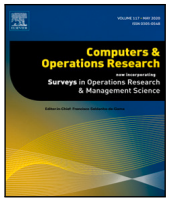
Computers & Operations Research 160 (2023) 106376

Contents lists available at ScienceDirect

Computers and Operations Research

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



A sample approximation solution procedure for chance-constrained districting problems

Silvia Baldassarre, Giuseppe Bruno, Antonio Diglio *, Carmela Piccolo

Università degli Studi di Napoli Federico II, Department of Industrial Engineering (DII), Piazzale Tecchio, 80 - 80125 Naples, Italy

ARTICLE INFO

Keywords:
Territory design
Facility location
Heuristic algorithm
Chance-constraints
Sample approximation
Optimization under uncertainty

ABSTRACT

In this paper, a Districting Problem with chance-constraints balancing requirements is investigated. The goal is to partition a set of basic Territorial Units into p contiguous and compact districts such that their probability of being balanced is above a minimum threshold. For such a problem, an approximate counterpart is considered, in which deterministic inequalities are used to express the need for the districting plan to be balanced across a large set of randomly drawn scenarios. This leads to a sample approximation problem. In order to solve the latter, a new heuristic algorithm is devised. The proposed procedure exploits a location–allocation scheme coupled with a so-called "balancing constraints-generation" procedure. In practice, the sample approximation problem is iteratively solved by adding demand scenarios (and, hence, the corresponding balancing constraints) on the fly. Several measures to drive the selection of such scenarios and embed them into the problem during the solution process are introduced and discussed. Extensive computational experiments on testbed instances from the literature prove the validity of the devised procedure, showing that it outperforms existing heuristics in the number of solved instances and/or computing times while assuring comparable solutions' quality, especially for larger-sized test cases.

1. Introduction and background

Districting Problems (DPs) represent a well-known family of optimization problems in which a set of basic geographical *Territorial Units (TUs)* has to be grouped into a given number of clusters (i.e., the *districts*). Typically, districts are created in such a way as to meet some desirable properties. The latter usually include *integrity*, *balancing*, *compactness*, and *contiguity*. *Integrity* means that each TU belongs to only one district. *Balancing* expresses the need for districts of similar size w.r.t. some *activity measures* associated with the TUs (e.g., areas, population, demand for a service). Without loss of generality, from now on, we refer to such activity measures as *demands*. *Contiguity* implies that the devised districting plan does not include enclaves, which also ensures that there is always a path connecting two TUs belonging to the same district that does not cross any other district. Finally, *compactness* is a topological property requiring that districts do not have elongated shapes. Other relevant criteria may apply depending on the specific application context (e.g., respect of natural or administrative boundaries, similarity w.r.t. an existing plan). Indeed, DPs have been successfully used to model and solve mid-to-long-term strategic decision problems in various fields, often referred to in the literature as: (i) *political districting* (Bozkaya et al., 2003; Ricca et al., 2013); (ii) *sales territory design* (Ríos-Mercado and Fernández, 2009; Salazar-Aguilar et al., 2011); (iii) *service districting* (Mendes et al., 2022; Ríos-Mercado and Bard, 2019); (iv) *distribution districting* (Bender et al., 2020; Konur and Geunes, 2019). The interested readers can find an extensive overview on districting in the chapter by Kalcsics and Ríos-Mercado (2019) and the book by Ríos-Mercado (2020).

The specific DP considered in this paper stems from the need to cope with uncertainty on the demands originating from the TUs. Indeed, if demand changes, one of the main features of a districting plan, i.e., balancing, may fail. This leads to a *stochasting districting problem*.

Scholars' interest towards *stochastic districting* is relatively recent. As the discussion by Kalcsics and Ríos-Mercado (2019) highlights, such a stream emerged in the context of distribution logistics for the design of pickup and delivery districts. In practice, when uncertainty on the demands exists, some authors resorted to districting approaches to deal with stochastic vehicle routing problems. This is the case, for instance, of the works by Haugland et al. (2007), Lei et al. (2012, 2016). In these studies, a *two-stage stochastic program* is adopted, where districts are devised in the first stage and routing decisions are taken in the second stage, once demands become known. The authors propose specific heuristic and metaheuristic procedures to solve these problems. A similar setting is also considered by Carlsson (2012), Carlsson and Delage (2013), who employ "computational geometry-based" methodologies for the problem at hand.

* Corresponding author.
E-mail address: antonio.diglio@unina.it (A. Diglio).

https://doi.org/10.1016/j.cor.2023.106376
Received 31 December 2022; Received in revised form 9 June 2023; Accepted 8 August 2023
Available online 19 August 2023

A two-stage stochastic programming model is also studied in Nikzad et al. (2021) and Diglio et al. (2020). Specifically, Nikzad et al. (2021) investigate a home health care planning problem with uncertainty on travel and service times. Districting decisions are taken in the first stage to partition the set of patients into clusters to be served by caregivers. Then, routing and scheduling decisions for care delivery are taken in the second stage. Instead, Diglio et al. (2020) ignore routing decisions and focus on a "more standard" formulation for districting, namely, optimizing compactness subject to integrity and balancing requirements. In the first stage, a districting plan is determined, seeking a set of compact districts; in the second stage, after uncertainty on demands is disclosed, redistricting decisions are taken, i.e., TUs are eventually reassigned to other districts to meet balancing.

An alternative framework for stochastic districting is considered in Darmian et al. (2021), who propose a *robust optimization* model – using the so-called uncertainty sets – to solve a healthcare districting problem motivated by uncertainty on the demands for health services. Besides, the authors also develop a graph-based genetic algorithm to achieve approximate solutions to the problem.

Finally, another recently explored approach consists of resorting to the use of *chance-constrained programming*. The latter is also the focus of the present work. In this case, a contiguous and compact districting plan is sought such that balancing requirements hold with a given probability, which is a user-defined parameter. The problem is first introduced by Diglio et al. (2021), who design a two-step sim-heuristic solution procedure (Juan et al., 2015) to solve it. In the first step, a contiguous solution is found by optimizing compactness and ignoring the balancing constraints. In the second step, a simulation routine is applied, which estimates the probability in the chance-constraints and drives the corrections to be made to the current solution (i.e., TUs' reassignments) if the given threshold is not met. For the same problem, Diglio et al. (2023) propose a *sample approximation* approach (see Luedtke and Ahmed, 2008). In that reference, the authors derive an approximate deterministic counterpart in which uncertainty is captured by a large finite set of randomly drawn scenarios. Such formulation is also the base of two solution matheuristic algorithms exploiting a well-established scheme in the location analysis and districting literature, known as *location–allocation* (Ríos-Mercado et al., 2021). The first method relies on a sub-sampled problem, while the second approach involves solving many single-scenario problems. Both methods provide an initial solution for the problem, which is iteratively improved. Extensive experiments prove the effectiveness of the proposed approaches, which produce higher-quality solutions w.r.t. the heuristic by Diglio et al. (2021), at the cost of a more expensive computational effort.

The present work introduces a novel matheuristic algorithm to solve the sample approximation model introduced by Diglio et al. (2023). Our research effort stems from some empirical observations gained from the latter reference. Firstly, a solution satisfying a given set of balancing requirements is often feasible for a larger set of such constraints. Also, attempting to solve a model embedding a high number of balancing constraints may be computationally prohibitive, even when a promising starting solution is used. Indeed, in Diglio et al. (2023), only a relatively small subset of larger-sized instances – involving 1000 TUs and 1000 demand scenarios – was solved. The key finding, in practice, is that an efficient solution procedure involves reducing the number of scenarios (hence, of balancing constraints) to be considered to obtain a feasible solution.

Therefore, the main contribution of this work is to design a more efficient algorithm for the discussed problem. The proposed algorithm is based on a location–allocation scheme coupled with a "*balancing constraints-generation*" procedure. In essence, the sample approximation model is iteratively solved by adding balancing constraints on the fly. Several measures/criteria to embed such constraints during the solution process are introduced and discussed. Extensive computational experiments on testbed instances from the literature prove the validity of the proposed procedure, showing that it outperforms existing heuristics in the number of solved instances and/or computing times while assuring comparable solutions' quality, especially for larger-sized test cases.

The remainder of the paper is organized as follows. In Section 2, we revisit the formulation of the investigated optimization problem. In Section 3, the proposed heuristic is described. In Section 4, results from the computational experiments are given. Conclusions and directions for future research finally follow in Section 5.

2. Optimization models

In this section, we start by presenting a general mathematical programming model for the chance-constrained districting problem. From this program, we build afterwards an approximate deterministic counterpart, i.e., a sample approximation model, which is also the core of the heuristic algorithm devised in Section 3. The models presented in this section are the same as in Diglio et al. (2023). Nevertheless, we briefly revisit them here for the sake of clarity. This way, we also ensure that this manuscript is self-contained.

2.1. A probabilistic districting model

The aim of the investigated problem is to partition a set, say I, of Territorial Units (TUs) into p contiguous districts, such that the balancing requirement holds with a given probability γ. Each TU i is associated with a demand, d_i, which is assumed to be a random variable with a known CDF. Accordingly, balancing is expressed by ensuring that each district serves a demand which deviates no more than a maximum desirable deviation α ($\alpha \in [0,1]$) from a reference value μ, accounting for the average demand per district ($\mu = \frac{\sum_{i\in I} d_i}{p}$). Note that μ is a random variable itself. We also denote by $\boldsymbol{\xi} = [d_1, \ldots, d_{|I|}]$ the random vector containing all the possible future realizations of the demands. Hence, we are looking for a districting plan whose probability $\mathbb{P}_{\boldsymbol{\xi}}$ of being balanced is above a given threshold γ whatever the realization of the demands might be.

As customary in districting problems, single assignments for the TUs are considered. This is accomplished by designating a representative TU for each district. To this end, binary decision variables x_{ij} are introduced for each pair of TUs $(i,j) \in I$, equal to 1 if TU i is assigned to TU j, and 0 otherwise. The *representatives* (or *centers*) of the districts are those TUs $j \in I$ such that $x_{jj} = 1$. Clearly, each district has a different representative. Hence, we abuse the language by saying that TU i is assigned to district j if it is assigned to TU j.

Besides, to model contiguity, we consider a graph $G = (I, E)$ representing a connected network underlying the problem, where I is the set of TUs, and E is the set of edges, i.e., of direct connections between the TUs. We say that two TUs are *adjacent* or *contiguous* if a direct connection exists between them. Accordingly, we denote by K_i the set of TUs adjacent to $i \in I$: $K_i = \{j \in I : (i,j) \in E \vee (j,i) \in E\}$.

The objective function is a distance-based measure of compactness, expressed as the sum of the allocation costs of the TUs to the (representatives of the) districts. We denote by c_{ij} the distances between TUs i and j.

Based on the introduced notation, the *Probabilistic Districting Problem* (PDP) can be formulated as follows:

$$\text{minimize} \quad \sum_{i\in I}\sum_{j\in I} c_{ij}x_{ij}, \tag{1}$$

$$\text{subject to} \quad \sum_{j\in N} x_{ij} = 1, \quad i \in I, \tag{2}$$

$$\sum_{j\in I} x_{jj} = p, \tag{3}$$

$$\mathbb{P}_{\boldsymbol{\xi}}\left[(1-\alpha)\mu x_{jj} \le \sum_{i\in I} d_i x_{ij} \le (1+\alpha)\mu x_{jj},\ j \in I\right] \ge \gamma, \tag{4}$$

$$\sum_{i\in N} x_{ij} - \sum_{i\in \cup_{v\in N}(K_v\setminus N)} x_{ij} \le |N| - 1,$$

$$j \in I,\ N \subset [I \setminus (K_j \cup \{j\})], \tag{5}$$

$$x_{ij} \in \{0,1\}, \quad i,j \in I. \tag{6}$$

Objective function (1) optimizes the compactness of the devised districting plan by minimizing the total assignment costs of the TUs to the districts. Constraints (2) are the single assignments constraints, assuring that each TU is assigned to only one district. Constraints (3) define the number of districts p to be designed. Constraints (4) state that balancing must hold with a probability higher than or equal to γ. Contiguity is ensured by means of Constraints (5), adopted from (Salazar-Aguilar et al., 2011). These inequalities specify that, for any given subset N of TUs allocated to a generic representative j not containing j, there must be an arc between N and the set containing j. Constraints (6) define the domain of the introduced decision variables.

Note that Constraints (5) are exponentially in number. Nevertheless, as (Salazar-Aguilar et al., 2011) discuss, they have two important pros. First, they only make use of the already introduced x-variables. Besides, they give rise to an exact row-generation approach to districting, where contiguity constraints are added on the fly during the solution process. As we discuss in Section 3, we also embed such an approach in our solution procedure. For this reason, we include these constraints in our formulation.

Finally, according to Salazar-Aguilar et al. (2011), we also consider the following valid inequalities enhancing our model:

$$\sum_{m \in K_i} x_{mj} \geq x_{ij}, \quad j \in I,\ i \in I \setminus (\{j\} \cup K_j). \tag{7}$$

These constraints state that, if a TU i is assigned to a representative j it is not adjacent to, a necessary condition for contiguity is that at least one TU m adjacent to i is also assigned to j. In the referenced paper, the authors demonstrate that such inequalities reduce the computational burden required to solve the problem up to optimality using a row-generation approach.

In summary, the PDP we wish to solve consists of minimizing (1), subject to (2)–(7).

2.2. A sample approximation model

As discussed by Diglio et al. (2023), an approximate deterministic counterpart to our PDP can be obtained by assuming that uncertainty can be captured by a finite set S of scenarios of large cardinality, randomly drawn from the known CDF underlying the demand. For such a random sample, chance-constraints (4) can be replaced by deterministic inequalities ensuring that the districting plan is balanced for at least $\gamma|S|$ of its elements. This leads, in fact, to a sample approximation approach (see Luedtke and Ahmed, 2008).

To model the problem, we need to index in S some relevant parameters. In particular, we denote by d_{is} the demand of TU $i \in I$ in scenario $s \in S$, and by μ_s the reference value for the balancing requirement under scenario s, expressed as the average demand per district under that scenario occurrence ($\mu_s = \frac{\sum_{i \in I} d_{is}}{p}$). Also, we introduce the binary decision variables ϕ_s, equal to 1 if the districting is not balanced under scenario s, and 0 otherwise.

With this additional notation, the probabilistic balancing constraints (4) can be replaced by the following inequalities:

$$\sum_{i \in I} d_{is} x_{ij} + f_s \phi_s \geq (1-\alpha)\mu_s x_{jj}, \qquad j \in I, s \in S, \tag{8}$$

$$\sum_{i \in I} d_{is} x_{ij} - g_s \phi_s \leq (1+\alpha)\mu_s x_{jj}, \qquad j \in I, s \in S, \tag{9}$$

$$\sum_{s \in S} \phi_s \leq \lfloor (1-\gamma)|S| \rfloor, \tag{10}$$

$$\phi_s \in \{0,1\}, \qquad s \in S. \tag{11}$$

Constraints (8)–(9) express the balancing requirements across the set of scenarios S. Note that these inequalities become inactive if the scenario is not balanced under scenario s. This is ensured through an appropriate setting of parameters f and g when the corresponding decision variable ϕ_s takes value one. Specifically, for each scenario $s \in S$, we set:

$$f_s = (1-\alpha)\mu_s - d_{[1]s},$$

$$g_s = (p-1-\alpha)\mu_s - \sum_{m=1}^{p-1} d_{[m]s},$$

where $d_{[m]s}$ is the demand vector under scenario $s \in S$ when demand values are sorted non-decreasingly.[1]

Inequality (10) ensures that the districting plan is balanced for at least $\gamma|S|$ scenarios, while the binary domain of the introduced ϕ-variables is stated by Constraints (11).

Based on the above, our *Sample Approximation Districting Problem*, say (SADP), consists of solving the following model: minimizing (1), subject to (2)–(3), (5)–(11). For such a problem, a heuristic procedure is described next.

Remark 1. The SADP is $\mathcal{NP}$-hard. This result is already discussed in Diglio et al. (2023) and in Ríos-Mercado et al. (2021) for a similar problem. Following the latter reference, we first note that, although inequalities (5) are exponential in number, checking the feasibility of a potential solution to our SADP (a p-partition of the set of TUs I) can be done in polynomial time using a Breadth-First Search or Depth-First Search algorithm. Hence, SADP is in NP. In addition, if we take the particular case of a complete underlying graph G and set $\gamma = 0$ (or, equivalently, if α takes very high values), which implies the balancing constraints (8)–(9) are not binding, the SADP becomes the p-Median Problem (pMP), which is well-known to be $\mathcal{NP}$-hard. In other words, the pMP is polynomially reducible to the SADP. From here, the result follows.

3. Solution algorithm

In this section, we develop a matheuristic algorithm for solving the presented Sample Approximation Districting Problem (SADP). The latter may become computationally prohibitive to tackle by commercial solvers even for a reduced number of TUs and scenarios involved (i.e., $|I|$ and $|S|$, respectively). Recall, in fact, that the considered problem is $\mathcal{NP}$-hard (see Remark 1). For this reason, resorting to heuristic approaches is necessary to solve instances of meaningful size in practical real-world-like applications.

The procedure we propose exploits a consolidated approach in the districting literature, i.e., the *location–allocation* scheme. The procedure, which we revisit in Algorithm 1, consists of two main steps: (i) the *Location phase*, in which a set of p candidate centers is determined (line 1); (ii) the *Allocation phase*, in which TUs are assigned to the identified centers obtaining a balanced and contiguous solution (line 4). At this point, new candidate centers are found by solving a 1-median problem within each district (line 5). This step helps minimizing our objective function (the allocation costs of the TUs). The procedure iterates until centers do not change in two consecutive iterations, that is, no improved solutions can be achieved.

[1] This result is rather intuitive if we consider that districts' representatives are assigned to themselves, which implies that each district contains at least one TU. Hence, in any given scenario s, we can say that the lowest demand a district can serve equals the lowest value of the demand observed in scenario s, i.e., $d_{[1]s}$. To the contrary, the highest possible value equals the total demand minus the lowest $(p-1)$ demand occurrences in that scenario, i.e., $\sum_{m=1}^{p-1} d_{[m]s}$. The latter sum accounts for the lowest possible value of the demand served by the other $(p-1)$ districts in case they serve only their representatives. From these observations, the above settings follow. We refer the reader to Diglio et al. (2023) for the proof of this result.

Algorithm 1 Location–allocation approach

1: Determine p initial centers. Let I_C be the resulting set;
2: **repeat**
3: $I^* \leftarrow I_C$; // best set of centers up to the current iteration
4: Impose the centers to the model, by fixing $x_{jj} = 1$, $\forall j \in I^*$, and solve it;
5: Update the centers of the districts; // solve a 1-median problem in each district
6: $I_C \leftarrow$ new set of p centers;
7: **until** $I_C = I^*$ // centers unchanged, i.e., no further improvement achieved
8: **return** $x^\star$; // a feasible districting solution.

Our main contribution is in the algorithmic approach devised to solve the underlying optimization model (line 4). Recall that the SADP involves $|I|\times|S|$ number of balancing constraints (see Inequalities (8)–(9)), which reduce to $p\times|S|$ when p centers are already identified. Since embedding all of them directly in the model may be impracticable, the core idea of the proposed algorithm is to start addressing a reduced model, restricted to a few selected scenarios (thus, a few balancing requirements). Once a solution for such a restricted model is found, we check if it is balanced in at least $\gamma|S|$ scenarios (as stated by Inequalities (10)). If that is the case, we have a feasible solution for the SADP. Otherwise, new scenarios are selected from the set of violated ones, i.e., those in which the current districting plan is not balanced. Accordingly, the corresponding balancing constraints are added to the model, in a sort of "balancing constraints-generation" approach, until the above condition is met. Several criteria are introduced to drive the selection of the scenarios to be added to the model. We detail these ideas in the next subsections.

3.1. A "balancing constraints-generation" procedure

The procedure we propose is based upon considering an initial restricted model that results from restricting model SADP to a random sample of scenarios S' drawn from set S. We denote this restricted model by SADP$[S']$. Of course, we target a sample such that $|S'|$ is clearly smaller than $|S|$. Note that we may also wish to start with a single scenario, that is, $|S'| = 1$. In general terms, our procedure starts by considering an `initialNumberOfScenarios`, which is a user-defined parameter.

We underline that we apply a slight modification to model SADP when restricted to the random sample S'. Specifically, we want to find a districting solution that satisfies all the considered scenarios (i.e., balancing requirements) simultaneously. The model we solve, denoted by SADP$'$ $[S']$, is obtained by replacing inequalities (8)–(11), by the following:

$$(1-\alpha)\mu_s x_{jj} \le \sum_{i\in I} d_{is}x_{ij} \le (1+\alpha)\mu_s x_{jj}, \qquad j \in I, s \in S'. \tag{12}$$

This way, the ϕ-variables do not need to be introduced at all in model SADP$'[S']$. Note that, of course, we can obtain the same result by imposing $\gamma = 1$ in model SADP$[S']$.

Let us denote by $\hat{\mathbf{x}}(S')$ the feasible solution obtained by solving the above model. After that, we evaluate the number of balanced scenarios $\hat{\pi}(\hat{\mathbf{x}}(S'))$ associated with this solution. This number can be easily computed by considering the demand vector associated with each scenario $s \in S$ and checking whether the solution is balanced for that demand occurrence. We denote by S_B the corresponding set of balanced scenarios. Clearly $S' \subset S_B$. If this number is higher than or equal to $\gamma|S|$, we stop. Otherwise, we look for an additional sample S'' of scenarios chosen in $S\backslash(S_B \cup S_T)$, and solve the model SADP$'[S' \cup S'']$. The number of new scenarios to be added at each iteration is again a user-defined parameter, denoted by `numberOfScenariosToAdd`. In this case, we update the `numberOfIterationsNeeded` to obtain a feasible solution. Also, we update the `totalElapsedTime` once each model is solved.

Note that S_T represents the subset of the so-called "*tabu*" scenarios. In fact, as our empirical evidence show, the addition of new scenarios (and the corresponding balancing constraints) may sometimes result in a lower number of balanced scenarios associated with the obtained solution (i.e., $\hat{\pi}$, see line 13 in Algorithm 2). However, as can be reasonably expected, we observed that the general trend of the number of balanced scenarios is non-decreasing with the number of iterations and, hence, the number of scenarios considered in the restricted model. Still, one may wish to avoid considering the scenarios causing such a reduction. Hopefully, a lower number of scenarios in the model may ease its resolution, yield a larger decision space and lead to higher-quality solutions, in terms of objective function (the model, in fact, is "more relaxed").

To this end, in a generic iteration of our solution procedure, we check if the above-mentioned number of balanced scenarios $\hat{\pi}$ is lower than that found in the previous iteration, say $\hat{\pi}_{OLD}$ (line 14). That being that the case, we update the set of tabu scenarios S_T by enlarging it to the set of scenarios added in the previous iteration (and leading to such reduction), namely S_{OLD}. We consider a maximum number of iterations, say k_{max}, in which scenarios in S_T are considered as tabu. Scenarios labeled as tabu for a number of iterations that exceeds such a threshold (updated at the beginning of each iteration as in line 9) are no longer considered in such status (line 17). We emphasize that the above considerations are actually into effect if $k_{max} > 0$. Otherwise, scenarios are never considered as tabu throughout our procedure.

The procedure stops either: (i) the desired number of balancing scenarios ($\gamma|S|$) is obtained, that is, a feasible solution has been found, or (ii) a `maxIter` number of iterations, or (iii) a `timeLimit` are attained, or (iv) the set of "eligible" scenarios that can be added to the model is empty (i.e., we are not able to add any new scenarios to our model, due to tabu conditions as the number of iterations increases).

If a feasible solution is obtained, we try to improve it. To this end, we perform a classic location–allocation scheme (as in Algorithm 1) without adding new balancing constraints. In practice, we solve model SADP$'[S']$, where S' denotes the latest (most updated) set of balancing constraints considered in the previous step.

It may be possible that, in this final loop, improved solutions may be associated with a number of balanced scenarios $\hat{\boldsymbol{\pi}}$ lower than $\gamma|S|$. When this circumstance is verified (line 33), we discard such solutions. This way, we ensure the feasibility of the obtained solutions across the original sample S.

We formalize the procedure in Algorithm 2.

Remark 2. Algorithm 2 is, in fact, a location-application scheme applied to our sample approximation problem (SADP). Specifically, we see the Location phase in line 3. The Allocation phase can be recognized in the main loop, i.e., lines 8–25. Indeed, at the end of this step, a feasible districting plan for the SADP is (hopefully) found. The final loop, i.e., lines 27–36, defines an Improvement phase, where we seek to optimize compactness by applying again a location–allocation approach starting from the feasible solution obtained in the previous step.

Remark 3. The p centers in line 3 are selected using the heuristic by Resende and Werneck (2004). The resolution of model SADP$'[S']$ in line 10 and in line 29 is performed through the row-generation approach by Salazar-Aguilar et al. (2011). In extreme summary, we start solving the model without inequalities (5). Note that the p obtained districts induce a p-partition of the underlying graph. Thus, we need to check if each partition contains some connected components, i.e., groups of connected TUs that are not connected to the center of their district, which we perform in polynomial time by depth-first search (see also Remark 1). If this circumstance occurs, each of these components induces a violated constraint, in the form of inequalities

Algorithm 2 Balancing constraints-generation procedure (`initialNumberOfScenarios`, `numberOfScenariosToAdd`, `maxIter`, `timeLimit`, k_{max})

1: Draw a random sample S' from sample S (with $|S'|$ = `initialNumberOfScenarios`);
2: Consider the restricted model SADP′[S'];
3: Using the restricted model select p centers. Let I_C be the resulting set of centers;
4: counter ← 1; // number of iterations needed to obtain a feasible solution
5: $\hat{\pi}_{OLD} \leftarrow 0$; // number of balanced scenarios in the previous iteration
6: $S_T \leftarrow \emptyset$; // set of tabu scenarios
7: $S_{OLD} \leftarrow \emptyset$; // set of scenarios added in the previous iteration
8: **repeat**
9: Update the number of iterations in which scenarios in S_T have been labeled as tabu;
10: Solve the restricted model SADP′[S'] fixing $x_{jj} = 1$, $\forall j \in I_C$;
11: Find the best center within each district. Let I'_C be the new set of centers; // solve a 1-median problem in each district
12: $I_C \leftarrow I'_C$; // best set of centers up to the current iteration
13: Evaluate the number of balanced scenarios $\hat{\pi}$ associated with the current solution. Let S_B be the set of balanced scenarios;
14: **if** ($\hat{\pi} < \hat{\pi}_{OLD}$) **then**
15: $S_T \leftarrow S_T \cup S_{OLD}$; // label the latest set of added scenarios as tabu
16: **end if**
17: Remove from S_T those scenarios that have been tabu for k_{max} number of iterations;
18: **if** $\hat{\pi} \leq \gamma|\Omega|$ **then**
19: Draw a random sample S'' from $S \setminus (S_B \cup S_T)$ (with $|S''|$ = `numberOfScenariosToAdd`);
20: $S_{OLD} \leftarrow S''$;
21: $S' \leftarrow S' \cup S''$;
22: counter ← counter + 1;
23: **end if**
24: Update `totalElapsedTime`; // total time spent so far
25: **until** ($\hat{\pi} \geq \gamma|\Omega|$) **or** (counter > `maxIter`) or (`totalElapsedTime` > `timeLimit`) or ($S \setminus (S_B \cup S_T) = \emptyset$);
26: **if** a feasible solution has been found (i.e., $\hat{\pi} \geq \gamma|S|$) **then**
27: **repeat**
28: $I^* \leftarrow I_C$; // best set of centers up to the current iteration
29: Solve the underlying optimization model fixing $x_{jj} = 1$, $\forall j \in I^*$;
30: Find the best center within each group;// solve a 1-median problem in each district
31: $I_C \leftarrow$ new set of p centers;
32: Evaluate the number of balanced scenarios $\hat{\pi}$ associated with the current solution;
33: **if** $\hat{\pi} \geq \gamma|S|$ **then**
34: save the current solution; // improved feasible solution found
35: **end if**
36: **until** $I_C = I^*$ //centers unchanged, i.e., no further improvement achieved
37: **end if**
38: **return** $x^\star$; // a feasible solution for the sample approximation.

(5), where the subset N corresponds to the connected component itself. The model is then solved again from scratch by adding these new cuts, until a contiguous solution is obtained.

3.2. Selecting new demand scenarios

We now need to clarify how we select the new demand scenarios to add et each iteration (line 19 in Algorithm 2). Recall that, in each iteration, we include `numberOfScenariosToAdd` scenarios to the model. We perform this step in three different ways, as we discuss next.

(i) Random selection.

We randomly select `numberOfScenariosToAdd` unbalanced and non-tabu scenarios (namely, in $S\setminus(S_B\cup S_T)$) and add them to our model.

(ii) Violation-based selection: Static case.

For each scenario $s \in S \setminus S_T$, we can associate each district j with the relative deviation, say Δ_{js}, from the upper/lower bounds of the balancing requirements. Let us denote by D_{js} the demand served by district j in scenario s (i.e., $D_{js} = \sum_{i\in I: x_{ij}=1} d_{is}$). Then, we can write:

$$\Delta_{js} = \max\{0, \frac{D_{js} - UB_s}{UB_s}, \frac{LB_s - D_{js}}{LB_s}\}, \qquad j \in I | x_{jj} = 1, \quad s \in S \setminus S_T,$$

where: $UB_s = (1+\alpha)\frac{\sum_{i\in I} d_{is}}{p}$, and $LB_s = (1-\alpha)\frac{\sum_{i\in I} d_{is}}{p}$, for each scenario $s \in S$. Clearly, $\Delta_{js} = 0$ if district j is balanced under scenario s. Accordingly, we can associate the scenario s with a *violation measure*, say $\overline{\Delta_s}$, calculated as the minimum, the maximum or the average deviation across the districts. Mathematically, we have:

- *Lowest Deviation*: $\overline{\Delta_s} = \min_{j\in I: x_{jj}=1}\{\Delta_{js}\}$;
- *Highest Deviation*: $\overline{\Delta_s} = \max_{j\in I: x_{jj}=1}\{\Delta_{js}\}$;
- *Average Deviation*: $\overline{\Delta_s} = \frac{\sum_{j\in I: x_{jj}=1}\Delta_{js}}{p}$.

Hence, one of the following options is applicable:

- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Lowest Deviation*;
- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Highest deviation*;
- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Lowest Average Deviation*;
- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Highest Average Deviation*.

The rationale behind this choice is that scenarios with the *Lowest Deviation* are "closer" to those currently considered, thus they may lead to an extended model which is (hopefully) still relatively easy to solve. Note that with "closer" we mean that such scenarios are almost satisfied by the current solution, or, in other words, "similar" to those already embedded in the model.

On the contrary, scenarios having the *Highest Deviation*, being "farther" from those already considered, may potentially lead to a solution that turns out to be balanced in a higher number of scenarios, while likely making the model more difficult to solve.

The *Average Deviation* trades-off between the above extreme cases.

Note that the above options determine four different settings for the proposed heuristic. Also, we underline that the chosen measure does not vary across the solution procedure. This is why we denote this as a "static" case.

(iii) Violation-based selection: Reactive case.

Another option is to adopt a selection criterion that varies alongside the solution process in a reactive posture to some occurring conditions. In particular, we assume to start with one of the above-mentioned violation-based measures and to keep it until the number of balanced scenarios is below a given threshold, say $\gamma'|S|$ (with $\gamma' < \gamma$). Then, we switch to its opposite. This is a way to embed some "dynamicity" into the selection process based on information gained from the procedure itself, as opposed to the static criteria previously described.

For example, let us consider $|S| = 100$, and $\gamma' = 0.25$. Suppose we start with the *Lowest Deviation* measure. That means we select the `numberOfScenariosToAdd` scenarios having the *Lowest Deviation* while the number of balanced scenarios associated with the current solution is less than 25. Otherwise, we switch to the *Highest Deviation* criterion. If such a number lowers again below 25, we revert to the *Lowest Deviation*, and so on.

Four reactive cases are considered, depending on the violation-based measure used to trigger the solution procedure:

- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Lowest Deviation* or the *Highest Deviation*, having the *Lowest Deviation* as starting selection measure;
- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Lowest Deviation* or the *Highest Deviation*, having the *Highest Deviation* as starting selection measure;
- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Lowest Average Deviation* or the *Highest Average Deviation*, having the *Lowest Average Deviation* as starting selection measure;
- selecting the non-tabu and unbalanced `numberOfScenariosToAdd` scenarios having the *Lowest Average Deviation* or the *Highest Average Deviation*, having the *Highest Average Deviation* as starting selection measure;

In total, we consider nine different selection criteria, corresponding to as many settings for the proposed heuristic: one random, and eight violation-based ones (four static, and four reactive).

4. Computational experiments

In this section, we report on the computational experiments performed to test the proposed matheuristic procedure. First, we describe the data test used and the implementation details in Section 4.1. Then, we present the obtained results in Section 4.2. Afterwards, we benchmark the introduced algorithm against two existing procedures in Section 4.3. Further results on harder instances, i.e., involving a higher number of TUs and scenarios, are given in Section 4.4. Some additional insights from the realized experiments are discussed in Section 4.5. Finally, a sensitivity analysis to some relevant parameters of the heuristic follows in Section 4.6.

4.1. Test data and implementation details

We use the same test instances as in Diglio et al. (2023) for our experiments. These instances are obtained using the so-called Gabriel graph (Gabriel and Sokal, 1969). The latter is a well-known approach in the districting literature to create a connected adjacency graph underlying the problem, where two generic points P_1 and P_2 in the plane are said to be adjacent if the closed disc having P_1P_2 as a diameter contains no other points.

In particular, our instances involve $|I|$ point-like basic TUs randomly generated in a $[0, 1000] \times [0, 1000]$ square following a uniform distribution. Specifically, we consider: $|I| \in \{100, 500, 1000\}$.

The demands d_i associated with TUs $i \in I$ are randomly generated from a uniform distribution $\mathcal{U}(a, b)$, whose extremes are determined by fixing the same expected value (equal to 50) and varying the corresponding Relative Standard Deviation (RSD).[2] The following values are considered: $RSD \in \{0.125, 0.250, 0.500\}$. Since the expected value remains the same, note that higher RSD values induce higher demand variability.

Also, various experiments are run by varying:

- the value of the maximum tolerance α : $\alpha \in \{0.10, 0.20\}$;
- the number of districts to create, p : $p \in \{6, 8\}$ for $|I| = 100$, $p \in \{10, 20\}$ for $|I| = 500$, $p \in \{20, 40\}$ for $|I| = 1000$;
- the desired probability for the balancing requirements to hold, γ: $\gamma \in \{0.70, 0.80, 0.90\}$.

The above settings lead to 36 instances for each value of $|I|$: three values for the RSD, two values of α, two values of p, and three values of γ. These instances can be made available to those interested upon writing to the authors.

Some additional parameters need to be defined to run our heuristic.

[2] For a random variable d, its RSD is given by: $\frac{\sqrt{\mathbb{V}ar[d]}}{\mathbb{E}[d]}$.

We recall that our procedure starts targeting a reduced model, including an initial number of demand scenarios (`initialNumberOfScenarios`), and iteratively adds new demand scenarios at each iteration (`numberOfScenariosToAdd`). We set both these parameters equal to 1. Also, we allow the procedure to run for at most 1000 iterations ($\texttt{maxIter} = \frac{|S|}{\texttt{numberOfScenariosToAdd}} = 1000$), and 1000 s ($\texttt{timeLimit} = 1000$ s). At each iteration, we focus on solving an SADP restricted to the subset of scenarios S' selected up to that iteration (i.e., SADP′$[S']$ – see line 10 in Algorithm 2). This step involves solving multiple Integer Programs (IPs) through the row-generation approach by Salazar-Aguilar et al. (2011) to ensure contiguity. For each individual IP, we impose the same time limit of 1000 s and a termination gap equal to 0.01. As our empirical evidence show, a few IPs are typically needed to converge to a balanced and contiguous solution to each SADP′$[S']$ (see Section 4.5.1 for further details). Therefore, no such limits are imposed on its resolution process at each algorithm iteration. Note that the same applies when we attempt to improve a feasible solution found to our SADP (if any, see line 29 in Algorithm 2).

Besides, we consider two values of k_{max}, i.e., the number of iterations in which demand scenarios are regarded as tabu: $k_{max} \in \{0, \texttt{maxIter}\}$ (with $\texttt{maxIter} = 1000$). This way, we have two extreme case: (i) scenarios can never be considered as tabu ($k_{max} = 0$); (ii) once a scenario is tabu, it remains in this status throughout all the solution procedure ($k_{max} = \texttt{maxIter} = 1000$).

Finally, parameter γ', used in the *Reactive Violation-based selection* criterion (see Section 3.2), is set as follows: $\gamma' = 0.50 \times \gamma$. Thus, in a generic iteration, the violation-based measure changes if the number of balanced scenarios associated with the current solution is above or below 50% of the target value $\gamma|S|$ (with $|S| = 1000$).

With the above-discussed settings, our empirical work consists of 1944 experiments in total: three values of $|I|$, 36 instances for each value of $|I|$, two values of k_{max}, and nine selection measures.

All these experiments were performed on an Intel(R) Core(TM) i7–8750H CPU at 2.20 GHz, with 16 GiB of RAM running Windows 10 Pro–64 bits operating system. The algorithm was coded in Python 3.6, and the IPs were solved using IBM ILOG CPLEX 12.10.

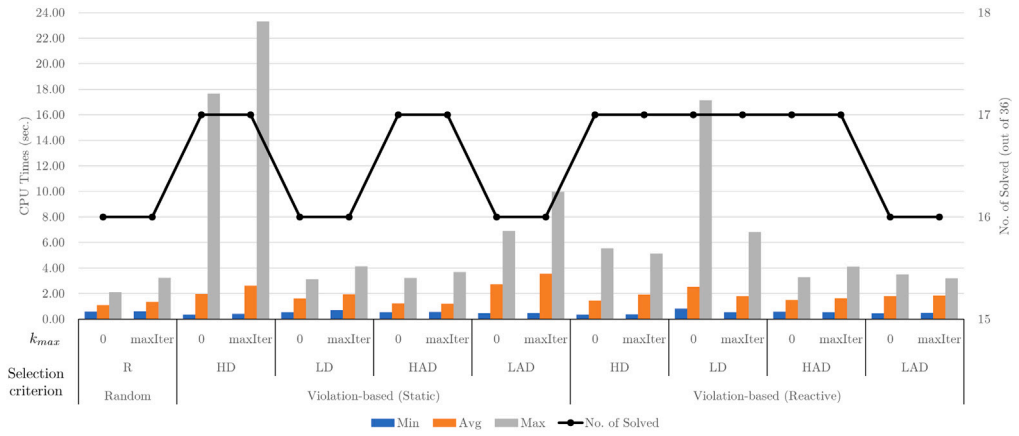
4.2. Computational results

This section presents the results obtained by implementing the proposed procedure for solving our Sample Approximation Districting Problem (SADP). We summarize them in Figs. 1–3. In these pictures, for each considered value of $|I|$, k_{max}, and the above-introduced selection criteria, we report on: (i) the minimum, average, and maximum computing times; (ii) the number of solved instances (out of 36); (iii) the minimum, average, and the maximum number of iterations needed for the convergence of the algorithm. Recall that the *Violation-based* selection criteria differ based on the specific measure used to evaluate new candidate demand scenarios to add to the model. For brevity, we denote these measures as follows:

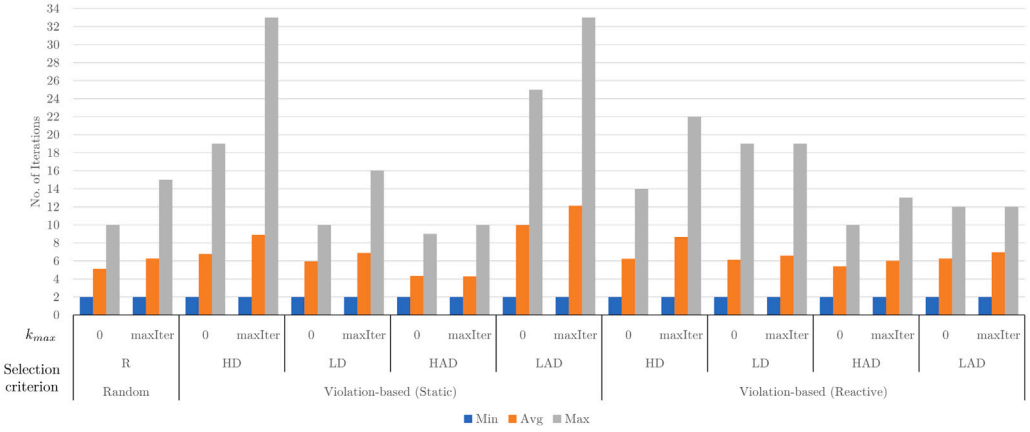
- HD: Highest Deviation;
- LD: Lowest Deviation;
- HAD: Highest Average Deviation;
- LAD: Lowest Average Deviation.

For the *Reactive* case, we use the above acronyms to denote the starting measure for our solution procedure. For instance, results displayed for the *Reactive* and *Lowest Deviation (LD)* case refer to the particular setting of the heuristic starting with the *LD* measure that will eventually change into the *Highest Deviation (HD)* one. All the detailed results are in the Electronic Supplement, Appendix A, Tables A21–A24

We start by focusing on $|I| = 100$. In general, the obtained results highlight the efficiency of the proposed heuristic. Indeed, computing times are very limited since most instances are solved, on average,



(a) Computing times (in sec.) and number of solved instances



(b) Number of iterations needed for the algorithms' convergence

Fig. 1. Results for $|I| = 100, |S| = 1000$.

within less than two seconds. In the worst cases, the average computing time equals 2.73 and 3.55 s, for $k_{max} = 0$ and $k_{max} = \texttt{maxIter}$, respectively. These circumstances both verify when the "static" LAD measure is employed (see Fig. 1(a), and Tables A1–A2. Also, as Fig. 1(b) shows, we notice that the algorithm converges in a relatively low number of iterations: less than ten and about 12 – on average – for $k_{max} = 0$, and $k_{max} = \texttt{maxIter}$. Note that the average number of iterations increases when the so-called tabu considerations apply. Nevertheless, the deriving effect on the corresponding computing times is negligible. Interestingly, tabu considerations lead to slightly higher-quality solutions. In fact, the average value of the objective function is lower when $k_{max} = \texttt{maxIter}$ for most of the measures employed (see Tables A5–A6). This result, which we wished to see in practice, proves that tabu considerations can determine some advantages in achieving higher-quality solutions at an acceptable extra computational effort.

Another relevant aspect of our analysis deals with the number of instances solved, depending on the specific selection criterion and the value of k_{max}. From Fig. 1(a), we notice that 16 instances are successfully tackled when a Random selection criterion is employed (see the solid black line). This number increases to 17 when a Violation-based criterion is adopted, particularly if the HD and HAD measures are used. Such a finding indicates that selecting scenarios based on their relative deviation from the balancing constraints is relevant to improving the effectiveness of the proposed algorithm. As we show in the next section, the above-discussed performance is in line with existing heuristics for the problem at hand. From Tables A1–A2, we also see that no feasible solutions are attained within the imposed time limit for the highest considered value of RSD (0.5). These are more challenging instances for our problem, as the higher variability in the demands makes the balancing constraints more difficult to meet. Other infeasibilities are noted for lower values of the RDS when α reduces to 0.1, which tightens the range for a districting solution to be balanced.

Overall, we note relatively better performance, in terms of both efficacy and efficiency, when the Highest Average Deviation measure is "statically" employed: indeed, it allows obtaining the highest number of feasible solutions (17), within very reduced computing times, namely 1.23 and 1.20 s, for $k_{max} = 0$, and $\texttt{maxIter}$, respectively.

A final comment on this first set of experiments is due. Recall that our SADP is, in fact, an approximate deterministic model for the "true" Probabilistic Districting Problem (PDP). Hence, the solution obtained using the proposed heuristic (which exploits the approximate model) may not satisfy the balancing requirements with the minimum probability γ and, hence, be unfeasible for the original PDP. Therefore, we perform an ex-post Monte Carlo simulation on each solution found by our heuristic, assessing the probability of a solution being balanced over a new sample of randomly drawn scenarios (out-of-sample validation). If the resulting probability is higher than γ, such a solution is considered feasible also for the PDP. These estimates are in Tables A7–A8, which show that almost all of the produced solutions also meet the desired probability in the ex-post simulation (see the last row in the table–#Feasible). In the remaining cases, the differences w.r.t. the threshold γ are not significant when looking at the values

involved. This evidence further validates our research effort, suggesting that the proposed heuristic provides robust approximate solutions to the investigated probabilistic problem.

The above findings are confirmed when looking at the results obtained for $|I| = 500$ (see Fig. 2). In this case, the heuristic solves 24 out of 36 instances regardless of the considered selection criterion and k_{max}. Obviously, we see an increase in the average computing times and the corresponding number of iterations needed for the algorithm to converge: all the solutions are achieved in less than 100 s and 33 iterations on average. Note, however, that these are two extreme upper bounds, as these indicators are much below the mentioned values in most cases. For these instances, the use of the Static Violation-based criterion with the HAD measure shows the best performance again, with an average computing time equal to 19.63 for $k_{max} = 0$, and 19.13 for $k_{max} = \texttt{maxIter}$. This fact further proves that selecting the most violated scenarios is a wise strategy to boost the computational efficiency of the algorithm.

These observations also hold when focusing on the larger instances, i.e., with $|I| = 1000$. An interesting point emerging from the analysis is that the choice of the selection criterion significantly affects the performance of the heuristic. In fact, the number of solved instances varies from nine to 20, reaching its maximum when employing a Static Violation-based criterion with the HAD and HD measures (in the latter case, for $k_{max} = 0$ only — see Fig. 3(a)). In particular, we notice again that the HAD "dominates" the others by taking, on average, about 300 s and 14 iterations. Additionally, it is worth highlighting that the running times are (averagely) 40 s lower, and slightly better average objective functions are observed when tabu considerations are into effect (see Tables A17–A22). Also, observe that, under the latter setting, all the obtained solutions are feasible to the PDP in the ex-post simulation (see Tables A23–A24).

In summary, our results prove the efficiency and effectiveness of the proposed procedure. Since the targeted instances involve a relatively high number of demand scenarios, the algorithm can attain approximate feasible solutions for the original probabilistic districting problem under investigation. Moreover, the realized experiments confirm the appropriateness of the introduced violation-based measures and the relevance of tabu considerations in the selection process of new demand scenarios and, hence, on the performance of our approach.

4.3. Comparison with existing heuristics

In order to assess the validity and competitiveness of the introduced procedure, the second step of our analysis consists of a comparative analysis against two existing algorithms, i.e., the heuristics by Diglio et al. (2023) and Diglio et al. (2021).

To this end, we use the results obtained with the best-performing setting of the proposed approach: (i) (Static) Violation-based selection criterion; (ii) Highest Average Deviation measure; (iii) $k_{max} = \texttt{maxIter} = 1000$.

4.3.1. Benchmarking against Diglio et al. (2023)

The developments presented in this paper stem from the attempt of finding a more efficient solution procedure for the sample approximation problem studied by Diglio et al. (2023). Therefore, the heuristics proposed in that work are a natural benchmark for our procedure. In the referenced study, the authors design two location–allocation schemes, which they tested under multiple settings of some relevant parameters (e.g., the constructors for identifying the initial p centers, the number of sub-sampled scenarios or single-scenario problems solved, to name a few). Each of these settings leads, in practice, to a different heuristic. In order to make a more robust assessment, we pick up the best solution found across all those settings, in terms of the objective function, and compare it against that achieved by the new proposed approach. If multiple solutions with the same objective function are attained, we use for comparison the one reporting the lowest computing time.

Results are summarized in Table 1. There, the heuristics proposed in the current paper and by Diglio et al. (2023) are respectively denoted by "New" and "Existing". The comparison is made based on the computing times and the gap between the objective functions, say Δ_{OF}, with the latter being computed as follows: $\Delta_{OF} = 100 \times \frac{OF_{New} - OF_{Existing}}{OF_{Existing}}$. Accordingly, a negative value of such indicators reveals that the new heuristic produces higher-quality solutions w.r.t. to the existing one. The extended results are in Appendix B, Table B1.

From the above table, it can be noticed that the new heuristic always outperforms that by Diglio et al. (2023) in terms of computing times and the number of solved instances (out of 36), especially for larger-sized cases, i.e., $|I| = 500$, and $|I| = 1000$. If we focus on the average computing times, they are reduced by two orders of magnitude for $|I| = 100$, and $|I| = 500$, and by a factor of approximately one-fifth for $|I| = 1000$. Observe that also the minimum and maximum values are incomparable. The very short computing times for $|I| = 100$ result in lower-quality objective functions. Indeed, the new heuristic produces solutions that are, on average, 2.62% worse than those obtained by the existing one. However, objective functions are comparable for the other cases, as shown by the significantly low values of the calculated deviations. Notably, the increased efficiency also yields greater effectiveness, as a remarkably higher number of instances is solved for the more challenging cases: 24 vs. 16 for $|I| = 500$, and 20 vs. 8 for $|I| = 1000$.

4.3.2. Benchmarking against Diglio et al. (2021)

We now present the comparison against the heuristic designed by Diglio et al. (2021). In that paper, the authors devise a sim-heuristic approach that embeds a simulation routine to drive the changes to be made in a current solution and verify that the chance-constraints balancing requirements are met. In practice, a feasible solution for the "true" Probabilistic Districting Problem (PDP) is found. In this paper, recall, we attempt to solve a Sample Approximation Problem as a way to attain feasible solutions to the PDP. Therefore, such a comparison is also meaningful. However, as the heuristic by Diglio et al. (2021) produces solutions to the "true" problem, we use for comparison only the "new" solutions that meet the balancing constraints with the required probability in the ex-post simulation, i.e., those that can be regarded as feasible also for the PDP.

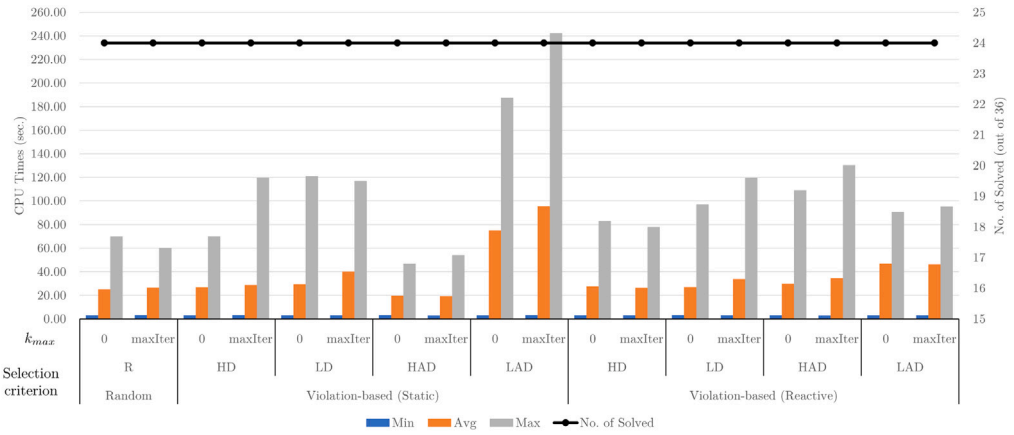
Results are summarized in Table 2. The information therein reported are the same as in Table 1. The detailed data can be found in Appendix B, Table B2.

Our findings reveal that, for $|I| = 100$, the method proposed in this paper solves a lower number of instances. However, the newly obtained solutions show lower objective functions' values (on average, 1.91% better than the existing ones) and are attained in shorter computing times (1.04 vs. 5.78 on average). When focusing on $|I| = 500$, the dominance of the new heuristic is clear: one more instance solved, slightly better objective functions (see the negative deviation), and a lower computational effort. Finally, results from larger-sized instances ($|I| = 1000$) confirm the higher effectiveness of our approach since five more instances are solved (20 vs. 15) with comparable solutions' quality at the cost of increased computing times (288.30 vs. 126.29 on average).

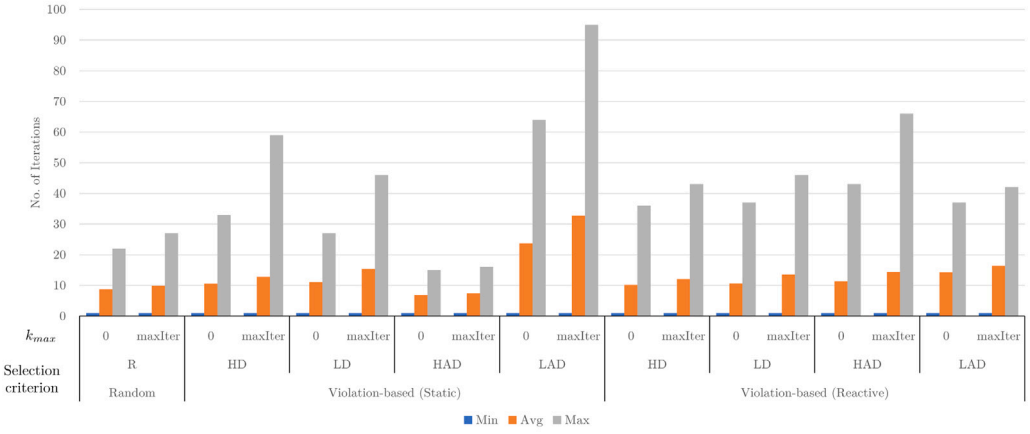
In conclusion, the general evidence emerging from these experiments is that the new heuristic is proven to be competitive w.r.t. existing algorithms, by always outperforming them in terms of computing times (efficiency) and/or the number of solved instances (effectiveness), while achieving comparable objective function values.

4.4. Further results — more challenging instances

In this section, we show the results of an additional set of computational tests we realized to assess the capability of the proposed



(a) Computing times (in sec.) and number of solved instances



(b) Number of iterations needed for the algorithms' convergence

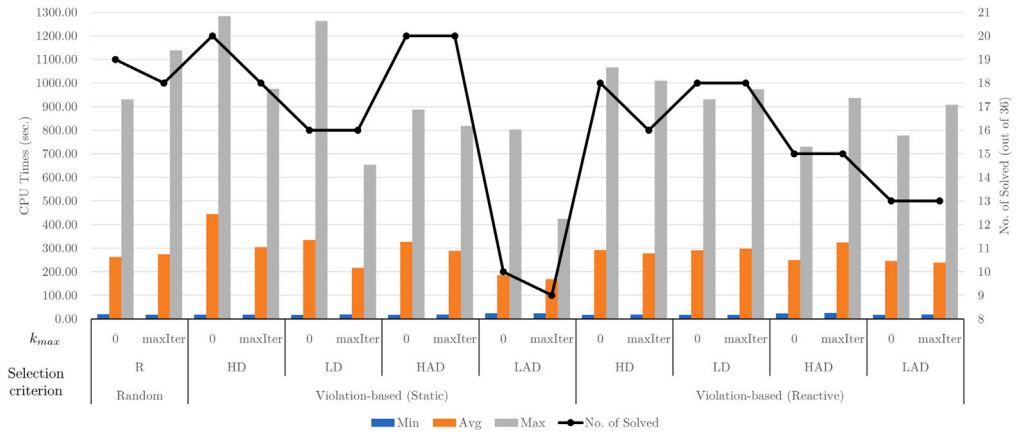
Fig. 2. Results for $|I| = 500, |S| = 1000$.

Table 1
Comparison against Diglio et al. (2023).

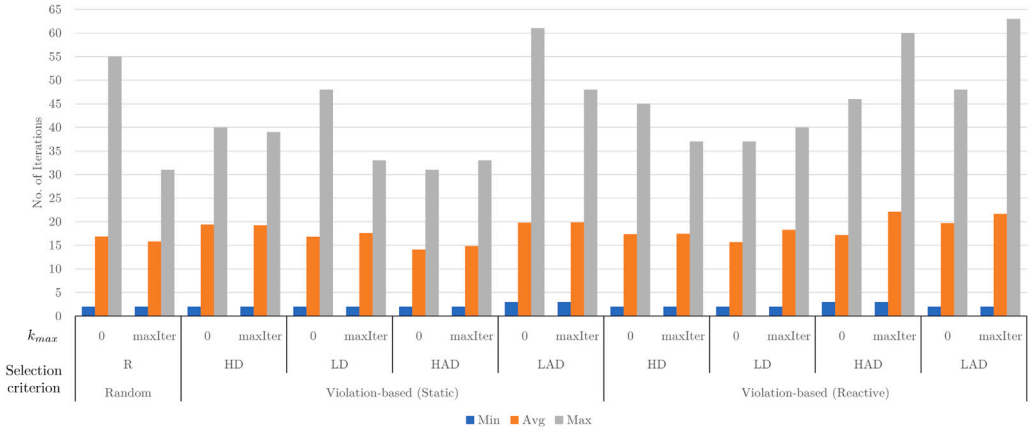
	$\|I\| = 100$			$\|I\| = 500$			$\|I\| = 1000$		
	CPU times (s)		Δ_{OF} (%)	CPU times (s)		Δ_{OF} (%)	CPU times (s)		Δ_{OF} (%)
	New	Existing		New	Existing		New	Existing	
Min	0.56	13.71	0.27	2.83	58.8	−0.09	18.45	624	−0.09
Max	3.69	2383.5	5.05	54.08	1344.69	1.25	818.45	1665.88	0.26
Avg	1.20	252.75	2.62	19.13	571.78	0.21	288.3	1086.09	0.05
#Solved	17	17		24	16		20	8	

Table 2
Comparison against Diglio et al. (2021).

	$\|I\| = 100$			$\|I\| = 500$			$\|I\| = 1000$		
	CPU times (s)		Δ_{OF} (%)	CPU times (s)		Δ_{OF} (%)	CPU times (s)		Δ_{OF} (%)
	New	Existing		New	Existing		New	Existing	
Min	0.56	0.41	−2.81	2.83	6.45	−1.36	18.45	62.3	−0.28
Max	2.04	66.93	−0.71	54.08	126.87	0.87	818.45	207.89	0.19
Avg	1.04	5.78	−1.91	19.52	36.42	−0.40	288.30	126.29	−0.10
#Solved	16	19		23	22		20	15	



(a) Computing times (in sec.) and number of solved instances



(b) Number of iterations needed for the algorithms' convergence

Fig. 3. Results for $|I| = 1000, |S| = 1000$.

procedure to solve more challenging instances for our Sample Approximation Districting Problem (SADP). Specifically, we target instances involving a higher number of scenarios ($|S|$) and Territorial Units (TUs – $|I|$).

In particular, two types of experiments are performed:

- *Experiment 1*. For this experiment, we consider the same 1000-TUs instances ($|I| = 1000$) but increase the number of scenarios to 10,000 ($|S| = 10000$). We perform these tests by setting the relevant parameters as in the previous empirical tests (α, γ, p);
- *Experiment 2*. Following the mechanism discussed in Section 4.1, we randomly generate an instance with 2000 TUs ($|I| = 2000$). We test it by only changing the number of districts p, here fixed equal to 40 and 60.

Again, the heuristic is run by considering the "Static" Higher Average Deviation, and $k_{max} = \texttt{maxIter}$. Note that the letter equals 2000 in Experiment 2. Also, the time limit is set at 2000 s. In total, 72 new experiments are performed.

The obtained results are in Table 3, which reports, for each of the two experiments, the computing times (CPU Times) and the number of iterations needed (# Iter) to attain a feasible solution to the SADP.

Briefly, we note that the heuristic seems relatively well-performing in the case of Experiment 1. The number of solved instances equals 18, and these are obtained, on average, within less than 800 s. Interestingly, we are able to solve two instances less than $|S| = 1000$. This is mainly due to the longer time required to check all the involved demand scenarios for balancing. In this regard, it is worth observing that the number of iterations needed for the convergence of the algorithm is relatively low, which indicates that the average time taken by a single iteration is significant. Also, only ten out of 18 instances would have been solved within the original time limit (1000 s).

Performance deteriorates for Experiment 2. Indeed, 11 instances are solved within an average computing time of about 980 s. In particular, none instances with RSD = 0.5 and only two for $p = 60$ are tackled. This is an indication that the heuristic "suffers" from the increase in the number of TUs.

Clearly, further effort is necessary to increase the effectiveness of the proposed procedure on very large-scale cases. Nevertheless, the number of solved instances is not negligible, and the overall results prove the heuristic to be competitive, especially when tackling instances involving large cardinalities of the scenario sets.

4.5. Additional insights

Having discussed in detail the performance of the proposed heuristic procedure, we now focus on some other aspects of relevance to our analysis. Specifically, in the following, we report on: the time performance of the single phases of the algorithm (Section 4.5.1); the number of the so-called *split units* (Section 4.5.2).

Table 3
Results for Experiments 1 and 2 — more challenging instances.

RSD	α	γ	Experiment 1 ($\|I\| = 1000, \|S\| = 10000$)			Experiment 2 ($\|I\| = 2000, \|S\| = 1000$)		
			p	CPU times (s)	# Iter	p	CPU times (s)	# Iter
0.125	0.1	0.7	20	315.11	10	40	1152.68	24
		0.8		415.81	11		1224.75	25
		0.9		538.73	15		1311.92	26
	0.2	0.7		68.11	2		216.16	5
		0.8		63.94	2		394.43	7
		0.9		67.81	2		463.56	11
	0.1	0.7	40	1367.24	21	60	t.l.	t.l.
		0.8		1296.88	21		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
	0.2	0.7		1275.01	20		1775.76	16
		0.8		1294.25	21		2276.75	19
		0.9		1664.07	26		t.l.	t.l.
0.25	0.1	0.7	20	1016.63	19	40	t.l.	t.l.
		0.8		1666.86	29		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
	0.2	0.7		89.56	3		517	12
		0.8		136.41	4		646.21	15
		0.9		207.88	6		852.14	23
	0.1	0.7	40	t.l.	t.l.	60	t.l.	t.l.
		0.8		t.l.	t.l.		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
	0.2	0.7		t.l.	t.l.		t.l.	t.l.
		0.8		t.l.	t.l.		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
0.5	0.1	0.7	20	t.l.	t.l.	40	t.l.	t.l.
		0.8		t.l.	t.l.		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
	0.2	0.7		748.81	19		t.l.	t.l.
		0.8		1616.59	33		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
	0.1	0.7	40	t.l.	t.l.	60	t.l.	t.l.
		0.8		t.l.	t.l.		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
	0.2	0.7		t.l.	t.l.		t.l.	t.l.
		0.8		t.l.	t.l.		t.l.	t.l.
		0.9		t.l.	t.l.		t.l.	t.l.
			Min	63.94	2	Min	216.16	5
			Max	1666.86	33	Max	2276.75	26
			Avg	769.43	14.67	Avg	984.67	16.64
			#Solved	18		#Solved	11	

4.5.1. Time performance of the phases of the algorithm

We recall that our algorithm involves three main phases:

- the *Location* phase, consisting of identifying an initial set of p centers/representatives for the districts, performed using the heuristic for the p-Median Problem by Resende and Werneck (2004) (line 3 in Algorithm 2);
- the *Allocation* phase, aiming at finding a feasible solution to the SADP by the iterative balancing-constraints generation procedure described in Section 3.1 (lines 8–25 in Algorithm 2);
- the *Improvement* phase, where we attempt to improve a feasible solution (if any) to our SADP (lines 27–36 in Algorithm 2).

Table 4 reports on the average proportion of time taken by each phase of the algorithm resulting from our computational experience for each tested combination of $|I|$ and p. Specifically, such evaluations are again made w.r.t. to the best-performing setting of the proposed heuristic: (i) (Static) Violation-based selection criterion; (ii) Highest Average Deviation measure; (iii) $k_{max} = \texttt{maxIter} = 1000$.

Table 4
Average proportion of time taken by each phase of the algorithm by $|I|$ and p (in %).

$\|I\|$	p	Location	Allocation	Improvement
100	6	8.04	83.83	8.13
	8	7.52	84.38	8.09
500	10	33.10	63.21	3.69
	20	4.34	90.47	5.19
1000	20	14.40	80.08	5.52
	40	0.90	94.68	4.42

As expected, the allocation phase is the most time-consuming. Indeed, it is the core of the algorithm as it involves solving an SADP restricted to a limited sample of selected scenarios at each iteration. Notably, the proportion of running time taken by this phase increases with p, which is consistent with the fact that this parameter tightens the balancing requirements and makes our instances harder to solve. Also,

Table 5
Detailed results for the allocation phase: Number of iteration needed for convergence and number of IPs solved at each iteration (i.e., in the row-generation approach).

$\|I\|$	p	No. of iterations and avg time				No. of IPs		
		Min	Avg	Max	Avg Time (s)	Min	Avg	Max
100	6	4	4.33	5	0.25	1	1	1
	8	2	4.25	10	0.30	1	1	1
500	10	1	6.13	15	2.05	1	1.27	10
	20	4	9.56	16	3.75	1	4.55	21
1000	20	2	10.54	26	9.60	1	2.07	12
	40	14	22.86	33	27.28	1	3.10	15

it tends to increase with the number of TUs ($|I|$). The only case such proportion falls below 70% is for $(|I|, p) = (500, 10)$. With these settings, in fact, the tested instances are solved in very limited computing times (about 10 s on average, see Table A10); thus, the running times of the location and allocation phases turn out to be more comparable (33.10% vs. 63.21%, respectively).

To gain more insights into the performance of the allocation phase, Table 5 informs on the number of iterations (minimum, average, and maximum) needed for convergence – i.e., to find a feasible solution to the SADP – by $|I|$ and p. Besides, the average running time per iteration is reported (Avg Time). In addition, it is worth reminding that each of these iterations involves solving multiple IPs to gain a contiguous solution. Therefore, the corresponding minimum, average, and maximum number of IPs solved are also displayed.

As our previous results already highlighted, we can observe that the heuristic converges in relatively few iterations. Not surprisingly, such a number also tends to increase with p and $|I|$. Interestingly, the average time taken by each iteration is acceptable and equals 27.28 s in the worst case (for $(|I|, p) = (1000, 40)$). Moreover, the number of IPs solved per iteration is limited. In the worst cases, in fact, its maximum is 21 for $(|I|, p) = (500, 20)$ and 15 for $(|I|, p) = (1000, 40)$, while the corresponding average values are much below these thresholds. In other words, the relatively low running times per iteration and the fast convergence to contiguity seem to further validate the efficiency of our heuristic. Also, they support our choice to resort to an off-the-shelf solver (although within a row-generation approach) for solving the required optimization models instead of more classical location–allocation schemes. A deeper discussion on this aspect follows in the next section.

4.5.2. Assessing the number of the split units

As stated throughout the manuscript, the required optimization models are always solved using a commercial solver within a row-generation approach (lines 10 and 29 in Algorithm 2). This differs from classical location–allocation schemes, which essentially work as follows: (i) p centers are fixed; (ii) a subproblem is solved, in which the tolerance α in the balancing constraints is set equal to zero, and the remaining integer variables are relaxed; (iii) the so-called *splits* (also referred to as *split nodes* or *split units*) are resolved to recover the integrity of the assignments.

The success of this approach, introduced by Hess et al. (1965) and used in Fleischmann and Paraschis (1988) and George et al. (1997) for classical Districting Problems (DPs), is due to two main factors: first, the subproblem in step (ii) is an easy-to-solve transportation-like problem; second, the number of fractional variables in the solution of this subproblem (i.e., the *splits*) is bounded by $p - 1$, regardless the size of the problem itself (i.e., $|I|$ - see Hojati, 1996).

However, the latter theoretical result does not hold when multiple-activity balancing requirements are simultaneously considered, as in Ríos-Mercado et al. (2021). In that reference, the authors investigate a DP with two-activity balancing requirements (that is, two demand scenarios, in our terminology), showing that the number of splits is well-behaved – i.e., often below $2(p - 1)$ – thus extending the classical location–allocation scheme by developing an ad-hoc split resolution heuristic.

Therefore, it is reasonable to check whether classical approaches are applicable here. Indeed, the SADP is a particular DP where at least $\gamma|S|$-activity balancing requirements are to be met (no less than 700, in our case - $\gamma = 0.7, |S| = 1000$). To this end, we evaluated the number of splits in the subproblem obtained from the SADP by applying the above steps (i) and (ii). We denote this relaxed linear program by (SADPR). Also, we denote by (SADPR[s]) the relaxed linear single-scenario program parametrized on each scenario $s \in S$. We further underline that, in line with (Ríos-Mercado et al., 2021), splits may come from two main sources: the TUs (without repetition) whose assignment variables are fractional in the solution of each SADPR[s]; the TUs that are integrally assigned to different centers in two different SADPR[s].

Table 6 reports on the splits found for the 500 and 1000-nodes instances, for each considered value of RSD used for demand generation, p, and different cardinalities of the scenario set S. For illustrative purposes, we vary $|S| \in \{2, \ldots, 5\}$. Finally, we underline that 20 different experiments are realized for each combination of $|I|$, RSD, p, and $|S|$, by varying the seed for generating the demands. Accordingly, the corresponding minimum, maximum, and average number of splits are reported.

The first aspect relevant to our analysis is that the number of splits is strongly affected by the RSD. Indeed, as the RSD influences the variability of the demands, it increases the number of units assigned to different centers in different SADP′[s], $s \in S$. Note that the average number of splits almost doubles when the RSD increases from 0.125 to 0.500 (all other conditions being equal, i.e., for the same values of $|I|$, $|S|$, and p).

Also, the splits grow by the number of TUs $|I|$. For RSD = 0.125, $|S| = 2$, and $p = 20$, the average number of splits increases from 22.35 (for $|I| = 500$) to 45.45 (for $|I| = 1000$). Similar considerations can be made for higher values of the RSD and $|S|$, although the involved values are obviously higher in absolute terms. Not surprisingly, the effect of $|I|$ is more severe if jointly occurring with an increase in p. Indeed, for the most challenging instances ($|I| = 1000, p = 40$), the average number of splits equals 101.95, 145.55, and 195.20 for RSD = 0.125, 0.25, and 0.5, respectively, when only two demand scenarios are considered ($|S| = 2$). Of course, in the worst cases, the maximum number of splits to be resolved is higher, although not much w.r.t. to the average if one looks at the values involved. Such observations hold for each tested value of $|S|$.

Clearly, the latter parameter plays a crucial role in this assessment. As expected, its increase yields the number of splits to grow. Such growth, in general, is more intense for higher values of $|I|$, p, and RSD (consistently with the above findings). Interestingly, the marginal growth of the number of splits exhibits a decreasing trend by $|S|$ (all other parameters kept equal). In other words, splits tend to stabilize with $|S|$. This may be an indication that there is a maximum number of scenarios – sampled from the same demand distribution – after which no significant variations would be devised, presumably due to recurring

Table 6
Number of splits units.

RSD	$\|I\| = 500$							$\|I\| = 1000$						
	p		$\|S\| = 2$	$\|S\| = 3$	$\|S\| = 4$	$\|S\| = 5$	Total	p		$\|S\| = 2$	$\|S\| = 3$	$\|S\| = 4$	$\|S\| = 5$	Total
0.125	10	Min	17	23	31	32	103	20	Min	40	50	53	65	208
		Max	26	37	39	41	143		Max	52	66	71	77	266
		Avg	22.25	30.60	34.95	37.15	124.95		Avg	45.45	56.45	65.20	70.55	237.65
	20	Min	30	40	46	50	166	40	Min	89	122	142	154	507
		Max	44	56	57	64	221		Max	106	145	163	179	593
		Avg	36.75	45.50	50.50	55.35	188.10		Avg	101.95	133.25	153.05	166.45	554.70
0.25	10	Min	24	38	43	50	155	20	Min	55	74	86	96	311
		Max	43	53	62	65	223		Max	80	101	107	117	405
		Avg	32.45	45.95	52.50	57.10	188.00		Avg	65.00	83.80	96.50	104.45	349.75
	20	Min	43	59	66	76	244	40	Min	133	170	206	229	738
		Max	81	74	89	94	338		Max	159	218	242	273	892
		Avg	51.50	67.90	78.60	85.90	283.90		Avg	145.55	194.20	225.40	250.75	815.90
0.5	10	Min	38	53	62	70	223	20	Min	75	108	123	140	446
		Max	60	78	91	96	325		Max	119	151	174	197	641
		Avg	48.10	65.20	74.60	84.70	272.60		Avg	92.70	124.60	147.15	166.50	530.95
	20	Min	56	85	94	111	346	40	Min	176	249	295	322	1042
		Max	88	109	130	133	460		Max	215	295	334	366	1210
		Avg	71.15	95.05	112.10	123.10	401.40		Avg	195.20	268.65	313.05	344.10	1121.00

similarities among the solutions of the relaxed single-scenario problem (SADPR$[s], s \in S$).

Also, recall that the number of splits is bounded by $p-1$ for each of the latter. However, no theoretical bounds are known for our problem. The average – and, often, the maximum – number of splits are found to be below $|S|(p-1)$ for most of the analyzed cases when RSD = 0.125. In the remaining ones, they turn out to be higher than that threshold (sometimes even significantly). This means that the number of splits is not guaranteed to be kept relatively low. In addition, consider that the split resolution problem should be tackled n times if n is the number of iterations needed for convergence. This implies, in practice, that for a problem converging to a feasible solution in five iterations, the total number of splits to be resolved would equal those displayed in Table 6 (see the "Total" columns). Indeed, we add one new demand scenario at each iteration; thus, the number of iterations and scenarios coincides (they may be slightly different only if tabu considerations are into effect). Therefore, although a proper comparison is not performed, the discussed efficiency of our procedure and the analyzed behavior of the splits make a strong case – in our opinion – why classical location–allocation approaches (at least in their current forms) may not be suitable for our SADP.

4.6. Sensitivity analysis to other parameters of the heuristic

Another step of our analysis consists of evaluating the sensitivity of the performance of our heuristic w.r.t. some of its other relevant parameters. In particular, one parameter is still to be tested: the number of scenarios added at each iteration (`numberOfScenariosToAdd`).

At this aim, we performed another full set of experiments using again the setting of the heuristic defined by the Static Highest Average Deviation measure for scenario selection, and $k_{max} = |S| = 1000$, by varying the parameter `numberOfScenariosToAdd` $\in \{2, \dots, 5\}$.

For brevity, we only report in Table 7 the comparison between our base-case (i.e., `numberOfScenariosToAdd` = 1) and the case `numberOfScenariosToAdd` = 4, which showed the best performance. We consider both the *test instances* described in Section 4.1 (i.e., those with 100, 500, 1000 TUs, and 1000 demand scenarios) and the *larger instances* presented in Section 4.4. Specifically, the comparison is made based on the computing times, the gap between the objective functions (Δ_{OF}), and the number of solved instances (#Solved). Note that Δ_{OF} is calculated as: $\Delta_{OF} = 100 \times \frac{OF_4 - OF_1}{OF_1}$, with OF_1 and OF_4 denoting the value of the objective function when `numberOfScenariosToAdd` equals 1 and 4, respectively.

The main result emerging from the above table is that increasing the number of scenarios added at each iteration yields more instances solved for $|I| = 1000$ (23 vs. 20). This parameter also seems to impact the efficiency of the heuristic. Although not reported in the table, it is worth underlining that, for $|I| = 1000$ and `numberOfScenariosToAdd` = 4, the heuristic takes, on average, about 160 s to solve the same 20 instances tackled when the same parameter takes value one. Observe that the average computing times are generally comparable (281.14 vs. 288.30 s). Besides, no significant differences are devisable when focusing on the gaps between the objective functions. Finally, as concerns the larger instances, our results show that the heuristic attains the same solutions (the gaps Δ_{OF} always equal zero) in similar running times.

In summary, our experiments prove the relevance of this parameter for the performance of the algorithm, at least in a subset of the tested instances. Nevertheless, further experiments performed on different test cases from those used in this paper would be necessary to fine-tune it and draw more general conclusions on its actual effects.

5. Conclusions

In this paper, a Probabilistic Districting Problem (PDP) was investigated. Assuming the demands associated with the Territorial Units (TUs) as a natural source of uncertainty, the problem consists of finding p contiguous and compact districts whose probability of being balanced is above a given threshold. For such a problem, a Sample Approximation approach was considered. In practice, an approximate counterpart was derived, in which the chance-constraints balancing requirements were replaced by deterministic inequalities expressing the need for the districting plan to be balanced across an extensive set of demand scenarios, randomly drawn from the (assumed) known CDF underlying the demands. In order to tackle the so-called Sample Approximation Districting Problem (SADP), a heuristic procedure was devised. The latter is based on a location–allocation scheme coupled with a "*balancing constraints-generation*" procedure. In essence, the algorithm starts targeting a reduced SADP, restricted to a few demand scenarios, and thus embedding a reduced number of balancing constraints. Then, it iteratively solves the problem by adding new demand scenarios (and the corresponding balancing constraints) on the fly. Several measures/criteria to embed such constraints during the

Table 7
Results of the heuristic by varying the number of scenarios added at each iteration (`numberOfScenariosToAdd`, denoted by $\#s_{add}$).

Test instances

	$\|I\| = 100$			$\|I\| = 500$			$\|I\| = 1000$		
	CPU times (s)		Δ_{OF} (%)	CPU times (s)		Δ_{OF} (%)	CPU times (s)		Δ_{OF} (%)
	$\#s_{add} = 1$	$\#s_{add} = 4$		$\#s_{add} = 1$	$\#s_{add} = 4$		$\#s_{add} = 1$	$\#s_{add} = 4$	
Min	0.56	0.70	−2.49	2.83	2.93	−0.41	18.45	17.63	−0.10
Max	3.69	4.43	1.69	54.08	62.11	0.31	818.45	1894.03	0.61
Avg	1.20	1.27	0.22	19.13	17.50	0.00	288.30	281.14	0.20
#Solved	17	17		24	24		20	23	

Larger instances

	$\|I\| = 1000, \|S\| = 10000$			$\|I\| = 2000, \|S\| = 1000$		
	CPU times (s)		Δ_{OF} (%)	CPU times (s)		Δ_{OF} (%)
	$\#s_{add} = 1$	$\#s_{add} = 4$		$\#s_{add} = 1$	$\#s_{add} = 4$	
Min	63.94	64.17	0.00	216.16	205.32	0.00
Max	1666.86	1879.73	0.00	2276.75	2154.80	0.00
Avg	769.43	783.20	0.00	984.67	941.45	0.00
#Solved	18	18		11	11	

solution process are introduced and discussed. Extensive computational experiments on testbed instances from the literature proved the validity of the proposed approach, demonstrating that it can tackle large-sized instances – involving up to 1000 TUs and 1000 demand scenarios – in acceptable computing times. Also, the introduced measures for scenario selection were shown to be relevant to the performance of our methodology. Besides, the comparison with existing works demonstrated that the current heuristic outperforms them in computing times and/or the number of solved instances while assuring comparable solutions' quality. Finally, the heuristic was also proven to be competitive on more challenging test cases, being able to solve instances involving 10,000 demand scenarios and 2000 TUs.

However, the latter results also revealed that the performance of our approach deteriorated when increasing the number of TUs. Additional tests would be necessary to fully understand "how far" our procedure can go on very large-scale cases. Nevertheless, further refinements to improve the computational efficiency and effectiveness of the approach are worth investigating. Applications are another interesting field to be explored. Of course, real-world problems may involve various additional requirements of practical relevance. However, as long as their underlying structure roots into districting, the proposed approach emerges as a viable tool to obtain approximate solutions to such problems. Finally, we note that the structure of the proposed heuristic makes it applicable to a broader range of closer optimization problems (e.g., Facility Location). Therefore, extending and testing the introduced method to design a general solution procedure for sample approximation and chance-constraints programming problems is an ambitious line of research to pursue.

Acknowledgments

The authors are thankful to two anonymous reviewers, whose comments helped us improve the quality of our paper.

CRediT authorship contribution statement

Silvia Baldassarre: Conceptualization, Methodology, Investigation, Validation, Writing – original draft, Writing – review & editing. **Giuseppe Bruno:** Conceptualization, Methodology, Investigation, Validation, Writing – original draft, Writing – review & editing. **Antonio Diglio:** Conceptualization, Methodology, Investigation, Validation, Writing – original draft, Writing – review & editing. **Carmela Piccolo:** Conceptualization, Methodology, Investigation, Validation, Writing – original draft, Writing – review & editing.

Data availability

Data will be made available on request.

Appendix 1. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cor.2023.106376.

References

Bender, M., Kalcsics, J., Meyer, A., 2020. Districting for parcel delivery services–a two-stage solution approach and a real-world case study. Omega 96, 102283.
Bozkaya, B., Erkut, E., Laporte, G., 2003. A tabu search heuristic and adaptive memory procedure for political districting. European J. Oper. Res. 144 (1), 12–26.
Carlsson, J.G., 2012. Dividing a territory among several vehicles. INFORMS J. Comput. 24 (4), 565–577.
Carlsson, J.G., Delage, E., 2013. Robust partitioning for stochastic multivehicle routing. Oper. Res. 61 (3), 727–744.
Darmian, S.M., Fattahi, M., Keyvanshokooh, E., 2021. An optimization-based approach for the healthcare districting under uncertainty. Comput. Oper. Res. 135, 105425.
Diglio, A., Nickel, S., Saldanha-da-Gama, F., 2020. Towards a stochastic programming modeling framework for districting. Ann. Oper. Res. 292 (1), 249–285.
Diglio, A., Peiró, J., Piccolo, C., Saldanha-da Gama, F., 2023. Approximation schemes for districting problems with probabilistic constraints. European J. Oper. Res. 307 (1), 233–248.
Diglio, A., Peiró, J., Piccolo, C., Saldanha-da-Gama, F., 2021. Solutions for districting problems with chance-constrained balancing requirements. Omega 103, 102430.
Fleischmann, B., Paraschis, J.N., 1988. Solving a large scale districting problem: a case report. Comput. Oper. Res. 15 (6), 521–533.
Gabriel, K.R., Sokal, R.R., 1969. A new statistical approach to geographic variation analysis. Syst. Biol. 18, 259–278.
George, J.A., Lamar, B.W., Wallace, C.A., 1997. Political district determination using large-scale network optimization. Socio-Econ. Plan. Sci. 31 (1), 11–28.
Haugland, D., Ho, S.C., Laporte, G., 2007. Designing delivery districts for the vehicle routing problem with stochastic demands. European J. Oper. Res. 180 (3), 997–1010.
Hess, S.W., Weaver, J., Siegfeldt, H., Whelan, J., Zitlau, P., 1965. Nonpartisan political redistricting by computer. Oper. Res. 13 (6), 998–1006.
Hojati, M., 1996. Optimal political districting. Comput. Oper. Res. 23 (12), 1147–1161.

Juan, A.A., Faulin, J., Grasman, S.E., Rabe, M., Figueira, G., 2015. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. Oper. Res. Perspect. 2, 62–72.

Kalcsics, J., Ríos-Mercado, R., 2019. Districting problems. In: Laporte, G., Nickel, S., Saldanha-da-Gama, F. (Eds.), Location Science. Springer International Publishing, 2nd edition, pp. 705–743.

Konur, D., Geunes, J., 2019. Integrated districting, fleet composition, and inventory planning for a multi-retailer distribution system. Ann. Oper. Res. 273 (1), 527–559.

Lei, H., Laporte, G., Guo, B., 2012. Districting for routing with stochastic customers. EURO J. Trans. Logist. 1 (1–2), 67–85.

Lei, H., Wang, R., Laporte, G., 2016. Solving a multi-objective dynamic stochastic districting and routing problem with a co-evolutionary algorithm. Comput. Oper. Res. 67, 12–24.

Luedtke, J., Ahmed, S., 2008. A sample approximation approach for optimization with probabilistic constraints. SIAM J. Optim. 19 (2), 674–699.

Mendes, L.H.P., Usberti, F.L., Cavellucci, C., 2022. The capacitated and economic districting problem. INFORMS J. Comput..

Nikzad, E., Bashiri, M., Abbasi, B., 2021. A matheuristic algorithm for stochastic home health care planning. European J. Oper. Res. 288 (3), 753–774.

Resende, M.G., Werneck, R.F., 2004. A hybrid heuristic for the p-median problem. J. Heuristics 10 (1), 59–88.

Ricca, F., Scozzari, A., Simeone, B., 2013. Political districting: from classical models to recent approaches. Ann. Oper. Res. 204 (1), 271–299.

Ríos-Mercado, R.Z. (Ed.), 2020. Optimal districting and territory design. Springer International Publishing, Cham.

Ríos-Mercado, R.Z., Álvarez-Socarrás, A.M., Castrillón, A., López-Locés, M.C., 2021. A location-allocation-improvement heuristic for districting with multiple-activity balancing constraints and p-median-based dispersion minimization. Comput. Oper. Res. 126, 105106.

Ríos-Mercado, R.Z., Bard, J.F., 2019. An exact algorithm for designing optimal districts in the collection of waste electric and electronic equipment through an improved reformulation. European J. Oper. Res. 276 (1), 259–271.

Ríos-Mercado, R.Z., Fernández, E., 2009. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. Comput. Oper. Res. 36 (3), 755–776.

Salazar-Aguilar, M.A., Ríos-Mercado, R.Z., Cabrera-Ríos, M., 2011. New models for commercial territory design. Netw. Spat. Econ. 11 (3), 487–507.