An Exact Algorithm for Designing Optimal Districts in the Collection of Waste Electric and Electronic Equipment through an Improved Reformulation

Roger Z. Ríos-Mercado[1]
Universidad Autónoma de Nuevo León (UANL)
Graduate Program in Systems Engineering
San Nicolás de los Garza, NL 66450, Mexico
roger.rios@uanl.edu.mx

Jonathan F. Bard
The University of Texas at Austin
Graduate Program in Operations Research and Industrial Engineering
Austin, TX 78712
jbard@utexas.edu

April 2018
Revised August 2018
Revised December 2018

[1]Corresponding author

Abstract

In this paper, a maximum dispersion districting problem is considered that arises in the collection of waste electric and electronic equipment. For a given geographic region, the problem is to partition a set of collection or basic units such that each district in the region is assigned to a different company in a way that maximizes a dispersion function subject to a set of planning requirements. Starting with the traditional mixed-integer programming model, a covering-type model is proposed that is shown to be much more effective. In addition, a new upper bound for the maximum dispersion partitioning problem – a relaxation of the problem studied – is developed. Next, an exact algorithm based on the reformulated model is presented along with the newly derived upper bound. The algorithm intelligently exploits properties of the problem in a manner that allows for a large number of binary variables to be fixed and eliminated in a pre-processing step.

Extensive testing indicates that the new bound yields speed-ups with respect to the best existing bound of up to 23%. The computations also show that the new covering-type model has a tighter linear programming relaxation, and thus is faster to solve than the standard model. The proposed exact algorithm is able to find optimal solutions to instances with up to 800 basic units and 12 companies, and to instances with up to 1400 basic units and 8 companies. Previously, the largest instances solved optimally had between 40 to 100 basic units and 4 companies.

Keywords: Large scale optimization; Integer programming; WEEE Collection; Districting; Maximum dispersion

1 Introduction

In 2003, the European Commission approved Directive 2002/96/EC that regulates the disposal, reuse, recycling, and other forms of recovery of waste electric and electronic equipment (WEEE). A revision in 2003 led to WEEE II Directive (2012/19/EU), which seeks to improve the environmental performance of all parties involved in the lifecycle of such products. As a result, many interesting strategic, tactical and operational problems have arisen in the last few years.

One such problem is concerned with assigning a given set of collection units or basic units (BUs) to the companies responsible for the collection of WEEE. This is a design problem that occurs prior to the actual routing and physical collection of the discarded items and products.

Specifically, the design problem consists of assigning a given set of BUs to a given set of companies in such a way that certain planning and legal requirements are met. In practice, each BU handles two different types of products: (i) those with toxic emissions and (ii) those without, and it is assumed that the market share for each product type and company is known. Also, each BU is classified into one of three categories according to its infrastructure capability; that is, the extent to which items can actually be recycled in its facilities. Among the requirements are that (a) the average number of items collected by each company for each product type be proportional to its market share, (b) a minimum number of collection stations must exist based on their infrastructure capability, (c) the number of split BUs is bounded, where a *split BU* is one that is assigned to different companies for each of its two product types, and (d) the BUs assigned to each company should be as geographically dispersed as possible. This last requirement is imposed by the WEEE Directive to avoid regional monopolies and is achieved by maximizing a dispersion function. The resultant problem is referred to as the maximum dispersion territory design problem (MaxD-TDP.)

Comprehensive reviews on districting and territory design, in general, are given by Kalcsics (2015), Duque et al. (2007), Ricca et al. (2013), and Zoltners and Sinha (2005). One important feature that makes the MaxD-TDP unique with respect to other districting and territory design problems is the nature of its objective function. While in practically every application of these problems the goal is to obtain compact territories (by minimizing a dispersion measure), in the problem addressed here the anti-monopolistic nature of the law leads to the opposite goal.

An examination of the literature on analytic approaches to WEEE recycling and collection indicates that most of the work has focused on issues such as routing, recollection, and treatment plant location. Very little has been done in terms of assigning BUs to waste management companies. To the best of our knowledge, this territory design problem has only been addressed by Fernández et al. (2010) and Ríos-Mercado et al. (2017). In the former, the authors introduce the problem, present a mixed-integer linear programming (MILP) model, prove its NP-completeness, and develop a greedy randomized adaptive search procedure (GRASP) that is able to efficiently handle instances with up to 400 BUs and 7 companies. In the latter, the authors present an enhanced tabu search metaheuristic.

Also related to our research is the study by Fernández et al. (2013) of a maximum dispersion partition problem called MaxDP. In that paper, the authors present an exact optimization methodology that relies heavily on an upper bounding scheme for a relaxation of MaxDP that provides very good results. Although we are not solving MaxDP in this paper due to the presence of split BUs, if we relax some of the constraints in our problem what results is precisely the unrestricted max-dispersion problem (called UMaxDP). This means that the previously developed bound is indeed valid for our problem, i.e., MaxD-TDP.

The contributions of this paper are several fold.

- First, a new and improved upper bound to the (related) maximum dispersion problem (UMaxDP), which in turn is also a bound for our problem, is presented along with a formal proof. This new bound can be efficiently computed and is empirically shown to be much tighter than the best current bound. In fact, our results indicate that the proposed bound yields average speed-ups of up to 23% with respect to the existing bound.

- The introduction of a new and improved reformulation of the MaxD-TDP, including several properties that allows for fixing some of the binary variables. The new model is motivated by the fact that problem reformulation has proven successful for other location problems (Church, 2008; Elloumi et al., 2004; Fernández et al., 2013; Marín et al., 2009).

- An exact optimization algorithm based on a biased binary search scheme. The methodology effectively exploits the tightness of the new upper bound and the new properties associated with the reformulated model.

- An extensive computational study to assess each component of the proposed methodology, including tests with very large instances (up to 1400 BUs) not tested previously. Our empirical work includes a comparison between the two MILPs mentioned. The results indicate that our algorithm can find optimal solutions to instances with up to 800 BUs and 12 companies and up to 1400 BUs and 8 companies. To date, the largest instances solved optimally had between 30 to 100 BUs and 4 companies. Previous to this work, only heuristics existed for the MaxD-TDP, so our algorithm is the first exact method.

The rest of the paper is organized as follows. In Section 2, we survey related work. In Section 3, we formally state the maximum dispersion problem, give its standard MILP formulation, and present our proposed reformulation. The derivation of our new upper bound is given in Section 4 followed in Section 5 by a full description of the optimization algorithm. Computational results are provided in Section 6 and concluding remarks are given in Section 7.

2 Related Work

Territory design or districting is a field that has been active since the mid-1960s, with the early work mostly focusing on political districting and the partitioning of sales regions. Over the past 15-20 years, its scope has been extended to a wide range of applications. One factor that makes our work unique with respect to the literature is the absence of a compactness criterion. While practically all existing research on districting is concerned with generating compact territories, our goal is to maximize dispersion (due to the anti-monopoly clause of the law). Thus, former models and methods developed for districting problems are not generally applicable to our work. For a broader view on districting research, the reader is referred to the survey papers by Zoltners and Sinha (2005), Duque et al. (2007), Pukelsheim et al. (2012), Ricca et al. (2013), and Kalcsics (2015).

With respect to WEEE decision-making problems, much of the modeling effort has centered on routing, recollection and treatment plant location, rather than system design (e.g., see Georgiadis and Besiou (2010); Grunow and Gobbi (2009); Hammond and Beullens (2007); Lee and Shih (2012); Mar-Ortiz et al. (2011, 2013); Tsai and Hung (2009); Queiruga et al. (2008); Rudăreanu (2013)).

Nevertheless, there has been some research addressing partitioning-type problems with dispersion objectives. The maximally diverse grouping problem, for example, is aimed at maximizing the sum of pairwise distances within groups. Fan et al. (2011) and Gallego et al. (2013) present heuristics for this problem. The maximum diversity problem in which a subset of elements must be chosen from a larger set so as to maximize the sum of distances between the chosen elements also falls in this category. Martí et al. (2010) present a recent overview on this problem and a branch-and-bound algorithm for its solution. Kuo et al. (1993) studied the same problem but with a different objective function. They introduced a model where the minimal pairwise distance instead of the sum of distances is maximized.

The maximum dispersion problem was introduced by Fernández et al. (2013) who present an integer programming model, develop several properties, and present an upper bounding scheme and an exact algorithm. Some of the results included in the above-mentioned papers will be further discussed in subsequent sections.

With respect to the location literature, a common problem known as the p-dispersion problem (Erkut, 1990; Erkut and Neuman, 1991), involves placing p points on a plane as far away from each other as possible. Although this is not a partitioning problem, maximum dispersion functions are typically used.

To the best of our knowledge, the only other work on WEEE districting is due to Fernández et al. (2010), who introduced the problem and proposed a GRASP that was successful in finding high quality solutions, and Ríos-Mercado et al. (2017) who presented an enhanced tabu search algorithm. These two approaches are heuristics. Our contribution centers on the development of an exact optimization scheme for the same problem. Although the problem we address possesses features similar to those in the maximum dispersion problem, it is important to note that we are not solving a partitioning problem due to the presence of additional constraints and the fact that BUs can be split.

3 Problem Description and Formulations

What makes the MaxD-TDP different and interesting is that the partitioning plan must also satisfy the WEEE Directive, which asserts that regional monopolies must be avoided; that is, BUs allocated in smaller subregions should be assigned to different companies to the greatest extent possible. As shown by Fernández et al. (2010), this is accomplished by maximizing a dispersion measure, in contrast to previous work where a dispersion measure is commonly minimized to achieve territory compactness.

Fernández et al. (2010) studied two dispersion functions, one based on maximizing the sum of inter-cluster distances and the other based on maximizing the minimum inter-cluster distance. Their results indicated that the latter was a more robust measure yielding more representative designs in terms of avoiding regional monopolies. Accordingly, this objective function will be used here.

In the remainder of this section we provide a summary of the main assumptions and planning requirements that underpin MaxD-TDP. We then present two MILPs: the first is from Fernández et al. (2010) called MDTDP, and the second is a new reformulation that makes use of covering-type variables, which we call MDTDP-CF.

3.1 Problem Statement

Let $V = \{1, \ldots, n\}$ be the set of BUs, where a BU corresponds to a collection point. Let h_i be the number of households associated with BU $i \in V$ and let $H = \sum_{i \in V} h_i$ be the sum of all households. It is assumed that the number of items to be collected is proportional to the number of households. Each BU is further classified according to the nature or quality of its infrastructure. Denote by V_1, V_2 and V_3 the set of BUs of good, medium and low quality, respectively. We use $q \in Q = \{1, 2, 3\}$ as an index for the quality set and $q_i \in Q$ as the quality index of BU i. Accordingly, we have $V_q = \{i \in V \ : \ q_i = q\}$ for all $q \in Q$.

Electric and electronic appliances and equipment are further subdivided into items that have freezing capabilities and those that do not (referred to as type 1 and type 2 products, respectively). This distinction is necessary due to the toxic cooling solvents contained in type 1 products which require special treatment. Let d_{ij} be the Euclidean distance between BUs i and j, $i, j \in V$. We denote by $C = \{1, \ldots, m\}$ the set of companies or territories and M_k^p the market share of company $k \in C$ for product type $p \in P = \{1, 2\}$. It is also assumed that $m < n$. In fact, for most practical instances $m << n$. As market shares may differ for the two product types, it is permissible to split BUs, i.e., for some BUs the company that collects type 1 products may not be the same as the one that is responsible for the type 2 products. A BU that is assigned to different companies for the two product types is called a *split unit*. Let $c^q(\bar{V}) = |\bar{V} \cap V_q|$ denote the cardinality of subset $\bar{V} \subset V$ with respect to quality index $q \in Q$.

The problem consists of assigning BUs and product types to companies in a way that avoids regional monopolies in accordance with law. This is achieved by maximizing a dispersion function.

The following requirements partially define the problem.

- For each product type $p \in P$, a BU must be assigned to a company; in particular, for each p the assignment forms an m-partition of V.
- The number of split units is bounded by a user-specfied parameter σ.
- BUs should be assigned to companies in such a way that the total items to be collected by each company is proportional to its market share for each product type. This is handled by assuming that the amount of items is proportional to the number of households associated to a collection unit.
- The number of BUs with a specific quality index should be proportionally assigned to companies based on their market share for each product type.

- It is assumed that each company is assigned at least two BUs, that is, there are no (trivial) singleton territories. Note that this assumption does not restrict the generality of our results for the following reason. A feasible singleton territory solution exists if there is a unit i and a territory k such that

  $$h_i \in [(1-\tau)HM_k^p, (1+\tau)HM_k^p] \quad \text{for } p \in P$$

  and

  $$1 \in [(1-\beta)|V_{q_i}|M_k^p, (1+\beta)|V_{q_i}|M_k^p] \quad \text{for } p \in P$$

  where τ and β are parameters (see constraints (4)–(7) below). Since these conditions can be easily checked beforehand, if such a unit exists we can simply remove it and the corresponding company. This would leave us with a problem where each territory has at least two units assigned to it.

- Connectivity among BUs is not considered because the WEEE II Directive does not require it.

For modeling the last two requirements, we introduce parameters $0 \le \tau, \beta \le 1$, respectively, that are used for setting a limit in the maximum deviation allowed from a known target value. See Eqs. (4)–(7) below.

Let D be a sufficiently large number representing an upper bound on the objective function value. A trivial bound would be $D = \max_{i,j \in V}\{d_{ij}\}$, which is valid under the assumptions that $m < n$ and that there are no trivial singleton territories; that is, each territory is formed by two or more BUs.

3.2 MDTDP Model

In the specification of the first MILP, we make use of the the following binary decision variables. For $i, j \in V$, $k \in C$, and $p = 1, 2$, let

$$y_{ik}^p = \begin{cases} 1 & \text{if BU } i \text{ is assigned to company } k \text{ for product type } p \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ik} = \begin{cases} 1 & \text{if BU } i \text{ is assigned to company } k \text{ for at least one of the two product types} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ijk} = \begin{cases} 1 & \text{if BUs } i \text{ and } j \text{ are both assigned to company } k \text{ for at least one of the two product types} \\ 0 & \text{otherwise} \end{cases}$$

$$w_i = \begin{cases} 1 & \text{if BU } i \text{ is assigned to different companies for product types 1 and 2} \\ 0 & \text{otherwise} \end{cases}$$

Noting that the z_{ijk}-variables are symmetric with respect to i and j, i.e., $z_{ijk} = z_{jik}$, we only need to consider the case $i < j$. Accordingly, we define $E = \{(i,j) : i, j \in V, i < j\}$. The MILP is given as follows.

MDTDP Model

$$\text{Maximize} \quad u \tag{1}$$

$$\text{subject to} \quad u \leq d_{ij} z_{ijk} + D(1 - z_{ijk}) \qquad (i,j) \in E,\ k \in C \tag{2}$$

$$\sum_{k \in C} y_{ik}^p = 1 \qquad i \in V,\ p \in P \tag{3}$$

$$\sum_{i \in V} h_i y_{ik}^p \leq (1+\tau) \cdot H \cdot M_k^p \qquad k \in C,\ p \in P \tag{4}$$

$$\sum_{i \in V} h_i y_{ik}^p \geq (1-\tau) \cdot H \cdot M_k^p \qquad k \in C,\ p \in P \tag{5}$$

$$\sum_{i \in V_q} y_{ik}^p \leq (1+\beta) \cdot |V_q| \cdot M_k^p \qquad k \in C,\ p \in P,\ q \in Q \tag{6}$$

$$\sum_{i \in V_q} y_{ik}^p \geq (1-\beta) \cdot |V_q| \cdot M_k^p \qquad k \in C,\ p \in P,\ q \in Q \tag{7}$$

$$\sum_{i \in V} w_i \leq \sigma \tag{8}$$

$$y_{ik}^1 - y_{ik}^2 \leq w_i \qquad i \in V,\ k \in C \tag{9}$$

$$y_{ik}^2 - y_{ik}^1 \leq w_i \qquad i \in V,\ k \in C \tag{10}$$

$$x_{ik} \leq y_{ik}^1 + y_{ik}^2 \qquad i \in V,\ k \in C \tag{11}$$

$$y_{ik}^p \leq x_{ik} \qquad i \in V,\ k \in C,\ p \in \mathrm{P} \tag{12}$$

$$x_{ik} + x_{jk} \leq 1 + z_{ijk} \qquad (i,j) \in E,\ k \in C \tag{13}$$

$$z_{ijk} \leq x_{ik} \qquad (i,j) \in E,\ k \in C \tag{14}$$

$$z_{ijk} \leq x_{jk} \qquad (i,j) \in E,\ k \in C \tag{15}$$

$$w_i, x_{ik}, y^p_{ik}, z_{ijk} \in \{0,1\} \qquad (i,j) \in E,\ k \in C,\ p \in P \tag{16}$$

The linearized dispersion objective is modeled by (1)-(2). Constraints (3) ensure that each BU is assigned to a company for each product type. Constraints (4)-(5) ensure that the number of households is fairly distributed to companies based on their market share. Due to the discrete nature of the problem, it is practically impossible to obtain a perfect balance. To get around this issue, we introduce a tolerance parameter $\tau \in (0,1)$ that measures the deviation from a perfect measure given by $H \cdot M^p_k$. Similarly, constraints (6)-(7) ensure that the good, medium and low quality BUs are fairly allocated to companies based on their market share. To this end, a user-specified tolerance parameter $\beta \in (0,1)$ is introduced. These two sets of constraints are referred to as the *household* and *infrastructure quality* balancing constraints, respectively. Note that tolerance values of $\tau = \beta = 0$ correspond to a perfect balance.

Constraint (8) places a limit on the number of split BUs allowed. In practice this is around 20% of the total number. Constraints (9)-(10) establish the relationship between the w- and y-variables, constraints (11)-(12) establish the relationship between the x- and y-variables, and constraints (13)-(15) establish the relationship between the x- and z-variables. Note that constraints (14) and (15) are not really needed due to the maximization objective defined by (1)-(2). To obtain the maximum value of u we would like to set $z_{ijk} = 0$ whenever possible. Thus, $z_{ijk} > 0$ only when the right-hand side of (13) is required to be 2 to achieve feasibility. Accordingly, we drop (14) and (15) from the formulation.

It should be mentioned that although we consider an index set Q with three elements, the model is valid for a general set Q. In terms of the number of products, the model is valid for only two product types; however, it can easily be extended to allow a general set P by replacing constraints (9)-(10) with the following.

$$y^p_{ik} - y^t_{ik} \leq w_i \qquad i \in V, k \in C, p, t \in P (p \neq t)$$

In this case, a split BU is defined as a unit whose different product types are not all assigned to the same company.

Fernández et al. (2010) showed that the MaxD-TDP is NP-hard. The state of the art says that tractable instances of the maximum dispersion problem, i.e., those that can be solved exactly, have on the order of 20-30 BUs and 3-4 companies. Our target instances have 300-800 BUs and 5-10 companies.

3.3 A Covering-Based Formulation

The following reformulation of the problem is motivated by the success evidenced by using covering-type models to represent other location problems such as the p-center problem (Elloumi et al., 2004), the p-median problem (Church, 2008), and the maximum dispersion territory design problem (Fernández et al., 2013), to name a few. The idea is as follows. Let $\{d^1, d^2, \ldots, d^{\tilde{r}}\}$ be the set of $\tilde{r}$ different distance values between BUs sorted in increasing order. That is, $0 < d^1 < d^2 < \ldots < d^{\tilde{r}} = D$, where $R = \{1, \ldots, \tilde{r}\}$ represent the index set. In the worst case, when all distances are unique, we have $\tilde{r} = n(n-1)/2$. In practice, $\tilde{r}$ is smaller than this value.

Now, defining the following binary variables for $r \in R$,

$$v_r = \begin{cases} 1 & \text{if the overall smallest pairwise distance is at most } d^r \\ 0 & \text{otherwise} \end{cases}$$

the MILP covering formulation is given by

MDTDP-CF Model

$$\text{Maximize} \quad d^{\tilde{r}} + \sum_{r=1}^{\tilde{r}-1} (d^r - d^{r+1}) v_r \tag{17}$$

$$\text{subject to} \quad x_{ik} + x_{jk} \le 1 + v_r \qquad i, j \in V, i < j,\ k \in C,\ r \in R : d_{ij} = d^r \tag{18}$$

$$v_{r-1} \le v_r \qquad r \in R \setminus \{1\} \tag{19}$$

$$\sum_{k \in C} y^p_{ik} = 1 \qquad i \in V,\ p \in P \tag{20}$$

$$\sum_{i \in V} h_i y^p_{ik} \le (1 + \tau) \cdot H \cdot M^p_k \qquad k \in C,\ p \in P \tag{21}$$

$$\sum_{i \in V} h_i y^p_{ik} \ge (1 - \tau) \cdot H \cdot M^p_k \qquad k \in C,\ p \in P \tag{22}$$

$$\sum_{i \in V_q} y_{ik}^p \leq (1+\beta) \cdot |V_q| \cdot M_k^p \qquad k \in C,\, p \in P,\, q \in Q \tag{23}$$

$$\sum_{i \in V_q} y_{ik}^p \geq (1-\beta) \cdot |V_q| \cdot M_k^p \qquad k \in C,\, p \in P,\, q \in Q \tag{24}$$

$$\sum_{i \in V} w_i \leq \sigma \tag{25}$$

$$y_{ik}^1 - y_{ik}^2 \leq w_i \qquad i \in V,\, k \in C \tag{26}$$

$$y_{ik}^2 - y_{ik}^1 \leq w_i \qquad i \in V,\, k \in C \tag{27}$$

$$x_{ik} \leq y_{ik}^1 + y_{ik}^2 \qquad i \in V,\, k \in C \tag{28}$$

$$y_{ik}^p \leq x_{ik} \qquad i \in V,\, k \in C,\, p \in \mathrm{P} \tag{29}$$

$$v_r, w_i, x_{ik}, y_{ik}^p \in \{0,1\} \qquad i,j \in V, i < j,\, k \in C,\, p \in P,\, r \in R \tag{30}$$

Basically, the objective function and the z-variables from the previous model are dropped along with all constraints in which they appear. A new objective function (17) and set of constraints (18)-(19) are then introduced in their place. For any feasible solution yielding the objective function value d^s, we have $v_r = 0$ for $r \leq s-1$ and $v_r = 1$ for $r \geq s$. This reduces the number of binary variables from $O(mn^2)$ in MDTDP down to $O(n^2)$ in MDTDP-CF.

Both models MDTDP and MDTDP-CF are equivalent as they yield the same objective function value at optimality. Moreover, the optimal values of the decision variables that define how the territory is partitioned are also the same. This type of reformulation has proven successful for similar problems and is a consequence of the model structure rather than a reduction in problem size. In fact, the reformulated model may be larger or smaller than MDTDP depending on the cardinality of R. However, by eliminating constraints (2) MDTDP-CF has a tighter linear programming (LP) relaxation than MDTDP. This is not surprising as it is well known that integer programming models based on big-M type constraints tend to have bad LP relaxations.

3.4 Model Properties

Given the nature of the objective function, the following properties for what Fernández et al. (2013) called the unrestricted maximum dispersion design problem (UMaxDP), also hold for the MDTDP model.

- *Property 1:* Let L be a lower (primal) bound on the optimal objective function value u^* in

(1). Then, in any optimal solution we must have $z_{ijk} = 0$ for all $(i, j) \in E$ and $k \in C$ such that $d_{ij} < L$.

- *Property 2:* Let U be an upper (dual) bound on u^*. Then we can fix $z_{ijk} = 1$ and eliminate constraints (13) for all $(i, j) \in E$ and $k \in C$ with $d_{ij} > U$.

- *Property 3:* If U is a valid upper bound on u^*, then the MILP can be strengthened by replacing D by U in constraints (2).

These properties have been successfully exploited in solution algorithms for problems with similar structure such as the maximum dispersion partition problem (MaxDP). Going a step farther, for the MDTDP-CF model, properties 1 and 2 can be extended to

- *Property 4:* Let L be a lower (primal) bound on the optimal objective value u^* such that $L = d^s$ for some $s \in R$. If L does not match any d^s, that is, $d^{s-1} < L < d^s$ for some $s \in R \cup \{0\}$, then L can be replaced by d^s. Therefore, in any optimal solution we must have $v_r = 0$ for $r \leq s - 1$.

- *Property 5:* Let U be an upper (dual) bound on u^* such that $U = d^s$ for some $s \in R$. Again, if U does not match any d^s, that is $d^s < U < d^{s+1}$ for some $s \in R$, then U can be replaced with d^s. Therefore, in any optimal solution we can fix $v_r = 1$ for $r \geq s$.

4 Upper Bounding Scheme

We begin the derivation of upper bounds for MDTDP by removing the binary variables y_{ik}^p and w_i and the constraints in which they appear, (3)-(12), to obtain UMaxDP. Although this problem remains NP-hard (Fernández et al., 2013), in practice, it is relatively straightforward to solve and any upper bound for UMaxDP is a valid upper bound for MDTDP. The relaxed model is given by:

$$\text{(UMaxDP) Maximize} \quad u \tag{31}$$

$$\text{subject to} \quad u \leq d_{ij} z_{ijk} + D(1 - z_{ijk}) \qquad (i, j) \in E,\ k \in C \tag{32}$$

$$\sum_{k \in C} x_{ik} = 1 \qquad i \in V \tag{33}$$

$$x_{ik} + x_{jk} \leq 1 + z_{ijk} \qquad (i, j) \in E,\ k \in C \tag{34}$$

$$z_{ijk} \leq x_{ik} \qquad (i,j) \in E,\, k \in C \tag{35}$$

$$z_{ijk} \leq x_{jk} \qquad (i,j) \in E,\, k \in C \tag{36}$$

$$x_{ik}, z_{ijk} \in \{0,1\} \qquad (i,j) \in E,\, k \in C \tag{37}$$

The following upper bounds are known for UMaxDP (Fernández et al., 2013). Their validity is based on the assumption that each company is assigned at least two BUs, which is precisely one of the assumptions made in Section 3.1.

- A simple *greedy upper bound* is given by

$$U^1 = \min_{i \in V} \max_{j \in V} d_{ij}. \tag{38}$$

- The authors developed an upper bound based on subsets of size $m+1$ (see next subsection). They empirically showed that this bound is significantly better than any other known bound. We call this bound $U^{(m+1)}$ or U^{FKN}.

4.1 New Upper Bound

The proposed upper bound is an extension of U^{FKN}. In their work, Fernández et al. (2013) proved that for any given subset S of size $m+1$, the term $\max_{i,j \in S}\{d_{ij}\}$ gives an upper bound on the optimal value u^* of UMaxDP. This result is valid under the assumption that there are no trivial singleton territories; that is, there are at least two BUs assigned to each territory. Then, by considering all possible families of subsets of size $m+1$, they set

$$U^{\text{FKN}} = \min_{S \subset V^{(m+1)}} \max_{i,j \in S}\{d_{ij}\},$$

where $V^{(k)} = \{S \subset V : |S| = k\}$ is the collection of all subsets of V of cardinality k. Clearly, $V^{(m+1)}$ grows exponentially with m; therefore, the authors suggest using a heuristic to compute this bound for a partial collection of subsets of $V^{(m+1)}$.

The main idea of our approach is to consider subsets of V of size $m+2$. This of course increases the number of combinations of the different cases, but the total number of cases is still tractable for realistic instances. The argument is as follows.

Let S be an arbitrary subset of V such that $|S| = m+2$. The idea is to consider different disjoint cases that may occur when distributing these $m+2$ BUs among m territories. Consider case t consisting of all possible combinations of $m + 2$ BUs among the m territories such that the largest subset has cardinality t. The smallest possible value for t is 2 (it is impossible to accommodate $m+2$ BUs in m territories by placing just one unit in each territory), and the largest possible value for t is 3 (this is the case when 3 units belong to the same territory and each of remaining $m - 1$ units belongs to one of the remaining $m - 1$ territories, one unit per territory). Thus, we have two disjoint cases as a function of $t \in \{2, 3\}$. Considering all disjoint cases, a valid upper bound for UMaxDP is given by $\max\{B^3(S), B^2(S)\}$. Let us now focusing on computing $B^3(S)$ and $B^2(S)$.

Bound computation

Computing $B^3(S)$**:** This is the case where $m+2$ units in set S are distributed in such a way that the largest subset of S has 3 BUs. This can occur only when there is a partition of S such that there is subset of cardinality 3 and each of the remaining subsets in the partition has cardinality 1. Because each of the latter is irrelevant to the bound computation, it suffices to compute the bound for each possible subest of S of size 3. Now, for an arbitrary subset H of S of size 3, a valid bound is given by $\min_{i,j \in H}\{d_{ij}\}$. If we consider the worst possible value over all subsets of size 3, $B^3(S)$ can be found as follows:

$$B^3(S) = \max_{\substack{H \subset S \\ |H|=3}} \min_{i,j \in H} \{d_{ij}\}$$

Computing this value can be done efficiently. The number of subsets of S of size 3 is given by $\binom{m+2}{3}$, so the computational effort is proportional to $O(m^3)$. Recall that m is a relatively small number in practice.

Computing $B^2(S)$**:** This is the case where the $m + 2$ units of set S are distributed in such a way that the largest subset of S has two BUs. That is, there are two subsets with two BUs and the remaining $m - 2$ subsets have just one unit each. Let us refer to these two subsets of cardinality 2 as S_1 and S_2. Because each of the remaining subsets has only one unit, they are irrelevant for computing the bound. If we think of S as a complete subgraph where each edge $(i, j) \in S \times S$ $(i \neq j)$ has length d_{ij}, or alternatively, edge $a \in S \times S$ has length d_a, then the worst case scenario for computing a valid upper bound is given by finding two disjoint edges a and b such that $\min\{d_a, d_b\}$ is maximized. This is valid because each of the disjoint edges, comprised of two nodes, corresponds to one of the two subsets of cardinality 2 considered in this case. Since the minimum of the two is an upper bound on the objective function, when considering all possible combinations, the largest of these will be a valid upper bound for the problem. Letting $T(S)$ be the collection of all pairs of disjoint edges for a given S, the bound is then given by:

$$B^2(S) = \max_{(a,b)\in T(S)} \min\{d_a, d_b\} \tag{39}$$

Computing this value can be done efficiently. In the worst case, we have only to search for all edges in S, which can done in $O((m+2)^2)$ time.

Accordingly, by considering all possible subsets of size $m+2$ we can compute an upper bound on the optimal value u^* of UMaxDP as follows.

$$U^{\mathrm{RB}} \equiv U^{(m+2)} = \min_{S\subset V^{(m+2)}} \max\{B^3(S), B^2(S)\} \tag{40}$$

The following theorem that states the dominance of upper bound $U^{(m+2)}$ over $U^{(m+1)}$.

Theorem 1. *For any m, $U^{(m+2)} \leq U^{(m+1)}$.*

Proof. Let $U^{(m+1)}$ be the best upper bound under subsets of size $m+1$ as described above. Furthermore, let $\hat{S}$ be the subset achieving this bound, that is,

$$\hat{S} = \operatorname*{argmin}_{S\subset V^{(m+1)}} \max_{i,j\in S}\{d_{ij}\}.$$

Without loss of generality, let $\hat{S} = \{v_1, v_2, \ldots, v_{m+1}\}$, where the first two BUs are the ones achieving this bound, that is,

$$(v_1, v_2) = \operatorname*{argmax}_{(i,j)\in\hat{S}\times\hat{S}} \{d_{ij}\}. \tag{41}$$

Hence, we know $U^{(m+1)} = d_{v_1,v_2}$ and $P = (\{v_1, v_2\}, \{v_3\}, \ldots, v_{m+1}\})$ is the m-partition of $\hat{S}$ yielding this bound. Now, let us consider an arbitrary BU $k \in V \setminus \hat{S}$ and the set $S = \hat{S} \cup \{k\}$, such that $|S| = m+2$. Our goal at this point is to compute a bound based on this subset of size $m+2$ and compare it with $U^{(m+1)}$. From partition P, there are two possible disjoint cases when attempting to compute the bound based on S. The bound is given by either $B^3(S)$ or $B^2(S)$. In the former case we must compute all possible subsets of size 3 and carefully analyze the different cases to determine whether or not k is part of a particular subset of size 3. In the latter case, joining k to a singleton subset in $\hat{S}$ would mean having two disjoint subsets of size 2, namely $\{v_1, v_2\}$ and $\{k, v_q\}$ for some $q = 3, 4, \ldots, m+1$, and thus a computation of $B^2(\hat{S} \cup \{k\})$ would be required.

Now let us consider these two disjoint cases.

- Case (A): Computing $B^3(S)$. To compute $B^3(S)$, let $H = \underset{\substack{H \subset S \\ |H|=3}}{\operatorname{argmax}} \min_{i,j \in H} \{d_{ij}\}$ such that H is the subset of S giving this bound. There are three possible disjoint sub-cases:
  - Sub-case (A.1): $k \notin H$. Now, if $k \notin H$, then $B^3(S) \leq d_{v_1,v_2}$ because $k \notin \hat{S}$ and by construction (v_1, v_2) is the largest edge in $\hat{S}$.
  - Sub-case (A.2) $k \in H$ and the smallest edge in H does not contain k. Letting (i, j) be the smallest edge, we have $B^3(S) = d_{ij} \leq d_{v_1,v_2} = U^{(m+1)}$ by definition of $U^{(m+1)}$ given by (41) and because $i, j \in \hat{S}$. Thus $B^3(S) \leq U^{(m+1)}$.
  - Sub-case (A.3): $k \in H$ and the smallest edge in H does contain k. Letting (i, k) be the smallest edge in $H = \{i, j, k\}$, we have $d_{ik} \leq d_{ij}$ by construction. However, because $i, j \in \hat{S}$ and by definition of $U^{(m+1)}$ given by (41), we have $d_{ij} \leq d_{v_1,v_2} = U^{(m+1)}$

  Thus, all three sub-cases imply $B^3(S) \leq U^{(m+1)}$.

- Case (B): Computing $B^2(S)$. Recall that from the computation of (39) above, the length of the second largest edge in $\hat{S}$ is a valid upper bound. By adding the new BU k, there are $(m+1)$ additional edges in the subgraph, namely (k, j) for each $j \in \hat{S}$. Therefore, depending on the length of these new edges, three disjoint cases may occur as shown in Figure 1.
  - Sub-case (B.1): Largest edge is (k, j) for some $j \in \hat{S}$ such that $j \neq v_1$ and $j \neq v_2$. That is, the largest edge is incident to the new BU k but not incident to either v_1 or v_2. In this case, (v_1, v_2) turns out to be the second largest edge in S (disjoint from (k, j)). Therefore, the bound $B^2(S)$, given by the second largest edge, is $B^2(S) = d_{v_1,v_2}$. For this sub-case then, $U^{(m+1)} = B^2(S)$ so the bound remains the same.
  - Sub-case (B.2): Largest edge is (k, j) for some $j \in \{v_1, v_2\}$, that is, the largest edge is incident to k and to one of the BUs v_1 or v_2. Without loss of generality, assume it is incident to v_1. Then, the second largest edge taken from the set $\hat{S} \setminus \{v_1\}$ cannot by definition be larger than d_{v_1,v_2}. Therefore, $B^2(S) \leq U^{(m+1)}$.
  - Sub-case (B.3): Largest edge is still (v_1, v_2). Thus the second largest edge must be less than or equal to d_{v_1,v_2}, and therefore, $B^2(S) \leq U^{(m+1)}$.

  Thus, for all three sub-cases we have $B^2(S) \leq U^{(m+1)}$.

Taken together, we can conclude $\max\{B^3(S), B^2(S)\}$ is equal or less than the $U^{(m+1)}$. In contrast, set S may not be the optimal set giving the best value of $U^{(m+2)}$, that is $U^{(m+2)} \leq \max\{B^3(S), B^2(S)\} \leq U^{(m+1)}$, and this completes the proof. ■

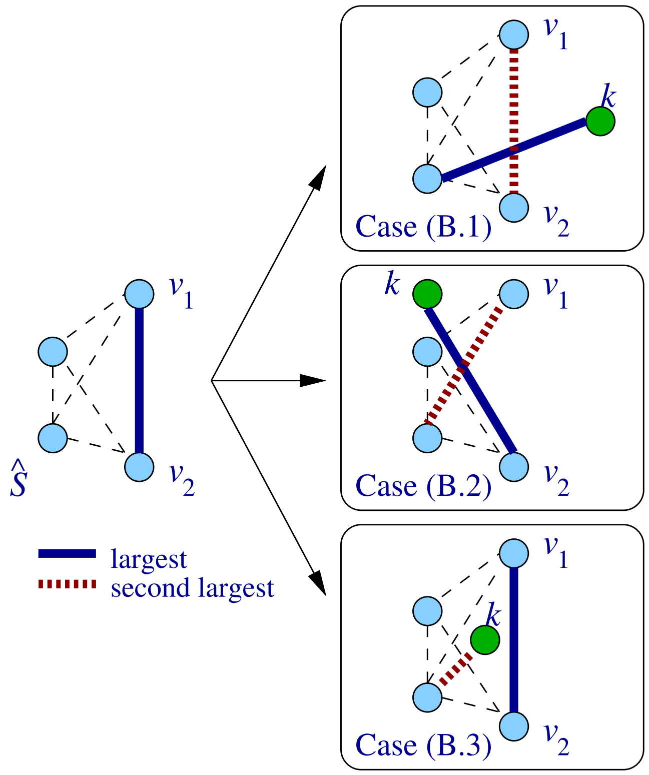

Figure 1: Different sub-cases in proof of Theorem 1 for case (B).

Remark. Empirically, $B^2(S)$ was always observed to be greater than or equal to $B^3(S)$; however, one can construct an example to show that this may not always be the case.

4.2 Description of the Upper Bounding Scheme

Now, when attempting to compute bound $U^{(m+2)}$ according to (40), $|V^{(m+2)}|$ grows exponentially with m, so evaluating all possible subsets could be overwhelming. However, a good approximation can be heuristically computed using the following idea. First, it should be evident that all subsets of $V^{(m+2)}$ whose BUs are located relatively far from each other will not play any role in the bound computation. In fact, the best upper bound will come from the subset yielding the lowest possible value. Therefore, rather than evaluating each possible subset of $V^{(m+2)}$, we focus on intelligently selecting only a fraction of subsets that are potentially good candidates for this computation. In other words, we want to select a collection of subsets of $V^{(m+2)}$ in which the BUs are relatively close to each other because these will be more likely to produce smaller values of $B^2(S)$ and $B^3(S)$.

Algorithm 1 outlines a proposed heuristic for computing U^{RB}. The basic idea is to iteratively include each BU in the computation of the bound. In the main loop, a BU i is chosen, then a set of size $m + 2$ is formed by adding $m + 1$ units to i in such a way that these are relatively close to each other. In all, the procedure computes n possible bounds and outputs the best, all in $O(n(n+m^2))$ time. Furthermore, the quality of the bound found is significantly better than the previously developed bounds, as we will see in Section 6.

Algorithm 1 UB-RB((d_{ij}))

```
Input: $(d_{ij}) \equiv$ distance matrix of dimension $n \times n$
Output: An upper bound for UMaxDP
 1: $best\_bound \leftarrow \infty$
 2: for ( $i = 1, \ldots, n$ ) do
 3:     Obtain $S(i)$, $m+1$ BUs "near" to $i$
 4:     $S \leftarrow \{i\} \cup S(i)$
 5:     $bound \leftarrow$ Compute $\min\{B^2(S), B^3(S)\}$
 6:     if $bound < best\_bound$ then
 7:         $best\_bound \leftarrow bound$
 8:     end if
 9: end for
10: return $best\_bound$
```

Obtaining set $S(i)$ in Step 3 can be done under different strategies. In our case, we proceed as follows: For a given BU i we choose the remaining $m+1$ BUs by iteratively choosing $j^* = \arg\min_{j \in V \setminus S} d(j, S)$ as the element to be added to S until $|S| = m+2$, where the distance between j and set S is given by $d(j, S) = \max_{k \in S}\{d_{ij}\}$. A different strategy might be to simply include the $m+1$ closest BUs to i, as was done by Fernández et al. (2013) in the computation of U^{FKN}. This approach, however, does not take into account the distances between the BUs added and has been shown empirically to yield a worse bound than the proposed approach.

The example depicted in Figure 2 illustrates these two approaches. Suppose that we are forming a subset of size 3 ($m = 2$) and that the iterating unit is node a. We have a partial subset denoted by $S = \{a, c\}$ as indicated in the figure. The numbers on the edges represent the distance between BUs. Thus, under the strategy from Fernández et al. (2013), the closest node to node a is node b. Accordingly, this node would be chosen to enter S and the corresponding bound would be $\max_{i,j \in S}\{d_{ij}\} = 9$. Now, under our proposed strategy we first compute $d(b, S) = 9$ and $d(e, S) = 7$. Because node e gives the minimum value among these, it is chosen to enter S with corresponding bound equal to 7 – clearly better than the previous bound. In any case, each remaining node in V must be examined and $d(j, S)$ must be computed so Step 3 takes $O(nm^2)$ time. Step 5 takes

$O(m^2)$ time.

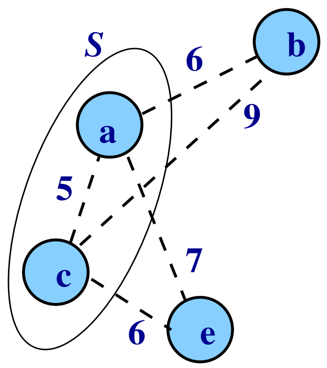

Figure 2: Example of strategies for forming subset S of size 3.

Remark. Applying the same ideas, we developed a family of bounds, call them $U^{(m+h)}$, based on subsets of size $m + h$ for different values of $h > 2$. To determine their quality, we did extensive testing and observed that there was practically no benefit for values of $h > 2$. As expected, runtimes increased; however, the solution quality remained virtually the same. This comes from the fact that the $U^{(m+2)}$ bound is very tight so very little is gained when $h > 2$.

5 Exact Optimization Algorithm

We wish to solve MDTDP-CF. The main idea of our algorithm is to perform a search for the optimal d^* over the set $\{d^1, d^2, \ldots, d^{\tilde{r}}\}$ by taking advantage of Properties 3 - 5 given in Section 3.4. To this end, let MDTDP-CF(L, U) be a parameterized version of model (17)-(30) with corresponding objective function value $z(L, U)$, under the additional condition that the optimal value (of this related problem) must lie between L and U. Now, if it turns out that these fixed values of L and U are indeed valid bounds on d^* for the original problem, then we have $z(L, U) = d^*$. However, during the search process, it could be that one of these bounds is fixed at a value that makes MDTDP-CF(L, U) infeasible. This situation can be intelligently used to reduce the search space. Given L and U, Properties 3 - 5 can be used in a pre-processing step to fix and thus eliminate many binary variables.

To make the notation simpler, note that we can refer to each of the different distance values in $\{d^1, d^2, \ldots, d^{\tilde{r}}\}$ by their corresponding one-to-one mapped index set $\{1, 2, \ldots, \tilde{r}\}$. Let $r^{\min}$ and $r^{\max}$ be the indices corresponding to the distance values matching a known lower and upper bound, respectively, that is, $LB = d^{r^{\min}}$ and $UB = d^{r^{\max}}$. At the start of the algorithm, if these values are not known, they can simply be set to 1 and $\tilde{r}$, respectively. In our case, Algorithm 1 is applied to obtain an initial value for UB. Let $\alpha \in (0,1)$ be the *biased binary search parameter* used to calculate an intermediate point $\bar{r}$ between $r^{\min}$ and $r^{\max}$. Thus, the search algorithm is equivalent to finding the optimal index r^* such that $d^* = d^{r^*}$.

Algorithm 2 depicts our proposed exact optimization scheme. It can be viewed as a biased binary search parameterized on α. Typically, in a binary search algorithm α is set to 1/2. However, here we exploit the observation that the proposed upper bound is very tight, implying that the optimal solution is more likely to be closer to the upper bound than to the lower bound. Therefore, setting a high value of α, say, to 0.9, significantly improves the convergence of the algorithm. This is shown in the next section.

Algorithm 2 EXACT($r^{\min}$, $r^{\max}$, α)

```
Input: ($r^{\min}$, $r^{\max}$, $\alpha$)
Output: $r^* \equiv$ index corresponding to optimal solution $d^* = d^{r^*}$
 1: while ( $r^{\min} < r^{\max}$ ) do
 2:     $\bar{r} \leftarrow \lceil r^{\min} + \alpha(r^{\max} - r^{\min}) \rceil$
 3:     status $\leftarrow$ Solve MDTDP-CF($\bar{r}$, $r^{\max}$)
 4:     if ( status = Infeasible ) then
 5:         $r^{\max} \leftarrow \bar{r} - 1$
 6:     else if ( status = Optimal ) then
 7:         $r^* \leftarrow \arg_r\{d^r = d^*\}$
 8:         break
 9:     else { status = Feasible (non-optimal)}
10:         $r^{\min} \leftarrow \bar{r}$
11:     end if
12: end while
13: return $r^*$
```

In Step 3, *status* stores one of the three solver status values. Any code for solving an integer program can be used. In our case, we used CPLEX with its default settings. If the modified problem is infeasible then *status* = Infeasible, which means that the index corresponding to the optimal solution does not fall in the interval $[\bar{r}, r^{\max}]$. Otherwise, the modified problem is feasible and two possible cases can occur. Either the solution is proven optimal (*status* = Optimal) or just feasible (*status* = Feasible (non-optimal)). In the former case, the algorithm stops and this solution is returned as the optimal solution to the original MDTDP-CF. The latter case may occur when the algorithm solving MDTDP-CF(L, U) stops before reaching optimality. This could be due to a time limit being reached, for example, or when a feasible solution is obtained. Given a new feasible solution, the lower bound is updated and the algorithm goes on to the next iteration. Note that in Step 10, if a feasible (non-optimal) solution is found it means that $\bar{r}$ improves (is larger than) $r^{\min}$ by at least one unit given the way $\bar{r}$ was updated in the previous iteration (Step 2).

6 Computational Experiments

All procedures were coded in C++ and compiled with the C++ command line compiler (g++) from Xcode 8.3.3 and run on an iMac with a 3.33 GHz Intel Core 2 Duo processor and 8 GB of RAM. For solving the MILPs, we used the CPLEX Studio 12.7.1 callable library from IBM with its default settings.

For the empirical evaluations, we used two sets of problem instances denoted by R and F. For set R, we generated random instances as follows. BU coordinates were uniformly located in the $[0, 10] \times [0, 10]$ square and pairwise distances were calculated as Euclidean distances between points. Then, we followed the same procedure used by Fernández et al. (2010). That is, the infrastructure quality indices were uniformly chosen to ensure that we have approximately the same number of good, medium, and low quality BUs. The maximum number of split units allowed was set to $\sigma = \lfloor 0.2n \rfloor$ while the market shares were determined independently for the two types of products. First, a market share is drawn uniformly from the interval $[0.75/m, 1.25/m]$, and then normalized to ensure a total sum of 1. For our experiments, we divided all generated instances from set R into three main groups whose size we classified as medium, large, and very large. For the medium-size group, we used tolerance values of $\beta = 0.2$ and $\tau = 0.05$, for the other two groups we used $\beta = 0.05$ and $\tau = 0.05$.

Set F was derived from real-world data by Fernández et al. (2010) and corresponds to German zip-code areas with a proportional number of households. To generate an instance, they extracted all zip-code areas lying within a rectangle. The instances represent rural, urban as well as mixed regions. They generated five instances for each number of BUs, except for the last, where they have just four instances. For our experiments, we used only the largest instances, that is, instances with 200, 250, and 300 basic units. The tolerance values were $\beta = 0.2$ and $\tau = 0.05$. All instances from set R used in the study are available from the authors.

Table 1 identifies the size of the data sets in terms of n and m for each group. For set R, a total of 10 instances were generated for each $n \times m$ combination. For set F, there are 5 instances for each

Table 1: Data set size

Group	Instance size ($n \times m$)			
Set R	40×3	40×4	40×5	
Medium	60×3	60×4	60×5	
Total = 120	80×3	80×4	80×5	
	100×4	100×5	100×6	
Set R	200×4	200×5	200×6	
Large	250×4	250×5	250×6	
Total = 150	300×4	300×5	300×6	
	350×4	350×5	350×6	
	400×4	400×5	400×6	
Set R	500×5	500×8	500×10	500×12
Very large	600×5	600×8	600×10	600×12
Total = 150	700×5	700×8	700×10	700×12
	800×5	800×8	800×10	800×12
Set R	1000×8			
Extra large	1200×8			
Total = 30	1400×8			
Set F	200×4	200×5	200×6	200×7
Large	250×4	250×5	250×6	250×7
Total = 56	300×4	300×5	300×6	300×7

$n \times m$ combination except for the 300-unit instances which contain 4 instances per combination, for a total of 56 instances.

6.1 Comparing the Upper Bounds

In this set of experiments our goal was to compare the upper bounds U^{FKN} and U^{RB} when used to solve model MDTDP (1) - (16). A 60-minute time limit was placed on the computations.

In our first test we solve model MDTDP using each bound separately for the medium-size instances. Results are shown in Table 2. The first column displays the instance size, the following two columns show the average and maximum, respectively, relative improvement of upper bound U^{RB} over U^{FKN} computed as $100 \times (U^{\mathrm{FKN}} - U^{\mathrm{RB}})/U^{\mathrm{FKN}}$. This statistic represents the direct quality comparison between the bounds. It is important to mention that, in the computation of U^{FKN}, Fernández et al. (2013) added an improvement step by performing an additional local search. We have included this step in our work when computing their bound, U^{FKN}. For computing our bound this step is omitted.

The next two columns show the average and maximum, respectively, relative speed-up achieved when using bound U^{RB} for solving MDTDP compared to using bound U^{FKN}. This is calculated as $100 \times [time(U^{\mathrm{FKN}}) - time(U^{\mathrm{RB}})]/time(U^{\mathrm{FKN}})$, where $time(B)$ is the runtime for solving model MDTDP under bound B. The average speed-up is based on the individual speed-ups for each instance. The last two columns show the direct average runtimes for solving model MDTDP under U^{FKN} and U^{LB}, respectively.

Table 2: Comparison between U^{FKN} and U^{RB} for medium-size instances of model MDTDP

Instance size	Relative bound improvement (%) Avg	Max	Solution speed-up (%) Avg	Max	Average solution time (sec) U^{FKN}	U^{RB}
40×3	0.0	0.0	0.0	0	0.1	0.1
40×4	2.0	11.5	13.2	82.2	1.5	1.0
40×5	0.0	0.0	0.0	0.0	94.1	92.7
60×3	0.0	0.0	0.0	0.0	0.2	0.2
60×4	0.6	5.5	8.1	61.2	0.5	0.4
60×5	0.0	0.0	0.0	0.0	5.1	5.1
80×3	0.0	0.0	0.0	0.0	0.2	0.2
80×4	0.0	0.0	0.0	0.0	0.6	0.6
80×5	0.8	7.9	2.4	17.0	1.8	1.7
100×4	0.6	5.3	3.5	72.8	0.8	0.6
100×5	1.4	8.9	23.1	89.0	4.1	1.4
100×6	0.4	4.2	11.1	97.8	14.3	3.9

The first observation is that the value of the proposed bound was always the same or better than the previous bound. For most of the instances, the value of both bounds was the same before optimization. For those instances where this was not the case, the new bound was always better, showing relative improvements of up to 11.5% in the best case, and yielding speed-ups of up to 97.8% after optimization. In terms of achieving optimality, all 120 instances were solved optimally using either bound. The best results for the new bound were obtained for the largest subset of instances ($n = 100$) where the average time speed-up ranged from 3.5 to 23.1% for the new bound compared to the previous bound .

In our next experiment, we performed the same tests but for the large-size instances. The results are presented in Table 3. Again, it was observed that for most of the instances both bounds were the same. Nevertheless, as we can see, U^{RB} still outperforms U^{FKN}. In terms of achieving optimality, all but one optimal solution was found under either bound. The average running time was lower under the new bound achieving average speed-ups of up to 24.3% and a maximum speed-up of 96.9%. This can be attributed to the number of binary variables that can be fixed and thus eliminated from the model during pre-processing. The more the number of variables that can be fixed the better the quality of the bound. For the remaining experiments bound U^{RB} will be used in all computations.

Table 3: Comparison between U^{FKN} and U^{RB} for large-size instances of model MDTDP

Instance size	Relative bound improvement (%) Avg	Max	Solution speed-up (%) Avg	Max	Average solution time (sec) U^{FKN}	U^{RB}
200 × 4	0.0	0.0	0.0	0.0	1.8	1.7
200 × 5	1.8	18.0	5.7	43.4	7.3	7.0
200 × 6	1.2	9.3	18.2	96.9	62.0	14.4
250 × 5	1.0	6.0	9.9	60.6	4.9	4.3
250 × 6	0.5	4.9	11.4	91.2	30.2	10.8
250 × 7	0.5	4.7	10.9	90.9	76.9	56.8
300 × 5	0.4	3.7	6.0	50.5	4.5	4.0
300 × 6	0.7	3.8	21.9	94.4	39.2	13.4
300 × 7	0.0	0.0	0.0	0.0	451.0	451.0
350 × 5	1.4	9.1	9.4	95.1	22.4	4.9
350 × 6	0.2	1.7	10.2	93.1	33.8	16.6
350 × 7	1.6	10.4	24.3	93.3	254.6	131.6
400 × 5	0.9	8.9	5.4	51.5	6.0	5.4
400 × 6	0.2	1.8	3.2	46.3	26.6	23.3
400 × 7	0.0	0.0	0.0	0.0	121.2	117.9

6.2 Comparing the MILP formulations

The purpose of this experiment is to evaluate the relative performance of models MDTDP and MDTDP-CF. Again, a time limit of 60 minutes was set for CPLEX Tables 4 and 5 show the results for the large-size and very large-size instances, respectively. In Table 4, the first two columns indicate instance size and number of instances tried per size. The next two columns show the number of optimal solutions found and CPU runtime (seconds), for model MDTDP. Columns 5 and 6 show the same statistics for model MDTDP-CF. The last column gives the average time speed-up achieved by model MDTDP-CF when compared to MDTDP. Table 5 displays the same statistics with the addition of a Gap column that reports the average relative optimality gap. Note that this gap, computed as before, is taken only over those unsolvable instances. Thus, when all instances were solved for a particular size, a hyphen (-) is displayed. This is also true for the Gap columns in Tables 6 - 8.

For the large-size data set, the first observation is that it was possible to obtain optimal solutions to practically all instances tested. Model MDTDP failed in just one out of 150 instances whereas model MDTDP-CF found optimal solutions for all cases. Nevertheless, model MDTDP-CF outperformed MDTDP with respect to runtimes, demonstrating significant speed-ups. For the 200-unit instances the difference between the models is small; however, for the 250- to 400-unit instances the runtimes under MDTDP-CF were 8 to 45% faster.

For the very large-size instances presented in Table 5, the first observation is that obtaining

Table 4: Comparison of models on large-size instances

Instance size	Rep	MDTDP Opt	MDTDP Time (sec)	MDTDP-CF Opt	MDTDP-CF Time (sec)	Time speed-up (%)
200×4	10	10	1.74	10	1.68	11
200×5	10	10	6.98	10	10.31	-10
200×6	10	10	14.39	10	11.13	1
250×5	10	10	4.32	10	2.65	33
250×6	10	10	10.75	10	7.91	21
250×7	10	10	56.75	10	44.64	20
300×5	10	10	4.04	10	2.85	25
300×6	10	10	13.35	10	11.76	8
300×7	10	9	451.02	10	196.87	45
350×5	10	10	4.93	10	3.88	19
350×6	10	10	16.60	10	9.32	28
350×7	10	10	131.58	10	64.14	40
400×5	10	10	5.43	10	3.86	28
400×6	10	10	23.28	10	12.12	37
400×7	10	10	121.16	10	60.03	39
Totals	150	149		150		23

optimal solutions for either model becomes a bit harder but the covering formulation still outperforms the traditional formulation. For the 5-territory instances, both models obtained optimal solutions to all instances tested; however, under model MDTDP-CF, we observed average speed-ups between 11 and 32%. For the 8-territory instances, 39 out of 40 and 40 out of 40 optimal solutions were found under models MDTDP and MDTDP-CF, respectively. Moreover, model MDTDP-CF demonstrated average speed-ups ranging form 35 to 54% with respect to model MDTDP. For the 10-territory instances, model MDTDP obtained 30 out of 40 optimal solutions whereas model MDTDP-CF obtained 38 out of 40. Finally, for the 12-territory instances, model MDTDP obtained just 2 out of 40 optimal solutions whereas model MDTDP-CF obtained 10 out of 40.

For those instances that were not solved, huge optimality gaps were observed. This was due to the fact that the best feasible solution found within the allotted time limit of 1 hour was relatively close to zero. Therefore, when computing the relative gaps a very high ratio is obtained. This behavior was primarily observed for the instances with $m = 12$.

For the last group, the time speed-ups reported in the last column in Table 5 have to be taken with care. Given that most of the instances were not solved (taking all 3600 seconds allotted) these speed-ups may not be representative. The total average speed-up for all instances was 33%, whereas the average increased to 40% when the 12-territory instances were not taken into account. In total, model MDTDP optimally solved 111 out of 160 instances while model MDTDP-CF optimally solved 128 out of 160, most in significantly less time. The comparative advantage of model MDTDP-CF

Table 5: Comparison of models on very large-size instances

Instance		MDTDP			MDTDP-CF			Time
size	Rep	Opt	Gap	Time (sec)	Opt	Gap	Time (sec)	speed-up (%)
500×5	10	10	-	7.99	10	-	5.86	26
600×5	10	10	-	9.04	10	-	7.97	11
700×5	10	10	-	11.91	10	-	8.25	32
800×5	10	10	-	11.49	10	-	9.12	20
500×8	10	10	-	348.46	10	-	123.27	35
600×8	10	10	-	327.26	10	-	138.61	43
700×8	10	10	-	425.61	10	-	156.29	46
800×8	10	9	1.9	720.66	10	-	290.24	54
500×10	10	9	23.5	2026.96	10	-	854.15	58
600×10	10	7	2570.0	1796.64	9	2.6	995.42	49
700×10	10	7	795.0	1969.10	10	-	784.19	56
800×10	10	7	190.0	2197.55	9	7.4	1192.55	48
500×12	10	0	6210.0	3600.00	2	1050.0	3296.68	8
600×12	10	0	6250.0	3600.00	1	2170.0	3316.83	8
700×12	10	1	3190.0	3329.01	5	814.0	2784.35	19
800×12	10	1	4150.0	3529.18	2	599.0	3164.54	12
Totals	160	111			128			33*

*When the 12-territory instances are ignored, the average time speed-up is 40%

can be directly attributed to the sharp reduction in the number of binary variables achieved during pre-processing.

6.3 Evaluation of the Exact Method

In this experiment, the goal was to assess the relative performance of the proposed exact optimization algorithm when compared to solving model MDTDP-CF with CPLEX. For the very large-size instances, the exact algorithm was applied with the binary search parameter $\alpha = 0.95$. The results are displayed in Table 6. Again, the first two columns give the instance size and the number of replicates, respectively. The next three columns show the following statistics for CPLEX applied directly to model MDTDP-CF: number of optimal solutions found, the average relative optimality gap for the unsolved instances, and runtime (in seconds). The next four columns report the number of optimal solutions found, the average relative optimality for the unsolved instances, runtime (in seconds), and the average number of iterations it took to converge for the exact algorithm. The last column indicates the average percent speed-up achieved by the exact algorithm with respect to solving MDTDP-CF with CPLEX.

The results demonstrate both the efficiency of the exact method and the quality of the solutions it provides. All of the first 120 instances were solved with speed-ups ranging from 26 to 76 % with respect to CPLEX. For the 5- and 8-territory instances, both methods solved all 80 instances

Table 6: Assessment of exact method on very large-size instances.

Instance		MDTDP-CF			Exact Algorithm				Time
size	Rep	Opt	Gap	Time (sec)	Opt	Gap	Time (sec)	Iter	speed-up (%)
500×5	10	10	-	5.86	10	-	3.30	1.0	41
600×5	10	10	-	7.97	10	-	5.72	1.0	26
700×5	10	10	-	8.25	10	-	5.75	1.0	29
800×5	10	10	-	9.12	10	-	5.93	1.0	33
500×8	10	10	-	123.27	10	-	33.39	1.0	46
600×8	10	10	-	138.61	10	-	49.33	1.3	51
700×8	10	10	-	156.29	10	-	63.75	1.5	38
800×8	10	10	-	290.24	10	-	177.49	1.3	32
500×10	10	10	-	854.15	10	-	138.34	1.3	76
600×10	10	9	2.6	995.42	10	-	154.35	1.0	62
700×10	10	10	-	784.19	10	-	188.22	1.0	65
800×10	10	9	7.4	1192.55	10	-	280.22	1.2	61
500×12	10	2	1050.0	3296.68	10	-	1026.14	1.6	70
600×12	10	1	2170.0	3316.83	7	1332.5	2163.95	1.7	38
700×12	10	5	814.0	2784.35	9	51.0	1143.58	1.1	60
800×12	10	2	599.0	3164.54	7	1130.0	1885.82	2.2	43
Totals	160	128			153				

relatively fast. Once more districts are considered, however, a difference between the two emerges. For the 10-territory instances, the exact method solved all instances, whereas CPLEX failed on two. The speed-up achieved by the exact method was between 61 and 76%. For the 12-territory instances, the exact algorithm solved all but 7 instances, whereas CPLEX solved only 10 out of 40. For these instances, the speed-up achieved by the exact method ranged from 38 and 70%.

As in the previous experiment, for those instances that were not solved, huge optimality gaps were observed. Again, this was due to the fact that the best feasible solution found within the allotted time limit of 1 hour was relatively close to zero. As a consequence, a very high ratio was obtained when computing the relative gaps. This behavior was notably evident for the instances with $m = 12$. To gain more insight into relative performance of the two approaches, we conducted several additional experiments in which the time limit was set to more than 24 hours. The results are reported below.

Our last observation is based on the statistics in the second-to-last column of Table 6, which shows that the average number of iterations needed by the exact algorithm is negligibly low. On average, it took less than 2.2 iterations to converge. This was due to the biased design of the binary search coupled with the fact that our upper bounding scheme provides tight results. Setting α to a high value means that a considerably greater number of binary variables can be fixed and eliminated in advance.

Finally, it is important to note that in the real world, instances have anywhere from 5 to 10 territories. The exact method performed exceedingly well on instances of this size, which should make it an attractive option for practitioners. For the more difficult 12-territory instances the exact algorithm showed its limitations. It was only able to solve 33 of the 40 instances.

Nevertheless, to get a clearer picture of algorithmic performance, we reran the 12-territory instances, but now with a 24-hour time limit. Again we compared models MDTDP, MDTDP-CF, and the exact method displaying the results in Table 7. Recall that the Gap column shows the relative optimality gap only over those unsolvable instances. Thus, when all instances were solved for a particular size, a dash (-) is displayed.

Table 7: Assessment of exact method on instances with $m = 12$ and time limit of 24 h.

Instance		MDTDP			MDTDP-CF			Exact Algorithm			
size	Rep	Opt	Gap	Time (h)	Opt	Gap	Time (h)	Opt	Gap	Time (h)	Iter
500×12	10	9	20.7	9.41	9	6.1	4.38	10	-	3.05	1.0
600×12	10	7	69.6	14.64	7	12.8	9.99	7	3.76	7.78	1.0
700×12	10	9	11.0	5.65	10	-	1.86	10	-	0.88	1.0
800×12	10	4	16.0	15.33	10	-	4.84	10	-	3.85	1.0
Totals	40	29			36			37			

The most striking observation is that for the 1-hour time limit in the previous experiments, only 2, 10 and 33 instances (with $m = 12$) out of 40 were optimally solved respectively by MDTDP, MDTDP-CF, and the exact method. By allowing runtimes up to 24 hours, 29, 36 and 37 optimal solutions were found by MDTDP, MDTDP-CF and the exact method, respectively. As a consequence, the relative optimality gaps were dramatically reduced. Again, the exact algorithm significantly outperformed CPLEX on the other two models. When comparing the results for MDTDP and MDTDP-CF, the statistics are consistent with the previous experiments; that is, the latter produces a significantly larger number of optimal solutions and provides a significant reduction in optimality gaps for those unsolved instances.

Now, when comparing MDTDP-CF with the exact method, we see that the latter is clearly better. Not only does it find an additional optimal solution but it also delivers solutions with measurably smaller optimality gaps for those few unsolved instances. This is also reflected in the runtimes, where the exact method takes significantly less time to converge – one iteration only for the solved instances.

In the last experiment in this subsection, we assessed both the MDTDP-CF model and the exact algorithm on instances with larger values of n. In particular, we tested instances with $n \in \{1000, 1200, 1400\}$ and $m = 8$ setting the time limit to 24 hours. The results are highlighted in Table 8

Table 8: Assessment of exact method on instances with $n \in \{1000, 1200, 1400\}$.

Instance		MDTDP-CF			Exact Algorithm				Time
size	Rep	Opt	Gap	Time (sec)	Opt	Gap	Time (sec)	Iter	speed-up (%)
1000×8	10	10	-	186.61	10	-	146.89	1.0	10.8
1200×8	10	10	-	248.62	10	-	253.72	1.0	6.6
1400×8	10	10	-	508.23	10	-	350.59	1.0	18.1
Totals	30	30			30				

The first observation is that all 30 instances were optimally solved with either approach so all optimality gaps are zero. Again, the exact method was faster showing average speed-ups from 6.6 to 18.1% with respect to the MDTDP-CF model. An additional observation is that problem difficulty is more affected by an increase in the number of companies m than by an increase in the number of basic units n.

6.4 Assessing the Models and Methods on Real-World Instances

In our last experiment, we assess the models and methods using data set F of real-world instances. To this end, we used both CPLEX and our exact method to solve models MDTDP and MDTDP-CF (under bound U^{RB}). The results are displayed in Table 9.

Table 9: Assessment of models and methods on real-world instances

		MDTDP		MDTDP-CF			Exact algorithm			
Instance			Time		Time	Speed-up		Time		Speed-up
size	Rep	Opt	(sec)	Opt	(sec)	(%)	Opt	(sec)	Iter	(%)
200×4	5	5	1.16	5	0.87	24.9	5	0.44	1.0	49.0
200×5	5	5	2.66	5	1.68	36.9	5	0.91	1.0	45.3
200×6	5	5	36.09	5	27.92	22.7	5	20.16	2.6	46.4
200×7	5	5	74.81	5	29.03	61.2	5	17.84	3.6	42.8
250×4	5	5	3.71	5	3.23	13.0	5	9.12	5.2	-99.6
250×5	5	5	4.66	5	2.63	43.6	5	1.93	1.0	28.4
250×6	5	5	45.26	5	51.67	-14.2	5	52.88	17.8	-15.9
250×7	5	5	41.03	5	25.28	38.4	5	7.23	1.0	54.9
300×4	4	4	1.21	4	1.16	4.7	4	0.62	1.0	46.3
300×5	4	4	2.63	4	2.37	9.8	4	0.90	1.0	62.9
300×6	4	4	14.02	4	10.40	25.8	4	3.43	1.2	52.2
300×7	4	4	36.75	4	29.11	20.8	4	8.52	1.5	54.9
Totals	56	56		56			56			

The first two columns indicate instance size and number of instances tested for each combination. The next four columns give the number of proven optimal solutions found and average runtime when CPLEX was used for solving MDTDP (under bounds U^{RB}) and MDTDP-CF, respectively. The seventh column reports the relative speed-up obtained from MDTDP-CF with respect to MDTDP. A negative value means MDTDP-CF was slower. The following three columns display the results for the number of optimal solutions found, the average runtime, and the average number of iterations required, all for the exact algorithm. The last column gives the relative speed-up obtained from the exact algorithm with respect to MDTDP-CF.

The first observation is that all 56 instances were optimally solved by all approaches. When comparing the two models, we see that MDTDF-CF runs faster then MDTDP for practically every size combination showing speed-ups of up to 61.2% (for the 200×7 instances). The only exception occurred for the 250×6 instances, where the average time taken by MDTDP-CF was 14% higher. A closer look reveals that this result was due to one difficult instance in the (250×6) data set, so MDTDP-CF was faster in 4 out of 5 of those instances. Again, this is consistent with previous results.

Finally, the exact method showed even lower average runtimes than MDTDP-CF for almost all instances tested, except for the 250×4 and 250×6 cases. This was in fact due to two particular instances taking an unusually large number of iterations to converge. While the exact method reliably converged in less than 4 iterations, 14 and 74 iterations were respectively required for the two instances, which led to higher average runtimes. Nevertheless, for the remaining 54 (out of 56 instances), the exact method was significantly faster showing speed-ups of up to 72% (for the 300×7 instances). For the most part, these results are also consistent with those from the previous experiments.

7 Summary and Conclusions

In this paper, we addressed a maximum dispersion territory design problem arising in WEEE collection. To begin, we introduced a new problem reformulation based on covering-type variables that provides a tighter linear programming relaxation than the standard formulation. Next, we developed a new upper bound that was seen to offer a considerable improvement over the best existing bound. Computational testing showed that the new model is faster to solve than the existing model when commercial software is used.

The most important contribution, though, has been the development of an exact optimization algorithm based on the integration of the new upper bound and the covering-based reformulation. The exact algorithm uses a biased binary search as it iterates towards optimality. Testing confirms that it can find optimal solutions to problem instances with up to 800 basic units and 12 companies, and problem instances with up to 1400 basic units and 8 companies. Previous to this research, the largest instances solved optimally had between 40 to 100 basic units and 4 companies. This represents a significant advance in the state of the art.

With respect to future research, several promising paths exist for tackling an expanded version of MaxD-TDP. In this paper, we basically studied a district design problem absent its routing and collection components. At present, the design and operational problems are handled separately. A more comprehensive approach would be to combine the two in an integrated model where both the design and routing decisions were taken simultaneously. Of course, this new problem would be more complex but the procedures developed here should prove useful when attempting to develop, say, decomposition-based algorithms to solve it.

Acknowledgments: We are grateful to three anonymous referees and the associate editor whose remarks helped improve the quality of the paper. The work of the first author was supported by the Mexican Council for Science and Technology (CONACyT) through grant CB11-1-166397 and by Universidad Autónoma de Nuevo Leoón through grant UANL-PAICYT CE331-15.

References

Church, R. L., 2008. BEAMR: An exact and approximate model for the p-median problem. Computers & Operations Research 35 (2), 417–426.

Duque, J. C., Ramos, R., Suriñach, J., 2007. Supervised regionalization methods: A survey. International Regional Science Review 30 (3), 195–220.

Elloumi, S., Labbé, M., Pochet, Y., 2004. A new formulation and resolution method for the p-center problem. INFORMS Journal on Computing 16 (1), 84–94.

Erkut, E., 1990. The discrete p-dispersion problem. European Journal of Operational Research 46 (1), 48–60.

Erkut, E., Neuman, S., 1991. Comparison of four models for dispersing facilities. INFOR 29 (2), 68–86.

Fan, Z. P., Chen, Y., Ma, J., Zeng, S., 2011. A hybrid genetic algorithmic approach to the maximally diverse grouping problem. Journal of the Operational Research Society 62 (1), 92–99.

Fernández, E., Kalcsics, J., Nickel, S., 2013. The maximum dispersion problem. Omega 41 (4), 721–730.

Fernández, E., Kalcsics, J., Nickel, S., Ríos-Mercado, R. Z., 2010. A novel maximum dispersion territory design model arising in the implementation of the WEEE-directive. Journal of the Operational Research Society 61 (3), 503–514.

Gallego, M., Laguna, M., Martí, R., Duarte, A., 2013. Tabu search with strategic oscillation for the maximally diverse grouping problem. Journal of the Operational Research Society 64 (5), 724–734.

Georgiadis, P., Besiou, M., 2010. Environmental and economical sustainability of WEEE closed-loop supply chains with recycling: A system dynamics analysis. International Journal of Advanced Manufacturing Technology 47 (5–8), 475–493.

Grunow, M., Gobbi, C., 2009. Designing the reverse network for WEEE in Denmark. CIRP Annals 58 (1), 391–394.

Hammond, D., Beullens, P., 2007. Closed-loop supply chain network equilibrium under legislation. European Journal of Operational Research 183 (2), 895–908.

Kalcsics, J., 2015. Districting problems. In: Laporte, G., Nickel, S., Saldanha da Gama, F. (Eds.), Location Science. Springer, Cham, Switzerland, Ch. 23, pp. 595–622.

Kuo, C.-C., Glover, F., Dhir, K. S., 1993. Analyzing and modeling the maximum diversity problem by zero-one programming. Decision Sciences 24 (6), 1171–1185.

Lee, S. C., Shih, L. H., 2012. A novel heuristic approach to determine compromise management for end-of-life electronic products. Journal of the Operational Research Society 63 (5), 606–619.

Mar-Ortiz, J., Adenso-Diaz, B., González-Velarde, J. L., 2011. Design of a recovery network for WEEE collection: The case of Galicia, Spain. Journal of the Operational Research Society 62 (8), 1471–1484.

Mar-Ortiz, J., González-Velarde, J. L., Adenso-Díaz, B. J., 2013. Designing routes for WEEE collection: The vehicle routing problem with split loads and date windows. Journal of Heuristics 19 (2), 103–127.

Marín, A., Nickel, S., Puerto, J., Velten, S., 2009. A flexible model and efficient solution strategies for discrete location problems. Discrete Applied Mathematics 157 (5), 1128–1145.

Martí, R., Gallego, M., Duarte, A., 2010. A branch and bound algorithm for the maximum diversity problem. European Journal of Operational Research 200 (1), 36–44.

Pukelsheim, F., Ricca, F., Simeone, B., Scozzari, A., Serafini, P., 2012. Network flow methods for electoral systems. Networks 59 (1), 73–88.

Queiruga, D., Walther, G., González-Benito, J., Spengler, T., 2008. Evaluation of sites for the location of WEEE recycling plants in Spain. Waste Management 28 (1), 181–190.

Ricca, F., Scozzari, A., Simeone, B., 2013. Political districting: From classical models to recent approaches. Annals of Operations Research 204 (1), 271–299.

Ríos-Mercado, R. Z., Maldonado-Flores, J. R., González-Velarde, J. L., 2017. Tabu search with strategic oscillation for improving recollection assignment plans of waste electric and electronic equipment. Technical Report PISIS–2017–01, Graduate Program in Systems Engineering, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Mexico.

Rudăreanu, C., 2013. Waste electrical and electronic equipment (WEEE) management in Europe. Economics, Management, and Financial Markets 8 (3), 119–125.

Tsai, W.-H., Hung, S.-J., 2009. Treatment and recycling system optimisation with activity-based costing in WEEE reverse logistics management: An environmental supply chain perspective. International Journal of Production Research 47 (19), 5391–5420.

Zoltners, A. A., Sinha, P., 2005. Sales territory design: Thirty years of modeling and implementation. Marketing Science 24 (3), 313–331.