Computers & Operations Research 38 (2011) 1678–1695

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

The vendor location problem

Yüce Çınar, Hande Yaman *

Bilkent University, Department of Industrial Engineering, 06800 Ankara, Turkey

ARTICLE INFO

Available online 4 March 2011

Keywords:
Location
Vendor location problem
Hierarchical facility location
Valid inequalities
Computational complexity

ABSTRACT

The vendor location problem is the problem of locating a given number of vendors and determining the number of vehicles and the service zones necessary for each vendor to achieve at least a given profit. We consider two versions of the problem with different objectives: maximizing the total profit and maximizing the demand covered. The demand and profit generated by a demand point are functions of the distance to the vendor. We propose integer programming models for both versions of the vendor location problem. We then prove that both are strongly NP-hard and we derive several families of valid inequalities to strengthen our formulations. We report the outcomes of a computational study where we investigate the effect of valid inequalities in reducing the duality gaps and the solution times for the vendor location problem.

1. Introduction

With a major beverage company about to launch its own brand for demijohn water, we recently worked on the following discrete facility location problem.

Unlike drinks sold in regular bottles, demijohn water has the distinctive feature of making it hard for customers to switch brands; every brand has its own containers and customers pay for the first container, replacing it when empty with a full one. In this way, the customer then continues to only pay for the contents of the bottles; switching brands would mean they would have to pay for a full bottle again. Suppliers of bottled gas for cooking and heating purposes also benefit from this quasimonopoly once the customer has made her choice of brands.

Water sold in large containers is the rule rather than the exception in Turkey: in 2008, 80% of consumption was demijohn water and the remaining 20% was water bottled in smaller containers. And the market itself is large: about 8.5 billion liters per year according to the Association of Packaged Water Producers in Turkey (SUDER [27]) and still expected to grow (by 10% in 2009).

A recent marketing survey carried out by the beverage company shows that customers value the quality of the water (taste, hygiene, chemical composition, etc.) and the quality of the service the most. The quality of the service is strongly related to service times and the satisfaction is affected by the presence of competitors in the same region who could provide shorter service times.

The number of potential customers in a given region mainly depends on the distance to the assigned vendor and on the proximity of competitors. This explains why selling in many locations could increase the market share. This strategy, however, has a price: some vendors may not reach a given profit. The beverage company wanted to ensure that each vendor would earn enough money and that the company would maximize its market share.

Inspired by this real-life problem, we define the *vendor location problem* (*VLP*) as follows. We are given a set of demand points corresponding to population zones and a set of possible locations for vendors. Each vendor can only use a given number of vehicles. We also know the (fixed) cost of a vendor office (rent, insurance, salaries of employees at office, etc.) at a given location as well as the cost (including the salary of the driver) and capacity of a vehicle.

For a given demand point, there is a set of eligible vendors. Each demand point has a potential demand. The market share that our company can have depends on the travel times of its vendors and the proximity of competitors. The profit (sales revenue minus the transportation cost) therefore depends on the vendor that serves a demand point.

The *VLP* is the problem of locating a given number of vendors and assigning each demand point to at most one vehicle of an eligible vendor such that capacities of vehicles are not exceeded and each vendor achieves at least a determined profit. We consider two objective functions. In *ProfitVLP*, the aim is to maximize the total profit and in *CoverageVLP*, the aim is to maximize the coverage, i.e., the total demand served.

Our problem can be seen as a hierarchical facility location problem where demand points are in level 0, vehicles are level 1 facilities, and vendors are level 2 facilities. Sahin and Sural [28]

* Corresponding author. Tel.: +90 312 290 27 68; fax: +90 312 266 40 54.
E-mail addresses: yuce@bilkent.edu.tr (Y. Çınar), hyaman@bilkent.edu.tr (H. Yaman).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.02.011

review hierarchical facility location models and propose a classification scheme. The first attribute in this scheme is flow pattern. In a single flow pattern, the flow starts from level 0 and ends at the highest level by passing through all intermediate levels. In a multiple flow pattern, flows can travel from any lower level to any higher level. Our problem has a single flow pattern in the opposite direction. The second attribute is service varieties. Here in a nested system, a higher level facility provides all services provided by a lower level facility; in a non-nested system, facilities in different levels provide different services. Our system is a non-nested system. As the third attribute, the authors consider the spatial configuration. In a coherent system, all demand that is served by a given lower level facility is served by the same higher level facility. Since in our system, each vehicle belongs to a vendor, we have a coherent system. The final attribute is the objective. Here the authors consider the three common objectives: median, covering, and fixed charge. *ProfitVLP* can be considered as a median type problem even though we maximize profit rather than minimize cost. *CoverageVLP* is a maximum covering type problem.

Multi-level facility location problems have been previously studied by many researchers. Aardal et al. [2] propose some facet defining and valid inequalities for the polytope associated with the two level uncapacitated facility location problem. Approximation algorithms are studied by Aardal et al. [1], Ageev [3], Ageev et al. [4], Bumb [9], Bumb and Kern [10], Gabor and van Ommeren [14], Guha et al. [16], Meyerson et al. [22], Shmoys et al. [26], Zhang [32], and Zhang and Ye [33]. Branch and bound algorithms are given by Kaufman et al. [18], Ro and Tcha [25], Tcha and Lee [29], and Tragantalerngsak et al. [31]. Barros and Labbé [7] present various formulations, a Lagrangean relaxation, and a primal heuristic. Gao and Robinson [12,13] propose dual-based solution procedures. Chardaire et al. [11] present two formulations, valid inequalities, a Lagrangian relaxation, and a simulated annealing algorithm. Linear and Lagrangian relaxations are studied by Bloemhof-Ruwaard et al. [8], Marín [20], Marín and Pelegrín [21], Pirkul and Jayaraman [24], Tragantalerngsak et al. [30] for different versions of the problem.

A recent work that is closely related to ours is on the capacity and distance constrained plant location problem by Albareda-Sambola et al. [5]. In this problem, a set of possible locations is given. A facility may house a number of identical vehicles. Each demand point must be assigned to a single vehicle of a facility. There are capacity restrictions for facilities and restrictions on the total distance traveled for vehicles. The aim is to determine where to open facilities, to decide on the number of vehicles for each facility, and to assign the demand points to vehicles and facilities with the aim of minimizing the costs of opening facilities, using vehicles, and assigning demand points to facilities and vehicles. The authors provide different models and a tabu search algorithm for this problem. This study is similar to ours in that it is concerned with assigning demand points to facility vehicles. It is different from ours in that it has capacity constraints for facilities and restrictions on the total distance traveled for vehicles; we have capacity constraints for vehicles and minimum profit constraints for facilities.

In this paper, we introduce two new two-level facility location problems, namely *ProfitVLP* and *CoverageVLP*, which are motivated by a real life problem. Different from the classical facility location problems, here we have minimum profit constraints for open facilities and capacity constraints for their vehicles. We investigate the computational complexity of these problems and prove that they are strongly NP-hard. We propose integer programming formulations, valid inequalities, and extra constraints to be able to use the cutting planes of off-the-shelf integer programming solvers. We report the outcomes of a computational study where we use four types of instances that differ in their demand and profit functions. We investigate the effect of valid inequalities on linear programming relaxation bounds and solution times for these different types of instances. Finally, we analyze the optimal solutions of *ProfitVLP* and *CoverageVLP* and report how the differences in demand and profit functions effect the service regions for an example problem. Hence, the contributions of the paper are two new facility location problems motivated by a real life problem, resolution of the status of their computational complexity, and strong mixed integer programming formulations for these problems.

The paper is organized as follows. In Section 2, we present integer programming formulations for *ProfitVLP* and *CoverageVLP* and prove that both problems are strongly NP-hard. We propose some valid inequalities in Section 3. Computational results are given in Section 4. We analyze the solutions of *ProfitVLP* and *CoverageVLP* for two different types of instances in Section 5. In Section 6, we conclude the paper.

2. Formulations and complexity

In this section, we first introduce the notation and then present formulations for *ProfitVLP* and *CoverageVLP*. Then we prove that both *ProfitVLP* and *CoverageVLP* are strongly NP-hard.

Let I be the set of demand points and J be the set of possible locations for vendors. For a demand point $i \in I$, J_i is the set of vendors that can serve i. In our problem, we define J_i to be the set of vendors whose travel time to i does not exceed a given bound. We also define $I_j = \{i \in I : j \in J_i\}$ for $j \in J$.

We denote with f_j the fixed cost of the vendor office and with v_j the fixed cost of a vehicle for a vendor located at $j \in J$. We assume that these cost values are non-negative. We define ρ_{min} to be the minimum profit a vendor should achieve.

We denote with p the number of vendors to be located. The vendor at location $j \in J$ may have up to k_j^{max} vehicles. Let $K_j = \{1, \ldots, k_j^{max}\}$ for $j \in J$. The capacity of a vehicle is equal to γ.

Demand point $i \in I$ has demand q_{ij} and generates profit ρ_{ij} if it is served by the vendor at location $j \in J_i$. We assume that q_{ij}'s are positive and that ρ_{ij}'s are non-negative.

We define the following decision variables. For $i \in I$, $j \in J_i$, and $k \in K_j$, x_{ijk} is 1 if demand point i is assigned to vehicle k of vendor j and 0 otherwise, for $j \in J$, and $k \in K_j$, z_{jk} is 1 if vendor j uses vehicle k and 0 otherwise, and finally, for $j \in J$, y_j is 1 if a vendor is located at location j and 0 otherwise.

Using these variables, the *ProfitVLP* can be modeled as follows:

$$\max \quad \sum_{i \in I}\sum_{j \in J_i}\sum_{k \in K_j}\rho_{ij}x_{ijk} - \sum_{j \in J}\sum_{k \in K_j}v_j z_{jk} - \sum_{j \in J} f_j y_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in J_i}\sum_{k \in K_j} x_{ijk} \le 1 \quad \forall i \in I \tag{2}$$

$$\sum_{j \in J} y_j = p \tag{3}$$

$$\sum_{k \in K_j} x_{ijk} \le y_j \quad \forall i \in I, \ j \in J_i \tag{4}$$

$$\sum_{i \in I_j} q_{ij} x_{ijk} \le \gamma z_{jk} \quad \forall j \in J, \ k \in K_j \tag{5}$$

$$\sum_{i \in I_j}\rho_{ij}\sum_{k \in K_j} x_{ijk} \ge \sum_{k \in K_j} v_j z_{jk} + (\rho_{min} + f_j) y_j \quad \forall j \in J \tag{6}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in I, \ j \in J_i, \ k \in K_j \tag{7}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J, \ k \in K_j \tag{8}$$

$$y_j \in \{0,1\} \quad \forall j \in J \tag{9}$$

Constraints (2) ensure that a demand point is assigned to at most one vehicle of one eligible vendor. Constraint (3) states that the number of vendors to be located is p. If a vendor is not located at a given location, then a demand point cannot be served by any of its vehicles due to constraints (4). Constraints (5) are capacity constraints for vehicles. At the same time, they ensure that demand points are not assigned to vehicles that are not in use. Constraints (6) ensure that each vendor makes a profit of at least ρ_{min} units. Constraints (7)–(9) state that the variables are binary. Objective function (1) is the total profit of all vendors.

Note here that constraints $z_{jk} \le y_j$ for $j \in J$ and $k \in K_j$ are not included in the model. Let $j \in J$ and $k \in K_j$. If there exists $i \in I_j$ with $x_{ijk} = 1$, then constraints (4) force y_j to one and constraints (5) force z_{jk} to one. On the other hand, if $x_{ijk} = 0$ for all $i \in I_j$, then there exists an optimal solution with $z_{jk} = 0$ since v_j's are non-negative. Hence constraints $z_{jk} \le y_j$ for $j \in J$ and $k \in K_j$ are not necessary for the validity of the model. We do not include them in the model not to increase the number of constraints. Later, we use them as valid inequalities and test their performance.

The *CoverageVLP* can be modeled as follows:

$$\max \quad \sum_{i \in I} \sum_{j \in J_i} \sum_{k \in K_j} q_{ij} x_{ijk}$$
$$\text{s.t.} \quad (2)-(9) \tag{10}$$

Here the objective function (10) is the total demand served.

To conclude this section, we investigate the computational complexity of problems *ProfitVLP* and *CoverageVLP*.

Theorem 1. *ProfitVLP and CoverageVLP are strongly NP-hard.*

Proof. We prove that the decision versions of *ProfitVLP* and *CoverageVLP* are NP-complete in the strong sense by a reduction from the decision version of the bin packing problem.

Given a finite set of items U, a size $s_i \in \mathbb{Z}_+$ for each $i \in U$, a positive integer bin capacity B, and a positive integer κ, the decision version of the bin packing problem is defined as follows. Is there a partition of set U into $U_1, \dots, U_\kappa$ such that $\sum_{i \in U_u} s_i \le B$ for all $u = 1, \dots, \kappa$? This problem is NP-complete in the strong sense (see problem [SR1] in Garey and Johnson [15]).

First note that when $v_j = f_j = 0$ for all $j \in J$ and $\rho_{ij} = q_{ij}$ for all $i \in I$ and $j \in J_i$, problems *ProfitVLP* and *CoverageVLP* become the same problem. Hence in the remaining part of the proof, we only consider *CoverageVLP* with $v_j = f_j = 0$ for all $j \in J$ and $\rho_{ij} = q_{ij}$ for all $i \in I$ and $j \in J_i$.

We define the decision version of *CoverageVLP* as follows. Given the parameters of the problem and a positive constant Φ, does there exist a feasible solution with coverage at least Φ? This problem is in NP.

Given an instance of the bin packing problem, let J be a singleton, $I = I_1 = U$, $p = 1$, $v_1 = 0$, $f_1 = 0$, $\rho_{min} = 0$, $k_1^{max} = \kappa$, $\rho_{i1} = q_{i1} = s_i$ for $i \in I$, $\gamma = B$, and $\Phi = \sum_{i \in I} q_{i1}$. Now there exists a solution to the decision version of the bin packing problem if and only if there exists a solution to the decision version of *CoverageVLP*. Hence, the decision version of *CoverageVLP* is NP-complete in the strong sense. □

3. Valid inequalities

In this section, we propose some valid inequalities for both versions of the *VLP*.

Let F be the set of solutions that satisfy constraints (2)–(9). We use some substructures in the formulation to derive our valid inequalities. We also propose some redundant constraints to convert some structures in our problem into knapsack structures so that we can use the lifted cover inequalities of off-the-shelf integer programming solvers.

3.1. Lower bounds on the number of vehicles

Albareda-Sambola et al. [5] propose the optimality cuts $\sum_{k \in K_j} z_{jk} \ge y_j$ for $j \in J$. These inequalities imply that if a vendor is open then it should use at least one vehicle. In our problem, since we have minimum profit constraints, in some cases we can obtain tighter bounds on the number of vehicles to be used by a vendor. Note that the resulting inequalities are valid inequalities for our problem rather than optimality cuts.

For $j \in J$ and a positive integer m, consider the following problem:

$$\delta_j(m) = \max \quad \sum_{i \in I_j} \sum_{k=1}^{m} \rho_{ij} \alpha_{ik} - \sum_{k=1}^{m} v_j \beta_k - f_j \tag{11}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \alpha_{ik} \le 1 \quad \forall i \in I_j \tag{12}$$

$$\sum_{i \in I_j} q_{ij} \alpha_{ik} \le \gamma \beta_k \quad \forall k = 1, \dots, m \tag{13}$$

$$\alpha_{ik} \in \{0,1\} \quad \forall i \in I_j, \; k = 1, \dots, m \tag{14}$$

$$\beta_k \in \{0,1\} \quad \forall k = 1, \dots, m \tag{15}$$

Here, the variable β_k takes value 1 if vehicle $k = 1, \dots, m$ is used and takes value 0 otherwise, and the variable α_{ik} takes value 1 if demand point $i \in I_j$ is assigned to vehicle $k = 1, \dots, m$ and takes value 0 otherwise. Constraints (12) ensure that each demand point is assigned to at most one vehicle and constraints (13) ensure that the sum of demands of demand points assigned to a given vehicle does not exceed the capacity of the vehicle if the vehicle is in use and no demand points are assigned to this vehicle if it is not in use. The objective function is equal to the sum of profits of demand points that are assigned to some vehicle minus the sum of costs of using vehicles and the vendor office j.

This problem hence maximizes the total profit for vendor j if vendor j can use at most m vehicles. Let m_j be the smallest integer with $\delta_j(m_j) \ge \rho_{min}$. Then for vendor j to achieve a minimum level of profit of ρ_{min} units, it should have at least m_j vehicles. If m_j is a positive integer less than or equal to k_j^{max}, then the inequality $\sum_{k \in K_j} z_{jk} \ge m_j y_j$ is a valid inequality. If m_j does not exist or if $m_j > k_j^{max}$, then vendor j cannot be profitable. Hence we can set $y_j = 0$.

The above problem is a capacitated facility location problem with single sourcing, which is an NP-hard problem (see, e.g., Neebe and Rao [23], Barcelo and Casanovas [6], Klincewicz and Luss [19], and Holmberg et al. [17]). As a result, computing the $\delta_j(m)$ values may be quite time consuming, hence we propose a way of computing lower bounds on m_j values.

Proposition 1. *Let $j \in J$ and $\sigma_j = \max_{i \in I_j} \rho_{ij}/q_{ij}$. The inequality*

$$\sum_{k \in K_j} z_{jk} \ge \left\lceil \frac{\rho_{min} + f_j}{\sigma_j \gamma - v_j} \right\rceil y_j \tag{16}$$

is valid for F.

Proof. For $j \in J$, $\sigma_j q_{ij} \ge \rho_{ij}$ for all $i \in I_j$. Multiplying constraints (5) with σ_j and summing over $k \in K_j$ yields $\sum_{i \in I_j} \sigma_j q_{ij} \sum_{k \in K_j} x_{ijk} \le \sigma_j \gamma \sum_{k \in K_j} z_{jk}$. Since $\sigma_j q_{ij} \ge \rho_{ij}$ for all $i \in I_j$, this implies $\sum_{i \in I_j} \rho_{ij} \sum_{k \in K_j} x_{ijk} \le \sigma_j \gamma \sum_{k \in K_j} z_{jk}$. Now combining this with constraint (6), we obtain

$$\sigma_j \gamma \sum_{k \in K_j} z_{jk} \geq \sum_{i \in I_j} \rho_{ij} \sum_{k \in K_j} x_{ijk} \geq \sum_{k \in K_j} v_j z_{jk} + (\rho_{min} + f_j) y_j$$

which gives

$$(\sigma_j \gamma - v_j) \sum_{k \in K_j} z_{jk} \geq (\rho_{min} + f_j) y_j$$

This implies that if $y_j = 1$, i.e., if a vendor is located at location j, then $\sum_{k \in K_j} z_{jk} \geq (\rho_{min} + f_j)/(\sigma_j \gamma - v_j)$. Since the left hand side is integer in a feasible solution, we can round up the right hand side.

If $y_j = 0$, then (16) becomes redundant. Hence we can conclude that inequality (16) is valid for F. □

For $j \in J$, σ_j can be computed in $O(|I_j|)$ time.

3.2. Cover inequalities for vehicle capacity constraints

For $i \in I$, $j \in J_i$, and $k \in K_j$, inequality

$$x_{ijk} \leq z_{jk} \tag{17}$$

is a valid inequality for F. These inequalities are often dominated by cover inequalities that may be generated using the knapsack structure of the capacity constraints (5) for the vehicles. Cover inequalities that are valid for each of these knapsack constraints are also valid for F. Let $j \in J$, $k \in K_j$, and $C \subseteq I_j$ be such that $\sum_{i \in C} q_{ij} > \gamma$. Then the cover inequality $\sum_{i \in C} x_{ijk} \leq (|C|-1) z_{jk}$ is a valid inequality for F. These inequalities can be strengthened by lifting.

Most of the integer programming solvers recognize knapsack constraints and use lifted cover inequalities as cutting planes. So here we limit our attention to some lifted cover inequalities that are not many in number so that they can be added to the formulation before giving it to the solver.

For a given location $j \in J$, we first consider all demand points with demand larger than half of the capacity of a vehicle. Then we know that at most one of these points may be assigned to a given vehicle of vendor j. This leads to the following set of inequalities.

Proposition 2. *For $j \in J$ and $k \in K_j$, the lifted cover inequality*

$$\sum_{i \in I_j : q_{ij} > \gamma/2} x_{ijk} \leq z_{jk} \tag{18}$$

is valid for F.

Next, we generate lifted cover inequalities for each demand point $i \in I_j$ with demand not more than half the capacity.

Proposition 3. *Let $i \in I_j$ be such that $q_{ij} \leq \gamma/2$. Define $C_{ij} = \{l \in I_j : q_{ij} + q_{lj} > \gamma\}$. Then the lifted cover inequality*

$$x_{ijk} + \sum_{l \in C_{ij}} x_{ljk} \leq z_{jk} \tag{19}$$

is valid for F.

Proof. If $x_{ijk} = 1$, then as $q_{ij} + q_{lj} > \gamma$ for each $l \in C_{ij}$, none of these demand points can be served by the same vehicle. If $x_{ijk} = 0$, then as $q_{lj} + q_{mj} > \gamma$ for l and m in C_{ij}, we know that $\sum_{l \in C_{ij}} x_{ljk} \leq z_{jk}$. □

Notice that if C_{ij} is empty, then inequality (19) reduces to (17).

3.3. Cover inequalities for the minimum profit constraints

Finally, we propose to model the minimum profit constraints in a different way so that we can use the lifted cover cuts of off-the-shelf solvers. To this end, we complement sums of assignment variables and rewrite the minimum profit constraints as 0–1 knapsack constraints as follows.

Let $j \in J$. For $i \in I_j$, define the variable $\bar{x}_{ij} = 1 - \sum_{k \in K_j} x_{ijk}$. Notice that $\bar{x}_{ij}$ is a 0–1 variable. Now the minimum profit constraint (6) can be rewritten as

$$\sum_{i \in I_j} \rho_{ij} \geq \sum_{i \in I_j} \rho_{ij} \bar{x}_{ij} + \sum_{k \in K_j} v_j z_{jk} + (\rho_{min} + f_j) y_j \tag{20}$$

which is a 0–1 knapsack inequality.

Now based on this substructure, we can derive cover inequalities that are valid for F.

Proposition 4. *Let $j \in J$, $S_1 \subseteq I_j$, and $S_2 \subseteq K_j$ with $|S_2| v_j + (\rho_{min} + f_j) > \sum_{i \in I_j \setminus S_1} \rho_{ij}$. The inequality*

$$\sum_{k \in S_2} z_{jk} \leq \sum_{i \in S_1} \sum_{k \in K_j} x_{ijk} + (|S_2| - 1) y_j \tag{21}$$

is valid for F.

Proof. Let $j \in J$. Consider the knapsack inequality (20). Suppose that $y_j = 1$. Let $S_1 \subseteq I_j$ and $S_2 \subseteq K_j$. If $\sum_{i \in S_1} \rho_{ij} + |S_2| v_j + (\rho_{min} + f_j) > \sum_{i \in I_j} \rho_{ij}$, then the cover inequality $\sum_{i \in S_1} \bar{x}_{ij} + \sum_{k \in S_2} z_{jk} \leq |S_1| + |S_2| - 1$ is valid. We can rewrite this inequality as $\sum_{i \in S_1} (1 - \sum_{k \in K_j} x_{ijk}) + \sum_{k \in S_2} z_{jk} \leq |S_1| + |S_2| - 1$, which simplifies to $\sum_{k \in S_2} z_{jk} \leq \sum_{i \in S_1} \sum_{k \in K_j} x_{ijk} + |S_2| - 1$.

If $y_j = 0$, then $x_{ijk} = 0$ for all $i \in I_j$ and $k \in K_j$ and $z_{jk} = 0$ for all $k \in K_j$. Hence inequality (21) is valid for F. □

4. Computational results

In this section, we report the outcomes of our computational study. Here, we investigate for which sizes we can solve the formulations to optimality in reasonable times and the effect of valid inequalities on the quality of upper bounds of linear programming relaxations and the solution times.

4.1. The data set and models

We use the data from the demijohn water company. The data includes 84 demand points, their estimated demands, the distances, and cost parameters. The set of possible locations for the vendors is the same as the set of demand points. Moreover, there is the additional restriction that if a vendor is located at a given demand point, then the demand of this point should be served by itself. To handle this, we added the constraint

$$\sum_{k \in K_j} x_{jjk} = y_j \quad \forall j \in J \tag{22}$$

We can also use this information to break the symmetry. We impose that if a vendor is located at a demand point, then the point should use its vehicle indexed as its first vehicle by adding the constraints

$$x_{jj1} = z_{j1} \quad \forall j \in J \tag{23}$$

$$x_{jj1} = y_j \quad \forall j \in J \tag{24}$$

Let *PM*0 and *CM*0 be the models obtained by adding the above constraints to *ProfitVLP* and *CoverageVLP*, respectively. Let *PM*1 and *CM*1 be the models *PM*0 and *CM*0 strengthened with the valid inequalities (16), which provide lower bounds on the number of vehicles for each vendor.

The fact that if a vendor is located at a demand point, then the point should use its first vehicle can further be used to obtain stronger lifted cover inequalities for the first vehicles:

$$\sum_{i \in I_j \setminus \{j\} : q_{ij} + q_{jj} > \gamma} x_{ij1} = 0 \quad \forall j \in J \tag{25}$$

$$\sum_{i \in I_j \backslash \{j\}: q_{ij} > (\gamma - q_{jj})/2} x_{ij1} \le z_{j1} \quad \forall j \in J \tag{26}$$

$$x_{ij1} + \sum_{l \in I_j \backslash \{j\}: q_{ij} + q_{lj} > \gamma - q_{jj}} x_{lj1} \le z_{j1} \quad \forall j \in J,\ i \in I_j \backslash \{j\} : q_{ij} \le \frac{\gamma - q_{jj}}{2} \tag{27}$$

We add the above cover inequalities for the first vehicles and inequalities (18) and (19) for the remaining vehicles to models *PM*1 and *CM*1 and call the resulting models *PM*2 and *CM*2, respectively.

We remove constraints (6) from models *PM*2 and *CM*2 and add the following variables and constraints to obtain models *PM*3 and *CM*3:

$$\overline{x}_{ij} = 1 - \sum_{k \in K_j} x_{ijk} \quad \forall i \in I, j \in J_i \tag{28}$$

$$\sum_{i \in I_j} \rho_{ij} \overline{x}_{ij} + \sum_{k \in K_j} v_j z_{jk} + (\rho_{min} + f_j) y_j \le \sum_{i \in I_j} \rho_{ij} \quad \forall j \in J \tag{29}$$

$$\overline{x}_{ij} \in \{0,1\} \quad \forall i \in I,\ j \in J_i. \tag{30}$$

The aim is to enable the solver to see the knapsack structure in the minimum profit constraints so that it can generate cover inequalities as discussed in Section 3.3.

We add the simple valid inequalities

$$z_{jk} \le y_j \quad \forall j \in J,\ k \in K_j \tag{31}$$

to models *PM*3 and *CM*3 to obtain models *PM*4 and *CM*4.

Finally, analyzing the results of our computational study, we also decided to repeat our experiment with additional models for *ProfitVLP* and *CoverageVLP*. For *ProfitVLP*, we tested model *PM*5, which is obtained by removing the cover inequalities obtained using vehicle capacity constraints, i.e., inequalities (18), (19), (25)–(27), from model *PM*4. For *CoverageVLP*, model *CM*5 is obtained by adding only valid inequalities $z_{jk} \le y_j$ for all $j \in J$ and $k \in K_j$ to model *CM*0.

In Tables 1 and 2, we give the constraints of the different models for *ProfitVLP* and *CoverageVLP*, respectively.

To evaluate the performances of the models defined above, we used the following test set. We let $p \in \{4,6,8\}$, $k_j^{max} = k^{max} \in \{6,8,10\}$ for all $j \in J$, and $\rho_{min} \in \{50,100,150\}$.

For each value of p, k^{max}, and ρ_{min}, we have four problems with different demand and profit structures. In **A**-type problems, we take $q_{ij} = q_i$ and $\rho_{ij} = \rho_i$ for all $j \in J_i$ and $i \in I$. So in **A**-type instances, the demand and profit are independent of the distance between the demand point and its vendor. In **B**-type problems, we take $q_{ij} = q_i$ and $\rho_{ij} = c_{ij} q_i$ for all $j \in J_i$ and $i \in I$ where c_{ij} is the unit profit that vendor j gains if it serves demand point i and is a function of the distance between i and j. In **C**-type problems, we take q_{ij} to be a function of the distance between i and j and $\rho_{ij} = c q_{ij}$ for all $j \in J_i$ and $i \in I$ where c is the unit profit and does not depend on distances. In this case, we let $q_{ij} = q_i$ for vendors j that are within a short traveling time of i and then let q_{ij} decrease with the distance between i and j for other eligible vendors. Precisely, for $i \in I$, we let $J_i = \{j \in J : d_{ij} \le 10\}$, where d_{ij} is the distance between the demand point i and the vendor j. For $i \in I$ and $j \in J_i$, we let $q_{ij} = q_i \min\{1, (1.5 - 0.1 d_{ij})\}$. Hence the demand generated by point i is equal to q_i if the vendor j is within 5 km of point i and is equal to $q_i(1.5 - 0.1 d_{ij})$ if j is farther. Finally, in **D**-type problems, we take both the demands and the profits as functions of the distances.

Both problems *ProfitVLP* and *CoverageVLP* are infeasible for $\rho = 150$, $p = 8$, and all four demand and profit structures. These instances are removed from the results.

All models are solved using GAMS 22.5 and CPLEX 11.0.0 on an AMD Opteron 252 processor (2.6 GHz) with 2 GB of RAM operating under the system CentOS (Linux version 2.6.9-42.0.3.ELsmp). We have a time limit of 1 h.

4.2. Results for *ProfitVLP*

In Tables 3–6, we report the results for *ProfitVLP* and the four types of instances, **A**, **B**, **C**, and **D**, respectively. For each instance and model, we report the percentage gap between the upper bound obtained by solving the linear programming relaxation of the corresponding model and the best lower bound for the integer problem in the column *LP gap*. Then we report the cpu times in seconds. If the problem is not solved to optimality in 1 h, then we report the remaining percentage gap in parentheses. Finally, we report the number of nodes in the branch and cut tree for each model and instance. The best results are marked bold.

Each table has a summary, where we can see the averages of linear programming relaxation gaps, final optimality gaps, cpu times, number of nodes, the number of instances solved to optimality with each model, and the number of times each model was among the best for the considered criterion.

In these tables we observe that the initial model *PM*0 has huge duality gaps and adding the valid inequalities (16), which impose

Table 1
Constraints of the models for *ProfitVLP*.

*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5
(2)–(9)	(2)–(9)	(2)–(9)	(2)–(5), (7)–(9)	(2)–(5), (7)–(9)	(2)–(5), (7)–(9)
(22)–(24)	(22)–(24)	(22)–(24)	(22)–(24)	(22)–(24)	(22)–(24)
	(16)	(16)	(16)	(16)	(16)
		(18), (19), (25)–(27)	(18), (19), (25)–(27)	(18), (19), (25)–(27)	
			(28)–(30)	(28)–(30)	(28)–(30)
				(31)	(31)

Table 2
Constraints of the models for *CoverageVLP*.

*CM*0	*CM*1	*CM*2	*CM*3	*CM*4	*CM*5
(2)–(9)	(2)–(9)	(2)–(9)	(2)–(5), (7)–(9)	(2)–(5), (7)–(9)	(2)–(9)
(22)–(24)	(22)–(24)	(22)–(24)	(22)–(24)	(22)–(24)	(22)–(24)
	(16)	(16)	(16)	(16)	
		(18), (19), (25)–(27)	(18), (19), (25)–(27)	(18), (19), (25)–(27)	
			(28)–(30)	(28)–(30)	
				(31)	(31)

Table 3
Results for *ProfitVLP* and **A**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5
50	6	4	76.77	76.77	29.21	29.21	**6.48**	**6.48**	397.61	203.00	284.25	454.59	261.57	**177.23**	29387	13208	14983	28200	13612	**8503**
50	8	4	48.05	48.05	8.56	8.56	**5.89**	25.82	411.81	**127.10**	562.07	2874.28	866.48	1002.29	20792	**7760**	26693	103257	23868	43420
50	10	4	42.51	42.51	**3.92**	**3.92**	**3.92**	42.31	244.58	153.81	32.81	12.25	**9.46**	43.52	27254	17360	3119	248	**85**	780
50	6	6	58.36	58.36	9.26	9.26	**5.04**	21.57	116.99	**110.11**	123.83	196.02	166.54	110.79	3847	**2575**	4269	4660	3846	2894
50	8	6	50.91	50.91	4.80	4.80	**4.15**	50.91	1180.66	568.44	**60.53**	112.02	91.72	653.58	49637	20099	**929**	1621	1683	28500
50	10	6	49.93	49.93	**4.12**	**4.12**	**4.12**	49.93	287.11	772.12	149.12	**84.97**	215.28	407.10	14542	25647	1890	**1496**	2853	10654
50	6	8	55.24	55.24	2.01	2.01	**1.91**	55.24	128.86	143.02	**105.59**	161.81	130.32	285.60	2767	2380	**1437**	2094	1617	3476
50	8	8	55.09	55.09	**1.93**	**1.93**	**1.93**	55.09	262.26	280.51	286.32	**218.60**	318.31	471.21	5330	3551	4569	**2248**	3228	6962
50	10	8	55.09	55.09	**1.93**	**1.93**	**1.93**	55.09	2062.02	996.55	1013.52	871.82	**838.39**	1068.35	57867	20087	19222	17513	**8925**	21266
100	6	4	76.77	76.77	29.21	29.21	**6.48**	**6.48**	**99.38**	224.41	214.90	226.33	174.20	161.32	**6639**	13122	10132	12692	9235	7816
100	8	4	48.05	48.05	8.56	8.56	**5.89**	25.82	907.42	**330.63**	1712.14	718.57	630.69	859.88	58284	**13090**	96939	39225	26042	39542
100	10	4	42.51	42.51	**3.92**	**3.92**	**3.92**	42.31	93.30	67.80	20.82	10.61	**7.48**	30.33	7063	3413	504	274	**64**	533
100	6	6	58.36	58.36	9.26	9.26	**5.04**	21.57	**85.56**	168.24	108.66	113.17	248.37	121.41	**1947**	7306	3870	2791	5043	2483
100	8	6	50.91	50.91	4.77	4.77	**4.12**	50.91	565.66	593.62	**73.19**	80.21	117.88	312.07	18574	24299	1034	**1032**	1598	12511
100	10	6	49.93	49.93	**4.09**	**4.09**	**4.09**	49.93	326.92	292.73	176.99	**100.61**	139.74	477.56	9777	11121	4441	**1315**	1655	14364
100	6	8	55.24	55.24	1.84	1.84	**1.82**	55.24	270.22	**159.86**	330.99	310.32	408.94	285.72	4837	**2495**	4595	3669	4189	2963
100	8	8	55.24	55.24	**1.84**	**1.84**	**1.84**	55.24	3259.19	1315.94	608.58	**460.38**	467.88	472.78	66262	30370	10986	**3431**	4171	4406
100	10	8	55.24	55.24	**1.84**	**1.84**	**1.84**	55.24	(0.05)	(0.05)	(0.05)	**741.72**	897.38	742.00	54635	50175	47481	**4803**	5475	5494
150	6	4	76.77	76.77	29.21	29.21	**6.48**	**6.48**	427.28	205.37	340.14	**142.27**	166.03	132.58	34745	10641	23289	6411	7191	**5580**
150	8	4	48.05	48.05	8.56	8.56	**5.89**	25.82	282.47	883.12	976.47	961.01	**164.50**	398.14	20508	46832	54511	38617	**5468**	22293
150	10	4	42.51	42.51	**3.92**	**3.92**	**3.92**	42.31	27.77	25.76	19.65	**9.84**	36.53	52.64	738	690	638	**160**	555	661
150	6	6	61.23	61.23	11.02	11.02	**6.85**	23.77	242.70	256.54	141.90	**97.67**	125.62	120.08	11248	6592	4629	**1580**	1757	3557
150	8	6	55.77	55.77	7.06	7.06	**6.76**	55.77	366.69	249.83	619.59	**52.91**	139.67	108.36	10633	11690	23780	652	666	**198**
150	10	6	54.73	54.73	**6.35**	**6.35**	**6.35**	54.73	979.08	2620.63	(0.04)	**202.62**	279.85	163.29	35910	64156	115684	**877**	928	981
Average			55.14	55.14	8.22	8.22	**4.45**	38.92	691.24	597.88	631.76	383.94	**287.62**	360.74	23051	17027	19984	11604	**5573**	10411
Avg. opt. gap (%)									**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**						
# of solved ins. (/24)									23	23	22	**24**	**24**	**24**						
# of best solutions (/24)					10	10	**24**	3	2	4	3	**10**	4	1	2	4	2	**9**	4	3

Table 4
Results for *ProfitVLP* and **B**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5
50	6	4	77.61	77.61	29.38	29.38	**6.93**	7.78	**118.79**	243.27	403.13	435.34	394.87	157.66	**6510**	14967	26960	26043	23355	6967
50	8	4	48.76	48.76	8.72	8.72	**6.04**	26.82	1004.05	2726.00	858.60	664.81	**140.27**	483.59	73179	103608	66030	33633	**3897**	16495
50	10	4	43.51	43.51	**4.32**	**4.32**	**4.32**	43.31	80.74	185.76	**10.28**	19.22	20.70	54.16	4358	10526	**278**	396	368	690
50	6	6	59.28	59.28	9.42	9.42	**5.33**	22.89	378.75	173.11	**108.48**	302.37	206.38	204.76	17782	8094	**4562**	7770	5924	10605
50	8	6	52.04	52.04	5.12	5.12	**4.55**	52.02	368.67	119.70	149.56	**71.49**	147.88	165.26	14233	4161	5163	**1678**	2169	5203
50	10	6	51.17	51.17	**4.53**	**4.53**	**4.53**	51.17	1969.78	350.27	**118.53**	144.66	203.60	1920.13	82051	13989	**2111**	2356	4122	51737
50	6	8	57.31	57.31	2.38	2.38	**2.21**	57.23	277.26	308.33	**101.84**	172.87	178.26	213.94	7455	5374	**1530**	2400	1923	2837
50	8	8	56.75	56.75	2.03	2.03	**2.02**	56.74	245.46	285.58	**147.91**	314.23	417.90	306.81	4151	4682	**2970**	4010	4293	4268
50	10	8	56.75	56.75	**2.04**	**2.04**	**2.04**	56.75	765.83	680.23	**428.27**	563.48	1128.88	847.33	16686	13179	**5266**	6304	9672	9794
100	6	4	77.61	77.61	29.38	29.38	**6.93**	7.78	625.21	392.02	**168.70**	187.98	204.20	698.42	49140	29850	10168	8666	14426	**6520**
100	8	4	48.76	48.76	8.72	8.72	**6.04**	26.82	371.98	836.81	356.90	782.17	568.03	**340.19**	25341	37881	18964	34894	22507	**14522**
100	10	4	43.51	43.51	**4.32**	**4.32**	**4.32**	43.31	148.09	44.02	**8.64**	17.50	39.84	68.94	10875	1288	**307**	348	576	1074
100	6	6	59.28	59.28	9.41	9.41	**5.33**	22.89	201.71	256.55	218.15	239.38	428.11	**180.65**	10180	10857	8823	9104	18007	**6085**
100	8	6	52.10	52.10	5.11	5.11	**4.53**	52.08	626.96	380.98	398.19	**75.75**	167.60	246.32	49338	28355	40180	**1286**	3042	10095
100	10	6	51.23	51.23	**4.52**	**4.52**	**4.52**	51.23	539.51	338.08	222.25	**148.77**	189.52	755.32	28173	22321	9924	**2428**	3587	25125
100	6	8	58.33	58.32	2.81	2.81	**2.70**	58.25	861.99	1562.33	1284.50	661.40	506.51	**447.86**	30967	35979	24283	6785	**4620**	6645
100	8	8	57.53	57.53	2.29	2.29	**2.27**	57.53	(0.21)	(0.03)	2493.71	**743.06**	775.63	1250.77	80039	112763	56433	**7183**	7549	14744
100	10	8	57.31	57.31	**2.15**	**2.15**	**2.15**	57.31	(0.06)	(0.01)	(0.05)	1384.88	1296.78	**1069.77**	70320	87082	101959	9137	**9011**	11935
150	6	4	77.61	77.61	29.38	29.38	**6.93**	7.78	1263.19	**107.18**	154.69	217.05	214.49	156.04	76293	6924	7789	11128	12527	**6577**
150	8	4	48.76	48.76	8.72	8.72	**6.04**	26.82	1382.57	2057.22	869.85	508.54	795.61	**407.60**	86130	120951	42405	31613	32705	**19499**
150	10	4	43.51	43.51	**4.32**	**4.32**	**4.32**	43.31	147.35	32.47	22.35	**11.43**	18.68	77.87	13428	550	685	**214**	339	1622
150	6	6	61.61	61.61	10.30	10.30	**6.71**	24.69	544.47	132.41	**117.53**	120.22	167.07	94.94	19639	4502	3275	**2975**	4778	3757
150	8	6	56.82	56.82	6.78	6.78	**6.50**	56.80	(0.01)	125.11	1006.17	**88.80**	124.49	152.24	113245	901	53107	664	**660**	1605
150	10	6	55.87	55.87	**6.15**	**6.15**	**6.15**	55.87	111.98	(0.00)	167.07	138.64	**53.78**	137.25	2345	89894	3173	743	**573**	764
Average			56.38	56.38	8.43	8.43	**4.73**	40.30	951.44	922.41	558.98	**333.91**	349.55	434.91	37161	32028	20681	8823	**7943**	9965
Avg. opt. gap (%)									(0.01)	**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**						
# of solved ins. (/24)									21	21	23	**24**	**24**	**24**						
# of best solutions (/24)					8	8	**24**		1	1	**9**	6	2	5	1		**7**	6	5	5

Table 5
Results for *ProfitVLP* and **C**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5
50	6	4	36.40	36.40	34.53	34.53	**3.99**	**3.99**	415.66	545.82	908.56	808.90	235.64	**137.35**	21173	29775	25878	16029	**4582**	5389
50	8	4	13.23	13.23	11.75	11.75	**11.27**	12.69	(0.20)	(0.77)	(1.46)	**1823.53**	(1.39)	(0.39)	263317	210493	**33642**	55561	113821	195620
50	10	4	13.27	13.27	**11.80**	**11.80**	**11.80**	13.27	(1.31)	(2.43)	(2.19)	(3.42)	(1.38)	**(1.16)**	153821	125472	**46926**	64855	58345	115305
50	6	6	21.78	21.78	17.42	17.42	**14.10**	14.80	(0.06)	(0.02)	(1.82)	(0.23)	**2466.03**	(0.35)	89914	81063	**22019**	25754	25452	66734
50	8	6	12.88	12.88	**9.04**	**9.04**	**9.04**	12.88	**1029.34**	(0.86)	(1.15)	(0.16)	(0.02)	(0.68)	33616	52149	**25086**	65312	41496	57322
50	10	6	12.88	12.88	**9.04**	**9.04**	**9.04**	12.88	(0.88)	(0.81)	(0.71)	(0.86)	(0.81)	**(0.58)**	60059	55828	24680	31578	**22146**	67819
50	6	8	18.37	18.37	11.74	11.74	**10.90**	17.56	(1.72)	(1.47)	(1.30)	(4.04)	(5.18)	**(0.68)**	29311	25123	18448	**17679**	24301	57322
50	8	8	16.49	16.49	**9.96**	**9.96**	**9.96**	16.49	(0.63)	(0.74)	(1.19)	(1.10)	(1.01)	**(0.55)**	22195	21204	15233	14515	**14275**	21764
50	10	8	16.53	16.53	**10.01**	**10.01**	**10.01**	16.53	**(1.05)**	(1.26)	(1.74)	(1.96)	(1.76)	(1.18)	21175	21012	12847	**11431**	13802	18843
100	6	4	36.40	36.40	34.53	34.53	**3.99**	**3.99**	901.64	1212.46	690.56	1312.17	227.35	**157.72**	44662	52312	21072	23443	**3208**	5496
100	8	4	13.23	13.23	11.75	11.75	**11.27**	12.69	**3191.66**	(0.81)	(0.67)	(0.75)	(0.58)	(0.39)	119696	182370	183491	72687	**67202**	217651
100	10	4	13.27	13.27	**11.80**	**11.80**	**11.80**	13.27	(3.44)	(2.10)	(2.07)	(2.41)	(2.58)	**(1.37)**	94767	153907	61780	52502	**42564**	116855
100	6	6	21.78	21.78	17.41	17.41	**14.10**	14.80	(0.58)	(0.02)	(0.06)	(0.41)	2271.14	**1773.18**	111839	75878	32600	28226	27477	**24158**
100	8	6	12.88	12.88	**8.95**	**8.95**	**8.95**	12.88	(0.74)	(0.47)	(0.55)	**(0.06)**	(0.52)	(0.48)	69973	73212	43094	68349	**41428**	85315
100	10	6	12.88	12.88	**8.95**	**8.95**	**8.95**	12.88	(0.66)	**(0.59)**	(0.96)	(0.76)	(0.78)	(0.81)	87953	57383	**25728**	26137	46347	49519
100	6	8	20.07	20.07	13.44	13.44	**12.61**	19.38	(1.68)	(1.32)	**(0.64)**	(1.49)	(2.55)	1.97	29552	23039	17985	**14123**	18713	22312
100	8	8	16.49	16.49	**9.95**	**9.95**	**9.95**	16.49	(0.53)	**(0.30)**	(2.43)	(0.81)	(1.58)	(0.91)	23977	27750	13911	**11026**	11332	15121
100	10	8	16.53	16.53	**10.00**	**10.00**	**10.00**	16.53	(2.09)	(1.31)	(1.82)	(2.36)	**(1.17)**	(2.99)	20528	18319	9930	11658	**9690**	17731
150	6	4	36.40	36.40	34.53	34.53	**3.99**	**3.99**	350.66	1091.24	783.26	3397.84	213.28	**126.14**	17131	61934	24382	106412	**3816**	4747
150	8	4	13.23	13.23	11.75	11.75	**11.27**	12.69	**2749.68**	(0.73)	(0.95)	3295.62	3170.46	(0.57)	245267	114423	**85248**	91799	118125	189367
150	10	4	13.27	13.27	**11.80**	**11.80**	**11.80**	13.27	**(1.16)**	(2.20)	(2.20)	(2.19)	(1.25)	**(1.16)**	149194	132028	52135	82438	**50298**	153399
150	6	6	21.78	21.69	17.20	17.20	**14.10**	14.80	1725.05	(0.02)	2271.70	(0.39)	(0.52)	**1430.80**	46143	67638	**31197**	26376	32231	17171
150	8	6	14.34	14.34	**10.18**	**10.18**	**10.18**	14.34	(1.34)	(1.34)	(1.18)	**(0.99)**	(1.55)	(1.14)	80373	82554	52963	44136	**42435**	77732
150	10	6	14.34	14.34	**10.18**	**10.18**	**10.18**	14.34	(1.86)	(1.77)	(1.77)	(1.77)	**(1.41)**	(1.49)	48787	58336	**30143**	34224	53502	67901
Average			18.28	18.28	14.49	14.49	**10.14**	13.23	**2982.99**	3268.79	3194.00	3293.33	3057.75	3001.11	78518	75133	37934	41510	**36941**	69608
Avg. opt. gap (%)									(0.83)	(0.89)	(1.49)	(1.09)	(1.09)	**(0.78)**						
# of solved ins. (/24)									**7**	3	4	5	6	5						
# of best solutions (/24)					13	13	**24**	3	5	2	1	3	3	**11**			8	4	**11**	1

Table 6
Results for *ProfitVLP* and **D**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5	*PM*0	*PM*1	*PM*2	*PM*3	*PM*4	*PM*5
50	6	4	36.72	36.72	34.79	34.79	**4.31**	4.77	561.12	488.34	633.79	1956.61	232.02	**160.37**	28765	25766	15534	34652	**5139**	7266
50	8	4	13.44	13.44	11.93	11.93	**11.47**	12.93	1164.72	**455.10**	(0.28)	(1.16)	(1.09)	2809.47	67315	**28114**	94733	100649	56289	170833
50	10	4	13.48	13.48	**11.98**	**11.98**	**11.98**	13.48	(3.12)	(2.15)	(2.67)	(2.30)	**(1.64)**	(2.26)	100368	110347	68070	**28127**	66041	111921
50	6	6	21.92	21.92	17.43	17.43	**13.98**	14.99	(0.01)	(0.01)	(0.60)	(0.70)	**1757.17**	2004.23	84878	98031	62948	22232	**14977**	26490
50	8	6	13.27	13.27	**9.32**	**9.32**	**9.32**	13.27	(1.58)	(0.74)	(0.73)	(0.54)	(0.71)	**(0.09)**	81713	56679	**29972**	30236	45316	72111
50	10	6	13.27	13.27	**9.32**	**9.32**	**9.32**	13.27	(1.04)	(0.75)	(0.86)	(0.79)	**(0.62)**	(0.82)	52052	47068	19260	**18872**	40327	57108
50	6	8	20.76	20.76	13.77	13.77	**12.87**	19.87	(3.12)	**(2.14)**	(2.17)	(2.63)	(4.98)	(2.56)	32632	29856	21022	**16450**	26431	29340
50	8	8	17.01	17.01	**10.25**	**10.25**	**10.25**	17.01	(1.10)	**(0.90)**	(1.36)	(1.71)	(1.25)	(1.53)	29339	18631	13130	15488	**12540**	22387
50	10	8	17.03	17.03	**10.27**	**10.27**	**10.27**	17.03	**(1.33)**	(1.62)	(1.69)	(1.61)	(2.39)	(3.12)	23767	20408	**11139**	11722	13368	20784
100	6	4	36.72	36.72	34.79	34.79	**4.31**	4.77	3386.70	507.64	1106.28	1203.65	698.44	**164.79**	121859	20229	30277	20402	13817	**7294**
100	8	4	13.44	13.44	11.93	11.93	**11.47**	12.93	**1712.80**	(1.62)	3393.36	(0.21)	3577.48	2669.71	143447	187751	115416	**91689**	133343	154873
100	10	4	13.48	13.48	**11.98**	**11.98**	**11.98**	13.48	(2.29)	(2.14)	(3.88)	(1.46)	(2.42)	**(0.82)**	150628	127016	61970	70603	**47248**	164907
100	6	6	21.92	21.92	17.42	17.42	**13.98**	14.99	(0.58)	(0.01)	(0.01)	(0.58)	1718.09	**1618.76**	90554	77487	56953	24518	**14288**	17998
100	8	6	13.27	13.27	**9.21**	**9.21**	**9.21**	13.27	(0.71)	(0.09)	(0.32)	**3097.56**	(0.91)	(0.64)	72710	67244	**35361**	40838	80211	71418
100	10	6	13.27	13.27	**9.21**	**9.21**	**9.21**	13.27	(0.93)	(1.19)	(0.81)	**(0.73)**	(0.81)	**(0.73)**	48044	44638	31362	31477	**21707**	52541
100	6	8	20.60	20.60	13.74	13.74	**12.86**	19.86	(1.25)	(0.47)	**(0.04)**	(2.09)	(2.89)	(1.35)	28266	29342	28226	**15084**	18162	19170
100	8	8	17.01	17.01	**10.24**	**10.24**	**10.24**	17.01	**(0.60)**	(0.78)	(0.76)	(2.54)	(1.35)	(1.71)	25717	19918	13032	**10990**	14204	20374
100	10	8	17.01	17.01	**10.24**	**10.24**	**10.24**	17.01	**(1.10)**	(1.22)	(1.29)	(2.77)	(3.31)	(1.22)	22651	17197	11800	**10205**	11563	18160
150	6	4	36.72	36.72	34.79	34.79	**4.31**	4.77	419.46	290.98	801.11	1717.41	224.96	**178.18**	23992	14481	38333	27987	**4510**	7150
150	8	4	13.44	13.44	11.93	11.93	**11.47**	12.93	2384.95	(0.62)	(1.68)	**932.03**	(0.53)	1703.53	118142	116417	99826	**32216**	110576	113171
150	10	4	13.48	13.48	**11.98**	**11.98**	**11.98**	13.48	(2.10)	(2.74)	(2.99)	**(1.81)**	(2.42)	(2.40)	118407	194520	**49374**	75712	64753	86667
150	6	6	22.45	22.45	17.82	17.82	**14.60**	15.60	(0.72)	(0.01)	(0.00)	(2.17)	3056.64	**1750.43**	59335	83396	46790	**19144**	25628	22813
150	8	6	14.78	14.78	**10.49**	**10.49**	**10.49**	14.78	(1.40)	(1.33)	(1.48)	(1.35)	**(0.83)**	(1.41)	80001	83067	53054	**39108**	43934	53446
150	10	6	14.78	14.78	**10.49**	**10.49**	**10.49**	14.78	(1.86)	(1.61)	(1.45)	**(1.44)**	(1.53)	(1.71)	48006	61615	**24584**	26854	36213	61914
Average			18.72	18.72	14.81	14.81	**10.44**	13.73	3101.29	3072.64	3247.35	3221.22	3019.44	**2794.21**	68858	65801	43007	**33965**	38358	57922
Avg. opt. gap (%)									(1.04)	**(0.92)**	(1.05)	(1.19)	(1.24)	(0.93)						
# of solved ins. (/24)									6	4	4	5	7	**9**						
# of best solutions (/24)					13	13	**24**		4	3	1	5	4	**8**		1	5	**10**	7	1

lower bounds on the number of vehicles, has almost no impact on these gaps. The average gaps are 55.14%, 56.38%, 18.28%, and 18.72% for **A**, **B**, **C**, and **D** instances, respectively. Here we remark that even though they are still very large, the instances of types **C** and **D** (instances where demands depend on the distances) have much smaller gaps compared to the instances of types **A** and **B**. This may be due to the fact that capacity constraints for vehicles are tighter for instances of type **A** and **B**. Indeed, the average gaps for the model with cover inequalities, *PM*2, are 8.22%, 8.43%, 14.49%, and 14.81% for **A**, **B**, **C**, and **D**, respectively. Here we see that these inequalities have reduced the gaps considerably for **A**- and **B**-type instances whereas their effect was much smaller for **C** and **D**-type instances. After all the valid inequalities are added, with model *PM*4, the average gaps are 4.45%, 4.73 %, 10.14%, and 10.44% for **A**, **B**, **C**, and **D**, respectively. Here we see that the valid inequalities are more effective in improving the quality of linear programming upper bounds for **A**- and **B**-type instances.

Except for **D**-type instances, model *PM*4 gives the smallest number of nodes on the average. If we compare the average number of nodes for the original model *PM*0 and the ones for model *PM*4, we observe that the reductions are 75.82%, 78.63%, 52.95%, and 44.29% for **A**, **B**, **C**, and **D** instances, respectively. We can conclude that our valid inequalities are more effective in reducing the size of the branch and cut tree for **A**- and **B**-type instances.

For **A**-type instances, models *PM*1 and *PM*2 could solve 23 instances, model *PM*3 22 instances, and models *PM*4, *PM*5, and *PM*6 24 instances to optimality in 1 h. The remaining gaps are quite small for the unsolved instances. The best average cpu time is given by model *PM*4 and is 58.39% less than the average cpu time of the original model *PM*0. Model *PM*4 has given the best cpu for only four instances, whereas model *PM*3 has given the best cpu for 10 instances out of 24. This model has the best average cpu time for **B**-type instances. It is interesting to note that for these instances, model *PM*2 has given the best cpu time for nine instances and model *PM*3 has given the best cpu time for six instances. But one of the instances could not be solved to optimality with model *PM*2. Models *PM*0 and *PM*1 could not solve three instances to optimality. The average cpu time of *PM*3 is 64.90% less than the average cpu time of *PM*0. For these instances, we see that both cover inequalities based on the vehicle capacities and the knapsack inequalities for minimum profit constraints are quite effective in reducing the cpu times on the average.

ProfitVLP is harder for instances of types **C** and **D**, where the demands are functions of distances. Here our valid inequalities are not useful in reducing cpu times and final gaps for unsolved instances. We see that models *PM*0 and *PM*5 are the best in terms of cpu times and the number of instances solved to optimality, for **C** and **D** instances, respectively. The largest final gap for **C** instances is 1.41%, and for **D** instances it is 2.14%.

4.3. Results for CoverageVLP

We report the results for *CoverageVLP* and the four types of instances, **A**, **B**, **C**, and **D** in Tables 7–10, respectively. *CoverageVLP* turned out to be easier to solve compared to *ProfitVLP* for our instances. First of all, the duality gaps were smaller for the original formulation. The average gaps are 16.62%, 16.71%, 5.02%, and 5.03% for **A**, **B**, **C**, and **D** instances, respectively. Again, the instances of types **C** and **D** have smaller gaps. Our valid inequalities reduced the average duality gaps to 1.22%, 1.26%, 1.06%, and 1.09% for **A**, **B**, **C**, and **D** instances, respectively. Even though it looks like the reduction in the duality gaps is mostly due to the use of valid inequalities (31), the differences between the average gaps of models *CM*4 and *CM*5 show that some of the remaining valid inequalities are also effective in strengthening the original model for **A**- and **B**-type instances.

In terms of number of nodes, *CM*4, the model with all valid inequalities, has given the best average results, decreasing the number of nodes by 78.77%, 75.51%, 72.79%, and 86.52% compared to *CM*0 for **A**, **B**, **C**, and **D** instances, respectively.

Only model *CM*4 could solve all 24 type **A** instances to optimality in 1 h of cpu time. Its average cpu is 78.14% less than the average cpu of the original model *CM*0. Similar results are obtained for **B**-type instances. For both types of instances, *CM*4 performs much better than all other models in terms of average cpu times.

All our models solve the 24 **C**-type instances to optimality within the time limit. Among these, *CM*5 has the best average cpu time. Our model with all valid inequalities has an average cpu time of 96.23 s, whereas model *CM*5 has an average cpu time of 68.41 s. Hence for these instances, we can conclude that even though the valid inequalities are effective in reducing the duality gaps and the sizes of the branch and cut trees, other than the simple inequalities $z_{jk} \le y_j$ for all $j \in J$ and $k \in K_j$, they are not very useful in reducing the cpu times.

Finally, for **D**-type instances, the model *CM*4 gives the best average cpu time, which is 73.04% less than the average cpu time for the original model. It is interesting to note that for these instances, model *CM*5 could not solve two problems to optimality.

4.4. Improvements in linear programming bounds

Here, we report the percentage improvement in linear programming bounds obtained by adding families of valid inequalities. We first solve the linear programming relaxation of the model without any valid inequalities. Then we add each family of valid inequalities separately to the original model. We use the inequalities (16), which impose lower bounds on the number of vehicles, cover inequalities (18), (19), (25)–(27), and the simple valid inequalities (31). We compute the percentage improvements in the linear programming bounds. The averages are reported in Table 11.

Here we observe that the inequalities (16), which impose lower bounds on the number of vehicles, do not improve the linear programming bounds. The cover inequalities result in significant improvements for *ProfitVLP*, especially for **A**- and **B**-type instances. However, they are not as useful for *CoverageVLP*. The valid inequalities (31) improve the linear programming bounds for all problems, more for **A**- and **B**-type instances and less for **C**- and **D**-type instances.

4.5. Comparison of profit and coverage values

In Table 12, we report the best profit and coverage values for all ranges of parameters considered in our experiment. Here, we observe that for a given ρ_{min} value, best profit and coverage values are achieved with medium or large k^{max} and p values. We depict the profit values for **A**-type instances in Fig. 1. Similar behavior is observed for the other types of instances.

For *CoverageVLP*, the best coverage values are achieved with $p=8$ and $k^{max}=8,10$ for $\rho_{min}=50,100$ and with $p=6$ and $k^{max}=10$ for $\rho_{min}=150$. Increasing p and k^{max} has a significant effect on the best coverage values.

5. Analysis of example optimal solutions

In this section, we analyze the optimal solutions for problems *ProfitVLP* and *CoverageVLP* for an example instance with

Table 7
Results for *CoverageVLP* and **A**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*
50	6	4	54.24	53.24	53.83	53.83	**1.91**	**1.91**	261.90	175.16	374.14	246.76	146.60	**113.41**	19701	13054	25542	11277	8610	**6198**
50	8	4	23.05	23.05	22.71	22.71	**1.58**	7.26	255.51	811.10	801.58	260.04	**51.14**	203.96	17862	53197	71304	15729	**2597**	8852
50	10	4	5.77	5.77	5.66	5.66	**0.62**	5.64	17.32	10.26	14.79	24.50	15.63	**8.96**	863	811	745	654	715	**629**
50	6	6	30.68	30.68	30.34	30.34	**1.31**	5.78	85.34	61.67	89.38	88.92	79.11	**53.68**	4634	2807	3692	2097	1910	**1710**
50	8	6	9.27	9.27	9.27	9.27	**1.15**	9.27	71.74	49.29	**39.93**	99.65	68.32	42.57	1907	843	685	1333	**570**	752
50	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	6.21	3.54	1.74	2.69	1.71	**0.61**	211	20	**0**	**0**	**0**	**0**
50	6	8	9.21	9.21	9.21	9.21	**0.44**	9.21	1329.72	1996.49	(0.29)	(0.30)	**106.77**	148.13	233085	32144	40052	31589	**625**	3298
50	8	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**3.79**	16.66	4.00	13.34	12.14	33.74	**0**	250	**0**	8	10	380
50	10	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	5.16	13.68	5.78	6.94	5.64	**3.02**	80	150	**0**	**0**	**0**	134
100	6	4	54.24	54.24	53.70	53.70	**1.91**	**1.91**	219.06	209.80	183.54	339.86	131.53	**88.21**	16737	15631	9254	13453	6647	**2866**
100	8	4	23.05	23.05	22.69	22.69	**1.58**	7.26	571.61	696.36	1687.35	543.48	**78.93**	888.37	32944	29950	66906	35824	**6046**	49649
100	10	4	5.77	5.77	5.66	5.66	**0.62**	5.64	22.05	19.96	7.91	15.94	**7.65**	16.25	1326	945	**257**	690	544	1141
100	6	6	30.68	30.68	30.34	30.34	**1.31**	5.78	**44.16**	63.87	75.78	112.94	97.16	52.00	1473	2951	2647	2710	2328	**1069**
100	8	6	9.44	9.44	9.44	9.44	**1.32**	9.44	58.18	67.67	87.82	214.85	**32.86**	50.73	1426	2435	830	4594	**425**	1163
100	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	8.88	4.77	2.19	3.38	1.66	**1.28**	381	67	**0**	**0**	**0**	**0**
100	6	8	9.78	9.78	9.78	9.78	**0.96**	9.78	3412.02	(0.63)	(0.57)	(0.57)	**952.09**	3014.50	29703	30739	30454	27678	**8222**	35885
100	8	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	(0.88)	220.31	**25.54**	142.29	199.87	(0.33)	12777	1615	**40**	557	800	14182
100	10	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	131.76	625.47	73.83	**20.53**	22.89	202.88	1516	7030	519	103	**30**	2267
150	6	4	54.24	54.24	53.62	53.62	**1.91**	**1.91**	105.72	189.55	262.52	242.02	137.16	**92.46**	8900	15466	18429	8928	6324	**5142**
150	8	4	23.05	23.05	22.69	22.69	**1.58**	7.26	947.77	**88.44**	238.58	1361.99	113.91	235.52	44811	**5313**	11125	40893	6276	9365
150	10	4	5.77	5.77	5.66	5.66	**0.62**	5.64	19.76	23.46	20.99	**8.33**	13.13	13.19	1085	1275	968	**576**	611	800
150	6	6	31.32	31.32	29.10	29.10	**1.78**	6.30	387.94	96.44	108.43	**87.50**	130.49	146.91	21164	2363	3592	1955	2730	**1854**
150	8	6	12.38	12.38	10.35	10.35	**3.69**	12.38	765.14	(0.34)	759.93	198.72	**171.76**	505.64	24278	78776	18036	963	**809**	18144
150	10	6	6.99	6.99	**5.08**	**5.08**	**5.08**	6.99	188.60	916.46	189.82	**135.33**	158.56	278.16	3086	18242	642	**480**	491	4892
Average			16.62	16.62	16.21	16.21	**1.22**	4.97	521.64	565.15	510.66	473.76	**114.03**	408.10	11248	13170	12738	8420	**2388**	7099
Avg. opt. gap (%)									(0.04)	(0.04)	(0.04)	(0.04)	**(0.00)**	(0.01)						
# of solved ins. (/24)									23	22	22	22	**24**	23						
# of best solutions (/24)			6	6	7	7	**24**	9	2	1	2	4	7	**8**	1	1	6	5	**10**	9

Table 8
Results for *CoverageVLP* and **B**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*
50	6	4	54.24	54.24	53.75	53.75	**1.91**	**1.91**	193.65	249.32	443.58	229.15	182.90	**93.13**	16136	19605	26851	9866	5938	**5315**
50	8	4	23.05	23.05	22.69	22.69	**1.58**	7.26	217.40	250.88	1371.16	326.72	**100.75**	653.08	15880	24977	58418	19454	**6358**	28674
50	10	4	5.77	5.77	5.66	5.66	**0.62**	5.64	18.51	16.73	14.12	8.75	19.83	**5.49**	1125	898	658	622	968	**498**
50	6	6	30.68	30.68	30.34	30.34	**1.31**	5.78	85.60	**70.00**	108.38	91.01	85.15	92.55	5087	3068	4460	2661	**2254**	3648
50	8	6	9.27	9.27	9.27	9.27	**1.15**	9.27	46.20	58.52	70.34	126.16	**35.54**	43.66	1618	962	666	920	**410**	733
50	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	2.78	3.43	1.96	1.58	**1.49**	1.62	20	49	**0**	**0**	**0**	**0**
50	6	8	9.21	9.21	9.21	9.21	**0.44**	9.21	993.46	2832.36	(0.31)	997.20	106.86	**85.75**	13939	34035	38492	10581	974	**874**
50	8	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	71.84	73.22	**9.42**	26.15	17.15	23.76	845	786	20	50	**10**	400
50	10	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	10.22	6.66	**4.26**	5.42	5.18	6.82	220	59	**0**	**0**	**0**	190
100	6	4	54.24	54.24	53.65	53.65	**1.91**	**1.91**	193.65	351.52	270.66	303.63	138.82	**58.47**	16136	24076	15971	10857	7955	**2824**
100	8	4	23.05	23.05	22.69	22.69	**1.58**	7.26	217.40	2141.64	300.77	1272.10	**77.90**	446.48	15880	129486	23257	73610	**5558**	24236
100	10	4	5.77	5.77	5.66	5.66	**0.62**	5.64	17.30	22.74	11.46	20.61	15.67	**11.45**	978	1065	714	**653**	968	902
100	6	6	30.68	30.68	30.28	30.28	**1.31**	5.78	67.19	**63.94**	113.23	106.41	76.37	65.47	3114	2682	6621	2508	**1761**	2360
100	8	6	9.44	9.44	9.44	9.44	**1.32**	9.44	108.21	79.64	145.11	173.98	**52.91**	100.91	3842	2313	1113	1836	**521**	4664
100	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	11.27	14.70	2.25	2.51	**1.48**	5.31	418	492	**0**	**0**	**0**	89
100	6	8	9.78	9.78	9.78	9.78	**0.96**	9.78	(0.44)	(0.46)	(0.47)	(0.36)	2115.30	**2009.85**	29528	29173	31932	21391	**15394**	18291
100	8	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	(0.88)	1016.32	2016.47	686.79	**481.46**	1789.27	20365	11097	6476	1574	**1477**	10274
100	10	8	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	725.10	343.28	42.00	**11.03**	58.59	(1.14)	7686	3190	100	20	**19**	15575
150	6	4	54.24	54.24	53.62	53.62	**1.91**	**1.91**	113.95	233.72	227.44	259.32	95.73	**73.11**	7836	18333	12214	10286	**3105**	4149
150	8	4	23.05	23.05	22.69	22.69	**1.58**	7.26	1028.98	598.60	160.18	232.00	71.69	**18.17**	68400	32616	8133	13838	**4084**	5525
150	10	4	5.77	5.77	5.66	5.66	**0.62**	5.64	23.13	23.08	9.60	10.04	**8.53**	18.17	1200	1301	491	**466**	645	776
150	6	6	31.32	31.32	28.23	28.23	**1.76**	6.30	100.34	**52.67**	162.01	62.76	64.42	96.12	7090	1542	7745	1524	**1489**	2055
150	8	6	13.14	13.14	10.29	10.29	**4.12**	13.14	310.53	312.13	879.47	195.80	**107.40**	450.92	9465	3899	14650	903	**457**	4974
150	10	6	8.34	8.34	5.63	5.63	**5.55**	8.34	545.08	269.56	348.89	**157.98**	653.42	208.04	5125	2907	773	**438**	1353	2306
Average			16.71	16.71	16.19	16.19	**1.26**	5.06	512.58	528.53	579.71	371.14	**190.60**	420.15	10497	14525	10823	7669	**2571**	5806
Avg. opt. gap (%)									(0.06)	(0.02)	(0.03)	(0.02)	**(0.00)**	(0.05)						
# of solved ins. (/24)									22	23	22	23	**24**	23						
# of best solutions (/24)			6	6	6	6	**24**	9		3	2	2	**9**	8			3	6	**16**	4

Table 9
Results for *CoverageVLP* and **C**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*	*CM0*	*CM1*	*CM2*	*CM3*	*CM4*	*CM5*
50	6	4	25.73	25.73	25.71	25.71	**1.19**	**1.19**	432.22	900.92	3165.87	1843.91	**150.42**	158.94	25535	48016	165456	43576	**5481**	7935
50	8	4	2.12	2.12	2.12	2.12	**1.73**	**1.73**	17.82	32.34	34.28	45.52	**22.19**	43.85	3541	8041	5304	5359	**3325**	7035
50	10	4	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.45**	0.50	0.81	0.94	0.91	0.46	**0**	**0**	**0**	**0**	**0**	**0**
50	6	6	9.33	9.33	9.32	9.32	**4.20**	4.40	786.90	485.58	725.56	1299.77	533.40	**261.94**	43195	15868	14914	19261	5390	**4445**
50	8	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.48	0.56	0.93	0.88	0.96	**0.46**	**0**	**0**	**0**	**0**	**0**	**0**
50	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.53**	0.67	0.90	0.96	0.73	0.67	**0**	**0**	**0**	**0**	**0**	**0**
50	6	8	3.07	3.07	3.01	3.01	**1.34**	2.54	**56.66**	62.70	96.86	144.78	82.68	105.36	3140	3299	2386	2468	**1164**	5045
50	8	8	0.04	0.04	**0.02**	**0.02**	**0.02**	0.04	2.06	1.45	**0.64**	0.79	0.82	1.38	163	40	**0**	**0**	**0**	20
50	10	8	0.04	0.04	**0.02**	**0.02**	**0.02**	0.04	5.09	4.43	**0.78**	1.02	1.00	6.75	456	418	**0**	**0**	**0**	486
100	6	4	25.73	25.73	25.71	25.71	**1.19**	**1.19**	342.98	379.13	477.49	1277.67	**91.70**	140.39	20271	19062	17330	30092	**2723**	7229
100	8	4	2.12	2.12	2.12	2.12	**1.73**	**1.73**	85.01	25.16	**14.24**	29.73	15.64	92.02	11960	3464	**1240**	4053	1927	27160
100	10	4	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	1.19	0.92	0.87	0.87	1.22	**0.56**	**0**	9	**0**	**0**	**0**	**0**
100	6	6	9.33	9.33	9.32	9.32	**4.20**	4.40	284.11	541.96	388.66	1703.63	475.63	**237.37**	10200	13656	10011	23266	**4724**	4902
100	8	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.42**	1.56	1.02	1.02	0.81	0.53	**0**	57	**0**	**0**	**0**	**0**
100	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	4.08	**0.60**	1.26	1.08	1.21	2.44	330	**0**	**0**	**0**	**0**	39
100	6	8	3.56	3.56	3.13	3.13	**1.76**	3.09	196.36	273.46	191.69	180.66	192.90	**170.01**	8225	17602	4506	2313	**1987**	8702
100	8	8	0.88	0.88	0.40	0.40	**0.38**	0.88	11.23	11.54	10.08	6.99	**1.41**	12.65	482	581	43	28	**0**	474
100	10	8	0.88	0.88	**0.40**	**0.40**	**0.40**	0.88	13.58	18.82	25.41	17.09	**4.97**	14.22	533	709	415	170	**12**	525
150	6	4	25.73	25.73	25.71	25.71	**1.19**	**1.19**	367.43	383.30	1142.58	2294.78	172.68	**91.51**	17763	18110	61517	64122	5660	**4664**
150	8	4	2.12	2.12	2.12	2.12	**1.73**	**1.73**	73.07	38.82	12.76	61.09	79.50	**11.40**	7779	4756	1314	6003	9639	**1095**
150	10	4	**0.06**	**0.06**	**0.06**	**0.06**	**0.06**	**0.06**	**1.48**	5.31	3.39	4.54	2.66	7.63	528	667	972	558	**230**	1002
150	6	6	9.33	9.33	8.99	8.99	**4.20**	4.40	572.66	417.78	585.98	1927.39	473.40	**267.09**	15987	10033	10880	19465	**4232**	6283
150	8	6	0.13	0.13	**0.06**	**0.06**	**0.06**	0.13	4.86	2.34	3.61	1.16	**1.06**	8.42	283	64	20	**0**	**0**	474
150	10	6	0.13	0.13	**0.06**	**0.06**	**0.06**	0.13	5.03	6.50	**1.32**	1.36	1.56	5.78	457	402	**0**	**0**	**0**	332
Average			5.02	5.02	4.93	4.93	**1.06**	1.24	135.70	149.85	286.96	451.98	96.23	**68.41**	7118	6869	12346	9197	**1937**	3660
Avg. opt. gap (%)									**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**	**(0.00)**						
# of solved ins. (/24)									**24**	**24**	**24**	**24**	**24**	**24**						
# of best solutions (/24)			7	7	12	12	**24**	13	5	1	4		6	**8**	5	3	9	10	**20**	**8**

Table 10
Results for *CoverageVLP* and **D**-type instances.

Parameters			LP gap (%)						Cpu time (s)/optimality gap (%)						Number of nodes					
ρ_{min}	k^{max}	p	*CM*0	*CM*1	*CM*2	*CM*3	*CM*4	*CM*5	*CM*0	*CM*1	*CM*2	*CM*3	*CM*4	*CM*5	*CM*0	*CM*1	*CM*2	*CM*3	*CM*4	*CM*5
50	6	4	25.73	25.73	25.71	25.71	**1.19**	**1.19**	353.54	609.85	598.64	758.12	259.94	**92.84**	17552	34889	17870	20915	7610	**4460**
50	8	4	2.12	2.12	2.12	2.12	**1.73**	**1.73**	47.49	19.55	35.59	**11.86**	15.30	47.08	7082	2734	4079	1497	**1449**	5086
50	10	4	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.41**	0.42	0.97	0.92	0.99	0.52	**0**	**0**	**0**	**0**	**0**	**0**
50	6	6	9.33	9.33	9.32	9.32	**4.20**	4.40	565.19	**349.39**	510.66	2235.29	490.20	669.14	15258	10412	18273	36960	**5097**	8462
50	8	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	1.16	**0.51**	0.90	1.09	0.91	1.30	70	**0**	**0**	**0**	**0**	**0**
50	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	2.04	1.88	**0.98**	1.11	1.20	1.25	69	51	**0**	**0**	**0**	39
50	6	8	3.20	3.20	3.12	3.12	**1.46**	2.67	48.02	**36.32**	113.23	113.57	104.06	119.09	3000	1902	3122	1808	**1366**	6319
50	8	8	0.04	0.04	**0.02**	**0.02**	**0.02**	0.04	2.39	2.92	1.06	0.86	**0.70**	3.69	156	270	**0**	**0**	**0**	342
50	10	8	0.04	0.04	**0.02**	**0.02**	**0.02**	0.04	4.06	3.19	0.86	0.86	**0.80**	5.31	363	255	**0**	**0**	**0**	400
100	6	4	25.73	25.73	25.71	25.71	**1.19**	**1.19**	334.94	430.85	779.60	2111.85	243.26	**110.99**	15638	23082	27070	47622	7012	**4881**
100	8	4	2.12	2.12	2.12	2.12	**1.73**	**1.73**	22.64	36.73	**17.40**	38.84	107.88	45.60	6179	5577	**1087**	4738	11227	6980
100	10	4	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.56	**0.48**	0.87	0.86	0.88	0.54	**0**	**0**	**0**	**0**	**0**	**0**
100	6	6	9.33	9.33	9.32	9.32	**4.20**	4.40	269.34	637.17	633.62	1841.72	468.53	**186.58**	9873	14758	10579	26504	5138	**3494**
100	8	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	3.58	1.58	**1.17**	1.60	1.19	2.98	365	46	**0**	**0**	**0**	474
100	10	6	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	5.05	5.59	5.35	**1.24**	1.32	2.94	486	500	474	**0**	**0**	61
100	6	8	3.68	3.68	2.99	2.99	**1.82**	**1.82**	420.18	321.45	**206.31**	415.81	311.55	395.33	35763	16979	8808	4794	**3219**	18690
100	8	8	1.21	1.21	**0.64**	**0.64**	**0.64**	0.68	(0.23)	(0.05)	(0.28)	**41.16**	86.22	(0.31)	104629	69792	94567	**515**	952	97678
100	10	8	1.21	1.21	**0.64**	**0.64**	**0.64**	1.21	(0.32)	(0.31)	(0.29)	123.07	**85.88**	(0.32)	117575	65763	60112	1633	**1493**	118411
150	6	4	25.73	25.73	25.71	25.71	**1.19**	**1.19**	1801.76	345.84	447.40	1900.28	187.28	**105.06**	94078	17714	18728	54512	**4178**	5010
150	8	4	2.12	2.12	2.12	2.12	**1.73**	**1.73**	14.85	60.46	40.80	**13.44**	39.14	70.16	**1580**	5394	6144	1782	4801	8268
150	10	4	**0.06**	**0.06**	**0.06**	**0.06**	**0.06**	**0.06**	4.72	6.56	5.40	**2.53**	3.87	6.35	827	559	569	**161**	254	760
150	6	6	9.33	9.33	8.98	8.98	**4.20**	4.40	431.83	363.12	985.60	1289.40	697.38	**187.72**	10803	8963	23610	14116	5708	**3302**
150	8	6	0.13	0.13	**0.04**	**0.04**	**0.04**	0.13	2.34	1.72	2.53	**0.92**	1.04	7.82	141	50	20	**0**	**0**	240
150	10	6	0.13	0.13	**0.04**	**0.04**	**0.04**	0.13	**0.72**	1.60	1.34	1.46	1.37	0.81	**0**	23	**0**	**0**	**0**	**0**
Average			5.03	5.03	4.94	4.94	**1.09**	1.20	480.71	434.89	482.94	454.49	**129.62**	385.98	18395	11655	12296	9065	**2479**	12223
Avg. opt. gap (%)									(0.02)	(0.01)	(0.02)	**(0.00)**	**(0.00)**	**(0.03)**						
# of solved ins. (/24)									22	22	22	**24**	**24**	22						
# of best solutions (/24)			7	7	13	13	**24**	14	2	4	4	**6**	3	5	4	3	8	12	**15**	7

$\rho_{min} = 100$, $k_j^{max} = 8$, and $p = 6$ for **A**- and **D**-types. The solutions are depicted in Figs. 2–5. In all these figures, the locations of vendors are denoted by rectangles and their service regions are marked by different colors. Demand points that are not served by any of the vendors are not colored. The areas that are not population zones are not numbered.

Table 11
Improvements in linear programming bounds.

Type	*ProfitVLP*			*Coverage VLP*		
	(16)	(18), (19), (25)–(27)	(31)	(16)	(18), (19), (25)–(27)	(31)
A	0	30.23	9.77	0	0.33	8.24
B	0	30.65	9.64	0	0.43	8.24
C	0	3.22	3.80	0	0.09	3.09
D	0	3.31	3.75	0	0.10	3.09

An optimal solution for *ProfitVLP* for an **A**-type instance is given in Fig. 2. Here we can see that as the demands and profits of demand points are independent of the distances to the vendors, the service regions are quite dispersed. For instance, demand point 14, which is assigned to the vendor at location 7, is surrounded by three other demand points that are all served by the vendor at location 15. Similarly, demand point 57 is served by the vendor at location 65 even though there is another vendor at a neighboring location.

We see an optimal solution for *ProfitVLP* for a **D**-type instance in Fig. 3. Here we observe that the service regions of vendors are rather compact and the vendors are located more centrally in their regions.

Optimal solutions for *CoverageVLP* for **A**- and **D**-type instances are given in Figs. 4 and 5, respectively. We see a similar pattern here, i.e., the service regions are more compact in the solution for the **D**-type instance.

Table 12
Best profit and coverage values.

Parameters			*ProfitVLP*				*Coverage VLP*			
ρ_{min}	k^{max}	p	**A**-type	**B**-type	**C**-type	**D**-type	**A**-type	**B**-type	**C**-type	**D**-type
50	6	4	832.8	806.1	852.8	828.8	1413	1413	1423	1423
50	8	4	1006.8	974.2	1033.8	1005.0	1790	1790	1761	1761
50	10	4	1053.3	1016.9	1033.8	1005.0	2095	2095	1799	1799
50	6	6	1094.2	1057.7	1158.7	1128.0	2042	2042	2069	2069
50	8	6	1152.7	1113.2	1250.7	1214.3	2450	2450	2263	2263
50	10	6	1160.2	1119.6	1250.7	1214.3	2677	2677	2263	2263
50	6	8	1070.6	1031.5	1250.7	1197.9	2508	2508	2435	2432
50	8	8	1071.6	1035.3	1271.1	1236.4	2739	2739	2509	2509
50	10	8	1071.6	1035.3	1270.6	1236.2	2739	2739	2509	2509
100	6	4	832.8	806.1	852.8	828.8	1413	1413	1423	1423
100	8	4	1006.8	974.2	1033.8	1005.0	1790	1790	1761	1761
100	10	4	1053.3	1016.9	1033.8	1005.0	2095	2095	1799	1799
100	6	6	1094.2	1057.7	1158.7	1128.0	2042	2042	2069	2069
100	8	6	1152.7	1112.8	1250.7	1214.3	2446	2446	2263	2263
100	10	6	1160.2	1119.2	1250.7	1214.3	2677	2677	2263	2263
100	6	8	1070.6	1024.8	1231.6	1197.9	2495	2495	2422	2420
100	8	8	1070.6	1030.2	1271.1	1236.4	2739	2739	2488	2480
100	10	8	1070.6	1031.6	1270.6	1236.4	2739	2739	2488	2480
150	6	4	832.8	806.1	852.8	828.8	1413	1413	1423	1423
150	8	4	1006.8	974.2	1033.8	1005.0	1790	1790	1761	1761
150	10	4	1053.3	1016.9	1033.8	1005.0	2095	2095	1798	1798
150	6	6	1074.7	1042.4	1158.7	1122.0	2032	2032	2069	2069
150	8	6	1116.7	1079.3	1234.7	1198.3	2382	2366	2260	2260
150	10	6	1124.2	1085.9	1234.7	1198.3	2502	2471	2260	2260

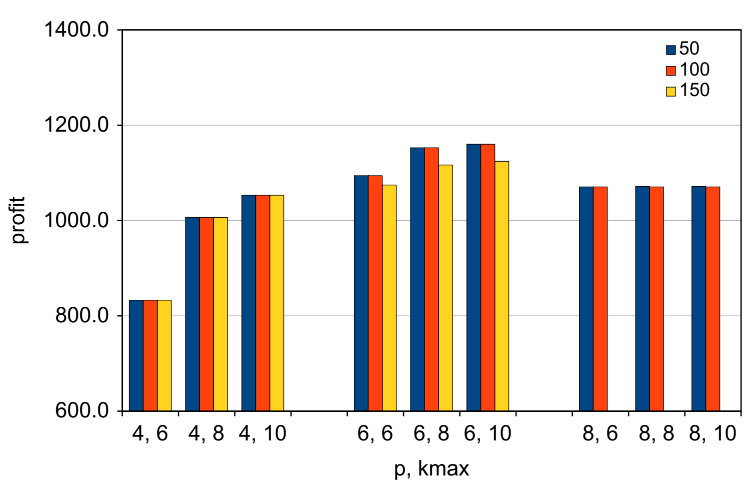

Fig. 1. Best profit values for **A**-type instances.

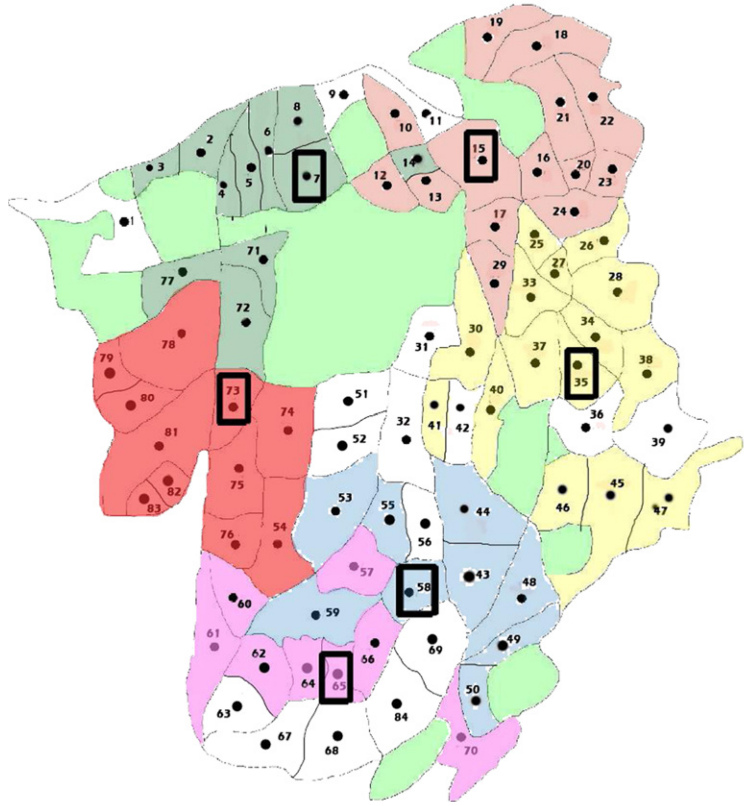

Fig. 2. Optimal solution of *ProfitVLP* for an **A**-type instance.

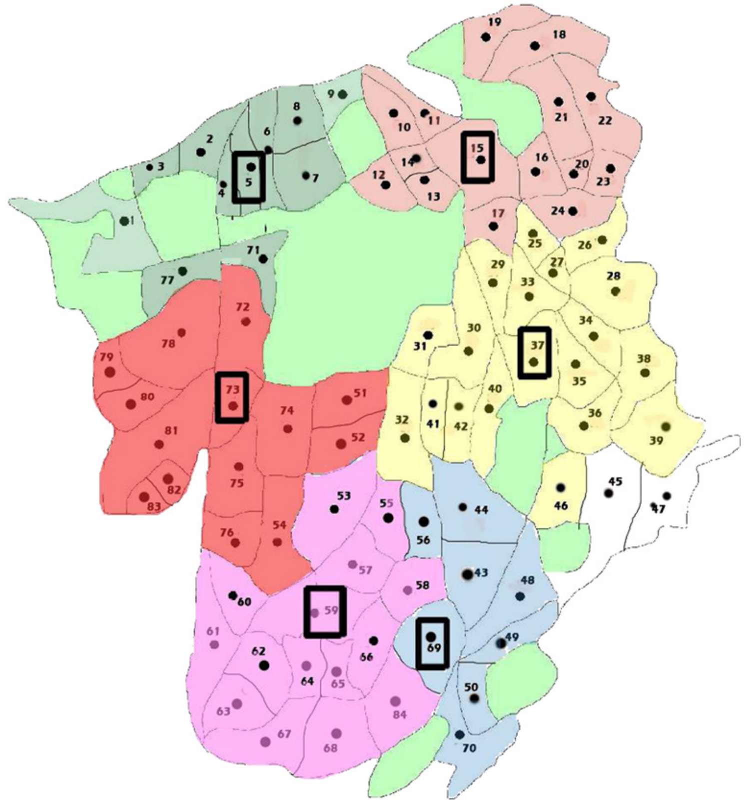

Fig. 3. Optimal solution of *ProfitVLP* for a **D**-type instance.

In summary, comparing these solutions, we see that demand points assigned to the same vendor lie around the vendor node for both *ProfitVLP* and *CoverageVLP* type **D** problems, whereas some demand points serviced from the same vendor are separated from the group in *ProfitVLP* and *CoverageVLP* for type **A** problems. This is expected since in **A**-type problems, profits and demands do not

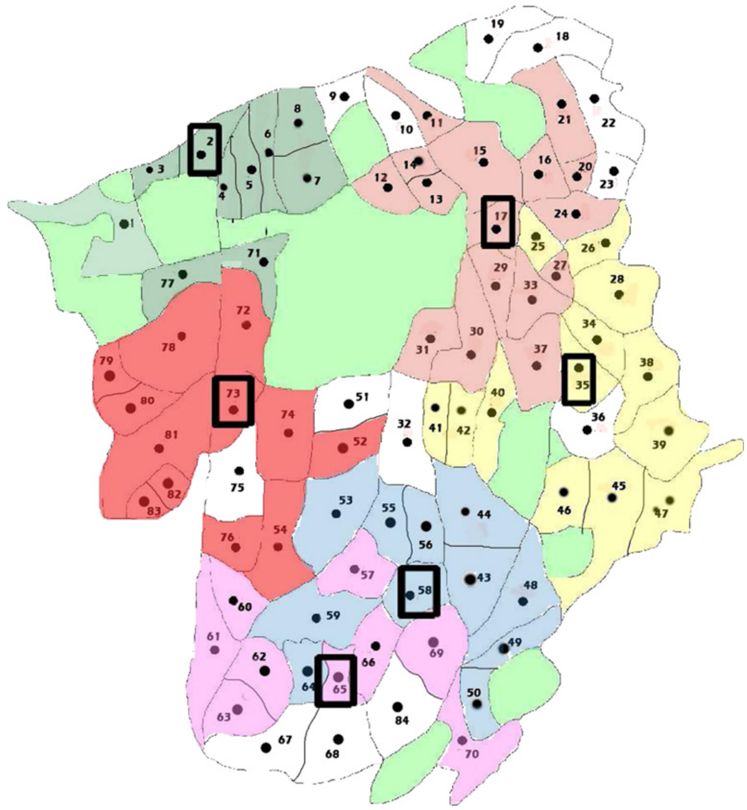

Fig. 4. Optimal solution of *CoverageVLP* for an **A**-type instance.

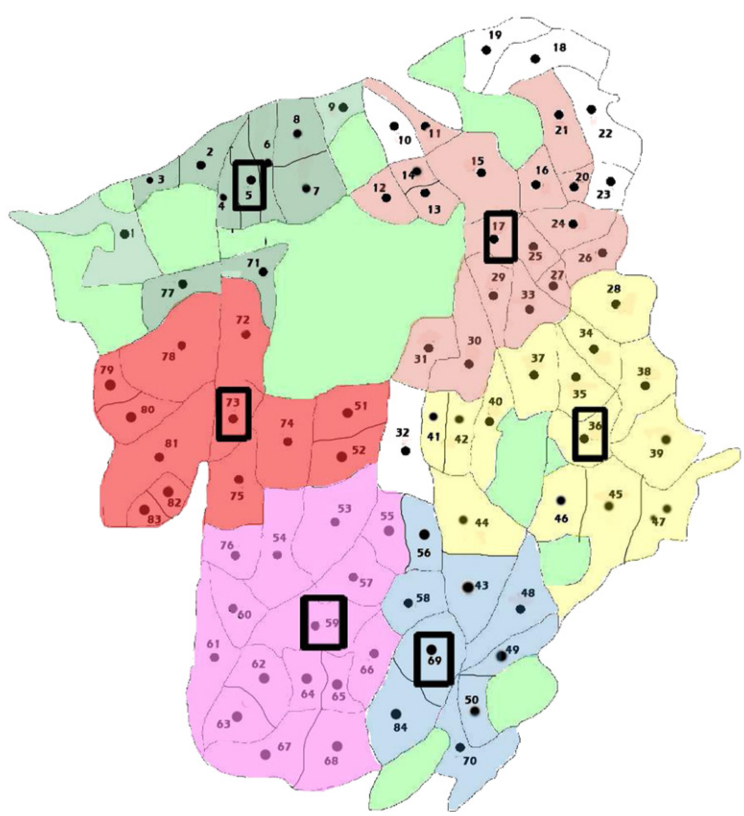

Fig. 5. Optimal solution of *CoverageVLP* for a **D**-type instance.

depend on the distances between demand points and their vendors.

Moreover, the number of demand points served is larger in type **D** problems compared to type **A** problems. This is again expected as the profits and demands decrease as distances increase in **D**-type instances.

The total profits are 1152.70, 1214.32, 1032.20, and 992.36 and the amounts of demand covered are 2180, 2229, 2446, and 2263 for *ProfitVLP* for type **A**, *ProfitVLP* for type **D**, *CoverageVLP* for type **A**, and *CoverageVLP* for type **D** instances, respectively.

6. Conclusion

In this study, motivated by a real life application, we introduced the vendor location problem. We considered two versions of the problem with different objective functions. We proved that both versions of the problem are strongly NP-hard and suggested valid inequalities to strengthen the integer programming formulations and to reduce the solution times.

Our computational experiments showed that the bounds of the linear programming relaxations of the problem with profit maximization objective are quite poor in quality and it is very difficult to solve these problems to optimality with integer programming solvers. Our valid inequalities strengthened our formulations significantly and reduced the computation times, however their effect was highly dependent on the instance. We also observed that the problem with the coverage objective was relatively easier to solve and valid inequalities were also useful in reducing the solution times for the instances of this problem.

We solved instances with different demand and profit functions and observed that the problems with profit maximization objective, where the demands change as a function of the distances between the demand points and their vendors are more difficult to solve compared to others. For some of these instances, we could not reach an optimal solution with any of our models. Even though the final gaps are not very large, still, we believe that alternative methods can be developed for these kinds of problems.

Acknowledgment

This research is supported by TUBITAK Project no. 107M460.

References

[1] Aardal K, Chudak FA, Shmoys DB. A 3-approximation algorithm for the k-level uncapacitated facility location problem. Information Processing Letters 1999;72:161–7.
[2] Aardal K, Labbé M, Leung J, Queyranne M. On the two-level uncapacitated facility location problem. INFORMS Journal on Computing 1996;8:289–301.
[3] Ageev A. Improved approximation algorithms for multilevel facility location problems. Operations Research Letters 2002;30:327–32.
[4] Ageev A, Ye Y, Zhang J. Improved combinatorial approximation algorithms for the k-level facility location problem. SIAM Journal on Discrete Mathematics 2004;18:207–17.
[5] Albareda-Sambola M, Fernández E, Laporte G. The capacity and distance constrained plant location problem. Computers & Operations Research 2009;36:597–611.
[6] Barcelo J, Casanovas J. A heuristic Lagrangian algorithm for the capacitated plant location problem. European Journal of Operational Research 1984;15: 212–26.
[7] Barros AI, Labbé M. A general model for the uncapacitated facility and depot location problem. Location Science 1994;2:173–91.
[8] Bloemhof-Ruwaard JM, Salomon M, van Wassenhove LN. On the coordination of product and by-product flows in two-level distribution networks: model formulations and solution procedures. European Journal of Operational Research 1994;79:325–39.
[9] Bumb A. An approximation algorithm for the maximization version of the two level uncapacitated facility location problem. Operations Research Letters 2001;29:155–61.
[10] Bumb A, Kern W. A simple dual ascent algorithm for the multilevel facility location problem. Lecture Notes in Computer Science 2001;2129:55–63.
[11] Chardaire P, Lutton J-L, Sutter A. Upper and lower bounds for the two-level simple plant location problem. Annals of Operations Research 1999;86: 117–40.
[12] Gao JJ, Robinson Jr. EP. A dual-based optimization procedure for the two-echelon uncapacitated facility location problem. Naval Research Logistics 1992;839:191–212.
[13] Gao JJ, Robinson Jr. EP. Uncapacitated facility location: general solution procedure and computational experience. European Journal of Operational Research 1994;76:410–27.
[14] Gabor AF, van Ommeren JKCW. A new approximation algorithm for the multilevel facility location problem. Discrete Applied Mathematics 2010; 158:453–60.
[15] Garey MR, Johnson DS. Computers and intractability, a guide to the theory of NP-completeness. New York: Freeman; 1979.
[16] Guha S, Meyerson A, Munagala K. Hierarchical placement and network design problems. In: Proceedings of the 41st annual IEEE symposium on foundations of computer science; 2000. p. 603612.
[17] Holmberg K, Rönnqvist M, Yuan D. An exact algorithm for the capacitated facility location problems with single sourcing. European Journal of Operational Research 1999;113:544–59.
[18] Kaufman L, Vanden Eede M, Hansen P. A plant and warehouse location problem. Operational Research Quarterly 1997;28:547–57.
[19] Klincewicz J, Luss H. A Lagrangian relaxation heuristic for capacitated facility location with single-source constraints. Journal of the Operational Research Society 1986;37:495–500.
[20] Marín A. Lower bounds for the two-stage uncapacitated facility location problem. European Journal of Operational Research 2007;179:1126–42.
[21] Marín A, Pelegrín B. Applying Lagrangean relaxation to the resolution of two-stage location problems. Annals of Operations Research 1999;86:179–98.
[22] Meyerson A, Munagala K, Plotkin S. Cost-distance: two-metric network design. In: Proceedings of the 41st IEEE symposium on foundations of computer science, FOCS; 2000. p. 624–30.
[23] Neebe A, Rao M. An algorithm for the fixed-charge assigning users to sources problem. Journal of the Operational Research Society 1983;34:1107–13.
[24] Pirkul H, Jayaraman V. Production, transportation, and distribution planning in a multi-commodity tri-echelon system. Transportation Science 1996;30: 291–302.
[25] Ro H, Tcha D. A branch-and-bound algorithm for the two-level uncapacitated facility location problem with some side constraints. European Journal of Operational Research 1984;18:349–58.
[26] Shmoys DB, Tardos E, Aardal K. Approximation algorithms for facility location problems. In: Proceedings of the 29th annual ACM symposium on theory of computing; 1997. p. 265–74.
[27] SUDER 〈http://www.suder.org.tr/sektor.html〉; 2009.
[28] Şahin G, Süral H. A review of hierarchical location models. Computers & Operations Research 2007;34:2310–31.
[29] Tcha D, Lee B. A branch-and-bound algorithm for the multi-level uncapacitated facility location problem. European Journal of Operational Research 1984;18:35–43.
[30] Tragantalerngsak S, Holt J, Ronnqvist M. Lagrangian relaxation heuristic for two-echelon single-source capacitated facility location problem. European Journal of Operational Research 1997;102:611–25.
[31] Tragantalerngsak S, Holt J, Ronnqvist M. An exact method for the two-echelon, single-source, capacitated facility location problem. European Journal of Operational Research 2000;123:473–89.
[32] Zhang J. Approximating the two-level facility location problem via a quasi-greedy approach. Mathematical Programming 2006;108:159–76.
[33] Zhang J, Ye Y. A note on the maximization version of the multi-level facility location problem. Operations Research Letters 2002;30:333–5.