Pressure Sensor Placement for Leak Localization Using Simulated Annealing with Hyperparameter Optimization

I. O. Morales-González[1], I. Santos-Ruiz[1,*], F. R. López-Estrada[1], V. Puig[2]

***Abstract*— This paper presents a machine learning method for optimal pressure sensor placement in water distribution networks. The proposed approach considers annealing metaheuristics, and it is focused on the optimal placement of the sensors to perform leak localization. Unlike other works, this method considers a limited number of sensors to be placed and some restrictions on critical nodes that can be excluded or preselected. The approach is based on minimizing a cost function; this cost function is assigned as the leak location error, which varies depending on the subset of nodes where the sensors are assigned and the configuration of the leak location method. A leak localization technique based on k-NN classifiers was used, and during the minimization of the cost function, classifier hyperparameters were simultaneously optimized. The proposed method was tested on the Hanoi water distribution network programmed in MATLAB.**

I. Introduction

Losses caused by leaks are one of the main problems in managing drinking water systems. Globally, about one-third of chemically treated water is lost due to leakage before reaching final consumers [1]. Even if they are small, these leaks generate considerable losses when they remain unrepaired for long periods. For such reason, it is of paramount importance to detect and locate them as soon as possible [2]. Leak diagnosis methods depend mainly on two factors: the quality of the available data and the algorithms. In general, there is a limited number of pressure or flow-rate measurements that are compared with nominal conditions of a hydraulic reference model to generate residuals [3]. Then, by evaluating the specific characteristics of these residuals, it is possible to diagnose the leaks. However, the algorithms are minimal for the quantity and quality of the available data. Ideally, it is possible to place sensors at each node of the WDN, but the installation, monitoring, and maintenance of such sensors is not possible [4]. Even the most instrumented networks have a minimal number of sensors. Therefore, it is of primary importance to select appropriately the network nodes where the sensors will be placed since an optimal distribution of the available sensors will facilitate pressure monitoring and leak localization.

The Sensor Placement Problem (SPP) for WDNs has been addressed since the last century, mainly for water quality control and pollutant detection [5]–[7]. Recently, SPP has also focused on detecting cyberattacks [8]. When the sensor placement is focused on locating leaks, the optimal node set is typically computed by minimizing a cost function associated with the average location error. Sensor placement is also often calculated by maximizing a leak isolability index. In the literature, there are different methods for sensor placement based on leak location error minimization. Some of them consider metaheuristic such as genetic algorithms [9], [10] or particle swarm optimization [11]. Other works are focused on minimizing the error in the leak detection based on the information theory [4], evolutionary algorithms [12], decision-making techniques, such as DEMATEL combined with fuzzy logic [13], among others. Recently, some works propose multi-target sensor placement algorithms to minimize both the error in leak detection and the error in leak localization [14], [15].

A characteristic of the works cited above is the common assumption that the sensors could be placed in any network node. However, in practice, physical criteria can restrict or prioritize some nodes over others. For example, in critical locations near a hospital or governmental buildings, it is required to monitor pressure levels even if the information is not relevant enough for the leak diagnosis algorithms. Furthermore, in partially instrumented networks, it is desired to take advantage of the sensors that are already installed to reduce the economic cost associated with new installations.

Another characteristic of the works reported in the literature is that only the placement of the sensors can increase the efficiency of some leak localization method, and the hyperparameters do not need to be modified. However, a set of detection nodes that leads to accurate leak location with specific hyperparameter values can also lead to poor performance with another hyperparameter. In this work, the proposed method considers tuning the algorithm's hyperparameters simultaneously with sensor placement. To the best of the author's knowledge, these problems remain open and are the subject of study in this paper.

This work proposes an optimal sensor placement method based on simulated annealing (SA). Such an algorithm searches for the subset of nodes that minimizes a cost function related to the accuracy leak detection employing a k-NN classifier. In addition, we propose a methodology for tuning the hyperparameters, which improves the leak localization. Furthermore, the method considers some restrictions on prioritizing some nodes, such as pre-installed sensors or critical nodes, to increase the applicability. The method effectiveness is tested in the well-accepted benchmark of the Hanoi Network.

[1]TURIX-Dynamics Diagnosis and Control Group, Tecnológico Nacional de México / Instituto Tecnológico de Tuxtla Gutiérrez. Carr. Panamericana S/N, 29050 Tuxtla Gutiérrez (Mexico).

[2] Department of Automatic Control (ESAII), Universitat Politècnica de Catalunya (UPC), Rambla de Sant Nebridi 10, 08222 Terrassa (Spain).

* Corresponding author: idelossantos@ittg.edu.mx

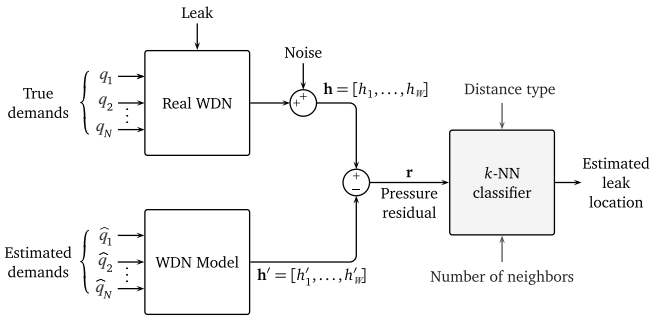

Fig. 1: Leak localization scheme.

II. Background on Leak Localization

The leak localization method used in this work is based on the approach presented in [16], where residual pressures are implemented as input data to a classifier algorithm that returns an estimation of the leak location. Fig. 1 summarizes the methodology used in this work. Pressure residuals consider the difference between the pressures (h) obtained by sensors and the estimated pressures (h') obtained from a hydraulic network model in leak-free conditions simulated in EPANET-2 [17]. This model represents the stationary hydraulic behavior of a well-calibrated WDN. However, it is worth mentioning that the model is fed with estimated demands ($\widehat{q}_1, \widehat{q}_2, \ldots, \widehat{q}_N$) since in practice, the real demands ($q_1, q_2, \ldots, q_N$) are not measured. In real WDN, the estimated demands are typically calculated based on the total demand of the network as a distribution at the nodal level according to historical consumption records [10]. This way of estimating the demand is also considered for some academic benchmark such as the Hanoi network. Also, this work assumes that the data are significantly affected by model uncertainties and sensor noise. It is assumed that the method is tested under the minimum night flow regime when the uncertainty in the demands is minimal and with only one leak. Simultaneous leaks are not considered and are beyond the scope of this paper. Finally, it is considered that the number of available sensors is less than the number of nodes.

Leak localization is performed using a k-NN classifier based on supervised learning. In order to train the classifier, first collects several samples associated with each class (training data), each sample containing a certain number of features of the events or objects to be classified. When a new sample is submitted for classification, distance measures are used as a similarity metric to associate the queried sample to the most similar class. In leak localization, a class is used for each possible leak location, and the leaks are associated with the closest node within the network, so that the number of classes to identify equals the number of nodes. This is explained in detail in [2]. When a sample $\mathbf{r}$ is queried, the proximity between this sample and the training data within the input space is measured, the k samples closest to $\mathbf{r}$ perform a vote. Consequently, the class with the most votes (the most frequent class among neighbors) is assigned to the queried sample. The input samples are vector-shaped, and each element provides information about its position in the input space. Therefore, the similarity between two samples $\mathbf{r} = [r_1, r_2, \ldots, r_n]$ and $\mathbf{r}' = [r'_1, r'_2, \ldots, r'_n]$ is measured as a distance between points. Although, the metric used to quantify the similarity does not necessarily have to be a Euclidean distance [18]. Table I shows the most common distance metrics for distance-based classifier. The distance type (later named δ) and the number of nearest neighbors (k) are the two hyperparameters of k-NN that we proposed to be optimized.

TABLE I: Types of distance.

Number	Name	Definition
1	Manhattan	$d(\mathbf{r}, \mathbf{r}') = \sum_{j=1}^{n} \lvert r_j - r'_j \rvert$
2	Euclidean	$d(\mathbf{r}, \mathbf{r}') = \Big(\sum_{j=1}^{n} \lvert r_j - r'_j \rvert^2 \Big)^{1/2}$
3	Chebychev	$d(\mathbf{r}, \mathbf{r}') = \max_j \lvert r_j - r'_j \rvert$
4	Cosine	$d(\mathbf{r}, \mathbf{r}') = 1 - \dfrac{\mathbf{r} \cdot \mathbf{r}'}{\lVert \mathbf{r} \rVert \, \lVert \mathbf{r}' \rVert}$

Considering the limited availability of pressure measurements under leakage and no-leak conditions, pressure residuals obtained from synthetic data are used to train the k-NN classifier. In order to generate training and testing data, the real WDN in Fig. 1 is replaced by the WDN model where the demands are increased to simulate different leak scenarios. For example, to simulate a leak of $8\,\mathrm{l/s}$ at node 3, the demand $\widehat{q}_3$ is increased by $8\,\mathrm{l/s}$ with respect to leak-free conditions. This process is repeated for different nodes and for different leak magnitudes. Half of the simulated data is used for training and the rest for testing/evaluation.

III. Formulation of Sensor Placement Problem

The major problem in sensor placement is the large number of possible combinations to evaluate. The total number of combinations, C, is given by:

$$C = \frac{N!}{S!(N-S)!}, \tag{1}$$

where N is the number of candidate nodes and S is the number of sensors to be placed. According to (1), when the number of sensors is doubled from 3 to 6 in a network containing 30 nodes, the combinations increase up to $14\,625\%$. In the case of doubling the number of nodes, the combinations increase by more than 800%. In the case where both parameters are doubled, the number of combinations increases by more than $1\,200\,000\%$. In a hypothetical case of a network with 60 nodes and 6 sensors to be installed, $C \approx 5 \times 10^7$. If we assume that it would take $0.1\,\mathrm{s}$ to evaluate each possible combination, the total time required would be 58 days. However, there are WDNs of $1\,200$ nodes or more and 10 or more sensors must be placed, where the combinations amount to 1.6×10^{24}, with computation times of 5.2×10^{15} years.

This work aims to place a certain number of pressure sensors in a WDN and the hyperparameters of a k-NN classifier to maximize its efficiency when used as a leak localization method. Since a way to evaluate the placement of sensors together with the hyperparameters of the classifier is needed, a vector is described as:

$$S = \{s_1, s_2, \ldots, s_W, k, \delta\}, \tag{2}$$

where S is a vector containing the sensor locations and the hyperparameters of the classifier, s_i is the index of the node where a sensor is installed, k is the number of nearest neighbors, and δ is the type of distance to use in k-NN. The optimality of a sensor placement is defined from the leak location error. According to [9], a suitable error-index is defined as:

$$\varepsilon_i(S) = \begin{cases} d_{ij}/d_{\max} & \text{if } d_{ij} < d_{\max}, \quad i = 1, 2, \ldots, l, \\ 1 & \text{otherwise,} \end{cases} \tag{3}$$

where d_{ij} is the shortest path distance between the node where the leak is detected (i) and its true location (j), $d_{\max}$ is a predefined threshold that defines the maximum allowable distance error, and l is the number of simulated leaks. The d_{ij} value is computed according to Floyd-Warshall's algorithm [19]. Since the objective is to maximize the isolability of leakage at all nodes in the network, the error rate that takes into account all node leaks is calculated as:

$$\overline{\varepsilon}(S) = 1 - \sum_{i=1}^{l} \frac{\varepsilon_i(S)}{l}, \tag{4}$$

where $\overline{\varepsilon}(S)$ provides the fraction of correctly located leaks for a given configuration S, i.e., $\overline{\varepsilon}(S) = 1$ when all l leaks were correctly located, and $\overline{\varepsilon}(S) = 0$ when all leaks were located beyond the allowable error distance.

IV. Sensor Placement Methodology

The Simulated Annealing (SA) algorithm is considered here, which is a metaheuristic optimization technique inspired by the metal annealing processing [20]. Unlike deterministic optimization methods (e.g. gradient-based methods), the main advantage of SA is its ability to avoid being trapped in local optimal. Metaphorically, SA is like dropping some bouncing balls over a landscape, and as the balls bounce and lose energy, they settle to some local minimum. But, if the balls are allowed to bounce long enough and lose energy slowly enough, some of the balls will eventually fall into the globally lowest locations. Essentially, SA is a search along a Markov chain, which converges to the global minimum under the right conditions.

The SA algorithm emulates the physical process of heating a material and then slowly lowering the temperature to decrease its defects, thus minimizing the energy of the system. In computational implementations for minimizing a function, "temperature" is a numerical variable used to control an iterative search for the minimum. In each iteration, a new point is randomly generated. The extent of the search (the distance of the new point from the current point) is based on a probability distribution with a scale proportional to temperature. The algorithm accepts all new points that lower the cost function, but also, with some probability, the points that raise the cost. By accepting points that raise the cost, the algorithm avoids being trapped in local minima and is able to explore globally for more possible solutions. The temperature is systematically lowered as the algorithm progresses, narrowing the search range to converge to a minimum. The proposed SA algorithm is described as follows:

1) Define an initial temperature (T_{init}), a random solution is generated as the current solution (S_{current}) and determine its cost ($\overline{\varepsilon}(S_{\text{current}})$).
2) By perturbing the current solution, a new candidate solution (S_{cand}) is chosen and its cost ($\overline{\varepsilon}(S_{\text{cand}})$) is calculated. The new solution (S_{cand}) is accepted as the current solution if $\overline{\varepsilon}(S_{\text{cand}}) > \overline{\varepsilon}(S_{\text{current}})$. Otherwise, the solution will be accepted depending on a probability function (PF) determined by the current temperature.
3) Repeat Step 2 until reaching equilibrium, which means that the probability of finding a better solution with current temperature is very low. The number of repetitions (L) is defined as a Markov chain.
4) The temperature is decreased and the process is repeated until a final temperature (T_{end}) is reached at which the probability of improving the cost of the solution is approximately zero, at which point the system is said to have reached thermal equilibrium.

The last candidate solution is assumed as the optimal solution. The pseudocode in Algorithm 1 formally expresses the steps described above.

Algorithm 1: SA-based sensor placement.

Input: $T_{\text{init}}, T_{\text{end}}, L, S_{\text{current}}, \alpha, \beta$
$T \leftarrow T_{\text{init}}$
$\overline{\varepsilon}(S_{\text{current}}) \leftarrow \mathsf{evaluate}(S_{\text{current}})$
$S_{\text{best}} \leftarrow S_{\text{current}}, \ \overline{\varepsilon}(S_{\text{best}}) \leftarrow \overline{\varepsilon}(S_{\text{current}})$
while $T > T_{\text{end}}$ **do**
  for k **from** 1 **to** L **do**
    $S_{\text{cand}} \leftarrow \mathsf{disturbance}(S_{\text{current}})$
    $\overline{\varepsilon}(S_{\text{cand}}) \leftarrow \mathsf{evaluate}(S_{\text{cand}})$
    $P_a \leftarrow \text{PF}(\overline{\varepsilon}(S_{\text{cand}}), \overline{\varepsilon}(S_{\text{current}}), T)$
    if $P_a > \text{random}[0, 1]$ **then**
      $S_{\text{current}} \leftarrow S_{\text{cand}}, \ \overline{\varepsilon}(S_{\text{current}}) \leftarrow \overline{\varepsilon}(S_{\text{cand}})$
      if $\overline{\varepsilon}(S_{\text{current}}) > \overline{\varepsilon}(S_{\text{best}})$ **then**
        $S_{\text{best}} \leftarrow S_{\text{current}}, \ \overline{\varepsilon}(S_{\text{best}}) \leftarrow \overline{\varepsilon}(S_{\text{current}})$
      end
    end
  end
  $T \leftarrow \alpha T, \ L \leftarrow \beta L$
end
return S_{best}

It should be clarified that although Algorithm 1 optimizes the solution S, it is actually a pre-solution $\widehat{S} = \{\widehat{s}_1, \widehat{s}_2, \ldots, \widehat{s}_M\}$, because the solution to be optimized by

the algorithm can not include the nodes that already contain a sensor, excluded nodes and prioritized or pre-selected nodes. Therefore, M is the number of sensors that do not have a defined location.

In the disturbance subroutine, the pre-solution to be perturbed is modified by changing one of its elements by an element in the neighboring set. However, for Algorithm 1, there are three sets of neighborhoods depending on which element will be modified. If the element to modify is a node, the possible new element to replace it will be taken from the set $\mathcal{A}$; if the element to modify will be the number of nearest neighbors it will be replaced by an element in the set $\mathcal{B}$, and finally, if the element to modify will be the distance type (δ) it will be replaced by an element in the set $\mathcal{C}$. This is explained as:

$$\widehat{S} = \{\underbrace{\widehat{s}_1, \widehat{s}_2, \ldots, \widehat{s}_M}_{\mathcal{A}}, \underbrace{k}_{\mathcal{B}}, \underbrace{\delta}_{\mathcal{C}}\} \tag{5}$$

Set $\mathcal{A}$ in (5) includes all network nodes, except the subsets with pre-assigned and excluded nodes:

$$\mathcal{A} = \{1, 2, \ldots, N\} - \mathcal{I} \cup \mathcal{U}, \tag{6}$$

where N is the number of nodes in the network, $\mathcal{I}$ is the set of nodes where they already have a sensor, and $\mathcal{U}$ is the set of excluded nodes. Set $\mathcal{B}$ is a vector describing the number of nearest neighbors to be used in the classifier, as defined by:

$$\mathcal{B} = \{1, 2, \ldots, l/N\}, \tag{7}$$

where l is the number of leaks simulated in the data generation stage. The set $\mathcal{C}$ is a vector as described in (8) being a vector of 4 elements, each element refers to a distance type used by the classifier. The distance type is represented by an integer, as shown in Table I.

$$\mathcal{C} = \{1, 2, 3, 4\} \tag{8}$$

For example, if the solution $\widehat{S} = \{\widehat{s}_1, \widehat{s}_2, \ldots, \widehat{s}_M, k, 4\}$ contains the number 4 as type of distance, in Table I, the distance number 4 is the cosine distance, which indicates that this configuration will be evaluated with a k-NN classifier with the cosine distance type. In the evaluate subroutine, $\overline{\varepsilon}(S_{\text{cand}})$ is calculated using (4) given the solution S, However, at this stage the solution S to evaluate is the union of the set of nodes with pre-installed sensors and pre-selected nodes $\mathcal{I}$ and the pre-solution $\widehat{S}$, therefore, the solution to be evaluated is the one shown in (9) this guarantees that the best solution found by Algorithm 1 is the best configuration of nodes that best locates leaks in conjunction with the preinstalled and prioritized sensors.

$$S = \mathcal{I} \cup \widehat{S} = \{i_1, \ldots, i_m, \widehat{s}_1, \ldots, \widehat{s}_M, k, \delta\}, \quad i_z \in \mathcal{I}, \tag{9}$$

where m is the sum of the number of pre-selected nodes and the number of nodes with pre-installed sensors, Therefore, $W = M + m$. Subsequently, to calculate the probability that the new solution is accepted or not it is necessary to define the probability function (PF), this returns a value between 0 and 1, PF $= 1$ being the maximum probability guaranteeing acceptance of the new solution, otherwise PF $= 0$ being the rejection of the new solution., but rather a probability function given by:

$$\text{PF}(\overline{\varepsilon}(S_{\text{cand}}), \overline{\varepsilon}(S_{\text{current}}), T) = \exp\left(\frac{\Delta\overline{\varepsilon}}{T}\right), \tag{10}$$

$$\Delta\overline{\varepsilon} = \overline{\varepsilon}(S_{\text{cand}}) - \overline{\varepsilon}(S_{\text{current}}), \tag{11}$$

where $\Delta\overline{\varepsilon}$ is the deterioration on the current solution. Thus, $\Delta\overline{\varepsilon} > 0$ is considered an improvement to the solution. On the other hand, the cooling function used in this work is a geometric evolution presented in [20]. This function decreases the temperature by successive multiplication with a "cooling factor" α, wich can assume values between 0.7 and 0.99, depending on the optimality required in the results, as:

$$T_{n+1} = \alpha\, T_n. \tag{12}$$

Since $\alpha < 1$, the temperature is guaranteed to decrease in each iteration until it is asymptotically zero. On the other hand, for the length of the Markov chain, a variable chain's length (L) is chosen, increasing its size as the temperature decreases, in order to intensify the search for better solutions at lower temperatures. The way in which it is increased is expressed as a function of the initial and final lengths (L_{init} and L_{end}), both proposed by the user, also, it is calculated as a function of α, T_{init} and T_{end}, the increase of the Markov chain is given by:

$$L_{n+1} = \beta\, L_n, \tag{13}$$

where β is given by:

$$\beta = \exp\left(\frac{\ln(L_{\text{end}}) - \ln(L_{\text{init}})}{n}\right), \tag{14}$$

$$n = \frac{\ln(T_{\text{end}}) - \ln(T_{\text{init}})}{\ln(\alpha)}. \tag{15}$$

Both T_{init} and T_{end} are calculated based on a given number of perturbations to an initial solution, T_{init} is selected such that the largest efficiency deterioration found in those perturbations has 25% chance of being accepted as the current solution and T_{end} is selected such that the smallest decrease found in those perturbations has 0.000000001% chance of being accepted as the current solution.

V. Results

The Hanoi WDN was considered for testing the proposed method. This benchmark proposed in [21] has 31 nodes, 36 pipelines, and one reservoir, as shown in Fig. 2. A previous work [10] states that the optimal number of sensors for this network are two by considering the relationship between installation costs, maintenance, and the information obtained from the network. Therefore, as a reference point, we also consider two sensors to be placed in the network.

Different leakage scenarios corresponding to 19 leak magnitudes were simulated on each node of the Hanoi network, totaling 589 leak scenarios, of which 295 scenarios were used for training, and the remaining ones were used for testing.

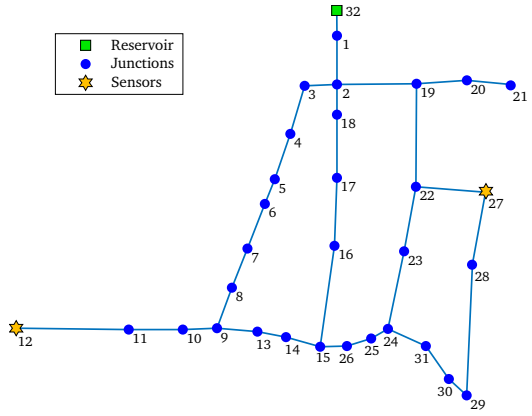

Fig. 2: Hanoi network with a 2-sensor placement.

Leakage was simulated assuming flow rates between $1\,\mathrm{l/s}$ and $10\,\mathrm{l/s}$, which is equivalent to adding 6% to 60% to the estimated minimum network consumption. In addition, to test the robustness of the computed sensor placement, the pressure residuals were intentionally contaminated by a zero-mean Gaussian noise with SNR $= 26\,\mathrm{dB}$.

The results of applying Algorithm 1 on the data of the Hanoi network is shown in Fig. 2, where the optimal locations selected to place two sensors were nodes 12 and 27. In order to maximize the leak isolability using k-NN, the algorithm also determines that for the simulated conditions the best distance metric is the cosine distance and the best number of nearest neighbors is $k = 19$. It can be explained that the algorithm selects all the neighbors corresponding to leaks from the same node by the large amount of noise added to the classifier input. In tests with noise-free data, the optimal number of neighbors was $k = 3$ in most cases. The computing time to find the best solution was $43.54\,\mathrm{s}$, obtaining an objective-function value $\bar{\varepsilon}(S) = 0.623$. It should be noted that the computation time increases geometrically with the number of nodes in the network and with the number of sensors to be placed, so strategies must be designed to reduce the computation time in more complex networks.

The convergence of Algorithm 1 over the number of iterations is shown in Fig. 3. This figure shows the behavior of this algorithm, where a series of solutions are evaluated and accepted as solution to the problem, even if the best solution was found, this is allowed in order to avoid local optima. While the temperature is high the accepted solutions can be worse than the previous one, however, as the control parameter "temperature" decreases, the worst solutions are less likely to be accepted.

Due to the small size of the problem in the case study, it is possible to compute and display the search space as in Fig. 4, where it can be seen that the algorithm managed to find the global maximum.

For the cases of excluded nodes and pre-assigned nodes, two scenarios are evaluated. In the first, nodes 12 and 27 are excluded, this in order to simulate a scenario where the best nodes according to the algorithm are excluded. In the second scenario, a critical sensor has been pre-assigned in

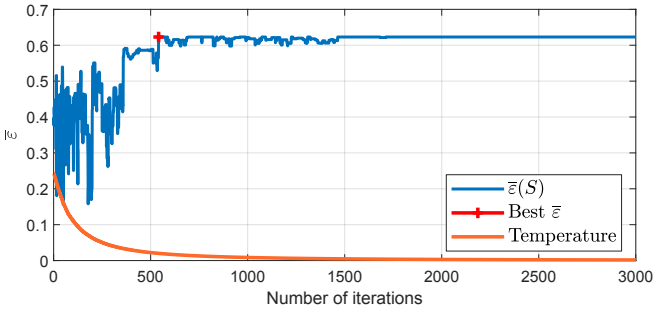

Fig. 3: Convergence of the simulated annealing algorithm.

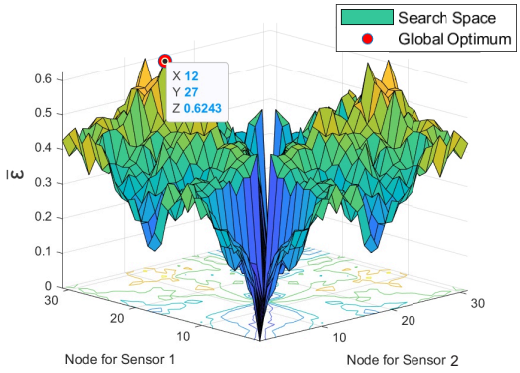

Fig. 4: Search space for placing two sensors.

node 21. The results regarding the best sensor placement and the computation time are shown in Table II. In the first case, the best nodes where to place two sensors are nodes 11 and 29, the hyperparameters (distance type and number of nearest neighbors) are cosine distance and $k = 13$, respectively. In order to verify these results, the solution space for this case was computed and is shown in Fig. 5, where it is demonstrated that both the global maximum and the maximum found by Algorithm 1 match. For the second case, Table II shows that the best locations to place sensors, under the restrictions mentioned above, are nodes 11 and 21, being the latter node where a pre-installed sensor was simulated, therefore, the algorithm evaluated the solutions that contained that node. In this case, the algorithm also returns the cosine distance as the best distance metric and 19 as the best number of nearest neighbors. In order to validate these results, the solution space was also computed and displayed in Fig. 6, where it is observed that the global maximum within the solution space is the same as the one obtained by the algorithm. Both Fig. 5 and Fig. 6 show that all solutions that violate the corresponding restrictions in the solution space have zero cost.

TABLE II: Algorithm 1 test results.

Case	Excluded nodes	Pre-selected nodes	Best node set found	Computing time (s)
1	12,27	None	11,29,13,4	34.11
2	12,27	21	11,21,19,4	42.37

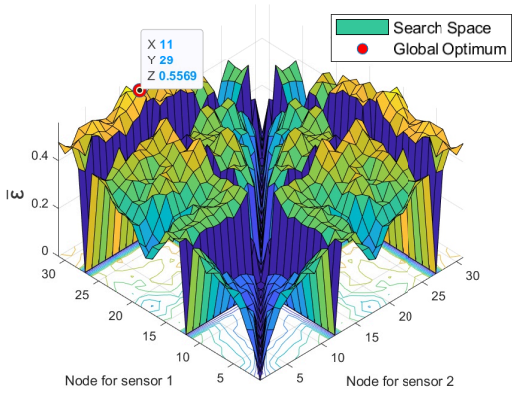

Fig. 5: Search space with two excluded nodes (12 and 27).

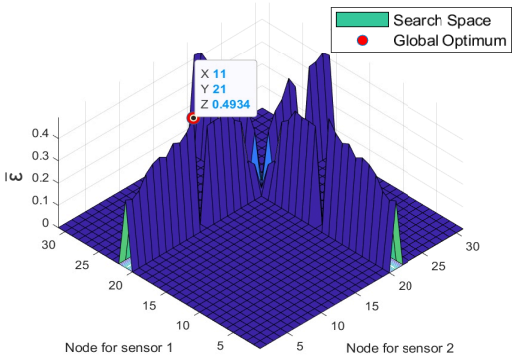

Fig. 6: Search space with two excluded nodes (12 and 27) and one pre-assigned node (node 21).

VI. Conclusions

In this paper, an optimal sensor placement method for water distribution networks has been proposed. The proposed solution considers restrictions on excluded and pre-assigned nodes, also optimizing the hyperparameters of a k-NN classifier for leak localization. The combinatorial optimization problem in sensor placement was solved using a simulated annealing algorithm. The proposed method has been tested in Hanoi network under different leak scenarios. Results show that the SA-based method is able to find the global optimum. Planned future works include optimal placement of heterogeneous sensors and testing on more complex networks.

Acknowledgment

This work has been co-financed by the European Regional Development Fund of the European Union in the framework of the ERDF Operational Program of Catalonia 2014–2020, under the research project 001-P-001643 Agrupació Looming Factory. The National Technological Institute of Mexico (TecNM) has also co-financed this work by granting the research project 11080.21-P.

References

[1] OECD, "Water Governance in Cities," *OECD Studies on Water*, feb 2016.

[2] I. Santos-Ruiz, F.-R. López-Estrada, V. Puig, J. Blesa, and M. Javadiha, "Localización de fugas en redes de distribución de agua mediante k-NN con distancia cosenoidal," in *Congreso Nacional de Control Automático*. Asociación de México de Control Automático (AMCA), 2019.

[3] V. Puig, C. Ocampo-Martínez, R. Pérez, G. Cembrano, J. Quevedo, and T. Escobet, Eds., *Real-time Monitoring and Operational Control of Drinking-Water Systems*. Springer International Publishing, 2017.

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

[5] W. E. Hart and R. Murray, "Review of sensor placement strategies for contamination warning systems in drinking water distribution systems," *Journal of Water Resources Planning and Management*, vol. 136, no. 6, pp. 611–619, 2010.

[6] S. Rathi and R. Gupta, "Sensor placement methods for contamination detection in water distribution networks: A review," *Procedia Engineering*, vol. 89, pp. 181–188, 2014.

[7] Q. Zhang, F. Zheng, Z. Kapelan, D. Savic, G. He, and Y. Ma, "Assessing the global resilience of water quality sensor placement strategies within water distribution systems," *Water research*, vol. 172, p. 115527, 2020.

[8] D. Nikolopoulos, A. Ostfeld, E. Salomons, and C. Makropoulos, "Resilience assessment of water quality sensor designs under cyber-physical attacks," *Water*, vol. 13, no. 5, p. 647, 2021.

[9] M. V. Casillas, V. Puig, L. E. Garza-Castanón, and A. Rosich, "Optimal sensor placement for leak location in water distribution networks using genetic algorithms," *Sensors*, vol. 13, no. 11, pp. 14 984–15 005, 2013.

[10] A. Soldevila, S. Tornil-Sin, R. M. Fernandez-Canti, J. Blesa, and V. Puig, "Optimal sensor placement for classifier-based leak localization in drinking water networks," in *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*. IEEE, 2016, pp. 325–330.

[11] M. V. Casillas, L. E. Garza-Castañón, and V. Puig, "Optimal sensor placement for leak location in water distribution networks using evolutionary algorithms," *Water*, vol. 7, no. 11, pp. 6496–6515, 2015.

[12] E. Raei, M. E. Shafiee, M. R. Nikoo, and E. Berglund, "Placing an ensemble of pressure sensors for leak detection in water distribution networks under measurement uncertainty," *Journal of Hydroinformatics*, vol. 21, no. 2, pp. 223–239, 2019.

[13] J. Frances-Chust, B. M. Brentan, S. Carpitella, J. Izquierdo, and I. Montalvo, "Optimal placement of pressure sensors using fuzzy dematel-based sensor influence," *Water*, vol. 12, no. 2, p. 493, 2020.

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

[15] C. Quintiliani, I. Vertommen, K. v. Laarhoven, J. v. d. Vliet, and P. v. Thienen, "Optimal pressure sensor locations for leak detection in a dutch water distribution network," in *Environmental Sciences Proceedings*, vol. 2, no. 1. Multidisciplinary Digital Publishing Institute, 2020, p. 40.

[16] L. Ferrandez-Gamot, P. Busson, J. Blesa, S. Tornil-Sin, V. Puig, E. Duviella, and A. Soldevila, "Leak localization in water distribution networks using pressure residuals and classifiers," *IFAC-PapersOnLine*, vol. 48, no. 21, pp. 220–225, 2015.

[17] L. A. Rossman, "EPANET 2: Users manual," US Environmental Protection Agency, Tech. Rep. EPA/600/R-00/057, September 2000.

[18] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *International Journal of Mathematical models and Methods in Applied Sciences*, vol. 1, no. 4, pp. 300–307, 2007.

[19] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[20] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[21] O. Fujiwara and D. B. Khang, "A two-phase decomposition method for optimal design of looped water distribution networks," *Water resources research*, vol. 26, no. 4, pp. 539–549, 1990.