Algorithmica (2015) 72:1–43
DOI 10.1007/s00453-013-9844-6

# Capacitated Domination: Problem Complexity and Approximation Algorithms

**Mong-Jen Kao · Han-Lin Chen · D.T. Lee**

Received: 20 August 2012 / Accepted: 13 October 2013 / Published online: 12 November 2013


**Abstract** We consider a local service-requirement assignment problem named *capacitated domination* from an algorithmic point of view. In this problem, we are given a graph with three parameters defined on each vertex, which are the cost, the capacity, and the demand, of a vertex, respectively. A vertex can be chosen multiple times in order to generate sufficient capacity for the demands of the vertices in its closed neighborhood. The objective of this problem is to compute a demand assignment of minimum cost such that the demand of each vertex is fully-served by some of its closed neighbors without exceeding the amount of capacity they provide.

In this paper, we provide complexity results as well as several approximation algorithms to compose a comprehensive study for this problem. First, we provide logarithmic approximations for general graphs which are asymptotically optimal. From the perspective of parameterized complexity, we show that this problem is $W[1]$-hard

Extended abstracts of this work appeared in the 4th Frontiers of Algorithmics Workshop (FAW'10), Wuhan, China (received the best student paper award) [31] and the 22nd International Symposium on Algorithms and Computation (ISAAC'11), Yokohama, Japan [32].

Part of this work was done when the author M.-J. Kao was with Karlsruhe Institute of Technology (KIT), Germany, as a visiting student.

M.-J. Kao (✉) · D.T. Lee
Institute of Information Science, Academia Sinica, Taipei, Taiwan
e-mail: mong@iis.sinica.edu.tw

D.T. Lee
e-mail: dtlee@nchu.edu.tw

H.-L. Chen
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

*Present address:*
D.T. Lee
Department of Computer Science and Engineering, National Chung-Hsing University, Taichung, Taiwan

with respect to treewidth and solution size. Moreover, we show that this problem is fixed-parameter tractable with respect to treewidth and the maximum capacity of the vertices. The latter result implies a pseudo-polynomial time approximation scheme for planar graphs under a standard framework.

In order to drop the pseudo-polynomial factor, we develop a constant-factor approximation for planar graphs, based on a new perspective which we call *general ladders* on the hierarchical structure of outer-planar graphs. We believe that the approach we use can be applicable to other capacitated covering problems.

**Keywords** Capacitated domination · Approximation algorithms · Dominating set · Graph theory

## 1 Introduction

Resource management is one of the most important issues to be addressed in many areas. One natural way to describe this scenario is to view resource management as a process of distributing resources to demanding targets. In some cases, there exist locality constraints between the supply and the demand, and it only makes sense to assign the resources from supplying source to demanding objects that lie in the neighborhood. Typical examples include communication in wireless ad-hoc networks, in which a peer can only be reached after entering the area it covers. This scenario is depicted by the well-known and extensively studied *dominating set* problem. In the dominating set problem, we are given a graph $G = (V, E)$ and the objective is to compute a vertex subset $S$ of minimum cardinality such that for each vertex $v \in V$, either $v$ belongs to $S$ or $v$ has a neighbor that does.

In practice, the service-requirement relation between two vertices is often more complicated than a binary relation. To be more precise, the demand of each vertex is more realistic when it is not simply assumed to be zero or one. The amount of supply each vertex can provide is often not unit or infinity. The cost of choosing a vertex into the subset, i.e., the activation cost of vertices, is often not identical. Therefore it is natural to take these quantities as part of the input instead of assuming them to be simple constants. When we are allowed to activate the vertices multiple times in exchange of more supply, the scenario is said to be *capacitated* [22].

A series of study on capacitated covering problems was initiated by Guha et al. [22], which addressed the vertex cover problem under the concept of capacitation, i.e., vertices are allowed to be chosen multiple times for more supply. Several follow-up papers have appeared since then, studying both this topic and related variations [10, 19, 20]. These problems are also closely related to work on the capacitated facility location problem, which has drawn a lot of attention since 1990s. See [9, 38, 40].

Motivated by the study on capacitated vertex covering and a more general local service-requirement assignment scenario, Kao et al. [31–33] considered a generalization of the dominating set problem called *capacitated domination*. In this problem, the input graph $G = (V, E)$ is given along with three non-negative parameters defined on the set of vertices, which are referred to as the *cost* $w\colon V \to \mathbb{R}^+ \cup \{0\}$, the

*capacity* $c\colon V \to \mathbb{R}^+ \cup \{0\}$, and the *demand* $d\colon V \to \mathbb{R}^+ \cup \{0\}$, respectively. The demand of a vertex stands for the amount of service it requires from its adjacent vertices, including the vertex itself. The capacity of a vertex represents the amount of service each multiplicity (copy) of that vertex can provide, and the cost of a vertex is what it requires to choose (activate) each multiplicity (copy) of that vertex.

In order to make the definition clear and to prevent confusion, we remark that in this problem setting, a vertex is allowed to be chosen multiple times in order to provide more service to its neighboring vertices, including itself.

By a demand assignment function $f$ we mean a function which maps pairs of vertices to non-negative real numbers. Intuitively, $f(u, v)$ denotes the amount of demand of $u$ that is served by $v$. Throughout this paper, we also use the phrase "assigned to" alternatively with "served by" to indicate the demand assignment of a vertex for the ease of presentation. We denote by $N_G(v)$ the set of neighbors of a vertex $v \in V$ and by $N_G[v] = N_G(v) \cup \{v\}$ the set of closed neighbors of $v$.

**Definition 1** (Feasible demand assignment function) A demand assignment function $f$ is said to be feasible if for each $u \in V$ we have

$$\sum_{v \in N_G[u]} f(u, v) \geq d(u).$$

That is, the demand assignment function is feasible if the demand of each vertex is fully-assigned to its closed neighbors.

For convenience, we say that a vertex $u$ is *dominated* or *served*, following the classical terminology, when the demand of $u$ is fully assigned to its closed neighbors, i.e., $\sum_{v \in N_G[u]} f(u, v) \geq d(u)$.

Given a demand assignment function $f$, the corresponding capacitated dominating multi-set $\mathcal{D}(f)$ is defined as follows. For each vertex $v \in V$, the *multiplicity* of $v$ in $\mathcal{D}(f)$ is defined to be

$$x_f(v) = \left\lceil \frac{\sum_{u \in N_G[v]} f(u, v)}{c(v)} \right\rceil.$$

The cost of the assignment function $f$, denoted $w(f)$, is defined as

$$w(f) = \sum_{v \in V} w(v) \cdot x_f(v).$$

**Definition 2** (Capacitated domination problem) Given a graph $G = (V, E)$ with cost, capacity, and demand defined on each vertex, the capacitated domination problem asks for a feasible demand assignment function $f$ such that $w(f)$ is minimized.

Depending on the way how the demand is assigned, there are different models to consider. In the *inseparable* demand model we require that $f(u, v)$ is either 0 or $d(u)$, for each edge $(u, v) \in E$, while in the *separable* demand model we do not have such constraints. Intuitively, in the inseparable demand model we require that the demand of a vertex must be fully served merely by one of its closed neighbors.

For the capacitated domination problem, Kao et al. [33], presented a $(\Delta + 1)$-approximation for general graphs with separable demands, where $\Delta$ is the maximum vertex degree of the graph. For trees with inseparable demands, they provided a linear-time algorithm that computes an exact solution. For trees with separable demands, they showed that this problem is NP-hard followed by presenting a fully polynomial-time approximation scheme.

Dom et al. [13] considered a variation of this problem where the demand of each vertex is unit and inseparable, and the number of multiplicities available at each vertex is limited by one. For this special case, they proved that this problem is $W[1]$-hard when parameterized by treewidth and solution size. Cygan et al. [11] considered the same case and presented an exact algorithm with running time $O(1.89^n)$. The running time of this algorithm was further improved by Liedloff et al. [36] to $O(1.8463^n)$.

### 1.1 Related Work

The dominating set problem has been one of the most fundamental and well-known problems in both graph theory and combinatorial optimization for decades. As this problem is known to be NP-hard, approximation algorithms have been proposed in the literature [3, 27, 29]. On the one hand, greedy algorithms are shown to achieve a guaranteed ratio of $\ln n$ [29], where $n$ is the number of vertices. The ratio is later proven to be tight by Feige [15]. On the other hand, algorithms based on dual-fitting provide a guaranteed ratio of $\Delta$ [27], where $\Delta$ is the maximum degree of the vertices in the graph. A polynomial-time approximation scheme for planar graphs was given by Baker [3].

In terms of parameterized complexity, dominating set has its special place as well. In contrast to the *vertex cover* problem, which is fixed-parameter tractable (FPT) with respect to solution size, i.e., can be solved in $f(k) \cdot n^{O(1)}$ time where $f(k)$ is a computable function of the solution size $k$ and $n$ is the number of vertices, dominating set is proven to be $W[2]$-complete, e.g., in [17, 37], in the sense that no fixed-parameter tractable algorithm exists (with respect to solution size) unless FPT $= W[2]$. Although dominating set is a fundamentally hard problem in the parameterized $W$-hierarchy, it has been used as a benchmark problem for sub-exponential time parameterized algorithms [1, 12, 18] and linear size kernels have been obtained for planar graphs [2, 8, 23, 24, 41], and more generally, for graphs that exclude a fixed graph $H$ as a minor [12].

In addition, a considerable amount of work has been done in the literature, which considers possible variations from purely theoretical aspects to practical applications. See [25, 39] for a detailed survey. In particular, variations of dominating set occur in numerous practical settings, ranging from strategic decisions, such as locating radar stations or emergency services, to computational biology and to voting systems. For example, Haynes et al. [26] considered the *power domination problem* in electricity networks [26, 35] while Wan et al. [43] considered the *connected domination problem* in wireless ad-hoc networks.

### 1.2 Our Contributions

The goal of this paper is to provide an overview on the problem complexity of the capacitated domination problem with respect to different graph classes. In particu-

lar, we present new approximation algorithms as well as complexity results for this problem. A table of summary on the results presented in this paper is also provided in Table 1.

First, we present logarithmic approximations with respect to both separable and inseparable demand models on general graphs. This is achieved by properly determining a strategy that greedily selects a vertex with a specific property in each iteration. The idea we use for the inseparable demand model is relatively conceivable and provides a hint towards separable demands for which a proper strategy is less obvious. This complements the previous approximation result, i.e., the $(\Delta+1)$-approximation for general graphs with separable demands [33], where $\Delta$ is the maximum vertex degree of the graph, comparing to existing results for the classical dominating set problem. Since the approximation threshold for the classical dominating set problem is known to be $\Omega(\log n)$ [15], the results we present are also asymptotically optimal up to a constant factor.

Second, from the perspective of parameterized complexity, we prove that this problem is $W[1]$-hard when parameterized by treewidth and solution size, regardless of demand assigning model. Then we present an exact FPT algorithm for both demand models with respect to treewidth and the maximum capacity of the vertices.

Although the embedding of the parameterized results into this paper may appear to be rather loose, it plays an important role in the overview of our study. On one hand, the hardness result provides a justification to further exploration of approximation algorithms for these graphs, as it is unlikely that we can solve this problem efficiently even when the treewidth is low. On the other hand, the algorithmic result we provide in this section will further serve as a basis to our results for planar graphs that follow.

As a step towards getting polynomial-time approximations for planar graphs, we consider outerplanar graphs, i.e., planar graphs of low treewidth, and present a

**Table 1** Summary of our results. In this table, $n$ denotes the number of vertices

| | Inseparable demand | Separable demand |
|---|---|---|
| General graphs | $(\ln n)$-Approximation, Theorem 1 | $(4\ln n+2)$-Approximation, Theorem 3 |
| | | $(2\ln n+1)$-Approximation for unit vertex cost, Theorem 4 |
| $W[1]$-*hard when parameterized by* treewidth, *Theorem 5* | | |
| | FPT w.r.t. *treewidth* and *maximum vertex capacity* | |
| Graphs of bounded treewidth | Exact solution in time $O(k\cdot 2^{2k(\log M+1)}n)$, Theorem 6 | Exact solution in time $O(2^{(2M+2N+1)\log k}n)$, Corollary 2 |
| | $k$: treewidth | |
| | $M$: the maximum capacity, $N$: the maximum demand | |
| $(1+\epsilon)$-*Approx. in* pseudo-poly *time, Theorem 9* | | |
| Planar graphs | Cannot be approximated to a factor better than $(\frac{3}{2}-\epsilon)$, for any $\epsilon>0$, Theorem 10 | 80-Approximation in *polynomial* time, Theorem 11 |

constant-factor approximation for separable demands. To this end, we handle the structure of outerplanar graphs using a new approach that enables us to tackle vertices of large degrees. This is crucial since in most of the techniques which transform pseudo-polynomial time algorithms into constant-factor approximations, the error produced between neighboring vertices is accumulated at the vertices via the edges. For vertices of large degrees, the overall error would not be bounded.

To be more precise, we use a new perspective on the hierarchical structure of outerplanar graphs, which enables us to simply the underlying structure by analyzing the primal linear program of this problem. To obtain an approximation for the reduced structure, we further analyze the dual linear program. We believe that the approach we used can be applicable to other capacitated covering problems to help tackle vertices of large degrees as well.

Then, as an overview to the problem complexity on planar graphs, we present pseudo-polynomial-time approximation schemes for both demand models and constant-factor approximations for separable demands by generalizing the above algorithms under a standard framework due to Baker [3]. Although the former one follows directly, the second one requires slight modifications and careful augmentation of the algorithm proposed for outerplanar graphs. On the negative side, we prove a lower bound of $(\frac{3}{2} - \epsilon)$ for the approximation ratio of any algorithm for planar graphs with inseparable demands, for any $\epsilon > 0$.

### 1.3 Organization of this Paper

The rest of this paper is organized as follows. In Sect. 2 we define the basic terminologies that will be used throughout this paper. In Sect. 3 we present our logarithmic approximation algorithms for both inseparable and separable demand models on general graphs.

In Sect. 4 we present our parameterized results. In Sect. 5 we develop a constant-factor approximation algorithm for the separable demand model on outerplanar graphs. In Sect. 6 we apply and combine the results obtained in Sects. 4 and 5 to jointly compose a study for both demand models on planar graphs. Finally we conclude in Sect. 7 with a brief overview and a discussion on future directions.

## 2 Preliminaries

We assume that all the graphs considered in this paper are simple, connected, and undirected. Let $G = (V, E)$ be a graph. We denote the number of vertices, $|V|$, by $n$. The set of neighbors of a vertex $v \in V$ is denoted by $N_G(v) = \{u : [u, v] \in E\}$. The closed neighborhood of $v \in V$ is denoted by $N_G[v] = N_G(v) \cup \{v\}$. We use "degree of $v$" and "closed degree of $v$", written $deg_G(v)$ and $deg_G[v]$, to denote the cardinality of $N_G(v)$ and $N_G[v]$, respectively. The subscript $G$ in $N_G[v]$ and $deg_G[v]$ will be omitted when the considered graph is clear from the context.

*Planar and Outerplanar Graphs* A planar embedding of a graph $G$ is a drawing of $G$ in the plane such that the edges intersect only at their endpoints. A graph is said to be planar if it has a planar embedding. An outer-planar graph is a graph which

allows for a planar embedding such that all the vertices lie on a fixed circle, and all the edges are straight lines drawn inside the circle. For $k \geq 1$, $k$-outerplanar graphs are defined as follows. A graph is 1-outerplanar if and only if it is outer-planar. For $k > 1$, a graph is called $k$-outerplanar if it has a planar embedding such that the removal of the vertices on the unbounded face results in a $(k-1)$-outerplanar graph.

*Parameterized Complexity and Tree Decomposition* Parameterized complexity is a well-developed framework for studying computationally hard problems [14, 17, 37]. A problem is called *fixed-parameter tractable* (FPT) with respect to a parameter $k$ if it can be solved in time $f(k) \cdot n^{O(1)}$, where $f$ is a computable function depending only on $k$. Problems (along with its defining parameters) being $W[t]$-hard for any $t \geq 1$ are believed not to admit any FPT algorithms (with respect to the specified parameters). Now we define the notion of parameterized reduction.

**Definition 3** (Parameterized reduction) Let $A$ and $B$ be two parameterized problems. We say that $A$ reduces to $B$ by a parameterized reduction if there exists an algorithm $\Phi$ that transforms $(x, k)$ into $(x', g(k))$ in time $f(k) \cdot |x|^{O(1)}$, where $f, g : \mathbb{N} \to \mathbb{N}$ are arbitrary functions, such that $(x, k) \in A$ if and only if $(x', g(k)) \in B$.

Among the diverse choices of parameters, a common and widely-used one that is also closely related to the structure of the input graph is the treewidth [6, 34], which we will define in the following. This parameter is particularly useful when dealing with planar graphs. See also [3, 37]. We start with the concept of *tree decomposition*.

**Definition 4** (Tree decomposition of a graph) A tree decomposition of a graph $G = (V, E)$ is a pair $(X = \{X_i : i \in I\}, T = (I, F))$ where each node $i \in I$ is associated with a subset of vertices $X_i \subseteq V$, called the *bag* of $i$, such that the following holds.

1. Each vertex belongs to at least one bag: $\bigcup_{i \in I} X_i = V$.
2. For each edge $(u, v) \in E$, there is a bag containing its end-points, $u$ and $v$.
3. For all vertices $v \in V$, the set of nodes containing $v$, which is $\{i \in I : v \in X_i\}$, induces a subtree of $T$.

The width of a tree decomposition is defined as $\max_{i \in I} |X_i| - 1$. The treewidth of a graph $G$ is the minimum width over all possible tree decompositions of $G$.

## 3 Logarithmic Approximations for General Graphs

In this section, we present logarithmic approximation algorithms for capacitated domination problems with respect to each of the two demand models. Specifically, we provide a $(\ln n)$-approximation for the inseparable demand model, where $n$ is the number of vertices. For the separable demand model, a $(4 \ln n + 2)$-approximation is presented. For the case of unit vertex costs,we give a $(2 \ln n + 1)$-approximation.

This is achieved by properly determining a strategy that greedily selects a vertex with a specific property in each iteration. The idea we use for the inseparable demand model is relatively conceivable and provides a hint towards separable demands, for which a proper strategy is less obvious.

### 3.1 $(\ln n)$-Approximation for the Inseparable Demand Model

Below we present our algorithm, denoted $\Pi^*$, for the inseparable demand model. Let $G = (V, E)$ be the input graph and $\mathcal{U}$ be the set of vertices which have not yet been served (dominated). Initially, we have $\mathcal{U} = V$. For each vertex $u \in V$, let $N_{ud}[u] = \mathcal{U} \cap N[u]$ be the set of undominated vertices in the closed neighborhood of $u$.

In each iteration, $\Pi^*$ chooses a vertex of the greatest efficiency from $V$, where the efficiency of a vertex, say $u$, is defined by the largest ratio between the number of vertices to be dominated by $u$ and the total cost required by this assignment, among all possible demand assignments from $N_{ud}[u]$ to $u$. To be precise, let $v_{u,1}, v_{u,2}, \ldots, v_{u,|N_{ud}[u]|}$ denote the undominated neighbors $N_{ud}[u]$ of $u$, sorted in non-decreasing order of their demands. The efficiency of $u$ is defined to be

$$\delta(u) := \max_{1 \le i \le |N_{ud}[u]|} \frac{i}{w(u) \cdot x_u(i)}, \quad \text{where } x_u(i) = \left\lceil \frac{\sum_{1 \le j \le i} d(v_{u,j})}{c(u)} \right\rceil$$

is the number of copies of $u$ necessary to dominate $v_{u,1}, v_{u,2}, \ldots, v_{u,i}$. For consistency, we define $\delta(u)$ to be zero when $N_{ud}[u]$ is empty. Let $u_0$ be the vertex of maximum efficiency, and $\delta^{-1}(u_0)$ denote the corresponding index such that the ratio $i/(w(u_0) \cdot x_{u_0}(i))$ is maximized. The algorithm removes the set of vertices to be dominated, $v_{u_0,1}, v_{u_0,2}, \ldots, v_{u_0,\delta^{-1}(u_0)}$, from $\mathcal{U}$ and assigns their demands to $u_0$. This process continues until $\mathcal{U} = \emptyset$.

**Theorem 1** *Algorithm $\Pi^*$ computes a $(\ln n)$-approximation for the capacitated domination problem with inseparable demands in $O(n^3)$ time, where $n$ is the number of vertices.*

*Proof* For the correctness of this algorithm, since it only removes vertices from $\mathcal{U}$ when their demand is assigned, it always produces a feasible demand assignment function. In the following we prove that this assignment function gives a $(\ln n)$-approximation.

For each iteration, say, $j$, let $OPT_j$ denote the cost of the optimal demand assignment function for the remaining problem instance. Clearly, we have $OPT_j \le OPT$, where $OPT$ is the cost of the optimal demand assignment function for the original problem instance. Let the cardinality of $\mathcal{U}$ at the beginning of iteration $j$ be $n_j$, and let $k_j = n_j - n_{j+1}$ be the number of vertices that are newly dominated in iteration $j$.

Denote by $S_j$ the cost we spend in iteration $j$. Assume that $\Pi^*$ repeats for $m$ iterations. Since we always choose the vertex with the maximum efficiency, the efficiency of the chosen vertex is no less than the efficiency of each vertex chosen in $OPT_j$, and therefore no less than any weighted average of them, including $n_j/OPT_j$. In other words, we have

$$\frac{k_j}{S_j} \ge \frac{n_j}{OPT_j},$$

which implies that

$$S_j \le \frac{k_j}{n_j} \cdot OPT_j, \quad \text{for all } j \text{ with } 1 \le j \le m.$$

Taking the sum over all $S_j$ and observing that $n_{j+1} = n_j - k_j$, we obtain

$$\sum_{1\le j\le m} S_j \le \sum_{1\le j\le m} \frac{k_j}{n_j}\cdot OPT_j \le \left(\sum_{1\le j\le n}\frac{1}{j}\right)\cdot OPT \le \ln n \cdot OPT,$$

where the second inequality follows from the fact that

$$\frac{k_j}{n_j} \le \frac{1}{n_j} + \frac{1}{n_j-1} + \frac{1}{n_j-2} + \cdots + \frac{1}{n_j-k_j+1} = \sum_{n_{j+1}<i\le n_j}\frac{1}{i}.$$

To see that the time complexity is $O(n^3)$, notice that it requires $O(n)$ time to compute a most efficient move for each vertex, which leads to an $O(n^2)$ computation for the most efficient choice in each iteration. The number of iterations is upper bounded by $O(n)$ since at least one vertex is satisfied in each iteration.

We remark that, although it may appear that the time it requires to fetch the most efficient move in each iteration can be further improved, the number of updates required to maintain the supporting data structure could easily amount up to $O(n)$, which makes this possibility unlikely to exist. □

### 3.2 $(4\ln n + 2)$-Approximation for the Separable Demand Model

We present an algorithm $\Pi^{\mathrm{s}}$ that computes a $(4\ln n + 2)$-approximation for the separable demand model. As the demand may be partially assigned during the algorithm, for each vertex $u \in V$, we denote by $\mathrm{rd}(u)$ the amount of demand of $u$ that has not yet been served. For convenience we also refer to this quantity, $\mathrm{rd}(u)$, as *residue demand* of $u$, and to the fraction, $\mathrm{rd}(u)/d(u)$, as the *remaining portion* of $u$. Initially, $\mathrm{rd}(u)$ is set to be $d(u)$, and will be updated accordingly when a fraction of the residue demand is assigned. The vertex $u$ is said to be dominated when $\mathrm{rd}(u) = 0$.

In each iteration, algorithm $\Pi^{\mathrm{s}}$ performs two stages of greedy choices. First, it chooses the vertex of the most efficiency from $V$, where the efficiency is defined in a similar fashion as in the previous section with some modification due to the separability of the demand.

For each vertex $u \in V$, let $N_{ud}[u] = \{v_{u,1}, v_{u,2}, \ldots, v_{u,|N_{ud}[u]|}\}$ denote the set of undominated neighbors of $u$, sorted in non-descending order with respect to their demands. Let $j_u$, $0 \le j_u \le |N_{ud}[u]|$, be the largest integer such that $c(u) \ge \sum_{i=1}^{j_u}\mathrm{rd}(v_{u,i})$. In other words, we choose the largest index $j_u$ such that the residue demand of the first $j_u$ vertices in the sorted list could be served by one single copy of $u$. Let $X(u) = \sum_{i=1}^{j_u}\frac{\mathrm{rd}(v_{u,i})}{d(v_{u,i})}$ be the corresponding sum of remaining portion. In addition, to effectively use the remaining capacity provided by this single copy of $u$, we let

$$Y(u) = \frac{c(u) - \sum_{i=1}^{j_u}\mathrm{rd}(v_{u,i})}{d(v_{u,j_u+1})}$$

if $j_u < |N_{ud}[u]|$ and $Y(u) = 0$ otherwise. Since we select the vertices in the sorted order of their demands, one can easily verify that this always results in the maximum

ALGORITHM $\Pi^{\mathrm{s}}$ for the separable demand model on general graphs

1: $\mathrm{rd}(u) \longleftarrow d(u)$, and $\mathrm{map}(u) \longleftarrow \emptyset$ for each $u \in V$.
2: **while** there exist vertices with non-zero residue demand **do**
3: // **1st greedy choice**
4: Pick a vertex in $V$ with the largest efficiency, say $u$.
5: **if** $j_u$ equals 0 **then**
6: Assign the amount $c(u) \cdot \lfloor \frac{\mathrm{rd}(v_{u,1})}{c(u)} \rfloor$ of residue demand of $v_{u,1}$ to $u$.
7: $\mathrm{map}(v_{u,1}) \longleftarrow \{u\}$
8: **else**
9: Assign the residue demands of the vertices in $\{v_{u,1}, v_{u,2}, \ldots, v_{u,j_u}\}$ to $u$.
10: **if** $j_u < |N_{ud}[u]|$ **then**
11: Assign the amount $c(u) - \sum_{i=1}^{j_u} \mathrm{rd}(v_{u,i})$ of residue demand from $v_{u,j_u+1}$ to $u$.
12: $\mathrm{map}(v_{u,j_u+1}) \longleftarrow \mathrm{map}(v_{u,j_u+1}) \cup \{u\}$
13: **end if**
14: **end if**
15:
16: // **2nd greedy choice**
17: **if** there is a vertex $u$ with $0 < \mathrm{rd}(u) < \frac{1}{2} \cdot d(u)$ **then**
18: Satisfy $u$ by doubling the demand assignment of $u$ to vertices in $\mathrm{map}(u)$.
19: **end if**
20: **end while**
21: return the demand assignment function as the output.

**Fig. 1** The $(4 \ln n + 2)$-approximation for the separable demand model

remaining portion among all possible combinations. The efficiency of the vertex $u$ is defined to be $(X(u) + Y(u))/w(u)$.

Second, the algorithm $\Pi^{\mathrm{s}}$ maintains for each vertex $u \in V$ a subset of vertices, denoted by $\mathrm{map}(u)$, which consists of vertices that have served the demand of $u$ before $u$ is dominated. In other words, for each $v \in \mathrm{map}(u)$ we have a non-zero demand assignment from $u$ to $v$. During the iterations, whenever there exists a vertex $u$ whose residue demand falls below half of its original demand, i.e., $0 < \mathrm{rd}(u) < \frac{1}{2} \cdot d(u)$, after the first greedy choice, the algorithm immediately doubles the demand assignment of $u$ to the vertices in $\mathrm{map}(u)$. Note that in this way, we can completely serve the demand of $u$ since $\sum_{v \in \mathrm{map}(u)} f(u, v) > \frac{1}{2} \cdot d(u)$. This procedure repeats until every vertex of the graph is dominated. A high-level description of $\Pi^{\mathrm{s}}$ is presented in Fig. 1.

### *3.2.1 Analysis of the Algorithm*

Let $m$ be the number of iterations algorithm $\Pi^{\mathrm{s}}$ repeats. First we show that $m = O(n)$, and the algorithm always produces a feasible demand assignment function. We begin with the following lemma.

**Lemma 1** *After each iteration, the residue demand of each unsatisfied vertex is at least half of its original demand.*

*Proof* Clearly, this lemma holds in the beginning when the demand of each vertex is not yet assigned. For later stages, we argue that the algorithm properly maintains the

set map($u$) for each vertex $u \in V$ such that in our second greedy choice, whenever there exists a vertex $u$ with $0 < \mathrm{rd}(u) < \frac{1}{2} \cdot d(u)$, it is always sufficient to double the demand assignment $f(u, v)$ for each $v \in \mathrm{map}(u)$. If map($u$) is only modified under the condition $0 < j_v < |N_{ud}[v]|$ (line 12 in Fig. 1), then map($u$) contains exactly the set of vertices that have partially served $u$. Therefore we have $\sum_{v \in \mathrm{map}(u)} f(u, v) > \frac{1}{2} d(u)$, and it is sufficient to double the demand assignment in this case. If map($u$) is reassigned under the condition $j_v = 0$ at some stage, then we have $c(v) < \mathrm{rd}(u) \leq d(u)$. Since we assign the amount $c(v) \cdot \lfloor \mathrm{rd}(u)/c(v) \rfloor$ of residue demand of $u$ to $v$, this leaves at most half amount of the original residue demand of $u$, which is also no larger than $c(v)$. Therefore $u$ will be immediately dominated by doubling this assignment in the same iteration. □

For each iteration $j$, $1 \leq j \leq m$, let $u_j$ be the chosen vertex of the maximum efficiency and $n_j = \sum_{u \in V} \mathrm{rd}(u)/d(u)$ be the sum of the remaining portion over all vertices at the beginning of that iteration. For the ease of presentation, $n_j$ is defined to be zero for $j > m$. We have the following lemma regarding the changes of $n_j$ over $j$.

**Lemma 2** *For each $j$ with $1 \leq j \leq m$, we have $n_j - n_{j+1} \geq \frac{1}{2}$.*

*Proof* For iteration $j$, $1 \leq j \leq m$, let $u_j$ be the chosen vertex of the maximum efficiency. Observe that $v_{u_j,1}$ will be satisfied after this iteration. By Lemma 1, we have $\mathrm{rd}(v_{u_j,1})/d(v_{u_j,1}) \geq \frac{1}{2}$. Therefore the remaining portion covered in each iteration is at least half. □

Since the algorithm repeats until all vertices are satisfied, by Lemma 2, we know that $m \leq 2n$, and the resulting demand assignment is feasible.

In the following we show that the demand assignment function is indeed a $(4 \ln n + 2)$ approximation. Let the cost incurred by the first greedy choice be $S_1$ and the cost by the second choice be $S_2$. To see that the solution achieves the desired approximation guarantee, notice that $S_2$ is bounded from above by $S_1$, for what we do in the second choice is merely to satisfy the residue demand of a vertex, if there exists one, by doubling its previous demand assignment.

It remains to bound the cost $S_1$. For each iteration $j$, $1 \leq j \leq m$, let $u_j$ be the chosen vertex of the maximum efficiency and $OPT_j$ be the cost of the corresponding optimal demand assignment function of this remaining problem instance. Denote by $S_{1,j}$ the cost incurred by the first greedy choice in iteration $j$. We have the following two lemmas.

**Lemma 3** *For each $j$, $1 \leq j \leq m$, we have*

$$S_{1,j} \leq \frac{n_j - n_{j+1}}{n_j} \cdot OPT_j,$$

*where $n_j - n_{j+1}$ is the remaining portion covered by $u_j$ in iteration $j$.*

*Proof* The optimality of our choice in each iteration is obvious since we consider the elements of $N_{ud}[u]$ in sorted order according to their demands. Note that only in the

case $c(u) < \mathrm{rd}(v_{u,1})$, the algorithm could possibly take more than one copy. In this case the efficiency of our choice remains unchanged since the cost and the remaining portion covered by $u$ grows by the same factor. Therefore the efficiency of our choice, $(n_j - n_{j+1})/S_{1,j}$, is always no less than the efficiency of each chosen vertex in the optimal solution, and therefore no less than any of their weighted averages, including $n_j/OPT_j$. Therefore this lemma follows. □

**Lemma 4**

$$\sum_{j=1}^{m-1} \frac{\lceil n_j - n_{j+1} \rceil}{\lfloor n_j \rfloor} \leq 2 \ln n$$

*Proof* Note that we have $n_j \geq 1$ for all $j < m$, since, whenever $n_j < 1$, the remaining portion will be covered in the same iteration according to Lemmas 1 and 2. We will argue that each item of this series together constitutes at most two harmonic series.

By expanding the summand we have

$$\frac{\lceil n_j - n_{j+1} \rceil}{\lfloor n_j \rfloor} \leq \frac{1}{\lfloor n_j \rfloor} + \frac{1}{\lfloor n_j \rfloor - 1} + \cdots + \frac{1}{\lfloor n_j \rfloor - \lceil n_j - n_{j+1} \rceil + 1} \tag{1}$$

Since

$$\begin{aligned} \lfloor n_{j+1} \rfloor &= \lfloor n_j - (n_j - n_{j+1}) \rfloor \\ &\leq \lfloor n_j \rfloor - \lfloor n_j - n_{j+1} \rfloor \leq \lfloor n_j \rfloor - \lceil n_j - n_{j+1} \rceil + 1, \end{aligned}$$

possible repetitions of the expanded items only occur at the first item and the last item of (1) if we expand each summand from the summation. By Lemma 2, the decrease between $n_j$ and $n_{j+1}$ is at least half. Therefore, each repeated item, $1/(\lfloor n_j \rfloor - \lceil n_j - n_{j+1} \rceil + 1)$, will never occur more than twice in the expansion, and we can conclude that $\sum_{j=1}^{m-1} \lceil n_j - n_{j+1} \rceil / \lfloor n_j \rfloor \leq 2 \ln n$. □

We conclude our result in the following theorem.

**Theorem 2** *Algorithm $\Pi^{\mathrm{s}}$ computes a $(4 \ln n + 2)$-approximation for the capacitated domination problem with separable demands in polynomial-time.*

*Proof* By Lemma 3, we have

$$\begin{aligned} \sum_{j=1}^{m} S_{1,j} &\leq \sum_{j=1}^{m-1} \frac{n_j - n_{j+1}}{n_j} \cdot OPT_j + \frac{n_m}{n_m} \cdot OPT_m \\ &\leq \left( \sum_{j=1}^{m-1} \frac{\lceil n_j - n_{j+1} \rceil}{\lfloor n_j \rfloor} + 1 \right) \cdot OPT, \end{aligned}$$

where the second inequality follows from the fact that $\lfloor r \rfloor \leq r \leq \lceil r \rceil$ for any real number $r$ and $OPT_j \leq OPT$ for each $1 \leq j \leq m$. By Lemma 4, the cost of the demand

assignment function returned by algorithm $\Pi^s$ is bounded by

$$S_1 + S_2 \leq 2 \cdot S_1 = 2 \cdot \sum_{j=1}^{m} S_{1,j} \leq 2 \cdot (2 \ln n + 1) \cdot OPT.$$

This proves the theorem. □

### *3.2.2 Regarding the Technical Implementation Details*

A naïve implementation of this algorithm leads to a running time cubic in the number of vertices. However, by exploiting the property we assumed during the iterations that the undominated neighbors of each vertex are sorted in non-descending order according their demands, we can improve the running time of the algorithm to $O(n^2 \log n)$, which is also the time required to build the sorted list of the closed neighborhood for each vertex.

**Theorem 3** *Algorithm $\Pi^s$ computes a $(4 \ln n + 2)$-approximation for the capacitated domination problem with separable demands in $O(n^2 \log n)$ time, where $n$ is the number of vertices.*

*Proof* Below we describe the implementation detail of this improvement. The idea is to maintain the efficiency of each vertex in a binary max-heap. In each iteration, we extract the vertex of the greatest efficiency from the heap, perform the demand assignments suggested by the most efficient move, and update the efficiencies of the vertices affected by each demand assignment.

To this end, for each vertex, we maintain a pointer to the vertex in its closed neighborhood that corresponds to the last item in the most efficient move. The pointer stored for each vertex will iterate over its closed neighborhood in sorted order at most once upon updates. Whenever the residue demand of a vertex, say $u$, gets assigned, we update the efficiencies as well as the most efficient moves of its closed neighborhood accordingly. To be more precise, for each $v \in N[u]$, depending on the relative position of $u$ and the vertex to which the pointer of $v$ points in $N[v]$, we have two cases. If $u$ lies after the pointer, then no updates are required. Otherwise we iterate the pointer of $v$ according to our predefined notion of efficiency.

Since at least one vertex will be dominated and at most one vertex will be partially assigned in each iteration, the number of partial assignment will be no more than $n$. For each demand assignment, the number of updates required is bounded by the cardinality of the closed neighborhood, which is $O(n)$. Therefore, the total number of updates is $O(n^2)$. Since the pointer we maintained for each vertex is iterated over its closed neighborhood at most once, the time required for all the updates is also bounded by $O(n^2)$.

For the running time of the whole algorithm, it takes $O(n^2 \log n)$ time to build the sorted list for each vertex. In each iteration, it takes $O(\log n)$ time to extract and to maintain the heap property, provided that the efficiency of each vertex is updated. The total time required to perform the update is $O(n^2)$ from the above discussion. Therefore the overall time complexity is $O(n^2 \log n)$. □

### 3.3 $(2\ln n + 1)$-Approximation for Separable Demand Model with Unit Vertex Cost

We show that, when the demand is separable and each vertex has uniform cost, we can compute a $(2\ln n + 1)$-approximation in polynomial time. To this end, we first make a reduction on the problem instance by spending a cost of at most $OPT$ such that it takes at most one copy to serve each remaining undominated vertex. Then we show that a $(2\ln n)$-approximation can be computed for this reduced problem instance, following a similar approach proposed in the last section.

For each $u \in V$, let $g_u \in N[u]$ be the vertex with the maximum capacity. First, for each $u \in V$, we assign $c(g_u) \cdot \lfloor \frac{d(u)}{c(g_u)} \rfloor$ of the demand of $u$ to $g_u$. Let $S$ be the total cost incurred by this assignment. We have the following Lemma 5.

**Lemma 5** *We have $S \leq OPT$, where $S$ is the total cost incurred by assigning $c(g_u) \cdot \lfloor \frac{d(u)}{c(g_u)} \rfloor$ demand of the vertex $u$ to $g_u$, for all $u \in V$, and $OPT$ is the cost of the optimal demand assignment function.*

*Proof* Notice that, when fractional multiplicities are allowed, an optimal demand assignment $O^*$ can be obtained by assigning the demand $d(u)$ of $u$ to $g_u$, for each vertex $u$. Let

$$w^*(O^*) = \sum_{u \in V} w(u) \cdot \frac{\sum_{v \in N[u]} O^*(v,u)}{c(u)}$$

be the total cost required by $O^*$. Since $S \leq w^*(O^*)$ and $w^*(O^*) \leq OPT$, the lemma follows. □

In the following, we will assume that $d(u) \leq c(g_u)$, for each $u \in V$. The algorithm provided in Sect. 3.2 is slightly modified. In particular, for the second greedy choice, whenever $\mathrm{rd}(u) < d(u)$ for some vertex $u \in V$, we immediately assign the residue demand of $u$ to $g_u$. Let $\Pi^{\mathrm{s,unit}}$ denote this modified algorithm. A high-level description of this algorithm is also given in Fig. 2. Recall that $n_j$ is the total remaining portions of the vertices at the beginning of iteration $j$. We have the following lemma, which is an updated version of Lemma 2.

**Lemma 6** *We have $n_j - n_{j+1} \geq 1$ for each $1 \leq j \leq m$.*

*Proof* Observe that in each iteration, at least one vertex is dominated and the residue demand of each vertex is either 0 or equal to its original demand. □

We conclude this result in the following theorem.

**Theorem 4** *Algorithm $\Pi^{\mathrm{s,unit}}$ computes a $(2\ln n + 1)$-approximation in time $O(n^2 \log n)$ for the capacitated domination problem with separable demand and uniform costs, where $n$ is the number of vertices.*

*Proof* We adopt the notation from the previous section. Clearly, $S_2$ is bounded above by $S_1$, as we always take one copy for the first greedy choice and at most one copy

ALGORITHM $\Pi^{s,\text{unit}}$ for the separable demand model with unit vertex cost

1: For each $u \in V$, assign $c(g_u) \cdot \lfloor \frac{d(u)}{c(g_u)} \rfloor$ demands of $u$ to $g_u$, where $g_u \in N[u]$ has the maximum capacity.
2: Reset the demands of the instance by setting $d(u) \longleftarrow \text{rd}(u)$ for each $u \in V$.
3: **while** there exist vertices with non-zero residue demand **do**
4: **// 1st greedy choice**
5: Pick a vertex in $V$ with the most efficiency, say $u$.
6: Assign the demands of the vertices in $\{v_{u,1}, v_{u,2}, \ldots, v_{u,j_u}\}$ to $u$.
7: **if** $j_u < |N_{ud}[u]|$ **then**
8: Assign the amount $c(u) - \sum_{i=1}^{j_u} \text{rd}(v_{u,i})$ of the residue demand of $v_{u,j_u+1}$ to $u$.
9: **end if**
10:
11: **// 2nd greedy choice**
12: **if** there is a vertex $u$ with $0 < \text{rd}(u) < d(u)$ **then**
13: Satisfy $u$ by assigning the residue demand of u to $g_u$.
14: **end if**
15: **end while**
16: return the demand assignment function as the output.

**Fig. 2** The pseudo-code for the separable demand model with unit vertex costs

for the second greedy choice in each iteration. By Lemma 6 and the fact that $n_j$ is integral for each $1 \leq j \leq m$, we have

$$\sum_{j=1}^{m} S_{1,j} \leq \sum_{j=1}^{m} \frac{n_j - n_{j+1}}{n_j} \cdot OPT_j \leq \ln n \cdot OPT,$$

and $S + S_1 + S_2 \leq OPT + 2 \cdot \sum_{j=1}^{m} S_{1,j} \leq (2 \ln n + 1) \cdot OPT$. □

## 4 Graphs of Bounded Treewidth

In this section, we present results for graphs of bounded treewidth. Specifically, we prove the $W[1]$-hardness when parameterized by treewidth and solution size, regardless of demand assigning models, for the capacitated domination problem. Then we develop a fixed-parameter tractable algorithm for this problem, taking both treewidth and maximum capacity as the parameters.

On one hand, the hardness result provides a justification to further exploration of approximation algorithms for these graphs, as it is unlikely that we can solve this problem efficiently even when the treewidth is low. On the other hand, the algorithmic result we provide in this section will further serve as a basis to our results for planar graphs that follow.

*Tree Decomposition of a Graph* Before presenting our results for this part, let us review the basic concept and terminologies regarding tree decompositions. Further references can be found in [6, 34, 37].

**Definition 5** (Tree decomposition of a graph) A tree decomposition of a graph $G = (V, E)$ is a pair $(X = \{X_i : i \in I\}, T = (I, F))$ where each node $i \in I$ is associated with a subset of vertices $X_i \subseteq V$, called the *bag* of $i$, such that

1. Each vertex belongs to at least one bag: $\bigcup_{i \in I} X_i = V$.
2. For each edge $(u, v) \in E$, there is a bag containing its end-points, $u$ and $v$.
3. For all vertices $v \in V$, the set of nodes containing $v$, which is $\{i \in I : v \in X_i\}$, induces a subtree of $T$.

The width of a tree decomposition is defined as $\max_{i \in I} |X_i| - 1$. The treewidth of a graph $G$ is the minimum width over all possible tree decompositions of $G$.

### 4.1 $W[1]$-Hardness w.r.t. Treewidth

In the following, we show that the capacitated domination problem is $W[1]$-hard when parameterized by treewidth. The reduction is done from the *k-Multicolored Clique Problem*, which is a restriction of the $W[1]$-hard *k-Clique* problem [16] and whose objective is to decide the existence of a clique of size $k$ from a given collection of $k$ vertex sets, each forms an independent set.

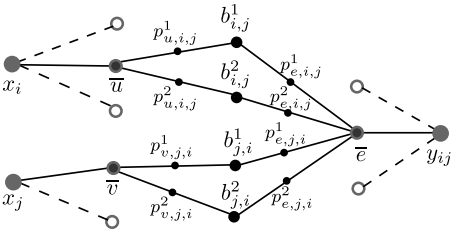
**Definition 6** (Multicolored clique) Given an integer $k$ and a connected undirected graph $G = (\bigcup_{i=1}^{k} V[i], E)$ such that $V[i]$ induces an independent set for each $i$, the MULTICOLORED CLIQUE problem asks whether or not there exists a clique of size $k$ in $G$.

Given an instance $(G, k)$ of multicolored clique, we show how an instance $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of treewidth $O(k^2)$ for the capacitated domination problem can be built such that $G$ has a clique of size $k$ if and only if $\mathcal{G}$ has a capacitated dominating multi-set of cardinality at most $k' = (3k^2 - k)/2$. Furthermore, in order to distinguish the vertices between the given graph $\mathcal{G}$ and the graph $G$ we construct for the reduction, we will refer the vertices of $\mathcal{G}$ to as *nodes* in the following content.

While the technical detail is subtle, the general idea behind is conceivable. For each independent set $V[i]$, $1 \leq i \leq k$, given in multicolored clique, we construct a star rooted at a node $x_i$ containing a node $\overline{u}$ for each vertex $u \in V[i]$. From the construction we guarantee that exactly one node from each star will be picked in the optimal capacitated dominating multi-set, and this will correspond to the selection of the vertices to form a clique in $G$. Similarly, we construct a rooted star $y_{i,j}$ for each $1 \leq i < j \leq k$ containing nodes corresponding to the set of edges between $V[i]$ and $V[j]$. This will represent the set of edges that form a clique together with the chosen vertices.

To ensure that the set of vertices and the set of edges we pick will form a clique, additional bridge nodes are created to link the node corresponding to each edge in $E$ and the nodes corresponding to its two end-vertices. Also refer to Fig. 3 for an illustration of the construction.

Let $N$ be the number of vertices. Without loss of generality, we label the vertices of $G$ by integers between 1 and $N$, which we denote by $label(v)$ for each $v \in V$. For each $i \neq j$, let $E[i, j]$ denote the set of edges between $V[i]$ and $V[j]$. The graph $\mathcal{G}$




$i < j$, $u \in V[i]$, $v \in V[j]$, and $e = (u, v) \in E[i, j]$

**Fig. 3** The connections between stars and bridge nodes

is defined as follows. For each $i$, $1 \le i \le k$, we create a node $x_i$ with $w(x_i) = k' + 1$, $c(x_i) = 0$, and $d(x_i) = 1$. For each $u \in V[i]$, we have a node $\overline{u}$ with $w(\overline{u}) = 1$, $c(\overline{u}) = 1 + (k-1)N$, and $d(\overline{u}) = 0$. We also connect $\overline{u}$ to $x_i$. For convenience, we refer to the star rooted at $x_i$ as vertex star $T_i$.

Similarly, for each $1 \le i < j \le k$, we create a node $y_{ij}$ with $w(y_{ij}) = k' + 1$, $c(y_{ij}) = 0$, and $d(y_{ij}) = 1$. For each $e \in E[i, j]$ we have a node $\overline{e}$ with $w(\overline{e}) = 1$, $c(\overline{e}) = 1 + 2N$, and $d(\overline{e}) = 0$. We connect $\overline{e}$ to $y_{ij}$. We refer to the star rooted at $y_{ij}$ as edge star $T_{ij}$. The selection of nodes in $T_i$ and $T_{ij}$ in the capacitated dominating multi-set will correspond to the decision of selecting the vertices to form a clique in $G$.

In addition, for each $i \ne j$, $1 \le i, j \le k$, we create two bridge nodes $b^1_{i,j}$, $b^2_{i,j}$ with $w(b^1_{i,j}) = w(b^2_{i,j}) = 1$ and $d(b^1_{i,j}) = d(b^2_{i,j}) = 1$. The capacities of the bridge nodes are to be defined later.

Now we describe how the leaf nodes of $T_i$ and $T_{ij}$ are connected to bridge nodes such that the result we claimed holds. For each vertex star $T_i$, $1 \le i \le k$, each $j$ with $1 \le j \le k$, $i \ne j$, and each $v \in V[i]$, we create two propagation nodes $p^1_{v,i,j}$, $p^2_{v,i,j}$ and connect them to $\overline{v}$. Besides, we connect $p^1_{v,i,j}$ to $b^1_{i,j}$ and $p^2_{v,i,j}$ to $b^2_{i,j}$. We set $w(p^1_{v,i,j}) = w(p^2_{v,i,j}) = k' + 1$ and $c(p^1_{v,i,j}) = c(p^2_{v,i,j}) = 0$. The demands of $p^1_{v,i,j}$ and $p^2_{v,i,j}$ are set to be $d(p^1_{v,i,j}) = label(v)$ and $d(p^2_{v,i,j}) = N - label(v)$.

For each edge star $T_{i,j}$, $1 \le i < j \le k$ and each $e = (u, v) \in E[i, j]$ such that $u \in V[i]$ and $v \in V[j]$, we create four propagation nodes $p^1_{e,i,j}$, $p^2_{e,i,j}$, $p^1_{e,j,i}$, and $p^2_{e,j,i}$, which are connected to $\overline{e}$, with zero capacity and $k' + 1$ cost. In addition, we also connect $p^1_{e,i,j}$, $p^2_{e,i,j}$, $p^1_{e,j,i}$, $p^2_{e,j,i}$ to $p^1_{i,j}$, $p^2_{i,j}$, $p^1_{j,i}$, $p^2_{j,i}$, respectively. The demands of the four nodes are set as the following: $d(p^1_{e,i,j}) = N - label(u)$, $d(p^2_{e,i,j}) = label(u)$, $d(p^1_{e,j,i}) = N - label(v)$, and $d(p^2_{e,j,i}) = label(v)$.

Finally, for each bridge node $b$, we set $c(b) = \sum_{u \in N[b]} d(u) - N$. In principle, the parameters are set in a way such that, when the vertices from a vertex star and an edge star are picked, the corresponding propagation nodes can reflect this decision in an optimal demand assignment function.

**Lemma 7** *The treewidth of $\mathcal{G}$ is $O(k^2)$.*

*Proof* Consider the set of bridge nodes, $Bridge = \bigcup_{i \ne j} \{b^1_{i,j} \cup b^2_{i,j}\}$. Since $\mathcal{G} \backslash Bridge$ is a forest, which is of treewidth 1, and the removal of a vertex from a graph decreases

the treewidth by at most one, the treewidth of $\mathcal{G}$ is upper bounded by the number of bridge nodes plus 1, which is $O(k^2)$. □

**Lemma 8** *$G$ admits a clique of size $k$ if and only if $\mathcal{G}$ admits a capacitated dominating set of cost at most $k' = (3k^2 - k)/2$.*

*Proof* Let $C$ be a clique of size $k$ in $G$. By choosing the bridge nodes, $b^1_{i,j}$ and $b^2_{i,j}$ for each $i \neq j$, $\overline{u}$ for each $u \in C$, and $\overline{e}$ for each $e \in C$ exactly once, we have a vertex subset of cost exactly $(3k^2 - k)/2$. One can easily verify that this is also a feasible capacitated dominating multi-set for $\mathcal{G}$.

On the other hand, let $D$ be a capacitated dominating multi-set of cost at most $k'$ in $\mathcal{G}$. We will argue that there exists a clique of size $k$ in $G$. First observe that none of the propagation nodes are chosen in $D$, otherwise the cost would exceed $k'$. This implies $b^1_{i,j} \in D$ and $b^2_{i,j} \in D$, for each $i \neq j$, as they are adjacent only to propagation nodes. Note that this already contributes $k(k-1)$ nodes to $D$ with cost at least $k(k-1)$ and the rest of the nodes in $D$ together contributes at most $k(k+1)/2$ cost.

Similarly, we conclude that $x_i \notin D$ and $y_{ij} \notin D$ for each $i \neq j$. Therefore, for each $1 \leq i \leq k$, there exists $u \in V[i]$ such that $\overline{u} \in D$, and for each $i \neq j$, there exists $e \in E[i, j]$ such that $\overline{e} \in D$. Since we have $k(k-1)/2 + k = k(k+1)/2$ such stars, exactly one node from each star is chosen to be included in $D$ and therefore the multiplicity of each node in $D$ is exactly one.

Next we argue that the nodes chosen in each star will correspond to a clique of size $k$ in $G$. For each $1 \leq i < j \leq k$, let $\overline{u} \in T_i$ and $\overline{v} \in T_j$ be the nodes chosen in $D$. Let $\overline{e} \in T_{ij}$ be the node chosen in $D$. In the following, we argue that the two end-vertices of $e$ are exactly $u$ and $v$, meaning that $e = (u, v)$. Note that, this will imply the existence of a clique of size $k$ in $G$, formed by the vertices corresponding to the nodes chosen in each $T_i$, $1 \leq i \leq k$.

Since the capacity of $\overline{u}$ equals the sum of the demands over $N[\overline{u}]$, without loss of generality we can assume that the demands of nodes from $N[\overline{u}]$ are served merely by $\overline{u}$. Consider the bridge vertex $b^1_{ij}$ and the set $S = N[b^1_{ij}] \backslash N[\overline{u}]$. The demand of vertices in $S$ can only be served by either $b^1_{ij}$ or $\overline{e}$, as they are the only two vertices in $N[S]$ that are chosen in dominating multi-set $D$. See also Fig. 4. In particular, vertices in $S \backslash \{p^1_{e,i,j}\}$ can only be served by $b^1_{ij}$. Therefore, we have

$$c\left(b^1_{ij}\right) \geq \sum_{u \in S \backslash \{p^1_{e,i,j}\}} d(u).$$

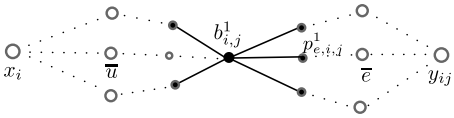



**Fig. 4** Local connections around the bridge node $b^1_{i,j}$. The solid circles represent the set $S = N[b^1_{i,j}] \backslash N[\overline{u}]$

Since $c(b^1_{ij}) = \sum_{u \in N[b^1_{ij}]} d(u) - N$ by our setting, the above inequality implies $d(p^1_{e,i,j}) \geq N - label(u)$, which in turn implies $d(p^2_{e,i,j}) \leq label(u)$ as we have $d(p^1_{e,i,j}) + d(p^2_{e,i,j}) = N$ by our construction.

By a symmetric argument on $b^2_{i,j}$, we obtain $d(p^2_{e,i,j}) \geq label(u)$. Hence $d(p^2_{e,i,j}) = label(u)$. By another symmetric argument on $b^1_{ji}$ and $b^2_{ji}$, we obtain $d(p^2_{e,j,i}) = label(v)$. Therefore $e = (u, v)$ and the lemma follows. □

Note that this proof holds for both separable and inseparable demand models. We have the following theorem.

**Theorem 5** *The capacitated domination problem is $W[1]$-hard when parameterized by treewidth, regardless of the demand model.*

*Proof* Clearly the reduction instance $\mathcal{G}$ can be computed in time polynomial in both $k$ and $N$. By Lemma 7, $\mathcal{G}$ has treewidth $O(k^2)$. This theorem follows directly from Lemma 8 and the $W[1]$-hardness of multicolored clique. □

We would like to remark that, from the construction of the graph $\mathcal{G}$ and the proof of Lemma 8, a feasible capacitated dominating multi-set with cost at most $k'$ for $\mathcal{G}$ also has a cardinality of at most $k'$, and vice versa. Hence we have the following corollary.

**Corollary 1** *The capacitated domination problem is $W[1]$-hard when parameterized by treewidth and solution size, regardless of the demand model.*

Although we can state a stronger statement by incorporating the cardinality of the solution size, i.e., the cardinality of the dominating multi-set as an extra parameter, knowing the solution size does not generally help us to obtain the demand assignment function.

For example, consider the graph drawn in Fig. 5 with inseparable demands. Even when we know that taking one multiplicity from each of $v_\ell$ and $v_r$ forms a feasible dominating multi-set, computing one of such demand assignment functions is still NP-hard, by making a reduction from *Partition*, a well-known weakly NP-hard problem. When the demand is separable, the computation can be done by using a standard *flow* algorithm. However, this does not give any guarantee on the cost of the resulting demand assignment.

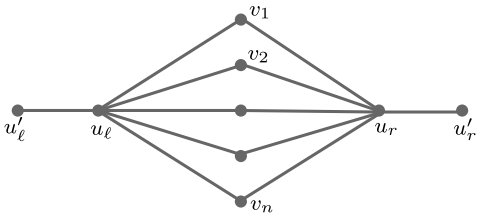



**Fig. 5** A construction relating the computation of a feasible demand assignment function and the solution to the *Partition* problem when the demand is inseparable

### 4.2 Fixed-Parameter Tractability w.r.t. Treewidth and Maximum Capacity

We have shown in the previous section that the capacitated domination problem is $W[1]$-hard, regardless of demand assignment model, when parameterized by the treewidth of the input graph. In this section we show that, by taking the maximum capacity as an extra parameter in addition to the treewidth, this problem becomes fixed-parameter tractable when the demand is inseparable and can be solved in $O(2^{2k((\log M+1)+\log k)} \cdot n)$ time, where $M$ is the maximum capacity of the vertices.

To this end, we give a dynamic programming algorithm on a *nice tree decomposition* [34], which is a special tree decomposition, of the input graph. The advantage of this decomposition lies in the fact that it provides the structural information of the given graph in a well-organized fashion. Below we give a formal definition for this concept.

**Definition 7** (Nice tree decomposition [34]) A tree decomposition

$$\big(X = \{X_i : i \in I\}, T = (I, F)\big)$$

is a nice tree decomposition if one can root $T$ in a way such that each node $i \in I$ is of one of the following four types.

1. *Leaf*: node $i$ is a leaf of $T$, and $|X_i| = 1$.
2. *Join*: node $i$ has exactly two children, say $j_1$ and $j_2$, and $X_i = X_{j_1} = X_{j_2}$.
3. *Introduce*: node $i$ has exactly one child, say $j$, and there is a vertex $v \in V$ such that $X_i = X_j \cup \{v\}$.
4. *Forget*: node $i$ has exactly one child, say $j$, and there is a vertex $v \in V$ such that $X_j = X_i \cup \{v\}$.

For a given tree decomposition of a graph with $O(n)$ nodes and a bounded width, a nice tree decomposition of the same width can be found in $O(n)$ time [34], where $n$ is the number of vertices.

Let $G = (V, E)$ be the input graph and $(\mathcal{X}, \mathcal{T})$ be a nice tree decomposition of $G$. For each node $i \in \mathcal{T}$, let $\mathcal{T}_i$ be the subtree of $\mathcal{T}$ rooted at $i$ and $\mathcal{Y}_i := \bigcup_{j \in \mathcal{T}_i} \mathcal{X}_j$. Literally, $\mathcal{Y}_i$ denotes the set of vertices that are contained in the bags of the nodes in $\mathcal{T}_i$.

Starting from the leaf nodes of $\mathcal{T}$, our algorithm proceeds in a bottom-up manner and maintains for each node $i \in \mathcal{T}$ a table $A_i$ whose columns consist of the following information.

- A subset of $\mathcal{X}_i$ indicating the set of vertices in $\mathcal{X}_i$ that have already been dominated, and
- for each $u \in \mathcal{X}_i$, the amount of residue capacity of $u$, denoted $\mathrm{rc}(u)$, where $0 \leq \mathrm{rc}(u) < c(u)$.

For each possible configuration of the columns described above, the algorithm will maintain a row in $A_i$ and computes the cost of the optimal demand assignment

function for the subgraph induced by $\mathcal{Y}_i$ under the condition that the set of vertices that have been dominated by this demand assignment and also the residue capacity of each vertex meet exactly the values specified by the row. In order to help present the algorithm, for each row we store in the table $A_i$, say row $r$, we will refer to the corresponding subset of $\mathcal{X}_i$ stored in row $r$ by $\mathcal{P}_r$.

In the following, we describe the computation of the table $A_i$ for each node $i$ in the tree $\mathcal{T}$. For the ease of presentation, we implicitly assumes that, whenever the algorithm attempts to insert a new row into a table while another row with identical configuration already exists, the one with the smaller cost will be kept. According to different types of vertices we encounter during the procedure, we have the following four cases.

– *$i$ is a leaf node*. Let $X_i = \{v\}$. We add two rows to the table $A_i$ which correspond to cases whether or not $v$ is served.

---

1: let $r_1 = (\{\emptyset\}, \{\text{rc}(v) = 0\})$ be a new row with $\text{cost}(r_1) \longleftarrow 0$
2: let $r_2 = (\{v\}, \{\text{rc}(v) \equiv d(v) \bmod c(v)\})$ be a new row with $\text{cost}(r_2) \longleftarrow w(v) \cdot \lceil \frac{d(v)}{c(v)} \rceil$
3: add $r_1$ and $r_2$ to $A_i$

---

– *$i$ is an introduce node*. Let $j$ be the child of $i$, and let $\mathcal{X}_i = \mathcal{X}_j \cup \{v\}$. The data in $A_j$ is inherited by $A_i$. We extend $A_i$ by considering, for each existing row $r$ in $A_j$, all $2^{|\mathcal{X}_j \setminus \mathcal{P}_r|}$ possible ways of choosing vertices in $\mathcal{X}_j \setminus \mathcal{P}_r$ to be assigned to $v$. In addition, $v$ can be either unassigned or assigned to any vertex in $X_i$. In either case, the cost and the residue capacity are modified accordingly.

---

1: **for all** rows $r_0 = (P, R) \in A_j$ **do**
2: **for all** possible $U$ such that $U \subseteq (X_j \setminus P) \cap N_G(v)$ **do**
3: let $R' = R \cup \{\text{rc}(v) = \sum_{u \in U} d(u) \bmod c(v)\}$, and let $r = (P \cup U, R')$ be a new row with $\text{cost}(r) = \text{cost}(r_0) + w(v) \cdot \lceil \frac{\sum_{u \in U} d(u)}{c(v)} \rceil$
4: add $r$ to $A_i$
5: **for all** $u \in X_i$ **do**
6: let $r' = (P \cup U \cup \{v\}, R' \cup \{\text{rc}(u) = (\text{rc}(u) - d(v)) \bmod c(u)\})$ be a new row with $\text{cost}(r') = \text{cost}(r)+$ the cost required by this assignment
7: add $r'$ to $A_i$
8: **end for**
9: **end for**
10: **end for**

---

– *$i$ is a forget node*. Let $j$ be the child of $i$, and let $\mathcal{X}_i = \mathcal{X}_j \setminus \{v\}$. In this case, for each row $r \in A_j$ such that $v \in \mathcal{P}_r$, we insert a row $r'$ to $A_i$ identical to $r$ except for the absence of $v$ in $\mathcal{P}_{r'}$. The remaining rows in $A_j$, which correspond to situations where $v$ is not served, are ignored without being considered.

1: **for all** rows $r_0 = (P, R) \in A_j$ such that $v \in P$ **do**
2: let $r = (P \backslash \{v\}, R \backslash \{\mathrm{rc}(v)\})$ be a new row with $\mathrm{cost}(r) = \mathrm{cost}(r_0)$
3: add $r$ to $A_i$
4: **end for**

– *$i$ is a join node*. Let $j_1$ and $j_2$ be the two children of $i$ in $\mathcal{T}$. We consider every pair of rows $r_1, r_2$ where $r_1 \in A_{j_1}$ and $r_2 \in A_{j_2}$. We say that two rows $r_1$ and $r_2$ are *compatible* if $\mathcal{P}_{r_1} \cap \mathcal{P}_{r_2} = \emptyset$. For each compatible pair of rows $(r_1, r_2)$, we insert a new row $r$ to $A_i$ with $\mathcal{P}_r = \mathcal{P}_{r_1} \cup \mathcal{P}_{r_2}$, $rc_r(u) = (rc_{r_1}(u) + rc_{r_2}(u)) \bmod c(u)$, for each $u \in \mathcal{X}_i$, and $\mathrm{cost}(r) = \mathrm{cost}(r_1) + \mathrm{cost}(r_2) - \sum_{u \in \mathcal{X}_i} \lfloor \frac{rc_{r_1}(u) + rc_{r_2}(u)}{c(u)} \rfloor$.

1: **for all** compatible pairs $r_1 = (P_1, R_1) \in A_{j_1}$ and $r_2 = (P_2, R_2) \in A_{j_2}$ **do**
2: let $r = (P_1 \cup P_2, R)$ be a new row
3: $\mathrm{cost}(r) \longleftarrow \mathrm{cost}(r_1) + \mathrm{cost}(r_2) - \sum_{u \in X_i} \lfloor \frac{rc_{R_1}(u) + rc_{R_2}(u)}{c(u)} \rfloor$, and
4: $R \longleftarrow \{rc_{R_1}(u) + rc_{R_2}(u) \bmod c(u) : u \in X_i\}$
5: add $r$ to $A_i$
6: **end for**

**Theorem 6** *The capacitated domination problem with inseparable demand on graphs of bounded treewidth can be solved in time $2^{2k(\log M+1)+\log k+O(1)} \cdot n$, where $k$ is the treewidth and $M$ is the maximum capacity of the graph.*

*Proof* The correctness of the algorithm follows from an inductive argument along with the description given above. Regarding the running time of this algorithm, first notice that the size of the table we maintained for each node of the tree decomposition is bounded by $2^k \cdot M^k$. The computation for leaf nodes takes $O(1)$ time, while the computation for introduce nodes and forget nodes take $O(k2^{2k} \cdot M^k)$ and $O(2^k \cdot M^k)$, respectively. In join nodes, however, we consider each compatible pair of rows from two tables. Therefore the computation requires $O(k2^{2k} \cdot M^{2k})$ time. The size of a nice tree decomposition is linear in the number of vertices of the graph. Therefore the overall running time of the algorithm is $O(k2^{2k} \cdot M^{2k} \cdot n) = 2^{2k(\log M+1)+\log k+O(1)} \cdot n$. □

We state without going into details that, by suitably replacing the set $P_i$ we maintained for each row of the table $\mathcal{A}_i$ with the residue demand of each vertex contained in the bag $\mathcal{X}_i$, the algorithm can be modified to handle separable demand model as well. We have the following corollary.

**Corollary 2** *The capacitated domination problem with separable demand on graphs of bounded treewidth can be solved in time $2^{(2M+2N+1)\log k+O(1)} \cdot n$, where $k$ is the treewidth, $M$ is the maximum capacity, and $N$ is the maximum demand.*

## 5 A Constant-Factor Approximation for Outer-Planar Graphs with Separable Demands

In this section, we develop an algorithm that computes a constant-factor approximation for outerplanar graphs. In order to help present the algorithm and the final analysis in a systematic manner, we divide the entire section in the following way.

In Sect. 5.1, we present a structural statement that classifies the outer-planar graphs into a class of graphs which we called *general ladders*, followed by showing how a corresponding representation can be extracted in $O(n \log^3 n)$ time. In Sect. 5.2 we present a linear program, formerly introduced in [33], for this problem with separable demands. In Sect. 5.3 we consider the primal linear program and propose a scheme to reduce the structure of the general ladders we introduce in Sect. 5.1. Then in Sect. 5.4 we consider the dual program of the relaxation of the linear program to obtain a constant-factor approximation for the reduced subgraph.

To give an exact idea on how the algorithm works, for any outerplanar graph $G = (V, E)$, our algorithm proceeds as follows. First we use the algorithm described in Sect. 5.1 to compute a general ladder representation of $G$ and a decomposition into three subproblems, denoted $\mathcal{G}_0$, $\mathcal{G}_1$, and $\mathcal{G}_2$. For each subproblem $\mathcal{G}_i$, $0 \leq i < 3$, we use the approach described in Sect. 5.3 to further reduce its structure and to obtain a reduced subgraph $\mathcal{H}_i$. For each of the reduced subgraph $\mathcal{H}_i$, we apply the algorithm described in Sect. 5.4 to obtain an approximation, represented by a demand assignment function $f_i$. The overall approximation for $G$, i.e., the demand assignment function $f$, is defined to be $f = \sum_{0 \leq i < 3} f_i$. An overall analysis of this algorithm is given in Sect. 5.5.

### 5.1 General Ladders and Outerplanar Graphs

First we define the notation which we will use later on. By a total order of a set we mean that each pair of elements in the set can be compared, and therefore an ascending order of the elements is well-defined. Let $P = (v_1, v_2, \ldots, v_k)$ be a path. We say that $P$ is an *ordered path* if a total order $v_1 \prec v_2 \prec \cdots \prec v_k$ or $v_k \prec v_{k-1} \prec \cdots \prec v_1$ is defined on the set of vertices. Note that, provided the notion of total orders, the concepts of *maximum elements* and *minimum elements* naturally extends, i.e., for any finite set $A$ with a total order defined, we have $\min_{x \in A} x \preccurlyeq y \preccurlyeq \max_{x \in A} x$ for all $y \in A$.

**Definition 8** (General ladder) A graph $G = (V, E)$ is said to be a general ladder if a total order on the set of vertices is defined, and $G$ is composed of a set of layers $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$, where each layer is a collection of subpaths of an ordered path such that the following holds. The top layer, $\mathcal{L}_1$, consists of a single vertex, which is referred to as the anchor, and for each $1 < j < k$ and $u, v \in \mathcal{L}_j$, we have (1) $N[u] \subseteq \mathcal{L}_{j-1} \cup \mathcal{L}_j \cup \mathcal{L}_{j+1}$, and (2) $u \prec v$ implies $\max_{p \in N[u] \cap \mathcal{L}_{j+1}} p \preccurlyeq \min_{q \in N[v] \cap \mathcal{L}_{j+1}} q$.

Note that each layer in a general ladder consists of a set of ordered paths which are possibly connected only to vertices in the neighboring layers. See Fig. 6(a). Although the definition of general ladders captures the essence and simplicity of an ordered

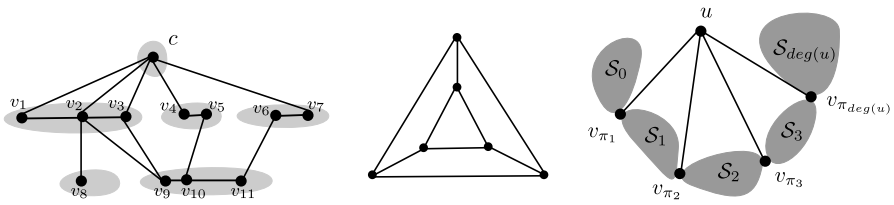



**Fig. 6** (**a**) A general ladder with anchor $c$ and two total orders $v_1 \preccurlyeq v_2 \preccurlyeq \cdots \preccurlyeq v_7$, $v_8 \preccurlyeq v_9 \preccurlyeq v_{10} \preccurlyeq v_{11}$. (**b**) A 2-outerplanar graph which fails to be a general ladder. (**c**) The subdivision formed by a vertex $u$ in an outer-planar embedding

hierarchical structure, there are planar graphs which fall outside this framework. See also Fig. 6(b).

In the following, we state and argue that every outerplanar graph meets the requirements of a general ladder. We assume that an outer-planar embedding for any outer-planar graph is given as well. Otherwise we apply the $O(n \log^3 n)$ algorithm provided by Bose [7] to compute such an embedding.

Let $G = (V, E)$ be an outer-planar graph, $u \in V$ be an arbitrary vertex, and $\mathcal{E}$ be an outer-planar embedding of $G$. We fix $u$ to be the smallest element and define a total order on the vertices of $G$ according to their orders of appearances on the outer face of $\mathcal{E}$ in a counter-clockwise order. For convenience, we label the vertices such that $u = v_1$ and $v_1 \prec v_2 \prec v_3 \prec \cdots \prec v_n$.

Let $N(u) = \{v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_{deg(u)}}\}$ denote the neighbors of $u$ such that $v_{\pi_1} \prec v_{\pi_2} \prec \cdots \prec v_{\pi_{deg(u)}}$. The set $N(u)$ divides the set of vertices except $u$ into $deg(u)+1$ subsets, namely, $\mathcal{S}_0 = \{v_2, v_3, \ldots, v_{\pi_1}\}$, $\mathcal{S}_i = \{v_{\pi_i}, v_{\pi_i+1}, \ldots, v_{\pi_{i+1}}\}$ for $1 \le i < deg(u)$, and $\mathcal{S}_{deg(u)} = \{v_{\pi_{deg(u)}}, v_{\pi_{deg(u)}+1}, \ldots, v_n\}$. See Fig. 6(c) for an illustration. For any $0 < i < j < deg(u)$, $v_{\pi_{i-1}} \prec p \prec v_{\pi_i}$, and $v_{\pi_{j-1}} \prec q \prec v_{\pi_j}$, there is no edge connecting $p$ and $q$. Otherwise it will result in a crossing with the edge $(u, v_{\pi_i})$, contradicting to the fact that $\mathcal{E}$ is a planar embedding.

For $1 \le i < deg(u)$, we partition $\mathcal{S}_i$ into two sets $L_i$ and $R_i$ as follows. Let $d_{\mathcal{S}_i}$ denote the distance function defined on the induced subgraph of $\mathcal{S}_i$. Let $L_i = \{v : v \in \mathcal{S}_i, d_{\mathcal{S}_i}(v_{\pi_i}, v) \le d_{\mathcal{S}_i}(v, v_{\pi_{i+1}})\}$ and $R_i = \mathcal{S}_i \backslash L_i$.

**Lemma 9** *We have* $\max_{a \in L_i} a \prec \min_{b \in R_i} b$ *for all* $1 \le i < deg(u)$.

*Proof* For convenience, let $a_i = \max_{a \in L_i} a$ and $b_i = \min_{b \in R_i} b$. Since $L_i \cap R_i = \emptyset$, we have $a_i \ne b_i$. Assume that $b_i \prec a_i$. Recall that in an outer-planar embedding, the vertices are placed on a circle and the edges are drawn as straight lines. Since $\mathcal{E}$ is an outer-planar embedding, the shortest path from $a_i$ to $v_{\pi_i}$ must intersect with the shortest path from $b_i$ to $v_{\pi_{i+1}}$. Let $c_i$ be the vertex for which the two paths meet. Since $L_i$ and $R_i$ form a partition of $\mathcal{S}_i$, either $c_i \in L_i$ or $c_i \in R_i$.

If $c_i \in L_i$, then $d_{\mathcal{S}_i}(c_i, v_{\pi_i}) \le d_{\mathcal{S}_i}(c_i, v_{\pi_{i+1}})$ by definition, which implies that $d_{\mathcal{S}_i}(b_i, v_{\pi_i}) \le d_{\mathcal{S}_i}(b_i, v_{\pi_{i+1}})$, a contradiction to the fact that $b_i \in R_i$. On the other hand, if $c_i \in R_i$, then $d_{\mathcal{S}_i}(c_i, v_{\pi_i}) > d_{\mathcal{S}_i}(c_i, v_{\pi_{i+1}})$, and we have $d_{\mathcal{S}_i}(a_i, v_{\pi_i}) > d_{\mathcal{S}_i}(a_i, v_{\pi_{i+1}})$, a contradiction to the fact $a_i \in L_i$. In both cases, we have a contradiction. Therefore we have $a_i \prec b_i$. □

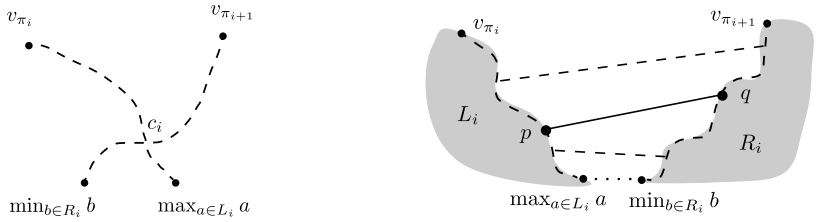



**Fig. 7** (**a**) A contradiction led by $\min_{b\in R_i} b \prec \max_{a\in L_i} a$. (**b**) Partition of $\mathcal{S}_i$ into $L_i$ and $R_i$

Let $\ell(v) := d_G(u, v)$ denote the distance between $v$ and the anchor $u$ in $G$. Let $\ell_i(v) := \min\{d_{\mathcal{S}_i}(v_{\pi_{(i)}}, v), d_{\mathcal{S}_i}(v_{\pi_{(i+1)}}, v)\}$, for any $1 \le i < deg(u)$ and $v \in \mathcal{S}_i$. Observe that $\ell(v) = \ell_i(v) + 1$, for any $1 \le i < deg(u)$ and $v \in \mathcal{S}_i$. Now consider the set of the edges connecting $L_i$ and $R_i$. Note that, this is exactly the set of edges connecting vertices on the shortest path between $v_{\pi_i}$ and $\max_{a\in L_i} a$ and vertices on the shortest path between $v_{\pi_{i+1}}$ and $\min_{b\in R_i} b$. We have the following lemma, which states that, when the vertices are classified by their distances to $u$, these edges can only connect vertices between neighboring sets and do not form any crossing. See also Fig. 7.

**Lemma 10** *For any edge $(p, q)$, $p \in L_i$, $q \in R_i$, connecting $L_i$ and $R_i$, we have*

- $|\ell(p) - \ell(q)| \le 1$, *and*
- $\nexists$ *edge* $(r, s), (r, s) \ne (p, q), r \in L_i, s \in R_i$, *such that* $\ell(r) = \ell(q)$ *and* $\ell(p) = \ell(s)$.

*Proof* The first half of the lemma follows from the definition of $\ell$. If $|\ell(p) - \ell(q)| > 1$, without loss of generality, suppose that $\ell(p) > \ell(q) + 1$, by going through $(p, q)$ then following the shortest path from $q$ to $u$, we find a shorter path for $p$, which is a contradiction. The second half follows from the fact that $\mathcal{E}$ is a planar embedding. □

Now we are ready to state the structural property.

**Lemma 11** *Any outer-planar graph $G = (V, E)$ together with an arbitrary vertex $u \in V$ is a general ladder anchored at $u$, where the set of vertices in each layer are classified by their distances to the anchor $u$.*

*Proof* We prove by induction on the number of vertices of $G$. First, an isolated vertex is a single-layer general ladder. For non-trivial graphs, let $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{deg(u)}$ be the subsets defined as above. By assumption, the induced subgraphs of $\mathcal{S}_0$ and $\mathcal{S}_{deg(u)}$ are general ladders with anchors $v_{\pi_1}$ and $v_{\pi_{deg(u)}}$, respectively. Furthermore, the layers are classified by $\ell - 1$. That is, vertex $v$ belongs to layer $\ell(v) - 1$. Similarly, the induced subgraphs of $L_i$ and $R_i$ are also general ladders with anchors $v_{\pi_i}$ and $v_{\pi_{i+1}}$ whose layers are classified by $\ell_i$.

Now we argue that these general ladders can be arranged properly to form a single general ladder with anchor $u$ and layers classified by $\ell$. Since there is no edge connecting $p$ and $q$ for any $p, q$ with $v_{\pi_{i-1}} \prec p \prec v_{\pi_i}$ and $v_{\pi_{j-1}} \prec q \prec v_{\pi_j}$,

$0 < i < j < deg(u)$, we only need to consider the edges connecting vertices between $L_i$ and $R_i$. By Lemma 10, when the general ladders $L_i$ and $R_i$ are hung over $v_{\pi_i}$ and $v_{\pi_{i+1}}$, respectively, the edges between them connect exactly only vertices from adjacent layers and do not form any crossing. Therefore, it forms a general ladder with $u$ being the anchor and the lemma follows. □

*5.1.1 Extracting the general ladder*

Let $\mathcal{G} = (V, E)$ be the input outer-planar graph and $u \in V$ be an arbitrary vertex. We can extract the corresponding general ladder in linear time.

**Theorem 7** *Given an outer-planar graph $\mathcal{G}$ and its outer-planar embedding, we can compute in linear time a general ladder representation for $\mathcal{G}$.*

*Proof* First we compute the shortest distance from each vertex $v \in V$ to $u$. Denote the shortest distance by $\ell(v)$. Since the number of edges in a planar graph is linear in the number of vertices, the computation can be done by a standard *breadth-first-search (BFS)* algorithm and takes time linear in the number of vertices.

Let $M = \max_{v \in V} \ell(v)$. We create $M+1$ empty queues, denoted $\text{layer}_0, \text{layer}_1, \ldots, \text{layer}_M$. The queues will be used to maintain the set of layers. We traverse the vertices of $\mathcal{G}$, starting from $u$, in a counter-clockwise order according to the given embedding. For each vertex $v$ visited, we attach $v$ to the end of $\text{layer}_{\ell(v)}$. The traversal of the outer face takes linear time. The correctness of this approach is assured by Lemma 11. □

For the rest of this paper we will denote the layers of this particular general ladder representation by $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_M$. The following additional structural property comes from the outer-planarity of $\mathcal{G}$ and our construction scheme.

**Lemma 12** *For any $0 < i \leq M$ and $v \in \mathcal{L}_i$, we have $|N(v) \cap \mathcal{L}_{i-1}| \leq 2$. Moreover, if $v$ has two neighbors in $\mathcal{L}_i$, say, $v_1$ and $v_2$ with $v_1 \prec v \prec v_2$, then there is an edge joining $v_1$ (and $v_2$, respectively) and each neighboring vertex of $v$ in $\mathcal{L}_{i-1}$ that is smaller (larger) than $v$.*

*Proof* First, since the layers are classified by the distances to the anchor $u$, if $|N(v) \cap \mathcal{L}_{i-1}| \geq 3$, then consider the shortest paths from vertices in $N(v) \cap \mathcal{L}_{i-1}$ to $u$. At least one vertex would be surrounded by another two paths, contradicting the fact that $\mathcal{G}$ is an outer-planar graph.

The second part is obtained from a similar argument. Let $v' \in N(v) \cap \mathcal{L}_{i-1}$ be a neighbor of $v$ in $\mathcal{L}_{i-1}$. If $v'$ is not joined to either $v_1$ or $v_2$, then consider the shortest paths from $u$ to $v_1$, $v'$, and $v_2$, respectively. $v'$ would be a vertex in the interior, which is a contradiction. □

*The Decomposition* The idea behind this decomposition is to help reduce the dependency between vertices of large degrees and their neighbors such that further techniques can be applied. To this end, we tackle the demands of vertices from every three layers separately.

For each $0 \le i < 3$, let $\mathcal{R}_i = \bigcup_{j \ge 0} \mathcal{L}_{3j+i}$. Let $\mathcal{G}_i = (V_i, E_i)$ consist of the induced subgraph of $\mathcal{R}_i$ and the set of edges connecting vertices in $\mathcal{R}_i$ to their neighbors. Formally, $V_i = \bigcup_{v \in \mathcal{R}_i} N[v]$ and $E_i = \bigcup_{v \in \mathcal{R}_i} \bigcup_{u \in N[v]} e(u, v)$. In addition, we set $d(v) = 0$ for all $v \in \mathcal{G}_i \backslash \mathcal{R}_i$. Other parameters remain unchanged.

**Lemma 13** *Let $f_i, 0 \le i < 3$, be an optimal demand assignment function for $\mathcal{G}_i$. The assignment function $f = \sum_{0 \le i < 3} f_i$ is a 3-approximation of $\mathcal{G}$.*

*Proof* First, for any vertex $v \in V$, the demand of $v$ is considered in $\mathcal{G}_i$ for some $0 \le i < 3$ and therefore is assigned by the assignment function $f_i$. Since we take the union of the three assignments, it is a feasible assignment to the entire graph $\mathcal{G}$.

Since the demand of each vertex in $\mathcal{G}_i$, $0 \le i < 3$, is no more than that of in the original graph $G$, any feasible solution to $G$ will also serve as a feasible solution to $\mathcal{G}_i$. Therefore we have $OPT(\mathcal{G}_j) \le OPT(G)$, for $0 \le j < 3$, and the lemma follows. □

## 5.2 A Linear Program for the Separable Demand Model

In the previous section, we decompose the input outerplanar graph $G$ into three subgraphs, i.e., $\mathcal{G}_i$, for $0 \le i < 3$. In this section, we formulate the problem as an integer linear program and derive basic properties. This allows us to further simply the subgraphs $\mathcal{G}_i$ and to obtain an approximation solution for each of them in the sections to follow.

An integer linear program (ILP) for capacitated domination with separable demand is given below in (2). The first inequality, i.e., Inequality (a), ensures the feasibility of the demand assignment function $f$ required in Definition 1. In the inequality (b), we model the multiplicity function $x$ as defined. The third constraint, Inequality (c), $d(v)x(u) - f(v, u) \ge 0$, which seems unnecessary in the problem formulation, is required to bound the integrality gap, which is the maximum ratio between the solution quality of the ILP and that of its relaxation [42]. The integrality gap of an ILP indicates the best approximation ratio one can possibly obtain through the relaxation of that ILP. To see that this additional constraint does not alter the optimality of any optimal solution, we prove the following Lemma 14.

$$\text{Minimize} \quad \sum_{u \in V} w(u)x(u) \tag{2}$$

subject to

$$\text{(a)} \quad \sum_{v \in N[u]} f(u, v) - d(u) \ge 0, \quad u \in V$$

$$\text{(b)} \quad c(u)x(u) - \sum_{v \in N[u]} f(v, u) \ge 0, \quad u \in V$$

$$\text{(c)} \quad d(v)x(u) - f(v, u) \ge 0, \quad v \in N[u], u \in V$$

$$\text{(d)} \quad f(u, v) \ge 0, \ \ x(u) \in \mathbb{Z}^+ \cup \{0\}, \quad u, v \in V$$

**Lemma 14** *Let $f$ be an arbitrary optimal demand assignment function. We have $d(v) \cdot x_f(u) - f(v,u) \geq 0$ for all $u \in V$ and $v \in N[u]$.*

*Proof* Without loss of generality, we may assume that $d(v) \geq f(v,u)$. Otherwise, we set $f(v,u)$ to be $d(v)$ and the resulting assignment would be feasible and the cost can only be better. If $x_f(u) = 0$, then we have $f(v,u) = 0$ by definition, and this inequality holds trivially. Otherwise, if $x_f(u) \geq 1$, then $d(v) \cdot x_f(u) - f(v,u) \geq f(v,u) \cdot (x_f(u) - 1) \geq 0$. □

However, without this constraint, the integrality gap can be arbitrarily large. This is illustrated by the following example. Let $\alpha > 1$ be an arbitrary constant, and $\mathcal{T}(\alpha)$ be an $n$-vertex star, where each vertex has unit demand and unit cost. The capacity of the central vertex is set to be $n$, which is sufficient to cover the demand of the entire graph, while the capacity of each of remaining $n-1$ petal vertices is set to be $\alpha n$.

**Lemma 15** *Without the additional constraint $d(v)x(u) - f(v,u) \geq 0$, the integrality gap of the ILP (2) on $\mathcal{T}(\alpha)$ is $\alpha$, where $\alpha > 1$ is an arbitrary constant.*

*Proof* The optimal dominating set consists of a single multiplicity of the central vertex with unit cost, while the optimal fractional solution is formed by spending $\frac{1}{\alpha n}$ multiplicity at a petal vertex for each unit demand from the vertices of this graph, making an overall cost of $\frac{1}{\alpha}$ and therefore an arbitrarily large integrality gap. □

Indeed, with the additional constraint applied, we can refrain from unreasonably assigning a small amount of demand to any vertex in any fractional solution. Take a petal vertex, say $v$, from $\mathcal{T}(\alpha)$ as example, given that $d(v) = 1$ and $f(v,v) = 1$, this constraint would force $x(v)$ to be at least 1, which prevents the aforementioned situation from being an optimal fractional solution.

For the rest of this paper, for any graph $G$, we denote the optimal values to the integer linear program (2) and to its relaxation by $OPT(G)$ and $OPT_f(G)$, respectively. Note that $OPT_f(G) \leq OPT(G)$.

### 5.3 Reducing the General Ladder

In this section we describe an approach to further simplifying the subgraphs $\mathcal{G}_i$ we obtained from the decomposition of the input graph $G$ proposed in Sect. 5.1, for $0 \leq i < 3$. Given any feasible demand assignment for $\mathcal{G}_i$, we can properly reassign the demand of a vertex to a constant number of neighbors while the increase in terms of fractional cost remains bounded.

For each $v \in \mathcal{R}_i$, we sort the closed neighbors of $v$ according to their cost in ascending order such that $w(\pi_v(1)) \leq w(\pi_v(2)) \leq \cdots \leq w(\pi_v(deg[v]))$, where $\pi_v : \{1, 2, \ldots, deg[v]\} \to N[v]$ is an injective function. For convenience, we implicitly extend the domain of $\pi_v$ and set $\pi_v(deg[v]+1) = \emptyset$ as a dummy entity. Suppose that $v \in \mathcal{L}_\ell$. We identify the following four vertices.

– Let $j_v$, $1 \leq j_v \leq deg[v]$, be the smallest integer such that $c(\pi_v(j_v)) > d(v)$. If $c(\pi_v(j)) \leq d(v)$ for all $1 \leq j \leq deg[v]$, then we let $j_v = deg[v] + 1$.

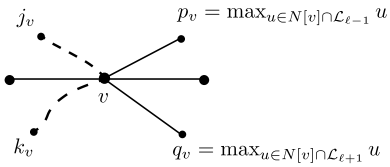



**Fig. 8** At most 6 incident edges for a vertex $v \in \mathcal{L}_\ell$ are to be kept in $\mathcal{H}_i$

– Let $k_v$, $1 \le k_v < j_v$, be the integer such that and $w(\pi_v(k_v))/c(\pi_v(k_v))$ is minimized. $k_v$ is defined only when $j_v > 1$.
– Let $p_v = \max_{u \in N[v] \cap \mathcal{L}_{\ell-1}} u$ and $q_v = \max_{u \in N[v] \cap \mathcal{L}_{\ell+1}} u$.

Intuitively, $\pi_v(j_v)$ is the first vertex in the sorted list whose capacity is greater than $d(v)$, and $\pi_v(k_v)$ is the vertex with best cost-capacity ratio among the first $j_v - 1$ vertices. $p_v$ and $q_v$ are the rightmost neighbors of $v$ in layer $\mathcal{L}_{\ell-1}$ and $\mathcal{L}_{\ell+1}$, respectively.

We will omit the function $\pi_v$ and use $j_v$, $k_v$ to denote $\pi_v(j_v)$, $\pi_v(k_v)$ without confusion. The reduced graph $\mathcal{H}_i$ is defined as follows. Denote the set of neighbors to be disconnected from $v$ by $R(v) = N[v] \backslash (\mathcal{L}_\ell \cup \{j_v \cup k_v \cup p_v \cup q_v\})$, and let $\mathcal{H}_i = \mathcal{G}_i \backslash \bigcup_{v \in \mathcal{R}_i} \bigcup_{u \in R(v)} \{e(u, v)\}$. Roughly speaking, in graph $\mathcal{H}_i$ we remove the edges which connect vertices in $\mathcal{R}_i$, say $v$, to vertices not in $\mathcal{R}_i$, except possibly for $j_v$, $k_v$, $p_v$, and $q_v$. See Fig. 8. Note that, although our reassigning argument applies to arbitrary graphs, we can remove the edge between them only when two vertices are unimportant to each other.

Recall that $OPT_f(\cdot)$ denotes the value of the optimal fractional solution for the graph it refers to. We have the following lemma.

**Lemma 16** *In the subgraph $\mathcal{H}_i$, we have*

1. *For each $v \notin \mathcal{R}_i$, at most one incident edge of $v$ which was previously in $\mathcal{G}_i$ will be removed.*
2. *For each $v \in \mathcal{R}_i$, the degree of $v$ in $\mathcal{H}_i$ is upper-bounded by* 6.
3. $OPT_f(\mathcal{H}_i) \le 2 \cdot OPT_f(\mathcal{G}_i)$.

*Proof* For the first part, let $v \notin \mathcal{R}_i$ be a vertex and denote $\mathcal{S} = N[v] \cap \mathcal{R}_i$ the set of neighbors of $v$ that are in $\mathcal{R}_i$. By the definition of general ladders, for any $u \in \mathcal{S}$, $u \ne \max_{a \in \mathcal{S}} a$, we have either $p_u = v$ or $q_u = v$, since $v$ serves as the rightmost neighbor of $u$. Therefore, by our approach, only the edge between $v$ and $\max_{a \in \mathcal{S}} a$ will possibly be removed.

For the second part, for any $v \in \mathcal{R}_i$, $v$ has at most two neighbors in the same layer, since each layer is a subgraph of an ordered path. We have removed all the edges connecting $v$ to vertices not in the same layer, except for at most four vertices, $j_v$, $k_v$, $p_v$, and $q_v$. Therefore $deg(v) \le 6$.

Now we prove the third part of this lemma. Let $f_{\mathcal{G}_i}$ be an optimal demand assignment for $\mathcal{G}_i$, and $x_{\mathcal{G}_i}$ be the corresponding multiplicity function. Note that, from the second and the third inequalities of (2), for each $v \in V$ and $u \in N[v]$, we have

$$x_{\mathcal{G}_i}(u) \ge \max\left\{ \frac{f_{\mathcal{G}_i}(v,u)}{d(v)}, \frac{f_{\mathcal{G}_i}(v,u)}{c(u)}. \right\} \tag{3}$$

For each $v \in \mathcal{R}_i$ and $u \in R(v)$ such that $f_{\mathcal{G}_i}(v,u) \neq 0$, we modify this assignment as follows. If $\pi_v^{-1}(u) \geq j_v$, then we assign it to $j_v$ instead of to $u$. Otherwise, we assign it to $k_v$. That is, depending on whether $\pi_v^{-1}(u) \geq j_v$, we raise either $f_{\mathcal{G}_i}(v, j_v)$ or $f_{\mathcal{G}_i}(v, k_v)$ by the amount of $f_{\mathcal{G}_i}(v,u)$ and then set $f_{\mathcal{G}_i}(v,u)$ to be zero. Note that, after this reassignment, the modified assignment function $f_{\mathcal{G}_i}$ will be a feasible assignment for $\mathcal{H}_i$ as well.

In order to cope with this change, $x_{\mathcal{G}_i}(j_v)$ or $x_{\mathcal{G}_i}(k_v)$ might have to be raised as well until both the second and the third inequalities are valid again. If $\pi_v^{-1}(u) \geq j_v$, then $x_{\mathcal{G}_i}(j_v)$ is raised by at most

$$\max\left\{ f_{\mathcal{G}_i}(v,u)/d(v),\, f_{\mathcal{G}_i}(v,u)/c(j_v) \right\},$$

which is equal to $f_{\mathcal{G}_i}(v,u)/d(v)$, since $c(j_v) > d(v)$. Hence the total cost will be raised by at most

$$w(j_v) \cdot \frac{f_{\mathcal{G}_i}(v,u)}{d(v)} \leq w(j_v) \cdot \max\left\{ \frac{f_{\mathcal{G}_i}(v,u)}{d(v)}, \frac{f_{\mathcal{G}_i}(v,u)}{c(u)} \right\} \leq w(u) \cdot x_{\mathcal{G}_i}(u),$$

by Inequality (3) and the fact that $w(j_v) \leq w(u)$. Similarly, if $\pi_v^{-1}(u) < j_v$, the cost is raised by at most

$$\begin{aligned} w(k_v) \cdot \max\left\{ \frac{f_{\mathcal{G}_i}(v,u)}{d(v)}, \frac{f_{\mathcal{G}_i}(v,u)}{c(k_v)} \right\} &= w(k_v) \cdot \frac{f_{\mathcal{G}_i}(v,u)}{c(k_v)} \\ &\leq w(u) \cdot \frac{f_{\mathcal{G}_i}(v,u)}{c(u)} \leq w(u) \cdot x_{\mathcal{G}_i}(u), \end{aligned}$$

since we have $\frac{w(k_v)}{c(k_v)} \leq \frac{w(u)}{c(u)}$ by definition of $k_v$ and Inequality (3).

In both cases, the extra cost required by this specific demand reassignment between $v$ and $u$ is bounded by $w(u) \cdot x_{\mathcal{G}_i}(u)$. By the first part of this lemma, we have at most one such pair for each $u \notin \mathcal{R}_i$, the overall cost is at most doubled and this lemma follows. □

We also remark that, although $OPT_f(\mathcal{H}_i)$ is bounded in terms of $OPT_f(\mathcal{G}_i)$, an $\alpha$-approximation for $\mathcal{H}_i$ is not necessarily a $2\alpha$-approximation for $\mathcal{G}_i$. That is, having an approximation $\mathcal{A}$ with $OPT(\mathcal{A}) \leq \alpha \cdot OPT(\mathcal{H}_i)$ does not imply that $OPT(\mathcal{A}) \leq 2\alpha \cdot OPT(\mathcal{G}_i)$, for $OPT(\mathcal{H}_i)$ could be strictly larger than $OPT_f(\mathcal{H}_i)$. Instead, to obtain our claimed result, an approximation with a stronger bound, in terms of $OPT_f(\mathcal{H}_i)$, is desired.

### 5.4 Greedy Charging Scheme

We show how we can further approximate the optimal solution for the reduced graph $\mathcal{H}_i$ by a sophisticated primal-dual charging argument. Let $V_i$ be the vertex set of $\mathcal{H}_i$. We apply a technique from [33] to obtain a feasible solution for the dual program of the relaxation of the ILP (2) from Sect. 5.2, which is given below in (4).

It is not difficult to see that the linear program (4) is exactly a *packing program*, i.e., a linear program whose objective is to pack "items" subject to certain "space"

constraints. Moreover, it is well-known from the linear program duality [42] that, the process of selecting the multiplicities of a *covering program*, i.e., program (2), corresponds to the process of packing items in the dual packing program, i.e., program (4).

The main idea we use here is that, through the process of *dual fitting* [42], whenever we decide to increase the multiplicity of a vertex and make a demand assignment from the neighboring vertices, the cost we spend can always be distributed to each unit of demand served by this assignment. The amount of charge each unit demand receives depends on the value of the dual variable that corresponds to the vertex for which the demand comes from. Then, by the structural property we provide in Lemma 12, we show that the approximation ratio can be further tightened.

$$\text{Maximize} \quad \sum_{u \in V} d(u) y_u \tag{4}$$

subject to

$$c(u) z_u + \sum_{v \in N[u]} d(v) g_{u,v} \leq w(u), \quad u \in V$$

$$y_u \leq z_v + g_{v,u}, \quad v \in N[u],\ u \in V$$

$$y_u \geq 0,\ z_u \geq 0,\ g_{v,u} \geq 0, \quad v \in N[u],\ u \in V$$

We first describe an approach to obtaining a feasible solution to (4) and how a corresponding feasible demand assignment can be found. Note that any feasible solution to (4) will serve as a lower bound to any feasible solution of (2) by the linear program duality.

During the process, we will maintain a vertex subset, $V^\phi$, which contains the set of vertices with non-zero unassigned demand. For each $u \in V_i$, let $d^\phi(u) = \sum_{v \in N[u] \cap V^\phi} d(v)$ denote the amount of unassigned demand from the closed neighbors of $u$. We distinguish between two cases. If $c(u) < d^\phi(u)$, then we say that $u$ is heavily-loaded. Otherwise, $u$ is lightly-loaded. During the process, some heavily-loaded vertices might turn into lightly-loaded due to the demand assignments of its closed neighbors. For each of these vertices, say $v$, we will maintain a vertex subset $D^*(v)$, which contains the set of unassigned vertices in $N[v] \cap V^\phi$ when $v$ is about to fall into lightly-loaded. For other vertices, $D^*(v)$ is defined to be the empty set.

Initially, $V^\phi = \{u : u \in V_i, d(u) \neq 0\}$ and all the dual variables are set to be zero. We increase the dual variable $y_u$ simultaneously, for each $u \in V^\phi$. To maintain the dual feasibility, as we increase $y_u$, we have to raise either $z_v$ or $g_{v,u}$, for each $v \in N[u]$. If $v$ is heavily-loaded, then we raise $z_v$. Otherwise, we raise $g_{v,u}$. Note that, during this process, for each vertex $u$ that has a closed neighbor in $V^\phi$, the left-hand side of the inequality $c(u)z_u + \sum_{v \in N[u]} d(v) g_{u,v} \leq w(u)$ is constantly raising. As soon as one of the inequalities $c(u)z_u + \sum_{v \in N[u]} d(v) g_{u,v} \leq w(u)$ is met with equality (saturated) for some vertex $u \in V_i$, we perform the following operations.

If $u$ is lightly-loaded, we assign all the unassigned demand from $N[u] \cap V^\phi$ to $u$. In this case, there are still $c(u) - d^\phi(u)$ units of capacity free at $u$. We assign the

ALGORITHM *Greedy-Charging*

$V^\phi \longleftarrow \{u : u \in V_i, d(u) \neq 0\}$, $V^* \longleftarrow V_i$.
$d^\phi(u) \longleftarrow \sum_{v \in N[u]} d(v)$, for each $u \in V_i$.
$w^\phi(u) \longleftarrow w(u)$, for each $u \in V$.
Let $\mathcal{Q}$ be a first-in-first-out queue.
**while** $V^\phi \neq \emptyset$ **do**
    $r_v \longleftarrow w^\phi(v)/\min\{c(v), d^\phi(v)\}$, for each $v \in V^*$.
    $u \longleftarrow \arg\min\{r_v : v \in V^*\}$. [$u$ is the next vertex to be saturated.]
    $w^\phi(v) \longleftarrow w^\phi(v) - w^\phi(u)$, for each $v \in V^*$.

    **if** $d^\phi(u) \leq c(u)$ **then**
        Assign the demand from $N[u] \cap V^\phi$ to $u$.
        Assign $c(u) - d^\phi(u)$ amount of unassigned demand from $D^*(u)$, if there is any, to $u$.
    **else**
        Attach $u$ to the queue $\mathcal{Q}$ and mark $u$ as *heavy*.
    **end if**
    Let $\mathcal{S}_u \longleftarrow (N[u] \cap V^\phi) \cup \{u\}$.
    Remove $N[u]$ from $V^\phi$ and update the corresponding $d^\phi(v)$ for $v \in V_i$.
    For each $v \in V^*$ such that $d^\phi(v) = 0$, remove $v$ from $V^*$.
    **for all** vertex $v$ becomes lightly-loaded in this iteration **do**
        $D^*(v) \longleftarrow N[v] \cap (V^\phi \cup \mathcal{S}_u)$.
    **end for**
**end while**
**while** $\mathcal{Q} \neq \emptyset$ **do**
    Extract a vertex from the head of $\mathcal{Q}$, say $u$.
    Assign the unassigned demand from $N[u]$ to $u$.
**end while**

**Fig. 9** A high-level pseudo-code for the primal-dual algorithm

unassigned demand from $D^*(u)$, if there is any, to $u$ until either all the demand from $D^*(u)$ is assigned or all the free capacity in $u$ is used. On the other hand, if $u$ is heavily-loaded, we mark it as heavy and delay the demand assignment from its closed neighbors.

Then we set $\mathcal{Q}_u = N[u] \cap V^\phi$ and remove $N[u]$ from $V^\phi$. Note that, due to the definition of $d^\phi$, even when $u$ is heavily-loaded, we still update $d^\phi(p)$ for each $p \in V_i$ with $N[p] \cap N[u] \neq \emptyset$, if needed, as if the demand was assigned. During the above operation, some heavily-loaded vertices might turn into lightly-loaded due to the demand assignments (or simply due to the update of $d^\phi$). For each of these vertices, say $v$, we set $D^*(v) = N[v] \cap (V^\phi \cup \mathcal{Q}_u)$. Intuitively, $D^*(v)$ contains the set of unassigned vertices from $N[v] \cap V^\phi$ when $v$ is about to fall into the lightly-loaded vertices.

This process continues until $V^\phi = \emptyset$. For those vertices which are marked as heavy, we iterate over them according to their chronological order of being saturated and assign at this moment all the remaining unassigned demand from their closed neighbors to them. A high-level description of this algorithm is provided in Fig. 9.

Let $f^* : V \times V \to R^+ \cup \{0\}$ denote the resulting demand assignment function, and $x^* : V \to Z^+ \cup \{0\}$ denotes the corresponding multiplicity function. The following lemma bounds the cost of the solution produced by our algorithm.

**Lemma 17** *For any $\mathcal{H}_i$ obtained from a general ladder $\mathcal{G}_i$, we have $w(f^*) \leq 7 \cdot OPT_f(\mathcal{H}_i)$.*

*Proof* We argue in the following that, for each $u \in V_i$, the cost resulted by $u$, which is $w(u) \cdot x^*(u)$, can be distributed to a certain portion of the demands from the vertices in $N[u]$ such that each unit demand, say from vertex $v \in N[u]$, receives a charge of at most $7 \cdot y_v$.

Let $u \in V_i$ be a vertex with $x^*(u) > 0$. If $u$ has been marked as heavy, then by our scheme, we have $g_{u,v} = 0$ and $y_v = z_u$ for all $v \in N[u]$. Therefore $w(u) = c(u) \cdot z_u = c(u) \cdot y_v$, and for each multiplicity of $u$, we need $c(u)$ units of demand from $N[u]$. If $x^*(u) > 1$, then at least $c(u) \cdot (x^*(u) - 1)$ units of demand are assigned to $u$, and by distributing the cost to them, each unit of demand gets charged at most twice. If $x^*(u) = 1$, then we charge the cost to any $c(u)$ units of demand that are counted in $d^\phi(u)$ when $u$ is saturated. Since $u$ is a heavily-loaded vertex, $d^\phi(u) > c(u)$ and there will be sufficient amount of demand to charge.

If $u$ is lightly-loaded, then $x^*(u) = 1$ and we have two cases. First, if $\sum_{v \in N[u]} d(v) \leq c(u)$, then $u$ is lightly-loaded in the beginning and we have $z_u = 0$ and $y_v = g_{u,v}$ for each $v \in N[u]$, which implies

$$w(u) = \sum_{v \in N[u]} d(v) \cdot g_{u,v} = \sum_{v \in N[u]} d(v) \cdot y_v.$$

The cost $w(u)$ of $u$ can be distributed to all the demand from its closed neighbors, each unit demand, say from vertex $v \in N[u]$, gets a charge of $y_v$.

If $\sum_{v \in N[u]} d(v) > c(u)$, then $u$ is heavily-loaded in the beginning and at some point turned into lightly-loaded. Let $\mathcal{U}_0 \subseteq D^*(u)$ be the set of vertices whose removal from $V^\phi$ achieves this change. By our scheme, $z_u$ is raised in the beginning and at some point when $d^\phi(u)$ is about to fall below $c(u)$, we fix $z_u$ and start raising $g_{u,v}$ for $v \in D^*(u) \backslash \mathcal{U}_0$. Note that we have

$$\begin{cases} y_{u_0} = z_u & \text{for all } u_0 \in \mathcal{U}_0, \\ y_v = z_u + g_{u,v} & \text{for each } v \in D^*(u) \backslash \mathcal{U}_0, \quad \text{and} \\ g_{u,v} = 0 & \text{for } v \in (N[u] \backslash D^*(u)) \cup \mathcal{U}_0. \end{cases}$$

Let $d^*_{\mathcal{U}_0} = c(u) - \sum_{v \in D^*(u) \backslash \mathcal{U}_0} d(v)$. We have

$$\begin{aligned} w(u) &= c(u) \cdot z_u + \sum_{v \in N[u]} d(v) \cdot g_{u,v} \\ &= \left( d^*_{\mathcal{U}_0} + \sum_{v \in D^*(u) \backslash \mathcal{U}_0} d(v) \right) \cdot z_u + \sum_{v \in D^*(u)} d(v) \cdot g_{u,v} \\ &= d^*_{\mathcal{U}_0} \cdot z_u + \sum_{v \in D^*(u) \backslash \mathcal{U}_0} d(v) \cdot (z_u + g_{u,v}) \\ &\leq \sum_{v \in \mathcal{U}_0} d_v \cdot y_v + \sum_{v \in D^*(u) \backslash \mathcal{U}_0} d(v) \cdot y_v. \end{aligned}$$

Again the cost $w(u)$ of the single multiplicity can be distributed to the demand of vertices in $D^*(u)$.

Finally, for each unit demand, say demand $d$ from vertex $u$, consider the set of vertices $V_d \subseteq N[u]$ that has charged $d$. First, by our assigning scheme, $V_d$ consists of at most one heavily-loaded vertex. If $d$ is assigned to a heavily-loaded vertex, then, by our charging scheme, we have $|V_d| = 1$, and $d$ is charged at most twice. Otherwise, if $d$ is assigned to a lightly-loaded vertex, then, by our charging scheme, each vertex in $V_d$ charges $d$ at most once, disregarding heavily-loaded or lightly-loaded vertices.

By the above argument and Lemma 16, we have

$$w(f^*) = \sum_{u \in V} w(u)x^*(u) \leq \sum_{u \in V} d(u) \cdot deg[u] \cdot y_u$$
$$\leq 7 \cdot \sum_{u \in V} d(u) \cdot y_u \leq 7 \cdot OPT_f(\mathcal{H}_i),$$

where the last inequality follows from the linear program duality of (2) and (4). □

By the structural property provided in Lemma 12 and the fact that the input graph is outerplanar, we can modify the algorithm slightly and further improve the bound given in the previous lemma. To this end, we consider the situation when a unit demand from a vertex $u$ with $deg[u] = 7$ gets charged and argue that, either the unit demand is not charged by all of its closed neighbors, or we can modify the demand assignment function, without raising the cost, to satisfy this condition.

**Lemma 18** *Provided the fact that $\mathcal{H}_i$ comes from an outerplanar graph, we can modify the algorithm to obtain a demand assignment function $f^*$ such that $w(f^*) \leq 6 \cdot OPT_f(\mathcal{H}_i)$.*

*Proof* Consider any unit demand, say demand $d$ from vertex $u$ in $\mathcal{L}_j$, and let $V_d \subseteq N[u]$ be the set of vertices that has charged $d$ by our charging scheme.

First, we have $|N[u]| \leq 7$ by Lemma 16. By our charging scheme, $|N[u]| < 7$ implies $|V_d| < 7$. Hence, it suffices to consider the case $|N[u]| = 7$ and argue that either we have $|V_d| < 7$, or we can modify the demand assignment function such that $|V_d| < 7$ holds. Recall that in Sect. 5.3, for vertex $u$, only the following neighboring vertices are kept in $\mathcal{H}_i$.

- $j_u$, the neighboring vertex of $u$ with the smallest capacity such that $c(j_u) > d(u)$. If $c(v) \leq d(u)$ for all $v \in N[u]$, then $j_u$ is set to be $deg[u] + 1$.
- $k_u$, the neighboring vertex of $u$ such that $w(k_u)/c(k_u)$ is minimized. $k_u$ is defined only when $j_u$ is not the vertex with the smallest capacity among $N[u]$.
- $p_v = \max_{u \in N[v] \cap \mathcal{L}_{\ell-1}} u$.
- $q_v = \max_{u \in N[v] \cap \mathcal{L}_{\ell+1}} u$.
- The neighboring vertices of $u$ in $\mathcal{L}_j$.

Let $u_1, u_2 \in N(u) \cap \mathcal{L}_j$, $u_1 \prec u \prec u_2$, denote the two neighbors of $u$ in $\mathcal{L}_j$. By our assumption that $|N[u]| = 7$, the vertices $u, u_1, u_2, j_u, k_u, p_u$, and $q_u$ are well-defined and mutually non-identical. By Lemma 12, we know that $|N(u) \cap \mathcal{L}_{j-1}| \leq 2$. Since

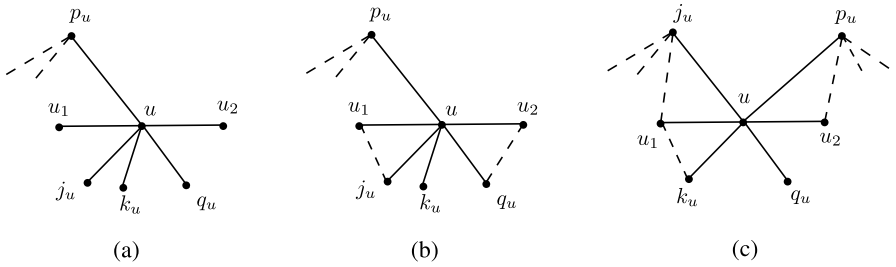



**Fig. 10** Situations when a unit demand of $u$ is fully-charged

$p_u \in \mathcal{L}_{j-1}$ by definition, we know that at most one of $j_u$ and $k_u$ can belong to $\mathcal{L}_{j-1}$. Hence, depending on the layers to which $j_u$ and $k_u$ belong, we only have the following two cases to consider.

*Case (a): Both $j_u$ and $k_u$ belong to $\mathcal{L}_{j+1}$.* If two of $\{j_u, k_u, q_u\}$, say, $j_u$ and $k_u$, are not joined to $u_1$ and $u_2$ by any edge, then at most one of $j_u$ and $k_u$ can charge $d$, since $u$ is the only vertex with possibly non-zero demand in their closed neighborhoods. When the first closed neighbor of $u$ is saturated and $u$ is removed from $V^\phi$, both $j_u$ and $k_u$ will be removed from $V^*$ and will not be picked in later iterations. Therefore, at most one of $j_u$ and $k_u$ can charge $d$.

On the other hand, if two of $\{j_u, k_u, q_u\}$, say $j_u$ and $q_u$, are joined to $u_1$ and $u_2$, respectively, then we argue that at most two out of $\{j_u, u, q_u\}$ can charge $d$. Indeed, $u_1$, $u$, and $u_2$ are the only vertices with non-zero demands in the closed neighborhoods of $\{j_u, u, q_u\}$. After two of $\{j_u, u, q_u\}$ is saturated, $u_1$, $u$, and $u_2$ will be removed from $V^\phi$. Therefore, at most two out of $\{j_u, u, q_u\}$ can charge $d$. See also Figs. 10(a) and 10(b).

*Case (b): Only one of $\{j_u, k_u\}$ belongs to $\mathcal{L}_{j+1}$ and the other belongs to $\mathcal{L}_{j-1}$.* Without loss of generality, we assume that $j_u \in \mathcal{L}_{j-1}$ and $k_u \in \mathcal{L}_{j+1}$. By Lemma 12, both $j_u$ and $p_u$ are joined either to $u_1$ or $u_2$ separately. Since $\mathcal{H}_i$ is outerplanar, we have $j_u \prec u \prec p_u$, otherwise $j_u$ will be contained inside the face surrounded by $p_u$, $u_1$, and $u$, which is a contradiction. See also Fig. 10(c).

*Case (b.1):* If both $k_u$ and $q_u$ are not joined to either $u_1$ or $u_2$, then by an argument similar to the one that we used in the previous case, at most one of $k_u$ and $q_u$ can charge $d$.

*Case (b.2):* Suppose that one of $\{k_u, q_u\}$, say, $k_u$, is joined to $u_1$ by an edge. We have two further subcases to consider.

– *Case (b.2a):* We argue that, if both $u_1$ and $k_u$ have charged $d$ after $d$ has been assigned in a feasible solution returned by our algorithm, then we can cancel the multiplicity placed on $k_u$ and reassign to $u_1$ the demand which was previously assigned to $k_u$ without increasing the cost spent on $u_1$.

  If $u_1$ is lightly-loaded in the beginning, then the above operation can be done without extra cost. Otherwise, observe that, in this case, $u_1$ must have been lightly-loaded when $d$ is assigned so that it can charge $d$ later. Moreover, $u_1$ is also in the set $V^\phi$ and not yet served, for otherwise $k_u$ will be removed from $V^*$ and will not be picked later. In other words, at the moment when $u_1$ becomes lightly-loaded,

we have $u_1 \in V^\phi$, meaning that it is possible to assign the demand of $u_1$ to itself later without extra cost.

– *Case (b.2b):* Conversely, if $u_1$ or $k_u$ is the first one to charge $d$, consider the relation between $q_u$ and $u_2$. If there is no edge between $q_u$ and $u_2$, then $q_u$ will not be picked and hence will not charge $d$. Otherwise, if the edge $(q_u, u_2)$ exists in $\mathcal{H}_i$, then the situation is symmetric to *case (b.2a)* where $k_u$ is joined by an edge to $u_1$ and none of $k_u$ and $u_1$ is the first vertex to charge $d$.

In both cases, either we have $|V_d| \leq 6$ or we can modify the demand assignment to make $|V_d| \leq 6$ hold. Therefore we have $w(f^*) \leq 6 \cdot OPT_f(\mathcal{H}_i)$ as claimed. □

### 5.5 Overall Analysis

We summarize the whole algorithm and our main theorem. Given an outer-planar graph $G = (V, E)$, we use the algorithm described in Sect. 5.1 to compute a general ladder representation of $G$, followed by applying the decomposition to obtain three subproblems, $\mathcal{G}_0$, $\mathcal{G}_1$, and $\mathcal{G}_2$.

For each $\mathcal{G}_i$, we use the approach described in Sect. 5.3 to further remove more edges and obtain the reduced subgraph $\mathcal{H}_i$. For each of the reduced subgraph $\mathcal{H}_i$, we apply the algorithm described in Sect. 5.4 to obtain an approximation, which is a demand assignment function $f_i$ for $\mathcal{H}_i$. The overall approximation, e.g., the demand assignment function $f$, for $G$ is defined as $f = \sum_{0 \leq i < 3} f_i$.

**Theorem 8** *Given an outerplanar graph $G$ as an instance of capacitated domination, one can compute a constant-factor approximation for $G$ in $O(n^2)$ time.*

*Proof* First, we argue that the procedures we describe can be done in $O(n^2)$ time. It takes $O(n \log^3 n)$ time to compute an outer-planar representation of $G$ [7]. By Theorem 7, computing a general ladder representation takes linear time. The construction of $\mathcal{G}_i$ takes time linear in the number of edges, which is linear in $n$ since $G$ is planar.

In the construction of the reduced graphs $\mathcal{H}_i$, for each vertex $v$, although we use a sorted list of the closed neighborhood of $v$ to define $j_v$, $k_v$, $p_v$, and $q_v$, the sorted lists are not necessary and $\mathcal{H}_i$ can be constructed in $O(n)$ time by a careful implementation. This is done in a two-pass traversal on the set of edges of $\mathcal{G}_i$ as follows. In the first pass, we iterate over the set of edges to locate $j_v$, $p_v$, and $q_v$, for each vertex $v \in V$. Specifically, we keep a current candidate for each vertex and for each edge $(u, v) \in E$ iterated, we make an update on $u$ and $v$ if necessary. In the second pass, based on the $j_v$ computed for each $v \in V$, we iterate over the set of edges again to locate $k_v$. The whole process takes time linear in the number of edges, which is $O(n)$.

In the following, we explain how the primal-dual algorithm, i.e., the algorithm presented in Fig. 9, can be implemented to compute a feasible solution in $O(n^2)$ time. First, we traverse the set of edges in linear time to compute the value $d^\phi(v)$ for each vertex. In each iteration, the next vertex to be saturated, which is the one with minimum $w^\phi(v)/\min\{c(v), d^\phi(v)\}$, can be found in linear time. The update of $w^\phi(v)$ for each $v \in V^*$ described in line 8 can be done in linear time. When a vertex $v \in N[u]$ with non-zero demand is removed from $V^\phi$, we have to update the value

$d^{\phi}(v')$ for all $v' \in N[v]$. By Lemma 16, the closed degree of such vertices is bounded by 7. This update can be done in $O(1)$ time. The construction of $\mathcal{S}_u$ can be done in linear time. Since $d^{\phi}(v)$ can only decrease, each vertex can turn into lightly-loaded at most once. Therefore the overall processing time for these vertices is linear in the number of vertices. The outer-loop iterates at most $O(n)$ times. Therefore the whole algorithm runs in $O(n^2)$ time.

The feasibility of the demand assignment function $f$ is guaranteed by Lemma 13 and the fact that $\mathcal{H}_i$ is a subgraph of $\mathcal{G}_i$. Since $\mathcal{H}_i \subseteq \mathcal{G}_i$, the demand assignment we obtained for $\mathcal{H}_i$ is also a feasible demand assignment for $\mathcal{G}_i$. Therefore, $f$ is feasible for $G$.

Putting everything together and we have

$$
\begin{aligned}
w(f) &\leq \sum_{0 \leq i < 3} w(f_i) && \text{Definition of } f,\\
&\leq 6 \cdot \sum_{0 \leq i < 3} OPT_f(\mathcal{H}_i) && \text{Lemma 18},\\
&\leq 12 \cdot \sum_{0 \leq i < 3} OPT_f(\mathcal{G}_i) && \text{Lemma 16},\\
&\leq 12 \cdot \sum_{0 \leq i < 3} OPT(\mathcal{G}_i) && \text{ILP relaxation},\\
&\leq 36 \cdot OPT(G) && \text{Lemma 13}.
\end{aligned}
$$

□

## 6 Planar Graphs

In this section, we show how the algorithms presented above can be extended to obtain approximation algorithms for planar graphs under a general framework due to [3]. In Sect. 6.1 we sketch the framework and describe how the algorithms provided in Sect. 4.2 can lead to a pseudo-polynomial-time approximation scheme for planar graphs.

Regarding polynomial-time approximation algorithms, we prove in Sect. 6.2 that approximating this problem on planar graphs with inseparable demands to a factor smaller than $\frac{3}{2} - \epsilon$ is weakly NP-hard, for any $\epsilon > 0$. For the separable demand model, we show in Sect. 6.3 how the algorithm presented in Sect. 5 can be adopted to obtain a constant-factor approximation for planar graphs.

### 6.1 The Framework

Let $G = (V, E)$ be a planar graph. We generate a planar embedding using the linear-time algorithm of Hopcroft and Tarjan [28]. To help present the framework, we define the levels of the vertices with respect to this embeddings as follows. By level 1 we mean the set of vertices the lie on the unbounded face. For any $j > 1$, we refer level $j$ to as the set of vertices on the unbounded face after the removal of vertices in levels $1, 2, \ldots, j-1$.

Let $m$ be the number of levels of this embedding, $OPT$ be the cost of the optimal demand assignment of $G$, and $OPT_j$ be the cost contributed by vertices at level $j$. For the ease of presentation, for any $j$ with $j \leq 0$ or $j > m$, the set of vertices in level $j$ is defined to be an empty set, and the corresponding cost $OPT_j$ is defined to be zero.

Let $k \geq 3$ be a constant to be determined. For $0 \leq i < k$, we define $C_i$ as

$$\sum_{0 \leq j \leq \frac{m}{k}} (OPT_{k \cdot j - i} + OPT_{k \cdot (j+1) - i - 1}).$$

Since $\sum_{0 \leq i < k} C_i \leq 2 \cdot OPT$, there exists an $i_0$ with $0 \leq i_0 < k$ such that $C_{i_0} \leq \frac{2}{k} \cdot OPT$. For each $0 \leq j \leq \frac{m}{k}$, define the graph $G_j$ to be the graph induced by vertices between level $k \cdot j - i_0$ and level $k \cdot (j+1) - i_0 - 1$. The parameters of the vertices in $G_j$ are set as follows. For those vertices which are from level $k \cdot j - i_0$ and level $k \cdot (j+1) - i_0 - 1$, their demands are set to be zero. The rest of the parameters remain unchanged. Clearly, $G_j$ is a $k$-outerplanar graph, which has treewidth at most $3k - 1$ [3–5, 30], and we have $\sum_{0 \leq j \leq \frac{m}{k}} OPT(G_j) \leq (1 + \frac{2}{k}) \cdot OPT$, where $OPT(G_j)$ is the cost of the optimal demand assignment of $G_j$.

The following theorem follows directly from Theorem 6 in Sect. 4.2, Corollary 2 in Sect. 4.2, Lemma 5 in Sect. 3.3, and the decomposition described above.

**Theorem 9** *When the input graph is planar, one can compute the following approximations for the capacitated domination problem in pseudo-polynomial time.*

- $(1 + \frac{2}{k})$*-approximation in* $O(2^{6k(\log M + 1) + \log 3k} \cdot n)$ *time when the demand is inseparable, where* $M$ *is the maximum capacity of the vertex set.*
- $(1 + \frac{2}{k})$*-approximation in time* $O(2^{(2M+2N+1)\log 3k} \cdot n)$ *and* $(2 + \frac{2}{k})$*-approximation in time* $O(2^{(4M+1)\log 3k} \cdot n)$*, when the demand is separable, where* $M$ *is the maximum capacity and* $N$ *is the maximum demand of the vertex set.*

Theorem 9 gives a parameterized approximation for both problem models in the sense that it allows a trade-off between the quality of the approximation results and the running time of the algorithm.

### 6.2 A Lower Bound of $(\frac{3}{2} - \epsilon)$ on the Approximation Factor for the Inseparable Demand Model

Below we show that, when the demand is inseparable, it is weakly NP-hard to approximate this problem within a factor of $(\frac{3}{2} - \epsilon)$. The reduction is made from *Partition*, which is a well-known combinatorial problem that is known to be weakly NP-hard [21].

**Definition 9** (Partition problem) Given a sequence of positive integers $a_1, a_2, \ldots, a_n$, the partition problem asks to decide whether or not there exists a subset $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in \mathcal{A}} a_i = \sum_{1 \leq i \leq n, i \notin \mathcal{A}} a_i$.

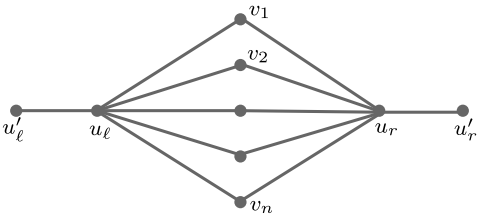



**Fig. 11** The construction of the reduction from *Partition*

Given a problem instance $\mathcal{I} = \{a_1, a_2, \ldots, a_n\}$ of *Partition*, we construct a planar graph $\mathcal{G}$ with $n + 4$ vertices as follows. For each $1 \le i \le n$, we create a vertex $v_i$ with demand $a_i$ and capacity 1. We create two vertices $u_\ell$ and $u_r$ with demand 1 and capacity $\sum_{1 \le i \le n} a_i + 1$, and connect $u_\ell$ and $u_r$ to each $v_i$, $1 \le i \le n$. We create two additional vertices $u'_\ell$ and $u'_r$, which are connected to $u_\ell$ and $u_r$, respectively, with demand $\frac{1}{2} \sum_{1 \le i \le n} a_i$ and capacity 1. The cost for each vertex is set to be 1. See also Fig. 11 for an illustration.

**Theorem 10** *For any $\epsilon > 0$, there exists no approximation algorithm which approximates the capacitated domination problem with inseparable demand on planar graphs to the factor of $(\frac{3}{2} - \epsilon)$, unless $P = NP$.*

*Proof* In the following, we argue that there exists a subset $\mathcal{A}$ satisfying the criterion of the partition problem if and only if the cost of the optimal demand assignment of $\mathcal{G}$ is at most 2. As a consequence, any algorithm that approximates $\mathcal{G}$ within the factor of $\frac{3}{2} - \epsilon$ would imply the correct decision for $\mathcal{I}$.

If there exists a subset $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in \mathcal{A}} a_i = \sum_{1 \le i \le n, i \notin \mathcal{A}} a_i$, then we assign the demand of $u'_\ell$ and $v_i$, for each $i \in \mathcal{A}$ to $u_\ell$, and assign the demand of $u'_r$ and $v_i$, for each $1 \le i \le n$, $i \notin \mathcal{A}$, to $u_r$. The total cost required is 2.

On the other hand, if the cost of the optimal demand assignment of $\mathcal{G}$ is 2, then we argue that there will exist a subset $\mathcal{A}$ satisfying the criterion. First, it is easy to see that the demand of $u'_\ell$ and the demand of $u'_r$ must have been assigned to $u_\ell$ and $u_r$, respectively. This leaves $\frac{1}{2} \sum_{1 \le i \le n} a_i$ residue capacity at both $u_\ell$ and $u_r$. Let $\mathcal{A} = \{i :$ the demand of $v_i$ is assigned to $u_\ell\}$. Therefore we have $\sum_{i \in \mathcal{A}} a_i = \sum_{1 \le i \le n, i \notin \mathcal{A}} a_i$ and $\mathcal{A}$ is a subset satisfying the criterion of the partition problem. This proves the theorem. □

### 6.3 Constant-Factor Approximation for Separable Demand

As the approximation algorithm presented in Sect. 5 is designed mainly for outer-planar graphs, to meet the minimum requirement of this framework, which is the ability to deal with planar graphs of at least three levels, i.e., $k$-outerplanar graphs with $k \ge 3$, we have to modify our algorithm to cope with this change.

Let $G_j$, $0 \le j \le \frac{m}{3}$, be the 3-outerplanar graphs obtained from the framework described the previous section. It suffices to show how each $G_j$ can be handled separately.

Consider a specific component, say, $G_{j_0}$. In the following, we describe how our approximation algorithm for outer-planar graphs can be modified slightly and ap-


**Fig. 12** (**a**) A 3-outerplanar graph. (**b**) Local connections w.r.t. a vertex $v$. Bold edges represent edges to be kept in the edge reduction process, while dashed edges represent edges to be removed

plied to $G_{j_0}$ to obtain a constant approximation. For convenience, we denote the set of vertices from the three levels of $G_{j_0}$ by $L_0$, $L_1$, and $L_2$, respectively. See also Fig. 12(a). Note that, from our construction scheme, only vertices in $L_1$ are demanding.

– *Obtaining the General Ladder*. We extract the general ladder merely from $L_1$ as we did before. When decomposing the ladder, for each vertex of the ladder, its incident edges to vertices in $L_0$ and $L_2$ are also included in addition to the ladder itself. Edges connecting vertices within $L_2$ and $L_0$ are discarded. As vertices in $L_2$ and $L_0$ are non-demanding, discarding these edges does not alter the optimal demand assignment.
– *Removing Edges*. In addition to the four neighboring vertices we identified for each vertex $v$ with non-zero demand, we identify two more vertices, which literally corresponds to the rightmost neighbors of $v$ in level $L_0$ and $L_2$, respectively. See also Fig. 12(b).

  As a result, the first part of Lemma 16 still holds, the degree upper-bound provided in the second part is increased by 2, and the third part holds automatically without alteration.

The remaining part of our algorithm, i.e., the primal-dual algorithm presented in Sect. 5.4 which computes a constant-factor approximation for the reduced ladder, remains unchanged. Since the algorithm is independent of the structure of general ladders, the assignment function computed remains feasible. As for the approximation factor, by Lemma 17 we know that the approximation factor is bounded by 9. From the above argument, we have the following theorem.

**Theorem 11** *One can compute a constant-factor approximation for the capacitated domination problem with separable demand on planar graphs in polynomial time.*

*Proof* Let $G_j$, $0 \le j \le \frac{m}{3}$ be the 3-outerplanar graphs obtained from the framework of Sect. 6.1, and denote by $\mathcal{G}_{j,i}$ and $\mathcal{H}_{j,i}$, $0 \le i < 3$, the general ladders and the corresponding reduced ladders the algorithm computes. Let $f_{j,i}$ be the demand assignment function computed by our algorithm and

$$f = \sum_{0 \le j \le \frac{m}{3}} \sum_{0 \le i < 3} f_{j,i}$$

be the demand assignment function to the entire graph. We have

$$
\begin{aligned}
w(f) &\leq \sum_{0\leq j\leq \frac{m}{3}} \sum_{0\leq i<3} w(f_{j,i}) && \text{Definition of } f,\\
&\leq 9\cdot \sum_{0\leq j\leq \frac{m}{3}} \sum_{0\leq i<3} OPT_f(\mathcal{H}_{j,i}) && \text{Lemma 17},\\
&\leq 18\cdot \sum_{0\leq j\leq \frac{m}{3}} \sum_{0\leq i<3} OPT_f(\mathcal{G}_{j,i}) && \text{Lemma 16},\\
&\leq 18\cdot \sum_{0\leq j\leq \frac{m}{3}} \sum_{0\leq i<3} OPT(\mathcal{G}_{j,i}) && \text{ILP relaxation},\\
&\leq 54\cdot \sum_{0\leq j\leq \frac{m}{3}} OPT(G_j) && \text{Lemma 13},\\
&\leq 90\cdot OPT(G) && \text{Framework of Sect. 6.1}.
\end{aligned}
$$

This proves the theorem. □

Notice that, the framework used in this section does not depend on the separability of the demand. Hence any approximation for the inseparable demand model on $k$-outerplanar graphs for $k \geq 3$ will imply an approximation for planar graphs as well. However, as our result for separable demand depends heavily on the linear program presented in Sect. 5.2, at this moment we are not sure how corresponding approximation results for inseparable demand model could be obtained.

## 7 Conclusion

In this paper, we studied a natural generalization of the dominating set under the concept of capacitation, called capacitated domination, and provided approximation algorithms as well as complexity results to compose a comprehensive study for this problem jointly with a series of previous work [31–33].

For general graphs, we provided logarithmic approximations which are asymptotically optimal. In terms of parameterized complexity, we showed that this problem is W[1]-hard with respect to the treewidth. For the positive side, we presented an FPT algorithm which takes the treewidth and the maximum capacity of the vertices as parameters. The latter implies a pseudo-polynomial time approximation scheme for planar graphs.

To drop the pseudo-polynomial factor of the approximation, we developed a constant-factor approximation for outerplanar graphs with separable demands followed by augmenting the algorithm to obtain a constant-factor approximation for planar graphs with separable demands. For planar graphs with inseparable demands, we showed that, unless $P = NP$, the approximation factor of any algorithm is at least $\frac{3}{2} - \epsilon$, for any $\epsilon > 0$.

Below, we conclude by listing related open problems. First, due to the flexibility of the ways the demand can be assigned, the approximation result we provided

for planar graphs in this paper seems to have room for further improvements. When the demand cannot be separated, however, it is NP-hard to approximate this problem on planar graphs within the factor of $\frac{3}{2} - \epsilon$, for any $\epsilon > 0$. These results are in some sense very different from the results stated in [33] for trees, which state that trees with inseparable demands are linear-time solvable while trees with separable demands are NP-complete to solve. Therefore, it would be very interesting to ask the approximability of this problem on planar graphs, i.e., to what extent can this problem be approximated for the two demand models?

Second, as we indicated implicitly in Sect. 5.1, the concept of general ladders does not extend directly to $k$-outerplanar graphs for $k \geq 2$. It would be interesting to formalize and extend this concept to $k$-outerplanar graphs, for it seems helpful not only to our problem, but also to other capacitated covering problems.

**Acknowledgements** The authors would like to thank the anonymous referees for their very helpful comments on the presentation of this work.

This work was supported in part by the National Science Council, Taipei 10622, Taiwan, under Grants NSC98-2221-E-001-007-MY3, NSC98-2221-E-001-008-MY3, NSC99-2911-I-002-055-2, NSC101-2221-E-005-026-MY2, and NSC101-2221-E-005-019-MY2.

## References

1. Alber, J., Bodlaender, H.L., Fernau, H., Kloks, T., Niedermeier, R.: Fixed parameter algorithms for dominating set and related problems on planar graphs. Algorithmica **33**, 461–493 (2002)
2. Alber, J., Fellows, M.R., Niedermeier, R.: Polynomial-time data reduction for dominating set. J. ACM **51**, 363–384 (2004)
3. Baker, B.S.: Approximation algorithms for NP-complete problems on planar graphs. J. ACM **41**, 153–180 (1994)
4. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC'93), pp. 226–234. ACM, New York (1993)
5. Bodlaender, H.L.: A partial $k$-arboretum of graphs with bounded treewidth. Theor. Comput. Sci. **209**, 1–45 (1998)
6. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial optimization on graphs of bounded treewidth. Comput. J. **51**, 255–269 (2008)
7. Bose, P.: On embedding an outer-planar graph in a point set. Comput. Geom. Theory Appl. **23**, 303–312 (2002)
8. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: lower bounds and upper bounds on kernel size. SIAM J. Comput. **37**, 1077–1106 (2007)
9. Chudak, F.A., Williamson, D.P.: Improved approximation algorithms for capacitated facility location problems. Math. Program. **102**, 207–222 (2005)
10. Chuzhoy, J.: Covering problems with hard capacities. SIAM J. Comput. **36**, 498–515 (2006)
11. Cygan, M., Pilipczuk, M., Wojtaszczyk, J.O.: Capacitated domination faster than $O(2^n)$. In: Proceedings of the 12th Scandinavian Conference on Algorithm Theory (SWAT'10), pp. 74–80. Springer, Berlin (2010)
12. Demaine, E.D., Fomin, F.V., Hajiaghayi, M., Thilikos, D.M.: Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. J. ACM **52**, 866–893 (2005)
13. Dom, M., Lokshtanov, D., Saurabh, S., Villanger, Y.: Capacitated domination and covering: a parameterized perspective. In: Proceedings of the 3rd International Conference on Parameterized and Exact Computation (IWPEC'08), pp. 78–90. Springer, Heidelberg (2008)
14. Downey, R.G., Fellows, M.: Parameterized Complexity. Springer, Berlin (1999)
15. Feige, U.: A threshold of $\ln n$ for approximating set cover. J. ACM **45**, 634–652 (1998)
16. Fellows, M.R., Fomin, F.V., Lokshtanov, D., Rosamond, F., Saurabh, S., Szeider, S., Thomassen, C.: On the complexity of some colorful problems parameterized by treewidth. Inf. Comput. **209**, 143–153 (2011)

17. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
18. Fomin, F.V., Thilikos, D.M.: Dominating sets in planar graphs: branch-width and exponential speed-up. SIAM J. Comput. **36**, 281–309 (2006)
19. Gandhi, R., Halperin, E., Khuller, S., Kortsarz, G., Srinivasan, A.: An improved approximation algorithm for vertex cover with hard capacities. J. Comput. Syst. Sci. **72**, 16–33 (2006)
20. Gandhi, R., Khuller, S., Parthasarathy, S., Srinivasan, A.: Dependent rounding in bipartite graphs. In: Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS'02), pp. 323–332. IEEE Comput. Soc., Washington (2002)
21. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
22. Guha, S., Hassin, R., Khuller, S., Or, E.: Capacitated vertex covering. J. Algorithms **48**, 257–270 (2003)
23. Guo, J., Niedermeier, R.: Linear problem kernels for NP-hard problems on planar graphs. In: Proceedings of the 34th International Conference on Automata, Languages and Programming (ICALP'07), pp. 375–386. Springer, Berlin (2007)
24. Hagerup, T.: Simpler linear-time kernelization for planar dominating set. In: Proceedings of the 6th International Conference on Parameterized and Exact Computation (IPEC'11), pp. 181–193. Springer, Berlin (2012)
25. Haynes, T.W., Hedetniemi, S., Slater, P.: Fundamentals of Domination in Graphs (Pure and Applied Mathematics). Dekker, New York (1998)
26. Haynes, T.W., Hedetniemi, S.M., Hedetniemi, S.T., Henning, M.A.: Domination in graphs applied to electric power networks. SIAM J. Discrete Math. **15**, 519–529 (2002)
27. Hochbaum, D.S.: Approximation algorithms for the set covering and vertex cover problems. SIAM J. Comput. **11**, 555–556 (1982)
28. Hopcroft, J., Tarjan, R.: Efficient planarity testing. J. ACM **21**, 549–568 (1974)
29. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. **9**, 256–278 (1974)
30. Kammer, F., Tholey, T.: Approximate tree decompositions of planar graphs in linear time. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12), pp. 683–698. SIAM, Philadelphia (2012)
31. Kao, M.-J., Chen, H.-L.: Approximation algorithms for the capacitated domination problem. In: FAW 2010, pp. 185–196. Springer, Berlin (2010)
32. Kao, M.-J., Lee, D.T.: Capacitated domination: constant factor approximations for planar graphs. In: Proceedings of the 22nd International Conference on Algorithms and Computation (ISAAC'11), pp. 494–503. Springer, Berlin (2011)
33. Kao, M.-J., Liao, C.-S., Lee, D.T.: Capacitated domination problem. Algorithmica **60**, 274–300 (2011)
34. Kloks, T.: Treewidth, Computations and Approximations. Lecture Notes in Computer Science, vol. 842. Springer, Berlin (1994)
35. Liao, C.-S., Lee, D.-T.: Power domination problem in graphs. In: Proceedings of the 11th Annual International Conference on Computing and Combinatorics (COCOON'05), pp. 818–828. Springer, Berlin (2005)
36. Liedloff, M., Todinca, I., Villanger, Y.: Solving capacitated dominating set by using covering by subsets and maximum matching. Discrete Appl. Math. (2012)
37. Niedermeier, R.: Invitation to Fixed Parameter Algorithms. Oxford University Press, Oxford (2006)
38. Pál, M., Tardos, E., Wexler, T.: Facility location with nonuniform hard capacities. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01), p. 329. IEEE Comput. Soc., Washington (2001)
39. Roberts, F.S.: Graph Theory and Its Applications to Problems of Society (1978)
40. Shmoys, D.B., Tardos, E., Aardal, K.: Approximation algorithms for facility location problems (extended abstract). In: STOC 1997, pp. 265–274. ACM, New York (1997)
41. van Bevern, R., Hartung, S., Kammer, F., Niedermeier, R., Weller, M.: Linear-time computation of a linear problem kernel for dominating set on planar graphs. In: Proceedings of the 6th International Conference on Parameterized and Exact Computation (IPEC'11), pp. 194–206. Springer, Berlin (2012)
42. Vazirani, V.V.: Approximation Algorithms. Springer, New York (2001)
43. Wan, P.-J., Alzoubi, K.M., Frieder, O.: A simple heuristic for minimum connected dominating set in graphs. Int. J. Found. Comput. Sci. **14**, 323–333 (2003)