A Memetic Level-based Learning Swarm Optimizer for Large-scale Water Distribution Network Optimization

Ya-Hui Jia
Victoria University of Wellington
Wellington, New Zealand
jiayahui@ecs.vuw.ac.nz

Yi Mei
Victoria University of Wellington
Wellington, New Zealand
yi.mei@ecs.vuw.ac.nz

Mengjie Zhang
Victoria University of Wellington
Wellington, New Zealand
mengjie.zhang@ecs.vuw.ac.nz

ABSTRACT

Potable water distribution networks are requisites of modern cities. Because of the city expansion, nowadays, the scale of the network grows rapidly, which brings great difficulty to its optimization. Evolutionary computation methods have been widely investigated on small-scale networks, but their performance is far from satisfactory on large-scale networks. Aimed at addressing this difficulty, a new memetic algorithm called level-based learning swarm optimizer with restart and local search is proposed in this paper to solve the large-scale water distribution network optimization problem. Instead of using traditional evolutionary computation algorithms, the level-based learning swarm optimizer that is especially proposed for large-scale optimization problems is applied as the population-based optimizer. Two restart strategies are incorporated to make the algorithm more effective. They can help the algorithm jump out from local optima thus to increase its exploration ability. Moreover, a simple yet effective local search algorithm is proposed based on the domain knowledge to further refine the solutions after the algorithm converges. Experimental results on both single-source and multi-source large-scale water distribution networks show that the proposed algorithm is more effective than the state-of-the-art evolutionary computation algorithms.

CCS CONCEPTS

• **Computing methodologies** → **Discrete space search**; **Randomized search**; • **Applied computing** → **Computer-aided design**;

KEYWORDS

Water Distribution Network, Level-based Learning Swarm Optimizer, Local Search, Large-scale Optimization

ACM Reference Format:
Ya-Hui Jia, Yi Mei, and Mengjie Zhang. 2020. A Memetic Level-based Learning Swarm Optimizer for Large-scale Water Distribution Network Optimization. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3389828

GECCO '20, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3389828

1 INTRODUCTION

The potable water distribution network (WDN) is one of the most essential infrastructures of a modern city [29]. It transfers clean water from sources to consumers. Constructing or rehabilitating the network not only needs the cooperation of specialists from different areas but also occupy a big proportion of government expenditure [2, 9]. Thus, how to decrease the expenditure without losing sufficient water supply capacity is a challenging problem that has great realistic significance to be studied.

Generally, there are three primary components, reservoirs, pipes, and nodes (consumers) in a network. Besides these three components, some complex networks also contain some other components such as valves, pumps, emitters, and sensors. Given the layout of the network, the WDN optimization problem can be defined as the minimization of the construction expenditure by choosing suitable size for each pipe under the constraint that the minimum head pressure of each consumer should be greater than a threshold value [4]. Thus, it is a constrained optimization problem. Some studies also take this problem as a multi-objective problem where the head pressure of each consumer is defined as an objective [8, 13, 20], which may be appropriate in some scenarios like the irrigation network. However, for urban potable WDNs, there is usually a government standard of minimum pressure head [1]. Thus, in this paper, we consider this problem as a constrained optimization problem rather than a multi-objective problem.

As an NP-hard problem [28], this problem has been studied for decades. Existing methods can be classified into three categories: deterministic methods, individual-based meta-heuristic methods, and population-based evolutionary computation (EC) methods. At the very beginning, deterministic methods including linear programming [10], non-linear programming [14], and integer linear programming [23] are widely studied. Due to the NP-hard characteristic of the problem, these methods cannot generate satisfactory solutions in a certain amount of time when the network grows larger. After the birth of the simulation tool EPANET2 [22], some individual-based meta-heuristic like simulated annealing [3], tabu search [4], and iterative local search [5] were tried on this problem. However, most of these methods are easy to be trapped into local optima. Because of the good global search ability and general applicability, population-based EC algorithms have taken place of deterministic methods and individual-based methods recently [29]. Many EC algorithms have been applied to this problem [7, 12, 16–18, 20, 21, 25, 26, 31], such as genetic algorithm (GA) [17, 20, 21], ant colony optimization (ACO) [12], particle swarm optimization (PSO) [16, 25], and differential evolution (DE) [26, 31]. These algorithms have achieved very good results on many small-scale networks that contain less than 100 pipes. However, the scale of WDN grows

rapidly with the city expansion nowadays. Both the exponentially growing search space and the number of local optima bring extreme pressure to traditional EC algorithms. Theoretical studies have shown that they also suffer from the "curse of dimensionality", and many traditional EC algorithms like PSO and GA cannot handle large-scale problems effectively [15]. Either they need massive computing resources to converge or just cannot function. Considering this situation, Zheng *et al.* [30] and Chen *et al.* [1] proposed two algorithms for large-scale multi-source WDN optimization problems successively based on the idea of divide-and-conquer. Although they have successfully applied EC techniques to some large-scale WDN with hundreds pipes, only the networks with multiple sources that can be partitioned into small sub-networks are considered. For the WDNs that only have one water source or cannot be partitioned, these methods are not effective either.

Considering the difficulty of the problem and the drawbacks of the existing EC algorithms, in this paper, we propose a new memetic algorithm called level-based learning swarm optimizer with restart and local search (LLSORL) to handle the large-scale WDN optimization problems regardless of the number of sources. Memetic algorithms are known as combinations of population-based algorithms and local improvement procedures [19]. In LLSORL, the applied population-based algorithm is the level-based learning swarm optimizer (LLSO) [27]. It is especially proposed for large-scale global optimization problems. By increasing the exploration ability, LLSO has shown stronger capacity than traditional EC methods in solving problems that have more than 100 decision variables. To the best of our knowledge, this is the first time LLSO is applied to the WDN optimization problem. Although the exploration ability of LLSO is better than some traditional algorithms, the experiments on multi-modal functions show that it still encounters the local optima problem [27]. To further enhance the exploration ability of LLSO, two restart strategies are designed in LLSORL with different sizes of search space. Meanwhile, the strong exploration ability of an EA is usually accompanied by a relatively weaker exploitation ability. Thus, a simple yet effective local search algorithm is proposed to improve the exploitation ability of LLSORL. Every time LLSORL is examined to converge, the swarm will be re-initialized and the local search algorithm will be applied to the best solution of the population to make a further refinement. In the experiment, the performance of LLSORL is testified on both single-source and multi-source WDNs compared with several state-of-the-art EC methods. Moreover, the effects of the restart strategies and the local search method are also checked.

The rest of this paper is organized as follows. First, the background including the definition of the WDN optimization problem and the LLSO algorithm are introduced in Section 2. In Section 3, the proposed algorithm LLSORL is explained in detail. Experiments are conducted in Section 4. Finally, Section 5 draws the conclusion.

2 BACKGROUND

2.1 Definition of Water Distribution Network Optimization Problem

First, a well-known small-scale WDN, Hanoi System, is shown in Figure 1 to facilitate understanding the problem definition [11]. This WDN consists of one reservoir, 31 nodes, and 34 pipes. Assuming

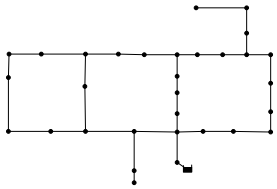

Figure 1: Hanoi System.

that the consumption velocity of each consumer, the length of each pipe, and the supply capacity of each reservoir are all known in advance, the optimization of such a WDN is to select an optimal type for each pipe to minimize the construction/rehabilitation cost while satisfying the minimum pressure head constraint and some other hydraulic constraints. A formal definition of this problem is given as follows. Given a WDN with Nn nodes, Np pipes, and Nt pipe types with different sizes, the optimization problem of WDN can be defined as:

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{Np} l_i \cdot \mu(x^i) \tag{1}$$

$$\text{s.t.} \quad x^i \in \{\zeta_1, \zeta_2, \ldots, \zeta_{Nt}\}, i = 1, \ldots, Np \tag{2}$$

$$Q_i^{\text{ext}} + \sum_{j=1}^{Nn} Q_{j,i}^{\text{in}} = Q_i^n + \sum_{k=1}^{Nn} Q_{i,k}^{\text{out}}, i = 1, \ldots, Nn \tag{3}$$

$$\Delta H_i = H_i^s - H_i^e, i = 1, \ldots, Np \tag{4}$$

$$\sum_{i \in P} \Delta H_i = H_P^s - H_P^e, P \in \mathbb{P} \tag{5}$$

$$H_{i,\min} \le H_i \le H_{i,\max}, i = 1, \ldots, Nn \tag{6}$$

where $\mathbf{x} = (x^1, x^2, \ldots, x^{Np})$ is the solution of the problem. l_i denotes the length of the ith pipe. $\mu(x_i)$ is the unit price of the x_i pipe type. $\{\zeta_1, \zeta_2, \ldots, \zeta_{Nt}\}$ are Nt commercial pipe types. Q_i^{ext} represents the external inflow water velocity of the ith node. $Q_{j,i}^{\text{in}}$ is the internal inflow water velocity from the jth node to the ith node. Q_i^n is the consumption velocity of node i. $Q_{i,k}^{\text{out}}$ is the outflow water velocity from node i to node k. ΔH_i denotes the head loss in the ith pipe. H_i^s and H_i^e are the head of each end of the ith pipe. P represents a path (a set of successively connected pipes) in the path set $\mathbb{P}$ of the network. H_P^s and H_P^e are the head of each end of the path. $H_{i,\min}$ and $H_{i,\max}$ are the minimum and maximum pressure head constraints, respectively. H_i is the real head of the ith node.

The commercial constraint (2) shows that all pipes should be selected from the commercially available types. Constraints (3), (4), and (5) are hydraulic constraints. Constraint (3) is the law of mass conservation. It implies that the velocity of inflow water should be equal to the sum of the consumption velocity and the outflow velocity. Constraints (4) and (5) shows the law of energy conservation, which imply that the head loss in a pipe or a path is equal to the difference between the heads of the start node and the end node. The standard constraint (6) stipulates that the actual head of each node should be within a government-specified range. For an urban potable WDN, usually only the minimum head constraint will be specified while the maximum head constraint is not considered

like the Chinese national standard file GB50282-98 [1]. Thus, in this paper, only $H_{i,\min}$ is considered.

Among these constraints, the commercial constraint (2) sets the boundary of the search space. The hydraulic constraints (3), (4), and (5) can be handled by the simulation tool EPANET2 [22]. Thus, during optimization, the only constraint we should deal with is the standard constraint (6). It should be also noted that although the variable type is discrete rather than continuous, we can still use continuous optimization methods since the variables are essentially ordinal rather than categorical. Before applying an EA, the pipe types should be ordered from 1 to Nt according to their diameters. Then, the variables can be treated in continuous way as usual in the algorithm. When a solution is evaluated by simulation, the integer part of the variable can be used to represent the pipe type.

2.2 Level-based Learning Swarm Optimizer

LLSO can be considered as a variant of PSO. In canonical PSO, every particle learns from the historical information including its own personal best position *pbest* and the global best position that the whole swarm ever found *gbest*. Led by *pbest* and *gbest*, PSO has a fast converging speed. Inspired by pedagogy, Yang *et al.* [27] proposed the LLSO algorithm. To increase the exploration ability of the algorithm, they have relieved the use of historical information (*pbest* and *gbest*). In LLSO, the particles will be sorted and evenly divided into different levels. Each particle will learn from two different particles in other levels that are better than itself. Assuming that there are NL levels and the first level is the best level, specifically, in the gth generation, particles will move according to:

$$\mathbf{v}_{i,j}^{g+1} \leftarrow r_1 \cdot \mathbf{v}_{i,j}^{g} + r_2 \cdot (\mathbf{x}_{a,m}^{g} - \mathbf{x}_{i,j}^{g}) + r_3 \cdot \varphi \cdot (\mathbf{x}_{b,n}^{g} - \mathbf{x}_{i,j}^{g}), \quad (7)$$

$$\mathbf{x}_{i,j}^{g+1} \leftarrow \mathbf{x}_{i,j}^{g} + \mathbf{v}_{i,j}^{g+1}, \quad (8)$$

where $\mathbf{x}_{i,j} = (x_{i,j}^1, \ldots, x_{i,j}^{Np})$ is the jth particle at the ith level and $\mathbf{v}_{i,j} = (v_{i,j}^1, \ldots, v_{i,j}^{Np})$ is its corresponding velocity, $2 \le i \le NL$. $\mathbf{x}_{a,m}$ and $\mathbf{x}_{b,n}$ are two exemplars that are better than $\mathbf{x}_{i,j}$. a, b, m, and n are randomly generated which follow $1 \le a < b < i$ when $i \ge 3$, or $a = b = 1 \wedge m < n$ when $i = 2$. The particles in the first level will not move. r_1, r_2, and r_3 are three random numbers generated within $[0, 1]$. φ is a parameter usually set to 0.4.

As to the setting of NL, a dynamic method is applied that different NL values are prepared to be chosen during the optimization. Suppose there are M different level numbers $\{NL_1, \ldots, NL_M\}$. LLSO maintains a gain list $\mathbf{G} = \{G_1, \ldots, G_M\}$ to record the relative performance improvement achieved under each level number. The value of G_i is updated in each generation as:

$$G_i = |F - \bar{F}|/F, \quad (9)$$

where F and $\bar{F}$ are the fitness values of the best solutions in the last generation and the current generation, respectively. At the beginning of each generation, a roulette wheel selection procedure is conducted to choose a level number. The probability to choose each NL_i is:

$$p_i = e^{7 \cdot G_i} / \sum_{j=1}^{M} e^{7 \cdot G_j}. \quad (10)$$

Algorithm 1 LLSORL

Input: fitness function $F(\mathbf{x})$, swarm size N, maximum generation of stagnancy T_{max}, minimum standard deviation of restart σ_{min}
Output: final solution $\hat{\mathbf{x}}$

1: initialize N particles of the swarm;
2: evaluate the fitness values of all particles and find the best one denoted as $\tilde{\mathbf{x}}$;
3: $T = 0$; $\hat{\mathbf{x}} = \tilde{\mathbf{x}}$;
4: **while** the stop criterion is not met **do**
5: $\quad \tilde{\mathbf{x}_o} = \tilde{\mathbf{x}}$;
6: $\quad$ evolve the swarm by LLSO according to (7) and (8);
7: $\quad$ find the new current best solution $\tilde{\mathbf{x}}$;
8: $\quad$ **if** $F(\hat{\mathbf{x}}) > F(\tilde{\mathbf{x}})$ **then**
9: $\quad\quad \hat{\mathbf{x}} = \tilde{\mathbf{x}}$;
10: $\quad$ **end if**
11: $\quad$ **if** $\tilde{\mathbf{x}_o} == \tilde{\mathbf{x}}$ **then**
12: $\quad\quad T++$;
13: $\quad$ **else** $T = 0$;
14: $\quad$ **end if**
15: $\quad$ **if** $T == T_{max}$ **then**
16: $\quad\quad \tilde{\mathbf{x}} = localSearch(\tilde{\mathbf{x}})$;
17: $\quad\quad$ **if** $F(\hat{\mathbf{x}}) > F(\tilde{\mathbf{x}})$ **then**
18: $\quad\quad\quad \hat{\mathbf{x}} = \tilde{\mathbf{x}}$;
19: $\quad\quad$ **end if**
20: $\quad\quad restart(\hat{\mathbf{x}}, \sigma_{min})$;
21: $\quad\quad T = 0$;
22: $\quad$ **end if**
23: **end while**
24: **return** $\hat{\mathbf{x}}$;

3 LEVEL-BASED LEARNING SWARM OPTIMIZER WITH RESTART AND LOCAL SEARCH

Although the experiments on the benchmark of large-scale global optimization problems have shown the competitiveness of LLSO, it still encounters the premature converging problem. Due to the NP-hardness, a WDN optimization problem has many local optima in the search space. Thus, to help the algorithm jump out from local optima, two restart strategies are employed considering two different scenarios. Moreover, to further exploit the area around the best solution, a simple yet effective local search method is also proposed based on the domain knowledge of the problem. The overall process of the new algorithm called LLSORL is shown Algorithm 1.

From Algorithm 1, we can see that LLSO is called in line 6. Besides this part, a variable T is set to record in how many generations there is no progress (line 5, 11-14). If it reaches a threshold value T_{max}, a local search method is called to refine the current best solution in the swarm, and a restart method is called to re-initialize the swarm (line 16-22).

Before describing the local search method and the restart method, the fitness function should be first specified. Since the WDN optimization problem is a constrained optimization problem, usually a penalty function is added to the objective function to punish the

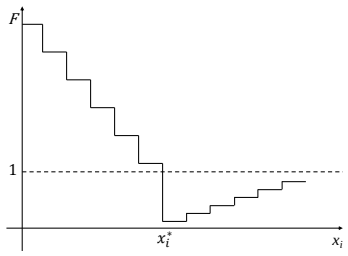

Figure 2: Fitness landscape of one variable in the WDN optimization problem, assuming that other variables are all fixed.

infeasible solutions [29]. In this paper, we adopt the fitness function proposed in [1] since it can perfectly match the constraint tournament selection strategy [6]:

$$F(\mathbf{x}) = f(\mathbf{x})/f(\mathbf{x}_{max}) + P(\mathbf{x}), \tag{11}$$

$$P(\mathbf{x}) = \sum_{i=1}^{Nn} \psi_i + \psi_i \cdot (H_{i,min} - H_i), \tag{12}$$

$$\psi_i = \begin{cases} 1 & \text{if } H_{i,min} > H_i \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

where $\mathbf{x}_{max} = (x_i = Nt | 1 \le i \le Np)$ is the biggest solution that every pipe takes the largest pipe type. According to this fitness function, all feasible solutions have fitness values smaller than or equal to 1. The fitness values of all infeasible solutions are greater than 1 since if a solution is infeasible, there must be one node whose demand cannot be satisfied and the corresponding $P(\mathbf{x})$ is greater than 1.

3.1 Local Search

According to the definition of the problem, we can know that the best solution is on the edge between the feasible zone and the infeasible zone in the search space. Assuming that we have an optimal solution, for a decision variable x_i in this solution, if we increase its value (replace it with a larger pipe), the total cost must be increased and the solution is still feasible; if we decrease its value (replace it with a smaller pipe), there will be at least one consumer whose demand cannot be satisfied. Based on the fitness definition, a figure of fitness landscape of one variable is shown in Figure 2. Usually, LLSO can find feasible solutions easily, which means in the vast majority of cases, the local search method works on a feasible solution. Under the circumstances, we try to decrease the sizes of some pipes to push the refined solution to the edge of the feasible zone. Algorithm 2 shows the proposed local search method.

First, all variables that are greater than 1 are put into a list. Each time before we decrease the values of the variables, the list is shuffled. Then, the variables in the list are checked one by one. If the value of a variable can be decreased without violating the constraint, we keep it in the list. Otherwise, it will be removed from the list, and the value of the variable keeps unchanged. Finally, when the list is empty, the local search procedure ends and returns the refined solution.

Algorithm 2 Local Search

Input: fitness function $F(\mathbf{x})$, a solution $\mathbf{x}$.
Output: a refined solution $\mathbf{x}$.

```
get all variables greater than 1, list = {x^i | x^i > 1 ∧ 1 ≤ i ≤ Np};
while list is not empty do
    shuffle list;
    for all x^i in list do
        x^i = x^i − 1;
        if F(x) > 1 then
            x^i = x^i + 1;
            erase x^i from list;
        else if x^i == 1 then
            erase x^i from list;
        end if
    end for
end while
return x;
```

Through applying the local search method, some pipes that are originally oversize now become smaller. Thus, the total cost of the solution is reduced without violating the minimum pressure constraint.

3.2 Restart

Two questions should be considered to design a restart method for an EA. 1) When should the restart method be executed? 2) How to re-initialize the individuals? For the first question, a parameter T_{max} is set to determine the time of executing the restart procedure. If the algorithm cannot find better solutions for successive T_{max} generations, the restart procedure will be executed.

For the second question, two different restart strategies are incorporated considering two different scenarios. 1) If the complexity of the problem is relatively simple and LLSO can locate the promising area in the search space, we re-initialize the particles in the neighborhood of the best solution that the algorithm has ever found. This restart strategy is called the local restart and the corresponding algorithm is denoted as LLSORL-L. 2) Otherwise, the global best solution may not be in the neighborhood of the discovered solution. Thus, we re-initialize the particles in the whole search space. This strategy is called the global restart strategy and the corresponding algorithm is denoted as LLSORL-G.

As Algorithm 3 shows, a lower bound and an upper bound around the best solution define the search space after restart. The value of each variable is re-initialized by a uniform distribution U(lb, ub).

Algorithm 3 Restart

Input: fitness function $F(\mathbf{x})$, the best solution $\hat{\mathbf{x}}$, minimum standard deviation σ_{min}.

```
for i = 1 → Np do
    lb = max(x̂^i − σ_min, 1);
    ub = min(x̂^i + σ_min, Nt);
    re-initialize x_j^i = U(lb, ub), j = {1, ..., N};
end for
calculate the fitness values of all particles;
```

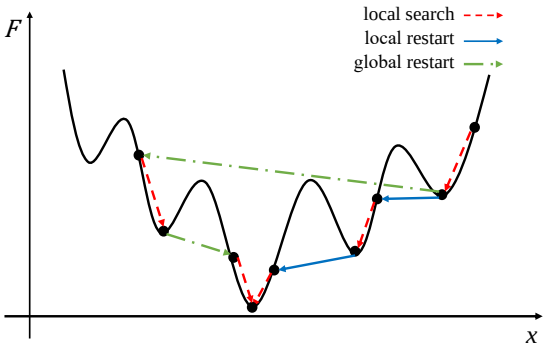

Figure 3: The behavior of the local search method and the two restart strategies.

For the local restart strategy, σ_{min} is set to a relatively small value to generate a small neighborhood near the best solution. For the global restart strategy, σ_{min} is just set to Nt to cover the whole search space.

Figure 3 shows a visual explanation how the local search method and the restart strategies help the algorithm to find better solutions. The local search method pushes a solution into a nearest local optimum. The local restart strategy moves the swarm in small steps, while the step size of the global restart strategy is totally random since after restart, the swarm will search the solution space from scratch.

4 EXPERIMENT

4.1 Experimental Setup

4.1.1 Test Cases. Most of the previous works focus on small-scale WDNs. These WDNs come from different cities or systems which vary greatly. To make systematic comparisons between different algorithms on large-scale WDNs, a set of synthetic networks are created in [1] that follows a same design method. However, these WDNs are all multi-source networks. To conduct a more comprehensive experiment, five imbalanced WDNs are adopted, and we also modified them to be single-source networks. Besides the synthetic WDNs, a real-world WDN, Balerma, is also adopted [21]. The information of these networks is shown in Table 1 and the structures of MN400 and Balerma are shown in Fig. 4 to demonstrate the complexities of the networks. For the synthetic WDNs, there are 26 pipe types available. For the Balerma network, there are 10 pipe types.

4.1.2 Compared Algorithms. To show the advantages of LLSORL against other EC algorithms, several representative EC algorithms are compared, including self-adaptive DE (SADE) [31], shuffled frog leaping algorithm (SFLA) [18], and WDNCC [1]. According to the experiments conducted in [1], SADE has significantly better performance than the canonical PSO and a recently proposed variant called developed swarm optimizer [24]. It is also competitive compared with the max-min ant system which is one of the most effective ACO methods for the WDN optimization problem [12]. SFLA is an optimization algorithm especially proposed for discrete optimization problems, and its new version has been proved effective on WDN optimization problems [18]. WDNCC is a cooperative co-evolution algorithm that is proposed for multi-source WDNs. Besides these three EAs, the original LLSO and the local search method proposed in this paper are also tested. The local search method, denoted as LS, is directly executed on $\mathbf{x}_{max}$ for each test case. $\mathbf{x}_{max}$ is the most expensive solution that every pipe chooses the largest type. Since WDNCC can only work on multi-source WDNs, it will not be tested on the single-source test cases.

In previous studies, the parameters of the applied algorithms are usually tuned separately for each specific WDN. However, it is unrealistic for us to tune the parameters of every tested algorithm on each test case, and it is not our focus either. Generally, according to the analysis made in [16] and [31], when using EAs to solve WDN optimization problems, setting the population size to the scale of the network can bring relatively good performance. Thus, for all the algorithms including the two LLSORL methods, the population size is set to the number of nodes. The population size of each algorithm on the Balerma network is set to 400. For WDNCC, the re-decomposition interval is set to 200. For SADE, the ranges of Fc and CR are set to [0.1, 0.9]. For SFLA, the number of frogs in each memeplex is set to 20. The maximum number of fitness evaluations (FEs) is limited to $Nt \cdot 4000$. Each method will be executed 20 independent times. The Wilcoxon rank-sum test is conducted

Table 1: Test Cases of WDNs

Name	#Node	#Pipe	#Reservoir
SN200	200	226	1
SN300	300	328	1
SN400	400	452	1
SN500	500	546	1
MN200	200	226	2
MN300	300	328	3
MN400	400	452	4
MN500	500	546	5
Balerma	443	454	4

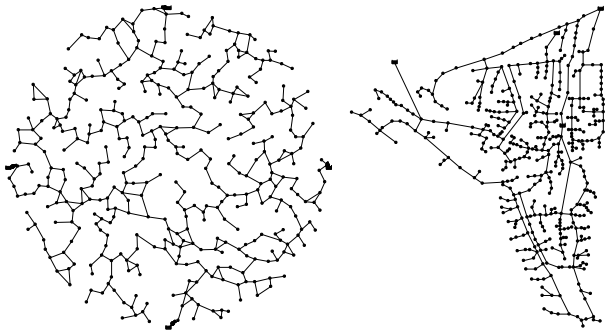

Figure 4: The structures of MN400 and Balerma water distribution networks.

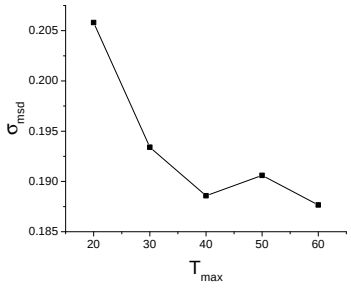

Figure 5: σ_{msd} under different T_{max} values.

between two LLSORL methods and each compared algorithm, and it is also conducted between LLSORL-L and LLSORL-G.

4.2 Parameter Selection

There are two parameters in LLSORL. The first one is the maximum generation of stagnancy T_{max}. The second one is the minimum standard deviation σ_{min} used in the restart strategy. For σ_{min} of LLSORL-L, we set it to 3 for the synthetic WDNs and to 2 for the Balerma networks. Generally, based on our empirical study, setting σ_{min} to $\max(Nt/8, 2)$ would be a applicable choice. For σ_{min} of LLSORL-G, it is set to Nt as aforementioned.

As to T_{max}, we compared 5 candidate values {20, 30, 40, 50, 60}. Choosing SN200 as the test case, we calculate the maximum standard deviation of each variable among all variables as the measurement to see when LLSO is converged, $\sigma_{msd} = \max(\sigma_1, \ldots, \sigma_{Np})$. It represents the largest difference of particles on one dimension. A small σ_{msd} means that even on the dimension that particles have largest difference with each other, the difference is actually very small. Under such circumstances, the algorithm has already converged. When T reaches different T_{max} values, σ_{msd} is calculated. The experiment is conducted 20 times to obtain the mean value. The results are shown in Figure 5.

From the results, we can see that when T_{max} grows from 20 to 40, σ_{msd} has clear decrease. However, from 40 to 60, there is no significant decrease of σ_{msd}. Instead, the value of σ_{msd} starts fluctuating. This phenomenon tells us that when T_{max} is equal to or greater than 40, the tendency of gathering of these particles slows down. The particles start wandering in a very small area which means the algorithm has basically converged. Thus, in the following experiment T_{max} is set to 40.

4.3 Comparison on Single-source Networks

First, all algorithms except WDNCC are tested on the single-source WDNs. Experimental results are shown in Table 2. The best mean values are highlighted. Since each algorithm is compared with LLSORL-L and LLSORL-G separately, there will be two arrows representing the results of Wilcoxon rank-sum tests, and there is only one arrow behind the result of LLSORL-L shows the comparison between LLSORL-L and LLSORL-G. $\downarrow$ and $\uparrow$ represent that the compared algorithm is significantly worse/better than LLSORL-L or LLSORL-G according to the significance level 0.05, respectively. $\rightarrow$ means that the two compared algorithms are well matched.

From Table 2, we can get following observations:

(1) Both LLSORL methods are generally significantly better than the other compared algorithms. Only on SN300, LLSO has similar performance with LLSORL-L. Between LLSORL-L and LLSORL-G, the results show that on SN200 and SN300, they have similar performance, but on SN400 and SN500, LLSORL-L is better according to the Wilcoxon rank-sum test. However, we cannot make the conclusion that LLSORL-L is definitely better than LLSORL-G yet since LLSORL-L has higher standard deviation. The pros and cons of these two methods will be discussed after the comparisons on the multi-source networks.
(2) Even without using local search and restart techniques, LLSO is better than SADE and SFLA according to the mean values, especially on large instances. The results testify that LLSO is truly more effective on large-scale optimization problems than some traditional EAs.
(3) The performance of SADE is highly affected by the network scale. It only has competitive performance on SN200 compared with LLSO.
(4) The local search method truly works. Starting from $\mathbf{x}_{max}$, it can reduce the cost a lot meanwhile guarantee the feasibility of the solution. Compared with the other algorithms, actually it is not the worst one. On each WDN, it is better than SFLA.

4.4 Comparison on Multi-source Networks

Following, all algorithms are tested on the multi-source WDNs including the Balerma network. Experimental results are shown in Table 3.

The results shown in Table 3 also verify the effectiveness of LLSORL. Specifically, the results show that:

(1) Two LLSORL methods are significantly better than the other algorithms on most of the test cases. Only on MN500, LLSO has got competitive performance. Actually according to the Wilcoxon rank-sum test, the advantage of LLSORL against LLSO decreases gradually with the growth of the WDN scale. Although this phenomenon does not appear on single-source WDNs, it is actually reasonable. When the scale of the network grows larger, the search space and the number of local optima also grow rapidly. Thus, the restart strategy will be less and less efficient, but generally the restart strategies and the local search method are always helpful more or less.
(2) This time, LLSORL-L and LLSORL-G are well matched according to the Wilcoxon rank-sum test. Although the mean values show that LLSORL-G is better, LLSORL-L still has larger standard deviation. Thus, we still cannot make a simple conclusion to say which one is better.
(3) Since a multi-source WDN are divided into sub-networks in WDNCC, its effectiveness does not degrade with the increase of scale. However, due to the same reason, it is hard for WDNCC to find a global best since the combination of the optimal sub-solutions is not always the best.
(4) SADE is still affected by the WDN scale greatly. It is more effective on small-scale networks than on large-scale networks compared with WDNCC, LLSO, and LLSORL.

Overall, on both single-source and multi-source WDNs, the two LLSORL algorithms has shown great superiority against the other

Table 2: Experimental Results on Single-source WDNs

Case	Index	LS	SADE	SFLA	LLSO	LLSORL-L	LLSORL-G
SN200	mean	1.02E-01↓↓	5.51E-02↓↓	3.08E-01↓↓	5.63E-02↓↓	**5.21E-02**→	5.24E-02
	std.	3.40E-03	8.85E-04	9.91E-02	2.46E-03	1.55E-03	5.86E-04
SN300	mean	1.84E-01↓↓	1.10E-01↓↓	9.08E-01↓↓	9.81E-02→↓	9.66E-02→	**9.46E-02**
	std.	7.80E-03	1.77E-03	1.34E+00	3.68E-03	3.19E-03	1.70E-03
SN400	mean	2.03E-01↓↓	1.16E-01↓↓	5.59E-01↓↓	8.28E-02↓↓	**8.06E-02**↑	8.17E-02
	std.	9.70E-03	2.47E-03	8.14E-02	1.34E-03	7.11E-04	8.71E-04
SN500	mean	2.86E-01↓↓	2.22E-01↓↓	5.02E+00↓↓	1.79E-01↓↓	**1.42E-01**↑	1.44E-01
	std.	1.28E-02	4.64E-03	1.60E+01	4.20E-03	3.36E-03	1.75E-03

Table 3: Experimental Results on Multi-source Networks

Case	Index	LS	SADE	SFLA	WDNCC	LLSO	LLSORL-L	LLSORL-G
MN200	mean	5.24E-02↓↓	3.49E-02↓↓	1.40E-01↓↓	3.54E-02↓↓	3.42E-02↓↓	**3.17E-02**→	3.18E-02
	std.	1.96E-03	5.57E-04	7.19E-02	2.02E-03	7.83E-04	5.99E-04	3.37E-04
MN300	mean	8.77E-02↓↓	6.57E-02↓↓	3.97E-01↓↓	5.50E-02↓↓	5.35E-02↓↓	5.18E-02→	**5.11E-02**
	std.	3.17E-03	9.83E-04	1.69E-01	2.18E-03	2.61E-03	2.35E-03	4.56E-04
MN400	mean	1.08E-01↓↓	8.55E-02↓↓	1.25E+01↓↓	6.32E-02↓↓	5.65E-02↓↓	5.40E-02→	**5.34E-02**
	std.	5.49E-03	3.00E-03	3.69E+01	6.12E-03	4.75E-03	3.21E-03	8.77E-04
MN500	mean	2.20E-01↓↓	2.18E-01↓↓	8.18E-01↓↓	1.04E-01↓↓	9.55E-02→→	9.39E-02→	**9.30E-02**
	std.	6.83E-03	1.35E-02	5.41E-01	7.46E-03	4.69E-03	4.58E-03	2.57E-03
Balerma	mean	1.34E-01↓↓	9.66E-02↓↓	8.50E+00↓↓	9.59E-02↓↓	9.41E-02↓↓	9.32E-02→	**9.22E-02**
	std.	1.04E-02	5.75E-04	2.16E+01	3.37E-03	1.46E-03	1.33E-03	6.72E-04

compared algorithms. Thus, we can first conclude that both LLSORL algorithms are effective on large-scale WDN optimization problems. Next, the difference between LLSORL-L and LLSORL-G are discussed.

4.5 Comparison between Two Restart Strategies

The results of these two algorithms on SN200, SN500, MN200, and MN500 are shown in the form of box chart in Figure 6. From the figures we can see that the performance of LLSORL-G is more stable than LLSORL-L, but LLSORL-L can always find better solutions than LLSORL-G. The median values and best values show that LLSORL-L is better than LLSORL-G. This case is different from the one indicated by the mean values in Table 3.

The reason of this situation is straightforward. The local restart strategy always re-initializes the swarm in the neighborhood that is near the best solution that the algorithm has ever found, denoted as $\hat{\mathbf{x}}$. Usually, the quality of the solutions in the neighborhood has relatively small difference with $\hat{\mathbf{x}}$ so that the local restart strategy can only make a little improvement each time. This little improvement has a big probability to be achieved since the whole swarm only searches a small space rather than the whole solution space. Thus, the quality of the final solution of LLSORL-L is highly related to the solution that LLSORL-L found before the first restart which is exact the one LLSO finds. If that solution is poor, the final solution has a large probability to be poor. If that solution is good, the final solution can be even better. Since the performance of LLSO itself

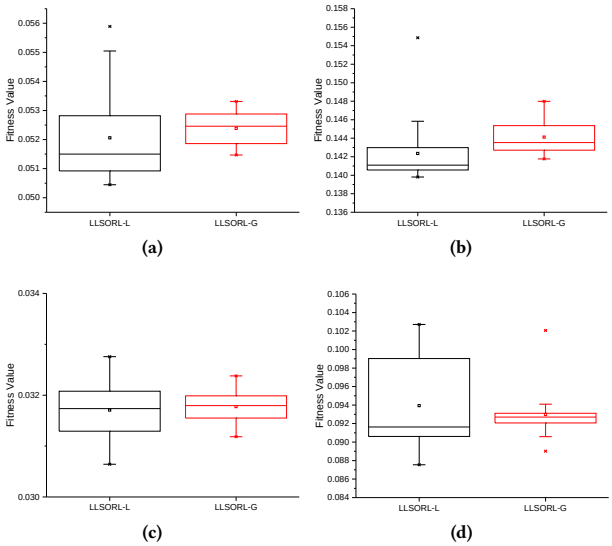

Figure 6: Comparison between LLSORL-L and LLSORL-G. (a) SN200, (b) SN500, (c) MN200, (d) MN500.

is not stable just like the standard deviations of LLSO shown in Table 2 and Table 3, the performance of LLSORL-L is not stable either. However, because of the large probability to make the little improvement after each restart, when the solution found before the

first restart is good, the local restart strategy can find some very good solutions that the global restart strategy cannot find.

In the global restart strategy, each time the swarm is re-initialized to search the whole solution space, which means the solutions found after each restart are relatively independent of each other. Thus, if choosing the best one after several times of restarts, the performance of LLSORL-G becomes stable since the bad outliers are abandoned. However, each time the swarm is re-initialized, the algorithm actually restarts from scratch. The improvement cannot be guaranteed. Its exploitation ability in a small area is worse than using local restart so that it cannot find the very good solutions.

Overall, both restart strategies have advantages and disadvantages against each other. If the applied optimizer is powerful enough to successfully locate the neighborhood of the global best solution, the local restart strategy is preferred. If the performance of the applied optimizer is not very stable, the global restart strategy will be a good choice. Nevertheless, how to utilize the advantages and eliminate the disadvantages of these two strategies would be a good direction for future research.

4.6 Convergence Behavior of LLSORL

From the experimental results, we can see that LLSO itself sometimes is already very effective but the two LLSORL algorithms can always improve the performance of LLSO to a better level. The reason that LLSORL is better than the original LLSO is revealed by showing the convergence behavior of the two LLSORL algorithms on SN200. For the two LLSORL algorithms, both the best solution in the swarm (denoted as LLSORL-SB) and the best solution that the algorithm ever found (denoted as LLSORL-EB) are drawn in the figures. LLSO is also drawn for comparison.

First, from Figure 7 we can see that the original LLSO (black line) is easy to be trapped into to a local optimum and it has no ability to jump out. Regarding the two LLSORL algorithms, they both have the ability to improve the performance. This ability comes from both the local search method and the restart strategy. From each figure, we can see some sudden drops of LLSORL-EB (red line) which represent the improvements caused by the local search method. The restart strategy also contributes where we can see that LLSORL-SB (blue line) surpasses LLSORL-EB, which means after one restart, the swarm has successfully found a better solution.

Then, Figure 7(a) shows that there are ten times restart of LLSORL-L, and Figure 7(b) shows seven times restart of LLSORL-G. Since the swarm in LLSORL-L only search a small area after each restart, its convergence speed is quicker than LLSORL-G. Thus, it has more times of restart. These two figures also perfectly verify the analysis we made in the previous subsection. The local restart strategy has a greater success rate that in nine times out of the ten times restart, the algorithm has made improvements, but every improvement is not very significant. For the global restart strategy, both the success rate and the significance of the improvement of each restart are quite random that cannot be predicted.

5 CONCLUSIONS

The goal of this paper is to promote the usage of EC algorithms in solving large-scale WDN optimization problems. This goal has been successfully achieved by proposing a new memetic algorithm called

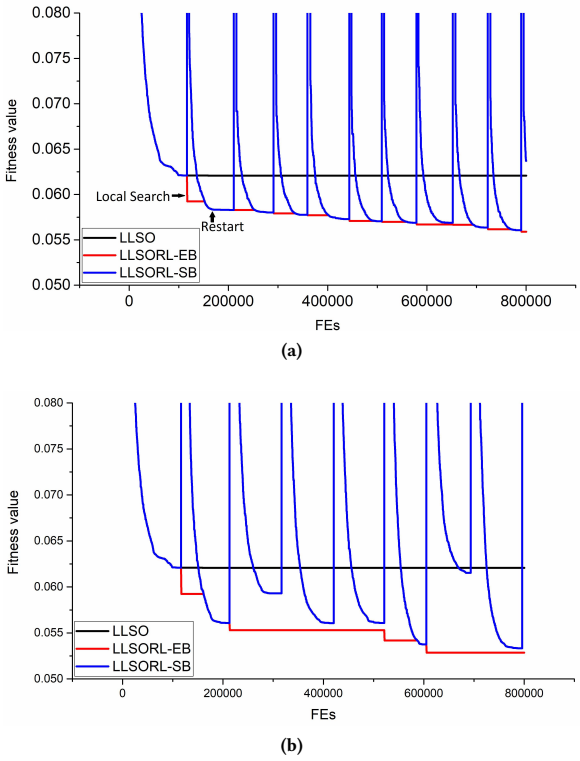

Figure 7: The convergence behaviors of LLSORL-L and LLSORL-G. (a) LLSORL-L, (b) LLSORL-G.

LLSORL. To the best of our knowledge, this is the first time an EC algorithm that was especially proposed for large-scale global optimization problems, is applied to the large-scale WDN optimization problem. Not only because of the usage of LLSO, but the proposed two restart strategies and local search method also contribute a lot to the success of LLSORL. Experiments have verified that both restart strategies has the ability to help LLSO jump out from local optima but their applicable scenarios are different. The local search method is also testified to be effective to refine the best solution. Combining these two techniques together, both the exploration and exploitation abilities of LLSO on the studied problem have been improved. Thus, LLSORL has shown significant advantages compared with the other algorithms.

In future research, there are still a number of directions that we can follow to solve the problem more effectively. First, the applied two restart methods have shown different characteristics in the experiments. How to combine them together to use the advantages of both methods is a problem that is worth studying. Second, although the experiments in this paper show that LLSO is better than some traditional EC algorithms, its performance is still not very stable. Thus, more powerful optimizers should be proposed for and applied to large-scale optimization problems.

REFERENCES

[1] Wei-Neng Chen, Ya-Hui Jia, Feng Zhao, Xiao-Nan Luo, Xing-Dong Jia, and Jun Zhang. 2019. A cooperative co-evolutionary approach to large-scale multisource

water distribution network optimization. *IEEE Transactions on Evolutionary Computation* (2019).

[2] B Coelho and A Andrade-Campos. 2014. Efficiency achievement in water supply systems—A review. *Renewable and Sustainable Energy Reviews* 30 (2014), 59–84.

[3] Maria da Conceicao Cunha and Joaquim Sousa. 1999. Water distribution network design optimization: simulated annealing approach. *Journal of water resources planning and management* 125, 4 (1999), 215–221.

[4] Maria da Conceicao Cunha and Luisa Ribeiro. 2004. Tabu search algorithms for water network optimization. *European Journal of Operational Research* 157, 3 (2004), 746–758.

[5] Annelies De Corte and Kenneth Sörensen. 2016. An Iterated Local Search Algorithm for multi-period water distribution network design optimization. *Water* 8, 8 (2016), 359.

[6] Kalyanmoy Deb. 2000. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering* 186, 2-4 (2000), 311–338.

[7] Muzaffar M Eusuff and Kevin E Lansey. 2003. Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources planning and management* 129, 3 (2003), 210–225.

[8] R Farmani, DA Savic, and GA Walters. 2005. Evolutionary multi-objective optimization in water distribution network design. *Engineering Optimization* 37, 2 (2005), 167–183.

[9] Raziyeh Farmani, Godfrey Walters, and Dragan Savic. 2006. Evolutionary multi-objective optimization of the design and operation of water distribution network: total cost vs. reliability vs. water quality. *Journal of Hydroinformatics* 8, 3 (2006), 165–179.

[10] O Fujiwara, B Jenchaimahakoon, and NCP Edirishinghe. 1987. A modified linear programming gradient method for optimal design of looped water distribution networks. *Water Resources Research* 23, 6 (1987), 977–982.

[11] Okitsugu Fujiwara and Do Ba Khang. 1990. A two-phase decomposition method for optimal design of looped water distribution networks. *Water resources research* 26, 4 (1990), 539–549.

[12] Consolación Gil, Raul Baños, Julio Ortega, Antonio López Márquez, Antonio Fernández, and MG Montoya. 2011. Ant colony optimization for water distribution network design: a comparative study. In *International Work-Conference on Artificial Neural Networks*. Springer, 300–307.

[13] Matthew B Johns, Herman A Mahmoud, David J Walker, Nicholas DF Ross, Edward C Keedwell, and Dragan A Savic. 2019. Augmented evolutionary intelligence: combining human and evolutionary design for water distribution network optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1214–1222.

[14] Kevin E Lansey and Larry W Mays. 1989. Optimization model for water distribution system design. *Journal of Hydraulic Engineering* 115, 10 (1989), 1401–1418.

[15] Xiaodong Li and Xin Yao. 2011. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation* 16, 2 (2011), 210–224.

[16] Idel Montalvo, Joaquín Izquierdo, Rafael Pérez, and Michael M Tung. 2008. Particle swarm optimization applied to the design of water supply systems. *Computers & Mathematics with Applications* 56, 3 (2008), 769–776.

[17] Pilar Montesinos, Adela Garcia-Guzman, and Jose Luis Ayuso. 1999. Water distribution network optimization using a modified genetic algorithm. *Water Resources Research* 35, 11 (1999), 3467–3473.

[18] Daniel Mora-Melia, Pedro Iglesias-Rey, F Martínez-Solano, and Pedro Muñoz-Velasco. 2016. The efficiency of setting parameters in a modified shuffled frog leaping algorithm applied to optimizing water distribution networks. *Water* 8, 5 (2016), 182.

[19] Pablo Moscato and Carlos Cotta. 2003. A gentle introduction to memetic algorithms. In *Handbook of metaheuristics*. Springer, 105–144.

[20] T Devi Prasad and Nam-Sik Park. 2004. Multiobjective genetic algorithms for design of water distribution networks. *Journal of Water Resources Planning and Management* 130, 1 (2004), 73–82.

[21] Juan Reca and Juan Martínez. 2006. Genetic algorithms for the design of looped irrigation water distribution networks. *Water resources research* 42, 5 (2006).

[22] Lewis A Rossman et al. 2000. *EPANET 2: users manual*.

[23] Hossein MV Samani and Alireza Mottaghi. 2006. Optimization of water distribution networks using integer linear programming. *Journal of Hydraulic Engineering* 132, 5 (2006), 501–509.

[24] R Sheikholeslami and S Talatahari. 2016. Developed swarm optimizer: A new method for sizing optimization of water distribution systems. *Journal of Computing in Civil Engineering* 30, 5 (2016), 04016005.

[25] CR Suribabu and TR Neelakantan. 2006. Design of water distribution networks using particle swarm optimization. *Urban Water Journal* 3, 2 (2006), 111–120.

[26] A Vasan and Slobodan P Simonovic. 2010. Optimization of water distribution network design using differential evolution. *Journal of Water Resources Planning and Management* 136, 2 (2010), 279–287.

[27] Qiang Yang, Wei-Neng Chen, Jeremiah Da Deng, Yun Li, Tianlong Gu, and Jun Zhang. 2017. A level-based learning swarm optimizer for large-scale optimization. *IEEE Transactions on Evolutionary Computation* 22, 4 (2017), 578–594.

[28] DF Yates, AB Templeman, and TB Boffey. 1984. The computational complexity of the problem of determining least capital cost designs for water supply networks. *Engineering Optimization* 7, 2 (1984), 143–155.

[29] Wanqing Zhao, Thomas H Beach, and Yacine Rezgui. 2015. Optimization of potable water distribution and wastewater collection networks: A systematic review and future research directions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46, 5 (2015), 659–681.

[30] Feifei Zheng, Angus R Simpson, and Aaron C Zecchin. 2013. A decomposition and multistage optimization approach applied to the optimization of water distribution systems with multiple supply sources. *Water Resources Research* 49, 1 (2013), 380–399.

[31] Feifei Zheng, Aaron C Zecchin, and Angus R Simpson. 2012. Self-adaptive differential evolution algorithm applied to water distribution system optimization. *Journal of Computing in Civil Engineering* 27, 2 (2012), 148–158.