance Constraint (BPDC) where the goal is to build a minimal number of bins (*i.e.* to choose a minimal number of servers) so that (i) each client is associated to exactly one server, (ii) the capacity of the server is large enough to satisfy the requests of its clients and (iii) the distance between two clients associated to the same server is minimized. We prove that this problem is hard to approximate even when using resource augmentation techniques : we compare the number of obtained bins when using polynomial time algorithms allowed to build bins of diameter at most  $\beta d_{\max}$ , for  $\beta > 1$ , to the optimal number of bins of diameter at most  $d_{\max}$ . On the one hand, we prove that (i) if  $\beta = (2 - \epsilon)$ , BPDC is hard to approximate within any constant approximation ratio, for any  $\epsilon > 0$ ; and that (ii) BPDC is hard to approximate at a ratio lower than  $\frac{3}{2}$  even if resource augmentation is used. On the other hand, if  $\beta = 2$ , we propose a polynomial time approximation algorithm for BPDC with approximation ratio  $\frac{7}{3}$  in the general case. We show how to turn an approximation algorithm for BPDC into an approximation algorithm for the non-uniform capacitated  $K$ -center problem and vice-versa. Then, we present a comparison of the quality of results for BPDC in the context of several Internet latency embedding tools such as Sequoia and Vivaldi, using datasets based on PlanetLab latency measurements.

## I. INTRODUCTION

### A. Problem definition and motivations

Bin Packing is a classical problem that has been studied under many variants (see [14] for a survey). In this paper, we study the generalization where elements belong to a metric space and the maximal distance between two elements belonging to the same bin has to be smaller than a given threshold  $d_{\max}$ . A given weight threshold  $W$  must not be overtaken for each bin and all the elements must belong to exactly one bin. Then, the goal is to minimize the number of bins of diameter at most  $d_{\max}$  and weighting no more than  $W$ .

The motivation for studying BPDC is inspired by our previous works on Bin Covering under Distance Constraint [4], [5], the distance constrained version of the Bin Covering problem [2].

BPDC is closely related to some classical problems in the area of replica and server location problems in large scale networks such as the Internet. Many variants of server location problems can be found in [26], [29], [7], [21], [19], [8], [22], [28]. Most of these problems consider that a set of users in a network want to have access to a given service. The aim is then to distinguish a set of service providers able to offer a sufficient amount of resources in order to satisfy the requests of the clients. Moreover, a quality of service following some requirements in terms of latencies is desirable. A smart repartition of the servers in the network may enable to minimize the latencies between any client and its associated server, and also to ensure good fault tolerance properties. Moreover, we assume that the frequency of requests issued by the clients is heterogeneous (*i.e.* depends on the client), and that the servers come with a maximal capacity.

The study of BPDC is also motivated by other practical problems such as the one presented in [25], where Lupton *et al.* aim at creating a two-dimensional map of the sky composed of quasars and galaxies. In order to gather data on those galaxies, they make multiple snapshots with a telescope, each capturing data for galaxies in the circular portion of the sky visible through the telescope. In this context, they introduce the Euclidean Capacitated Covering by Disks (ECCD) problem, where the aim is to cover points in a Euclidean plane with a minimum number of disks having a fixed diameter, without violating the maximum capacity of any disk (where the capacity corresponds to the maximum number of galaxies for which spectral data can be gathered in a single snapshot). Heuristics for ECCD have been proposed in [25] and are validated through simulations only. It is worth noting that ECCD is simpler than BPDC, in the sense that the approximation algorithm we propose does not require the underlying metric space to be Euclidean.

### B. Related Works

**Bin Packing and  $K$ -center problems:** An *APTAS* (Asymptotic *PTAS*) has been proposed for Bin Pack-

ing [13]. Simple algorithms have also been proposed with approximation ratios slightly larger than 1.15 (see [14] for a survey). The First-Fit-Decreasing algorithm used in this paper is one of the simplest : it consists in sorting the items to be packed in decreasing order of their weights. Then, elements are inserted in this order into the first bin with enough remaining capacity. If no such bin is available, a new one is created. This algorithm has been proved to achieve a  $\frac{11}{9}$ -approximation ratio in [30].

BPDC is equivalent to Bin Packing with Conflicts [15]. An instance of BPDC can be transformed into an instance of Bin Packing with Conflicts where 2 items are in conflict iff their distance is larger than  $d_{\max}$  in the instance of BPDC. Similarly, an instance of Bin Packing with Conflicts can be transformed into an instance of BPDC by setting the distance between two elements to 1 if they are not in conflict and 2 otherwise. However, the notion of resource augmentation, although being natural in the context of distances, has no clear counterpart in the case of Bin Packing with Conflicts.

The (uniform) capacitated  $K$ -center problem [3] has been introduced by Bar-Ilan *et al.*. This problem involves a set of elements in a metric space and a fixed number  $K$ , corresponding to the number of centers to be placed in the metric space. Each element has to be assigned to one of the  $K$  centers, with the additional constraint that each center can only handle a maximal number  $L$  of elements. Then, the goal is to minimize the maximal distance between a node and its associated center, for a fixed number  $K$  of centers and a fixed capacity  $L$  for each center.

In both problems ( $K$ -center and BPDC), the goal is to build a small number of groups with bounded weight. In BPDC, the maximum diameter of a bin is fixed and the number of groups has to be minimized whereas in the  $K$ -center problem, the number of bins is fixed and the maximal distance between a client and its associated server (which is different from the diameter) has to be minimized. In [20], approximation algorithms for the uniform (weights) capacitated  $K$ -center problem are provided. BPDC is also closely related to variants of the original  $K$ -center problem [18] where intra-cluster distances are to be minimized [17] instead of the distance to the center of the cluster.

**Internet Latency Embedding tools:** In order to assess the practical performance of our algorithm, we consider two embedding tools for Internet latencies. Internet embedding tools are used to map distributed resources connected through the Internet into a simple (usually metric) space. Among the most widely encountered embedding tools in the literature are Vivaldi [10], [12] and Sequoia [27], [1]. These tools assign to each resource a position in a simple space, such that the distance between any two nodes (the latency between them) can be approximated by their distance in the simple space. Of course, another possibility would consist in computing and using the whole latency matrix  $\mathcal{L}$ ,

where  $\mathcal{L}_{i,j}$  denotes the measured latency between resources  $i$  and  $j$ . Nevertheless, this approach has two main drawbacks. First, in the context of large scale distributed networks, it is unrealistic to assume that all  $\mathcal{L}_{i,j}$  values can be determined accurately due to the cost of performing all measurements. Then, when using the latency matrix as input, we work in the most general space without any specific topological property. Designing efficient (*i.e.* with good approximation ratios) algorithms in this context turns out to be extremely difficult. On the other hand, embedding tools induce a (small) distortion of latencies, but they enable to work in simpler spaces and therefore to design more efficient approximation algorithms. In the context of a given application, such as resource clustering, only the performance of the pair embedding/clustering algorithm is meaningful. In this paper, we consider 3 embeddings: no embedding (direct use of the latency matrix), Vivaldi [10], [12] and Sequoia [27], [1].

Since the Internet latencies space contains a lot of triangular inequality violations [24], it can be described as a semi-metric space (a space where the distance function does not satisfy the triangular inequality). Thus, it cannot be embedded in a metric space without encountering a loss of accuracy. In fact, the Internet latencies space seems to be close to an infra-metric [23], *i.e.* a space in which the triangular inequality is relaxed in the following way :  $d(u, v) \leq \rho \max(d(u, w), d(v, w))$  for any triple of nodes  $(u, v, w)$ . Results presented in [23] show that most triples satisfy above inequality with  $\rho = 2$ , and almost every triples satisfy it with  $\rho = 10$ . In this paper, we will consider semi-metric spaces in which most of triples satisfy this inequality for a small value of  $\rho$  and approximation results will be given as a function of  $\rho$ .

### C. Contributions

In this paper, we present an approximation algorithm for BPDC based on resource augmentation. More specifically, we compare the number of obtained bins when using a polynomial time algorithm allowed to build bins of diameter at most  $\beta d_{\max}$ , for  $\beta > 1$ , to the optimal number of bins of diameter at most  $d_{\max}$ . More precisely, we will say that  $\mathcal{A}$  is an  $(\alpha, \beta)$ -approximation algorithm based on resource augmentation [11], [16] for BPDC if (i) it runs in polynomial time, (ii) the bins returned by  $\mathcal{A}$  have diameter at most  $\beta d_{\max}$  and (iii) the number of bins is at most  $\alpha \text{OPT}_{\text{BPDC}}$ , where  $\text{OPT}_{\text{BPDC}}$  denotes the optimal number of bins with distance constraint set to  $d_{\max}$ .

In the context of large scale distributed platforms, resource augmentation is both efficient and realistic. Indeed, the aggregate amount of requests a server can handle really needs to be lower than the capacity of the server, in order for the server to be able to process the requests, whereas the threshold on the maximal latency between a server and one of its client is somehow weaker, since the server would still

be able to handle its requests if it is violated (it would then simply do it slowly).

In this context, we prove that if  $\beta = (2-\epsilon)$ , this problem is hard to approximate in the general case, within any constant approximation ratio (even in any function in the number of elements), and that it is also hard to approximate within any ratio lower than  $\frac{3}{2}$ , whatever the resource augmentation. On the other hand, we propose in this paper a  $(\frac{7}{3}, \rho)$ -approximation algorithm for Bin Packing under Distance Constraint in a  $\rho$ -inframetric.

We also prove that an approximation algorithm for BPDC can be turned into an approximation algorithm for capacitated  $K$ -center problem and vice-versa. As far as we know, this provides the first approximation algorithm for non-uniform (weights) capacitated  $K$ -center problem. In the simpler context of the uniform capacitated  $K$ -center problem, the performance of the adaptation of the algorithm we propose for BPDC equals the performance of the best known approximation algorithm [20].

Then we present a comparison of the performance of the proposed algorithm for BPDC in the context of several embedding tools for the latencies over Internet. Using different embedding tools, together with the same algorithm and actual latency measures, we can decide which embedding tool offers in practice for realistic datasets the best embedding for the specific optimization problem we consider.

The rest of this paper is organized as follows. In Section II, we present the definitions and the notations used throughout this paper. In Section III we prove that BPDC cannot be approximated even by using a  $(2-\epsilon)$  resource augmentation on the diameter, and that it is also hard to approximate at any ratio lower than  $\frac{3}{2}$  whatever the resource augmentation. In Section IV, we propose a  $(\frac{7}{3}, \rho)$ -approximation algorithm in  $\rho$ -inframetric, therefore achieving the best possible resource augmentation ratio in classical metric spaces. We also present a  $(2, \rho)$ -approximation algorithm for the uniform version of BPDC. Section V is devoted to the relationships between BPDC and  $K$ -center problems and how to turn an approximation algorithm of one problem into an approximation algorithm for the other. Eventually, in Section VI, we present the comparison, on actual datasets, of the performance of the  $(\frac{7}{3}, \rho)$ -approximation algorithm for BPDC using different embedding tools for the latencies over Internet.

## II. NOTATIONS AND DEFINITIONS

In this section, we present the definitions and notations that will be used throughout this paper.

An instance  $\mathcal{I}$  of BPDC can be described as a 5-tuple  $\mathcal{I} = (S, d, w, W, d_{\max})$ , where  $S$  is a set  $S = \{e_1, \dots, e_n\}$  of elements,  $(S, d)$  is a semi-metric space,  $w$  is a weight function,  $W$  is a weight threshold and  $d_{\max}$  is the distance threshold.

Throughout this paper, for the sake of simplicity, we set  $W = 1$  and we normalize the weights of the elements accordingly (*i.e.* divide them by  $W$ ). Moreover, we do not deal with elements whose weight is larger than 1, since such elements cannot be packed in any bin. Thus, an instance of BPDC can be described by a 4-tuple  $\mathcal{I} = (S, d, w, d_{\max})$ , where  $w : S \rightarrow [0; 1[$ , hence the following definition.

*Definition 2.1 (BPDC: Distance Constrained Bin Packing):* Given an instance

$\mathcal{I} = (S, d, w, d_{\max})$ , find a collection of pairwise disjoint subsets  $S_1, \dots, S_K$  of  $S$  of minimal cardinality  $K$  such that  $\forall i \leq K, \sum_{e \in S_i} w(e) \leq 1, \forall (e_u, e_v) \in S_i, d(e_u, e_v) \leq d_{\max}$ .

We denote by  $\text{OPT}_{\text{BPDC}}(\mathcal{I})$  (or simply  $\text{OPT}_{\text{BPDC}}$ ) the minimum value of  $K$  for a given instance  $\mathcal{I}$ .

In order to work on graphs, we will rely on the following tool that builds a graph from a set of points in a metric space.

*Definition 2.2 (Compatibility Graph):* The compatibility graph  $\text{Comp}(\mathcal{I}, d)$  associated to an instance  $\mathcal{I}$  is the graph  $G = (S; E)$  such that  $\forall (u, v) \in S^2, (u, v) \in E \Leftrightarrow d(u, v) \leq d$ .

Observe that  $(e_i, e_j) \in E$  and  $(e_j, e_k) \in E \Rightarrow d(e_i, e_k) \leq \rho d$  in a  $\rho$ -inframetric space ( $\rho \leq 2$  in a metric space). Note that if  $S$  are points in a Euclidean space, and  $L_2$  norm is used to define distances,  $\text{Comp}(\mathcal{I}, 1)$  is the unit-disk graph [9].

## III. INAPPROXIMABILITY WITH SMALL RESOURCE AUGMENTATION

As stated in the introduction, BPDC is equivalent to Bin Packing with Conflicts [15]. Hence BPDC is NP-Complete and hard to approximate. In what follows, we present Theorem 3.1 and Theorem 3.2 that both provide insights on the difficulty of approximating BPDC, even when using resource augmentation.

*Theorem 3.1:*  $\forall \epsilon > 0$ , and  $\forall 0 < \alpha \leq |S|^{1/7-\delta}$ , there is no polynomial time  $(\alpha, (2-\epsilon))$ -approximation algorithm for BPDC unless  $P = NP$ .

*Proof:* Since Bin Packing with Conflicts is itself a generalization of the graph coloring problem, and that it has been shown to be hard to approximate within  $|V|^{1/7-\delta}$  for any  $\delta > 0$  [6], so is BPDC. Moreover, consider a reduction from Bin Packing with Conflicts to BPDC. In a conflict graph, distances between elements have integer values, so that the diameter of any set of elements in a corresponding instance of BPDC is an integer. Thus, for every bin  $B$  built on such an instance of BPDC, if the diameter of  $B$  is less or equal than  $(2-\epsilon)$  then it is at most 1. Thus, the use of a resource augmentation ratio smaller than  $(2-\epsilon)$  does not help to approximate BPDC. ■

*Theorem 3.2:*  $\forall \beta \geq 1$  and  $\forall 1 \leq \alpha < 3/2$ , there is no polynomial time  $(\alpha, \beta)$ -approximation algorithm for BPDC unless  $P = NP$ , whatever the metric space used to define distances.

*Proof:* Ignoring the distance constraint brings us back to classical Bin Packing. Thus a  $(\alpha, \beta)$ -approximation algorithm for BPDC, with  $\alpha \leq 3/2$ , could be used as an approximation algorithm for Bin Packing, ensuring an approximation ratio of  $\alpha$ , which is impossible since it is well known that Bin Packing is hard to approximate within such a ratio (otherwise the 2-partition problem could be solved). ■

#### IV. A GREEDY $(\frac{7}{3}, \rho)$ -APPROXIMATION ALGORITHM FOR BPDC

In this section, we present Algorithm 1, which is an adaptation of the algorithm proposed by Epstein and Levin in [15] for Bin Packing with Conflicts, using First-Fit-Decreasing algorithm to build bins (see Section I-B for a brief description of First-Fit-Decreasing algorithm).

As for the number of built bins, Algorithm 1 ensures an approximation ratio of  $\frac{7}{3}$  in any semi-metric space. When this semi-metric space is a  $\rho$ -inframetric, Algorithm 1 is a  $(\frac{7}{3}, \rho)$ -approximation algorithm for BPDC. When the space is metric, it is a  $(\frac{7}{3}, 2)$ -approximation algorithm for BPDC.

In order to adapt the algorithm presented in [15] to BPDC, we rely on the definition of the extended weight  $e(x)$  of an element  $x$ . If  $w(x) > \frac{1}{2}$ , then  $e(x) = 1$ , and if  $w(x) \in \mathcal{I}_j = (\frac{1}{j+1}, \frac{1}{j}]$  for some integer  $j > 1$ , then  $e(x) = w(x) + \frac{1}{j(j+1)}$ . Moreover, let us denote by  $\text{OPT}_{\text{BPC}}$  the cardinality of an optimal solution for the Bin Packing with Conflicts problem, and by  $\text{OPT}_{\text{GRAPHCOL}}$  the cardinality of an optimal solution for the precoloring extension problem used in [15] to approximate Bin Packing with Conflicts. This precoloring extension problem is a simple extension of the classical coloring problem with the additional constraint that some nodes are already assigned colors in the input.

---

#### Algorithm 1 Greedy $(\frac{7}{3}, \rho)$ -approximation algorithm for BPDC

---

- 1:  $U \leftarrow S$  // elements not grouped yet
  - 2:  $\mathcal{C} = \text{Comp}(\mathcal{I}, d_{\max})$
  - 3: **while** there is a set of three connected items  $\{a, b, c\}$  that can fit into one bin, *i.e.*  $w(a) + w(b) + w(c) \leq 1$ , such that  $e(a) + e(b) + e(c) > 1$  and  $w(c) \leq w(b) \leq w(a) \leq \frac{1}{2}$ , or a pair of connected items  $(a, b)$  that can fit into one bin, *i.e.*  $w(a) + w(b) \leq 1$ , such that  $e(a) + e(b) > 1$  **do**
  - 4: choose such a set of maximum overall extended weight, and put all the elements of this set into a new bin
  - 5: remove from  $U$  this set of elements
  - 6: **end while**
  - 7: build a partition  $M$  of  $U$
  - 8: apply First-Fit-Decreasing on each set of the  $M$  parts induced by the previous partition.
- 

#### Theorem 4.1 (Reformulation of Theorem 12 of [15]):

In Algorithm 1, if  $|M| \leq \text{OPT}_{\text{BPDC}}$  and each set of  $M$  is of diameter at most  $\rho d_{\max}$ , then the number of bins returned by Algorithm 1 is  $(\frac{7}{3}, \rho)$ -approximation algorithm for BPDC is at most  $\frac{7}{3} \text{OPT}_{\text{BPDC}}$  and each bin is of diameter at most  $\rho d_{\max}$ .

In [15] the authors work on a particular class of graphs where computing a coloring (more precisely, solve the precoloring extension problem) of the nodes can be done optimally in polynomial time. Thus, they can build a partition  $M$  such that  $|M| \leq \text{OPT}_{\text{GRAPHCOL}}$  on line 7 of Algorithm 1 and each set of  $M$  is of diameter at most  $d_{\max}$ . Since, from [15], Bin Packing with Conflicts is a generalization of the graph coloring problem,  $\text{OPT}_{\text{GRAPHCOL}} \leq \text{OPT}_{\text{BPC}}$ . Eventually, since Bin Packing with Conflicts and BPDC are equivalent, then  $|M| \leq \text{OPT}_{\text{BPC}} = \text{OPT}_{\text{BPDC}}$  and Theorem 4.1 can be used.

Indeed, each of the  $|M|$  built sets corresponds to a color class, and bins can be built "locally", without taking into account any conflicts within each color class, using a classical Bin Packing algorithm.

Taking advantage of resource augmentation, we can obtain a general result for any compatibility graph on a  $\rho$ -inframetric. Indeed, Algorithm 2 describes how to build a partition  $M$  of  $U$  into sets of diameter  $\rho d_{\max}$ , such that  $|M| \leq \text{OPT}_{\text{BPDC}}$ . Then, in each set, bins can be built without taking into account any distance constraint, using the same classical Bin Packing algorithm. On the other hand, some bins may have diameter  $\rho d_{\max}$  instead of  $d_{\max}$ , thus the resource augmentation ratio.

---

#### Algorithm 2 Building a partition $M$ of $U$ into sets of diameter at most $\rho d_{\max}$

---

- 1: build  $\text{Comp}'(\mathcal{I}, d_{\max})$ , the graph  $\text{Comp}(\mathcal{I}, d_{\max})$  where the edges between two elements of weight larger than  $\frac{1}{2}$  have been removed
  - 2: build a Maximal Independent Set (MIS)  $M$  such that elements of weight larger than  $\frac{1}{2}$  belong to  $M$ .
  - 3: arbitrarily associate each node  $\notin M$  to one of its neighbors in  $M$  so that  $|M|$  packs of elements are returned.
- 

*Lemma 4.2:* The partition  $M$  of  $U$  built by Algorithm 2 satisfies  $|M| \leq \text{OPT}_{\text{BPDC}}$ .

*Proof:* Two elements of  $M$  cannot belong to the same bin in an optimal packing. Indeed, since they belong to the MIS of  $\text{Comp}'(\mathcal{I}, d_{\max})$ , either their distance in  $\text{Comp}(\mathcal{I}, d_{\max})$  is at least 2 (and thus they are too far away to belong to the same bin) or both elements have weight larger than  $\frac{1}{2}$  and in this case, the bin would be too heavy. Hence,  $|M| \leq \text{OPT}_{\text{BPDC}}$ . ■

*Theorem 4.3:* Algorithm 1 together with Algorithm 2 is a  $(\frac{7}{3}, \rho)$ -approximation algorithm for BPDC when  $d$  is a  $\rho$ -inframetric.

Theorem 4.3 is directly obtained using Theorem 4.1 and Lemma 4.2. Thus, Algorithm 1 is optimal with respect to resource augmentation, in the sense that no constant approximation ratio can be achieved using a resource augmentation strictly smaller than 2 in the metric case (cf Section III).

*Remark concerning a uniform weights version:* Let us consider the following "uniform weights" version of BPDC, where all items have the same weight  $x$ . Since the weight of each bin is at most 1, the goal is to build bins containing at most  $\lfloor \frac{1}{x} \rfloor$  elements from  $S$  (still valid for the distance constraint). In this case, lines 2-6 of Algorithm 1 are useless and we obtain Corollary 4.4.

*Corollary 4.4:* Algorithm 1 together with Algorithm 2 is a  $(2, \rho)$ -approximation algorithm for the uniform version of BPDC.

*Proof:* Note that in the uniform case, First-Fit-Decreasing algorithm (or any greedy algorithm) will create bins of  $\lfloor \frac{1}{x} \rfloor$  elements (thus, only optimal ones) except for at most one bin in each of the  $|M|$  packs built by Algorithm 2. Thus, the number of non optimal bins is smaller than  $|M|$  and, by Lemma 4.2,  $|M| \leq \text{OPT}_{\text{BPDC}}$ . Combined with the other optimal bins, this provides the claimed approximation ratio. ■

#### V. COMPARISON WITH THE CAPACITATED $K$ -CENTER PROBLEM

In this section, we present how to build an approximation algorithm for the capacitated (uniform or non-uniform)  $K$ -center problem (CapKcenter for short) from an approximation algorithm for BPDC.

*Definition 5.1 (Non-uniform Capacitated  $K$ -center):* Given an instance  $\mathcal{K} = (S, d, w, K)$ , i.e. a set  $S = \{e_1, \dots, e_n\}$  of elements, a metric space  $(S, d)$ , a weight function  $w : S \rightarrow [0; 1]$  and an integer  $K$ , find the smallest value  $r_{max}$  and a subset  $X \subseteq S$  of centers whose size is at most  $K$  and an assignment of the elements to the centers in  $X$  such that the overall weight of elements assigned to any center is smaller than 1, and every element is assigned to a center at distance less than  $r_{max}$  from itself.

We denote by  $\text{OPT}_{\text{CAPKCENTER}}(\mathcal{K})$  (or simply  $\text{OPT}_{\text{CAPKCENTER}}$ ) the optimal value of  $r_{max}$  for a given instance  $\mathcal{K}$ . The usual formulation of the capacitated  $K$ -center (see [20]) deals with uniform weights :  $\forall x \in S, w(x) = w$ . Thus, the weight condition corresponds to assign no more than  $1/w$  elements to any center. Usually, for the  $K$ -center problem,  $K$  is given and  $r_{max}$  is to be minimized.

To use resource augmentation for this problem, let us say that  $\mathcal{A}$  is an  $(\alpha, \gamma, \beta)$ -approximation algorithm for (non-uniform) capacitated  $K$ -center if it runs in polynomial time and builds a solution using at most  $\alpha K$  centers (instead of  $K$ ), with an assignment of nodes to centers such that the overall weight of nodes assigned to any center is smaller

than  $\gamma$  (instead of 1), and such that every node is assigned to a center at distance at most  $\beta \text{OPT}_{\text{CAPKCENTER}}$  (instead of  $\text{OPT}_{\text{CAPKCENTER}}$ ).

*Theorem 5.1:* An  $(\alpha, \beta)$ -approximation algorithm for BPDC can be turned into an  $(\alpha, 1, 2\beta)$ -algorithm for CapKcenter. Conversely, an  $(\alpha, 1, \beta)$ -approximation algorithm for CapKcenter can be turned into an  $(\alpha, 2\beta)$ -approximation algorithm for BPDC.

*Proof:* Given  $(S, d, w)$ , a set of weighted elements in a metric space, we consider two natural transformations,  $T$  and  $T'$ , between packings and assignments:

Transformation  $T$ :

- takes as input a set of bins, solution of instance  $\mathcal{I} = (S, d, w, d_{max})$  of BPDC, for a fixed value of  $d_{max}$ ,
- in each bin, an element is arbitrary chosen to become the center of the bin,
- all other elements are assigned to it;

the radius of the obtained assignment is at most the maximal diameter among the input bins.

Transformation  $T'$ :

- takes an assignment as input, solution of instance  $\mathcal{K} = (S, d, w, K)$  of CapKcenter, for a fixed value of  $K$ ,
- for each center,  $T'$  builds a bin composed of the center and all elements assigned to it;

the diameter of the obtained bin is at most twice the radius of the corresponding assignment.

Let us now describe how a  $(\alpha, \beta)$ -approximation algorithm  $\mathcal{A}$  for BPDC can be turned into a  $(\alpha, 1, 2\beta)$ -approximation algorithm for the non-uniform capacitated  $K$ -center problem. Let

$\mathcal{K} = (S, d, w, K)$  be an instance of the non-uniform capacitated  $K$ -center problem. Let

$\mathcal{I}(D) = (S, d, w, D)$  be an instance of BPDC, based on the same set of weighted elements, for any positive value  $D$ .

Let  $d_\alpha$  denote the smallest value such that algorithm  $\mathcal{A}$  computes a solution of BPDC with at most  $\alpha K$  bins on the instance  $\mathcal{I}(d_\alpha)$ . Let  $d_1, d_2, \dots, d_{n(n+1)/2}$  denote the distances between each pair of elements of  $S$  in non decreasing order ( $d_1$  is the distance between the two closest elements and  $d_{n(n+1)/2}$  is the diameter of  $S$ ). Since for any  $d_i \leq D < d_{i+1}$ ,  $\text{Comp}(\mathcal{I}(D), D) = \text{Comp}(\mathcal{I}(d_i), d_i)$ , there is only a polynomial number of values to be considered to compute  $d_\alpha$ .

Since  $\mathcal{A}$  is a  $(\alpha, \beta)$ -approximation algorithm for BPDC, the number of bins created by  $\mathcal{A}$  on  $\mathcal{I}(D)$  is at most  $\alpha \text{OPT}_{\text{BPDC}}(\mathcal{I}(D))$ . Hence, for any  $D < d_\alpha$ ,  $\text{OPT}_{\text{BPDC}}(\mathcal{I}(D)) > \frac{\alpha K}{\alpha} = K$ . Applying  $T'$  to the optimal solution of  $\mathcal{K}$  provides a valid solution for  $\mathcal{I}(2\text{OPT}_{\text{CAPKCENTER}}(\mathcal{K}))$  with at most  $K$  bins. From the last observation, we obtain that  $2\text{OPT}_{\text{CAPKCENTER}}(\mathcal{K}) \geq d_\alpha$ .

Applying  $T$  to the result of  $\mathcal{A}$  on  $\mathcal{I}(d_\alpha)$ , we obtain an assignment of size  $\alpha K$ , of maximum radius at most  $\beta d_\alpha \leq 2\beta \text{OPT}_{\text{CAPKCENTER}}(\mathcal{K})$  and of maximum weight at most 1 for instance  $\mathcal{K}$ . Hence, this adaptation of  $\mathcal{A}$  provides a  $(\alpha, 1, 2\beta)$ -approximation algorithm for the non-uniform capacitated  $K$ -center problem.

Let us now prove how to turn an approximation algorithm for the non-uniform  $K$ -center problem into an approximation algorithm for BPDC. Let us consider a  $(\alpha, 1, \beta)$ -approximation algorithm  $\mathcal{A}'$  for the non-uniform capacitated  $K$ -center problem. Let  $\mathcal{I} = (S, d, w, d_{\max})$  be an instance of BPDC. Let  $\mathcal{K}(K) = (S, d, w, K)$  be an instance of non-uniform capacitated  $K$ -center, based on the same set of weighted elements, for any positive value  $K$ .

Let  $K_\beta$  denote the smallest value such that algorithm  $\mathcal{A}'$  computes a solution for the non-uniform  $K$ -center problem with radius at most  $\beta d_{\max}$  on the instance  $\mathcal{K}(K_\beta)$ . Since  $\mathcal{A}'$  is a  $(\alpha, 1, \beta)$ -approximation algorithm for non-uniform capacitated  $K$ -center, the radius of the assignment built by  $\mathcal{A}'$  on  $\mathcal{K}(K)$  is at most  $\beta \text{OPT}_{\text{CAPKCENTER}}(\mathcal{K}(K))$ . Hence, for any  $K < K_\beta$ ,  $\text{OPT}_{\text{CAPKCENTER}}(\mathcal{K}(K)) > \frac{\beta d_{\max}}{\beta} = d_{\max}$ . Applying  $T$  to the optimal solution of  $\mathcal{I}$  provides a solution for  $\mathcal{K}(\text{OPT}_{\text{BPDC}}(\mathcal{I}))$  with radius at most  $d_{\max}$ . From the last observation, we obtain that  $\text{OPT}_{\text{BPDC}}(\mathcal{I}) \geq K_\beta$ .

Applying  $T'$  to the result of  $\mathcal{A}'$  on  $\mathcal{K}(K_\beta)$  provides a packing with at most  $\alpha K_\beta \leq \alpha \text{OPT}_{\text{BPDC}}(\mathcal{I})$  bins with diameter at most  $2\beta d_{\max}$  and with weight at most 1 on the instance  $\mathcal{I}$ . Hence this adaptation of  $\mathcal{A}'$  provides a  $(\alpha, 2\beta)$ -approximation algorithm for BPDC. ■

Note that this result can also be applied on uniform versions of BPDC and capacitated  $K$ -center. In such a setting, combining Theorem 5.1 with Corollary 4.4, Algorithm 1 provides a  $(2, 1, 4)$ -approximation algorithm for the capacitated  $K$ -center problem in the general case. This result has to be compared to the one presented in [20], where a polynomial time algorithm is proposed, providing a  $(\frac{2}{c}, c, 4)$ -approximation ratio, where  $c = \frac{m+1}{m}$  for any  $m \geq 1$ . For example, if  $m$  tends becomes arbitrarily large, this algorithm provides a  $(2, 1, 4)$ -approximation algorithm, which is exactly the same as the one provided by the adaptation of Algorithm 1. Note that in the other way, this particular case providing a  $(2, 1, 4)$ -approximation algorithm for the capacitated  $K$ -center problem becomes a  $(2, 8)$ -approximation algorithm for uniform BPDC, by Theorem 5.1, whereas Algorithm 1 is already a  $(2, 2)$ -approximation algorithm for the same problem.

Moreover, combining Theorem 5.1 with Theorem 4.3, Algorithm 1 provides a  $(\frac{7}{3}, 1, 4)$ -approximation algorithm for CapKcenter. To the best of our knowledge, it is the first approximation result for a non uniform version of CapKcenter, since the family of algorithms presented in [20] cannot easily be adapted to such a non-uniform weights version. Eventually, one could see the difference between those approximation ratios (between  $\frac{7}{3}$  and 2) as the cost of

the non-uniformity of the weights.

## VI. EXPERIMENTAL EVALUATION

In this section, we propose a comparison of the most widely encountered embedding tools for latency estimation, namely Vivaldi [10], [12] and Sequoia [27], [1], in the context of the location problems considered in this paper. For all the embeddings, Algorithm 1 will be used to compute the solution of BPDC.

Vivaldi associates with each node of a network two coordinates in the Euclidean plane plus a height. In such a space, the distance between two nodes having coordinates  $(a_x, a_y, a_h) \in \mathbb{R}^2 \times \mathbb{R}^+$  and  $(b_x, b_y, b_h) \in \mathbb{R}^2 \times \mathbb{R}^+$  is given by  $d(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} + a_h + b_h$ .

Sequoia embeds Internet nodes into one or several weighted trees in which each node is either a leaf or the root, and in which internal nodes are virtual nodes. The distance between two nodes in the original network is approximated by the distance in the embedding tree (or as the median of these distances in the case of multiple trees). In [27], it is claimed that the accuracy of the prediction is higher using Sequoia as soon as a few trees rooted at different nodes are used (10 or 15 trees).

The main originality of the approach we propose is that it is done in the context of a specific application, *i.e.* the placement of servers for heterogeneous amounts of requests. Indeed, the accuracy of embedding tools have already been compared in [27], but the goal was only to estimate the closeness between latency predictions returned by the embeddings and the actual measurements.

In practice, when considering an specific application, the chain is more complicated and the performance of the chain embedding+algorithm has to be evaluated as a whole. Indeed, our experiments will show that the performance obtained when running the same algorithm in different embedding strongly depends on the embedding for server location problems and that the ranking of the embeddings is not the same as the one obtained in [27].

To compare the respective performance of the different embedding tools, we will compare the results obtained using Algorithm 1 with the following embeddings:

- no embedding (direct use of the latency matrix)
- Vivaldi embedding
- 1-tree Sequoia embedding
- 5-trees Sequoia embedding
- 10-trees Sequoia embedding
- 15-trees Sequoia embedding.

### A. Experimental Protocol

In order to perform realistic simulations, we need to estimate the distance (the latencies) between any pair of resources and we need to estimate the heterogeneous capacity of the resources.

In order to estimate latencies, we ran simulations on a real dataset taken from the Meridian project<sup>1</sup> containing all-pairs latency measurements between 2500 nodes arbitrarily chosen from PlanetLab<sup>2</sup>. In this matrix, latency measurements are symmetric.

This matrix contains a lot of triangular inequality violations. The study in [24] makes a difference between triangular inequality violations due to the structure of the Internet itself or due to the traffic on the Internet, and violations due to measurements inaccuracy. Considering the study by Lebhar *et al.* in [23], for each triple, we consider that if the value  $\rho$  associated with this triple is larger than 10, then it might be due to the inaccuracy of the measurement.

Thus, in a first step, we compute for each latency measurement between two nodes, the number of times it appears in a triple for which the  $\rho$  value is larger than 10. Figure 1 depicts the repartition function of this number: a point  $(x, y)$  means that  $y\%$  of the latency values appear in less than  $x$  triples having a  $\rho$  value larger than 10.

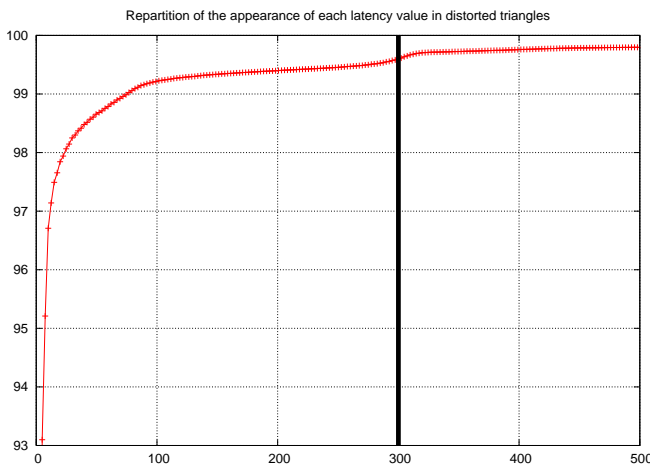


Figure 1. Repartition function of the number of triples having a  $\rho$  value larger than 10.

Using Figure 1, we consider that a latency value is *not-valid* if it appears in more than 300 triangles in which the value of  $\rho$  is larger than 10. Each not-valid latency is replaced by the 2-hop shortest path made of valid latencies. The resulting latency matrix is called the *fixed matrix*.

To obtain this "fixed matrix", only 0.13% of the latency values in the original latency matrix have to be modified. In what follows, we run simulations using both the original latency matrix, and the "fixed matrix".

In order to estimate the heterogeneous amount of requests of the clients, we choose to rely on two distributions of the weights of the elements, namely, the uniform distribution in  $[0, 1)$  and the log-normal distribution with parameters 0

and 1, since both seem to arise naturally in many practical situations (the log-normal distribution can be seen as a bounded power-law distribution, which seems more realistic than a classical power-law). We choose, according to those distributions, two sets of 2500 values.

For each simulation, we rely on the following protocol

- We embed the latency matrix, using either Vivaldi or Sequoia (using one or multiple trees in the case of Sequoia).
- We assign to each element a weight according either to the uniform distribution (in  $[0, 1)$ ) or to the log-normal distribution (with parameters 0 and 1).
- We apply Algorithm 1 for BPDC to the resulting embedding, with the same set of weighted elements but for different values of  $d_{\max}$ , from the minimal distance previously identified to 200ms (since beyond this value, no evolution is observed).

As mentioned in the introduction, it is unrealistic in practice to assume that the whole latency matrix is known. Indeed, in the context of large scale dynamic platforms, the time taken to estimate all latencies is much too high with respect to the dynamics of the system. It is worth noting that the implementation of Vivaldi does not require a centralized knowledge of the matrix, what is not the case for Sequoia to the best of our knowledge.

For each simulation, we plot the total number of bins built. For each built bin, we also estimate its diameter as the maximal distance between any two nodes in the bin, where distances are actual ones, *i.e.* the one of the latency matrix, whatever the intermediate distance estimated by the embedding tool is. For each simulation and for each value of  $d_{\max}$ , we plot the average diameter of built bins, and the percentage of "valid" bins, *i.e.* of bins whose diameter in the latency matrix used is lower than  $2d_{\max}$ , the maximum diameter allowed by the corresponding algorithm.

We ran simulations for different values of  $W$  (the size of bins), and choose a different value for each distribution of the weights. Indeed, we wanted to work with the more meaningful and representative value. For small values of  $W$ , solutions tend to put fewer and fewer elements in each bin, since element weights get closer to the threshold. Thus, the average number of elements per bin tends towards one for any value of  $d_{\max}$ . For large values of  $W$ , the distance constraint becomes the most important and the problem is close to MINIMUM CLIQUE PARTITION. Thus we chose, for each distribution, a weight threshold such that bins contain on average approximately 8 elements.

- In the case of the uniform distribution, we chose  $W = 4$ , since the average weight of an element is 0.5,
- In the case of the log-normal distribution, we chose  $W = 5.2$ .

Eventually, since simulation results were highly similar when using one or the other distribution law to generate

<sup>1</sup><http://www.cs.cornell.edu/People/egs/meridian/data.php>

<sup>2</sup><http://www.planet-lab.org/>

elements' weights, we only present results obtained using the log-normal distribution, with  $W = 5.2$ .

### B. Simulation results

Figures 2 and 3 and 4 respectively depict, for each considered embedding, the number of built bins, the average value of the diameter of built bins and the percentage of valid bins built in each case, for several values of  $d_{\max}$  and using the *fixed* latency matrix. Figures 5 and 6 and 7 also depict, for each considered embedding, the number of built bins, the mean diameter of the built bins and the percentage of valid built bins in each case, for several values of  $d_{\max}$  and using the *original* latency matrix.

Figures 2 and 5 show that there are three intervals for the value of  $d_{\max}$  in each of which the difficulty of BPDC appears for different reasons.

- $d_{\max} \leq 10ms$ . In this case, the weight constraint is weaker than the distance constraint. The difficulty comes from the topology and the problem is close to the MINIMUM CLIQUE PARTITION PROBLEM in the compatibility graph.
- $10ms < d_{\max} \leq 60ms$ . In this case, both constraints (weight and distance) have to be taken into account.
- $60ms < d_{\max}$ . The distance constraint is the weakest one and the topology is not crucial anymore. The weighted constraint is the most important and BPDC is close to classical Bin Packing.

Figures 4 and 7 show that Algorithm 1 builds a higher percentage of valid bins when using Vivaldi embedding, using either the original or the fixed matrix as input. By contrast, when using Sequoia embedding with 1-tree only (either with the original or the fixed matrix as input) the percentage of valid bins is much lower. The direct use of the latency matrix (whatever the latency matrix used) provides performances between those obtained using Vivaldi embedding and those obtained using Sequoia 1-tree embedding.

The information provided by the comparison between the use of the original latency matrix and of the fixed matrix as input of the embedding tools is related to the robustness of the behavior of Algorithm 1 with each embedding. In fact, only a few amount of latency values have been modified between the original and the fixed matrix (0.13% of the values). Thus if, for a given embedding, the performance differences are important when using each of the two matrix as input, this means that only a few changes in the instance can deeply modify the result of the execution of the algorithm, meaning that the joint use of the algorithm and the given embedding is not robust for our application.

Thus, when comparing the figures depicting results obtained using each of the two latency matrices, we observe almost no difference between both datasets when using Vivaldi embedding. It is also the case when no embedding is used (direct use of the latency matrix).

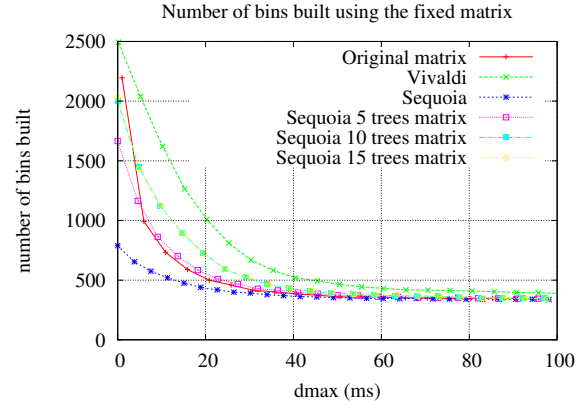


Figure 2. Number of built bins for BPDC, for different values of  $d_{\max}$ , using different embeddings, using the fixed latency matrix.

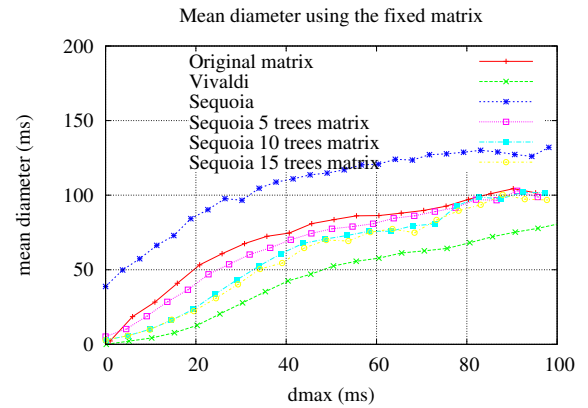


Figure 3. Mean diameter of built bins for BPDC, for different values of  $d_{\max}$ , using different embeddings, using the fixed latency matrix.

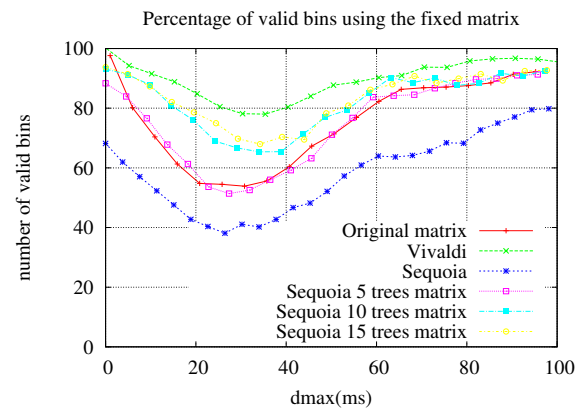


Figure 4. Percentage of valid bins for BPDC, for different values of  $d_{\max}$ , using different embeddings, using the fixed latency matrix.

By contrast, when comparing the results obtained using the two different latency matrix as input for the different Sequoia embeddings, we observe major differences.

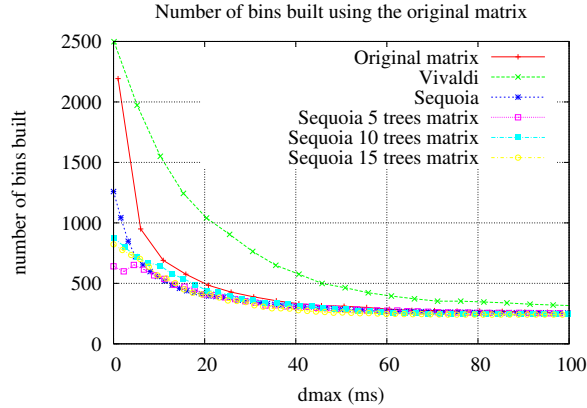


Figure 5. Number of built bins for BPDC, for different values of  $d_{max}$ , using different embeddings, using the original latency matrix.

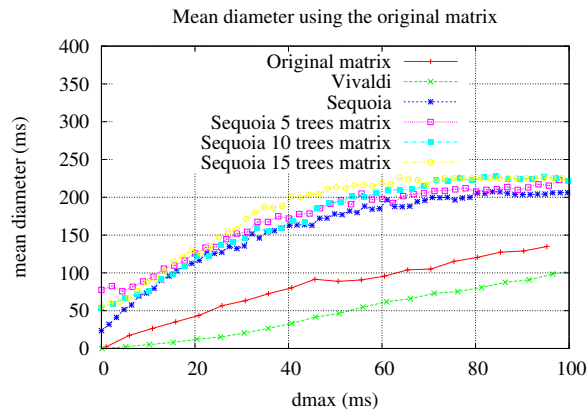


Figure 6. Mean value of the diameter of built bins for BPDC, for different values of  $d_{max}$ , using different embeddings, using the original latency matrix.

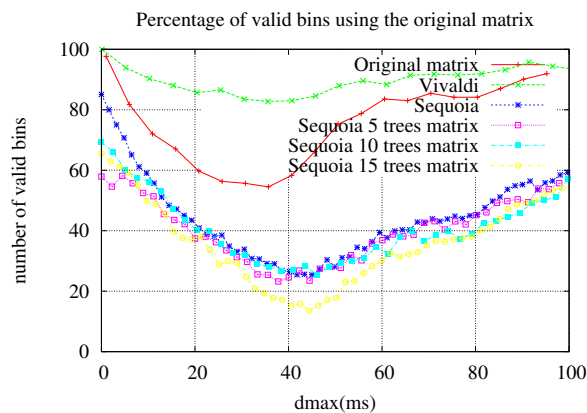


Figure 7. Percentage of valid bins for BPDC, for different values of  $d_{max}$ , using different embeddings, using the original latency matrix.

- All four Sequoia embeddings (using 1, 5, 10 or 15 trees) provide bad performances (in terms of percentage of

valid bins and mean diameter of the bins) when using the original matrix as input,

- As evoked before, the performance of Sequoia-1-tree embedding is still poor when using the fixed matrix,
- The Sequoia-5-trees embedding, when working with the fixed matrix, ensures performance that are comparable to those obtained without using any embedding on the same matrix,
- both Sequoia-10-trees and Sequoia-15-trees embeddings perform better than when no embedding is used, when using the fixed matrix.

To explain the lack of robustness of Sequoia, it is necessary to go into the details of Sequoia algorithm. In fact, in some cases, even just one triangular inequality violation can, in the subtree rooted at the least common ancestor to the three nodes concerned by this violation, induce important distance distortions. Distances between nodes in this subtree can therefore be highly underestimated. This ends in building bins having a diameter in the original matrix violating the imposed distance constraint (thus, those bins are not valid). Therefore, reducing the number of such Triangular Inequality Violations in the original matrix to obtain the fixed matrix can highly improve the performance of Sequoia.

We can conclude from these observations that the joint use of Algorithm 1 and Sequoia, with any number of trees used to embed the network, is not robust in the context of our application. By contrast, the use of Vivaldi embedding, while offering good performances, is a robust solution for BPDC.

## VII. CONCLUSION

In this paper, we have considered an extension the classical Bin Packing problem, where the distance between two elements belonging to the same bin has to be lower than a given threshold. This problem is closely related to several servers and replicas location problems that have been widely studied in the literature. We have provided both inapproximability results and approximation algorithms based on resource augmentation for this problem. Another important contribution of this paper is the comparison, in the specific context of server location problems, of several embedding tools for Internet latencies, on an actual dataset corresponding to PlanetLab nodes. We prove that comparing Vivaldi and Sequoia embeddings on a specific application and using an actual dataset enables to discuss their robustness.

## REFERENCES

- [1] Ittai Abraham, Mahesh Balakrishnan, Fabian Kuhn, Dahlia Malkhi, Venugopalan Ramasubramanian, and Kunal Talwar. Reconstructing approximate tree metrics. In *PODC '07: Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 43–52, New York, NY, USA, 2007. ACM.

- [2] S.F. Assmann, D.S. Johnson, D.J. Kleitman, and J.Y.T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of algorithms*, 5(4):502–525, 1984.
- [3] J. Bar-Ilan, G. Kortsars, and D. Peleg. How to allocate network centers. *J. Algorithms*, pages 15:385–415, 1993.
- [4] O. Beaumont, N. Bonichon, Ph. Duchon, and H. Larchevêque. Distributed approximation algorithm for resource clustering. In Alexander A. Shvartsman and Pascal Felber, editors, *SIROCCO 2008*, volume 5058 of *Lecture Notes in Computer Science*, pages 61–73. Springer, 2008.
- [5] O. Beaumont, N. Bonichon, Ph. Duchon, and H. Larcheveque. Distributed approximation algorithm for resource clustering. In *OPODIS 2008*, volume 5401 of *Lecture Notes in Computer Science*, pages 564–567. Springer, 2008.
- [6] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps and non-approximability-towards tight results. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:422, 1995.
- [7] Fabián A. Chudak and David P. Williamson. Improved approximation algorithms for capacitated facility location problems. In *IPCO*, pages 99–113, 1999.
- [8] Julia Chuzhoy. Covering problems with hard capacities. *SIAM J. Comput.*, 36(2):498–515, 2006.
- [9] B.N. Clark, S. Colbourn David, and J. Charles. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.
- [10] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical, distributed network coordinates. *ACM SIGCOMM Computer Communication Review*, 34(1):113–118, 2004.
- [11] János Csirik and Gerhard J. Woeginger. Resource augmentation for online bounded space bin packing. *J. Algorithms*, 44(2):308–320, 2002.
- [12] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, 2004.
- [13] W. Fernandez de la Vega and G.S. Lueker. Bin packing can be solved within  $1 + \epsilon$  in linear time. *Combinatorica*, pages 1:349–355, 1981.
- [14] Jr. E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. pages 46–93, 1997.
- [15] L. Epstein and A. Levin. On Bin Packing with Conflicts. *SIAM Journal on Optimization*, 19(3):1270–1298, 2008.
- [16] Leah Epstein and Rob van Stee. Online bin packing with resource augmentation. *Discrete Optimization*, 4(3-4):322–333, 2007.
- [17] T.F. Gonzalez. Clustering to minimize the intercluster distance. *Theoretical Computer Science*, pages 38:293–306, 1985.
- [18] Dorit S. Hochbaum and David B. Shmoys. A Best Possible Heuristic for the k-Center Problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.
- [19] M.J. Kao, C.S. Liao, and D.T. Lee. Capacitated domination problem. *Algorithmica*, pages 1–27, 2009.
- [20] S. Khuller and Y.J. Sussmann. The Capacitated K-Center Problem. *SIAM Journal on Discrete Mathematics*, 13:403, 2000.
- [21] M.R. Korupolu, C.G. Plaxton, and R. Rajaraman. Analysis of a Local Search Heuristic for Facility Location Problems. *Journal of Algorithms*, 37:146188, 2000.
- [22] Fabian Kuhn and Thomas Moscibroda. Distributed approximation of capacitated dominating sets. In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 161–170, New York, NY, USA, 2007. ACM.
- [23] E. Lebhar, P. Fraigniaud, and L. Viennot. The inframetric model for the internet. In *Proceedings of the 27th IEEE International Conference on Computer Communications (IN-FOCOM)*, pages 1085–1093, Phoenix, 2008.
- [24] Cristian Lumezanu, Randy Baden, Neil Spring, and Bobby Bhattacharjee. Triangle inequality variations in the internet. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 177–183, New York, NY, USA, 2009. ACM.
- [25] R. Lupton, F.M. Maley, and N. Young. Data Collection for the Sloan Digital Sky Survey—A Network-Flow Heuristic. *Journal of Algorithms*, 27(2):339–356, 1998.
- [26] M. Pál, É. Tardos, and T. Wexler. Facility location with nonuniform hard capacities. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 329, Washington, DC, USA, 2001. IEEE Computer Society.
- [27] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella. On the treeness of internet latency and bandwidth. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 61–72, New York, NY, USA, 2009. ACM.
- [28] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.
- [29] D.B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *In Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1997.
- [30] J.D. Ullman. The performance of a memory allocation algorithm. Technical report 100, Princeton University, Princeton, NJ, 1971.