AN ADAPTIVE MULTI-POPULATION EVOLUTIONARY ALGORITHM FOR CONTAMINATION SOURCE IDENTIFICATION IN WATER DISTRIBUTION SYSTEMS

Changhe Li[1], Rui Yang[2], Li Zhou[3], Sanyou Zeng[4], Michalis Mavrovouniotis[5], Ming Yang[6], Shengxiang Yang[7], and Min Wu[8]

[1]Professor, School of Automation, China University of Geosciences, Wuhan 430074, China; and Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, Wuhan 430074, China(corresponding author). Email: changhe.lw@gmail.com

[2]School of Automation, China University of Geosciences, Wuhan 430074, China. Email: 2280920465@qq.com

[3]School of Automation, China University of Geosciences, Wuhan 430074, China. Email: 441837060@qq.com

[4]Professor, School of Mechanical Engineering and Electronic Information, China University of Geosciences, Wuhan 430074, China. Email: sanyouzeng@gmail.com

[5]KIOS Research and Innovation Center of Excellence, Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus. Email: mavrovouniotis.michalis@ucy.ac.cy

[6]Associate Professor, School of Computer Science, China University of Geosciences, Wuhan 430074, China. Email: mingyang@cug.edu.cn

[7]Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U. K. Email: syang@dmu.ac.uk

[8]Professor, School of Automation, China University of Geosciences, Wuhan 430074, China; and Hubei Key Laboratory of Advanced Control and Intelligent Automation for Complex Systems, Wuhan 430074, China. Email: wumin@cug.edu.cn

ABSTRACT

Real-time monitoring of drinking water in a water distribution system (WDS) can effectively warn and reduce safety risks. One of the challenges is to identify the contamination source through these observed data due to the real-time, non-uniqueness, and large scale characteristics. To address the real-time and non-uniqueness challenges, we propose an adaptive multi-population evolutionary optimization algorithm to determine the real-time characteristics of contamination sources, where each population aims to locate and track a different global optimum. The algorithm adaptively adjusts the number of populations using a feed-back learning mechanism. To effectively locate an optimal solution for a population, a co-evolutionary strategy is used to identify the location and the injection profile separately. Experimental results on three WDS networks show that the proposed algorithm is competitive in comparison with three other state-of-the-art evolutionary algorithms.

Keywords: Multi-population adaptation, dynamic bilevel optimization, evolutionary computation, contamination source identification

INTRODUCTION

Water distribution systems (WDSs) are highly susceptible to various threat attempts, including uncertain natural disasters, deliberate destruction, and system failures. For example, a contamination source injected into a WDS will spread through the system rapidly and expose the people to health risks. Detection of the contamination in a WDS using sensors could yield useful observations to identify and locate such contamination threat events. Based on these observations, we can deduce the location, initiation time, and historical injection rate by solving an inverse problem with an optimization algorithm given the observation data under a water distribution simulation model. Because of the rapid diffusion of contaminants in a WDS, we should identify the source characterizations quickly and accurately.

The problem is challenging due to the real-time, non-uniqueness/multi-modal, large-scale, and expensive characteristics. The real-time property requires the search to be data-driven, i.e., the search starts immediately after the contamination is detected at any sensor,

and continuously adapts to changes when new observation data come. The non-uniqueness means that more than one solutions conform to the observation data, which requires the algorithm to have the capability of searching more than one solution simultaneously. The large scale property means that the search space will increase exponentially due to the increase of the number of dimensions of the vector of the injection rate as observation data increase. The problem will become expensive to simulate when the scale of the network increases. There are mainly three kinds of methods for the identification of contamination source in WDSs, including the particle inversion method (Zierolf et al. 1998; Laird et al. 2005; Shang et al. 2002; Costa et al. 2013), the machine learning method (Perelman and Ostfeld 2013; Taormina and Galelli 2018; Huang and Mcbean 2009; Yang et al. 2011), and the simulation-optimization method (Guan et al. 2006; Liu et al. 2008; Seth et al. 2016; Hu et al. 2015). The first two methods can only deduce the location of the contamination source, while the third method can deduce all the information of a contamination event. Therefore, the simulation-optimization method is adopted in this paper.

To address the real-time and non-uniqueness challenges, we propose a new adaptive multi-population evolutionary algorithm where the water distribution network model is coupled directly with a dynamic bilevel optimization model to evaluate solutions as new observation data come. To address the non-uniqueness difficulty, we incorporate a multi-population method where each population aims to locate a different solution, and the number of populations is adaptively adjusted to adapt to the increase of observation data by a feed-back learning mechanism and hence to identify alternative solutions as many as possible. The population diversity across all network nodes will be adaptively increased according to the evolving state of populations measured by a node covering ratio. Thus, at any stage of the observation event, possible solutions that best conform to the observations are identified. In order to speed up the search for a possible solution, in each population, a cooperative co-evolutionary strategy is proposed to locate the location and historical injection rate separately. Experiments in this paper are based on an EPANET 2.0 model (Rossman 2000) of

the water distribution network. Experimental results on three different networks show that the proposed algorithm outperforms several other peer algorithms.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work. Section 3 presents a simulation-optimization model for the problem and introduces our proposed method. Section 4 presents experimental results on three networks. Finally, conclusions and future work are given in Section 5.

RELATED WORK

In this section, we focus on the review of simulation-optimization methods, which can be classified into three categories according to the optimization method used. These algorithms include the gradient descent (GD) method, the particle swarm optimization (PSO) , and the genetic algorithm (GA).

The basic idea of a GD method is to use the gradient direction of the current position as the search direction. Guan et al. (2006) adopted the gradient descent method as the search operator to solve the problem of locating pipe network contamination sources based on the simulation-optimization model. Xin et al. (2013) got the locally optimal solution according to the direction of gradient descent search. In this method, when approaching the optimal value, the convergence speed will gradually slow down, that is, the closer to the objective value, the step length will be smaller, and even causes a zigzag drop.

The genetic algorithm is based on the Darwinian evolution of "survival of the fittest". In the literature, several papers (Preis and Ostfeld 2008; Preis and Ostfeld 2006; Sreepathi et al. 2007) used GAs to solve the problem and achieved promising results. Cristo et al. (2008), firstly established the potential node-set representing the solution of pollution sources – water contamination matrix, then used a GA to search the optimal solution. Yan et al. (2016) used a hybrid encoding method to code the contaminant source identification problem according to the properties of a variable, and combined the crossover and mutation operations. Sankary et al. (2018) proposed a framework to obtain monitoring data by placing mobile sensors using an adaptive GA.

The PSO, which is a swarm intelligence method, originated from the study of the flocking behavior of birds simulating their behavior of flying and foraging in groups. Guneshwor et al. (2018) proposed a simulation-optimization model by using a radial point collocation method and the PSO to identify the unknown groundwater contaminant sources. In this method, due to the lack of dynamic adjustment of particle velocity, it is easy to fall into local optima. Besides the above search algorithms, an evolution strategy algorithm based on a Gaussian mutation operator (GD-ES) was proposed to generate new individuals (Zechman and Ranjithan 2009), and an elite graduation selection strategy was introduced to determine the offspring.

From 2008 to 2011, Liu et al. (2008, 2010, 2011) proposed a multi-population method with GA, which uses an adaptive dynamic optimization technology (ADOT) to make a real-time response to injection events to locate contamination sources. To overcome the premature convergence issue, the algorithm (Liu and Ranjithan 2010) starts with a large number of randomly generated populations and removes populations that are close to each other during the optimization process. A diversity-driven mechanism is used to increase the population diversity for the survival selection in the later evolution stage. The algorithm achieves a great performance in comparison with other optimization algorithms, but it still lacks the mechanism to search the spaces that are not covered by any population at the global level. The number of populations only decreases as time goes on. This issue will be addressed in this paper.

In addition to the literature on contamination source identification described above, there are some important studies on the use of evolutionary algorithms to determine the optimal locations of sensors and the developing an early warning system. Ostfeld and Salomons (2004) presented a method for finding the optimal layout of an early warning detection system. In the next year, Ostfeld and Salomons (2005)extended their previous work by a introducing uncertainties to the demands and injected contamination events. Berry at al. (2006) introduced a mixed-integer programming method for sensor placement. In the

same year, Propato (2006)formulated a mixed-integer linear programming model to identify optimal sensor locations for early warning against accidental and intentional contaminations. Oliker and Ostfeld (2014) improved the event-detection ability by including the support vector machine for the detection of outliers and a multivariate analysis for examining the relationship between water-quality parameters and their mutual patterns.

METHODOLOGY

An inverse problem can be constructed to identify the contamination source characteristics, where the input is a set of concentration observations at sensors and the objective is to minimize the error between predicted concentration and actual observation at sensors on the network using a water distribution simulation model.

Simulation-optimization Model

The simulation-optimization model has two sub-models: a simulation model and an optimization model. The simulation model mainly describes the water movement in a WDS, including the water flow model and the solute transport model. The optimization model can transform the problem into an optimization problem of describing the location of the contamination source, the injection start time, and the injection history information.

The water flow model and water quality model in urban common water supply network can be realized by the hydraulic simulation function of water quality in EPANET 2.0. The model can simulate the diffusion of solutes in contamination events and feedback of node concentration data. We define

$$y_j(t) = \mu(\mathbf{x}(t)), j = 1, \ldots, K \tag{1}$$

where $y_j(t)$ denotes the concentration data of contaminants detected by sensor j at time step t (each sensor is set at a different node); $\mathbf{x}(t) = (x_u, \mathbf{x}_l(t))$ denotes information of a contamination event at a single source at time step t, $x_u \in \mathbb{N}$ denotes the location of the contamination source; $\mathbf{x}_l(t) = (x_0, x_{x_0}, x_{x_0+1}, \ldots, x_t) \in \mathbb{R}^{t-x_0+2}$ denotes the injection profile,

where x_0 is the starting time and $(x_{x_0}, x_{x_0+1}, \ldots, x_t)$ is a time series of injection rate from x_0 to t; μ is a simulation model of water distribution systems with the input $\mathbf{x}(t)$ and the observed output $\boldsymbol{y}(t)$ of K sensors.

Given the water quality hydraulic simulation model, the problem of locating and tracing contamination sources can be converted into a dynamic bilevel optimization problem. The upper-level optimization task is to find the location (x_u) of a contamination event, and the lower-level task is to find the injection history profile $(\mathbf{x}_l(t))$, which is defined as a dynamic optimization problem whose variables will increase as time goes on. Note that, the problem is defined as a single-level optimization problem in previous work discussed in Section 2. In our experiments, we found that the location variable has a significant influence on the distribution of the objective value and the injection profile variables do not. Therefore, we divide the problem into two levels and solve them cooperatively.

In this paper, the square root of the mean squared error between the observed data of a possible solution and the actual observed data of contamination sources is used as the objective, which is shown below:

$$\begin{aligned} \min_{x_u, \mathbf{x}_l(t)} \quad & f_u(x_u, \mathbf{x}_l(t)) = \sqrt{\frac{\sum_{j=1}^{K} \sum_{s=0}^{t} [y_j(s) - \check{y}_j(s)]^2}{K \cdot t}} \\ s.t. \quad & \mathbf{x}_l(t) \in \arg\min_{\mathbf{x}_l(t)} \{f_l(x_u, \mathbf{x}_l(t))\} \end{aligned} \tag{2}$$

where $\check{y}_j(s)$ denotes the observed data at sensor j at time step s of the actual contamination source. Therefore, the objective is to minimize the cumulated error over all sensors so far since the initial observed data are obtained, i.e., to find a solution that complies with the actual observation data. The two levels of problems all aim to minimize the prediction error but have different tasks. In this paper, we solve the problem in an online manner under a real-world scenario.

As mentioned above, the problem is difficult due to the real-time, non-uniqueness, large scale, and expensive properties. For a typical WDS, the number of sensors is far less than the

number of nodes in the network due to the cost reason, which leads to the incompleteness of the observation data and causes many global optima of Eq. (2) at a time during the optimization process. It is a typical multi-modal optimization problem, and hence, multi-modal optimization techniques are needed to find as many global optima as possible for decision-makers to decide the actual one.

For the real-time property, the contamination source identification needs to start once contaminants are detected immediately. With the increase of the observation data, the input of the objective function will continuously change, which may cause the optimal solution set to change. Therefore, it is necessary to identify the optimal solution location in real-time when solving the problem, that is, to solve the problem with dynamic optimization method, to ensure that the source information can be obtained in time when the contamination event occurs.

The large-scale property couples with the expensive property. There are mainly three aspects to be considered. Firstly, when the scale of a WDS is quite large, the simulation time of evaluating a solution will become unaffordable (i.e., it becomes expensive), and the value range of the location of x_u will be increased sharply, resulting in a sharp increase in solution space. Secondly, the dimension for historical contamination injection rate of $\mathbf{x}_l$ will increase as the contamination event keeps on. As the dimension of $\mathbf{x}_l$ increases linearly, the solution space increases exponentially, which makes it challenging for an algorithm to find the optimal solution. Thirdly, when multiple contamination sources exist, the solution representation becomes multiple time series, which makes the problem much more difficult to solve.

In this paper, we mainly focus on addressing the former two considerations in a network with a single contamination source. To address them, we propose an adaptive multiple population framework to find multiple global solutions and adaptively adjust the number of populations to adapt to the changes in observed data.

Adaptive Multi-population Algorithm

Multi-population (MP) methods are very efficient for tracking and locating multiple optima in dynamic environments. This section presents the details of the adaptive multi-population (AMP) framework proposed in this paper.

Algorithm Framework

The framework has three basic components: clustering, parallel search, and diversity increasing components. Figure 1 shows the framework, where m is the size of a population (fixed in the paper), $k(T)$ is the total number of populations at time T (a counter that keeps the number of times the diversity increasing component is triggered after clustering), $\tilde{k}(T)$ is the number of populations survived after the parallel searching, and $\Delta k(T)$ is the number of population to be added.

Algorithm 1 presents the pseudo-code of the framework of the adaptive multi-population algorithm. The framework starts with a set of randomly initialized individuals at T=0, then clusters these new individuals to a set of populations. Populations simultaneously search for global optima and merge if their best individuals locate at the same node on the network. When the coverage ratio over all nodes does not change over two successive generations, it will trigger the diversity increasing procedure where a feedback learning method is used to control the number of populations to be added.

Generation of Multiple Populations

The idea of *divide-and-conquer* is adapted to make each population search for different regions, which is equivalent to reducing the search area of each population. For this purpose, the k-means clustering method is adopted in this paper to generate multiple populations whose search areas are not overlapping with each other.

We firstly generate a certain number of individuals at node i with a probability p_i, which is the same ($p_i = 1/n$, n is the number of nodes of a network) for all nodes at the beginning

Algorithm 1: Pseudo-code of the adaptive multi-population framework

```
Set T ← 0;
Randomly initialize k(T) * m solutions;
while new observation data are detected do
    Clustering newly generated solutions;
    while coverage ratio changes do
        Cooperatively co-evolve each population by GL (Xia and Li 2016) and
         SaDE (Qin et al. 2009) for one iteration;
        if the best solutions of more than one population cover the same node then
            Merge these populations by the competition mechanism;
        Update the coverage ratio on the whole network;
    Update the node selection probability by Eq. (3);
    Estimate the number of populations (Δk(T)) to be increased by Eq. (5);
    Generate Δk(T) · m solutions on the network according to the probability
     obtained by Eq. (3);
    T ← T + 1;
```

of a run and is updated before the increase of populations by:

$$p_i(T) = 1 - \sum_{s=0}^{T} \kappa_i(s) / \max_{i \in n} \sum_{s=0}^{T} \kappa_i(s) \tag{3}$$

where $\kappa_i(s)$ is the accumulated times of node i visited by individuals since the start of the run. From Eq. (3), we can see that the more times a node is visited, the smaller the probability of being selected when generating new solutions. For example, if a node has never been visited by any individuals, the probability of being chosen will be one when generating new solutions.

After that, the nodes covered by all new individuals are then clustered by the k-means clustering method. Algorithm 2 shows the procedures of the k-means clustering method, where the estimation of the number of clusters $\Delta k(T)$ to be clustered will be given later in Section 3. Eventually, individuals of a cluster consist of a new population. Note that the shortest distance of two nodes across the network can be used instead of the Euclidean distance used in this paper.

Algorithm 2: Pseudo-code of the k-means clustering

```
Initialize Δk cluster centers;
while clusters do not converge, i.e., cluster centers are still changing do
    for each node do
        Calculate the Euclidean distance from the node to each cluster center;
        The node is assigned to the cluster whose center is nearest to the node;
    Recalculate the center of each cluster;
```

Cooperative Co-evolutionary Search

The representation of the solution to the problem is composed of discrete and continuous parts. The location variable is an integer, and the injection profile is represented by a vector of real values. To alleviate the difficulty in searching the solution of the hybrid representation, we use the cooperative co-evolution (CC) strategy (Potter and De Jong 1994) to solve the upper-level and lower-level problems separately.

Cooperative Co-evolution

Cooperative co-evolution (Potter and De Jong 1994; Yang et al. 2017) is an extension of the traditional evolutionary algorithm inspired by the strategy of *divide-and-conquer* when dealing with large scale optimization problems. By decomposing decision variables into a set of independent groups (each group of variables consists of a subproblem), the original problem can be solved by solving these subproblems by different algorithms. We use two different single-population based algorithms to solve the two levels problems using the CC strategy, i.e., when evaluating a solution in one population, the value of the remaining information is taken from the best solution of the other population.

Genetic Learning

Genetic learning (GL) (Xia and Li 2016) is an optimization method based on probability and statistics. It consists of two components: gene prediction and gene exploration. The gene prediction uses a probability model build on historical data to select genes, and the gene exploration attempts to discover new genes to increase the population diversity. These two components interact to form a feedback system. The genetic learning method makes

Algorithm 3: Pseudo-code of the genetics learning algorithm

for *each individual* **do**
  $idx \leftarrow$ node index of the current individual;
  $obj \leftarrow$ prediction error of the current individual;
  $sum\_obj_{idx} \leftarrow sum\_obj_{idx} + obj$;
  $count_{idx} \leftarrow count_{idx} + 1$;

full use of the historical information generated in the iteration process. It analyses the fitness of each individual in every possible value of each decision variable and calculates the probability of each value being selected. It then performs the mutation operation in each dimension according to these probabilities.

We use this method to find the injection location of the upper-level problem. For each population, we count the cumulative prediction objective value sum_obj of each node in the network. Algorithm 3 shows the specific steps. The probability of node idx to be selected can be obtained by:

$$p_{idx} = \frac{mean_{max} - mean_{idx}}{\sum_{i=1}^{n} mean_{max} - mean_{idx}} \tag{4}$$

where $mean_{idx} = sum\_obj_{idx}/count_{idx}$, $mean_{max} = \max(mean_i \mid i = 1, 2, \cdots, n)$. The probability will be updated every iteration. The nodes with smaller errors have larger probabilities to be selected for GL.

Strategy Adaptation Differential Evolution

Differential evolution (DE) (Storn and Price 1997) is a simple yet effective optimization algorithm, especially for solving continuous optimization problems. The strategy adaptive differential evolution algorithm (SaDE) (Qin et al. 2009) is an enhanced version, where the differential operators are selected adaptively according to their performance in previous generations. Here four classical difference operators *DE/rand/1*, *DE/best/1*, *DE/target-to-best/1* and *DE/best/2* are used to optimize the initial time and historical injection rate. Note that the recommended values for parameters of SaDE (Qin et al. 2009) were used in this paper.

Population Management

Population removal and population increase are two critical components of the adaptive multi-population framework, especially in dynamic environments (Yang and Li 2010; Li and Yang 2012; Li et al. 2015). The population removal component aims to remove redundant populations, and hence to save computational resources. The population increase component aims to increase the diversity in the areas which have not been searched or not sufficiently searched so far, and hence to find more global optima solutions.

Population Removal

After the clustering, each population will cover a unique search area containing one or serval geographically closed nodes on the network. As the search goes on, some of the populations may move toward the same area, and finally, converge at the same node of that area. This causes redundant search, which should be prevented. To prevent more than one population from searching in the same area, we check their best solutions. When their best solutions locate at the same node on the network, these populations are deemed to be overcrowded.

To address the overcrowding issue, we introduced a competition mechanism. When two or more populations overcrowd in one area, they will compete with each other. We first merge all overcrowding populations in that area, then rank all individuals of the combined population according to the objective value, finally keep the best m individuals and remove the remaining. The overcrowding detection is performed every iteration, and the competition mechanism will be triggered whenever overcrowding populations are detected.

Population Increase

Increasing populations means increasing the diversity for exploring unexplored areas. However, to increase populations, we need to know when, how many, and where to generate new random solutions.

The moment to increase populations has a significant impact on the performance of an algorithm in dynamic environments (Li et al. 2015). Frequently increasing the diversity

causes the inefficiency issue in the exploitation of global optima, while infrequently increasing the diversity causes the algorithm to fail to track the changes of the optima. In order to solve this problem, we increase populations when all populations enter into a stable status on the network indicated by the coverage ratio over all nodes, i.e., we increase populations when the coverage ratio does not change over two successive iterations.

After the identification of the moment to increase populations, the next step is to determine how many populations to be increased. Inspired by (Li et al. 2016), we propose a feedback learning strategy based on historical data to estimate the number of populations to be increased. After the detection of the moment to increase populations, we record the number of populations ($\tilde{k}(T)$) survived from the competition, then compare the reduced number with the previous increased number ($\Delta k(T-1)$), which is set to $k(T=0)/2$ initially. If the reduced number is less than the previous increased number, which is a positive feedback and means that new areas have been discovered, then the current increased number will be increased by one based on the previous increased number ($\Delta k(T-1)$); if the reduced number is greater than the previous increased number, which is a negative feedback and the current increased number will be decreased by one based on the previous increased number; otherwise, there is no change:

$$\Delta k(T) = \begin{cases} \Delta k(T-1)+1 & \text{if } \Delta k(T-1) < k(T) - \tilde{k}(T) \\ max(\Delta k(T-1)-1, 1) & \text{if } \Delta k(T-1) > k(T) - \tilde{k}(T) \\ \Delta k(T-1) & \text{if } \Delta k(T-1) = k(T) - \tilde{k}(T) \end{cases} \tag{5}$$

Note that, there is at least one population to be increased in the negative feedback case to make sure that finding new global optima is always possible.

Finally, we can generate random solutions to increase the population diversity. Different from the traditional random immigrant scheme where new solutions are uniformly randomly generated at all nodes without considering the distribution of the current and historical solutions on the network, we generate new solutions at node i with probability p_i obtained

by Eq. (3). This way, infrequently visited nodes have large probabilities of being explored, which is very helpful for finding new global optima.

Note that, in order to respond to changes in dynamic environments, change detection is usually needed, or an algorithm is directly informed at the moment when a change occurs. For example, the algorithm (Liu et al. 2011) is informed once new data come, and mutation step lengths are reinitialized. In this paper, we do not need to detect changes. Here, to respond to changes, the population increase is determined only based on the current evolutionary status of the whole populations but not necessarily at the moment of a change occurring, i.e., the distribution of all solutions on the network does not changes means that the algorithm becomes converging and needs to be diversified to enhance the exploration capability.

RESULTS AND DISCUSSION

In this section, we conduct two groups of experiments to study the performance of the proposed algorithm. The first group of experiments aims to analyze the effect of the critical parameters on the performance of our algorithm, including the initial number of populations k and the size of a single population m. The second group of experiments aims to compare the performance of our algorithm with several state-of-the-art evolutionary algorithms.

Experimental Setup

Our algorithm treats the water distribution network as a "black box", the structure of the networks has little impact on the performance of the algorithm. When the scale of a WDS is quite large, the simulation time of evaluating a solution will become unaffordable, and the value range of the location will be increased sharply, resulting in a sharp increase in solution space. As a result, the algorithm performance may deteriorate. Then we take scales of the networks as main factor influencing the performance of algorithms.

Three networks were chosen with the scale from 97 nodes to 430 nodes, as shown in Figure 2. The configuration of each network is shown in Table 1, where the sensor nodes were set according to the literature in the way that they can cover as many nodes as possible

during the injection events. For each network, we set three different test cases, which are listed in Table 2, where the injection rate changes every 10 minutes.

In the experiments, the simulation duration was set to 24 hours, and observation data were collected from sensors every 10 minutes. Given the above configurations in Table 2, we assume that the range of $[0h, 4h]$ for the start time and $[5g, 30g]$ for the injection mass. Therefore, new solutions are randomly initialized within these ranges. To reduce the complexity of the problem, we assume the injection duration is known for all algorithms in this paper. All the results in this paper are averaged over 20 independent runs, and each test case was run for 200,000 objective evaluations.

Before the experiments, we conduct the hydraulic simulation for three water distribution network, we analyzed the water age of all nodes, and concluded that if the pollutants were injected at the water source, it can reach at each node during the simulation. We set sensors according to the locations provided by literature and our experience to make the coverage of the sensor as large as possible. Therefore, the occurrence of injection events can be detected within the simulation time even if some sensors are in low-flow zones.

We use a success rate and the prediction error to evaluate the performance of an algorithm. A successful run is a run where the true injection node is found, so the success rate is the number of successful runs over the total number of runs. Note that, the prediction error is averaged over all successful runs, not over the total number of runs in this paper.

Parameter Sensitivity Analysis

In this subsection, network 1 with configuration 1-3 in Table 2 was used to test the performance change of our algorithm with different combinations of $k \in \{10, 20, 40\}$ and $m \in \{20, 50, 100\}$, where each combination is listed in Table 3.

As shown in Table 4, the success rate is one regardless of the change in the size of a single population and the initial number of populations. The prediction error varies a little with different combinations. Figure 3 shows the change in the coverage rate and the number of populations on instance 1-3 of network 1 with a fixed single population size of m=50,

where the initial number of populations is set to k=10/20/40 from left to right. From the figure, it can be seen that the number of the diversity increase during the run is the same (i.e, five times) and the number of populations finally survived is maintained at all about nine. It indicates that the initial number of populations has no significant impact on the performance of the proposed algorithm. Due to the adaptation mechanism, the number of populations will be adjusted to an appropriate number for a specific problem. For the following experiments, the default initial number of populations is set to 20.

Figure 4 shows the change in the coverage rate and the number of populations on instance 1-3 of network 1 with a fixed initial number of populations of 20, where the size of a single population is set to m=20/50/100 from left to right. For the instance 1-3-4 with m=20, compared with the instance 1-3-5 with m=50, the size of the overall populations is not enough to explore the whole search space. Therefore, the algorithm will frequently increase new populations to make up for the lack of the overall population size to improve its exploration capability. When m=100, the population size is enough for the problem, and the frequency of adding new population becomes low, which saves unnecessary computational overhead. Similarly, it can also be seen that the different sizes of a single population have no significant impact on the performance of the algorithm due to the diversity trigger mechanism. For a small size, more new populations will be added to achieve the best exploration capability and vice versa.

Comparison with Other Algorithms

In this subsection, three state-of-the-art simulation-optimization-based algorithms are chosen to compare with our algorithm, named AMP-CC(GL-SaDE) for short. The peer algorithms are ADOT-CC(GL-SaDE) (Liu et al. 2011), GD-ES (Zechman and Ranjithan 2009), and LRM-ADOT-CC(GL-SaDE) (Liu et al. 2012). Both ADOT-CC(GL-SaDE) and LRM-ADOT-CC(GL-SaDE) use an adaptive dynamic optimization technique based on multi-populations. The differences is that LRM-ADOT-CC(GL-SaDE) uses a logistic regression to predict the possible location of an injection event. GD-ES (Zechman and Ranjithan 2009)

uses evolution strategies (ESs) implemented with a tree-based encoding to represent variable-length decision variables. Note that, in order to compare the performance of the adaptive framework proposed in this paper with the one used in ADOT-CC(GL-SaDE) (Liu et al. 2011) and LRM-ADOT-CC(GL-SaDE) (Liu et al. 2012), we use the same search method for each population (the cooperative co-evolutionary algorithm introduced in Section 3) for these three algorithms. The suggested parameter values were used for all the other three algorithms. Besides, the number of objective evaluations and consumed time for optimization are equivalent among our algorithm and three state-of-the-art algorithms.

Comparison of the Success Rate and Prediction Error

Table 5 shows the comparison of the success rate obtained by the four algorithms on each test instance where the best result of each instance is shown in bold font. It can be seen that AMP-CC(GL-SaDE) outperforms all the other three algorithms on five out of nine instances. AMP-CC(GL-SaDE) achieves a success rate of one on five out of nine instances and a minimum success rate of 0.9 on the other instances. For simple test instances with a short injection history, all the four algorithms can find the true location for all runs except GD-ES on instances of networks 2 and 3. However, when the injection profile increases, the performance of all the algorithms becomes worse. Compared with ADOT-CC(GL-SaDE), LRM-ADOT-CC(GL-SaDE) does improve the success rate due to the location prediction mechanism.

Table 6 presents the comparison of the prediction error obtained by all the four algorithms on all the test instances where the best result of each instance is shown in bold font. In each test case, the Wilcoxon rank sum test at a significant level of 0.05 is performed on the prediction error between our algorithm and other peer algorithms. The errors with suffixes $+$, $-$, or $\approx$ indicate that the prediction errors obtained by other peer algorithms are significantly better than, significantly worse than, and statistically equivalent to our algorithm, respectively.

From Table 6, it can be seen that AMP-CC(GL-SaDE) outperforms all the other three

algorithms on all test instances. The performance of AMP-CC(GL-SaDE) is significantly better than the other three algorithms on instances 1-1, 2-1, 2-2, and 3-3. The error achieved by GD-ES is the worst among all the algorithms on small scale instances (networks 1 and 2). LRM-ADOT-CC(GL-SaDE) introduces a pre-screening mechanism based on a logical regression model, which improves the performance on the small instances of networks 1 and 2, but not on the large instances of network 3. Due to the adaptation of the number of populations, AMP-CC(GL-SaDE) achieves much smaller errors than ADOT-based algorithms.

To test the impact of changing the total number of objective evaluations on the performance of the four algorithms, we run all the algorithms on instance 1-3 with different maximum numbers of objective evaluations. Figure 5 presents the comparison of the number of populations obtained when the run finishes and the average best prediction error of each algorithm under different maximum numbers of evaluations on the test instance 1-3. From the comparison, we can have (1) AMP-CC(GL-SaDE) achieves the largest number of populations and it is always superior to other algorithms in terms of the prediction error and (2) the advantage of multi-population methods over single population methods can also be seen in the comparison, where all the errors obtained by the three multi-population methods decrease when the maximum number of objective evaluations increases.

Comparison of the Diversity Maintaining

Figure 6 presents the comparison of the change in the coverage ratio and the number of populations for the four algorithms on instances 1-3 (sub-figures on the top) and 3-1 (sub-figures on the bottom). To some extent, the change in the coverage ratio during the runtime reflects the exploring capability of an algorithm, i.e., the ability to maintain the population diversity. Among the four algorithms, only GD-ES does not consider the population diversity. In Figure 6, the coverage ratio of GD-ES sharply drops to a low level even at the beginning of the run and maintains at that low level till the end of the run on both test instances. This may results in a premature convergence issue since the problem is highly multi-modal, which can explain the reason for the lower success rate of GD-ES than the other three algorithms.

The improvement of diversity is the key to improve the success rate of the algorithm, so how to increase diversity will directly affect the ability to find optimal solutions. ADOT-CC(GL-SaDE), LRM-ADOT-CC(GL-SaDE), and AMP-CC(GL-SaDE) all adopt the same search methods but use different multi-population frameworks. AMP-CC(GL-SaDE) uses the AMP framework proposed in this paper, and the other two use the ADOT framework (Liu et al. 2011). The essential difference between the two frameworks is the mechanism of increasing diversity. The ADOT framework uses a distance evaluation strategy to disperse individuals in the population to other regions, but it does not increase the coverage ratio as necessary. It is because ADOT does not increase individuals, but move them to other nodes already covered by existing solutions. This can be validated from the coverage ratio of ADOT-CC(GL-SaDE) and LRM-ADOT-CC(GL-SaDE) in Figure 6. Although this strategy increases the exploring ability to a certain extent, it does not expand the scope of exploration in essence, and its ability to search for new optima is not strong enough, which leads to the lower success rate in comparison with the AMP framework.

From the two curves of AMP-CC(GL-SaDE) on instance 1-3 in Figure 6 (the top right), it can be seen that when the coverage ratio remains for several iterations, the number of populations increases and the coverage ratio rises sharply to nearly one. At this moment, AMP-CC(GL-SaDE) expands the exploring area to the whole network. The AMP framework increases the population diversity at the global level, while the ADOT framework increases the population diversity at the local level within each population. Therefore, the exploring capability of the AMP framework is much stronger than that of the ADOT framework.

Moreover, in the AMP framework, new solutions are initialized in the areas, where no solution is covering, with a very large probability. During the runtime, the number of times for each node visited by solutions is recorded. The more times the node is visited, the more computing resources are spent at that node to optimize the injection history. However, when new solutions are added, more computing resources are placed at the nodes with a few visitors to find more global optima.

Different from the behavior of AMP-CC(GL-SaDE) on instance 1-3, the algorithm does not increase populations during the whole run on instance 3-1, i.e., the diversity increasing component is not triggered (similar to the ADOT framework). The scale of the network 3 is much larger than that of network 1. Although the lowest level of the coverage ratio is similar in the two cases, the number of nodes covered on instance 3-1 is much larger than that on instance 1-3. This means that for the same given number of individuals, the population diversity on instance 3-1 is also much larger than that on instance 1-3. Therefore, given a limited evaluation budget, the algorithm spends more computing resources on exploiting the current area rather than exploring new areas and tries to improve its performance on the optimization of the historical injection profile. The comparison in the two scenarios shows that the AMP framework can adapt to problems with different scales.

Comparison of the Injection Profile

Figure 7 shows the time-varying mean of contaminants concentrations observed at four detection points on instance 1-3, where the solid red curve refers to the actual observation data, the solid dark curve refers to the contaminants concentrations simulated by the best solution found by the algorithms, and the gray dashed curves are for the remaining solutions. It can be seen that the general trend of the dark solid curves are very close to the actual data, i.e., the optimal solutions found by the algorithms are sound. Compared with the other three algorithms, AMP-CC(GL-SaDE) finds more optimal solutions. The comparison further verifies that AMP-CC(GL-SaDE) has stronger exploring capability than the other three algorithms.

When we observe the change in the prediction error of the best solutions obtained by the four algorithms in Figure 8, all the three multi-population based algorithms can quickly locate the injection event in the beginning of the run except GD-ES. However, it is interesting to observe that the performance of the three algorithms deteriorates when the new observation data keep coming. The prediction errors of both ADOT-CC(GL-SaDE) and LRM-ADOT-CC(GL-SaDE) are even worse than that of GD-ES after the number of evaluations reaches

30,000. Among the four algorithms, AMP-CC(GL-SaDE) achieves the minimal error for most of the running time.

Performance Investigation on Dynamic Contamination Sources

In this subsection, we identify dynamic contamination sources by our algorithm. To make sure that there is enough time for distributing the injected masses, the simulation duration was set to 48 hours, and observation data were collected from sensors every 10 minutes, other parameters of our algorithm were set as above. However, the location of the contamination source changes every six hours. Network 1 with four contamination sources change was used to test the performance by our algorithm. The injection rate of four sources was all set to 30,25,20,15,10,5,5,10,15,20,25,30 for simplicity. Considering that the current water quality of the water distribution network will be affected by the contamination sources in the previous stages, in our algorithm. Each simulation is from the beginning to the current time phase. In each stage, when a sensor detects that the contamination exceeds a certain threshold, the algorithm starts to optimize and search for the optimal location and injection rate of the pollution source in the current stage.

Table 7 shows the errors and success rates over 20 runs. The success rate of finding the real contamination source in each stage is 0.95, 1, 1 and 1. It can be seen that our algorithm also works well for dynamic contamination sources and can accurately find the real sources at each stage. The prediction error is shown in Figure 9. There are three peaks in the figure, because when the pollution source changes, the optimal solution of the existing population is no longer the global optimal solution, and the error will suddenly increase. However, the error quickly drops due to the parallel search ability of the proposed algorithm. Figure 10 shows the time-varying mean of contaminants concentrations observed at four detection points of one single run, where the solid red curve refers to the actual observation data, the solid dark curve refers to the contaminants concentrations simulated by the best solution found by the algorithm in terms of the error, and the gray dashed curves are for the remaining two best solutions. It can be seen that that there are four peaks during the runtime, which means

there are four different sources of pollution, and our algorithm can track the optimal solution well.

CONCLUSIONS

This paper proposes an adaptive multi-population framework to solve the contamination source identification problem in the water distribution system. The problem is defined as a dynamic bilevel optimization problem. To handle the real-time and non-uniqueness characteristics of the problem, we develop an adaptive mechanism to enhance the exploring ability of multi-population methods and a cooperative co-evolution strategy for the bilevel optimization problem. The AMP framework can automatically remove redundant populations and adds a proper number of populations at a proper moment, which makes it adaptable to different problems.

From the experimental results, we can draw the following two conclusions. Firstly, the multi-population based methods have advantages over single-population based algorithms. Secondly, by removing crowding populations in over-exploited areas and adding new populations in unexplored areas, the AMP framework can find more candidate solutions than the ADOT framework. As a result, it significantly improves the success rate in finding the true optimal solution.

We want to pursue the following work in the future. Firstly, more uncertain factors should be considered during the simulation, e.g., network structures and the sensor locations. Secondly, multiple contamination sources should also be considered in a dynamic scenario. Thirdly, the simulation of large scale problems is very time-expensive, which is challenging to simulation-optimization based methods. Finally, the number and the location of sensors on the network is also an interesting topic.

DATA AVAILABILITY

Some or all data, models, or code generated or used during the study are available in a repository online in accordance with funder data retention policies.

The source code for all the involved algorithms in this paper will be released in OFEC, which is an open framework for evolutionary computation, at the link `https://github.com/Changhe160/OFEC`.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grants 61673355, 61673331, and 62076226, in part by the Fundamental Research Funds for the Central Universities China University of Geosciences (Wuhan) under Grants CUG170603 and CUGGC02, in part by the Hubei Provincial Natural Science Foundation of China under Grant 2015CFA010, in part by the 111 project under Grant B17040, in part by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 739551 (KIOS CoE) and from the Government of the Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

REFERENCES

Berry, J., Hart, W., Phillips, C., Uber, J., and Watson, J. (2006). "Sensor placement in municipal water networks with temporal integer programming models." *Journal of Water Resources Planning & Management*, 132(4), p. 218–224.

Costa, D. M., Melo, L. F., and Martins, F. G. (2013). "Localization of contamination sources in drinking water distribution systems: A method based on successive positive readings of sensors." *Water Resources Management*, 27(13), 4623–4635.

Cristo, C. D. and Leopardi, A. (2008). "Pollution source identification of accidental contamination in water distribution networks." *Journal of Water Resources Planning and Management*, 134(2), 197–202.

Guan, J., Aral, M. M., Maslia, M. L., and Grayman, W. M. (2006). "Identification of contaminant sources in water distribution systems using simulation-optimization method: Case study." *Journal of Water Resources Planning and Management*, 132(4), 252–262.

Guneshwor, L., Eldho, T. I., and Kumar, A. V. (2018). "Identification of groundwater con-

tamination sources using meshfree rpcm simulation and particle swarm optimization." *Water Resources Management*, 32(4), 1517–1538.

Hu, C., Zhao, J., Yan, X., Zeng, D., and Guo, S. (2015). "A map reduce based parallel niche genetic algorithm for contaminant source identification in water distribution network." *Ad Hoc Networks*, 35(DEC.), 116–126.

Huang, J. J. and Mcbean, E. A. (2009). "Data mining to identify contaminant event locations in water distribution systems." *Journal of Water Resources Planning and Management*, 135(6), 466–474.

Laird, C. D., Biegler, L. T., Waanders, B. G. V. B., and Bartlett, R. A. (2005). "Contamination source determination for water networks." *Journal of Water Resources Planning and Management*, 131(2), 125–134.

Li, C., Nguyen, T. T., Yang, M., Mavrovouniotis, M., and Yang, S. (2016). "An adaptive multipopulation framework for locating and tracking multiple optima." *IEEE Transactions on Evolutionary Computation*, 20(4), 590–605.

Li, C., Nguyen, T. T., Yang, M., Yang, S., and Zeng, S. (2015). "Multi-population methods in unconstrained continuous dynamic environments: The challenges." *Information Sciences*, 296, 95–118.

Li, C. and Yang, S. (2012). "A general framework of multipopulation methods with clustering in undetectable dynamic environments." *IEEE Transactions on Evolutionary Computation*, 16(4), 556–577.

Liu, L. and Ranjithan, S. R. (2010). "An adaptive optimization technique for dynamic environments." *Engineering Applications of Artificial Intelligence*, 23(5), 772–779.

Liu, L., Ranjithan, S. R., and Mahinthakumar, G. (2011). "Contamination source identification in water distribution systems using an adaptive dynamic optimization procedure." *Journal of Water Resources Planning and Management*, 137(2), 183–192.

Liu, L., Zechman, E. M., Brill, Jr, E. D., Mahinthakumar, G., Ranjithan, S., and Uber, J. (2008). "Adaptive contamination source identification in water distribution systems using

564 an evolutionary algorithm-based dynamic optimization procedure." *Water Distribution*
565 *Systems Analysis Symposium 2006*, 1–9.
566 Liu, L., Zechman, E. M., Mahinthakumar, G., and Ranji Ranjithan, S. (2012). "Identifying
567 contaminant sources for water distribution systems using a hybrid method." *Civil Engi-*
568 *neering and Environmental Systems*, 29(2), 123–136.
569 Oliker, N. and Ostfeld, A. (2014). "A coupled classification - evolutionary optimization model
570 for contamination event detection in water distribution systems." *Water Research*, 51, 234–
571 245.
572 Ostfeld, A. and Salomons, E. (2004). "Optimal layout of early warning detection stations for
573 water distribution systems security." *Journal of Water Resources Planning and Manage-*
574 *ment*, 130(5), 377–385.
575 Ostfeld, A. and Salomons, E. (2005). "Optimal early warning monitoring system layout
576 for water networks security: inclusion of sensors sensitivities and response delays." *Civil*
577 *Engineering and Environmental Systems*, 22(3), 151–169.
578 Perelman, L. and Ostfeld, A. (2013). "Bayesian networks for source intrusion detection."
579 *Journal of Water Resources Planning and Management*, 139(4), 426–432.
580 Potter, M. A. and De Jong, K. A. (1994). "A cooperative coevolutionary approach to function
581 optimization." *Parallel Problem Solving from Nature (PPSN)*, 249–257.
582 Preis, A. and Ostfeld, A. (2006). "Contamination source identification in water systems: A
583 hybrid model trees-linear programming scheme." *Journal of Water Resources Planning*
584 *and Management*, 132(4), 263–273.
585 Preis, A. and Ostfeld, A. (2008). "Genetic algorithm for contaminant source characterization
586 using imperfect sensors." *Civil Engineering and Environmental Systems*, 25(1), 29–39.
587 Propato, M. (2006). "Contamination warning in water networks: General mixed-integer linear
588 models for sensor location design." *Journal of Water Resources Planning & Management*,
589 132(4), p. 225–233.
590 Qin, A. K., Huang, V. L., and Suganthan, P. N. (2009). "Differential evolution algorithm

with strategy adaptation for global numerical optimization." *IEEE Transactions on Evolutionary Computation*, 13(2), 398–417.

Rossman, L. A. (2000). "Epanet 2 users' manual." *Report no.*, U. S. Environmental Protection Agency.

Sankary, N. and Ostfeld, A. (2018). "Multiobjective optimization of inline mobile and fixed wireless sensor networks under conditions of demand uncertainty." *Journal of Water Resources Planning and Management*, 144(8), 04018043.

Seth, A., Klise, K. A., Siirola, J. D., Haxton, T., and Laird, C. D. (2016). "Testing contamination source identification methods for water distribution networks." *Journal of Water Resources Planning & Management*, 142(4), 04016001.1–04016001.11.

Shang, F., Uber, J. G., and Polycarpou, M. M. (2002). "Particle backtracking algorithm for water distribution system analysis." *Journal of Environmental Engineering*, 128(5), 441–450.

Sreepathi, S., Mahinthakumar, K., Zechman, E., Ranjithan, R., Brill, D., Ma, X., and Von Laszewski, G. (2007). "Cyberinfrastructure for contamination source characterization in water distribution systems." *International Conference on Computational Science*, Springer, 1058–1065.

Storn, R. and Price, K. (1997). "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces." *Journal of global optimization*, 11(4), 341–359.

Taormina, R. and Galelli, S. (2018). "Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems." *Journal of Water Resources Planning and Management*, 144(10), 04018065.

Xia, Y. and Li, C. (2016). "Memory-based statistical learning for the travelling salesman problem." *2016 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2935–2941.

Xin, K., Sheng, X., Tao, T., and Xiang, N. (2013). "Contamination source identification in water distribution networks based on negative gradient method." *Journal of Tongji*

University, 41(1), 116–120.

Yan, X., Zhao, J., Hu, C., and Wu, Q. (2016). "Contaminant source identification in water distribution network based on hybrid encoding." *Journal of Computational Methods in Sciences and Engineering*, 16(2), 379–390.

Yang, M., Omidvar, M., Li, C., Li, X., Cai, Z., Kazimipour, B., and Yao, X. (2017). "Efficient resource allocation in cooperative co-evolution for large-scale global optimization." *IEEE Transactions on Evolutionary Computation*, 21(4), 493–505.

Yang, S. and Li, C. (2010). "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments." *IEEE Transactions on Evolutionary Computation*, 14(6), 959–974.

Yang, X., Boccelli, D. L., and De Sanctis, A. E. (2011). "A bayesian approach for probabilistic contamination source identification." *World Environmental and Water Resources Congress 2011: Bearing Knowledge for Sustainability*, American Society of Civil Engineers, 304–313.

Zechman, E. M. and Ranjithan, S. R. (2009). "Evolutionary computation-based methods for characterizing contaminant sources in a water distribution system." *Journal of Water Resources Planning and Management*, 135(5), 334–343.

Zierolf, M. L., Polycarpou, M. M., and Uber, J. G. (1998). "Development and autocalibration of an input-output model of chlorine transport in drinking water distribution systems." *IEEE transactions on control systems technology*, 6(4), 543–553.

List of Tables

1 Network configurations 30

2 Configurations for nine test instances 31

3 Combinations of the initial number of populations (k) and the population size (m) for test instance 1-3 in Table 2 32

4 Success rates and prediction errors of different combinations of k and m on network 1 with configuration 1-3 33

5 Success rate of all the four algorithms in all the test cases 34

6 Prediction error and standard deviation of all the four algorithms in all the test cases 35

7 Results of dynamic sources over 20 runs on network 1 36

TABLE 1. Network configurations

Network	# of nodes	# of sensors	Sensor location
1	97	4	113, 147, 211, 120
2	279	12	J-124, J-202, J-204, J-196, J-122, J-267, J-115, J-197, J-14, J-55, J-3, J-58
3	430	16	J-56, J-321, J-296, J-11, J-258, J-209, J-118, J-345, J-112, J-121, J-69, J-171, J-200, J-6, J-342, J-229

TABLE 2. Configurations for nine test instances

Instance		Location	Start time	Duration (min)	Injection rate
Network 1	1-1	113	0:00	60	5,10,15,20,15,10
	1-2	157	2:00	120	30,25,20,15,10,5,5,10,15,20,25,30
	1-3	267	4:00	240	30,5,30,5, ..., 30,5,30,5
Network 2	2-1	J-196	0:00	60	5,10,15,20,15,10
	2-2	J-124	2:00	120	30,25,20,15,10,5,5,10,15,20,25,30
	2-3	J-146	4:00	240	30,5,30,5, ..., 30,5,30,5
Network 3	3-1	J-37	0:00	60	5,10,15,20,15,10
	3-2	J-242	2:00	120	30,25,20,15,10,5,5,10,15,20,25,30
	3-3	J-56	4:00	240	30,5,30,5, ..., 30,5,30,5

TABLE 3. Combinations of the initial number of populations (k) and the population size (m) for test instance 1-3 in Table 2

Instance	1-3-1	1-3-2	1-3-3	1-3-4	1-3-5	1-3-6	1-3-7	1-3-8	1-3-9
k	10	10	10	20	20	20	40	40	40
m	20	50	100	20	50	100	20	50	100

TABLE 4. Success rates and prediction errors of different combinations of k and m on network 1 with configuration 1-3

Instance	1-3-1	1-3-2	1-3-3	1-3-4	1-3-5	1-3-6	1-3-7	1-3-8	1-3-9
Success rate	1	1	1	1	1	1	1	1	1
Error	1.55E+01	1.33E+01	1.36E+01	1.31E+01	1.30E+01	1.38E+01	1.55E+01	1.40E+01	1.41E+01

TABLE 5. Success rate of all the four algorithms in all the test cases

Algorithm	1-1	1-2	1-3	2-1	2-2	2-3	3-1	3-2	3-3
GD-ES	**1**	0.1	0.1	0.7	0.8	0.6	0.8	0.7	0.8
ADOT-CC(GL-SaDE)	**1**	0.65	0.5	**1**	**1**	0.7	**1**	0.9	0.4
LRM-ADOT-CC(GL-SaDE)	**1**	0.65	0.7	**1**	**1**	0.3	**1**	0.7	**1**
AMP-CC(GL-SaDE)	**1**	**0.95**	**1**	**1**	**1**	**0.9**	**1**	**0.9**	**0.9**

TABLE 6. Prediction error and standard deviation of all the four algorithms in all the test cases

Instance	GD-ES	ADOT-CC(GL-SaDE)	LRM-ADOT-CC(GL-SaDE)	AMP-CC(GL-SaDE)
1-1	7.8E+00±2.4E+00^{-}	5.4E-02±2.3E-03-	2.7E-02±4.5E-03-	**0**
1-2	6.4E+00±0≈	3.6E+00±1.2E+00≈	3.5E+00±1.7E+00≈	**2.6E+00±1.9E+00**
1-3	**1.3E+01±0**≈	1.5E+01±3.2E+00≈	1.5E+01±4.2E+00≈	**1.3E+01±3.0E+00**
2-1	6.6E+00±1.5E+00^{-}	8.7E-03±2.8E-04^{-}	2.3E-02±3.3E-03^{-}	**0**
2-2	9.3E+00±9.5E-01^{-}	3.1E-02±7.5E-03^{-}	6.2E-02±6.1E-03^{-}	**0**
2-3	1.5E+01±1.4E+00≈	1.6E+01±1.7E+00≈	1.5E+01±4.2E+00≈	**1.4E+01±1.3E+00**
3-1	7.7E+00±2.9E+00≈	6.2E+00±2.2E+00≈	8.6E+00±1.9E+00≈	**4.5E+00±2.3E+00**
3-2	8.5E+00±2.3E+00≈	7.5E+00±3.8E+00≈	1.1E+01±4.3E+00≈	**7.2E+00±2.6E+00**
3-3	1.2E+01±1.7E+00^{-}	1.3E-01±1.1E-02^{-}	2.1E-01±1.8E-02^{-}	**5.4E-02±2.1E-03**

TABLE 7. Results of dynamic sources over 20 runs on network 1

Run ID	Location	Error	Standard deviation
1	193 120 205 113	3.48E-02	2.94E+00
2	193 120 205 113	8.67E-02	4.80E+00
3	193 120 205 113	3.75E-02	4.63E+00
4	193 120 205 113	2.51E-02	4.46E+00
5	193 120 205 113	2.70E-02	2.70E+00
6	193 120 205 113	5.00E-02	1.85E+00
7	273 120 205 113	1.50E-01	8.26E+00
8	193 120 205 113	3.29E-02	2.89E+00
9	193 120 205 113	3.74E-02	4.02E+00
10	193 120 205 113	1.97E-02	3.17E+00
11	193 120 205 113	1.15E-02	3.31E+00
12	173 120 205 113	7.90E-02	6.37E+00
13	193 120 205 113	8.35E-02	2.36E+00
14	193 120 205 113	1.70E-02	4.37E+00
15	193 120 205 113	1.72E-02	4.43E+00
16	193 120 205 113	3.52E-02	5.75E+00
17	193 120 205 113	2.42E-02	3.36E+00
18	193 120 205 113	4.78E-02	3.54E+00
19	193 120 205 113	2.01E-02	2.28E+00
20	193 120 205 113	3.07E-02	1.80E+00

List of Figures

1 Framework of the adaptive multi-population method 38

2 The picture of three network with 97/279/430 nodes 39

3 The change in the number of populations and the coverage ratio on instances 1-3-2/5/8 40

4 The change in the number of populations and the coverage ratio on instances 1-3-4/5/6 41

5 The comparison of the number of populations (left) and the prediction error (right) obtained by the four algorithms, where a single population is used in GE-ES 42

6 Comparison of the change in the coverage ratio and the number of populations for the four algorithms on instances 1-3 (top) and 3-1 (bottom) 43

7 Solutions found by the four algorithm on instance 1-3 44

8 Comparison of the change in the prediction error on instance 1-3 45

9 The prediction error on case of dynamic contamination sources 46

10 Sensor concentration in the case of dynamic contamination sources 47

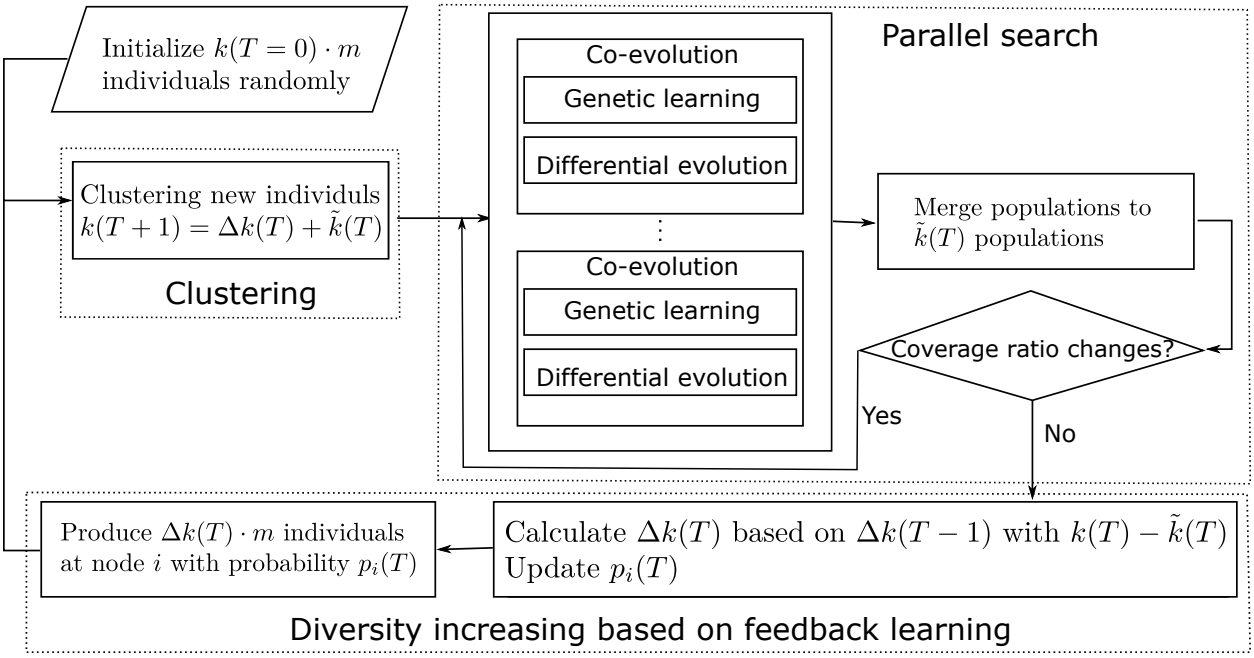

FIG. 1. Framework of the adaptive multi-population method

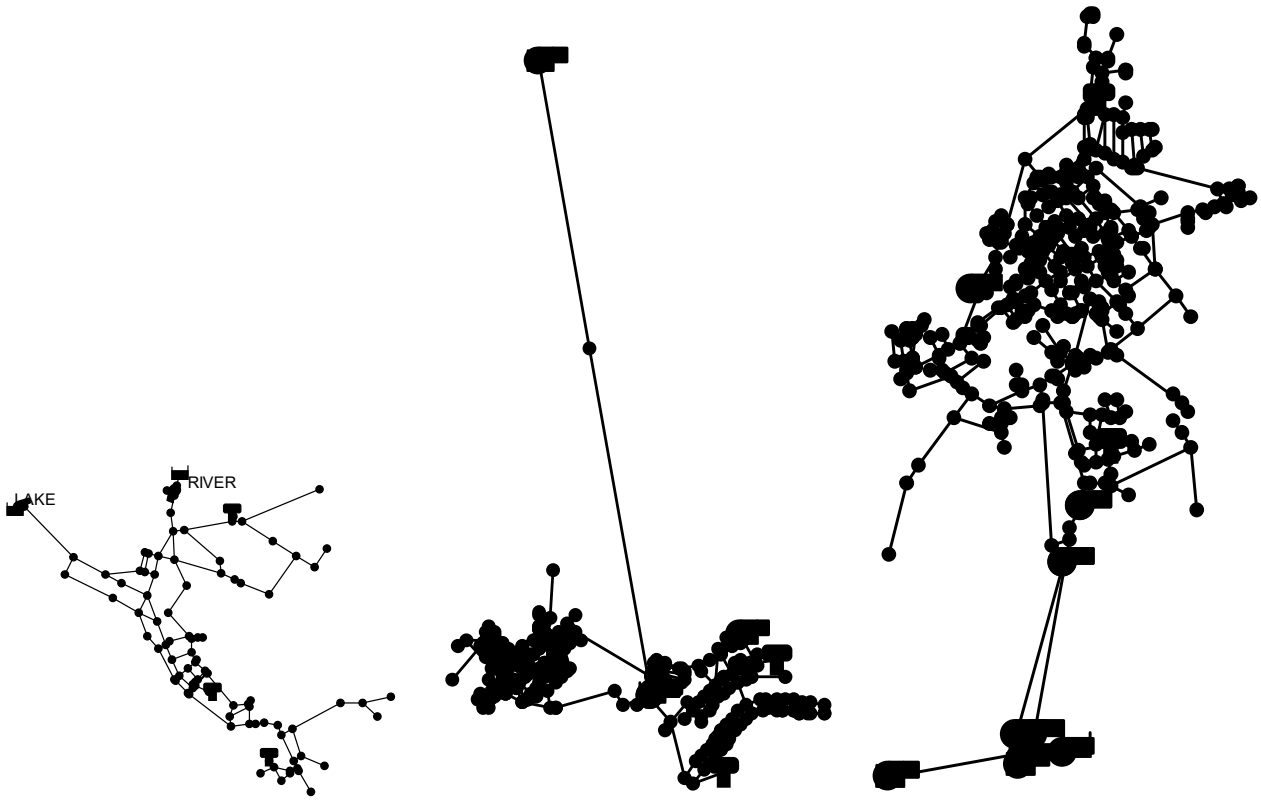

FIG. 2. The picture of three network with 97/279/430 nodes

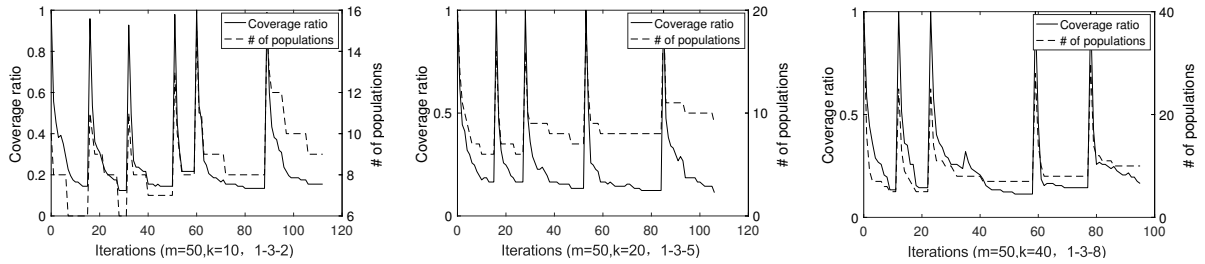

FIG. 3. The change in the number of populations and the coverage ratio on instances 1-3-2/5/8

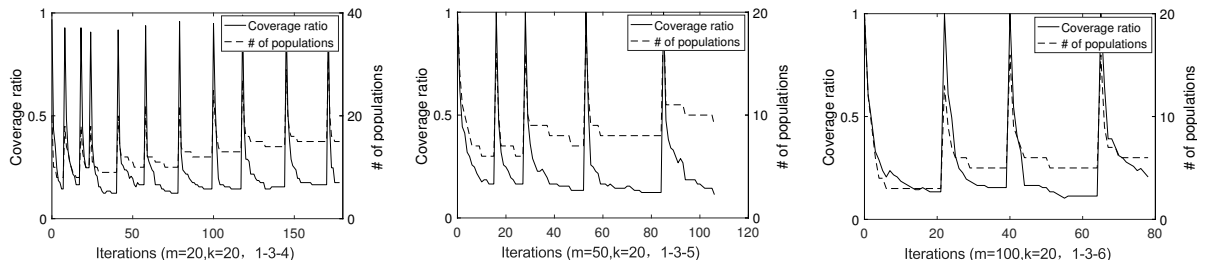

FIG. 4. The change in the number of populations and the coverage ratio on instances 1-3-4/5/6

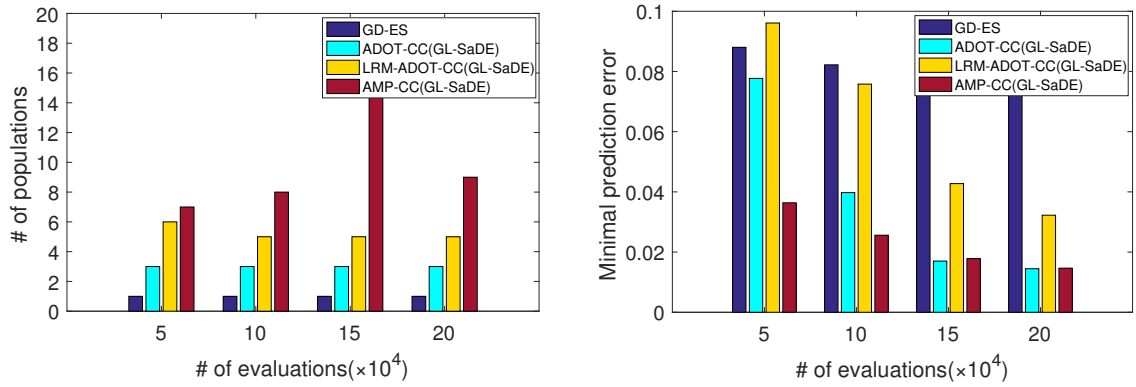

FIG. 5. The comparison of the number of populations (left) and the prediction error (right) obtained by the four algorithms, where a single population is used in GE-ES

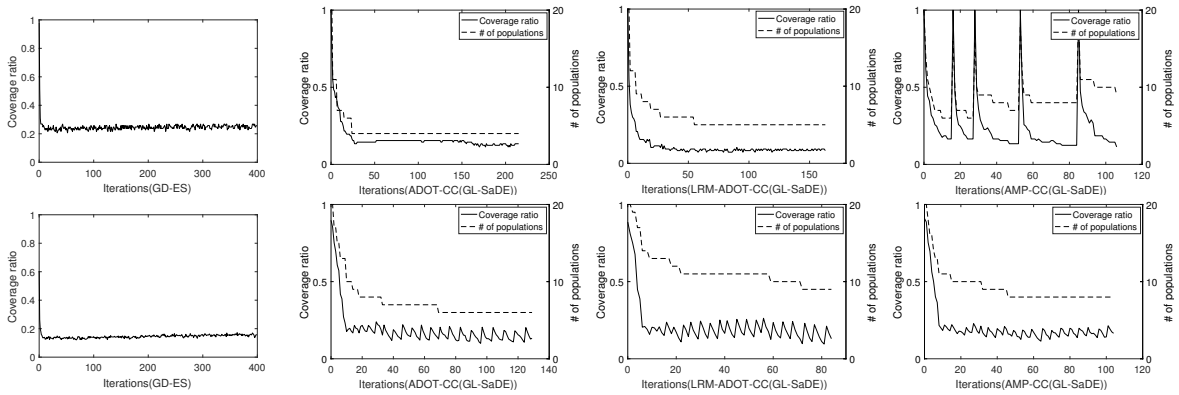

FIG. 6. Comparison of the change in the coverage ratio and the number of populations for the four algorithms on instances 1-3 (top) and 3-1 (bottom)

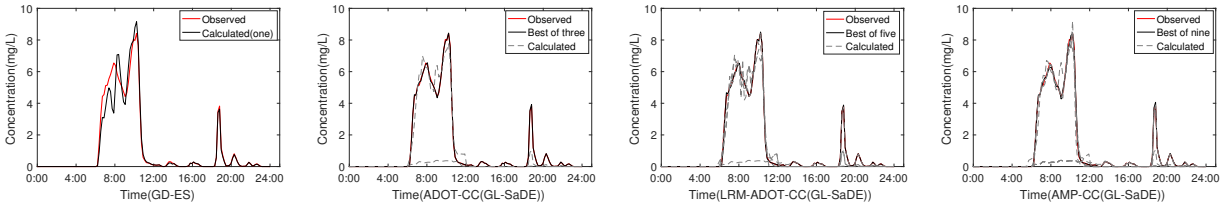

FIG. 7. Solutions found by the four algorithm on instance 1-3

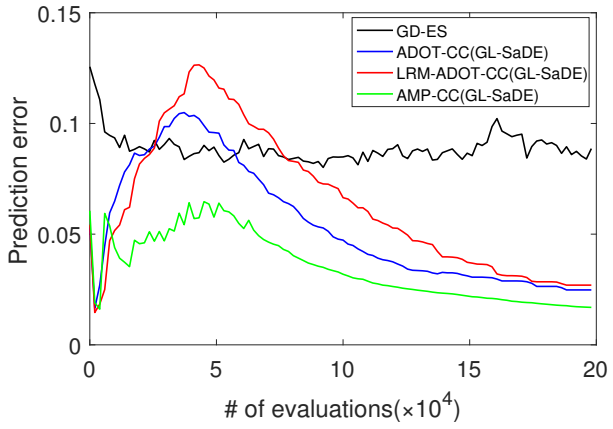

FIG. 8. Comparison of the change in the prediction error on instance 1-3

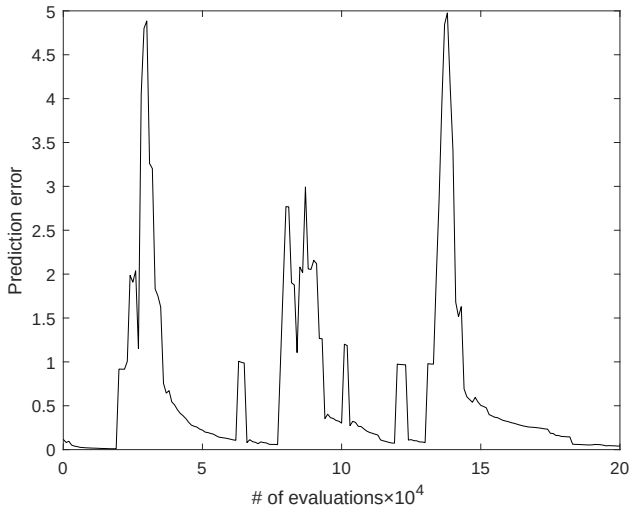

FIG. 9. The prediction error on case of dynamic contamination sources

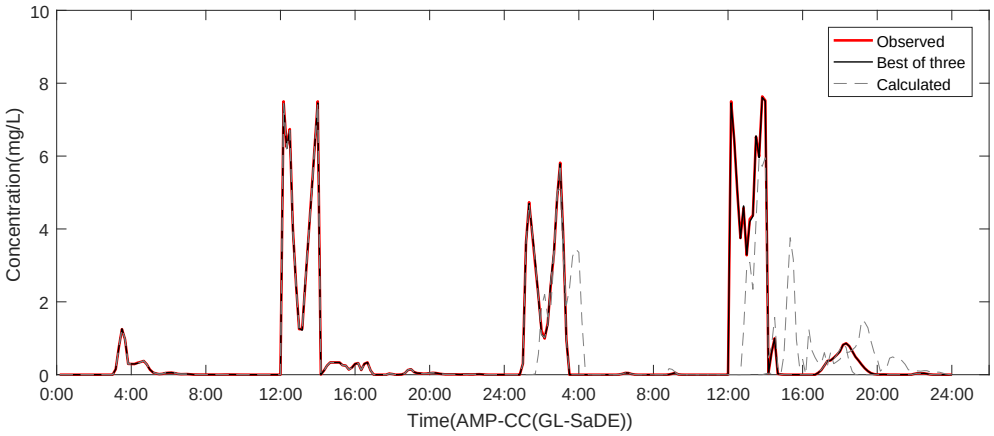

FIG. 10. Sensor concentration in the case of dynamic contamination sources