






An Effective VNS for Delivery Districting

Ahmed Aly¹ , Adriana F. Gabor^{2,4} , and Nenad Mladenovic^{3,4} 

¹ AHOY DMCC: Dubai, Dubai, UAE

ahmed.oss.aly@gmail.com

² Applied Mathematics, Khalifa University, Abu Dhabi, UAE

adriana.gabor@ku.ac.ae

³ Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi, UAE

nenad.mladenovic@ku.ac.ae

⁴ Research Center on Digital Supply Chain and Operations Management, Khalifa University, Abu Dhabi, UAE

Abstract. This paper deals with the Delivery Territory Design Problem (DTDP), in which n points have to be allocated to p territories, such that balancing and path connectivity requirements are satisfied, while minimizing the maximum diameter over the created territories. The model is inspired by tactical planning situations faced by delivery companies. We propose two best improvement local search procedures and a Basic Variable Neighborhood Search algorithm following the LIMA paradigm. The results suggest that our algorithm is able to find high-quality solutions within a relatively low time.

Keywords: Territory design · Basic VNS · Less-is-more approach

1 Introduction

The territory design problem (*TDP*), sometimes referred to as the districting problem or territory alignment problem, is the problem of grouping small geographical units called basic units *BUs* into larger geographical clusters, named territories, according to relevant planning criteria. Typical applications of the territory design problem include sales territory design [1], political districting [2], school districting [3], and public services districting [4,5]. TDP and its variations have been researched since the 1960s [6], using a variety of models and algorithms. The territory design problem is NP-Hard [7] and thus metaheuristics were used in order to solve this problem. For state-of-the-art models, algorithms, and applications to the territory design problem we refer the reader to [8].

An essential criterion in the TDP is the compactness of districts. One way to achieve this is by minimizing a dispersion measure. A common dispersion measure used in classical problems such as p -median and p -center, is the distance to the centroid of the district. Using a center based measure has some limitations

A. Aly—The work of the first author was performed while being a research assistant at Khalifa University.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Sleptchenko et al. (Eds.): ICVNS 2022, LNCS 13863, pp. 69–81, 2023.

https://doi.org/10.1007/978-3-031-34500-5_6

for problems where it is not clear how to define possibly “good” centers. In such cases, a dispersion measure based on the diameter of a territory is more appropriate [9].

In this paper, we address a special version of the TDP, called the Delivery Territory Design Problem (DTDP). DTDP aims to design a set of p territories such that the maximum diameter of a territory is minimized, while satisfying several planning requirements such as disjoint territories, and balancing in terms of three attributes: driver workload, commodity demand, and number of customers. Unlike in the TDP, we allow two nodes to be assigned to the same territory as long as there exists a path between them, not necessarily fully contained in the district. The problem is motivated by real-world applications from delivery companies, where a driver can be assigned to serve two BUs as long as there is a street between them.

The rest of this paper is structured as follows; In Sect. 2, we discuss the related works to the territory design problems and their applications. In Sect. 3, we describe the problem and provide the mathematical model of the TDP, while in Sect. 4 we outline the VNS heuristic and its components. We present computational experiments and results in Sect. 5 and conclude our findings in Sect. 6.

2 Related Work

The problem studied in this paper is related to the commercial territory design problem (*CTDP*) which was proposed by Rios-Mercado and Escalante [7]. *CTDP* seeks to maximize a compactness criterion of p territories subject to planning criteria such as disjoint districts, attribute balancing, and district connectivity. In their paper, compactness is measured by the distance of a node from the center it is assigned to. The authors propose a GRASP algorithm consisting of three phases: construction, adjustment, and local search. The algorithm produces good quality solutions, however that comes at a high computational cost. Rios-Mercado et. al. [9] expanded upon the *CTDP* with a new model that makes use of a diameter-based dispersion measure instead of its center-based counterpart. They used the GRASP metaheuristic in combination with path relinking to solve instances of 500 nodes and $p = 10$ districts. The algorithm provides good results in terms of the dispersion measure however they are computationally expensive.

In this paper, we propose to solve the DTDP by a Variable Neighborhood Search (*VNS*) algorithm. The core paradigm behind VNS is to systematically change neighborhood structures to prevent plateaus at local optima [10]. Over the years, VNS has been extensively researched and now boasts a wide array of extensions [11]; General VNS, Variable Neighborhood Descent, and Reduced VNS to name a few.

VNS has been successfully used to solve related problems to TDP. Mladenovic et al. [12] proposed a Basic VNS metaheuristic with vertex substitution local search to solve the p -center problem. Their results show that VNS, on average, outperforms Tabu Search, whereas Tabu Search is better for a small p .

Hindi and Fleszar [13] proposed to solve the capacitated p -median problem by a VNS metaheuristic based on the generalized assignment problem. They test their results on five standardized sets of benchmark instances and show that the proposed heuristic finds the best known solutions as well as it improves a previously best-known solution.

Brimberg et. al. [14] showed that Skewed General VNS performs well for the capacitated clustering problem. They showed that evaluating moves prior to accepting inferior solutions is preferable to random shaking procedures. The authors tested the algorithm on the largest set of instances (MDG2000). The Skewed General VNS showed to be the fastest procedure out of the tested metaheuristics with up to 1.55% improvement over other metaheuristics.

Mladenovic et. al. successfully used the Basic VNS to solve the obnoxious p -median problem in the Less-is-more approach (LIMA) [15]. They proposed a simple facility best improvement local search that lies in between the first and best improvement strategies. They found new best solutions for four instances and ties with 133 instances out of a set of 144 benchmark instances.

Contribution: In this work we propose a VNS based meta-heuristic for solving the DTDP problem, with relaxed connectivity criteria. To the best of our knowledge, this technique has not been previously applied to this problem. Via numerical experiments, we show that the VNS procedure outperforms the algorithm of [9] by 6.35% on average.

3 Problem Description and Mathematical Model

The input to the DTDP is a graph $G = (V, E)$, where the nodes are the set of basic units (BU) and the edges represent the streets between BUs. For each node, we are given a set of attributes A such as number of customers, product demand, and workload. The value of attribute $a \in A$ of BU $i \in V$ will be denoted by w_a^i .

We denote a p -partition of the set V by $X = (X_1, \dots, X_p)$ where $X_m \subset V$ is called a territory of V . The size of the territory X_m with respect to attribute $a \in A$ is denoted by $w^a(X_m) = \sum_{i \in X_m} w_a^i$.

We call a partition X *balanced* w.r.t an attribute a , if the size of each territory X_m in X satisfies $\frac{w^a(X_m)}{\mu^a} \in [1 - \tau^a, 1 + \tau^a]$, where μ^a is the average of attribute a over all the nodes. Here, τ^a is a tolerance parameter that is prespecified by the user.

The goal of DTDP is to find a balanced p - partition w.r.t. each attribute, that minimizes the maximum diameter over the territories created. Moreover, we require that any two BUs in a territory are connected by a path in G .

We denote the collection of all the p -partitions of the set V by Π . The TDTP can be formulated as a combinatorial optimization as follows:

$$\min_{X \in \Pi} \max_{m \in M} \max_{i, j \in X_m} \{d_{ij}\} \quad (1)$$

s. t.

$$\frac{w^a(X_m)}{\mu^a} \in [1 - \tau^a, 1 + \tau^a] \quad m \in M, a \in A \quad (2)$$

$$G_m = G(V_m, E) \text{ is connected} \quad m \in M \quad (3)$$

The formulation in this paper is based on the mathematical model outlined for the CTDP (see Rios-Mercado and Escalante [9]). The objective function (1) minimizes the maximum diameter in a p -partition X . Constraints (2) requires that the p -partition should be balanced in each attribute $a \in A$. Constraints (3) stipulate that each node of a district must be connected by a path in graph G .

4 Variable Neighborhood Search Procedure

We propose to solve the DTDP by a Basic VNS (BVNS) procedure, in the spirit of the LIMA paradigm [15]. One of the reasons the BVNS is used is that it does not require high computational resources and provides high-quality solutions. Using this variant, we are able to diversify the solution through random neighborhood structures and intensify it through the deterministic neighborhood structures. By doing so, we are able to avoid plateauing at local optima.

The next subsections will outline the construction of the initial solution, the Basic VNS procedure, the local search variants, and the shaking procedure.

4.1 Initial Solution

To generate the initial solution, we use the construction phase of the GRASP algorithm outlined in [9] once. The construction phase starts with a set of p randomly chosen seeds in V . The algorithm then greedily assigns nodes $i \in V$ to the p seeds while attempting to maintain the balancing criteria. If it is not possible to maintain the balancing constraints, the unassigned nodes are allocated to the closest seed. We denote the solution obtained by X_{in} .

4.2 Basic Variable Neighborhood Search

Consider a p -partition $X = (X_1, \dots, X_p)$. We define a *neighborhood* $S_k(X)$ as the set of solutions obtained by reallocating k nodes from a territory X_{m_1} to a territory X_{m_2} .

To evaluate the quality of a solution $X = \{X_1, \dots, X_p\}$, the VNS procedure uses the function $\Psi(X)$ introduced in [9] and defined as a linear combination between a function related to the maximum diameter and a measure of the infeasibility of X . More precisely

$$\Psi(X) = \lambda F(X) + (1 - \lambda)G(X),$$

where

$$F(X) = \left(\frac{1}{d_{max}} \right) \max_{m \in M} \max_{i, j \in X_m} \{d_{ij}\},$$

$$d_{max} = \max_{i, j \in V} \{d_{ij}\},$$

and

$$G(X) = \sum_{m=1}^p \sum_{a \in A} g^a(X_m),$$

where,

$$g^a(X_m) = \frac{1}{\mu^a} \max\{w^a(X_m) - (1 + \tau^a)\mu^a, (1 - \tau^a)\mu^a - w^a(X_m), 0\}.$$

Furthermore, λ is a user specified parameter that controls whether the objective function or the infeasibility is more favored in the cost function calculation.

A general outline of the Basic Variable Neighborhood Search (BVNS) procedure is given in Algorithm 1. The BVNS uses a shake procedure and two local search procedures, called LS-NBI and LS-DBI, that will be described in Sect. 4.3.

Algorithm 1. BVNS($X_{in}, k_{max}, \beta_{max}$)

```

1:  $\beta \leftarrow 1$ 
2:  $X \leftarrow X_{in}$ 
3: while  $\beta \leq \beta_{max}$  do
4:    $k \leftarrow 1$ 
5:   while  $k \leq k_{max}$  do
6:      $X' \leftarrow \text{Shake}(X, k)$ 
7:      $X'' \leftarrow \text{LS-NBI}(X')$ 
8:     if  $\Psi(X'') < \Psi(X)$  then
9:        $k \leftarrow 1$ 
10:       $X \leftarrow X''$ 
11:     else
12:        $k \leftarrow k + 1$ 
13:     end if
14:   end while
15: end while
16:  $X \leftarrow \text{LS-DBI}(X)$ 

```

The BVNS takes in as input the initial solution X_{in} , the maximum number of neighborhoods used in the shaking procedure k_{max} , and the maximum number of repetitions β_{max} . While the number of repetitions is not reached, the algorithm executes a shake procedure followed by the the local search procedure LS-NBI (lines 6–7). If the value of Ψ is improved, the algorithm performs a sequential neighborhood change step (lines 8–13). Lines (14–15) lead to repeating the BVNS procedure if k_{max} is reached. Finally line (16) applies the second local search variant, LS-DBI.

4.3 Local Search and Shaking Procedures

The BVNS uses two best improvement local search procedures: Node Best Improvement Local Search (LS-NBI) and District Best Improvement Local Search (LS-DBI).

In both procedures, a $move(m, i)$ is defined as re-allocating a node $i \in V \setminus \{X_m\}$ to territory X_m .

Node Best Improvement Local Search (LS-NBI). A pseudo-code for Node Best Improvement (LS-NBI) is given in Algorithm 2. The LS-NBI procedure iterates over all the territories of a solution X . For each territory m , $move(m, i)$, $i \in V \setminus \{X_m\}$ that leads to the best improvement in Ψ is performed (lines 5–11). LS-NBI terminates either when the maximum number of moves is reached (line 3) or no improved solution is found (lines 12–18).

Algorithm 2. LS-NBI(X)

```

1:  $nmoves \leftarrow 0$ 
2:  $optima \leftarrow False$ 
3: while  $nmoves < max\_moves$  and  $optima = False$  do
4:    $improvement \leftarrow False$ 
5:   for all  $m \in \{1, \dots, p\}$  do
6:     Find  $move(m, i)$  that leads to best improvement of  $\Psi$ 
7:     if  $\Psi$  is improved then
8:       Perform  $move(m, i)$ 
9:        $improvement \leftarrow True$ 
10:    end if
11:  end for
12:  if  $improvement \leftarrow True$  then
13:     $nmoves = nmoves + 1$ 
14:     $optima = False$ 
15:  else
16:     $optima = True$ 
17:  end if
18: end while

```

District Best Improvement Local Search (LS-DBI). Algorithm District Best Improvement (LS-DBI) is similar to LS-NBI with the main difference being that it iterates over all nodes $i \in V$ instead of territories. For each node i , it performs the best $move(m, i)$, where district m is such that $i \notin X_m$. The procedure terminates either when the maximum number of moves is reached or when no improved solution is found. We refer to Algorithm 3 for the detailed pseudo-code.

Algorithm 3. LS-DBI(X)

```

1:  $nmoves \leftarrow 0$ 
2:  $optima \leftarrow False$ 
3: while  $nmoves < max\_moves$  and  $optima = False$  do
4:    $improvement \leftarrow False$ 
5:   for all  $i \in V$  do
6:     Find  $move(m, i)$  that leads to best improvement of  $\Psi$ 
7:     if  $\Psi$  is improved then
8:       Perform  $move(m, i)$ 
9:        $improvement \leftarrow True$ 
10:    end if
11:  end for
12:  if  $improvement \leftarrow True$  then
13:     $nmoves = nmoves + 1$ 
14:     $optima = False$ 
15:  else
16:     $optima = True$ 
17:  end if
18: end while

```

Shaking Procedure. The shake procedures serves to diversify the search space. $Shake(X, k)$ chooses two random territories X_1, X_2 of the current solution X and moves k random nodes from X_1 to X_2 .

Algorithm 4. Shake(X, k)

```

1: Choose two random districts  $X_1, X_2 \in X$ 
2: Choose a set  $K$  of random nodes in  $X_1, |K| = k$ 
3: Remove  $K$  from  $X_1$  and re-allocate the nodes in  $K$  to  $X_2$ 

```

5 Computational Experiments

In this section, we present the results of the numerical experiments we have performed in order to test the proposed algorithms. The algorithms discussed in this section were coded in Python 3.9, and all of the experiments were run on Intel® Xeon X5650 2.67GHz with 72GB RAM.

The computational experiments were performed on randomly generated planar graphs consisting of 500 nodes. We started with a grid graph of 30×30 nodes, divided into 7 regions as in Fig. 1. We generated three types of graphs; Graph Type Center, Graph Type Diagonal, and Graph Type Corners (G-C, G-D, G-CN) by randomly removing $900 - n$ nodes where $n = 500$ from certain regions while maintaining the connectivity of the nodes in the graph.

We remove $\lfloor \frac{3}{4}(900 - n) \rfloor$ nodes from the regions R3, R4, and R5 for G-C; R1, R4, and R7 for G-D; R1, R2, R6, and R7 for G-CN. Finally, we remove

$\lfloor \frac{1}{4}(900 - n) \rfloor$ from the remaining regions for each graph type. When a node is removed, its adjacent edges are also removed.

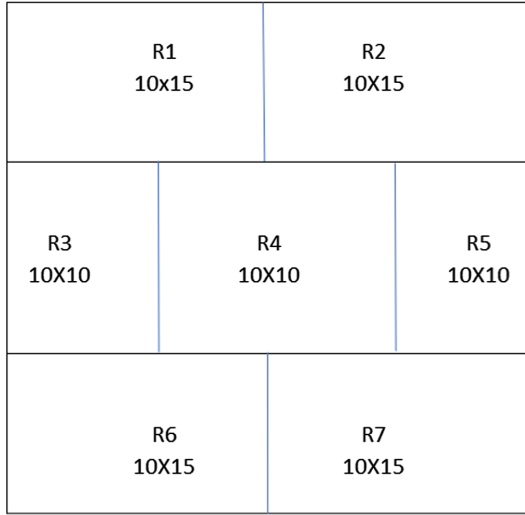


Fig. 1. Graphical representation of the regions of the graph.

Each node in a graph G has three attributes: demand, workload, and number of customers. The attributes were generated from a uniform distribution in the ranges of $[15, 400]$, $[15, 100]$, and $[4, 20]$ respectively [7]. Furthermore, to simulate real-world scenarios, we generated a distance attribute for edges from a uniform distribution in the range of $[5, 20]$. The value of the parameters was chosen as follows. For the cost function $\Psi(X)$, parameter λ was set to $\lambda = 0.7$. Furthermore, $p = 10$, $\beta_{max} = 5$, $m_{max} = 4$, and $max_moves = 100$.

5.1 Impact of Local Search on Initial Solution

In Table 1, the columns LS-NBI and LS-DBI refer to the algorithms in which the initial solutions are improved by applying the respective local search once. Furthermore, the column labeled with “GRASP-LS” refers to the local search procedure described in [9].

The average improvement of LS-NBI, LS-DBI, and GRASP in terms of the objective function value over the initial solutions was 4.00%, 4.32%, and 1.59% respectively. Table 1 shows the average percentage improvement of the local search variants on each graph type. We note that LS-DBI outperformed the rest of the variants over graph types G-CN and G-D at 5.26% and 3.81% respectively. While LS-NBI, outperformed the rest of the variants for graph type G-C at 4.72%.

Table 1. Local search procedures percentage improvement over the initial solution.

Graph		LS-NBI	LS-DBI	GRASP-LS
Graph-Type	Measure			
G-C	<i>Max</i>	14.29%	9.13%	5.34%
	<i>Average</i>	4.72%	3.88%	1.18%
	<i>Min</i>	0%	0%	0%
G-D	<i>Max</i>	9.53%	12.61%	7.16%
	<i>Average</i>	2.97%	3.81%	2.43%
	<i>Min</i>	0%	0%	0%
G-CN	<i>Max</i>	14.56%	14.56%	4.98%
	<i>Average</i>	4.31%	5.26%	1.15%
	<i>Min</i>	0%	0%	0%

We observe that the local search variants LS-NBI and LS-DBI have a different impact based on the graph type that the variant was performed on. Furthermore, we notice that both local search procedures had a high variance in graph type G-CN. This suggests that this particular graph type is difficult to improve upon.

We note that the local search variant LS-DBI outperformed LS-NBI and GRASP-LS for graph types G-D and G-CN at 3.81% and 5.26% respectively. On the other hand, LS-NBI outperformed all other variants in graph type G-C at 4.72%. Due to the different performance of both LS-NBI and LS-DBI based on the graph type, we used both local search variants in $BVNS(X, k_{max}, \beta_{max})$.

5.2 Computational Experiments on BVNS

We compared the results of the proposed BVNS algorithm on all graph types with the results of the static Path-Relinking (PR) algorithm presented in [9].

Figure 2 shows an example solution and a comparison between BVNS and PR. Figure 2(a) shows the BVNS solution of that particular instance while Fig. 2(b) shows the PR solution where each have ten distinct districts. We can see that due to the shaking procedure of the BVNS algorithm, the solution was able to escape local optima regardless of the graph structure and provide significant improvements.

Table 2. Objective Function Percentage Improvement of BVNS over PR.

Graph	Min	Max	Average
G-C	0.95%	18.62%	6.42%
G-D	0%	13.20%	5.21%
G-CN	0%	20.86%	7.42%
Average			6.35%

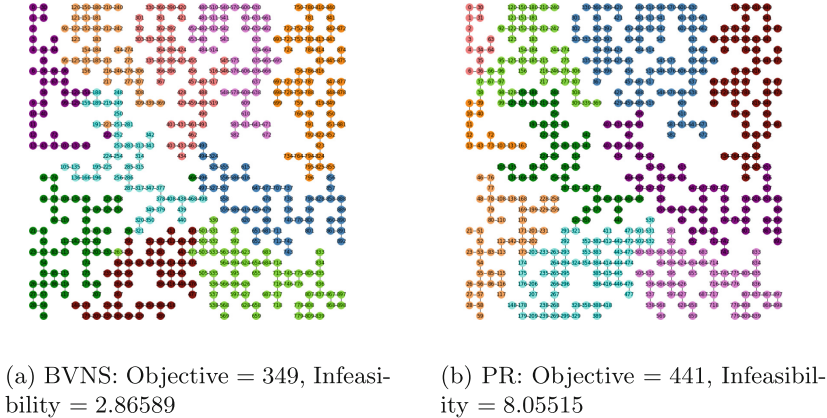


Fig. 2. Example comparison of BVNS and PR.

Table 2 summarizes the results of the objective function improvement of BVNS over PR. We show the minimum, maximum, and average improvement of BVNS over PR for each graph type. On average, BVNS outperforms PR by 6.35% over all graph types with a maximum improvement of 20.86% for G-CN. On average, BVNS outperformed PR by 6.42% for graph type G-C, 5.21% for graph type G-D, and by 7.42% for graph type G-CN. We note that BVNS outperformed PR for 27 out of the 30 graphs. PR outperformed BVNS for 3 out of the 30 graphs in terms of objective function value. We note that 2 out of the 3 graphs where PR outperformed BVNS occurred in graph type G-D.

Figure 3 shows the box plot of the percentage improvement of the relative infeasibility of BVNS over PR. We can see that the relative infeasibility of BVNS has shown consistent improvements over PR with the certain outliers in each graph type. This indicates that using the BVNS procedure with LS-NBI and LS-DBI led to improvements of the infeasibility of the solution compared to PR, on average by 21.7%. The relative infeasibility in graph type G-CN has been the most unstable where 3 out of the 6 instances show PR outperforming BVNS in terms of relative infeasibility.

Furthermore, Fig. 3 along with the results presented for the average objective function improvement, suggest that the graph type G-D is difficult to improve for both algorithms.

5.3 Running Times

The average running time of BVNS was 394.49s over all graph types. Among the local search procedures, LS-DBI was less time consuming, with an average of 13.17s, followed by LS-NBI with an average running time of 17.24s. Thus, the time difference between the two local search procedures is negligible and provides great benefit in tackling different types of graphs with considerable improvements.

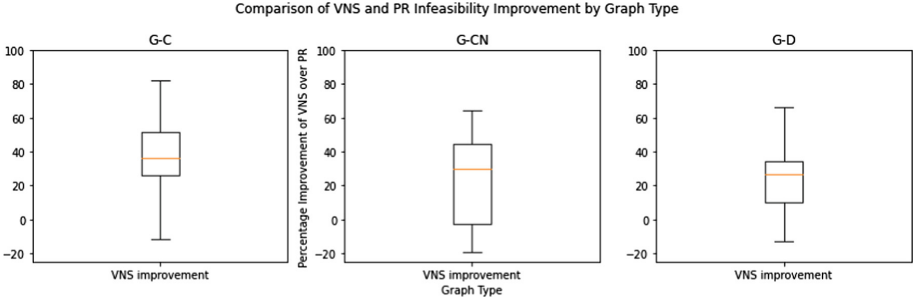


Fig. 3. Relative Infeasibility of BVNS and PR.

To examine the scalability of the algorithm, we tested BVNS on graphs with nodes $V = \{500, 600, 700\}$ with 10 graphs for each graph type. Table 3 shows the CPU time (in mins.) and the standard deviation of every graph type at a different number of nodes.

Table 3. CPU Time (in sec.) and the Standard Deviation of Each Graph Type.

Graph		500 Nodes	600 Nodes	700 Nodes
Graph Size	CPU Time			
G-C	<i>Mean</i>	405.12	821.50	941.52
	<i>SD</i>	103.49	444.32	243.47
G-CN	<i>Mean</i>	369.77	843.76	869.22
	<i>SD</i>	75.55	242.31	227.85
G-D	<i>Mean</i>	408.58	627.08	1117.42
	<i>SD</i>	125.19	169.98	363.40

As expected, the running time increases as the problem size grows larger at $V = 100$ increments. We note that for graph sizes $V = 500$ and $V = 700$, graph type G-CN had the lowest mean and standard deviation in their respective graph sizes. On the other hand, for graph size $V = 600$, graph type G-D had the lowest mean and standard deviation.

Furthermore, we can see that in the first increment between $V = 500$ and $V = 600$, the mean CPU time increased at a higher rate than between $V = 600$ and $V = 700$ for graph types G-C and G-CN. This suggests that graph types G-C and G-CN scale well as the problem increases in size. In addition, we note that for graph type G-D, the mean and standard deviation had their lowest rate of increase between graph sizes $V = 500$ and $V = 600$. This suggests that for this particular graph type, scaling the problem size up to $V = 600$ would not present significant increases in run time and that the algorithm would provide a solution in an adequate timeframe.

We note that the high mean and standard deviation for graph type G-D in graph size $V = 700$ can be attributed to the high convergence time caused by relatively small improvements in the balancing constraints for certain instances in this graph type.

6 Concluding Remarks

In this paper, we studied the DTDP, a districting problem often occurring in delivery operations, in which balancing and connectivity constraints are taken into account while minimizing the maximum diameter. We proposed two local search procedures that improve the objective function value and lower the relative infeasibility of a given solution. We used a Basic VNS following the LIMA paradigm under which we used both the local search variants and a simple shake procedure. We conducted computational experiments on graphs of $V = 500$ nodes and $p = 10$ districts with a competitive running time and average improvement at 6.35% over all graph types and a maximum improvement of 20.86%. Furthermore, we conducted computational experiments on graphs of $V = 600$ and $V = 700$ to showcase the scalability of the algorithm.

There are several areas of future research that arise from this problem. One promising area of research, given the results of the Basic VNS, is exploring other variants of VNS such as the General VNS.

Furthermore, applying different local search neighborhoods in conjunction with different neighborhood change steps could allow for further diversification and intensification of the solution space.

Acknowledgements. This research is supported by Khalifa University under Grant No. FSU-2020-19 and Award No. RC2 DSO.

References

1. Zoltners, A.A., Sinha, P.: Sales territory design: thirty years of modeling and implementation. *Mark. Sci.* **24**(3), 313–331 (2005)
2. Ricca, F., Scozzari, A., Simeone, B.: Political districting: from classical models to recent approaches. *Ann. Oper. Res.* **204**(1), 271–299 (2013)
3. Caro, F., Shirabe, T., Guignard, M., Weintraub, A.: School redistricting: embedding GIS tools with integer programming. *J. Oper. Res. Soc.* **55**(8), 836–849 (2004)
4. Enayati, S., Mayorga, M.E., Rajagopalan, H.K., Saydam, C.: Real-time ambulance redeployment approach to improve service coverage with fair and restricted workload for EMS providers. *Omega (Westport)* **79**, 67–80 (2018)
5. Sudtachat, K., Mayorga, M.E., Mclay, L.A.: A nested-compliance table policy for emergency medical service systems under relocation. *Omega (Westport)* **58**, 154–169 (2016)
6. Sandoval, G.M., Diaz, J.A., Rios-Mercado, R.: An improved exact algorithm for a territory design problem with p-center-based dispersion minimization. *Expert Syst. Appl.* **146**, 113150 (2020)

7. Rios-Mercado, R.: Fernandez, E: a reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Comput. Oper. Res.* **36**(3), 755–776 (2009)
8. Kalcsics, J., Nickel, S., Schröder, M.: Towards a unified territorial design approach: applications, algorithms, and GIS integration. *TOP* **13**(1), 1–56 (2005)
9. Rios-Mercado, R., Escalante, H.: GRASP with path relinking for commercial districting. *Expert Syst. Appl.* **44**, 102–113 (2015)
10. Mladenović, N., Hansen, P.: Variable neighborhood search. *Comput. Oper. Res.* **24**(11), 1097–1100 (1997)
11. Hansen, P., Mladenovic, N., Todosijevic, R., Hanafi, S.: Variable neighborhood search: basics and variants. *EURO J. Comput. Optim.* **5**, 423–454 (2016)
12. Mladenovic, N., Labbe, M., Hansen, P.: Solving the p-center problem with Tabu search and variable neighborhood search. *Networks* **42**(1), 48–64 (2003)
13. Hindi, K.S., Fleszar, K.: An effective VNS for the capacitated p-median problem. *Eur. J. Oper. Res.* **191**, 612–622 (2008)
14. Brimberg, J., Mladenovic, N., Todosijevic, R., Urošević, D.: Solving the capacitated clustering problem with variable neighborhood search. *Ann. Oper. Res.* **272**, 289–321 (2019)
15. Mladenovic, N., Alkandari, A., Pei, J., Todosijevic, R., Pardalos, M.P.: Less is more approach: basic variable neighborhood search for the obnoxious p-median problem. *Intl. Transn. in Op. Res.* **27**(1), 1–14 (2019)