Model-free Sensor Placement for Water Distribution Networks using Genetic Algorithms and Clustering [★]

Luis Romero-Ben [*] **Gabriela Cembrano** [*] **Vicenç Puig** [*,**] **Joaquim Blesa** [*,**]

[*] *Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Barcelona 08028, Spain (e-mail: [luis.romero.ben, gabriela.cembrano]@upc.edu)*
[**] *Supervision, Safety and Automatic Control Research Center (CS2AC) of the Universitat Politécnica de Catalunya, Terrassa 08222, Spain (e-mail: [joaquim.blesa, vicenc.puig]@upc.edu).*

Abstract: This paper presents a model-free methodology for the placement of pressure sensors in water distribution networks (WDNs) with the aim of performing leak detection/localization tasks. The approach is based on a custom genetic algorithm (GA) optimization scheme, which considers a population whose individuals are binary vectors encoding the network nodes with/without sensors. The optimization process pursues the minimization of a distance-based metric, computed considering the pipe distance from the possible sensors to the complete set of nodes of the network, hence removing the necessity of a hydraulic model of the WDN. The methodology is completed by means of an iterative clustering technique that seeks the enhancement of incoming individuals. The proposed methodology is tested over a well-known case study, L-TOWN from the BattLeDIM2020 challenge, in order to assess its performance.

Keywords: sensor placement, water distribution networks, model-free, GAs, clustering

1. INTRODUCTION

The appearance of leaks in water distribution networks (WDNs) involves an important water loss, around 126 billion cubic meters per year worldwide (expressed as non-revenue water), according to Liemberger and Wyatt (2019). This fact justifies the interest of water utilities in the development of leak management strategies, considering the associated economic and environmental costs.

Nowadays, leak management is performed through software-based methods that require hydraulic information of the WDN status, gathered by means of a set of sensors (typically measuring pressure due to their reduced cost and easier installation), to operate. The most common leak detection/localization approaches are model-based, i.e., they require a hydraulic model of the WDN. This model computes every possible leak scenario, generating a fault sensitivity matrix (FSM) that stores the effect of each leak on every pressure measurement. Thus, several state-of-the-art pressure sensor placement solutions seek the performance maximization in these FSM-based leak management strategies by means of optimization-based schemes. Genetic algorithms (GAs) are typically used due to the possibility of encoding the sensor distribution over the network nodes as binary vectors: Pérez et al. (2009) presents this kind of sensor placement solution for leak detection task, Casillas et al. (2013) proposes a similar placement strategy but focusing on leak localization, and Quiñones-Grueiro et al. (2019) combines both approaches to consider detection and localization simultaneously. In Blesa et al. (2015), the FSM-based solution proposes a multi-objective exhaustive search scheme, together with a clustering method for search-space reduction.

An alternative to FSM-based methods lies in the use of information and entropy theory. GAs are also used in Khorshidi et al. (2020) to place sensors using Value of Information and Transinformation Entropy methods. Another information theory based method is proposed by Santos-Ruiz et al. (2022), ranking candidate nodes using a heuristic algorithm with a quadratic computational cost. Machine learning is also used in recent methods like Peng et al. (2022): the network is split into monitoring areas via graph neural network clustering, to then select the sensor through a leak localization analysis.

This paper presents a model-free sensor placement methodology seeking the minimization of a user-defined metric based on topological distance, by means of a customized implementation of GAs. Additionally, an iterative clustering strategy is proposed to help in the minimization of this metric. Thus, the main contribution of the method is its independence of the hydraulic model of the WDN, which is not required because the method only considers structural information. Despite the convenience of using a hydraulic model for sensor placement, motivated by the possibility of optimizing leak localization performance explicitly in an off-line process, it is also true that many water utilities do not have reliable hydraulic models and may not prioritize investments in periodic re-calibrations.

2. METHODOLOGY

The proposed sensor placement method pursues the minimization of a user-defined metric by means of a GA scheme: considering an initial population, individuals are selected depending on their score, combined and mutated with intent to generate new

[★] This work has been partially funded by the project PID2020-115905RB-C21 (L-BEST) funded by MCIN/ AEI /10.13039/501100011033.

individuals with a better score than their predecessors. Unlike most of the state-of-the-art methodologies, this work considers an evaluation function that removes the need for a hydraulic model. Moreover, an iterative clustering strategy based on the proximity of the sensors to the network nodes (henceforth denoted as nearest-sensor iterative clustering, NS-IC) is applied at several key steps to improve incoming individuals' score. A diagram of the complete methodology is provided in Fig. 1, involving some key steps, namely:

(1) Before proceeding with the operation, the required input parameters must be configured.
(2) Then, an initial random population of individuals is generated. The NS-IC process is used to improve the random individuals, and then they are evaluated to obtain a score that represents their suitability as solutions.
(3) In this moment, the iterative phase (formed by an external loop and an internal loop) starts, conducting the computation of the different steps of the GA: selection, combination, mutation and evaluation again.
(4) The internal loop is exited if the population gets stuck, leading to a preliminary convergence that triggers different countermeasures to renew the population: NS-IC is applied, and if the population's score is not improved, then a population diversity increase policy (PDI-P) is used.
(5) Finally, if the conditions for the final convergence are met, the algorithm stops and the solution is achieved.

The following subsections describe the details of these steps and specifies their underlying algorithms.

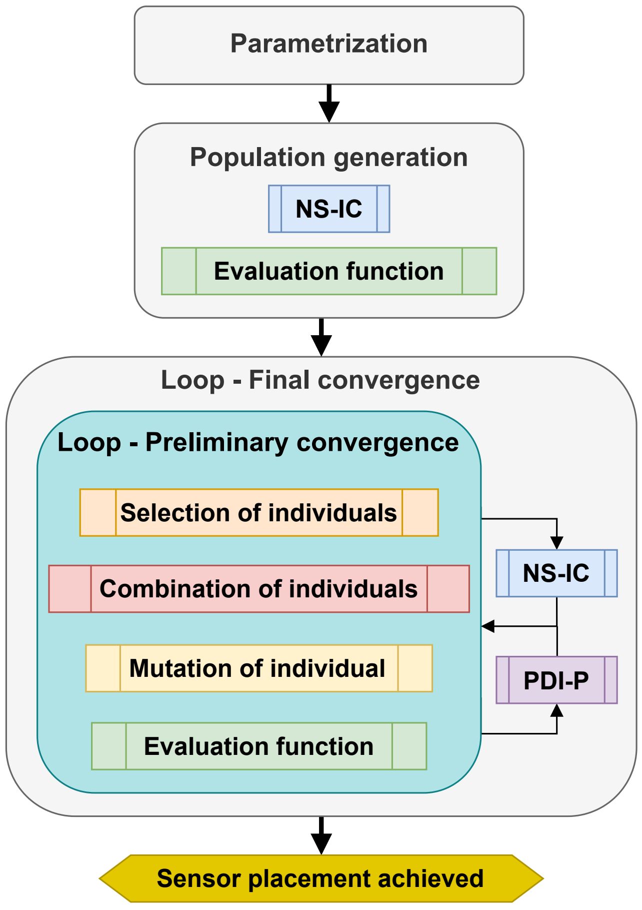

Fig. 1. Block diagram of the operation of the methodology.

2.1 Parameterization

Several parameters or input settings must be provided by the user, i.e., the water utility operator:

- Number of sensors (n_s): amount of pressure sensors that are considered to be located throughout the WDN.
- Number of individuals in the population (n_i): amount of individuals that compose the population of the GA. A larger value would imply a larger diversity of individuals, but it would also increase the computation time.
- Mutation probability (p): probability between 0 and 1 for the mutation process to be activated.
- Preliminary convergence counter thresholds (thr_1 and thr_2): these parameters trigger certain processes, depending on the value of the preliminary convergence counter cnt_p, with the aim of: (a) applying NS-IC, for the case of thr_1; (b) increasing the population diversity once it is insufficient using PDI-P, for the case of thr_2.
- Additionally, the methodology is prepared to let the user provide sensor locations that must be mandatorily selected and maintained during the sensor placement operation. Henceforth, let us refer the set of user-defined sensors locations as $\mathcal{S}_{ud}$. Note that $|\mathcal{S}_{ud}| < n_s$ for the sensor placement problem to not be trivial.

2.2 Population generation

The design of a GA-based strategy entails the conceiving of its population, i.e., its individuals and the information they encode.

As noted by several state-of-the-art works considering GAs for sensor placement over WDNs, the problem naturally leads to the use of binary vectors as individuals, encoding the sensor location information as follows:

$$\begin{aligned} \boldsymbol{P} &= [\boldsymbol{p}_1\ \boldsymbol{p}_2\ \dots\ \boldsymbol{p}_{n_i}] \in \mathbb{R}^{n\times n_i}; \\ \text{with}\ \ \boldsymbol{p}_i &= [p_{i1}, p_{i2}, ..., p_{in}] \end{aligned} \tag{1}$$

where $\boldsymbol{P}$ is the population of individuals (denoted as $\boldsymbol{p}_i \in \mathbb{R}^n$, with $i = 1, 2, ..., n_i$), n is the number of nodes of the WDN, and:

$$p_{ij} = \begin{cases} 1, & \text{if node } j \text{ is sensorized} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Besides, note that the following requirements must be fulfilled:

$$\sum_{j=1}^{n} p_{ij} = n_s \quad \forall i = 1, 2, \dots, n_i \tag{3}$$

$$p_{ik} = 1, \forall k \in \mathcal{S}_{ud} \tag{4}$$

Thus, the population is generated by computing n_i random binary vectors which accomplish the requirements in Eq. (3-4). Note that if $\mathcal{S}_{ud} \neq \emptyset$, the number of sensors to allocate by the method is reduced to $n_s - |\mathcal{S}_{ud}|$.

2.3 Evaluation function

The aim of the GA is the improvement, generation after generation, of the sensor location solution encoded by the individuals. Thus, a function that evaluates the individuals, giving them a score, is required. In this work, the suitability of the individuals is not assessed in terms of hydraulic measurements but in terms of distance-based metrics. Specifically, the proposed evaluation function rewards individuals whose encoded sensors are scattered throughout the WDN.

First, the pipe-distance matrix $\boldsymbol{D}$ of the WDN must be generated from its structural properties. Let us model the network as a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, defined by the node set $\mathcal{V}$ representing the junctions/reservoirs of the network, and edge set $\mathcal{E}$ that models the pipes of the network. The weights of $\mathcal{G}$ would be defined by the pipe lengths, which can be easily provided by the water utility. Note that extra information could be considered during their, e.g., diameters, roughness, etc. Thus, $\boldsymbol{D}$ can be generated from $\mathcal{G}$ as:

$$d_{ij} = \mathrm{f}(\mathcal{G}, v_i, v_j) \tag{5}$$

where $\mathrm{f}(\mathcal{G}, v_i, v_j)$ corresponds to a function that computes the shortest path from node v_i to node v_j across the weighted graph $\mathcal{G}$ (see Festa (2006) for a review of shortest-path algorithms).

Then, the evaluation function can be implemented to obtain the scores of the individuals of a population $\boldsymbol{P}$, as indicated by Algorithm 1. Note that $\boldsymbol{D}(:, \boldsymbol{p}_i)$ is obtained by selecting the columns of $\boldsymbol{D}$ indicated by the individuals encoded in $\boldsymbol{p}_i$. Besides, $\min_{\mathrm{col}}(\cdot)$ is a function that computes the minimum value at each row of an input matrix, yielding a column vector of minimum values (in particular, $\min_{\mathrm{col}}(\boldsymbol{D}_s)$ provides $\boldsymbol{d}_s^{min}$). Function $\mathrm{g}(\cdot)$ implements the user-defined function that calculates the individual's score by means of $\boldsymbol{d}_s^{min}$.

Algorithm 1: Evaluation function

```
Data: $\boldsymbol{P} \in \mathbb{R}^{n \times n_i}$, $\boldsymbol{D} \in \mathbb{R}^{n \times n}$
Result: $\boldsymbol{v} \in \mathbb{R}^{n_i}$
for $i \leftarrow 1$ to $n_i$ do
    $\boldsymbol{D}_s \leftarrow \boldsymbol{D}(:, \boldsymbol{p}_i)$;
    $\boldsymbol{d}_s^{min} \leftarrow \min_{\mathrm{col}}(\boldsymbol{D}_s)$;
    $v_i \leftarrow \mathrm{g}(\boldsymbol{d}_s^{min})$;
end
```

In this work, $\mathrm{g}(\boldsymbol{d}_s^{min})$ is defined as follows:

$$\mathrm{g}(\boldsymbol{d}_s^{min}) = 2\overline{\boldsymbol{d}_s^{min}} + \max(\boldsymbol{d}_s^{min}) \tag{6}$$

where $\overline{\boldsymbol{d}_s^{min}}$ is the mean value of $\boldsymbol{d}_s^{min}$. Thus, the GA would seek to generate an individual that minimizes the maximum distance from any node to its closest sensor in the network, while also considering the mean of closest distances to avoid outliers. This selection has been found suitable for localization in WDNs, where the effect of a leak is generally more apparent at the nodes closest to the leak.

2.4 Selection of individuals

The GA requires selecting a couple of individuals from the population in order to combine them, pursuing the generation of a high-scoring descendant. Thus, the selection process design is a vital step during the development of the GA. The selection procedure for this work is described by Algorithm 2.

The main idea behind Algorithm 2 is to use the individuals evaluation score to derive probabilities of selection, so that the lower the score (and hence, the minimized distance metric at Eq. (6)), the higher the selection probability. To this end, the score vector $\boldsymbol{v}$ is normalized to fit the range of the $\mathrm{rand}(\cdot)$ function, i.e., $\hat{v}_i \in [0, 1]$. It is then sorted in ascending order by the $\mathrm{sort}(\cdot)$ function (it provides two outputs: the sorted vector, $\boldsymbol{y}$, and the indices of the original one that perform this specific sorting, $\boldsymbol{o}$). Vector $\boldsymbol{y}^{inv}$, which is the result of $\mathbf{1}_{n_i} - \boldsymbol{y}$ (where $\mathbf{1}_{n_i} \in \mathbb{R}^{n_i}$ is a vector whose components are all one), would be a vector whose values are lower when the probability of selection is higher, and the best individual would always be selected, as its value in $\boldsymbol{y}^{inv}$ is 0 (an offset parameter k_s can be defined by the user to avoid this). The selection process is embedded into a loop that checks if two individuals have already been selected once the population has been completely considered.

Algorithm 2: Selection of individuals

```
Data: $\boldsymbol{v} \in \mathbb{R}^{n_i}$, $k_s \in \mathbb{R}$
Result: $\boldsymbol{j}^{sel} \in \mathbb{R}^2$
$\hat{\boldsymbol{v}} \leftarrow \boldsymbol{v}/\max(\boldsymbol{v})$;
$\{\boldsymbol{y}, \boldsymbol{o}\} \leftarrow \mathrm{sort}(\hat{\boldsymbol{v}})$;
$\boldsymbol{y} \leftarrow \boldsymbol{y} - \mathbf{1}_{n_i}(y_1 + k_s)$;
$m \leftarrow 0$;
while $m < 2$ do
    for $i \leftarrow 1$ to $n_i$ do
        $r \leftarrow \mathrm{rand}()$;
        if $y_i^{inv} \leq r$ and $m < 2$ then
            $j_m^{sel} \leftarrow o_i$;
            $m \leftarrow m + 1$;
        end
    end
end
```

2.5 Combination of individuals

Once two individuals are selected, they can be combined to obtain a descendant. The combination strategy must consider the characteristics of sensor placement problem and the distance-based metric to minimize. In this work, the combination operation has been designed as illustrated by Algorithm 3.

The operation starts by extracting the two selected individuals (with indices $\boldsymbol{j}^{sel}$) from the population $\boldsymbol{P}$, achieving $\boldsymbol{P}_{\boldsymbol{j}^{sel}} \in \mathbb{R}^{n \times 2}$. This matrix is converted from its binary form to a node-index matrix form using the function $\mathrm{b2v}(\cdot)$, generating a new matrix $\boldsymbol{X}^{sel} \in \mathbb{R}^{n_s \times 2}$ whose columns store the indices of the 1-valued elements of the corresponding columns of $\boldsymbol{P}_{\boldsymbol{j}^{sel}}$. Matrix $\boldsymbol{X}^{sel}$ would be used to keep track of the sensors in the predecessor individuals that have not yet being considered during the combination process.

The first stage of the combination process checks if there are common sensors to both individuals, which must be kept considering that they help the individuals obtain a high score. Thus, the elements of the descendant vector corresponding to those sensors are provided a value of 1. Besides, the considered sensors are removed from $\boldsymbol{X}^{sel}$ by the $\mathrm{del}_{\mathrm{list}}(\cdot,\cdot)$ function, so that they are not used during the next stage.

The second phase consists of an iterative process that generates a new sensor location from two current locations, one from each predecessor individual, repeating this process until the new individual has n_s sensors (note that the $\mathrm{num}_{\mathrm{el}}(\cdot)$ returns the number of elements in the case of vectors and the length in the case of matrices). First, one of the predecessors is randomly

Algorithm 3: Combination of individuals

```
Data: $\boldsymbol{j}^{sel} \in \mathbb{R}^2$, $\boldsymbol{P} \in \mathbb{R}^{n \times n_i}$, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Result: $\boldsymbol{p}^{new} \in \mathbb{R}^n$
$\boldsymbol{X}^{sel} \leftarrow \text{b2v}(\boldsymbol{P}_{\boldsymbol{j}^{sel}})$;
for $i \leftarrow 1$ to $n$ do
    if $p_{ij_1^{sel}} = 1$ and $p_{ij_2^{sel}} = 1$ then
        $p_i^{new} \leftarrow 1$;
        $\boldsymbol{X}^{sel} \leftarrow \text{del}_{\text{list}}(\boldsymbol{X}^{sel}, v_i)$;
    end
end
while $\text{num}_{\text{el}}\big(\text{b2v}(\boldsymbol{p}^{new})\big) < n_s$ do
    $r_{row} \leftarrow \text{randi}\big(\text{num}_{\text{el}}(\boldsymbol{X}^{sel})\big)$;
    $r_{col} \leftarrow \text{randi}(2)$;
    $v_o \leftarrow x^{sel}_{r_{row} r_{col}}$;
    $j \leftarrow \text{set}_{\text{dif}}(\{1, 2\}, r_{col})$;
    for $i \leftarrow 1$ to $\text{num}_{\text{el}}(\boldsymbol{X}^{sel})$ do
        $d_i^{sp} \leftarrow \text{f}(\mathcal{G}, v_o, v_{x^{sel}_{ij}})$;
    end
    $i^* \leftarrow \text{argmin}(\boldsymbol{d}^{sp})$;
    $\boldsymbol{c} \leftarrow \text{argf}(\mathcal{G}, v_o, v_{x^{sel}_{i^* j}})$;
    $v^{new} \leftarrow c_{\text{round}\left(\text{num}_{\text{el}}(\boldsymbol{c})/2\right)}$;
    if $p^{new}_{v^{new}} = 0$ then
        $p^{new}_{v^{new}} \leftarrow 1$;
        $\boldsymbol{X}^{sel} \leftarrow \text{del}_{\text{list}}(\boldsymbol{X}^{sel}, \{v_o, v_{x^{sel}_{i^* j}}\})$;
    end
end
```

selected, to then randomly choose a sensorized node from the corresponding column of $\boldsymbol{X}^{sel}$, achieving v_o (the random indices, i.e., r_{row} and r_{col} are achieved by means of the randi($\cdot$) function, which generates a random natural number from 1 to the input). The shortest-path distance from v_o to the stored nodes at the remaining column (denoted as j and obtained using the set$_{\text{dif}}(\cdot, \cdot)$ function, which returns the values of the first entry that are not present on the second one) are computed using the function f$(\cdot, \cdot, \cdot)$ at Eq. (5). Then, the path from v_o to the closest node of the other column of $\boldsymbol{X}^{sel}$, i.e., $v_{x^{sel}_{i^* j}}$ ($x^{sel}_{i^* j}$ is the component of X^{sel} from row i^* and column j), is obtained. Note that argmin($\cdot$) returns the index of the minimum value of the input vector, and argf($\cdot$,$\cdot$,$\cdot$) provides the shortest path corresponding to the shortest-path distance computed by f($\cdot$,$\cdot$,$\cdot$).

The new sensor location v_{new} would be selected as the middle node of the obtained path. However, v_{new} is accepted, and thus its corresponding origin nodes are deleted from $\boldsymbol{X}^{sel}$, if and only if v_{new} was not already included in the new individual, to avoid that $\boldsymbol{p}^{new}$ ends up with less than n_s sensors.

2.6 *Mutation of individuals*

The final operation that can be performed over a new individual is the mutation. Again, this process must be adapted to meet the requirements of the sensor placement problem. The mutation function used in this work is represented in Algorithm 4.

Basically, a random sensorized node is deselected, and another random not-metered node is selected as sensor location. Additionally, the existence of user-defined sensors is considered, preventing their deselection.

Algorithm 4: Mutation of an individual

```
Data: $\boldsymbol{p}^{new} \in \mathbb{R}^n$, $\mathcal{S}_{ud}$
Result: $\boldsymbol{p}^{new} \in \mathbb{R}^n$
$\boldsymbol{x}^{new} \leftarrow \text{b2v}(\boldsymbol{p}^{new})$;
$\boldsymbol{x}^{dif} \leftarrow \text{set}_{\text{dif}}(\{1, 2, ..., n\}, \boldsymbol{x}^{new})$;
$r \leftarrow \text{randi}(n_s)$;
$r_d \leftarrow \text{randi}\big(\text{num}_{\text{el}}(\boldsymbol{x}^{dif})\big)$;
while $x_r^{new} \in \mathcal{S}_{ud}$ do
    $r \leftarrow \text{randi}(n_s)$;
end
$p^{new}_{x_r^{new}} \leftarrow 0$;
$p^{new}_{x^{dif}_{r_d}} \leftarrow 1$;
```

2.7 *Preliminary convergence*

The preliminary convergence is achieved once one of the following two events has occurred:

(1) All the individuals of the population are equal, and the solution is a local optimal.
(2) The individuals are different, but they have remained the same for a defined number of iterations.

These events would raise the mutation probability p, thus increasing the probability of generation of new individuals (the preliminary conv. counter cnt_p is also incremented). If the preliminary convergence gets disrupted, both p and cnt_p are set to their initial values. However, if this state holds, there are two thresholds that trigger extra measures depending on cnt_p:

i. First, if $cnt_p = thr_1$, NS-IC is applied to the population to rapidly generate a better set of individuals, if possible. This measure is especially useful for the second case of preliminary convergence (the most common), as each individual could produce a different outcome.
ii. Second, if the counter reaches thr_2, the PDI-P is used. This aggressive measure maintains the best individual and substitutes the rest of individuals of the population by new random individuals (NS-IC is again executed).

2.8 *Final convergence*

The use of GAs does not guarantee to achieve a globally optimal solution. Thus, a mechanism to stop the operation, after some conditions are met, is necessary. In this work, two stopping criteria have been selected:

(1) The cnt_p counter surpasses thr_2 a certain number of times.
(2) The amount of GA iterations exceeds an user-defined limit.

Nevertheless, different criteria may be defined by the user depending on their requirements or implementations.

2.9 *NS-IC*

An iterative clustering scheme has been devised to complement the GA-based approach by enhancing incoming individuals, considering the proximity from the their encoded sensors to the network nodes. The NS-IC method is described in Algorithm 5.

At each iteration, NS-IC starts computing the score of the input/current individual by means of the evaluation function, i.e., eval$_{\text{func}}(\cdot)$ (described in Section 2.4, and considering the distance metric in Eq. 6); to then calculate the closest sensor to

Algorithm 5: NS-IC

```
Data: $\boldsymbol{p}^{in} \in \mathbb{R}^n, \boldsymbol{D} \in \mathbb{R}^{n \times n}, \mathcal{S}_{ud}$
Result: $\boldsymbol{p}^{out} \in \mathbb{R}^n$
$\boldsymbol{n}^{in} \leftarrow \mathrm{b2v}(\boldsymbol{p}^{in})$;
$f^{conv} \leftarrow 0$;
while $f^{conv} = 0$ do
    $v^{in} \leftarrow \mathrm{eval}_{\mathrm{func}}(\boldsymbol{p}^{in}, \boldsymbol{D})$;
    $\boldsymbol{s}^{near} \leftarrow \mathrm{argmin}_{\mathrm{col}}(\boldsymbol{D}(:, \boldsymbol{p}^{in}))$;
    for $i \leftarrow 1$ to $n_s$ do
        if $n_i^{in} \notin \mathcal{S}_{ud}$ then
            $\mathscr{C}^i \leftarrow \{j \in \mathbb{N} | s_j^{near} = i\}$;
            $j^{cent} \leftarrow \mathrm{argmin}_{\mathrm{orig}}(\max_{\mathrm{col}}(\boldsymbol{D}_{\mathscr{C}^i}))$;
            $p_{j^{cent}}^{out} \leftarrow 1$;
        else
            $p_{n_i^{in}}^{out} \leftarrow 1$;
        end
    end
    $\boldsymbol{v}^{out} \leftarrow \mathrm{eval}_{\mathrm{func}}(\boldsymbol{p}^{out}, \boldsymbol{D})$;
    if $\boldsymbol{p}^{out} = \boldsymbol{p}^{in}$ or $\boldsymbol{v}^{out} > \boldsymbol{v}^{in}$ then
        $f^{conv} \leftarrow 1$;
    else
        $\boldsymbol{p}^{in} \leftarrow \boldsymbol{p}^{out}$;
    end
end
```

every node of the network (note that $\mathrm{argmin}_{\mathrm{col}}(\cdot)$ is analogue to $\mathrm{min}_{\mathrm{col}}(\cdot)$ from Section 2.4, but returning the index of the minimum value). Thus, for each sensor i, the nodes whose nearest sensor is i are grouped into a cluster (set of nodes). Then, the node whose maximum distance to any other node of the cluster is minimum is calculated, setting this node as the new sensor associated to the cluster (note that $\mathrm{argmin}_{\mathrm{orig}}(\cdot)$ operates like $\mathrm{argmin}(\cdot)$, explained in Section 2.5, but returning the index in terms of the original set of nodes, i.e., the complete set of nodes of the network). The computation must consider user-defined sensors to maintain them. In this way, the output individual is generated, the output score is computed and the convergence is evaluated, considering two stopping criteria: input and output individuals are equal, or the score is not improved.

3. CASE STUDY

In this work, the L-TOWN benchmark has been considered as a case study to assess the performance of the methodology, due to its open-access availability and its use by several research groups in the BattLeDIM2020 (Vrachimis et al., 2020) competition. A diagram of the complete network can be found in Figure 2.

The network is divided into three zones depending on the elevation of the nodes: Area B for nodes below 16 meters, Area A for nodes between 16 and 48 meters and Area C for nodes above 48 meters. Besides, there are two reservoirs feeding the network with water through Area A, as well as a tank that stores incoming water and supplies it to Area C (Areas A and B are separated by a pressure regulation valve - PRV).

In order to assess the suitability of the proposed method, the set of sensors obtained from its application in L-TOWN is compared to the original set of sensors from BattLeDIM2020, placed by the organizers by means of a method that maximized the collective sensitivity of the sensors to any possible leak.

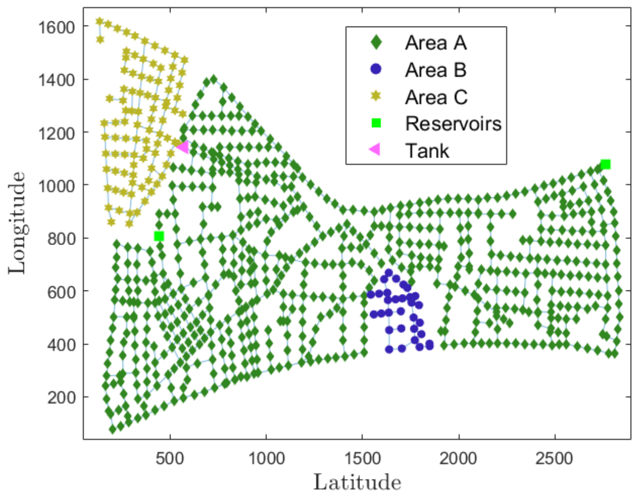

Fig. 2. Schematic representation of L-TOWN.

However, the comparison of performance is hindered by the differences in hydraulic behaviour among the different Areas (due to the tank and PRV operation) and the minimal number of pressure sensors at Areas B (1) and C (3). Thus, the comparison is considered for Area A, where 29 original sensors are placed.

The set of original sensors, $\mathcal{S}_{orig}$, as well as the set of GA-generated sensors, $\mathcal{S}_{GA}$, are located over Area A in Fig. 3. The GA-generated sensors are obtained with $n_s = 29$, $n_i = 5$ individuals, $p = 0.1$, $thr_1 = 70$ and $thr_2 = 100$.

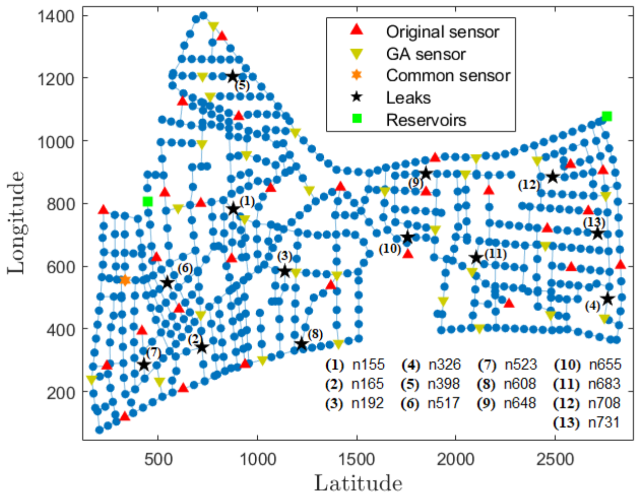

Fig. 3. Sensor locations over Area A.

4. RESULTS

The comparison between $\mathcal{S}_{orig}$ and $\mathcal{S}_{GA}$ is performed by means of an assessment of the leak localization results yielded by both sensor configurations, considering a common leak localization method. In this work, the data-driven approach explained in Romero et al. (2021) is selected to perform this task.

The BattLeDIM2020 challenge provided hydraulic measurements datasets containing several leaks. However, they can not be used for the comparison because the pressure measurements are only supplied for $\mathcal{S}_{orig}$. Thus, a subset of leaks from the 2019 dataset is selected to be scattered throughout Area A and replicated by EPANET (Rossman, 2000) simulations to retrieve the pressure at all the nodes. Their location over Area A is represented in Fig. 3. Note that several factors have been considered during the dataset generation:

- The peak leak sizes are kept from the original dataset, but the leaks are considered to be constant. Each leak is simulated for 24 hours, with a time-step of 5 minutes.
- A 5% uncertainty is applied over pipe diameters and roughness, whereas a 1% is applied to the demand patterns. Moreover, the sensors precision is limited to ±0.01 m, which is also considered in the original dataset.
- The leak localization method selects nodes as potential leak origins, so the generated dataset locates each leak at one of the connected nodes by the corresponding leaky pipe from the original dataset.

Leak localization results for both sensor placement solutions are presented in Table 1, using the following metrics:

- L_{orig} and L_{gen}: they respectively encode the pipe where the leak appeared in the original dataset and the selected node to represent that leak on the generated EPANET dataset.
- $dist_{leak}$: it corresponds to the pipe distance in meters from the leak to the best candidate (shortest-path).
- $nnodes_{leak}$: it shows the pipe distance in number of nodes between the leak to the best candidate (shortest-path).

Table 1. Leak localization performance results

L_{orig}	L_{gen}	$dist_{leak}$		$nnodes_{leak}$	
		$\mathcal{S}_{orig}$	$\mathcal{S}_{GA}$	$\mathcal{S}_{orig}$	$\mathcal{S}_{GA}$
p123	n155	159.55	177.21	2	3
p586	n165	220.35	285.27	3	6
p142	n192	440.45	140.22	8	3
p193	n326	483.06	85.64	10	1
p331	n398	418.01	202.34	7	3
p514	n517	233.09	415.97	3	8
p523	n523	107.57	413.75	1	7
p653	n608	368.99	415.29	7	8
p710	n648	222.31	205.41	3	3
p721	n655	312.75	255.00	5	5
p762	n683	268.52	295.73	6	6
p800	n708	94.45	114.32	1	2
p827	n731	152.48	292.61	2	5
Mean		267.81	253.75	4.46	4.61
Max		483.06	415.97	10	8
Min		94.45	85.64	1	1

Several upshots can be extracted from the presented results:

i. Both $\mathcal{S}_{GA}$ and $\mathcal{S}_{orig}$ produce similar results (a difference of 14.06 meters/0.15 nodes) considering the average values. However, the maximum and minimum values are lower for $\mathcal{S}_{GA}$. Thus, the usage of the proposed sensor placement methodology would improve the localization, with the additional advantage of being model-free.

ii. Focusing on each specific result, and considering the leak map at Fig. 3, it can be appreciated how the worst localization indices for each sensor configuration come from leaks at distant nodes to the corresponding set of sensors. This justifies the proposal of a distance-based metric in the evaluation function.

5. CONCLUSIONS

This article presents a model-free sensor placement methodology for WDNs based on an GA-optimization scheme. The individuals are binary vectors encoding the sensor locations, and the processes of the GA-scheme are customized for the sensor placement problem. The algorithm pursues the minimization of a distance-based metric. An iterative clustering technique, NS-IC, is devised to rapidly enhance incoming individuals.

The approach is applied over a well-known case study, L-TOWN from the BattLeDIM2020 challenge, comparing the sensor placement from the competition with a sensors set generated using the proposed method. The comparison is carried out by means of the leak localization performance of a selected method using each corresponding sensor configuration. The results show that the generated sensors set slightly improves the performance of the original set, with the advantage of its model-free origin.

Thus, the method is highly interesting for water utilities due to their growing interest on data-driven leak management approaches. Moreover, the evaluation function allows the user to define the aim to be sought by the sensor placement process, allowing to customize the solution to the needs of the user.

REFERENCES

Blesa, J., Nejjari, F., and Sarrate, R. (2015). Robust sensor placement for leak location: analysis and design. *Journal of Hydroinformatics*, 18(1), 136–148.

Casillas, M.V., Puig, V., Garza-Castanón, L.E., and Rosich, A. (2013). Optimal sensor placement for leak location in water distribution networks using genetic algorithms. *Sensors*, 13(11), 14984–15005.

Festa, P. (2006). Shortest path algorithms. In M. Resende and P. Pardalos (eds.), *Handbook of Optimization in Telecommunications*, 185–210. Springer, Boston, MA.

Khorshidi, M.S., Nikoo, M.R., Taravatrooy, N., Sadegh, M., Al-Wardy, M., and Al-Rawas, G.A. (2020). Pressure sensor placement in water distribution networks for leak detection using a hybrid information-entropy approach. *Information Sciences*, 516, 56–71.

Liemberger, R. and Wyatt, A. (2019). Quantifying the global non-revenue water problem. *Water Supply*, 19(3), 831–837.

Peng, S., Cheng, J., Wu, X., Fang, X., and Wu, Q. (2022). Pressure sensor placement in water supply network based on graph neural network clustering method. *Water*, 14(2), 150.

Pérez, R., Puig, V., Pascual, J., Peralta, A., Landeros, E., and Jordanas, L. (2009). Pressure sensor distribution for leak detection in barcelona water distribution network. *Water science and technology: water supply*, 9(6), 715–721.

Quiñones-Grueiro, M., Verde, C., and Llanes-Santiago, O. (2019). Multi-objective sensor placement for leakage detection and localization in water distribution networks. In *2019 4th Conference on Control and Fault Tolerant Systems (SysTol)*, 129–134. IEEE.

Romero, L., Puig, V., Cembrano, G., Blesa, J., and Meseguer, J. (2021). A fully data-driven approach for leak localization in water distribution networks. In *2021 European Control Conference (ECC)*, 1851–1856. IEEE.

Rossman, L.A. (2000). EPANET 2: Users Manual.

Santos-Ruiz, I., López-Estrada, F.R., Puig, V., Valencia-Palomo, G., and Hernández, H.R. (2022). Pressure sensor placement for leak localization in water distribution networks using information theory. *Sensors*, 22(2), 443.

Vrachimis, S.G., Eliades, D.G., Taormina, R., Ostfeld, A., Kapelan, Z., Liu, S., Kyriakou, M., Pavlou, P., Qiu, M., and Polycarpou, M.M. (2020). BattLeDIM: Battle of the Leakage Detection and Isolation Methods.