Oblivious Buy-at-Bulk in Planar Graphs⋆

Srivathsan Srinivasagopalan, Costas Busch, and S. Sitharama Iyengar

Computer Science Department
Louisiana State University
{ssrini1,busch,iyengar}@csc.lsu.edu

Abstract. In the oblivious buy-at-bulk network design problem in a graph, the task is to compute a fixed set of paths for every pair of source-destination in the graph, such that any set of demands can be routed along these paths. The demands could be aggregated at intermediate edges where the fusion-cost is specified by a canonical (non-negative concave) function f. We give a novel algorithm for planar graphs which is oblivious with respect to the demands, and is also oblivious with respect to the fusion function f. The algorithm is deterministic and computes the fixed set of paths in polynomial time, and guarantees a $O(\log n)$ approximation ratio for any set of demands and any canonical fusion function f, where n is the number of nodes. The algorithm is asymptotically optimal, since it is known that this problem cannot be approximated with better than $\Omega(\log n)$ ratio. To our knowledge, this is the first tight analysis for planar graphs, and improves the approximation ratio by a factor of $\log n$ with respect to previously known results.

1   Introduction

A typical client-server model has many clients and multiple servers where a subset of the client set wishes to route a certain amount of data to a subset of the servers at any given time. The set of clients and the servers are assumed to be geographically far apart. To enable communication among them, there needs to be a network of cables deployed. Moreover, the deployment of network cables has to be of minimum cost that also minimizes the communication cost among the various network components. This is what we roughly call as a typical network design problem. The same problem can be easily applied to many similar practical scenarios such as oil/gas pipelines and the Internet.

The "Buy-at-Bulk" network design considers the economies of scale into account. As observed in [5], in a telecommunication network, bandwidth on a link can be purchased in some discrete units $u_1 < u_2 < \cdots < u_j$ with costs $c_1 < c_2 < \cdots < c_j$ respectively. The economies of scale exhibits the property where the cost per bandwidth decreases as the number of units purchased increases: $c_1/u_1 > c_2/u_2 > \ldots c_j/u_j$. This property is the reason why network capacity is bought/sold in "wholesale", or why vendors provide "volume discount".

The generalized form of the buy-at-bulk problem is where there are multiple demands from sources to destinations, and it is commonly referred as *Multi-Sink Buy-at-Bulk* (MSBB). Typically, the demand flows are in discrete units and are unsplittable (indivisible), i.e., the flow follows a single path from the demand node to its destination. These problems are often called "discrete cost network optimization" in operations research.

As mentioned in [6], if information flows from x different sources over a link, then, the cost of total information that is transmitted over that link is proportional to $f(x)$, where $f : \mathbb{Z}^+ \to \mathbb{R}^+$. The function f is called a *canonical fusion function* if it is concave, non-decreasing, $f(0) = 0$ and has the subadditive property $f(x_1 + x_2) \leq f(x_1) + f(x_2)$, $\forall x_1, x_2, (x_1 + x_2) \in \mathbb{Z}^+$. Generally, MSBB problems use the subadditive property to ensure that the 'size' of the aggregated data is smaller than the sum of the sizes of individual data.

We study the *oblivious* buy-at-bulk network design problem (MSBB) with the following constraints: an unknown set of demands and an unknown concave fusion cost function f. An abstraction of this problem can be found in many applications, one of which is data fusion in wireless sensor networks where data from sensors is aggregated over time in multiple sinks. Other application include Transportation & Logistics (railroad, water, oil, gas pipeline construction) etc. Many of these problems are formulated as networks on a plane that can be mapped to planar graphs.

1.1   Problem Statement

Assume that we are given a weighted graph $G = (V, E, \mathfrak{w})$, with edge weights $\mathfrak{w} : E \longrightarrow \mathbb{Z}^+$. We denote $\mathfrak{w}_e$ to be the weight of edge e. Let $d_i = (s_i, t_i)$ be a unit of demand that induces an unsplittable unit of flow from source node $s_i \in V$ to destination node $t_i \in V$. Let $A = \{d_1, d_2, \ldots, d_r\}$ be a set of demands that are routed through paths in G. It is possible that some paths may overlap. The flow of these demands forms a set of paths $P(A) = \{p(d_1), p(d_2), \ldots, p(d_r)\}$.

There is an arbitrary canonical function f at every edge where data aggregates. This f is same for all the edges in G. Let $\varphi_e(A) = \{p(d_i) : e \in p(d_i)\}$ denote the set of paths that use an edge $e \in E$. Then, we define the cost of an edge e to be $C_e(A) = f(|\varphi_e(A)|) \cdot w_e$. The total cost of the set of paths is defined to be $C(A) = \sum_{e \in E} C_e(A)$. For this set A, there is an optimal set of paths $P^*(A)$ with respective cost $C^*(A)$. The approximation ratio for the paths $P(A)$ is defined as $\frac{C(A)}{C^*(A)}$. The MSBB optimization problem on input A is to find a set of paths $P(A)$ that minimizes the approximation ratio. We note that MSBB is NP-Hard as the Steiner tree problem is its special case (when $f(x) = 1$ and when there is only one destination node) [17].

An *oblivious algorithm* $\mathcal{A}_{obl}$ for the MSBB problem, computes a fixed set of paths, denoted $P(G)$ for every pair of source destination nodes in V. Given any set of demands A, the path $p(d_i)$ for each $d_i = (s_i, t_i) \in A$, is the fixed path in $P(G)$ from s_i to t_i. This gives a set of paths $P(A)$ to route the demands A. We define the *approximation ratio* of $\mathcal{A}_{obl}$, as:

$$A.R.(\mathcal{A}_{obl}) = \max_A \frac{C(A)}{C^*(A)}.$$

We aim to find algorithms that minimizes the above approximation ratio for any canonical function f which is unknown to the algorithm. The best known oblivious algorithm is by Gupta *et al.* [11] and provides approximation ratio $O(\log^2 n)$ for general graphs. No better result is known for planar graphs. This problem is NP-hard, since MSBB is NP-hard.

1.2   Contribution

We provide an oblivious algorithm FindPaths for MSBB problems in planar graphs. Our algorithm is deterministic and computes in polynomial time a fixed set of paths that guarantees $O(\log n)$-approximation ratio for any canonical function f (where f is unknown to the algorithm). We also give a lower bound for the approximation ratio for A to be of $\Omega(\log n)$, where n is the number of nodes in the graph. A lower bound of $\Omega(\log n)$ for planar graphs is provided in the context of the online Steiner tree problem by Imase and Waxman [12]. Thus, our bound is tight with respect to planar graphs. It is also a $\log n$ factor improvement over the best previously known result [11].

We build the set of paths based on sparse covers (see [16] for an overview of sparse covers). A γ-cover consists of clusters such that for each node there is some cluster that contains its γ-neighborhood. We construct $O(\log n)$ levels of covers with exponentially increasing locality parameter γ. For every cluster we elect a leader. For any pair of nodes u, v we identify an appropriate common lowest-level cluster that contains both u and v, and the cluster has a respective common leader ℓ. Then the path from u to v is formed by connecting successive path segments emanating from both u and v and using intermediate leaders of lower level clusters until the common leader ℓ is reached.

In the analysis, we introduce the notion of coloring sparse covers, where two clusters that are close receive different color. We show the existence of a sparse cover with constant coloring (based on the sparse covers in [4]). This enables us to obtain optimal approximation at every level. When we combine all the levels, we get an $O(\log n)$ approximation.

1.3   Related Work

Oblivious Network Design. Below, we present the related work on oblivious network design and Table 1 summarizes some results and compares our work with their's. What distinguishes our work with the others' is the fact that we provide a set of paths for the MSBB problem while others provide an overlay tree for the SSBB version.

Goel *et al.* in [6] build an overlay tree on a graph that satisfies triangle-inequality. Their technique is based on maximum matching algorithm that guarantees $(1+\log k)$-approximation, where k is the number of sources. Their solution is oblivious with respect to the fusion cost function f. In a related paper [7], Goel

Table 1. Our results and comparison with previous results for data-fusion schemes. n is the total number of nodes in the topology, k is the total number of source nodes.

Related Work	Algorithm Type	Graph Type	Oblivious Function f	Oblivious Sources	Approx Factor
Lujun Jia *et al.* [14]	Deterministic	Random Deployment	✗	✓	$O(\log n)$
Lujun Jia	Deterministic	Arbitrary Metric	✗	✓	$O(\frac{\log^4 n}{\log\log(n)})$
et al. [13]	Deterministic	Doubling Metric	✗	✓	$O(\log(n))$
Ashish Goel *et al.* [6]	Randomized	General Graph $\triangle$-inequality	✓	✗	$O(\log k)$
Ashish Goel *et al.* [7, 8]	Randomized	General Graph	✓	✗	$O(1)$
Anupam Gupta *et al.* [11]	Randomized	General Graph	✓	✓	$O(\log^2 n)$
	Randomized	Low Doubling	✓	✓	$O(\log n)$
This paper	Deterministic	Planar	✓	✓	$O(\log n)$

et al. construct (in polynomial time) a set of overlay trees from a given general graph such that the expected cost of a tree for any f is within an $O(1)$-factor of the optimum cost for that f. A recent improvement by Goel [8] provides the first constant guarantee on the simultaneous ratio of $O(1)$.

Jia *et al.* in [14] build a Group Independent Spanning Tree Algorithm (GIST) that constructs an overlay tree for randomly deployed nodes in an Euclidean 2 dimensional plane. The tree (that is oblivious to the number of data sources) simultaneously achieves $O(\log n)$-approximate fusion cost and $O(1)$-approximate delay. However, their solution assumes a constant fusion cost function. We summarize and compare the related work in Table 1.

Lujun Jia *et al.* [13] provide approximation algorithms for TSP, Steiner Tree and set cover problems. They present a polynomial-time $(O(\log(n)), O(\log(n)))$-partition scheme for general metric spaces. An improved partition scheme for doubling metric spaces is also presented that incorporates constant dimensional Euclidean spaces and growth-restricted metric spaces. The authors present a polynomial-time algorithm for Universal Steiner Tree (UST) that achieves poly-logarithmic stretch with an approximation guarantee of $O(\log^4 n/\log\log(n))$ for arbitrary metrics and derive a logarithmic stretch, $O(\log(n))$ for any doubling, Euclidean, or growth-restricted metric space over n vertices. They provide a lower bound of $\Omega(\log n/\log\log n)$ for UST that holds even when all the vertices are on a plane.

Gupta *et al.* [11] develop a framework to model *oblivious network design* problems (MSBB) and give algorithms with poly-logarithmic approximation ratio. They develop oblivious algorithms that approximately minimize the total cost of routing with the knowledge of aggregation function, the class of load on each edge and nothing else about the state of the network. Their results show that

if the aggregation function is summation, their algorithm provides a $O(\log^2 n)$ approximation ratio and when the aggregation function is max, the approximation ratio is $O(\log^2 n \log \log n)$. The authors claim to provide a deterministic solution by derandomizing their approach. But, the complexity of this derandomizing process is unclear.

Non-Oblivious Network Design. There has been a lot of research work in the area of approximation algorithms for network design. Since network design problems have several variants with several constraints, only a partial list has been mentioned here. The "single-sink buy-at-bulk" network design (SSBB) problem has a single "destination" node where all the demands from other nodes have to be routed to. Network design problems have been primarily considered in both Operations Research and Computer Science literatures in the context of flows with concave costs. The single-sink variant of the problem was first introduced by Salman *et al.* [17]. They presented an $O(\log n)$-approximation for SSBB in Euclidean graphs by applying the method of Mansour and Peleg [15]. Bartal's tree embeddings [2] can be used to improve their ratio to $O(\log n \log \log n)$. A $O(\log^2 n)$-approximation was given by Awerbuch *et al.* [1] for graphs with general metric spaces. Bartal *et al.* [3] further improved this result to $O(\log n)$. Guha [10] provided the first constant-factor approximation to the problem, whose ratio was estimated to be around 9000 by Talwar [18]. The constant has been further improved in [9].

Organization. In the next section, we present some definitions and notations used throughout the rest of the paper. Section 3 describes the concept of sparse covers. In addition, we derive a coloring for the cover. In section 4, we describe the FindPaths algorithm that builds a set of paths between all pairs of nodes in G. Section 5 provides the analysis of the FindPaths algorithm. Finally, we conclude with a summary and future work in Section 6.

2   Definitions

Consider a weighted graph $G = (V, E, \mathfrak{w})$, where $\mathfrak{w} : E \to \mathbb{Z}^+$. For any path p in G let the length be $\text{length}(p) = \sum_{e \in p} w_e$, that is, the sum of the weights of the edges in p. For any two nodes $u, v \in V$, their *distance* $\text{dist}(u, v)$ is the length of the shortest path that connects the two nodes in G. The *diameter* D is the length of the longest shortest path in G. The radius of a node v is $\text{rad}(v) = \max_{u \in V}(\text{dist}(v, u))$. The *radius* of G is defined as $\text{rad}(G) = \min_v(\text{rad}(v))$. We denote by $N_k(v)$ the *k-neighborhood* of v which is the set of nodes distance at most k from v. For any set of nodes $S \subseteq V$, we denote by $N_k(S)$ the k-neighborhood of S which contains all nodes which are within distance k from any node in S.

A set of nodes $X \subseteq V$ is called a *cluster* if the induced subgraph $G(X)$ is connected. Let $Z = \{X_1, X_2, \ldots, X_k\}$ be a set of clusters in G. For every node $v \in G$, let $Z(v) \subseteq Z$ denote the set of clusters that contain v. The *degree* of v in Z is defined as $\beta_v(Z) = |Z(v)|$, which is the number of clusters that contain v. The degree of Z is defined as $\beta(Z) = \max_{v \in V} \beta_v(Z)$, which is largest degree

of any of its nodes. The radius of Z is defined as $\mathrm{rad}(Z) = max_{X \in Z}(\mathrm{rad}(X))$, where $\mathrm{rad}(X)$ is

Consider a *locality parameter* $\gamma > 0$. A set of clusters Z is said to γ-*satisfy* a node v in G, if there is a cluster $X \in Z$, such that the γ-neighborhood of v, $N_\gamma(v)$, (nodes within distance γ from v) is included in X, that is, $N_\gamma(v) \subseteq X$. A set of clusters Z is said to be a γ-*cover* for G, if every node of G is γ-satisfied by Z in G. The *stretch* $\sigma(Z)$ of a γ-cover Z is the smallest number such that $\mathrm{rad}(Z) = \sigma(Z) \cdot \gamma$.

We define the following coloring problem in a set of clusters Z. We first define the notion of the distance between two clusters $X_i, X_j \in Z$, $X_i \neq X_j$. We say that $\mathrm{dist}(X_i, X_j) \leq k$, if there is a pair of nodes $u \in X_i$ and $v \in X_j$ such that u is k-satisfied in X_i, v is k-satisfied in X_j, and $\mathrm{dist}(u,v) \leq k$. A valid distance-$k$ coloring of Z with a palette of χ colors $[1, \chi]$, is an assignment of an integer $\mathrm{color}(X) \in [1, \chi]$ to every $X \in Z$, such that there is no pair of clusters $X_i, X_j \in Z$, $X_i \neq X_j$, with $\mathrm{dist}(X_i, X_j) \leq k$ which receive the same color. The objective is to find the smallest χ that permits a valid distance-k coloring.

3 Sparse Cover

A γ-cover is *sparse* if it has small degree and stretch. In [4, Section 5] the authors present a polynomial time sparse cover construction algorithm Planar-Cover(G, γ) for any planar graph G and locality parameter γ, which finds a γ-cover Z with constant degree, $\beta \leq 18$, and constant stretch, $\sigma \leq 24$. Here, we show that this cover also admits a valid distance-γ coloring with a constant number of colors $\chi \leq 18$.

For any node $v \in G$, we denote by $\mathrm{depth}_v(G)$ the shortest distance between v and an external node (in the external face) of G. We also define $\mathrm{depth}(G) = max_{v \in V} \mathrm{depth}_v(G)$. The heart of sparse cover algorithm in [4, Section 5] concerns the case where $\mathrm{depth}(G) \leq \gamma$ which is handled in Algorithm Depth-Cover(G, γ). The general case, $\mathrm{depth}(G) > \gamma$, is handled by dividing the graph into zones of depth $O(\gamma)$, as we discuss later. So, assume for now that $\mathrm{depth}(G) \leq \gamma$.

The Algorithm Depth-Cover(G, γ), relies on forming clusters along shortest paths connecting external nodes (in the external face) of G. For every shortest path p, Algorithm Shortest-Path-Cluster$(G, p, 4\gamma)$ in [4, Section 3] returns a set of clusters around the 4γ neighborhood of p with radius at most 8γ and degree 3. Then, p and all its 2γ-neighborhood is removed from G producing a smaller subgraph G' (with possibly multiple connected components). The algorithm proceeds recursively on each connected component H of G' by selecting an appropriate new shortest path p' between external nodes of H. The algorithm terminates when all the nodes have been removed. The initial shortest path that starts the algorithm consists of a single external node in G. The resulting γ-cover Z consists of the union of all the clusters from all the shortest paths. The shortest paths are chosen in such a way that a node participates in the clustering process of at most 2 paths, and this bounds the degree of the γ-cover to be at most $\beta \leq 6$, and stretch $s \leq 8$.

The analysis in [4, Section 5.1.1] of Algorithm Depth-Cover relies on representing the clustering process of G as a tree T as we outline here. Each tree node $w \in T$ represents a pair $w = (G(w), p(w))$ where $G(w)$ is a planar subgraph of G that is to be clustered, and $p(w)$ is a shortest path between two external nodes of $G(w)$. The root of the tree is $r = (G, v)$, where v is a trivial initial path with one external node $v \in G$. The children of a node $w \in T$ are all the nodes $w' = (G(w'), p(w'))$, such that $G(w')$ is a connected component that results from removing $p(w)$ and its 2γ-neighborhood from G.

Next, we extend [4, Lemma 3.1] to show that we can color the clusters obtained by a shortest path clustering using a constant number of colors.

Lemma 1. *For any graph G, shortest path $p \in G$, the set of clusters returned by Algorithm* Shortest-Path-Cluster$(G, p, 4\gamma)$ *admits a valid distance-γ coloring with 3 colors.*

We can obtain a coloring of Z by coloring the respective levels of the tree T. Assume that the root is at level 0.

Lemma 2. *The union of clusters in any level $i \geq 0$ of the tree T, admits a valid distance-γ coloring with 3 colors.*

Lemma 3. *Algorithm* Depth-Cover(G, γ) *returns a set of clusters Z which admits a valid distance-γ coloring with 6 colors.*

We are now ready to consider the case depth$(G) > \gamma$. Algorithm Planar-Cover (G, γ) decomposes the graph into a sequence of bands, such that each band W_i has depth γ. The bands are organized into zones, such that zone S_i consists of three consecutive bands W_{i-1}, W_i, W_{i+1}. Thus, zone S_i overlaps with bands $S_{i-2}, S_{i-1}, S_{i+1}$ and S_{i+2}. The algorithm invokes Depth-Cover$(S_i, 3\gamma-1)$ for each zone giving a γ-cover Z with degree $\beta \leq 3 \cdot 6 = 18$ and stretch $\sigma \leq 3 \cdot 8 = 24$.

We can obtain the following coloring result. Using Lemma 3, for every zone S_i we can get a valid distance-$(3\gamma - 1)$ coloring with a palette of 6 colors. This implies that we can obtain a valid distance-γ coloring for the zone with at most 6 colors. Zones S_i and S_{i+3} do not overlap and any two nodes satisfied in them (one from each zone) with respect to G must be more than γ distance apart. Therefore, we can color all the zones with three different palettes each consisting of 6 colors, so that zone S_i, uses the $((i \mod 3) + 1)$th palette. The coloring can be found in polynomial time. Therefore, we obtain:

Theorem 1. *Algorithm* Planar-Cover(G, γ) *produces a set of clusters Z which has degree $\beta = 18$, stretch $\sigma \leq 24$, and admits a valid distance-γ coloring with $\chi = 18$ colors.*

4   Algorithm

We describe how to find paths between each pair of nodes in graph $G = (V, E, \mathfrak{w})$ to route demands. To find such paths, we use Algorithm FindPaths (Algorithm 2) which relies on Algorithm AuxiliaryPaths (Algorithm 1).

Algorithm 1. AuxiliaryPaths(G)

Input: Graph $G = (V, E, \mathfrak{w})$
Output: Set of auxiliary paths for all nodes in G

```
// Z_i is a γ_i-cover of G, 0 ≤ i ≤ κ (γ_i, κ specified in Section 5)
// Assume each cluster has a designated leader node
```

$\mathcal{Q} \leftarrow \emptyset$;   `// set of auxiliary paths for all nodes in G`
foreach $v \in V$ **do**
$\quad q(v) \leftarrow \emptyset$;   `// auxiliary path for node v from level 0 to κ`
$\quad x \leftarrow v$;
$\quad$ **for** $i = 0$ **to** $\kappa - 1$ **do**
$\quad\quad$ Let $X \in Z_{i+1}$ be a cluster that γ_{i+1}-satisfies v;
$\quad\quad \ell_{i+1}(v) \leftarrow$ leader of X;
$\quad\quad q_i(v) \leftarrow$ shortest path from x to $\ell_{i+1}(v)$;
$\quad\quad q(v) \leftarrow$ concatenate $q(v)$ and $q_i(v)$;
$\quad\quad x \leftarrow \ell_{i+1}(v)$;
$\quad \mathcal{Q} \leftarrow \mathcal{Q} \cup q(v)$;
return $\mathcal{Q}$;

Algorithm 2. FindPaths(G)

Input: Graph $G = (V, E, \mathfrak{w})$.
Output: Set of paths between all pair of nodes in G

```
// Let Z_i, ℓ_i(x), and q(x), be as in Algorithm AuxiliaryPaths; ℓ_0(x) ← x
// Let q̄_i(x) be the auxiliary path segment from x to ℓ_i(x); q̄_0(x) ← x
```

$\mathcal{P} \leftarrow \emptyset$;   `// set of paths for all pairs of nodes in G`
foreach *pair* $u, v \in V$ **do**
$\quad$ Let γ_i be the smallest locality parameter such that $\gamma_i \geq 2 \cdot \text{dist}(u, v)$;
$\quad$ Let $X \in Z_i$ be a cluster that γ_i-satisfies u (and hence $\gamma_i/2$-satisfies v);
$\quad$ Let ℓ' be the leader of X (*common leader* of u, v);
$\quad q' \leftarrow$ concatenate shortest paths from $\ell_{i-1}(u)$ to ℓ' to $\ell_{i-1}(v)$;
$\quad p(u, v) \leftarrow$ concatenate $\overline{q}_{i-1}(u)$, q', and $\overline{q}_{i-1}(v)$;
$\quad \mathcal{P} \leftarrow \mathcal{P} \cup p(u, v)$;
return $\mathcal{P}$;

Both algorithms use $\kappa + 1$ covers $Z_0, \ldots, Z_\kappa$, where in Z_0 every node in V is a cluster, and Z_i is a γ_i-cover of G, for $i \geq 1$, where the parameters γ_i and κ are defined in Section 5. We refer to the cover Z_i as the level i cover of G. We assume that each cluster in the covers has a designated leader node. There is a unique cluster, containing all nodes in G, and leader node ℓ_κ at level κ.

Algorithm AuxiliaryPaths computes an *auxiliary path* $q(v)$ from every node $v \in V$ to ℓ_κ. The auxiliary paths are built in a bottom-up fashion. An auxiliary path from any node $v \in V$ at level 0 is built recursively. In the basis of the recursion, we identify a cluster $X_1 \in Z_1$, which γ_1-satisfies node v. Let $\ell_1(v)$ denote the leader X_1. We now compute a shortest path, denoted $q_0(v)$, from v to $\ell_1(v)$. This forms the first path segment of $q(v)$. Suppose we have computed $q(v)$ up to level i, $i < \kappa$. We now want to extend this path to the next higher

level $i+1$. To compute the path segment from level i to level $i+1$, we repeat the process of finding a cluster $X_{i+1} \in Z_{i+1}$ that γ_{i+1}-satisfies node v. Let $\ell_{i+1}(v)$ denote the leader X_1. We compute the shortest path, denoted $q_i(v)$ from $\ell_i(v)$ to $\ell_{i+1}(v)$. We then append this new path segment $q_i(v)$ to $q(v)$ to form the current extended path $q(v)$. The path building process terminates when the last leader reaches level κ.

We are now ready to describe how Algorithm FindPaths computes the shortest paths between all pair of nodes in G. For a pair of nodes $u, v \in V$, let y be the distance between them. Let γ_i be the smallest locality parameter such that $\gamma_i \geq 2y$. Let $X \in Z_i$ be the cluster that γ_i satisfies u, and let ℓ' be the respective leader of X. Note that by the way that we have chosen γ_i, cluster X also $\gamma_i/2$-satisfies v. Let $\overline{q}_i(u)$ denote the segment of the auxiliary path $q(u)$ from u to $\ell_i(u)$. We concatenate $\overline{q}_{i-1}(u)$, with a shortest path from $\ell_{i-1}(u)$ to ℓ', with a shortest path from ℓ' to $\ell_{i-1}(v)$, and $\overline{q}_{i-1}(v)$. This gives the path $p(u, v)$.

5   Analysis

Let $G = (V, E, \mathfrak{w})$ be a planar graph with n nodes. In this section we use the following parameters:

$$
\begin{aligned}
&\kappa = 1 + \lceil \log_{4\sigma} D \rceil && //\text{highest cluster level in } G \\
&\beta = 18 && //\text{cover degree bound} \\
&\sigma = 24 && //\text{cover stretch bound} \\
&\gamma_i = (4\sigma)^{i-1} && //\text{locality parameter of level } i \geq 1 \text{ cover} \\
&\chi = 18 && //\text{coloring of each level } i
\end{aligned}
$$

Consider $\kappa + 1$ levels of covers $Z_0, \ldots, Z_{\kappa+1}$, where in Z_0 each node in V is a cluster, and each Z_i, $i \geq 1$, is a γ_i-cover of G which is obtained from Theorem 1. Thus, each Z_i, $i \geq 1$, has degree at most β, stretch at most σ, and can be given a valid distance-γ_i coloring with χ colors.

Let A denote an arbitrary set of demands. For any demand $d = (s, t) \in A$ let $p(d) = p(s, t)$ be the path given by Algorithm FindPaths. Suppose that the common leader of s and t is ℓ. The path $p(d)$ consists of two path segments: the *source path segment* $p(s)$, from s to ℓ, and the *destination path segment* $p(t)$ from ℓ to t. We denote by $p_i(s)$ the subpath between level i and level $i+1$ (we call this the level i subpath).

Let $C^*(A)$ denote the cost of optimal paths in A. Let $C(A)$ denote the cost of the paths given by our algorithm. We will bound the competitive ratio $C(A)/C^*(A)$. For simplicity, in the approximation analysis, we consider only the cost of the source path segments $p(s_i)$. When we consider the destination segments the approximation ratio increases by a factor of 2.

The cost $C(A)$ can be bounded as a summation of costs from the different levels as follows. For any edge e let $\varphi_{e,i}(A) = \{p_i(s) : ((s,t) \in A) \wedge (e \in p_i(v))\}$ be the set of layer-i subpaths that use edge e. Denote by $C_{e,i}(A) = f(|\varphi_{e,i}(A)|) \cdot w_e$ the cost on the edge e incurred by the level-i subpaths. Since f is subadditive, we get $C_e(A) \leq \sum_{i=0}^{\kappa-1} C_{e,i}(A)$. Let $C_i(A) = \sum_{e \in E} C_{e,i}(A)$ denote the cost incurred by the layer-i subpaths. Since $C(A) = \sum_{e \in E} C_e(A)$, we have that:

$$C(A) \leq \sum_{i=0}^{\kappa-1} C_i(A). \tag{1}$$

For any cluster X let $X(A)$ denote the set of demands with source in X whose paths leave from the leader of X toward the leader of a higher level cluster.

Lemma 4. *For any Z_i, $2 \leq i \leq \kappa - 1$, $C^*(A) \geq R(i)/\chi$, where $R(i) = \sum_{X \in Z_i} f(|X(A)|) \cdot \gamma_i/2$.*

Proof. Let $Z_i(k)$ to be the set of clusters at level i which receive color $k \in [1, \chi]$. Consider a cluster $X \in Z_i(k)$. Consider a demand $(s, t) \in X(A)$. Since $X \in Z_i(k)$ the common leader of s and t is at a level $i + 1$ or higher. From the algorithm, $\text{dist}(s, t) \geq \gamma_{i+1}/2$. Consider the subpaths from $X(A)$ of length up to $\gamma_i/2$. In the best case, these subpaths from $X(A)$ may be combined to produce a path with smallest possible total cost $f(|X(A)|) \cdot \gamma_i/2$. Any two nodes $u \in X(A)$ and $v \in Y(A)$, where $X, Y \in Z_i(k)$ and $X \neq Y$, have $\text{dist}(u, v) > \gamma_i$, since each node is γ_i-satisfied in its respective cluster and X and Y receive the same color in the distance-γ_i coloring of Z. Therefore, the subpaths of lengths up to $\gamma_i/2$ from the demands $X(A)$ and $Y(A)$ cannot combine. Consequently, $C^*(A) \geq R(i, k)$ where $R(i, k) = \sum_{X \in Z_i(k)} f(|X(A)|) \cdot \gamma_i/2$. Let $R_{\max} = \max_{k \in [1, \chi]} R(i, k)$. We have that $C^*(A) \geq R_{\max}$. Since $R(i) = \sum_{k=1}^{\chi} R(i, k) \leq R_{\max} \cdot \chi$. We obtain $C^*(A) \geq R(i)/\chi$, as needed.

We also get the following trivial lower bound for the special case where $0 \leq i \leq 1$, which follows directly from the observation that each demand needs to form a path with length at least 1.

Lemma 5. *For any Z_i, $0 \leq i \leq 1$, $C^*(A) \geq \sum_{X \in Z_i} f(|X(A)|)$.*

We obtain the following upper bound.

Lemma 6. *For any Z_i, $0 \leq i \leq \kappa - 1$, $C_i(A) \leq Q(i)$ where $Q(i) = \sum_{X \in Z_i} f(|X(A)|) \cdot \beta \sigma \gamma_{i+1}$.*

Proof. For any cluster $X \in Z_i$, we can partition the demands $X(A) = Y_1 \cup Y_2 \cup \ldots \cup Y_k$, where $Y_i \neq Y_j$, $i \neq j$, according to the leaders at level $i + 1$ that they use, so that all demands in Y_i use the same leader in Z_{i+1}, and Y_i and Y_j use a different leader of Z_{i+1}. Next, we provide a bound on k.

Consider any two demands $d_1 = (s_1, t_1) \in X(A)$ and $d_2 = (s_2, t_2) \in X(A)$. Let ℓ_i be the leader of X. Since s_1 and s_2 are γ_i-satisfied by the cluster X of ℓ_i, they are both members of that cluster. Therefore, $\text{dist}(s_1, \ell_i) \leq \sigma\gamma_i$, and $\text{dist}(s_2, \ell_i) \leq \sigma\gamma_i$. Thus, $\text{dist}(s_1, s_2) \leq 2\sigma\gamma_i = \gamma_{i+1}/2$. Suppose that demand d_1 chooses leader ℓ_{i+1} at level $i + 1$ with respective cluster X_{i+1}. Since s_1 is at least $\gamma_{i+1}/2$-satisfied in X_{i+1}, s_2 is a member of X_{i+1}. Since any node is a member of at most β clusters at level $i + 1$, it has to be that the number of different level $i + 1$ leaders at level $i + 1$ that the demands in $X(A)$ select is bounded by β. Consequently, $k \leq \beta$.

Since f is subadditive and for any demand (s, t), $|p_i(s)| \leq \sigma\gamma_{i+1}$, $C_i(Y_j) \leq f(|Y_j|) \cdot \sigma\gamma_{i+1}$. Therefore, $C_i(X(A)) \leq \sum_{j=1}^{k} C_i(Y_j) \leq f(|X(A)|) \cdot \beta\sigma\gamma_{i+1}$. Which gives: $C_i(A) \leq \sum_{X \in Z_i} f(|X(A)|) \cdot \beta\sigma\gamma_{i+1}$, as needed.

Lemma 7. *For any $0 \leq i \leq \kappa - 1$, $C_i(A) \leq C^*(A) \cdot 8\beta\sigma^2\chi$.*

Proof. From Lemma 6, for any $0 \leq i \leq \kappa - 1$, $C_i(A) \leq Q(i)$. From Lemma 4, for any $2 \leq i \leq \kappa - 1$, $C^*(A) \geq R(i)/\chi$. Note that $Q(i) = R(i) \cdot 2\beta\sigma\chi\gamma_{i+1}/\gamma_i = R(i) \cdot 8\beta\sigma^2\chi$. Therefore, $C_i(A) \leq C^*(A) \cdot 8\beta\sigma^2\chi$. For $0 \leq i \leq 1$, we use the lower bound of Lemma 5, and we obtain $C_i(A) \leq C^*(A) \cdot \beta\sigma\gamma_2 = C^*(A) \cdot 4\beta\sigma^2$.

We now give the central result of the analysis:

Theorem 2. *The oblivious approximation ratio of the algorithm is $O(\log n)$.*

Proof. (Sketch.) Since the demand set A is arbitrary, from Lemma 7 and Equation 1 we obtain oblivious approximation ratio bounded by $8\kappa\beta\sigma^2\chi$. When we take into account the source path segments together with the destination path segments, the approximation ratio bound increases by a factor of 2, and it becomes $16\kappa\beta\sigma^2\chi$. Since, β, σ, χ, are constants and $\kappa = O(\log D)$, we obtain approximation ratio $O(\log D)$. With a more fine-tuned analysis where we separate the demands into ranges according to the distance between sources and destinations, we can obtain approximation ratio $O(\log n)$.

6 Conclusions

We provide a set of paths for the multi-sink buy-at-bulk network design problem in planar graphs. Contrary to many related work where the source-destination pairs were already given, or when the source-set was given, we assumed the obliviousness of the set of source-destination pairs. Moreover, we considered an unknown fusion cost function at every edge of the graph. We presented nontrivial upper and lower bounds for the cost of the set of paths. We have demonstrated that a simple, deterministic, polynomial-time algorithm based on sparse covers can provide a set of paths between all pairs of nodes in G that can accommodate any set of demands. We have shown that this algorithm guarantees $O(\log n)$-approximation. As part of our future work, we are looking into obtaining efficient solutions to other related network topologies, such as minor-free graphs.

References

[1] Awerbuch, B., Azar, Y.: Buy-at-bulk network design. In: FOCS 1997, Washington, DC, USA, p. 542. IEEE Computer Society, Los Alamitos (1997)
[2] Bartal, Y.: Competitive analysis of distributed online problems - distributed paging. Ph.D. Dissertation (1994)
[3] Bartal, Y.: On approximating arbitrary metrices by tree metrics. In: STOC 1998, pp. 161–168. ACM, New York (1998)

[4] Busch, C., LaFortune, R., Tirthapura, S.: Improved sparse covers for graphs excluding a fixed minor. In: PODC 2007: Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing, pp. 61–70. ACM, New York (2007)

[5] Chekuri, C., Hajiaghayi, M.T., Kortsarz, G., Salavatipour, M.R.: Approximation algorithms for non-uniform buy-at-bulk network design. In: FOCS 2006, Washington, DC, USA, pp. 677–686 (2006)

[6] Goel, A., Estrin, D.: Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In: SODA 2003, pp. 499–505. SIAM, Philadelphia (2003)

[7] Goel, A., Post, I.: An oblivious o(1)-approximation for single source buy-at-bulk. In: Annual IEEE Symposium on Foundations of Computer Science, pp. 442–450 (2009)

[8] Goel, A., Post, I.: One tree suffices: A simultaneous o(1)-approximation for single-sink buy-at-bulk. In: Annual IEEE Symposium on Foundations of Computer Science (2010)

[9] Grandoni, F., Rothvoß, T.: Network design via core detouring for problems without a core. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 490–502. Springer, Heidelberg (2010)

[10] Guha, S., Meyerson, A., Munagala, K.: A constant factor approximation for the single sink edge installation problems. In: STOC 2001, pp. 383–388. ACM, New York (2001)

[11] Gupta, A., Hajiaghayi, M.T., Räcke, H.: Oblivious network design. In: SODA 2006: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, New York, NY, USA, pp. 970–979 (2006)

[12] Imase, M., Waxman, B.M.: Dynamic steiner tree problem. SIAM Journal on Discrete Mathematics 4(3), 369–384 (1991)

[13] Jia, L., Lin, G., Noubir, G., Rajaraman, R., Sundaram, R.: Universal approximations for tsp, steiner tree, and set cover. In: STOC 2005: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, pp. 386–395. ACM, New York (2005)

[14] Jia, L., Noubir, G., Rajaraman, R., Sundaram, R.: GIST: Group-independent spanning tree for data aggregation in dense sensor networks. In: Gibbons, P.B., Abdelzaher, T., Aspnes, J., Rao, R. (eds.) DCOSS 2006. LNCS, vol. 4026, pp. 282–304. Springer, Heidelberg (2006)

[15] Mansour, Y., Peleg, D.: An approximation algorithm for minimum-cost network design. Technical report, Jerusalem, Israel, Israel (1994)

[16] Peleg, D.: Distributed computing: a locality-sensitive approach. SIAM, Philadelphia (2000)

[17] Salman, F.S., Cheriyan, J., Ravi, R., Subramanian, S.: Approximating the single-sink link-installation problem in network design. SIAM J. on Optimization 11(3), 595–610 (2000)

[18] Talwar, K.: The single-sink buy-at-bulk LP has constant integrality gap. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 475–480. Springer, Heidelberg (2002)