


# Two-Level Capacitated Discrete Location with Concave Costs

Aditya Malik,<sup>a</sup> Ivan Contreras,<sup>a,\*</sup> Navneet Vidyarthi<sup>a</sup>

<sup>a</sup>Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, Montreal, Quebec H3G 1M8, Canada

\*Corresponding author

Contact: [aditya.malik@mail.concordia.ca](mailto:aditya.malik@mail.concordia.ca),  <https://orcid.org/0000-0002-5365-9517> (AM); [ivan.contreras@concordia.ca](mailto:ivan.contreras@concordia.ca),

 <https://orcid.org/0000-0002-0235-8108> (IC); [n.vidyarthi@concordia.ca](mailto:n.vidyarthi@concordia.ca) (NV)

Received: February 25, 2021

Revised: October 22, 2021; February 3, 2022;  
February 25, 2022

Accepted: March 8, 2022

Published Online in Articles in Advance:  
June 3, 2022

<https://doi.org/10.1287/trsc.2022.1150>

Copyright: © 2022 INFORMS

**Abstract.** In this paper, we study a general class of two-level capacitated discrete location problems with concave costs. The concavity arises from the economies of scale in production, inventory, or handling at the facilities and from the consolidation of flows for transportation and transshipment on the links connecting the facilities. Given the discrete nature of the problem, it is naturally formulated as a mixed-integer nonlinear program that uses binary variables for locational decisions and continuous variables for routing flows. We present an alternative formulation that only uses continuous variables and discontinuous functions, resulting in a nonlinear program with a concave objective function. Our main goal is to computationally compare these two modeling approaches under the same solution framework. In particular, we present an exact branch-and-bound algorithm that uses (integer) linear relaxations of the proposed formulations to optimally solve large-scale instances. The algorithm is enhanced with a cost-dependent spatial branching strategy and preprocessing step to improve its convergence. Extensive computational experiments are performed to assess the performance of the exact algorithm. Based on real location data from 3,109 counties in the contiguous United States, we also present a sensitivity analysis to showcase the impact of considering concave costs in location and assignment decisions.

**Funding:** This research was partly funded by the Canadian Natural Sciences and Engineering Research Council [Grants 2017-06732 and 2018-06704]. This support is gratefully acknowledged.

**Supplemental Material:** The online appendix is available at <https://doi.org/10.1287/trsc.2022.1150>.

**Keywords:** hierarchical facility location • concave costs • mixed-integer nonlinear programming • branch-and-bound

## 1. Introduction

Discrete location problems (DLPs) are central problems in location science with applications ranging from supply chain management (Melo, Nickel, and da Gama 2009), public policy (Salman and Yücel 2015), healthcare (Ahmadi-Javid, Seyedi, and Syam 2017), and telecommunications (Fortz 2015), among many others (Drezner and Hamacher 2002; Laporte, Nickel, and da Gama 2019). Generally speaking, DLPs seek to determine the location of facilities from a discrete set and to allocate customers to open facilities to satisfy customer demands while optimizing a given objective function. This paper focuses on an important class of DLPs arising in the design of hierarchical systems. Multilevel facility location problems (MFLPs) deal with the location of interacting facilities at different levels of a hierarchical system. Applications of MFLPs arise in a wide variety of contexts such as production-distribution systems, healthcare systems, telecommunications systems, urban and air transportation systems, and cargo and postal delivery systems. We refer to Şahin and Süral (2007), Farahani et al. (2014), Ortiz-Astorquiza, Contreras, and Laporte (2018), and

Contreras and Ortiz-Astorquiza (2019) for detailed surveys on MFLPs.

We study a general class of MFLPs denoted as *multilevel capacitated facility location problems with concave costs* (MCFLP-Cs), which can be defined as follows. Let  $I$  be the set of customers,  $V$  be the potential set of facilities partitioned into  $m$  levels, and the facilities at every level 1 to  $m$  are assumed to be capacitated. We consider a fixed setup cost for opening a facility and a variable operating cost at each facility modeled via a concave function of the amount of flow passing through the facility. Transportation costs are modeled with concave functions of the amount of flow routed through each link. We assume a *single-flow* pattern in which flows start from facilities at the highest level  $m$  and pass through all levels until they arrive at their demand points at the customer level. Moreover, we consider the possibility of transshipment flows at intermediate facilities. That is, a facility at level  $\ell \in \{1, 2, \dots, m-1\}$  can send flows to the facilities located at the same level  $\ell$ . All customers receive shipments only from the facilities opened at the first level. We assume a multiple allocation strategy in which

customers can receive demand from more than one facility at level 1. We also consider multiple allocation setting at the facilities at intermediate levels. The MCFLP-C consists of selecting a set of facilities to open at each level, such that the incoming flow on each facility satisfies its operating capacity limitations while ensuring that all the demand is met. The objective is to minimize the total fixed and variable costs to open and operate a set of facilities and the variable transportation cost for all used links. For ease of exposition and to alleviate the mathematical notation, we will focus on the particular case of  $m = 2$ , that is, the *two-level capacitated facility location problem with concave costs* (2CFLP-Cs), to illustrate the proposed models and solutions algorithms. However, the models and algorithms presented in this work can be readily extended to the more general case of  $m > 2$ .

Potential applications of 2CFLP-Cs (or MCFLP-Cs) arise in the design and reconfiguration of production-distribution systems in which a two-level hierarchical structure is used. The economies of scale observed in production, warehousing, and distribution operations can be modeled as concave functions of the quantities produced, stored, and distributed, respectively (Soland 1974). These concave functions also provide flexibility to model carbon emissions from transportation (Elhedhli and Merrick 2012). Hajiahyai, Mahdian, and Mirrokni (2003) deal with general setup cost functions arising in applications where multiple types of facilities are to be colocated (e.g., the problem of placing Internet servers). Their objective function comprises site setup costs (fixed) and facility setup costs, which are concave functions of the number of customers assigned or the number of facilities colocated at each site. In the context of production-inventory-distribution systems, Shen, Coullard, and Daskin (2003), Shen and Daskin (2005), and Vidyarthi et al. (2007) use concave functions to model safety stock inventory levels and inventory costs. Saif and Elhedhli (2016) consider a facility location problem in which technology acquisition costs are concave.

In this paper, we introduce the 2CFLP-C, a general class of MFLPs in which variable operating costs at facilities and transportation costs on arcs are univariate concave functions of the flows. We also present two formulations for the 2CFLP-C. The first one is a mixed-integer nonlinear programming (MINLP) formulation that models location decisions and fixed costs using binary variables and flows with continuous variables. This is the modeling approach that has been extensively used over the last 60 years when modeling DLPs (Laporte, Nickel, and da Gama 2019). The second formulation is a nonlinear program (NLP) that contains only continuous flow variables on facilities and arcs. At every facility, a discontinuous but concave fixed-charge function is used to model the

setup and variable costs. Although this class of NLP formulations has already been introduced to model and solve DLPs (Soland 1974), to the best of our knowledge, these two modeling approaches have not been directly compared under the same model and solution framework. Therefore, an important contribution of this paper is to develop an exact algorithm based on state-of-the-art methods commonly used for solving MINLPs and NLPs to solve large-scale instances of 2CFLP-Cs. The proposed exact algorithm is a branch-and-bound (BB) algorithm that partitions the feasible region on the domain of the continuous variables. We present two lower bounding procedures: (i) an integer linear relaxation of the MINLP formulation and (ii) a linear relaxation of the NLP formulation. The standard BB algorithm is enhanced with a cost-dependent spatial branching strategy and a preprocessing step to improve its convergence. Results obtained on large-scale instances with up to 2,250 customers and 150 potential facilities confirm the effectiveness of our exact algorithm, in particular when using linear relaxations of NLPs as a bounding procedure. We also present a sensitivity analysis on an instance considering the 3,109 counties in the contiguous United States to understand the impact of economies of scale in facility operations and transportation costs on location and assignment decisions.

The remainder of the paper is organized as follows. Section 2 presents a literature review on related discrete location and concave network flow problems. Section 3 provides a formal definition of the 2CFLP-C and two mathematical programming formulations. Section 4 describes the exact solution algorithm and some algorithmic enhancements. The results of extensive computational experiments are reported in Section 5. Conclusions follow in Section 6.

## 2. Literature Review

In this section, we provide a succinct literature review in two closely related streams of research to our work: MFLPs and *minimum concave-cost network flow problems* (MCCNFPs). The latter class of problems are relevant given that most DLPs can be transformed into variants of network flow problems by splitting nodes into arcs in an auxiliary network. This means that it may be possible to exploit existing state-of-the-art exact algorithms to solve DLPs when the resulting network flow transformation corresponds to a known problem.

We start with MFLP. Ortiz-Astorquiza, Contreras, and Laporte (2017) present different formulations for MFLPs considering location decisions and cardinality constraints at each level. To the best of our knowledge, Ortiz-Astorquiza, Contreras, and Laporte (2019) study the most general class of uncapacitated MFLPs,

in which not only cardinality constraints on the number of open facilities at each level are considered but also setup costs for activating the links of the network. In all these MFLPs, the objective functions are assumed to be linear functions of location, allocation, and operational decisions.

Single-level DLPs with nonlinear objective functions have also received attention. The case of concave costs has been studied by Soland (1974), Kelly and Khumawala (1982), and Kubo and Kasugai (1991), whereas other nonlinear objectives have also been considered to represent different scenarios such as congestion costs (Vidyarthi and Jayaswal 2014; Fischetti, Ljubić, and Sinnl 2016; Berman and Krass 2019) and S-shaped costs (Harkness and ReVelle 2003; Lu, Gzara, and Elhedhli 2014). Concave costs functions have also been considered in supply chain network design and production-distribution systems design (Cohen and Moon 1991; Lin, Nozick, and Turnquist 2006; Baumgartner, Fuetterer, and Thonemann 2012; Elhedhli and Merrick 2012). We refer to Malik (2021) for a comprehensive review of models and solutions algorithms for DLPs and supply chain design problems with nonlinear costs. Solution methods for DLPs with congestion costs are mainly based on cutting plane algorithms. However, these methods do not guarantee global optimality for concave and S-shaped cost functions. Therefore, these problems are only approximately solved by using either piecewise linear approximations of concave functions or Lagrangian relaxations.

We now discuss MCCNFPs. Minimum-cost network flow problems can be divided into single-commodity and multicommodity versions and into uncapacitated and capacitated versions. It is possible to derive single-commodity capacitated network flow reformulations of both MINLP and NLP formulations presented in Section 3. The capacities arise when splitting a (capacitated) facility node into an arc. In particular, we need to identify exact algorithms for solving capacitated minimum-cost network flow problems (MCNFPs) with either (i) continuous concave functions or (ii) fixed-charge (i.e., discontinuous) concave functions.

In the former case, if the concave functions are replaced by linear functions, the problem corresponds to the *single-commodity capacitated fixed-charge network flow problem*. This problem contains binary decisions for activating a subset of links of the network. Ortega and Wolsey (2003) present an exact branch-and-cut algorithm for solving uncapacitated variants, whereas Atamtürk, Küçükyavuz, and Tezel (2017) develop a polyhedral study for the capacitated variant and a branch-and-cut algorithm. In both models, the flow costs are linear. To the best of our knowledge, there is no exact algorithm that has been developed for

solving the case with (continuous) concave flow costs. In the latter case (i.e., fixed-charge concave functions), the problem has already been studied in the literature and is referred to as the *minimum concave cost capacitated network flow problem*. To the best of our knowledge, the state-of-the-art (and most recent) algorithm for this problem is the BB algorithm presented in Horst and Thoai (1998). Although this algorithm uses one of the bounding procedures we present in Section 4, both BB algorithms are significantly different in terms of the branching and preprocessing steps. In Section 5, we compare the computational performance of our BB algorithm with that of Horst and Thoai (1998) for solving 2CFLP-Cs.

The most effective solution methods available in the literature for MCCNFPs can be classified into three categories: (1) heuristic procedures (Guisewite and Pardalos 1990a; Larsson, Migdalas, and Rönnqvist 1994; Kim, Pan, and Pardalos 2006; Dang et al. 2011); (2) exact dynamic programming (DP) algorithms (Zangwill 1968); and (3) exact BB algorithms (Horst and Thoai 1998; Fontes, Hadjiconstantinou, and Christofides 2006; Manousiouthakis, Thomas, and Justanieh 2011). DP-based algorithms are mainly effective for networks with a small number of source/sink nodes given the exponential growth of the state space with the problem size (Lamar 1993). There exist other more recent papers describing effective exact algorithms for particular cases of minimum-cost flow problems such as minimum convex-cost flow problems on bipartite networks (Castro and Nasini 2021) and MCCNFPs on grid networks (He, Ahmed, and Nemhauser 2015; Ahmed et al. 2016). However, these algorithms are not applicable for 2CFLP-Cs as they do not apply when considering more general network topologies used in 2CFLP-Cs. Finally, we refer to Guisewite and Pardalos (1990b) and Fontes (2008) for reviews on algorithms for other classes of MCCNFPs.

Solution approaches for general concave minimization problems are usually based on implicit enumeration, outer approximation, polyhedral annexation (Benson 1996), and piecewise linear approximations (Baumgartner, Fuetterer, and Thonemann 2012). See Benson (1996) for a detailed discussion on them. Our BB algorithm is different from the previous methods in several ways: (i) it does not implicitly enumerate the extreme points, (ii) the bounding step in our case is (integer) linear (as opposed to outer-approximation), and (iii) it does not require exploring a large number of points at every iteration (unlike polyhedral annexation). We can also compute an optimality gap at termination, which is not possible when using piecewise linear approximations. Belotti et al. (2013) state that BB is arguably the best-known method for solving nonconvex problems. We refer to Lee and Leyffer (2012), Burer and Letchford (2012),



D'Ambrosio and Lodi (2013), and Belotti et al. (2013) for surveys on nonconvex problems, including concave minimization.

### 3. Problem Definition and Formulations

The 2CFLP-C is defined as follows. Let  $G = (V, E)$  be a directed graph with vertex set  $V = V_1 \cup V_2 \cup I$ , where  $V_1$  and  $V_2$  are the set of potential facilities at level 1 (intermediate facilities) and level 2 (origin facilities), respectively, and  $I$  is the set of customers. We define the directed arc set  $E = E_1 \cup E_2 \cup E_t$ , where  $E_1 = \{(j, i) : j \in V_1, i \in I\}$  is the set of arcs connecting the intermediate facilities (level 1) to customers,  $E_2 = \{(k, j) : k \in V_2, j \in V_1\}$  is the set of arcs representing the flows from the origin facilities (level 2) to the intermediate facilities (level 1), and  $E_t = \{(j, j') : j \in V_1, j' \in V_1, j \neq j'\}$  is the set of transshipment arcs interconnecting the intermediate facilities. Let  $D_i$  be the customer's demand at node  $i \in I$ . For  $r = 1, 2$ , let  $f_{rj}$  denote the nonnegative fixed cost associated with locating/opening facility at node  $j \in V_r$  and  $b_{rj}$  be its capacity.

To model the 2CFLP-C, for  $j \in V_1$ , we define continuous operational variables  $v_j \in R^+$  equal to the amount of flow shipped from node  $j$ , and  $W_j(v_j)$  its corresponding operational cost function. For  $k \in V_2$ , we define continuous variables  $u_k \in R^+$  equal to the amount of flow originated at node  $k$ , and  $P_k(u_k)$  its corresponding operational cost function. All functions  $W_j(v_j)$  and  $P_k(u_k)$  modeling operating costs at the first-level and second-level facilities, respectively, are assumed to be univariate concave functions. Moreover, for each  $e \in E$  we define and classify the set of continuous flow variables on the arcs as follows: (i)  $x_{1ji} : (j, i) \in E_1$ , (ii)  $x_{2kj} : (k, j) \in E_2$ , and (iii)  $y_{jj'} : (j, j') \in E_t$ . Similarly, we define  $C_{1ji}(x_{1ji})$ ,  $C_{2kj}(x_{2kj})$  and  $C_{jj'}(y_{jj'})$  as univariate concave cost functions on arc sets  $E_1$ ,  $E_2$ , and  $E_t$ , respectively. The 2CFLP-C consists of selecting sets of facilities to open at each level, determining the flows through the facilities at the two levels, and assigning each customer to a set of open facilities at the first level while minimizing the sum of fixed location costs and variable operating and transportation costs. Because of the fixed costs for opening facilities, the 2CFLP-C has discontinuous cost functions, and we next present two ways to model them: an MINLP with continuous concave functions and an NLP with discontinuous concave functions.

#### 3.1. Mixed-Integer Nonlinear Programming Formulation

To model the problem as an MINLP, we define binary location variables  $z_{1j}$  equal to one if and only if a facility is located at node  $j \in V_1$ . Similarly, for each  $k \in V_2$  we define binary location variables  $z_{2k}$  equal to one if and only if a facility is located at node  $k$ . With the

previous notations, the MINLP formulation for the 2CFLP-C is as follows:

$$\begin{aligned}
 (NF_1) \quad & \text{minimize} \sum_{j \in V_1} \left( f_{1j} z_{1j} + W_j(v_j) + \sum_{j' \in V_1: j' \neq j} C_{jj'}(y_{jj'}) \right) \\
 & + \sum_{i \in I} C_{1ji}(x_{1ji}) + \sum_{k \in V_2} \left( f_{2k} z_{2k} + P_k(u_k) + \sum_{j \in V_1} C_{2kj}(x_{2kj}) \right) \quad (1) \\
 \text{subject to} \quad & \sum_{j \in V_1} x_{2kj} = u_k \leq b_{2k} z_{2k} \quad k \in V_2, \quad (2) \\
 & \sum_{k \in V_2} x_{2kj} + \sum_{j' \in V_1: j' \neq j} y_{j'j} = v_j \leq b_{1j} z_{1j} \quad j \in V_1, \quad (3) \\
 & \sum_{k \in V_2} x_{2kj} + \sum_{j' \in V_1: j' \neq j} y_{j'j} = \sum_{j' \in V_1: j' \neq j} y_{jj'} + \sum_{i \in I} x_{1ji} \\
 & \quad \quad \quad j \in V_1, \quad (4) \\
 & \sum_{j \in V_1} x_{1ji} = D_i \quad i \in I, \quad (5) \\
 & z_{1j}, z_{2k} \in \{0, 1\} \quad j \in V_1, k \in V_2, \quad (6) \\
 & v_j, u_k, x_{1ji}, x_{2kj}, y_{jj'} \geq 0 \quad j \in V_1, k \in V_2, i \in I. \quad (7)
 \end{aligned}$$

The objective function consists of two sets of costs. The first set contains the fixed costs of opening and variable costs of operating first-level facilities, transshipment costs between first-level facilities, and flow costs from first-level facilities to customers. The second set consists of the fixed costs of opening and variable costs of operating second-level facilities and the flow costs from second- to first-level facilities. Constraint Sets (2) and (3) are the linking and capacity restrictions for facilities at the second and first levels, respectively. Constraints (4) are flow balance constraints at the intermediate facilities. Constraints (5) ensure that the customers' demands are met. Constraints (6) and (7) are standard integrality and nonnegativity restrictions.

#### 3.2. Nonlinear Programming Formulation

Observe that in the MINLP, fixed costs at both levels are modeled using the binary variables  $z_{1j}$  and  $z_{2k}$  in the objective function. We next present an NLP formulation for the 2CFLP-C that does not require the use of binary variables. It is based on the following discontinuous concave functions:

$$\begin{aligned}
 \Theta_j(v_j) &= \begin{cases} 0, & \text{if } v_j = 0, \\ f_{1j} + W_j(v_j), & \text{if } v_j > 0. \end{cases} \\
 \Omega_k(u_k) &= \begin{cases} 0, & \text{if } u_k = 0, \\ f_{2k} + P_k(u_k), & \text{if } u_k > 0. \end{cases} \quad (8)
 \end{aligned}$$

Given that fixed costs are embedded into the *modified* operational functions  $\Theta_j(v_j)$  and  $\Omega_k(u_k)$ , there is no need to model the fixed costs using binary variables as it is most commonly done when modeling and solving DLPs. Combining discontinuous functions  $\Theta_j(v_j)$  and  $\Omega_k(u_k)$  with the continuous variables used

in MINLP, the 2CFLP-C can be reformulated as

$$(NF_2) \text{ minimize } \sum_{j \in V_1} \left( \Theta_j(v_j) + \sum_{j' \in V_1: j' \neq j} C_{jj'}(y_{jj'}) \right) + \sum_{i \in I} C_{1ji}(x_{1ji}) + \sum_{k \in V_2} \left( \Omega_k(u_k) + \sum_{j \in V_1} C_{2kj}(x_{2kj}) \right) \quad (9)$$

$$\text{subject to } \sum_{j \in V_1} x_{2kj} = u_k \leq b_{2k} \quad k \in V_2, \quad (10)$$

$$\sum_{k \in V_2} x_{2kj} + \sum_{j' \in V_1: j' \neq j} y_{jj'} = v_j \leq b_{1j} \quad j \in V_1, \quad (11)$$

$$\sum_{k \in V_2} x_{2kj} + \sum_{j' \in V_1: j' \neq j} y_{jj'} = \sum_{j' \in V_1: j' \neq j} y_{jj'} + \sum_{i \in I} x_{1ji} \quad j \in V_1, \quad (12)$$

$$\sum_{j \in V_1} x_{1ji} = D_i \quad i \in I, \quad (13)$$

$$v_j, u_k, x_{1ji}, x_{2kj}, y_{jj'} \geq 0 \quad j \in V_1, k \in V_2, i \in I. \quad (14)$$

The objective function consists of the discontinuous concave functions  $\Theta_j(v_j)$  and  $\Omega_j(v_j)$  that capture the fixed opening and variable operating costs of the first- and second-level facilities, respectively. Constraint Sets (10) and (11) are the capacity constraints at facilities. Constraint Set (12) corresponds to the flow balance conditions for first-level facilities. Constraint Set (13) ensures that the customers' demands are satisfied. Constraints (14) are the standard nonnegativity restrictions.

#### 4. B&B Algorithms for Solving the MINLP and Discontinuous NLP Formulations of the 2CFLP-C

We next present an exact algorithm for the 2CFLP-C. It is based on the BB algorithm introduced in Falk and Soland (1969) and Soland (1974) for univariate concave minimization problems. Before presenting our own version of this algorithm, we first explain its exact functionality.

This BB algorithm is an iterative procedure that solves a convex relaxation of the original problem at every node of the enumeration tree. The tightest convex relaxation of a univariate concave function is the line segment joining the endpoints of the concave function on the domain of the variable. This is the relaxation that we follow. The optimal solution value of the relaxed problem provides a valid lower bound on the original problem. At each node, an upper bound is obtained either heuristically or simply by evaluating the original nonlinear objective function using the solution of the relaxed problem at the current node. If the termination criteria are not met, a node is selected from a list of unexplored (unsolved) nodes and two child nodes are created by branching (i.e., partitioning) on the domain of a continuous decision variable. Based on this idea, Ryoo and Sahinidis

(1996) introduce a branch-and-reduce algorithm that enforces strong domain reduction rules in addition to standard domain reduction techniques commonly used in BB algorithms. This algorithm is used in BARON, a state-of-the-art global optimization solver for nonconvex optimization problems. In our computational experiments, we compare the performance of our proposed BB algorithm with that of BARON.

We next provide a baseline of the BB algorithm. We refer to Falk and Soland (1969), Soland (1974), and Lee and Leyffer (2012) for additional details. We then describe how we adapt and extend this baseline algorithm for solving large-scale instances of the 2CFLP-C. In particular, we present (i) two lower bounding procedures (the first is based on an integer linear relaxation of  $NF_1$ , whereas the second is based on a network transformation of a linear relaxation of  $NF_2$ ) (ii) a preprocessing phase that may be used to fix as many location decisions as possible before branching on the continuous flow variables, and (iii) two branching strategies. The use of preprocessing and choice of branching strategy depend on simple but effective rules that exploit information generated by solving the root node relaxation of  $NF_1$ . These rules help us in determining the most promising combination of algorithmic features to use for each instance.

##### 4.1. BB Algorithm for Univariate Concave Minimization Problems

Consider the following concave minimization problem:

$$(NP) \text{ minimize } \sum_{i=1}^n f_i(x_i) \\ \text{subject to } x \in D \\ L_i \leq x_i \leq U_i \quad i = 1, \dots, n.$$

We assume  $D$  is a polyhedral set, and each  $f_i(x_i)$  is univariate and concave over the set  $G_i = \{x_i \mid L_i \leq x_i \leq U_i\}$ , where  $L_i$  and  $U_i$  are finite. We define  $G = \cup_{i=1}^n G_i$ .  $NP$  is a concave minimization problem whose optimal solution lies at some extreme point of the polytope  $D \cap G$ . Furthermore, let  $N^0, N^1, \dots$  be the nodes of the enumeration tree. We denote  $N^0$  as the root node at which the initial bounds on  $x$  are defined by the rectangle  $G$ , and we initialize  $N^0 = NP$ . The problem at any node  $N^a$  is defined over the feasible region  $D \cap G^a$ , where  $G^a \subseteq G$ , and each  $G_i^a$  is defined as

$$G_i^a = \{x_i \mid L_i \leq L_i^a \leq x_i \leq U_i^a \leq U_i\} \quad i = 1, \dots, n.$$

We first describe the relaxation step of the algorithm. The concave problem at node  $N^a$  is relaxed using  $\phi_i^a$  as the underestimator of  $f_i(x_i)$ .  $\phi_i^a(x_i)$  is the linear function connecting the points  $(L_i^a, f_i(L_i^a))$  and  $(U_i^a, f_i(U_i^a))$ . Given that  $f_i(x_i)$  is concave, for  $x_i \in [L_i^a, U_i^a]$ , we have  $\phi_i^a(x_i) \leq f_i(x_i)$ , and for  $x \in G^a$ , we have  $\sum_{i=1}^n \phi_i^a(x_i) \leq$

$\sum_{i=1}^n f_i(x_i)$ . Therefore, a lower bound  $\underline{v}(N^a)$  can be obtained at node  $N^a$  by solving the following problem:

$$(LP^a) \underline{v}(N^a) = \text{minimize } \sum_{i=1}^n \phi_i^a(x_i) \\ \text{subject to } x \in D \\ L_i^a \leq x_i \leq U_i^a \quad i = 1, \dots, n.$$

Let  $\hat{x}^a$  be an optimal solution of  $LP^a$ . An upper bound  $\bar{v}(N^a)$  can be trivially obtained at each node  $N^a$  by evaluating the original concave function of  $NP$  at  $\hat{x}^a$ , that is,  $\bar{v}(N^a) = \sum_{i=1}^n f_i(\hat{x}_i^a)$ . From now on, we refer to  $\phi_i(x_i)$  as the *linear underestimation function* (LUF) of  $f_i(x_i)$ , and  $LP^a$  as the *linear underestimation problem* (LUP) at node  $N^a$ .

We next discuss the spatial branching step. At iteration  $r$ , this step partitions a selected node  $N^s$  into two child nodes  $N^{2r+1}$  (left node) and  $N^{2r+2}$  (right node), with rectangles  $G^{2r+1}, G^{2r+2} \subset G^s$ , such that  $G^s = G^{2r+1} \cup G^{2r+2}$ . Let  $\pi = \{N^p, N^q, \dots\}$  be the set of unsolved nodes at iteration  $r$ , that is, the nodes from which no branching has been performed yet. We use a *best bound strategy* (Belotti et al. 2013) to select the node for branching, that is,  $N^s = \pi_n$ , where

$$n \in \arg \min \{\underline{v}(\pi_t) : t = 1, \dots, |\pi|\}.$$

Given that  $N^s$  is the node with the lowest lower bound value in the set  $\pi$ , the best lower bound value  $\underline{v}^*$  of  $NP$  is equal to  $\underline{v}(N^s)$ . In  $N^s$ , the index of the branching variable is selected as

$$j \in \arg \max \{f_i(\hat{x}_i^s) - \phi(\hat{x}_i^s) : i = 1, \dots, n\},$$

that is, a concave function with the worst underestimation in solution  $\hat{x}_i^s$  of  $LP^s$  at node  $N^s$ . The solution value of the variable  $x_j$  lies in the interval  $[L_j^s, U_j^s]$ . We divide this interval into two subintervals,  $[L_j^s, S_j]$  and  $[S_j, U_j^s]$ , and create the associated child nodes  $N^{2r+1}$  and  $N^{2r+2}$ , respectively. The performance of the algorithm is further dependent on how point  $S_j$  is selected. The concave subproblems at nodes  $N^{2r+1}$  and  $N^{2r+2}$  differ from each other and the subproblem at node  $N^s$  only in the interval bounds of variable  $x_j$ , whereas the bounds for all other variables  $x_i$  for each  $i = \{1, \dots, n\} \setminus \{j\}$  are the same in these subproblems. A pseudo-code of this BB algorithm is depicted in Algorithm 1. We define  $r$  as the iteration counter,  $\epsilon$  as the target optimality gap,  $\pi$  as the set of unsolved nodes,  $\underline{v}^*$  and  $\bar{v}^*$  as the values of best lower and upper bounds of  $NP$ , respectively,  $x^*$  as the incumbent solution, and  $\hat{x}^k$  the solution of LUP  $LP^k$ .

**Algorithm 1** (Generic Branch-and-Bound Algorithm for  $NP$ )

- 1: **Initialize**  $r \leftarrow 0$ ;  $\pi \leftarrow \emptyset$ ;  $N^s \leftarrow N^0$
- 2: Solve  $LP^0$  at root node  $N^0$  and set  $\underline{v}^* \leftarrow \underline{v}(N^0)$ ;  $\bar{v}^* \leftarrow \bar{v}(N^0)$ ;  $x^* \leftarrow \hat{x}^0$ ;  $\rho \leftarrow \frac{(\bar{v}^* - \underline{v}^*)}{\bar{v}^*}$

- 3: **while**  $\rho > \epsilon$  **do**
- 4: Branch on parent node  $N^s$  to create child nodes  $N^{2r+1}$  and  $N^{2r+2}$
- 5: Solve  $LP^{2r+1}$  at left node  $N^{2r+1}$ :  
     **if**  $\bar{v}(N^{2r+1}) < \bar{v}^*$  **then**  $\bar{v}^* \leftarrow \bar{v}(N^{2r+1})$  and  $x^* \leftarrow \hat{x}^{2r+1}$   
     **if**  $\underline{v}(N^{2r+1}) < \underline{v}^*$  **then**  $\pi \leftarrow \pi \cup \{N^{2r+1}\}$
- 6: Solve  $LP^{2r+2}$  at right node  $N^{2r+2}$ :  
     **if**  $\bar{v}(N^{2r+2}) < \bar{v}^*$  **then**  $\bar{v}^* \leftarrow \bar{v}(N^{2r+2})$  and  $x^* \leftarrow \hat{x}^{2r+2}$   
     **if**  $\underline{v}(N^{2r+2}) < \underline{v}^*$  **then**  $\pi \leftarrow \pi \cup \{N^{2r+2}\}$
- 7: Select node  $N^s \leftarrow N^n$ , where  $n \in \arg \min \{\underline{v}(\pi_t) : t = 1, \dots, |\pi|\}$
- 8: Update  $\pi \leftarrow \pi \setminus \{N^s\}$ ;  $\underline{v}^* \leftarrow \underline{v}(N^s)$ ;  $\rho \leftarrow \frac{(\bar{v}^* - \underline{v}^*)}{\bar{v}^*}$ ;  $r \leftarrow r + 1$
- 9: **end while**

The algorithm begins by solving  $LP^0$  at root node  $N^0$  to provide  $\underline{v}^*$  and  $\bar{v}^*$ . It then branches from  $N^0$  to create two child nodes  $N^1$  and  $N^2$ . It then solves both  $LP^1$  and  $LP^2$  at the two newly created nodes and updates the value of  $\bar{v}^*$  accordingly. The two child nodes are appended to the set  $\pi$  from which we select a node  $N^s$  for further branching in the enumeration tree. The node  $N^s$  is then removed from the list of unsolved nodes, and the values of  $\underline{v}^*$ ,  $\rho$  and  $r$  are updated. The algorithm performs the iterations until  $\rho \leq \epsilon$ .

## 4.2. Using $NF_1$ as Lower Bounding Procedure

We now describe how to generate lower bounds using  $NF_1$ . At any node  $N^a$ , we define  $p_k^a(u_k)$  and  $w_j^a(v_j)$  as LUFs for the univariate concave functions  $P_k(u_k)$  and  $W_j(v_j)$ , respectively. Similarly,  $c_{1ji}^a(x_{1ji})$ ,  $c_{2kj}^a(x_{2kj})$ , and  $c_{jj'}^a(y_{jj'})$  are LUFs for  $C_{1ji}^a(x_{1ji})$ ,  $C_{2kj}^a(x_{2kj})$ , and  $C_{jj'}^a(y_{jj'})$ , respectively. Therefore, the integer LUP associated with  $NF_1$  at node  $N^a$  is defined as

$$(LP_1^a) \text{ minimize } \sum_{j \in V_1} \left( f_{1j} z_{1j} + w_j^a(v_j) + \sum_{j' \in V_1 : j' \neq j} c_{jj'}^a(y_{jj'}) \right) \\ + \sum_{i \in I} c_{1ji}^a(x_{1ji}) + \sum_{k \in V_2} \left( f_{2k} z_{2k} + p_k^a(u_k) \right) \\ + \sum_{j \in V_1} c_{2kj}^a(x_{2kj}) \quad (15)$$

$$\text{subject to } (2) - (7) \quad (16)$$

$$L_{2k}^a \leq u_k \leq U_{2k}^a \quad k \in V_2, \quad (17)$$

$$L_{1j}^a \leq v_j \leq U_{1j}^a \quad j \in V_1, \quad (18)$$

$$L_{2kj}^a \leq x_{2kj} \leq U_{2kj}^a \quad k \in V_2, j \in V_1, \quad (19)$$

$$L_{jj'}^a \leq y_{jj'} \leq U_{jj'}^a \quad j \in V_1, j' \in V_1 : j' \neq j, \quad (20)$$

$$L_{1ji}^a \leq x_{1ji} \leq U_{1ji}^a \quad j \in V_1, i \in I. \quad (20)$$

Objective (15) is linear, and Constraints (16)–(20) impose reduced lower and upper bounds on continuous variables at node  $N^a$ . At root node  $N^0$ , the lower

**Table 1.** Upper Bounds at Root Node

Parameter	Value
$U_{2k}^0$	$\min\{b_{2k}, \sum_{i \in I} D_i\}$
$U_{1j}^0$	$\min\{b_{2k}, \sum_{i \in I} D_i\}$
$U_{2kj}^0$	$\min\{b_{2k}, b_{1j}, \sum_{i \in I} D_i\}$
$U_{jj'}^0$	$\min\{b_{1j}, b_{1j'}, \sum_{i \in I} D_i\}$
$U_{1ji}^0$	$\min\{b_{1j}, D_i\}$

bounds of all variables in  $LF_1^0$  are initialized to zero, and the upper bounds are given in Table 1.

Note that  $LF_1^a$  is an MILP problem in which integrality conditions on the  $z_{1j}$  and  $z_{2k}$  variables are kept. Therefore, similar to BB algorithms based on Lagrangian relaxations (Contreras, Cordeau, and Laporte 2011), we use an MILP as a lower bounding procedure at each node of the enumeration tree.

#### 4.3. Using $NF_2$ as Lower Bounding Procedure

We now describe how to generate lower bounds using  $NF_2$ . At any node  $N^a$ , we define  $\theta_j^a(v_j)$  and  $\omega_k^a(u_k)$  as LUFs for concave operating cost functions  $\Theta_j(v_j)$  and  $\Omega_k(u_k)$ , respectively. The LUFs for transportation costs are the same as in  $LF_1^a$ . The LUP associated with  $NF_2$  at node  $N^a$  is defined as

$$(LF_2^a) \text{ minimize } \sum_{j \in V_1} \left( \theta_j^a(v_j) + \sum_{j' \in V_1: j' \neq j} c_{jj'}^a(y_{jj'}) + \sum_{i \in I} c_{1ji}^a(x_{1ji}) \right) + \sum_{k \in V_2} \left( \omega_k^a(u_k) + \sum_{j \in V_1} c_{2kj}^a(x_{2kj}) \right) \quad (21)$$

subject to (10)–(14), (16)–(20).

Objective (21) is linear, and Constraints (16)–(20) are the same used in  $LF_1^a$ . Note that  $LF_2^a$  is a two-level transportation problem with continuous flow variables and no binary variables for the location decisions. Therefore, we can state  $LF_2^a$  as a *minimum cost network flow problem*. This allows us to use highly efficient commercial network solvers that are significantly

faster than commercial LP solvers. In what follows, we provide the details of the transformation.

Let node sets  $V_2$ ,  $V_1$ , and  $I$  of the graph  $G$  be the supply nodes, intermediate nodes, and sink nodes, respectively. To transform  $LF_2^a$  into an MCNFP, we extend  $G$  with sets of dummy supply nodes  $V_2'$ , dummy intermediate nodes  $V_1'$ , and a dummy sink node  $i'$ . Let  $d(k)$  and  $d'(k')$  be the mapping for the supply nodes such that  $k' = d(k)$ , and  $k = d'(k')$ . Similarly, let  $e(j)$  and  $e'(j')$  be the mapping for the intermediate nodes such that  $j' = e(j)$ , and  $j = e'(j')$ . Supply and demand are balanced by setting the demand at dummy sink node  $i'$  as  $\sum_{k \in V_2} b_{2k} - \sum_{i \in I} D_i$ . The extended graph  $G^N$  for the network flow representation is depicted in Figure 1.

The dummy node  $i'$  receives all its inflow directly from supply nodes set  $V_2$  with zero link operating cost. A dummy supply node  $k' \in V_2'$  is linked only to its original supply node  $k = d'(k')$ . Flow and capacity on an arc  $(k, k')$  represent operational quantity decision  $u_k$  and upper limit restriction on  $u_k$  at facility  $k \in V_2$ , respectively. An intermediate node  $j \in V_1$  receives inflow from dummy supply nodes set  $V_2'$  and dummy intermediate nodes set  $V_1' \setminus \{j'\}$ , where  $j' = e(j)$ . Also, dummy intermediate node  $j' \in V_1'$  has only one inflow arc from original intermediate node  $j = e'(j')$ , and the operational quantity decision  $v_j$  and upper limit restriction on  $v_j$  at facility  $j \in V_1$  are represented through flow on arc  $(j, j')$ . Also, a sink node  $i \in I$  receives all its inflow from dummy intermediate nodes set  $V_1'$ . We define flow variables on arcs as (i)  $u'_{kk'}$ , (ii)  $v'_{jj'}$ , (iii)  $x'_{2k'j}$ , (iv)  $y'_{j'j'}$ , (v)  $x'_{1j'i}$ , and (vi)  $x'_{ki'}$ . The  $LF_2^a$  can be stated as the following MCNFP:

$$(TLF_2^a) \text{ minimize } \sum_{j \in V_1, j' = e(j)} \left( \theta_j^a(v'_{jj'}) + \sum_{j'' \in V_1: j'' \neq j} c_{jj''}^a(y'_{jj''}) \right) + \sum_{i \in I} c_{1ji}^a(x'_{1j'i}) + \sum_{k \in V_2, k' = d(k)} \left( \omega_k^a(u'_{kk'}) + \sum_{j \in V_1} c_{2kj}^a(x'_{2k'j}) \right) \quad (22)$$

subject to

$$u'_{kk'} + x'_{ki'} = b_{2k} \quad k \in V_2, k' = d(k), \quad (23)$$

$$u'_{kk'} = \sum_{j \in V_1} x'_{2k'j} \quad k' \in V_2', k = d'(k'), \quad (24)$$

$$\sum_{k' \in V_2'} x'_{2k'j} + \sum_{j'' \in V_1: j'' \neq j} y'_{j'j''} = v'_{jj'} \quad j \in V_1, j' = e(j), \quad (25)$$

$$v'_{jj'} = \sum_{j'' \in V_1: j'' \neq j} y'_{j'j''} + \sum_{i \in I} x'_{1j'i} \quad j' \in V_1', j = e'(j'), \quad (26)$$

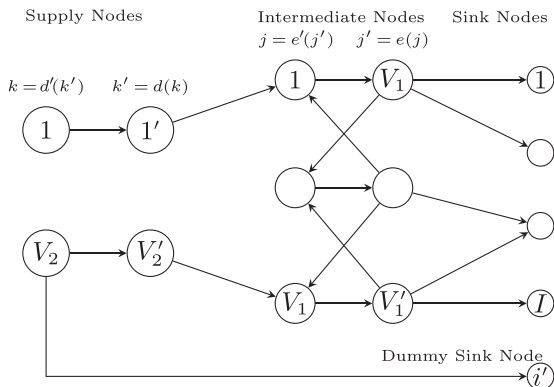
$$\sum_{j' \in V_1'} x'_{1j'i} = D_i \quad i \in I, \quad (27)$$

$$\sum_{k \in V_2} x'_{ki'} = D_{i'} \quad (28)$$

$$L_{kk'}^a \leq u'_{kk'} \leq U_{kk'}^a \quad k \in V_2, k' = d(k), \quad (29)$$

$$L_{jj'}^a \leq v'_{jj'} \leq U_{jj'}^a \quad j \in V_1, j' = e(j), \quad (30)$$

**Figure 1.** Network Flow Representation of  $LF_2^a$





$$L_{2k'j}^a \leq x'_{2k'j} \leq U_{2k'j}^a \quad k' \in V'_2, j \in V_1, \quad (31)$$

$$L_{j'j}^a \leq y'_{j'j} \leq U_{j'j}^a \quad j' \in V'_1, j \in V_1 : j \neq e'(j'), \quad (32)$$

$$L_{1j'i}^a \leq x'_{1j'i} \leq U_{1j'i}^a \quad j' \in V'_1, i \in I, \quad (33)$$

$$u'_{kk'} \geq 0, v'_{jj'} \geq 0 \quad k \in V_2, k' = d(k), j \in V_1, j' = e(j), \quad (34)$$

$$x'_{2k'j} \geq 0, x'_{1j'i} \geq 0, x'_{k'v'} \geq 0, y'_{j'j} \geq 0 \\ k' \in V'_2, j \in V_1, j' \in V'_1, i \in I, k \in V_2. \quad (35)$$

The costs terms in Objective (22) are in the same order as in (21). Constraints (23)–(28) are flow conservation at node sets  $V_2$ ,  $V'_2$ ,  $V_1$ ,  $V'_1$ ,  $I$  and  $i'$ , respectively. Bounds on variables associated with concave functions are imposed in (29)–(33). Finally, Constraints (34) and (35) impose nonnegativity restrictions on flow variables.

The main advantage of using  $LF_1^a$  over  $LF_2^a$  as a bounding procedure is that it can provide stronger lower bounds at the nodes of the enumeration tree. From Figure 2, (a) and (b), at any point  $\hat{x}$ , the error deviation  $\delta_N = F + f(\hat{x}) - \theta_N \hat{x}$  in  $LF_2^a$  is greater than the error deviation  $\delta_M = f(\hat{x}) - \theta_M \hat{x}$  in  $LF_1^a$ . This difference arises given that the fixed cost of opening a facility is included in the LUF of  $LF_2^a$  but not in the LUF of  $LF_1^a$ . However, to obtain these stronger bounds from  $LF_1^a$ , we need to solve an MILP at each node of the tree, which is significantly more time consuming compared with solving an MCNFP with a specialized network solver. In Section 5.2, we show that using  $LF_2^a$  as a bounding procedure allows the BB algorithm to explore more nodes in the enumeration tree in the same amount of time. This helps improve the lower and upper bounds more quickly and hence has the potential impact to decrease the computational time to solve an instance. In Section 5, we perform experiments to assess the computational benefits of using both bounding procedures.

#### 4.4. Preprocessing Phase

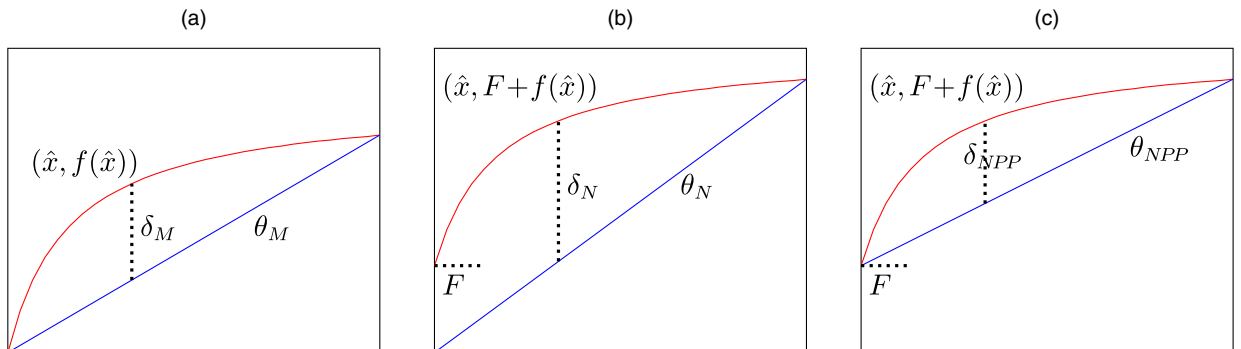
We recall that  $LF_1^a$  at a given node  $N^a$  corresponds to a two-level capacitated discrete location problem, which can take a significant amount of computing time when solving large-size instances. One way to reduce this time is by fixing as many binary location variables as possible before enumeration begins. Fixing location decisions can also have a positive impact in reducing the computation time for solving  $LF_2^a$ . In this case, when a facility is permanently closed, we can eliminate all the inflow/outflow from that facility, whereas if a facility is permanently opened, the cost functions in Equation (8) at that facility reduces to  $\Theta_j(v_j) = f_{1j} + W_j(v_j)$  and  $\Omega_k(u_k) = f_{2k} + P_k(u_k)$ . That is, the constant terms in the LUFs ( $\theta_j^a(v_j)$  and  $\omega_k^a(u_k)$ ) always include the fixed costs of facilities, and the slope is calculated on the concave operating cost only. As a consequence, at any point  $\hat{x}$ ,  $\theta_{NPP} < \theta_N$ , because of which the linear underestimation error  $\delta_{NPP}$  (Figure 2(c)) is lower than  $\delta_N$  (Figures 2(b)), thus providing stronger lower bounds.

The preprocessing phase begins with solving  $LF_1^0$  to provide a valid upper bound  $\bar{v}(N^0)$  and location decision vector  $\hat{z}$ . It then proceeds to temporarily fixing a facility decision variable  $z = 1$  (or 0) if  $\hat{z} = 0$  (or 1) and reoptimize  $LF_1^0$ . If the lower bound  $\underline{v}(N^0)$  obtained after reoptimizing is greater than  $\bar{v}(N^0)$ , then we permanently fix  $z = 0$  (or 1) in subsequent problems of the preprocessing phase and during enumeration. Let  $z_{1j}^F$  and  $z_{2k}^F$  be the decision vectors for fixing facilities at the first and second levels, respectively. The preprocessing phase is depicted in Algorithm 2.

##### Algorithm 2 (Preprocessing Phase)

- 1: Solve  $LF_1^0$  to obtain  $\bar{v}(N^0)$ ,  $\hat{z}_{1j}$  and  $\hat{z}_{2k}$ .
- 2: **for**  $k = 1$  to  $|V_2|$  **do**
- 3:   Temporarily set  $z_{2k} \leftarrow 1$  (0) if  $\hat{z}_{2k} = 0$  (1)
- 4:   Solve  $LF_1^0$  by considering  $z_{2k'}^F \forall k' = \{1, \dots, k-1\}$  that are already fixed
- 5:   **if**  $(\underline{v}(N^0) > \bar{v}(N^0))$  **then** update  $z_{2k}^F \leftarrow 0$  (1)

**Figure 2.** (Color online) Underestimation of Formulation at Node  $N^a$



Notes. (a)  $LF_1^a$ . (b)  $LF_2^a$ . (c)  $LF_2^a$  after preprocessing.



```

6: end for
7: for  $j = 1$  to  $|V_1|$  do
8:   Temporarily set  $z_{1j} \leftarrow 1$  (0) if  $\hat{z}_{1j} = 0$  (1)
9:   Solve  $LF_1^0$  by considering  $z_{2k}^F$  and  $z_{1j'}^F, \forall j' =$ 
      $\{1, \dots, j-1\}$  that are already fixed
10:  if  $(\underline{v}(N^0) > \bar{v}(N^0))$  then update  $z_{1j}^F \leftarrow 0$  (1)
11: end for

```

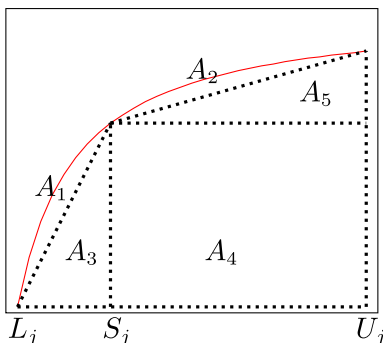
#### 4.5. Partitioning Point Selection

At the  $r$ th iteration of Algorithm 1, the node selected for branching is partitioned into two child nodes that differ from their parent node and from each other in terms of the lower and upper bounds to be set on the variable selected for spatial branching. We have empirically tested two strategies to select the branching point  $S_j$ . The first strategy to select the branching point is the same as in Soland (1974), in which  $S_j = \hat{x}_j$ , where  $j$  is the index of variable  $x$  that gives the maximum deviation (see Section 4.1), and  $\hat{x}_j$  is the solution value of variable  $x_j$  in the LUP at the parent node. We denote it as the *solution point* (SP) strategy. The advantage of this spatial branching strategy is that when the solution of the two problems at child nodes are close to that of the parent node, it has a positive impact in improving the lower bound quickly. We consider a second spatial branching strategy in which we select a point such that the sum of the area between a concave function and its respective linear underestimation of the resulting child problems is minimum (Liu, Sahinidis, and Shtetman 1996). This is referred as the *minimum area point* (MAP) strategy (Figure 3). The MAP  $S_j$  is obtained by solving the following continuous optimization problem:

$$\begin{aligned}
 (CP) \quad & \min_{S_j} \{A_1 + A_2 : L_j \leq S_j \leq U_j\} \\
 & \equiv \max_{S_j} \{A_3 + A_4 + A_5 : L_j \leq S_j \leq U_j\}.
 \end{aligned}$$

In Section 5, we perform an extensive comparison of both partitioning strategies to determine under which configuration of input parameters one strategy may dominate the other.

**Figure 3.** (Color online) Minimum Area Point Spatial Branching Strategy



#### 4.6. Overview of BB Algorithm for the 2CFLP-C

We now explain how to combine all ingredients of our algorithm into a unified BB framework capable of using either  $LF_1$  or  $LF_2$  as bounding procedures for solving the 2CFLP-C. A pseudo-code of our BB algorithm is depicted in Algorithm 3. We define  $\underline{v}^*$  and  $\bar{v}^*$  as the values of best lower and upper bounds of an instance of the 2CFLP-C, respectively, and  $(z, x, y, u, v)^*$  the incumbent solution. The input of the algorithm is the desired bounding strategy:  $LF_1$  or  $LF_2$ . We initialize each element  $z_{1j}^F$  and  $z_{2k}^F$  equal to  $-1$ . A first (second) level facility  $j$  ( $k$ ) not fixed at the end of preprocessing step is represented by  $z_{1j}^F = -1$  ( $z_{2k}^F = -1$ ). Next, regardless of the bounding choice, we first solve  $LF_1^0$  to obtain an upper bound  $\bar{v}^h$  and facility decision vectors  $z_{1j}^h$  and  $z_{2k}^h$ . The choice of spatial branching strategy and use of the preprocessing phase depend on simple but effective rules that exploit information generated by the root node  $LF_1^0$ . In particular, if the contribution of fixed cost for opening facilities in the objective function value  $\bar{v}^h$  is smaller or equal to a user-defined threshold percentage  $\tau_1$  of the total cost, then the SP branching strategy is selected. Otherwise, the MAP strategy is used. Similarly, if the contribution of fixed cost for opening facilities is larger or equal to a user-defined threshold percentage  $\tau_2$  of the total cost, then the preprocessing phase is executed. Otherwise, this phase is not used, and the branching phase starts. Based on the spatial branching strategy selected in Step 3,  $z_{1j}^F$  and  $z_{2k}^F$  vectors obtained in Step 4, and bounding procedure in Step 1, we start the BB algorithm by solving the root node in  $LF_i^0$  in Step 6 and the rest of the algorithm works the same as Algorithm 1.

**Algorithm 3** (Enhanced Branch-and-Bound Algorithm for the 2CFLP-C)

```

1: Input: Bounding procedure  $i \leftarrow 1 \vee 2$ 
2: Initialize:  $z_{1j}^F = -1$  and  $z_{2k}^F = -1$ 
3: Solve  $LF_1^0$  and set  $\bar{v}^h \leftarrow \bar{v}(N^0); z_{1j}^h \leftarrow \hat{z}_{1j}^0; z_{2k}^h \leftarrow \hat{z}_{2k}^0$ 
4: if  $(\sum_{j \in V_1} f_{1j} z_{1j}^h + \sum_{k \in V_2} f_{2k} z_{2k}^h) / \bar{v}^h \leq \tau_1$ 
   then Select Spatial Branching Strategy SP
   else Select Spatial Branching Strategy MAP
5: if  $(\sum_{j \in V_1} f_{1j} z_{1j}^h + \sum_{k \in V_2} f_{2k} z_{2k}^h) / \bar{v}^h \geq \tau_2$ 
   then Perform Preprocessing Phase using  $\bar{v}^h, z_{1j}^h$ 
     and  $z_{2k}^h$  to update  $z_{1j}^F$  and  $z_{2k}^F$ 
6: Initialize  $r \leftarrow 0; \pi \leftarrow \emptyset; N^s \leftarrow N^0$ 
7: Solve  $LF_i^0$  with  $z_{1j}^F$  and  $z_{2k}^F$  as input and set
    $\underline{v}^* \leftarrow \underline{v}(N^0); \bar{v}^* \leftarrow \bar{v}(N^0); (z, x, y, u, v)^* \leftarrow (\hat{z}, \hat{x}, \hat{y}, \hat{u}, \hat{v})^0; \rho \leftarrow \frac{(\bar{v}^* - \underline{v}^*)}{\bar{v}^*}$ 
8: while  $\rho > \epsilon$  do
9:   Branch on parent node  $N^s$  to create child nodes
      $N^{2r+1}$  and  $N^{2r+2}$ 
10:  Solve  $LF_i^{2r+1}$  with  $z_{1j}^F$  and  $z_{2k}^F$  as input at left node
      $N^{2r+1}$ ;

```

- if  $\bar{v}(N^{2r+1}) < \bar{v}^*$  then  $\bar{v}^* \leftarrow \bar{v}(N^{2r+1})$  and  
 $(z, x, y, u, v)^* \leftarrow (\hat{z}, \hat{x}, \hat{y}, \hat{u}, \hat{v})^{2r+1}$   
 if  $\underline{v}(N^{2r+1}) < \bar{v}^*$  then  $\pi \leftarrow \pi \cup \{N^{2r+1}\}$   
 11: Solve  $LF_i^{2r+2}$  with  $z_{1j}^F$  and  $z_{2k}^F$  as input at right node  
 $N^{2r+2}$ :  
 if  $\bar{v}(N^{2r+2}) < \bar{v}^*$  then  $\bar{v}^* = \bar{v}(N^{2r+2})$  and  
 $(z, x, y, u, v)^* \leftarrow (\hat{z}, \hat{x}, \hat{y}, \hat{u}, \hat{v})^{2r+2}$   
 if  $\underline{v}(N^{2r+2}) < \bar{v}^*$  then  $\pi \leftarrow \pi \cup \{N^{2r+2}\}$   
 12: Select node  $N^s \leftarrow N^n$ , where  $n \in \arg \min \{\underline{v}(\pi_i) : t = 1, \dots, |\pi|\}$   
 13: Update  $\pi \leftarrow \pi \setminus \{N^s\}$ ;  $\underline{v}^* \leftarrow \underline{v}(N^s)$ ;  $\rho \leftarrow \frac{(\bar{v}^* - \underline{v}^*)}{\bar{v}^*}$ ;  
 $r \leftarrow r + 1$   
 14: **end while**

Although the basic ingredients of our BB algorithm are the same as in Soland (1974), the two algorithms differ in several ways. First, the set of constraints for LUPs is different. In Soland (1974), the objective function  $\sum_{i=1}^n \phi_i^a(x_i)$  of LUP,  $LP^a$ , at any node  $N^a$  is always optimized over the same feasible region given by the polytope  $D \cap G$ . In our BB algorithm, the feasible region at node  $N^a$  corresponds to  $D \cap G^a$ , which can strengthen the obtained lower bounds given that  $G^a \subseteq G$ . Moreover, before the enumeration process begins, our algorithm performs a preprocessing step that requires the solution of several integer linear relaxations of  $NF_1$ . Finally, in Soland (1974), the relaxation step corresponds to a linear program (LP), whereas our BB uses either a *mixed-integer linear program* (MILP) or an MCNFP, where the latter can be efficiently solved with ad hoc network solvers. Our algorithm also differs from the BB algorithm proposed by Horst and Thoai (1998). Their algorithm ensures the endpoints of the domain of a variable to be integer (i.e., integral rectangular partitioning); however, we do not impose such restriction. Also, they select the variable with the largest domain for partitioning, and they branch at the midpoint of the domain.

Various BB algorithms similar to Algorithm 1 have been developed for other classes of concave minimization problems. For instance, Lamar (1993) provides a BB algorithm for MCCNFPs in which the concave functions are represented using piecewise linear segments. Fontes, Hadjiconstantinou, and Christofides (2006) also use the BB algorithm to solve single-source uncapacitated MCCNFPs in which the bounding step is solved by using a DP algorithm. Manousiouthakis, Thomas, and Justanieah (2011) present a BB algorithm in which the bounding step requires the solution of a convex problem obtained after approximating power law functions. This approximation imposes bounds on continuous variable, that is, the value of each variable must be greater than a very small value, making the method numerically unstable. Other BB algorithms are based on partitioning of simplices and cones (Benson 1996), whereas our method falls into

the category of rectangular partitioning, where the rectangles represent the domain of the function. Finally, Burer and Letchford (2012) discuss BB algorithms for nonconvex problems, and their main focus is on rectangular partitioning for spatial branching and convex envelopes for bounding steps, among other methods.

## 5. Computational Experiments

We conducted an extensive computational study to assess the empirical performance of different variants of Algorithm 3 described in Section 4. Our goal is to analyze and evaluate under which instance size, cost structure, and capacity scenario each formulation may provide a better performance in terms of the number of instances solved to proven optimality, final optimality gap (for unsolved instances), and computational time. We also compare the results of our BB algorithm with (i) BARON, a state-of-the-art commercial MINLP solver, and (ii) the exact BB algorithm introduced in Horst and Thoai (1998) for solving minimum concave-cost capacitated network flow problems. We also present a graphical analysis using real location data from the 3,109 counties of the contiguous United States to analyze the solution output of our model under different scenarios.

All algorithms are coded in C++ and executed on an Intel Xeon E5-2687W v3 processor at 3.10 GHz in a Linux environment. The algorithms are implemented using CPLEX 12.6 Concert Technology with its default settings using one thread. In what follows, we present a summary of the results of our experiments, whereas the detailed results are given in detailed tables included in the online appendix.

### 5.1. Benchmark Instances

To assess the performance of the solution algorithms, we generate a set of benchmark instances with different levels of solution difficulties by varying cost and capacity structures. For that, we vary the scaling parameters associated with fixed costs of the facilities ( $\alpha_1, \alpha_2$ ), variable costs at the facilities ( $\gamma_1, \gamma_2$ ), transportation cost between the facilities ( $\beta$ ), and capacity at the facilities ( $\kappa$ ). The instances are characterized as balanced cost (BC), dominant fixed cost (DFC), dominant variable cost (DVC), excess capacity (EC), and tight capacity (TC) scenarios. In the BC scenario, the facility fixed cost, facility variable cost, and transportation cost are fairly evenly distributed. In DFC scenario, the total fixed cost accounts for more than half of the total cost. In the DVC scenario, the total variable cost is more than half of the total cost. In the TC scenario, the sum of the capacities of the potential facilities at each level is set to three times the total demand, whereas it is seven times in the EC scenario.

**Table 2.** Scaling Parameters to Generate Instances Under Different Cost and Capacity Scenarios

Scenario	Parameter values					
	$\beta$	$\alpha_1$	$\alpha_2$	$\gamma_1$	$\gamma_2$	$\kappa$
Balanced cost (BC)	1	1	1	1	1	5
Dominant fixed cost (DFC)	1	$\sim U[4,6]$	$\sim U[4,6]$	1	1	5
Dominant variable cost (DVC)	$\sim U[4,6]$	1	1	$\sim U[4,6]$	$\sim U[4,6]$	5
Excess capacity (EC)	1	1	1	1	1	7
Tight capacity (TC)	1	1	1	1	1	3

The test problems are generated using a scheme similar to the one proposed in Vidyarthi et al. (2007). The coordinates for facilities and customers are generated uniformly as  $U[10,100]$ . Customer demand  $D_i$  is randomly generated with  $U[50,300]$ . The capacities of the potential facilities at the first level ( $b_{1j}$ ) and second level ( $b_{2k}$ ) are first generated uniformly on  $U[10,160]$  and then scaled such that  $\frac{\sum_{j \in V_1} b_{1j}}{\sum_{i \in I} D_i} = \frac{\sum_{k \in V_2} b_{2k}}{\sum_{i \in I} D_i} = \kappa$ , where,  $\kappa \geq 0$  is a scaling parameter varied to represent different capacity scenarios (e.g., tight capacity and excess capacity). Fixed costs of the first- and second-level facilities are set to  $f_{1j} = \alpha_1 \times (U[0,250] + U[10,100] \times (b_{1j})^{0.5})$  and  $f_{2k} = \alpha_2 \times (U[0,250] + U[10,100] \times (b_{2k})^{0.5})$ , respectively, where  $\alpha_1 \geq 0$  and  $\alpha_2 \geq 0$  are scaling parameters. We model various concave costs with power law functions  $f(x) = Ax^\lambda$ , where  $A$  is a nonnegative constant and exponent  $\lambda \in (0,1)$  ensures that  $f(x)$  is concave over its domain. We set  $\lambda = 0.5$  in our computational experiments unless otherwise stated. The transportation costs on the links are captured using the following concave functions:  $C_{1ji}(x_{1ji}) = \beta \times (0.2 \times d_{1ji})x_{1ji}^\lambda$ ,  $C_{2kj}(x_{2kj}) = \beta \times (0.2 \times d_{2kj})x_{2kj}^\lambda$  and  $C_{jj'}(y_{jj'}) = \beta \times (0.2 \times d_{jj'})y_{jj'}^\lambda$ , where  $d_{1ji}$ ,  $d_{2kj}$  and  $d_{jj'}$  are Euclidean distances between nodes and  $\beta \geq 0$  is a scaling parameter. Variable costs at the first- and second-level facilities are  $W_j(v_j) = \gamma_1 \times U[10,50] \times v_j^\lambda$  and  $P_k(u_k) = \gamma_2 \times U[10,50] \times u_k^\lambda$ , respectively, where  $\gamma_1 \geq 0$  and  $\gamma_2 \geq 0$  are scaling parameters. The scaling parameters ( $\beta, \alpha_1, \alpha_2, \gamma_1, \gamma_2$ , and  $\kappa$ ) are varied to generate instances with different cost and capacity scenarios as indicated in Table 2.

Table 3 provides the details of test problems. We classify these problems into five sets based on the number of potential facilities at the first and second levels and the number of customers, which is one of the main characteristics in assessing the difficulty of solving the models. For example, Set I consists of four problems in which the number of potential facilities at first level and second level are 40 and 10, respectively, whereas the number of customers varies from 200 (problem 1) to 500 (problem 4). The largest problem (20 of Set V) consists of 100 potential facilities at the first level, 50 potential facilities at the second level, and 2,250 customers. Next, for each of these 20 problems, we generate five instances, one for every scenario BC,

DFC, DVC, EC, and TC by varying the scaling parameters as shown in Table 2. These instances of a problem differ in their input parameters. Thus, our testbed comprises a total of 100 instances. These instances can be downloaded from the website: <https://users.ensc.concordia.ca/~icontr/~/web/Instances2CFLPC.7z>.

## 5.2. Computational Performance of BB Algorithm

We first evaluate the performance of two spatial branching strategies. We then analyze the impact of preprocessing on each of the two bounding procedures. Next, we report the performance of the BB algorithm using both bounding procedures  $LF_1$  and  $LF_2$  and compare the results with the BB algorithm proposed by Horst and Thoai (1998) under a variety of cost and capacity structures. Finally, we compare the results of our BB algorithm with that of BARON, a state-of-the-art solver for nonconvex optimization problems.

**5.2.1. Performance of Branching Strategies.** We compare the performance of two spatial branching strategies using both  $LF_1$  and  $LF_2$  bounding procedures. Throughout the computational experiments, we consider that the univariate concave functions are power law functions of the form  $f(x_j) = Ax_j^\lambda$ , where  $A$  is a nonnegative constant and  $\lambda \in (0,1)$ . Let  $L_j$  and  $U_j$  be the lower and upper bounds, respectively, on variable  $x_j$ . Next, we solve CP to provide a closed form expression to compute the MAP,  $S_j$ , for a concave power law function as follows:

$$S_j = \left[ \frac{1}{\lambda} \left( \frac{U_j^\lambda - L_j^\lambda}{U_j - L_j} \right) \right]^{\frac{1}{\lambda-1}}.$$

For this set of experiments, we select 30 medium-size instances and solve them using the proposed BB

**Table 3.** Details of Test Instances

Set	Problem	No. of potential facilities		
		Level 1 (V <sub>1</sub> )	Level 2 (V <sub>2</sub> )	No. of customers (I)
I	1, 2, 3, 4	40	10	200, 300, 400, 500
II	5, 6, 7, 8	55	20	600, 700, 800, 900
III	9, 10, 11, 12	70	30	1,000, 1,100, 1,200, 1,300
IV	13, 14, 15, 16	85	40	1,400, 1,500, 1,625, 1,750
V	17, 18, 19, 20	100	50	1,875, 2,000, 2,125, 2,250

**Table 4.** Summary of Performance of Spatial Branching Strategies on MINLP and NLP Formulations

Bounding step	Scenario	Average gap (%)		Average time (seconds)			Average nodes		
		$SP_g$	$MAP_g$	$SP_t$	$MAP_t$	$\%Red_t$	$SP_n$	$MAP_n$	$SP/MAP$
$LF_1$	BC	0.15	0.16	Time <sup>a</sup>	Time	–	11,375	15,796	4/2
	DFC	0.11	0.10	34,409	24,271	29	1,915	1,439	1/5
	DVC	0.20	0.19	79,476	Time	–9	7,714	11,306	3/3
	EC	0.15	0.14	50,115	52,208	–4	3,179	4,528	2/4
	TC	0.40	0.39	Time	Time	–	7,651	8,821	2/4
$LF_2$	BC	0.11	0.14	63,697	Time	–36	157,383	205,430	6/0
	DFC	0.10	0.10	12,056	1,699	86	27,573	4,018	0/6
	DVC	0.13	0.14	61,433	Time	–41	166,500	193,880	5/1
	EC	0.10	0.11	33,257	42,846	–29	76,953	81,201	5/1
	TC	0.30	0.29	Time	Time	–	244,135	194,737	4/2

<sup>a</sup>Time = 86,400 seconds.

algorithm under various scenarios. The termination criteria are set to an optimality gap of 0.1% and a time limit of 86,400 seconds (24 hours). A summary of results is presented in Table 4, whereas the detailed results are reported in Table 11 in the online appendix. The columns  $SP_g$  and  $MAP_g$  report the average optimality gap (%) obtained from the SP and MAP branching strategies, respectively, whereas the columns  $SP_t$  and  $MAP_t$  report average CPU time in seconds. The reduction in the computational time is reported in the column  $\%Red_t$ , which is computed as follows:  $\%Red_t = \frac{(SP_t - MAP_t) \times 100}{SP_t}$ . For instances where the BB algorithm did not converge to an optimality gap of 0.1% within a prescribed time limit, we write “Time” in the corresponding entry of the table. The columns  $SP_n$  and  $MAP_n$  report the average number of nodes explored in the BB algorithm. Finally, in the last column we use the notation  $n/m$  to indicate that out of  $(n + m)$  instances, SP outperformed MAP on  $n$  instances and MAP outperformed on  $m$  instances.

From these results, we observe that the efficient spatial branching strategy for an instance is characterized by the cost and capacity structures of the instance. MAP strategy outperforms SP using both bounding procedures under DFC scenario, where it not only reduces the CPU time (29% using  $LF_1$  and 86% using  $LF_2$ ) but also explores fewer nodes using both formulations. In the DFC scenario, the average fixed cost is 65% (Table 14 in the online appendix) of the total cost. Hence, it is likely that the facilities selected in the solutions obtained at nodes early in the BB algorithm are closer to the optimal network. Thus, we are primarily left with improving the relaxation, which is, by definition, tighter when using the MAP strategy. However, the effectiveness of MAP strategy decreases as the variable cost percentage increases in the feasible solution. This is evident from the DVC scenario in which using  $LF_2$ , SP reduces CPU time by 41% and explores 16% fewer nodes. In DVC scenario, transportation cost on an average is 52% (Table 15 in the online appendix) of

the total cost; therefore, the focus of BB algorithm is to select the right set of arcs and their corresponding flows from the very beginning. As the SP strategy considers the solution of the parent node while partitioning, the likelihood of the child node solution at later stages of the BB tree being closer to initial solutions increases, resulting in efficient branching and fewer explored nodes. Therefore, to identify the right spatial branching strategy we use the upper bound obtained at the root node  $LF_1^0$  (see Algorithm 3). In particular, we determine the percentage of the facility fixed costs in the upper bound, and if it is greater than  $\tau_1 = 0.5$ , we use MAP branching strategy. Otherwise, we use the SP strategy.

**5.2.2. Performance of Preprocessing.** In the second set of experiments, we analyze the performance of preprocessing when used with both bounding procedures  $LF_1$  and  $LF_2$ . For this, we use the same set of instances and termination criteria as in the previous experiments. The results are summarized in Table 5 and the detailed results are provided in Table 12 in the online appendix. Columns  $WoP_g$  and  $WoP_t$  report the average optimality gap and CPU time without preprocessing, respectively. Similarly, columns  $WP_g$  and  $WP_t$  report the average optimality gap and CPU time with preprocessing, respectively. The column  $\%Red_t$  reports the percentage reduction obtained in CPU time as a result of preprocessing and is computed as  $\%Red_t = \frac{(WoP_t - WP_t) \times 100}{WoP_t}$ . The columns  $WoP_n$  and  $WP_n$  report the average number of nodes explored in the BB algorithm, respectively. In the last column  $WoP/WP$ , we use  $n/m$  notation to indicate that, out of  $(n + m)$  instances,  $n$  instances performed better without preprocessing and  $m$  with preprocessing.

For instances in the BC scenario, preprocessing reduces CPU times by 17% and 21% using  $LF_1$  and  $LF_2$ , respectively. Preprocessing fixes 64% of facilities in BC scenario (Table 6), which helps BB algorithm in



**Table 5.** Summary of Performance of Preprocessing Phase

Bounding step	Scenario	Average gap (%)		Average time (seconds)			Average nodes		
		WoP <sub>g</sub>	WP <sub>g</sub>	WoP <sub>t</sub>	WP <sub>t</sub>	%Red <sub>t</sub>	WoP <sub>n</sub>	WP <sub>n</sub>	WoP/WP
LF <sub>1</sub>	BC	0.15	0.14	Time <sup>a</sup>	71,834	17	11,375	27,252	0/6
	DFC	0.10	0.10	24,271	8,847	64	1,439	2,921	1/5
	DVC	0.20	0.20	79,476	81,610	−3	7,714	8,126	3/3
	EC	0.15	0.12	50,115	47,203	6	3,179	14,177	0/6
	TC	0.40	0.38	Time	Time	–	7,651	12,362	0/6
LF <sub>2</sub>	BC	0.11	0.11	63,697	50,043	21	157,383	150,507	1/5
	DFC	0.10	0.10	1,699	2,428	−43	4,018	2,478	4/2
	DVC	0.13	0.14	61,433	71,883	−17	166,500	155,796	6/0
	EC	0.10	0.09	33,257	26,341	21	76,953	85,244	3/3
	TC	0.30	0.29	Time	Time	–	244,135	242,181	2/4

<sup>a</sup>Time = 86,400 seconds.

exploring a larger number of nodes (16,877) in less time. Because of this, the BB algorithm solves two of six instances to optimality with preprocessing and none without for BC scenario. We can observe that preprocessing does not improve performance in instances under DVC scenario using both bounding procedures and under DFC using LF<sub>2</sub>. We do not recommend preprocessing for DVC scenario because it fixes only 6% of facilities (Table 6), whereas for DFC we do, because preprocessing fixes 93% of facilities in instances under the DFC scenario. Similarly, we recommend preprocessing for the BC, EC, and TC scenarios as well. To decide whether to perform preprocessing in our implementation, we first determine the percentage of fixed cost in the upper bound obtained from the root node LF<sub>1</sub><sup>0</sup>. If this percentage is less than  $\tau_2 \leq 0.2$ , then we categorize the instance as DVC and do not use the preprocessing phase for that instance. Otherwise, we use it.

**5.2.3. Comparison with Generic BB Algorithms.** We next compare the results of our BB algorithm with BARON when used to solve NF<sub>1</sub> directly. BARON currently does not support discontinuous concave functions, and therefore, we cannot use it for solving NF<sub>2</sub> directly. The termination criteria are set to an optimality gap of 0.2% and a time limit of 86,400

seconds. The results for Set I instances are reported in Table 7. These results indicate that BARON takes excessive time to solve even a small-size instance to optimality. Moreover, even for Set I instances, BARON yields large optimality gaps (overall average of 21.27%) in one day of CPU time.

We now analyze the CPU time required to obtain solutions with different optimality gaps: 1%, 0.5%, 0.2%, and 0.1%. For this, we choose LF<sub>1</sub> as the bounding procedure as its CPU times are significantly higher than LF<sub>2</sub>. The results of 30 medium-size instances under various scenarios and efficient branching and preprocessing strategies are presented in Table 8. For the considered instances, the algorithm converges to an optimality gap of 1% in less than one hour. However, as we attempt to close this optimality gap further, the CPU time increases many folds. For example, in the BC scenario, the average CPU time to reach 0.2% and 0.1% optimality gaps are 12 and 30 times higher, respectively. We observe a similar multifold increase in CPU times in other scenarios as well. Thus, in the remainder of our experiments, we set the termination criteria to an optimality gap of 0.2% and a time limit of 86,400 seconds.

In the next set of computational experiments, we compare the performance of Algorithm 3 with that of the BB algorithm presented in Horst and Thoai (1998).

**Table 6.** Effect of Preprocessing on the Percentage of Facilities Fixed at the Root Node

Problem	Percentage of facilities fixed under scenario				
	BC (%)	DFC (%)	DVC (%)	EC (%)	TC (%)
9	59	98	4	70	50
10	62	98	18	59	62
11	83	95	2	59	53
12	68	90	0	86	49
13	39	90	11	54	26
14	70	85	2	66	46
Average	64	93	6	66	48

**Table 7.** Summary of Performance of BARON Solver and BB Algorithm for Set I Instances

Scenario	Average gap (%)		Average time (seconds)	
	BARON <sub>g</sub>	LF <sub>2g</sub>	BARON <sub>t</sub>	LF <sub>2t</sub>
BC	24.01	0.20	Time <sup>a</sup>	78
DFC	3.61	0.19	45,336	23
DVC	32.23	0.19	Time	87
EC	20.66	0.19	Time	34
TC	25.86	0.20	Time	638
Average	21.27	0.19	78,188	172

<sup>a</sup>Time = 86,400 seconds.

**Table 8.** CPU Times to Reach Different Optimality Gaps Using Bounding Procedure  $LF_1$ 

Scenario	Branching strategy	With or without preprocessing	Average time (seconds)			
			Gap = 1%	Gap = 0.5%	Gap = 0.2%	Gap = 0.1%
BC	SP	WP	2,348	2,913	29,132	71,816
DFC	MAP	WP	1,352	1,353	1,585	8,847
DVC	SP	WoP	2,211	7,823	52,072	81,598
EC	SP	WP	2,837	4,721	19,816	47,199
TC	SP	WP	1,496	10,900	Time <sup>a</sup>	Time
Average			2,049	5,542	37,801	59,712

<sup>a</sup>Time = 86,400 seconds.

A summary of their performance at root node and at termination is given in Tables 9 and 10, respectively. The detailed results are provided in Tables 13–17 in the online appendix. In Table 9, the first column lists the scenario, and the next three columns provide the average gap at the root node. This gap is calculated using lower bound obtained from every method at the root node against the best upper bound value over all three methods at the termination. The main idea here is to compare the strengths of the bounding procedures  $LF_1$ ,  $LF_2$ , and the one used in the Host and Thoai (H&T) method. The last three columns report the average computation time taken (in seconds) to solve the root node. The averages are reported over all 20 instances in each scenario. The average CPU time to solve the root node of the problem is lower for  $LF_2$  and H&T methods compared with  $LF_1$ . However, the average optimality gap at the root node for  $LF_1$  is 0.86%, whereas that of  $LF_2$  and H&T methods are 2.40% and 2.89%, respectively. These results substantiate our claim that  $LF_1$  yields the lowest average optimality gap at the root node among the three methods. This is because, in the LUP for  $LF_1$ , fixed costs are modeled independently of concave functions, and hence, it provides a tighter linear underestimation of concave function (Figure 2).

Table 10 summarizes the performance of the BB algorithms at termination. The first and second columns indicate the scenarios and sets of instances, respectively. The next three columns report the average optimality gap at termination. In the sixth column,

we report the presolve time (in secs), which includes the average time taken by preprocessing step before the BB algorithm begins. The average time taken to reach an optimality gap of 0.2% is presented in the subsequent three columns under “Average time(s).” The times reported in columns  $LF_{1t}$  and  $LF_{2t}$  include the presolve time. The column “%Red<sub>t</sub>” refers to the percentage reduction in CPU time of  $LF_2$  compared with  $LF_1$ . The average number of nodes solved in the BB tree is reported in the next two columns. In the last column “H/M/N,” we use  $h/m/n$  notation to report that the H&T method performs better in  $h$ ,  $LF_1$  in  $m$  and  $LF_2$  in  $n$  number of instances out of  $(h + m + n)$  instances.

The results in Table 10 show that the average gap of the H&T’s algorithm, at termination, is 4.22% compared with 0.25% when using any of the bounding procedures in our BB algorithm. The average computational time of the H&T method is more than two times that of  $LF_1$  and three times that of  $LF_2$ . Moreover, the H&T method was unable to solve medium- and large-size instances (Sets III, IV, and V) under any of the five scenarios. The superior performance of our BB algorithm both in terms of optimality gap and CPU times can be attributed to the following reasons:

- Preprocessing plays a significant role in the better performance of the BB algorithm over the H&T’s BB algorithm. As preprocessing not only facilitates solving the subproblem faster but also steers the BB algorithm in the right direction because many of the location decisions are fixed and the solution space reduces to the facilities that may appear in the final solution. This is most evident from the results of DFC scenario, in which nearly 93% of the facilities are fixed (Table 6).

- Our BB algorithm considers the solution of the current node and the approximation errors for each concave function while selecting the branching variable and partitioning point. Whereas in the H&T method the variable corresponding to the concave function with the largest domain (i.e., the difference between the lower and upper bounds) is selected for branching and the branching point is the midpoint of the domain. The benefits of this can be noted from the results of the

**Table 9.** Summary of Performance of BB Algorithms at the Root Node

Scenario	Average gap (%)			Average time (seconds)		
	H&T <sub>rg</sub>	$LF_{1rg}$	$LF_{2rg}$	H&T <sub>rt</sub>	$LF_{1rt}$	$LF_{2rt}$
BC	2.75	0.89	2.32	2	27	<1
DFC	3.07	0.41	1.73	2	7	<1
DVC	3.55	1.32	3.55	2	30	<1
EC	3.43	0.99	2.87	2	12	<1
TC	1.65	0.68	1.51	2	28	<1
Average	2.89	0.86	2.40	2	21	<1

**Table 10.** Summary of Performance of BB Algorithms at Termination

Scenario	Set	Average gap (%)			Average time (seconds)					Average nodes		
		H&T <sub>g</sub>	LF <sub>1g</sub>	LF <sub>2g</sub>	Presolve	H&T <sub>t</sub>	LF <sub>1t</sub>	LF <sub>2t</sub>	%Red <sub>t</sub>	LF <sub>1n</sub>	LF <sub>2n</sub>	H&T/ LF <sub>1</sub> / LF <sub>2</sub>
BC	I	0.19	0.18	0.20	48	2,867	90	78	13	268	519	0/1/3
	II	0.54	0.20	0.19	222	73,812	689	467	32	547	1,432	0/0/4
	III	2.34	0.20	0.20	1,051	Time <sup>a</sup>	7,564	3,774	50	2,066	9,297	0/0/4
	IV	4.04	0.23	0.20	3,204	Time	76,646	32,960	57	9,160	59,410	0/0/4
	V	11.89	0.41	0.28	9,168	Time	Time	64,853	25	1,995	81,266	0/0/4
	Average	<b>3.80</b>	<b>0.24</b>	<b>0.21</b>	<b>2,739</b>	<b>67,178</b>	<b>34,289</b>	<b>20,426</b>	<b>40</b>	<b>2,807</b>	<b>30,385</b>	<b>0/1/19</b>
DFC	I	0.20	0.19	0.19	21	2,372	22	23	−5	21	23	0/3/1
	II	1.19	0.19	0.19	177	Time	236	226	4	200	252	0/0/4
	III	0.57	0.20	0.20	583	Time	668	647	3	108	178	0/1/3
	IV	3.23	0.20	0.20	2,412	Time	5,797	2,885	50	627	910	0/0/4
	V	8.69	0.20	0.20	9,373	Time	38,411	14,625	62	3,778	9,267	0/0/4
	Average	<b>2.77</b>	<b>0.20</b>	<b>0.20</b>	<b>2,513</b>	<b>69,597</b>	<b>9,027</b>	<b>3,681</b>	<b>59</b>	<b>947</b>	<b>2,126</b>	<b>0/4/16</b>
DVC	I	0.51	0.20	0.19	1	25,499	175	87	50	269	1,132	0/0/4
	II	1.17	0.19	0.19	7	79,802	2,104	2,241	−7	450	11,910	0/2/2
	III	9.05	0.20	0.19	10	Time	34,888	26,218	25	3,114	72,004	0/1/3
	IV	9.66	0.31	0.22	23	Time	Time	61,100	29	4,038	106,246	0/0/4
	V	16.34	0.58	1.04	22	Time	Time	Time	0	1,538	97,668	0/3/1
	Average	<b>7.35</b>	<b>0.29</b>	<b>0.36</b>	<b>13</b>	<b>72,901</b>	<b>42,003</b>	<b>35,210</b>	<b>16</b>	<b>1,882</b>	<b>57,792</b>	<b>0/6/14</b>
EC	I	0.16	0.18	0.19	26	545	34	34	0	50	96	0/3/1
	II	0.60	0.19	0.19	258	59,756	386	396	−3	53	613	0/3/1
	III	4.70	0.20	0.18	1,653	Time	4,339	3,577	18	624	6,506	0/1/3
	IV	4.44	0.20	0.20	2,806	Time	24,354	7,813	68	2,326	10,087	0/1/3
	V	10.06	0.21	0.20	3,968	Time	45,841	10,210	78	2,599	9,020	0/0/4
	Average	<b>3.99</b>	<b>0.20</b>	<b>0.19</b>	<b>1,742</b>	<b>63,902</b>	<b>14,991</b>	<b>4,406</b>	<b>71</b>	<b>1,130</b>	<b>5,264</b>	<b>0/8/12</b>
TC	I	0.19	0.19	0.20	34	24,704	2,708	638	76	19,821	15,427	0/1/3
	II	1.02	0.20	0.20	383	Time	71,761	24,236	66	38,352	217,117	0/0/4
	III	3.39	0.37	0.29	1,427	Time	Time	68,889	20	14,642	225,531	0/0/4
	IV	5.91	0.40	0.33	1,698	Time	Time	Time	0	5,086	156,671	0/0/4
	V	5.51	0.53	0.42	2,596	Time	Time	Time	0	1,693	84,012	0/0/4
	Average	<b>3.20</b>	<b>0.34</b>	<b>0.29</b>	<b>1,228</b>	<b>74,063</b>	<b>66,749</b>	<b>53,314</b>	<b>20</b>	<b>15,919</b>	<b>139,751</b>	<b>0/1/19</b>
	Average	<b>4.22</b>	<b>0.25</b>	<b>0.25</b>	<b>1,647</b>	<b>69,528</b>	<b>33,412</b>	<b>23,407</b>	<b>30</b>	<b>4,537</b>	<b>47,063</b>	<b>0/20/80</b>

<sup>a</sup>Time = 86,400 seconds. Bold entries are averages of the numbers for every scenario.

DVC scenario in which no preprocessing is performed for any of the methods and  $LF_2$  reduces the optimality gap by nearly 7% and computational time by 50%.

- The gaps at the root node from the H&T method and  $LF_2$  in Table 9 are of the same order and yet the bounding procedure  $LF_2$  performs exceptionally better at termination. Thus, it is evident that preprocessing, branching variable selection and spatial branching strategy employed in our BB algorithm are capable of finding high-quality upper bounds as well.

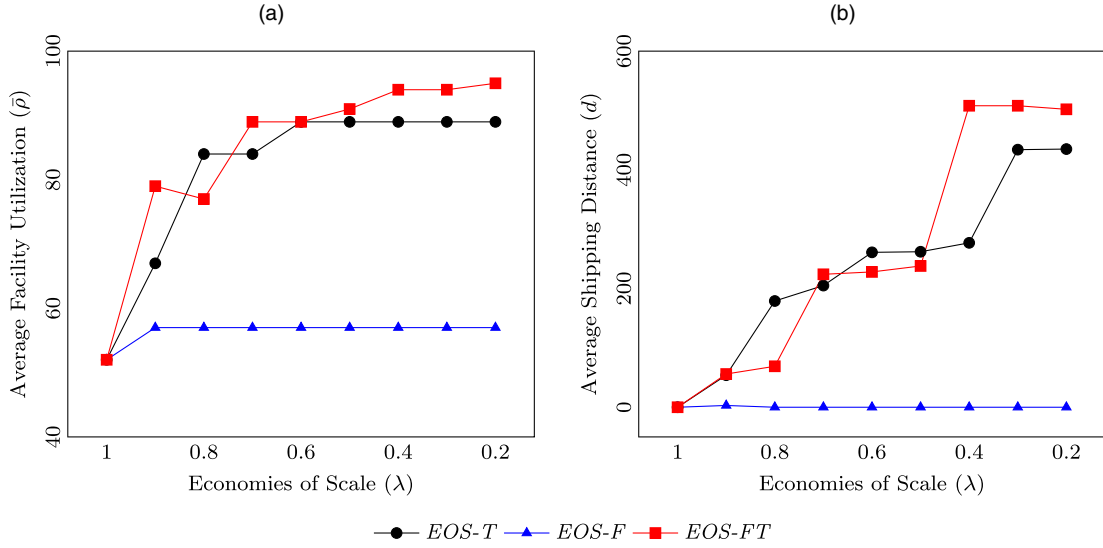
Finally, we compare the performance of the BB algorithm using both bounding procedures  $LF_1$  and  $LF_2$ . From the results, we observe that on average  $LF_2$  takes 30% less CPU time than  $LF_1$  and outperforms on 80 of 100 instances. The significant reduction in CPU times using  $LF_2$  are obtained in the EC (71%) and DFC scenarios (59%) followed by the BC scenario (40%). We also observe that  $LF_2$  is particularly faster on large-size instances. For example,  $LF_2$  reduces CPU times by 78%, and 62% under the EC and DFC scenarios, respectively, in the instances belonging to Set V. Similar savings in CPU time are also observed in Set IV instances, where the reductions are 68%, 57%, and

50% under the EC, BC, and DFC scenarios, respectively. Also,  $LF_2$  overall solves 79 instances to the desired optimality gap, whereas  $LF_1$  solves 70.

### 5.3. Sensitivity Analysis

In this section, we perform a sensitivity analysis of the proposed model to changes in the input parameters, especially the economies of scale (concave costs) in operations at the facilities and transportation. We provide insights by comparing the solutions (locations and allocation decisions) across various scenarios.

For this analysis, we consider a large-size instance using real location data available on the demographic information of the 3,109 counties in the contiguous United States. For every county, we obtain its population, latitude/longitude information from the United States Census Bureau (<https://www2.census.gov/geo/docs/reference/cenpop2010/county>), along with its average housing price (<https://www.census.gov/support/USACdataDownloads.html#HSG>). Each county represents an aggregate set of customers (demand points), and hence there are 3,109 demand points ( $|I| = 3,109$ ). We consider a two-level production-distribution system, where

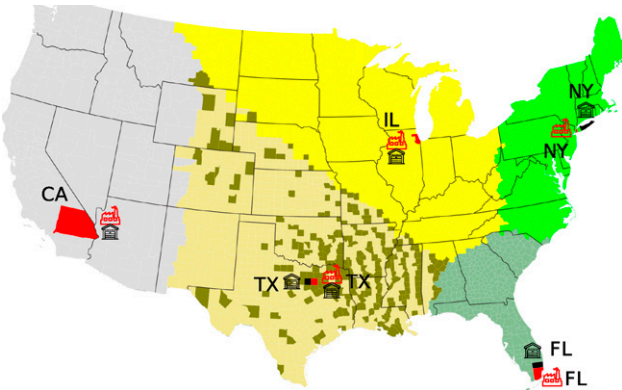
**Figure 4.** (Color online) Effect of Varying Economies of Scale on Average Facility Utilization and Shipping Distance

the first level represents distribution centers (DCs) and the second level represents plants. Thirty most populated counties are used as potential sites for DCs ( $|V_1| = 30$ ), and 15 most populated counties are used as potential sites for plants ( $|V_2| = 15$ ).

The demand at every customer node ( $D_i$ ) is obtained by dividing the county's population by 1,000. The fixed cost of opening a plant  $k$  is set proportional to the average residential prices as  $f_{2k} = 0.5 \times$  (county's average residential price). Similarly, the fixed cost of opening a DC  $j$  is set proportional to the average residential prices as  $f_{1j} = 0.2 \times$  (county's average residential price). Plant's capacity  $b_{2k}$  and DC's capacity  $b_{1j}$  are randomly generated as  $U[1.0, 1.5] \times f_{2k}$  and  $U[1.0, 1.5] \times f_{1j}$ , respectively. These capacities are then scaled such that the total plant capacities is 10 times the total demand, and the total DC capacities is

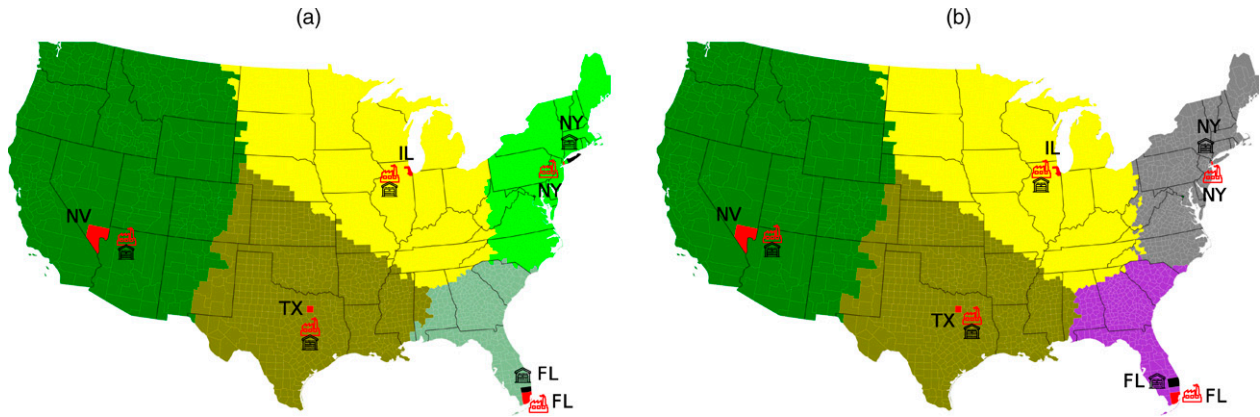
10 times the total demand. The transportation costs between two nodes are set as  $C_{1ji}(x_{1ji}) = 0.2 \times d_{1ji} \times x_{1ji}^\lambda$ ,  $C_{2kj}(x_{2kj}) = 0.2 \times d_{2kj} \times x_{2kj}^\lambda$  and  $C_{jj'}(y_{jj'}) = 0.2 \times d_{jj'} \times y_{jj'}^\lambda$ , where  $d_{1ji}$ ,  $d_{2kj}$  and  $d_{jj'}$  are obtained by dividing the spherical distance between the nodes by 100. Finally, we set production cost at facilities to  $P_k(u_k) = U[5, 36] \times u_k^\lambda$  and handling cost at the DC to  $W_j(v_j) = U[5, 36] \times v_j^\lambda$ . The economies of scale in operations at facilities and/or in transportation are captured using concave power law cost functions  $f(x) = A(x)^\lambda$ , where  $0 \leq \lambda \leq 1$  as described in Section 5.1.

In our sensitivity analysis, we vary  $\lambda$  to generate four scenarios. In the first scenario (referred to as base-case scenario), we set  $\lambda$  to one, resulting in linear production and handling costs at the facilities, as well as linear transportation costs on the links. In the second scenario (referred to as EOS-F), we set production cost  $P_k(u_k)$  and handling cost  $W_j(v_j)$  as concave functions of the flow, whereas transportation costs are linear. In the third scenario (referred to as EOS-T), we use concave transportation costs  $C_{2kj}(x_{2kj})$ ,  $C_{jj'}(y_{jj'})$  and  $C_{1ji}(x_{1ji})$ , whereas facility variable costs are linear. In the fourth scenario (referred to as EOS-FT), both facility variable costs and transportation costs are considered concave functions of flow. The network configurations for these scenarios are shown in Figures 5–8. Also, Figure 4(a) shows the effect of varying economies of scale on the average utilization ( $\bar{\rho}$ ) of the facilities (plants and DCs) in the network. The effect of economies of scale on the average shipping distance from plants to customers in the network is shown in Figure 4(b). In the remainder of the section, we compare the solution under different scenarios.

**Figure 5.** (Color online) Solution Without Economies of Scale ( $\lambda = 1$ )



**Figure 6.** (Color online) Solution with Economies of Scale in Operations at Facilities (EOS-F)



Notes. (a)  $\lambda = 0.9$ . (b)  $\lambda = 0.8$ .

### 5.3.1. Base-Case Scenario (No Economies of Scale).

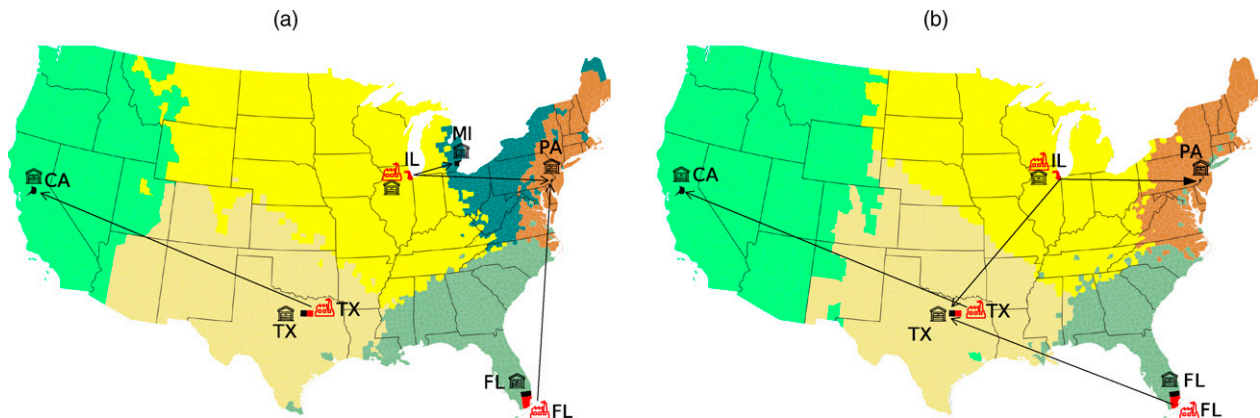
Figure 5 depicts the configuration of the two-level production-distribution network without any economies of scale in facilities operations and transportation (i.e.,  $\lambda = 1$  for all the facility variable costs at the plants and DCs and transportation costs between facilities). In this case, the optimal solution consists of 11 facilities, that is, five plants and six DCs. Counties where plants are located are marked with a plant symbol facility, whereas a county marked with black office-building symbol represents a DC. The total cost of the configuration is \$3,060,741, where the facility fixed cost is 34%, the facility variable cost is 25%, and the transportation cost is 41% of the total cost. The average utilization of plants and DCs is 41% and 68%, respectively, and the average outbound shipping distance (from DCs to customers) is 401 km/unit. Under this scenario, the model opens DCs very close to plants, e.g., Queens (NY) and Suffolk (NY) or in the same county in some cases, for example, in Cook County (IL). As the transportation cost dominates the

other two components in the base case, the model minimizes inbound transportation costs (plants to DCs) by collocating these facilities, and it reduces the outbound transportation cost by opening six facilities closer to demand regions. For instance, Cook (IL) is 2nd, Dallas (TX) is 9th, and San Bernardino (CA) is the 12th most populated county in our set of 30 counties as potential locations for DCs.

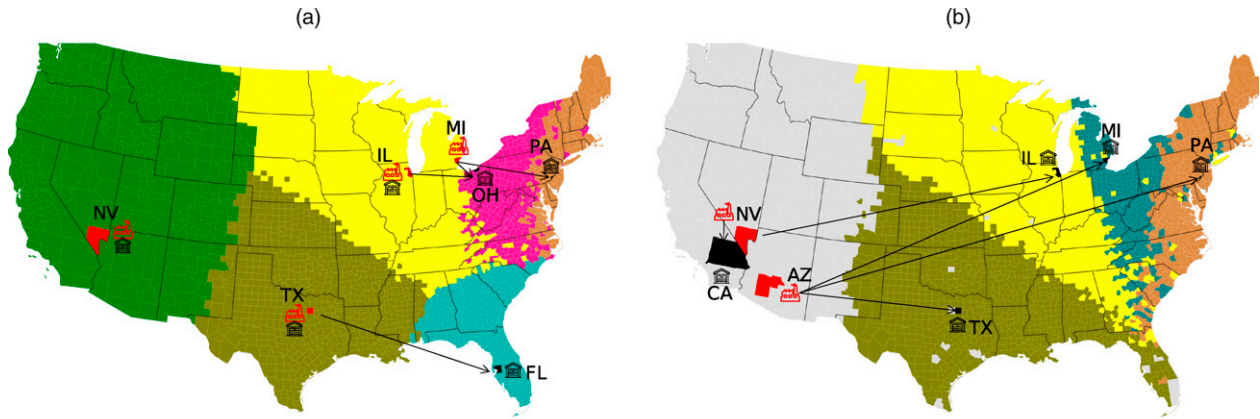
### 5.3.2. Effect of Economies of Scale in Facility Operations.

As we include economies of scale in facility operations, that is, production at plants and handling at DCs, the unit facility variable cost ( $c_f$ ) decreases from \$2.51/unit at  $\lambda = 1$  to \$0.88/unit at  $\lambda = 0.9$  and \$0.37/unit at  $\lambda = 0.8$ , resulting in a discount of 65% and 85%, respectively. The decrease in  $c_f$  prompts the consolidation of flows at DCs, which results in increasing DC utilization to 80% at  $\lambda = 0.9$  and closing DC (Figure 6(a)) in Tarrant (TX) and moving DC from San Bernardino (CA) to Clark (NV). Also, at  $\lambda = 0.9$ , transportation cost is 50% of the total cost; thus, the

**Figure 7.** (Color online) Solution with Economies of Scale in Transportation (EOS-T)



Notes. (a)  $\lambda = 0.8$ . (b)  $\lambda = 0.5$ .

**Figure 8.** (Color online) Solution with Economies of Scale in Both Operations at Facilities and Transportation (EOS-FT)

Notes. (a)  $\lambda = 0.8$ . (b)  $\lambda = 0.3$ .

model once again opens plants and DCs closer to each other, and DCs closer to highly populated counties to minimize transportation costs. Also, the facility variable cost is only 11% of the total cost obtained at  $\lambda = 0.9$ ; hence, further reduction in  $\lambda$  to 0.8 does not bring any major changes in the network (Figure 6(b)). It only moves two facilities in NY and FL to the nearby counties, and the average DC utilization stays around 80%. This is also reflected by the lines marked with triangles in Figure 4, (a) and (b), which show that further decrease in  $\lambda$  do not impact average utilization and average shipping distance, respectively.

### 5.3.3. Effect of Economies of Scale in Transportation.

Next, we introduce economies of scale only in transportation between the facilities and from DCs to the customers. For instance, when  $\lambda = 0.8$ , the unit transportation cost is  $c_t = \$2.02$ , which corresponds to a discount of 20% compared with the base-case scenario ( $\lambda = 1$ ). Similarly, for  $\lambda = 0.5$ ,  $c_t = \$0.44$  amounts to a discount of 82%. With discounts in transportation costs, the sourcing from distant facilities becomes economical; therefore, the model opens fewer plants at  $\lambda = 0.8$  (Figure 7(a)) compared with the base case. This increases the average utilization of plants to 87% and inbound shipping distance to 625 km. The model also tries to minimize the fixed cost by opening DCs that are both farther from highly populated areas and relatively smaller in size, for instance, DCs in Wayne (MI), Philadelphia (PA), and Sacramento (CA). This results in increasing DC utilization to 89%, and with further discounts at  $\lambda = 0.5$ , the model consolidates the flows even more by closing the DC in Wayne (MI) and increasing average DC utilization to 95% and inbound shipping distance to 832 km and outbound to 621 km.

**5.3.4. Effect of Economies of Scale in Facility Operations and Transportation.** Finally, we analyze the effect of introducing economies of scale in facility operations and transportation costs on the network configuration. Because we discount both  $c_f$  and  $c_t$ , the model simultaneously focuses on consolidation and opening economical facilities, which changes the solution significantly in comparison with the network at  $\lambda = 1$ . Figure 8, (a) and (b), illustrates the network configuration for  $\lambda = 0.8$  and 0.3, respectively. At  $\lambda = 0.8$ , the network comprises four plants and six DCs. The consolidation of facility operations and discount in inbound unit transportation cost leads to fewer plants in the network, which increases the plant utilization to 72% from 41% at  $\lambda = 1$ . Also, although the number of DCs opened is six, the majority of these are opened in less populated counties and are smaller in size, for example, Cuyahoga (OH) and Hillsborough (FL). Finally, at  $\lambda = 0.3$ , the fixed cost completely drives the solution, prompting the model to close two plants and one DC, and improving the average utilization of plants to 90% and DC utilization to 99% compared with the solution at  $\lambda = 0.8$  and 1.0.

To summarize, the three scenarios lead to different network configurations with EOS-FT mostly dominating the other two in terms of utilization and shipping distance (Figure 4, (a) and (b)). Moreover, in all three scenarios, we observe a tradeoff between facility utilization and shipping distance. As the value of  $\lambda$  decreases, the facility utilization certainly increases; however, it is coupled with longer shipping arcs, indirectly making the network prone to uncertainties in transportation. Additional details on the plants and DC locations, the cost structure of the network, and facilities' utilization for varying levels of economies of scale under every scenario are presented in Table 18 in the online appendix.

## 6. Conclusions

We studied a general class of two-level capacitated facility location problems with concave costs, where the concavity arises because of economies of scale in production and handling at the facilities and/or transportation between levels of facilities. We presented two formulations to model this class of problems. The first one is an MINLP and the second one is an NLP containing only continuous flow variables and discontinuous functions. One of the main contributions of this work was to computationally compare these two modeling approaches under the same solution framework. We developed an exact algorithm based on a branch-and-bound method to solve large-scale instances of the considered problem. This algorithm can use two lower bounding procedures: (i) an integer linear relaxation of the MINLP formulation and (ii) a linear relaxation of the NLP formulation. The algorithm was enhanced with a cost-dependent spatial branching strategy and preprocessing step to improve its convergence. Instances with up to 2,250 customers and 150 potential facilities, and two levels of hierarchy were solved to an optimality gap of 0.2% under varying costs and capacity structures. The results of extensive computational experiments confirm that the bounding procedure based on the NLP outperforms the one based on the MINLP on 80% of the considered instances, especially on medium- to large-size instances. A 30% reduction of average CPU time was obtained with the NLP-based bounding procedure as compared with the MINLP-based bounding procedure. We also analyzed the impact of varying economies of scale in the operations at the facilities and/or transportation on the facility location and customer allocation decisions. Our results clearly demonstrated that concave costs have a significant impact on the topology of solution networks.

## Acknowledgments

The authors thank the associate editor and two anonymous reviewers for valuable comments on a previous version of this paper.

## References

- Ahmadi-Javid A, Seyedi P, Syam SS (2017) A survey of healthcare facility location. *Comput. Oper. Res.* 79:223–263.
- Ahmed S, He Q, Li S, Nemhauser GL (2016) On the computational complexity of minimum-concave-cost flow in a two-dimensional grid. *SIAM J. Optim.* 26(4):2059–2079.
- Atamtürk A, Küçükyavuz S, Tezel B (2017) Path cover and path pack inequalities for the capacitated fixed-charge network flow problem. *SIAM J. Optim.* 27(3):1943–1976.
- Baumgartner K, Fuetterer A, Thonemann UW (2012) Supply chain design considering economies of scale and transport frequencies. *Eur. J. Oper. Res.* 218(3):789–800.
- Belotti P, Kirches G, Leyffer S, Linderoth J, Luedtke J, Mahajan A (2013) Mixed-integer nonlinear optimization. *Acta Numer.* 22:1–131.
- Benson HP (1996) Deterministic algorithms for constrained concave minimization: A unified critical survey. *Naval Res. Logist.* 43(6):765–795.
- Berman O, Krass D (2019) Stochastic location models with congestion. Laporte G, Nickel S, Saldanha da Gama F, eds. *Location Science*, 2nd ed. (Springer, Cham, Switzerland), 443–486.
- Burer S, Letchford AN (2012) Non-convex mixed-integer nonlinear programming: A survey. *Survey Oper. Res. Management Sci.* 17(2):97–106.
- Castro J, Nasini S (2021) A specialized interior-point algorithm for huge minimum convex cost flows in bipartite networks. *Eur. J. Oper. Res.* 290(3):857–869.
- Cohen MA, Moon S (1991) An integrated plant loading model with economies of scale and scope. *Eur. J. Oper. Res.* 50(3):266–279.
- Contreras I, Ortiz-Astorquiza C (2019) Hierarchical facility location problems. Laporte G, Nickel S, Saldanha da Gama F, eds. *Location Science*, 2nd ed. (Springer, Cham, Switzerland), 365–389.
- Contreras I, Cordeau JF, Laporte G (2011) The dynamic uncapacitated hub location problem. *Transportation Sci.* 45(1):18–32.
- D'Ambrosio C, Lodi A (2013) Mixed integer nonlinear programming tools: An updated practical overview. *Ann. Oper. Res.* 204(1):301–320.
- Dang C, Sun Y, Wang Y, Yang Y (2011) A deterministic annealing algorithm for the minimum concave cost network flow problem. *Neural Networks* 24(7):699–708.
- Drezner Z, Hamacher HW (2002) *Facility Location: Applications and Theory* (Springer Verlag, Berlin).
- Elhedhli S, Merrick R (2012) Green supply chain network design to reduce carbon emissions. *Transportation Res. Part D Transportation Environ.* 17(5):370–379.
- Falk JE, Soland RM (1969) An algorithm for separable nonconvex programming problems. *Management Sci.* 15(9):550–569.
- Farahani RZ, Hekmatfar M, Fahimnia B, Kazemzadeh N (2014) Hierarchical facility location problem: Models, classifications, techniques, and applications. *Comput. Industrial Engrg.* 68:104–117.
- Fischetti M, Ljubić I, Sinnl M (2016) Benders decomposition without separability: A computational study for capacitated facility location problems. *Eur. J. Oper. Res.* 253(3):557–569.
- Fontes DB (2008) On minimum concave cost network flow problems. *Internat. J. Pure Appl. Math.* 49(4):517–524.
- Fontes DB, Hadjiconstantinou E, Christofides N (2006) A branch-and-bound algorithm for concave network flow problems. *J. Global Optim.* 34(1):127–155.
- Fortz B (2015) Location problems in telecommunications. Laporte G, Nickel S, Saldanha da Gama F, eds. *Location Science*, 2nd ed. (Springer, Cham, Switzerland), 537–554.
- Guiseite G, Pardalos P (1990a) Algorithms for the uncapacitated single-source minimum concave-cost network flow problem. *Oper. Res.* 90:703–713.
- Guiseite GM, Pardalos PM (1990b) Minimum concave-cost network flow problems: Applications, complexity, and algorithms. *Ann. Oper. Res.* 25(1):75–99.
- Hajiaghay MT, Mahdian M, Mirrokni VS (2003) The facility location problem with general cost functions. *Networks* 42(1):42–47.
- Harkness J, ReVelle C (2003) Facility location with increasing production costs. *Eur. J. Oper. Res.* 145(1):1–13.
- He Q, Ahmed S, Nemhauser GL (2015) Minimum concave cost flow over a grid network. *Math. Programming* 150(1):79–98.
- Horst R, Thoai NV (1998) An integer concave minimization approach for the minimum concave cost capacitated flow problem on networks. *Oper. Res. Spektrum* 20(1):47–53.
- Kelly DL, Khumawala BM (1982) Capacitated warehouse location with concave costs. *J. Oper. Res. Soc.* 33(9):817–826.
- Kim D, Pan X, Pardalos PM (2006) An enhanced dynamic slope scaling procedure with tabu scheme for fixed charge network flow problems. *Comput. Econom.* 27(2):273–293.



- Kubo M, Kasugai H (1991) A Lagrangean approach to the facility location problem with concave costs. *J. Oper. Res. Soc. Japan* 34(2):125–136.
- Lamar BW (1993) An improved branch and bound algorithm for minimum concave cost network flow problems. *J. Global Optim.* 3(3):261–287.
- Laporte G, Nickel S, da Gama FS (2019) *Location Science*, 2nd ed. (Springer, Cham, Switzerland).
- Larsson T, Migdalas A, Rönnqvist M (1994) A Lagrangean heuristic for the capacitated concave minimum cost network flow problem. *Eur. J. Oper. Res.* 78(1):116–129.
- Lee J, Leyffer S (2012) *Mixed Integer Nonlinear Programming* (Springer, New York).
- Lin JR, Nozick LK, Turnquist MA (2006) Strategic design of distribution systems with economies of scale in transportation. *Ann. Oper. Res.* 144(1):161–180.
- Liu ML, Sahinidis NV, Srethman JP (1996) Planning of chemical process networks via global concave minimization. *Global Optimization in Engineering Design* (Springer, Boston), 195–230.
- Lu D, Gzara F, Elhedhli S (2014) Facility location with economies and diseconomies of scale: Models and column generation heuristics. *IIIE Trans.* 46(6):585–600.
- Malik A (2021) Location problems in supply chain design: Concave costs, probabilistic service levels, and omnichannel distribution. PhD thesis, Concordia University, Montreal, Canada.
- Manousiouthakis VI, Thomas N, Justanieah AM (2011) On a finite branch and bound algorithm for the global minimization of a concave power law over a polytope. *J. Optim. Theory Appl.* 151(1):121–134.
- Melo M, Nickel S, da Gama FS (2009) Facility location and supply chain management: A review. *Eur. J. Oper. Res.* 196(2):401–412.
- Ortega F, Wolsey LA (2003) A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks* 41(3):143–158.
- Ortiz-Astorquiza C, Contreras I, Laporte G (2017) Formulations and approximation algorithms for multilevel uncapacitated facility location. *INFORMS J. Comput.* 29(4):767–779.
- Ortiz-Astorquiza C, Contreras I, Laporte G (2018) Multi-level facility location problems. *Eur. J. Oper. Res.* 267(3):791–805.
- Ortiz-Astorquiza C, Contreras I, Laporte G (2019) An exact algorithm for multi-level uncapacitated facility location. *Transportation Sci.* 53(4):1085–1106.
- Ryoo HS, Sahinidis NV (1996) A branch-and-reduce approach to global optimization. *J. Global Optim.* 8(2):107–138.
- Şahin G, Süral H (2007) A review of hierarchical facility location models. *Comput. Oper. Res.* 34(8):2310–2331.
- Saif A, Elhedhli S (2016) A Lagrangian heuristic for concave cost facility location problems: The plant location and technology acquisition problem. *Optim. Lett.* 10(5):1087–1100.
- Salman FS, Yücel E (2015) Emergency facility location under random network damage: Insights from the Istanbul case. *Comput. Oper. Res.* 62:266–281.
- Shen ZJM, Daskin MS (2005) Trade-offs between customer service and cost in integrated supply chain design. *Manufacturing Service Oper. Management* 7(3):188–207.
- Shen ZJM, Coullard C, Daskin MS (2003) A joint location-inventory model. *Transportation Sci.* 37(1):40–55.
- Soland RM (1974) Optimal facility location with concave costs. *Oper. Res.* 22(2):373–382.
- Vidyarthi N, Jayaswal S (2014) Efficient solution of a class of location-allocation problems with stochastic demand and congestion. *Comput. Oper. Res.* 48:20–30.
- Vidyarthi N, Çelebi E, Elhedhli S, Jewkes E (2007) Integrated production-inventory-distribution system design with risk pooling: Model formulation and heuristic solution. *Transportation Sci.* 41(3):392–408.
- Zangwill WI (1968) Minimum concave cost flows in certain networks. *Management Sci.* 14(7):429–450.