A Cooperative Co-Evolutionary Approach to Large-Scale Multisource Water Distribution Network Optimization

Wei-Neng Chen, *Senior Member, IEEE*, Ya-Hui Jia, Feng Zhao, Xiao-Nan Luo, Xing-Dong Jia, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Potable water distribution networks (WDNs) are important infrastructures of modern cities. A good design of the network can not only reduce the construction expenditure but also provide reliable service. Nowadays, the scale of the WDN of a city grows dramatically along with the city expansion, which brings heavy pressure to its optimal design. In order to solve the large-scale WDN optimization problem, a cooperative co-evolutionary algorithm is proposed in this paper. First, an iterative trace-based decomposition method is specially designed by utilizing the information of water tracing to divide a large-scale network into small subnetworks. Since little domain knowledge is required, the decomposition method has great adaptability to multiform networks. Meanwhile, during optimization, the proposed algorithm can gradually refine the decomposition to make it more accurate. Second, a new fitness function is devised to handle the pressure constraint of the problem. The function transforms the constraint into a part of the objective to punish the infeasible solutions. Finally, a new suite of benchmark networks are created with both balanced and imbalanced cases. Experimental results on a widely used real network and the benchmark networks show that the proposed algorithm is promising.**

***Index Terms*—Cooperative co-evolution, divide-and-conquer, evolutionary algorithm (EA), hydraulics, large-scale optimization, network optimization, water distribution networks (WDNs).**

Manuscript received May 31, 2018; revised September 9, 2018 and November 15, 2018; accepted January 6, 2019. Date of publication January 17, 2019; date of current version October 1, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61332002, and Grant 61876111, in part by the Natural Science Foundation of Guangdong under Grant 2015A030306024, and in part by the Science and Technology Plan Project of Guangdong Province under Grant 2018B050502006.

W.-N. Chen and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China (e-mail: cwnraul634@aliyun.com; junzhang@ieee.org).

Y.-H. Jia is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

F. Zhao is with Yulin Normal University, Yulin 537000, China.

X.-N. Luo is with the School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China.

X.-D. Jia is with Shenzhen Polytechnic, Shenzhen 518055, China.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. This is a PDF file containing a comparison between WDNCC and SaNSDE and information about execution time. The total size is 363 KB.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2019.2893447

I. Introduction

POTABLE water distribution networks (WDNs) are essential components of the urban water infrastructure, which are built to transfer potable water from sources to consumers. Building new WDNs or rehabilitating existing networks is one of the most pressing tasks faced by governments or service suppliers, because the capital cost of these networks would occupy a big proportion of government expenditures [1]. Meanwhile, the construction of a WDN is related with many practical factors, such as the government standards, other infrastructures, and topographies [2]. Thus, how to reduce the capital cost while guaranteeing the quality of service (usually defined by the tap water pressure, water age, chlorinity, etc.) becomes a very challenging and practical problem.

If we take the design of WDNs as an optimization problem, it can be defined as finding appropriate settings of water network components (e.g., diameter of pipes) for a pregiven network layout, so that the capital cost is minimized subject to a number of hydraulic, physical, and standardized constraints (standardized constraints mean market or government standards) [3]–[5]. The problem has been proven nondeterministic polynomial-time hard (NP-hard) [6]. To solve this problem, many methods have been proposed in the past few decades. Roughly, they can be classified into two categories: 1) deterministic methods and 2) metaheuristic methods. Among deterministic methods, linear programming (LP) and its variants were first proposed in earlier time, attempting to solve the problem with low computational burden [7], [8]. However, due to the nonlinear essence of the problem, LP methods tend to fall into local optima easily [9]. Then, some non LP (NLP) methods [10], [11] were proposed. But these methods are still not effective enough to find the optimal or near-optimal solutions, since the final solution generated by an NLP method highly depends on the initial status of the method. Considering that the variables in the problem are essentially discrete, Samani and Mottaghi [4] proposed an integer LP method which could change the solution iteratively according to the hydraulic simulation. But the optimality and convergence of the method are questioned, particularly on large-scale networks [12]. Recently, methods in the second category, i.e., metaheuristic algorithms, have attracted a lot of attention in this domain as they are able to handle different kinds of constraints easily, and locate near-optimal solutions effectively. Both single-solution

algorithms [13]–[15], such as simulated annealing (SA) [13], tabu search (TS) [3], iterated local search [14], cellular automata [15], and population-based algorithms [16]–[22], such as genetic algorithm (GA) [16], [17], ant colony optimization (ACO) [18], particle swarm optimization (PSO) [19], and differential evolution (DE) [20]–[22], are widely investigated. These metaheuristic methods have shown competitive performance on small-scale networks which contain less than 200 pipes.

However, due to the acceleration of the process of urbanization, cities in developing countries become larger and larger, which also enlarges WDNs greatly. Taking Xiongan New Area in China as an example, only the starting area of its first developing stage is planned to be 100 square kilometers, and a large number of buildings will be constructed [23]. To build a municipal water supply network for the area, hundreds even thousands of pipes must be included. However, studies on large-scale optimization [24], [25] have shown that traditional metaheuristic algorithms like PSO, DE, are not capable to solve problems with that large scale. Some researchers have noticed this problem and come up with several decomposition methods which can partition a big network into small pieces, but the partitioned subnetworks are often optimized separately [21]. Thus, they can only get some partially best solutions. In order to meet the realistic demand of large-scale WDN optimization and to make up the aforementioned deficiencies of existing works, in this paper we intend to propose a cooperative co-evolutionary algorithm (CCEA) named WDNCC to solve the large-scale WDN optimization problem.

Rooted in the divide-and-conquer strategy, cooperative co-evolutionary (CC) methods solve large-scale optimization problems by decomposing them into a number of small-scale subcomponents. After decomposition, the subcomponents are handled by an equal number of cooperative optimizers [26]. A large-scale WDN can be seen as a semi-separable structure which highly fits to CC methods. Because we usually divide a large-scale WDN into different district metering areas (DMAs) by valves during WDN management [27]. Meanwhile, to guarantee the quantity and quality of water supplement, generally there will be more than one water source in a large-scale WDN, no matter which type the sources are, reservoirs, tanks, or groundwater sources [21]. The division of DMAs and the independent sources provide the realistic basis for the partition of CC. Furthermore, to guarantee the reliability and safety of the whole network, subnetworks in different DMAs will not be completely isolated, which means they are still connected according to some hydraulic rules. Thus, they cannot be optimized separately. In this regard, the co-evolutionary mechanism of CC can ensure that the subcomponents are optimized in an interactional way.

Hitherto, many scholars have devoted themselves to the research of CC methodology. Both multipopulation paradigm and mono-population paradigm are well studied [28]–[33]. However, most of the works which specially study the decomposition methods focus on continuous functions [28]–[30]. Since these functions are usually unconstrained, the decomposition methods proposed for them cannot be directly used in real-world applications. Thus, for real-world applications, scholars generally need to devise some other appropriate decomposition methods [32]–[36]. To apply CC to the large-scale WDN optimization problem, an effective decomposition method is also required. With the assistance of EPANET which is a well-known WDN simulation tool [37], we propose an iterative trace-based decomposition method to divide a large-scale WDN into subnetworks in this paper. This is a key innovation of WDNCC. Meanwhile, there are many constraints in the problem, and most of them can be satisfied by utilizing EPANET except one, the minimum pressure constraint. To handle the minimum pressure constraint, the objective function of WDNCC is specially designed with a penalty function. The applied evolutionary algorithm (EA) in WDNCC is the self-adaptive DE with neighborhood search (SaNSDE) [38].

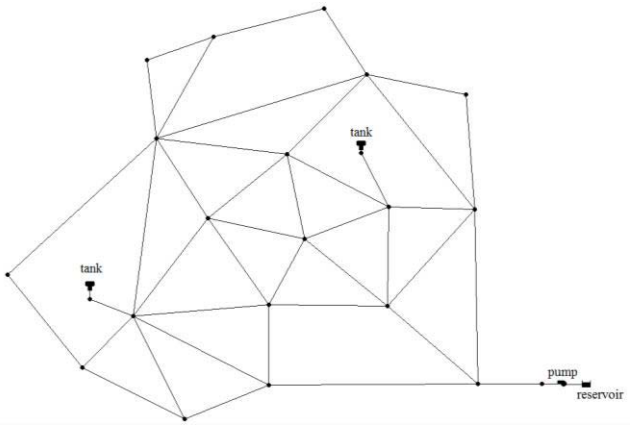


Fig. 1. Anytown network.

To show the effectiveness of WDNCC, a series of benchmark networks are generated with different scales. Experiments on the benchmark networks and an established network show that WDNCC is promising in tackling large-scale WDN optimization problems.

The rest of this paper is organized as follows. In Section II, the WDN optimization problem is formally described. Then, Section III introduces some previous researches about the WDN optimization. The WDNCC algorithm is shown in Section IV. Experiments are conducted in Section V. Finally, Section VI draws the conclusion.

II. Water Distribution Network

In the first place, a small-scale instance, Anytown network [39], is shown in Fig. 1 to facilitate the explanation of WDN. The Anytown network has one reservoir and one pump station serving as the main water source. Two tanks are set in the city for auxiliary supplement. Each junction represents a consumer. Links between junctions are pipes which are the variables of the problem. Assuming that the demands of all consumers are known in advance, the goal of the WDN optimization is to reduce the constructing expenditure by selecting an appropriate type for each pipe, while satisfying all consumers' demands and the hydraulic constraints.

All related notations which will be used are listed in Table I. Formally, suppose there are Nn nodes including all sources and

TABLE I
NOTATIONS ABOUT THE WDN OPTIMIZATION PROBLEM

notation	meaning
i, j, k	index or counter
Nn	the number of nodes
Np	the number of pipes
Nt	the number of pipe types
l_i	the length of the ith pipe
$\mathbb{C}$	the problem which contains Np pipes
θ_i	the ith pipe
$\{\zeta_1,\ldots\zeta_{Nt}\}$	Nt commercial pipe types
$u(\theta_i)$	unit price of the ith pipe
Q_i^{ext}	the velocity of external inflow water of the ith node
$Q_{j,i}^{in}$	the velocity of incoming water from node j to i
Q_i^n	the water consumption velocity of the ith node
$Q_{i,k}^{out}$	the velocity of outgoing water from node i to k
$\mathbf{P}$	a path consisting of a series of successive pipes
$\mathbf{SP}$	complete set of $\mathbf{P}$
H^s	the head pressure of the start node of $\mathbf{P}$
H^e	the head pressure of the end node of $\mathbf{P}$
H_i^s	the head pressure of the start node of the ith pipe
H_i^e	the head pressure of the end node of the ith pipe
ΔH_i	the head loss of the ith pipe
H_i	the actual head pressure of the ith node
$H_{i,\min}$	the minimum pressure constraint
$H_{i,\max}$	the maximum pressure constraint
Q_i	the water flow rate of the ith pipe
D_i	the diameter of the ith pipe
C_i	the Hazen-Williams roughness coefficient of the ith pipe
α, β, γ	the parameters of the Hazen-Williams formula
T	the length of time
$\boldsymbol{\theta}$	a solution of the problem
$\boldsymbol{\theta}_\mathbf{f}$	a feasible solution
$\boldsymbol{\theta}_\mathbf{max}$	a solution consists of pipes with largest diameter

junctions, Np pipes, and Nt different types of pipes available. The problem is defined as follows [1], [4]:

$$\min\ f(\mathbb{C}) = \sum_{i=1}^{Np} l_i \cdot u(\theta_i) \tag{1}$$

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

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

$$\begin{cases} \sum_{i\in\mathbf{P}} \Delta H_i = H^s - H^e, & \mathbf{P} \in \mathbf{SP} \\ \Delta H_i = H_i^s - H_i^e, & i = 1, \ldots, Np \end{cases} \tag{4}$$

$$H_{i,\min} \le H_i \le H_{i,\max}, i = 1, \ldots, Nn \tag{5}$$

where $\mathbb{C} = \{\theta_1, \ldots, \theta_{Np}\}$ represents the problem, l_i represents the length of the ith pipe, $u(\theta_i)$ represents the unit price of the ith pipe with type θ_i, $\{\zeta_1, \ldots, \zeta_{Nt}\}$ are the Nt commercially available pipe types, Q_i^{ext} is the velocity of external inflow water of the ith node, $Q_{j,i}^{\text{in}}$ is the velocity of incoming water from node j to i, Q_i^n denotes the water consumption velocity of the ith node, $Q_{i,k}^{\text{out}}$ denotes the velocity of outgoing water from node i to k, ΔH_i is the head loss in the ith pipe, $\mathbf{P}$ is one path in the network consisting of a series of successive pipes, $\mathbf{SP}$ is a complete set of $\mathbf{P}$, H^s and H^e are the pressure head at the start node and the end node of $\mathbf{P}$, respectively, [actually which point is taken as the start or the end will not affect the validity of (4)], H_i^s and H_i^e are the pressure head at the start node and the end node of the ith pipe, H_i is the actual pressure head provided by the network, $H_{i,\min}$ and $H_{i,\max}$ are the minimum and maximum pressure constraints, respectively.

The mass conservation law (3) implies that, for a specific node, the velocity of incoming water should be equal to its consumption velocity plus the outgoing velocity. If there is no external incoming water (e.g., purified rainwater), Q^{ext} is equal to 0. Also, if the jth node or the kth node is not directly connected to the ith node, $Q_{j,i}^{\text{in}}$ or $Q_{i,k}^{\text{out}}$ is equal to 0. The energy conservation law (4) indicates that the head losses accumulated along a path should be equal to the difference between the head of the start node and the head of the end node. The head loss of each pipe can be roughly calculated by the Hazen–Williams formula

$$\Delta H_i = \alpha \frac{(Q_i)^\beta l_i}{(C_i)^\beta (D_i)^\gamma}, i = 1, \ldots, Np \tag{6}$$

where Q_i, D_i, and C_i are the water flow rate, diameter, and Hazen–Williams roughness coefficient of the ith pipe, respectively. α, β, and γ are three empirical parameters which are commonly set as 10.667, 1.852, and 4.871. Regarding (5), usually the minimum head requirement $H_{i,\min}$ is defined by the government (e.g., in China, it is set to 28 m according to the national standard file GB50282-98) and the maximum head requirement $H_{i,\max}$ is usually not defined except for some special occasions.

Furthermore, in this paper, both the multiple-loading paradigm with different demand patterns, also known as the extended-period paradigm, and the single-loading paradigm, also known as the steady-state paradigm, are considered. In the former case, the water demand of a consumer may change over time and different consumers have different patterns [40]. Thus, we actually simulate the behavior of a water network in an extended period. Cutting the period into T pieces, the ith consumer's demand can be defined by a sequence of water consumption velocities

$$Q_i^n = \left\{Q_{i,1}^n, Q_{i,2}^n, \ldots, Q_{i,T}^n\right\}. \tag{7}$$

In the single-loading paradigm, each consumer requires a fixed amount of water.

III. RELATED WORK

In this section, some metaheuristic algorithms and the way in which they handle the constraints are introduced.

Da Conceição Cunha and Sousa [13] used the SA algorithm to solve the WDN design problem. The hydraulic network equations [constraints (3) and (4)] are solved by a Newton search method during the execution of the algorithm. However, there is not an efficient way to handle the constraint of user requirements (5). If the current solution is infeasible, the algorithm would replace the solution by randomly selecting one of its neighbors. Thus, the search is nondirectional if the initial solution is infeasible.

Afterward, Cunha and Ribeiro [3] proposed a TS method which incorporated a simulator to handle the hydraulic constraints. To satisfy the consumers' requirements, they always initialize the solution by setting all pipes to the type of

the largest diameter. This initialization method may greatly degrade the search ability of the algorithm. Consequently, the algorithm is easy to be trapped into the same local optima.

In the past decade, population-based metaheuristics received more attention in this domain. GAs are one of the most popular algorithms that have been investigated [17]. Simpson *et al.* [41] first applied a canonical GA method to optimize a very small network. A penalty function was designed to punish infeasible solutions which could not satisfy the minimum head constraint. If the constraint was violated, an extra value which was equal to the maximum pressure deficit multiplied by a penalty factor would be added to the objective value. The latest related work which used a GA was proposed by Bi *et al.* [16]. They focused on proposing a new initialization method by using heuristic domain knowledge. Which GA variant was applied and how they treated the constraints were not specified. But in order to make a fair comparison, they still used Simpson's GA method.

Suribabu and Neelakantan [42] combined a PSO algorithm with EPANET, developing a tool called PSONET. Infeasible solutions in PSONET are also punished by adding a big number which is larger than the largest cost of the network to their objective values. Montalvo *et al.* [19] also utilized PSO in their work. The only difference between their algorithm and PSONET is the penalty function, which is designed as the sum of all pressure deficits. Recently, another PSO variant called developed swarm optimizer (DSO) was proposed, which had shown competitive performance on small-scale networks [43].

After PSO, Suribabu [44] later tried a DE algorithm to solve the WDN optimization problem which also achieved good results. The way in which they handled the pressure constraint was inherited from [42]. Zheng *et al.* [22] proposed a self-adaptive DE (SADE) algorithm which could adaptively change the parameters of the algorithm. Meanwhile, they used the constraint tournament selection strategy which was proposed by Deb [45] to compare solutions. This strategy contains three rules: 1) infeasible solutions are always worse than feasible solutions; 2) between two feasible solutions, the one who gets better objective value is preferred; and 3) between two infeasible solutions, the one with lower degree of constraint violation is better.

Soon after, Zheng *et al.* [46] came up with another approach based on ACO, called adaptive convergence-trajectory controlled ant colony optimization (ACO_{CTC}). Since both feasible and infeasible solutions are used to update pheromone values, a penalty function method is used instead of the constraint tournament selection. The penalty is defined as the maximum pressure deficit. Zecchin *et al.* [18] compared five different ACO variants on WDN problems and found that the elitist-rank ant system (AS_{rank}) and the max–min ant system were better than the others.

A special case is the approach proposed by Zheng *et al.* [21], in which a two-stage DE (TSDE) method is used to optimize multisource WDNs. In the method, a WDN is partitioned into several subnetworks in advance. Then, during the first stage of optimization, the subnetworks are optimized separately. Afterward, the whole network is handled according to the result obtained in the first stage. The defect of the proposed decomposition method is that it only fits some simple single-loading networks, where pumps or valves do not exist. Meanwhile, since the subnetworks are optimized separately during the first stage, the TSDE may perform poorly on closely linked networks.

Besides the aforementioned algorithms, some other metaheuristic algorithms have been also investigated [47]–[50]. Observing the methods used to handle the constraints, we can find that generally the hydraulic constraints are solved by simulation tools, and the penalty function method is applied to handle the minimum pressure constraint. However, the penalty functions in most algorithms are imperfect.

1) Some of them set a fixed big number as penalty. In such a case, the violation degree of the constraints is ignored. Consequently, the infeasible solutions cannot be compared with each other.
2) Some of them set the penalty as the maximum pressure deficit of a node or the sum of pressure deficits of all nodes. In this case, infeasible solutions with small degree of violation may be considered better than some feasible solutions. If the minimum pressure constraint is viewed as a hard constraint, this situation is not acceptable. Aiming at these defects, we intend to propose a novel fitness function in WDNCC.

IV. Cooperative Co-Evolution for Water Distribution Network Optimization

The first CCEA proposed by Potter and De Jong [51] was originally applied to optimize low-dimensional functions. Due to the extraordinary scalability of CC, many people have promoted its usage on large-scale problems [34], [52]. This is one of the reasons why we adopt CC in this paper. In this section, WDNCC is described in detail. The trace-based decomposition is shown in the first place. Then, the utilized EA, i.e., SaNSDE, and the fitness function are introduced, respectively. Finally, the WDNCC framework is shown by integrating these components together.

A. Trace-Based Decomposition

Good decomposition is a prerequisite for the success of a CCEA. The principle of decomposition is to put interactional variables into the same group and divide independent variables into different groups [28]. Although there are already some decomposition methods proposed for large-scale continuous functions like differential grouping [28], [29], usually these decomposition methods cannot be directly applied to WDN optimization problems due to two reasons.

1) These methods need massive times of simulations to calculate the relationships between every pair of variables. This process is extremely inefficient.
2) These methods rely on plenty of hypothetic solutions which may be infeasible for WDN optimization problem, and simulations on these infeasible solutions are meaningless. For WDN optimization problems, with the help of relevant domain knowledge, we can make an accurate enough decomposition by conducting a small

Algorithm 1 Trace-Based Decomposition

Input: one feasible solution $\boldsymbol{\theta}_f$, node demand $Q^n = \{Q_i^n | 1 \le i \le Nn\}$, number of water source No.
Output: pipe partition $\mathbf{PP} = \{pp_i | 1 \le i \le Np\}$.

1 Set the values of the pipes according to $\boldsymbol{\theta}_f$;
2 $flow_{1\to Np} = \{0\}$; //flow velocity of each pipe
3 **for** $i = 1$ to T **do**
4 run the hydraulic simulation of time slice i;
5 **for** $j = 1$ to Np **do**
6 get the flow velocity of the jth pipe fv_j;
7 $flow_j = flow_j + fv_j$;
8 **end for**
9 **end for**
10 $quantity_{1\to Np,1\to No} = \{\{0\}\}$; //quantity of water from every source to every node
11 **for** $i = 1$ to No **do**
12 **for** $j = 1$ to T **do**
13 run the trace simulation of source i in time slice j;
14 **for** $k = 1$ to Nn **do**
15 get the percentage of water quantity $pwq_{k,i}$ from source i to node k;
16 $quantity_{k,i} = quantity_{k,i} + pwq_{k,i} \cdot Q_{k,j}^n$;
17 **end for**
18 **end for**
19 **end for**
20 **for** $i = 1$ to Nn **do**
21 $nodeBelong_i = \arg\max_{1\le j\le No}(quantity_{i,j})$;
22 **end for**
23 **for** $i = 1$ to Np **do**
24 get the two node s and e linked by the ith pipe;
25 **if** $flow_i > 0$ **then**
26 $pp_i = nodeBelong_e$;
27 **else**
28 $pp_i = nodeBelong_s$;
29 **end if-else**
30 **end for**
31 return **PP**

number of simulations or even without conducting any simulation.

To divide a large-scale WDN with multiple sources into subnetworks, a natural thought is putting *congeneric* nodes whose water comes from the same source into the same group, and dividing nodes which belong to different sources into different groups. Some methods tried to use the value of friction slope to specify the ownership between consumer nodes and sources without conducting any simulation [21]. However, since different sources have different water supply capacities and pumps would also affect the water supplement, the minimum friction slope method sometimes may be not accurate. Thus, in this paper, we propose an iterative trace-based decomposition method which uses a few times of simulations based on EPANET to check the real trace of water, finding the real water source for each node.

It should be noted that the variable in WDN optimization is pipe rather than node. However, before dividing pipes, nodes should be divided at first, since EPANET only provides the source tracing function for nodes. Based on the results of node partition, we can divide the pipes into different groups. Before showing the trace-based decomposition method, three principles are stated at first.

1) If the water consumed by a node comes from only one source, the node definitely belongs to that source.

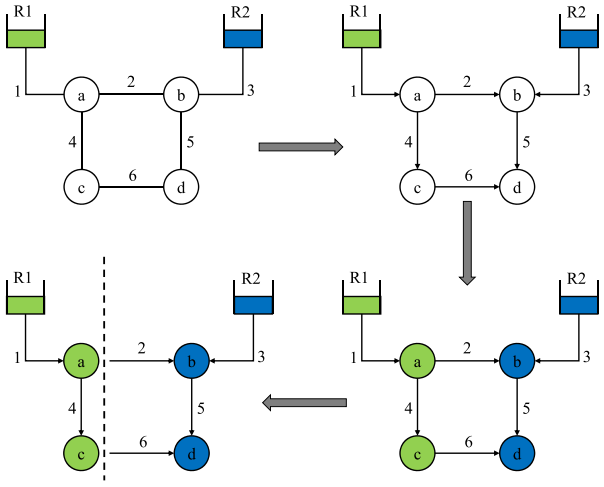


Fig. 2. Trace-based decomposition.

2) If the water consumed by a node comes from multiple sources, it belongs to the source which provides more water to it.
3) Pipes should take charge of the nodes that they can affect. It means that a pipe belongs to the source which its downstream node belongs to.

Based on these three rules, the pseudo code of the trace-based decomposition method is shown in Algorithm 1.

To begin with, all pipes are initialized according to a given feasible solution (line 1). Then a series of hydraulic simulations are conducted to judge the flow direction of every pipe and to make preparation for the following trace simulations (lines 2–9). Denoting the start node and the end node of the ith pipe as s and e, respectively, if the water flows from s to e, $fv_i > 0$; otherwise $fv_i < 0$. Thus, the variable $\text{flow}_{1\to Np}$ (line 2) actually records a general flow direction of each pipe by accumulating the velocity in each time slice. Afterward, the quantity of water that each source has provided to each consumer is calculated by conducting a series of trace simulations (lines 10–19). Since the trace simulation only provides the proportion value, the total demand of each time slice needs to be multiplied to get the real quantity (lines 15 and 16). Based on the quantity data and the former two principles, all nodes are divided into different groups (lines 20–22). Finally, the pipe partition is executed. All pipes are classified in line with their downstream nodes (lines 23–30).

An example is shown in Fig. 2 to illustrate the trace-based decomposition method. There are two reservoirs, $R1$ and $R2$, and four consumers $\{a, b, c, d\}$, linked by six pipes which are ordered from 1 to 6. First, the hydraulic simulation is made. We can find that the flow direction of pipe 2 is from a to b, and the flow direction of the pipe 6 is from c to d. It means that only the water from $R2$ cannot satisfy the requirements of b and d. Some water flows from $R1$ to b and d to support them. Nevertheless, the trace simulation shows that most of the water consumed by b and d still comes from $R2$. Thus, b and d are grouped with $R2$, meanwhile a and c are grouped with $R1$. According to the result of node partition, pipe 1 and 4 are

partitioned into a group, pipe 2, 3, 5, and 6 are partitioned into the other group.

B. SaNSDE

The optimizer used in WDNCC is SaNSDE, which is a variant of DE. Given a population which contains M individuals $\{x_1, x, \ldots, x_M\}$, and an Np-dimensional problem, SaNSDE can be summarized by the following three steps.

1) *Mutation:* For each solution, the mutant vector is generated by

$$v_i = \begin{cases} x_a + F_i \cdot (x_b - x_c), & \text{if } r_1 < p \\ x_i + F_i \cdot (x_{\text{best}} - x_i) + F_i \cdot (x_a - x_b), & \text{otherwise} \end{cases} \tag{8}$$

where $a, b, c \in [1, M]$ are random and mutually different integers. Also they are different from i. r_1 is a uniform random number generated within (0, 1). p is a self-adaptive parameter. It is initially set to 0.5. After evaluating all offspring, the number of offspring which is generated by the former mutation strategy of (8) and successfully reserved is recorded as ns_1. The success number corresponding to the latter mutation strategy is recorded as ns_2. The numbers of failed offspring of the two mutation strategies are recorded as nf_1 and nf_2, respectively. During every 50 generations, these four values accumulate. After that, p is recalculated as

$$p = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_2 \cdot (ns_1 + nf_1) + ns_1 \cdot (ns_2 + nf_2)}. \tag{9}$$

Once p is updated, ns_1, ns_2, nf_1, and nf_2 will be reset to 0. F_i in (8) is a real factor calculated by

$$F_i = \begin{cases} N(0.5, 0.3), & \text{if } r_2 \le p \\ C(0, 1), & \text{otherwise} \end{cases} \tag{10}$$

where $N(0.5, 0.3)$ is a normal distribution, and $C(0, 1)$ is a Cauchy distribution. r_2 is another uniform random number within (0, 1).

2) *Crossover:* The value of each dimension of the newly-generated solution is determined by

$$u_i(j) = \begin{cases} v_i(j), & \text{if } r_3 \le \text{CR}_i \text{ or } j = j_{\text{rand}} \\ x_i(j), & \text{otherwise} \end{cases} \tag{11}$$

where $j \in [1, Np]$.r_3 is a uniform random number within (0, 1). j_{rand} is randomly chosen to ensure that u_i does not replicate x_i. CR_i is calculated by

$$\text{CR}_i = N_i(CRm, 0.1). \tag{12}$$

CRm is initially set to 0.5. During each generation, the CR values associated with the individuals which successfully enter the next generation are recorded in an array *CRrec*. After every 25 generations, *CRm* is recalculated by

$$CRm = \sum_{k=1}^{|CRrec|} w_k \cdot CRrec(k) \tag{13}$$

$$w_k = \Delta f_{\text{rec}}(k) \Big/ \left(\sum_{k=1}^{|\Delta f_{\text{rec}}|} \Delta f_{\text{rec}}(k) \right) \tag{14}$$

where Δf_{rec} is the improvement on fitness value.

Algorithm 2 WDNCC Framework

Input: problem $\mathbf{C} = \{\theta_1, \ldots\ldots, \theta_{Np}\}$
Output: solution *gbest*

```
decompose the problem into Ns groups using θ_max;
for i = 1 to Ns do
    initialize the ith populations S_i;
end for
rc = 0; rstart = false;
for i = 1 to MAXROUND do
    if rc % RI == 0 && gbest is feasible
        re-decomposition using gbest;
        re-assemble solutions;
        rstart = true;
    end if
    if rstart == true
        rc++
    end if
    for j = 1 to Ns do
        evolve the jth population S_j;
        calculate fitness values using gbest;
        update best_j of S_j;
        update the j-th part of gbest;
    end for
end for
return the gbest;
```

3) *Selection:* Finally, the offspring is generated according to

$$x_i^* = \begin{cases} u_i, & \text{if } f(u_i) \le f(x_i) \\ x_i, & \text{otherwise} \end{cases} \tag{15}$$

x_i^* is the offspring of x_i (assume the problem is minimization problem).

Although the variables of WDN optimization problems are discrete, they are essentially ordinal rather than categorical. Thus, to apply SaNSDE which is a continuous optimizer to the WDN optimization problem, we just need to turn the pipe type into numbers. In this paper, the pipe types are ordered from 1 to Nt according to their diameters (both ascending order and descending order are feasible). With respect to the continuous values in the algorithm, they are operated normally in continuous way during the execution of the algorithm. When conducting a simulation, they will be directly rounded down to the nearest integers to represent the pipe types. Although the EA applied in WDNCC is SaNSDE, without loss of generality, any EA which has been proved effective on small-scale WDN optimization problems can be employed as the optimizer.

C. Fitness Design

As mentioned earlier, the WDN optimization problem is essentially a constrained problem. Among the three constraints, (3)–(5), the former two can be handled by the simulation tool EPANET. The last one, i.e., the minimum pressure constraint, should be handled by the optimization algorithm. Thus, in this paper, "infeasible solution" is used to describe a solution which cannot satisfy the minimum pressure constraint. To handle the minimum pressure constraint, we use the penalty function method in WDNCC.

For a constrained problem, since infeasible solutions are always worse than feasible solutions, we can define the penalty

as a primary objective in order to find feasible solutions. Then the original objective, i.e., the expenditure, can be taken as the secondary objective which only works when comparing two feasible solutions. Inspired by the work [53] which studied the vehicle routing problem with also two different objectives, the fitness $F(\boldsymbol{\theta})$ of a solution $\boldsymbol{\theta}$ in WDNCC is defined as follows:

$$F(\boldsymbol{\theta}) = f(\boldsymbol{\theta})/f(\boldsymbol{\theta}_{\max}) + P(\boldsymbol{\theta}) \tag{16}$$

where

$$\boldsymbol{\theta}_{\max} = \{\theta_i = Nt | 1 \le i \le Np\}, \tag{17}$$

$$\begin{cases} P(\boldsymbol{\theta}) = \sum_{i=1}^{T} \sum_{j=1}^{Nn} \varphi(i,j) + \varphi(i,j) \cdot \left(H_{j,\min} - H_{j,i}\right) \\ \varphi(i,j) = \begin{cases} 1, & \text{if } H_{j,\min} > H_{j,i} \\ 0, & \text{otherwise.} \end{cases} \end{cases} \tag{18}$$

As we can see from (16), a solution's fitness consists of two parts, the objective part $f(\boldsymbol{\theta})/f(\boldsymbol{\theta}_{\max})$ and the penalty part $P(\boldsymbol{\theta})$. If a solution is feasible, the penalty part will be equal to 0. Since the objective part is always divided by the objective value of $\boldsymbol{\theta}_{\max}$ whose pipes are all set to the most expensive type, the objective part will be always less than or equal to 1. (The most expensive type usually has the biggest diameter.) Adding these two parts together, the fitness value of a feasible solution will be always less than or equal to 1. If a solution is infeasible, there will be at least a node whose real head is lower than the threshold value, i.e., $H_{j,\min} > H_{j,i}$. According to (18), the penalty part of the solution's fitness will be larger than 1. Still the objective part is less than 1. (Here, the objective parts of infeasible solutions will never be equal to 1, because the solution $\boldsymbol{\theta}_{\max}$ must be feasible, otherwise feasible solutions do not exist.) Adding these two parts together, for an infeasible solution, its fitness value will be always larger than 1. The single-loading paradigm can be seen as a special case of the multiple-loading paradigm, where T in (18) is equal to 1.

D. Architecture

The architecture of WDNCC is shown in Algorithm 2 in the form of pseudo code. For a WDN optimization problem which contains Np pipes $\mathbb{C} = \{\theta_1, \ldots\ldots, \theta_{Np}\}$, WDNCC first divides it into Ns subcomponents (line 1) according to the solution $\boldsymbol{\theta}_{\max}$, represented as

$$\mathbb{C} = \mathbb{C}_1 \bigcup \mathbb{C}_2 \bigcup \ldots \bigcup \mathbb{C}_{Ns} \tag{19}$$

$$\text{s.t. } \forall i \in [1, Ns], \mathbb{C}_i \neq \emptyset \tag{20}$$

$$\forall i, j \in [1, Ns] \wedge i \neq j, \mathbb{C}_i \bigcap \mathbb{C}_j = \emptyset. \tag{21}$$

The trace-based decomposition method used in WDNCC always needs a feasible solution. The first decomposition is made based on $\boldsymbol{\theta}_{\max}$, since it is the only feasible solution guaranteed before optimization. From (19) to (21), we can see that Np pipes are divided into Ns groups. After the first decomposition, Ns populations $\{S_1, S_2, \ldots, S_{Ns}\}$ are initialized corresponding to Ns groups, and Ns optimizers will be initialized too (lines 2–4). Each population S_j maintains a best subsolution it has ever found, denoted as best_j. The whole

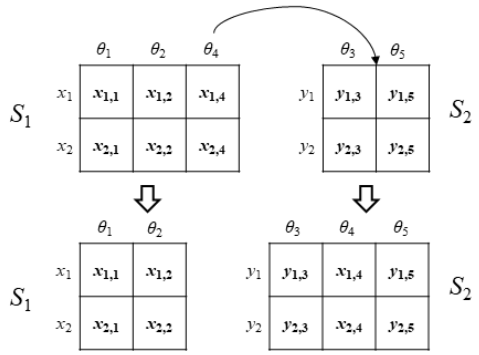


Fig. 3. Reassemble solutions. The whole problem $\mathbb{C}$ consists of five variables $\{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$. It is divided into two subcomponents $\mathbb{C}_1$ and $\mathbb{C}_2$. Before redecomposition, $\mathbb{C}_1 = \{\theta_1, \theta_2, \theta_4\}$, $\mathbb{C}_2 = \{\theta_3, \theta_5\}$. After redecomposition, $\mathbb{C}_1 = \{\theta_1, \theta_2\}$, $\mathbb{C}_2 = \{\theta_3, \theta_4, \theta_5\}$. Solution reassembling is realized by extracting the values of θ_4 from S_1, and assembling them with solutions of S_2.

algorithm maintains a global best solution, denoted as $g\text{best}$

$$g\text{best} = \text{best}_1 \bigcup \text{best}_2 \bigcup \ldots \bigcup \text{best}_{Ns}. \tag{22}$$

Here, the union symbol is used to represent the concatenation among subsolutions. When the algorithm finds a feasible solution for the first time, the redecomposition procedure will be activated. For every RI iterations, WDNCC will redecompose the problem based on $g\text{best}$ (lines 7–14). Thus, the decomposition in WDNCC is actually an iterative process. After redecomposition, all subsolutions in all populations should be reassembled. To keep the algorithm simple, we directly split and assemble these subsolutions according to their indices. An example is shown in Fig. 3. Initially, the problem is divided into two subcomponents, $\mathbb{C}_1 = \{\theta_1, \theta_2, \theta_4\}$, $\mathbb{C}_2 = \{\theta_3, \theta_5\}$. Correspondingly there are two populations, and both have two individuals, $S_1 = \{x_1, x_2\}$, $S_2 = \{y_1, y_2\}$. After redecomposition, the classification of θ_4 changes. Then, we directly extract the values $x_{1,4}$ and $x_{2,4}$ from S_1, and assemble them with y_1 and y_2 in S_2.

In each iteration, the populations evolve successively and update $g\text{best}$ one by one (lines 15–20). Specifically, in the ith iteration, the jth optimizer first operates on its population S_j. The fitness value of each individual x in S_j is evaluated

$$g(x) = F\left(x \bigcup \overline{\text{best}_j^{i-1}}\right) \tag{23}$$

where

$$\begin{aligned} \overline{\text{best}_j^{i-1}} &= g\text{best} - \text{best}_j^{i-1} \\ &= \text{best}_1^i \bigcup \ldots \bigcup \text{best}_{j-1}^i \bigcup \text{best}_{j+1}^{i-1} \bigcup \ldots \bigcup \text{best}_{Ns}^{i-1}. \end{aligned} \tag{24}$$

Since x is a subsolution which only has the values of the pipes belonging to $\mathbb{C}_j$, its fitness value has to be evaluated by combining it with the pipes in other subcomponents. The values of other pipes come from $g\text{best}$. Then best_j^i is updated as

$$\text{best}_j^i = \arg\min_{x \in S_j} g(x). \tag{25}$$

Afterward, $g$best is updated according to best_j^i

$$g\text{best} = \text{best}_1^i \bigcup \ldots \bigcup \text{best}_j^i \bigcup \text{best}_{j+1}^{i-1} \ldots \bigcup \text{best}_{Ns}^{i-1}. \tag{26}$$

When the jth population finishes these procedures, the $j+1$th optimizer starts working. Finally, after MAXROUND iterations of optimization, the whole algorithm stops and returns the best solution $g$best.

V. Experiments

In this section, the proposed WDNCC is tested on several networks with different scales. Both multiple-loading and single-loading schemes are considered. The scale of a network is defined by the number of pipes. First, a series of test cases are generated and introduced in detail. Second, the redecomposition interval RI is investigated for the single-loading cases. Third, we compare WDNCC with some other methods on the single-loading and multiple-loading cases to check its performance. Finally, the partitions on multiple-loading cases are also shown to discuss the rationality of the redecomposition strategy. Furthermore, to show that the advantage of WDNCC does not merely come from SaNSDE, a comparison between WDNCC and the pure SaNSDE is made. In addition, the efficiency of WDNCC is also demonstrated by checking the execution time. However, due to the page limit, the comparison between WDNCC and SaNSDE, and the experiment of execution time are shown in the supplemental material.

To demonstrate the effectiveness of WDNCC, several representative methods which were proposed to solve WDN optimization problems are compared. The first one is the PSO algorithm [19], which has shown similar performance with some GA methods and ACO methods. The second one is the SADE algorithm [22], which is effective on large-scale networks. The third one is the DSO algorithm [43], which was recently proposed and showed competitive performance to the SADE algorithm on small-scale cases. The final one is a two-stage algorithm which was specifically designed for networks with multiple sources [21], and it also used the divide-and-conquer strategy. Since the standard DE was applied in the algorithm, we call it TSDE in the following experiments.

According to the study in [19] and [22], setting the population size of the algorithm approximately equal to the scale of the problem is appropriate. In the applied test cases, the number of pipes is about 100 times of the number of sources. Thus, if we set aside the network structure and the water supplying capacity of each source, after decomposition, each subcomponent contains roughly 100 pipes. Hence, the population size of WDNCC is set to 100 for each subcomponent of the network. For the former three compared algorithms, PSO, DSO, and SADE, the population size is set equal to the node number of the network. As to TSDE, the authors customized the population size for each network even each subnetwork in their work. However, they did not give instructions about how to decide the population size. Thus, for TSDE, according to the study in [19] and [22] and the population size set in [21], we give a simple strategy that the population size of the first stage is set equal to the number of the nodes of the subnetworks, while the population size of the second stage is set to the half of the node number of the whole network due to the limit of the number of fitness evaluations (FEs). Other parameters in the algorithms are directly inherited from their original settings, which are shown in Table II.

TABLE II
Parameter Settings of the Compared Algorithms

algorithm	notation	value
PSO	c_1	3
	c_2	2
	w	(1+1/(ln$k$+1))/2 where k is the iteration number.
DSO	c_1	[0.1L, 0.5L] where L is value range of variables
	c_2	0.5
SADE	Fc	[0.1, 0.9]
	CR	[0.1, 0.9]
TSDE	Fc	0.3
	CR	0.5

A. Benchmark Description

Although there are already several real WDNs which have been widely used as benchmarks in previous studies, most of these test cases are not generated following a unified standard. Moreover, lots of the well-known networks were proposed decades ago, whose scales were too small to simulate current real cases. Also, as mentioned in [1], it would be beneficial to design benchmarks factoring in different levels of complexities and scales for pure algorithmic developers. Thus, to conduct a systematic experiment, a series of artificial WDNs following the same construction method are generated. Since our main purpose is to test the proposed approach rather than building a real network of any real city, only the very necessary components, such as pipes, pumps, reservoirs, and consumers are included in the artificial networks.

To create a WDN, first, a certain number of nodes are generated within a circle region which represent consumers. Then we connect these nodes by nonintersecting pipes which are the variables of the problem. Usually, besides the necessary pipes which link all nodes to ensure that the network is fully connected, there will be some auxiliary pipes to improve the reliability of the network. Thus, in the artificial networks, the number of pipes is roughly 1.1 times of the node number. Reservoirs are generated on the edge of the circle region, linked with the nearest nodes by pumps. Enough pumps are provided so that the solution $\boldsymbol{\theta}_{\max}$ is always feasible. Through adjusting the number of pumps connected to each reservoir, we can endow different supplying capacities to different reservoirs. Totally 20 instances are generated. Half of them are balanced networks, in which the reservoirs are endowed with almost the same supplying capacity. The other half are imbalanced, in which reservoirs are endowed with different supplying capacities. For multiple-loading cases, totally five demand patterns are considered. Each pattern contains six time slices, and each consumer is assigned to a pattern randomly. For single-loading cases, the patterns are removed. The details about the generated benchmark are shown in Table III. The structures of the smallest size 200, the middle size 400, and the biggest size 600 are shown in Fig. 4. The other networks

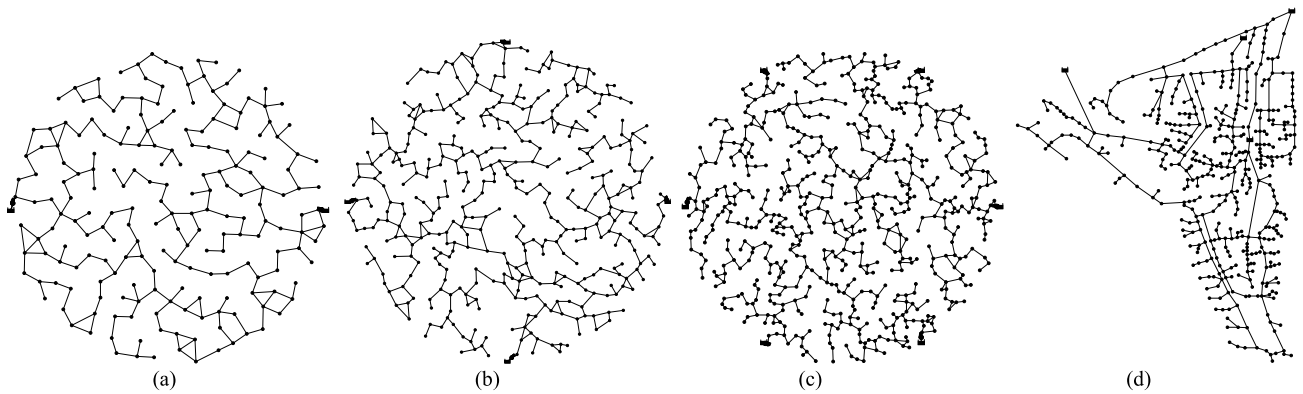


Fig. 4. WDN structures. (a) 200. (b) 400. (c) 600. (d) Balerma.

TABLE III
INFORMATION ABOUT THE ARTIFICIAL WDNS

name	nodes	pipes	reservoirs	paradigm
200-B-S	200	226	2	single
200-B-M	200	226	2	multiple
200-I-S	200	226	2	single
200-I-M	200	226	2	multiple
300-B-S	300	328	3	single
300-B-M	300	328	3	multiple
300-I-S	300	328	3	single
300-I-M	300	328	3	multiple
400-B-S	400	452	4	single
400-B-M	400	452	4	multiple
400-I-S	400	452	4	single
400-I-M	400	452	4	multiple
500-B-S	500	546	5	single
500-B-M	500	546	5	multiple
500-I-S	500	546	5	single
500-I-M	500	546	5	multiple
600-B-S	600	661	6	single
600-B-M	600	661	6	multiple
600-I-S	600	661	6	single
600-I-M	600	661	6	multiple

'B' represents 'balanced' which means reservoirs have balanced supplying capacity. 'I' represents 'imbalanced'. 'S' represents single loading. 'M' represents multiple loadings.

TABLE IV
INFORMATION ABOUT THE PIPE TYPES

diameter(mm.)	material	price(¥/m)	H-W Roughness
14	PE	2.5	140
17.7	PE	3.22	140
22	PE	5.22	140
29	PE	6.88	140
36.3	PE	10.63	140
45.4	PE	16.53	140
57.2	PE	26.25	140
68.2	PE	36.7	140
81.8	PE	53.08	140
100	PE	79.13	140
113.6	PE	102.47	140
127.3	PE	127.93	140
157.4	DI	138.43	140
209.2	DI	186.5	140
260.4	DI	222.7	140
311.6	DI	263.06	140
362.6	DI	330.38	140
412.8	DI	393.98	140
462.8	DI	464.35	140
514	DI	544.87	140
615.2	DI	717.06	140
716.4	DI	929.01	140
818.6	DI	1132.85	140
919.8	DI	1431.83	140
1021	DI	1768.07	140
1234.4	DI	2405.75	140

"PE" represents the polyethylene pipe. "DI" represents the ductile cast iron pipe.

have similar structures. Based on the domain knowledge, the number and the scale of loops in a network are two important indicators of the complexity [1]. Observing Fig. 4, we can find that the artificial networks contain many loops with different scales, so that the complexity of the problem is ensured. In addition, the minimum pressure constraint is set to 16 m according to the Chinese city standard.

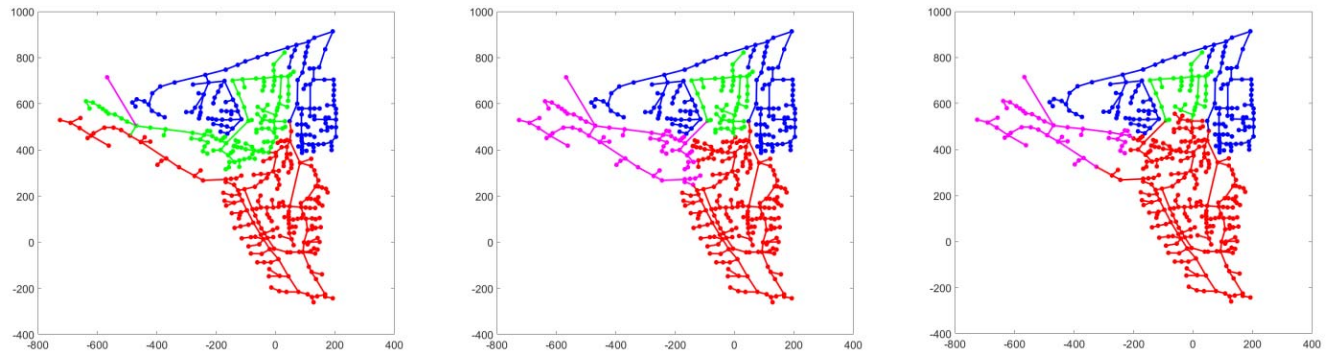
Regarding the pipe type, two kinds of pipes are widely used in the municipal water supply system: 1) polyethylene pipes and 2) ductile cast iron pipes. We investigated their market prices and manufacturing standards, adopting 12 types of polyethylene pipes and 14 types of ductile cast iron pipes into the experiment. Information about pipes is shown in Table IV.

Besides the networks we designed, a real and widely used WDN, i.e., the Balerma network, is adopted [54]. The structure of Balerma is also shown in Fig. 4. Comparing the artificial networks generated in this paper and the Balerma network, we can find that the artificial networks are more complex. According to Fig. 4(b) and (d), although 400-B-S and the Balerma network have similar scale, 400-B-S obviously has more loops, and the links among subnetworks are connected more closely.

B. Redecomposition for Single-Loading Cases

First, we discuss the performance of the redecomposition strategy on single-loading cases. Two questions, whether the redecomposition strategy is effective and how much the redecomposition interval RI should be, are investigated.

To answer the first question, WDNCC is applied to the Balerma network, and we keep track of the decomposition status to see whether the redecomposition strategy has refined the partitions of the network. The number of generations MAXROUND is set to 2000. The redecomposition interval RI is set to 200. Experimental results are shown in Fig. 5. In



(a) (b) (c)

Fig. 5. Decomposition of the Balerma network. Decomposition generated by the (a) max solution and (b) first feasible solution. (c) Final decomposition.

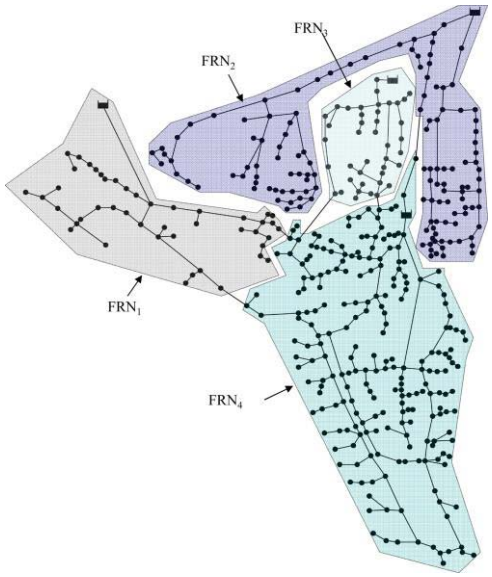


Fig. 6. Ideal decomposition of the Balerma network. FRN means four-reservoir network.

TABLE V
COMPARISON AMONG DIFFERENT RI VALUES

Instance	Index	50	100	200	400
400-B-S	median	0.048052	0.048153	0.048836	0.048842
	std.	0.001202	0.001054	0.0014	0.001377
	Wil.	-/E/E/E	E/-/E/E	E/E/-/E	E/E/E/-
400-I-S	median	0.061851	0.060435	0.061793	0.061172
	std.	0.008675	0.006211	0.00286	0.004958
	Wil.	-/E/E/E	E/-/E/E	E/E/-/E	E/E/E/-
Balerma	median	0.093503	0.092155	0.092978	0.092155
	std.	0.001235	0.001887	0.00167	0.00153
	Wil.	-/E/E/W	E/-/E/E	E/E/-/E	B/E/E/-

'E' means that the compared two values performs equally. 'B' means that the value is significantly better than the other. 'W' means that the value is significantly worse than the other. Taking the results of value 50 on the Balerma network as an example, the Wilcoxon rank sum test shows that it is equal to 100 and 200 and significantly worse than the value 400.

total, three decomposition results generated in different stages of WDNCC are shown: the first decomposition generated by $\boldsymbol{\theta}_{\max}$, the decomposition generated by the first found feasible solution, and the final decomposition. The decomposition given in [21] is shown in Fig. 6 as reference.

Observing the three figures in Fig. 5, we can find the decomposition of the network is becoming more and more precise along with the progressing of WDNCC. Compared with the ideal node decomposition shown in Fig. 6, in Fig. 5(a), the first decomposition is far away from accurate that one of the partitions owns only one pipe. However, the partitions in the final decomposition shown in Fig. 5(c) are extremely similar to the partitions in Fig. 6. Such a result shows that the redecomposition strategy is able to generate accurate partitions.

To answer the second question, how much the redecomposition interval RI should be, we choose three networks, 400-B-S and 400-I-S, and the Balerma network as test cases. RI is set to four different values {50, 100, 200, 400}. Other settings are kept unchanged. WDNCC is executed 20 times under each RI setting. The median and the standard deviation values of the results are shown in Table V. Also the Wilcoxon rank sum test is conducted to see whether there is a value which is significantly better than others. Observing the results of 400-B-S and 400-I-S, we can find that these four RI values have similar performance. This fact implies that WDNCC is actually not sensitive to the parameter RI on these two instances. However, in order to select a rational value, checking the median values, we take 100 as an applicable choice for the artificial networks. As for the Balerma network, although the value 400 gets the best performance on each measurement index, we can still see that except for 50, WDNCC performs similarly under the other three RI values. But 400 is absolutely a rational choice for the Balerma network.

Overall, the experiment shows that WDNCC is not quite sensitive to the parameter RI especially when the subnetworks are closely linked. As shown in Fig. 5(b) and (c), the difference between the decomposition generated by the first feasible solution and the final decomposition is little. In most cases, such small changes will not affect the optimization too much which means WDNCC has the ability to solve the problem when the decomposition is not that accurate. However, if the redecomposition process is executed too frequently, the individuals in different populations will be reassembled frequently. That will badly affect the optimization process just as the results on the

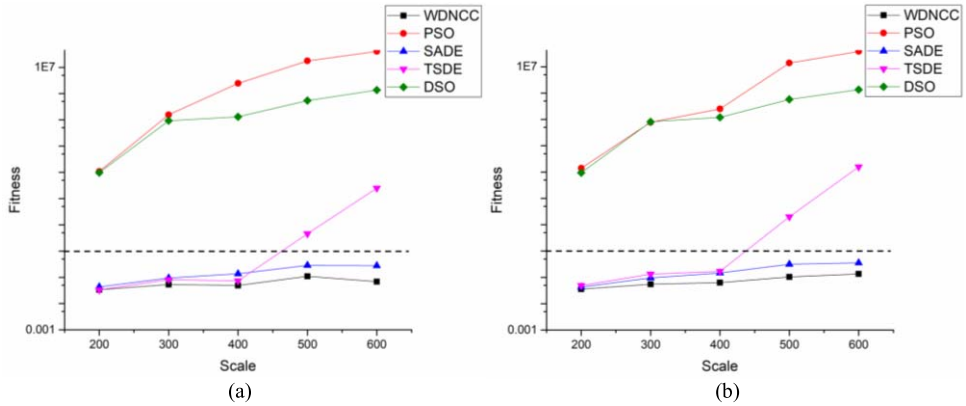


Fig. 7. Median values of fitness of the five algorithms on single-loading cases. (a) On balanced cases. (b) On imbalanced cases. The black dotted line is the feasible reference line which represents the fitness value 1. The area above the reference line represents the infeasible zone. The area below represents the feasible zone.

Balerma network. Thus, for real applications, we recommend that RI should be at least 100.

C. Performance on Single-Loading Cases

To show the effectiveness of WDNCC on single-loading cases, we compare it with the four aforementioned algorithms. The FEs is given as $2000 \cdot Nn$. The Balerma network is treated as a network of scale 400. Each algorithm is executed 20 times on each test case to get the statistic information, i.e., the median, the mean, and the standard deviation values. Also the Wilcoxon rank sum test is made to show whether WDNCC is significantly better than the other four methods. Numerical results are shown in Table VI. Furthermore, the median values are shown in Fig. 7 to demonstrate how the algorithms' performances are affected by the network scale.

Generally, except the 200-B-S network on which WDNCC performs similarly to TSDE, WDNCC is significantly better than the other four algorithms on the rest of test cases. Specifically, we can see that PSO is the worst among the five algorithms. Only on 200-I-S and Balerma, it has found feasible solutions. Although DSO is better than PSO, it still cannot find feasible solutions. TSDE and SADE have their own advantages, respectively. The results show that SADE can always find feasible solutions although the fitness values are very good. TSDE is able to find feasible solutions when the scale of the network is smaller than or equal to 400, and on some cases it performs better than SADE. But when the scale is larger, its ability degrades. As to WDNCC, on all test cases, it is capable to find feasible solutions. Meanwhile, the objective values are usually the best among the five algorithms, which means WDNCC is truly effective in optimizing large-scale WDNs with single-loading paradigm.

Also, Fig. 7 has demonstrated the great stability and scalability of WDNCC. Although all five algorithms' performances degenerate along with the growth of the network scale, the increase of the fitness of WDNCC is the smallest among them. The reason is that, by dividing a large-scale network into pieces, each optimizer in WDNCC actually faces a small-scale network which is easier to optimize than the whole network. Thus, the influence of the growth of network scale to WDNCC is not as fierce as to other algorithms.

D. Performance on Multiple-Loading Cases

The performance of WDNCC on multiple-loading cases is also checked by comparing it with the four algorithms. Experimental settings are kept unchanged except that ten independent runs are given to each algorithm rather than 20 because running a simulation on a multiple-loading network spends much more time than on a single-loading network. Numerical results are shown in Table VII. Also, median values are shown in Fig. 8.

Observing the results, we can find that generally the performances of the five algorithms on multiple-loading cases are similar to their performances on single-loading cases. Specifically, on 200-B-M, TSDE and WDNCC are well-matched. For other instances, numerical results show that WDNCC is always better. Regarding the other three algorithms, PSO is still not capable to find feasible solutions efficiently. DSO finally finds feasible solutions on the smallest network, but it still cannot handle large-scale networks. Although SADE shows valid and stable capability to find feasible solutions and to optimize the expenditure, its performance is still worse than WDNCC.

Since in multiple-loading networks, the demand of each node is multiplied with a coefficient which is in the range (0, 1), the minimum pressure constraint is much easier to be met than on the single-loading cases. Thus, we can find that except the 600-I-M network, TSDE can find feasible solutions on the other networks. However, the solutions it found still cost more than the solutions found by WDNCC. Results shown in Fig. 8 verify that WDNCC maintains good stability and scalability on multiple-loading cases too.

Overall, the experimental results show that WDNCC is also effective and efficient on the multiple-loading networks.

E. Redecomposition for Multiple-Loading Cases

Generally, a multiple-loading WDN is more difficult to be partitioned precisely than a single-loading WDN, since in different time slices, the nodes require different amount of water.

TABLE VI
RESULTS ON SINGLE-LOADING CASES

Algorithm	Mean	Median	std.	Wilcoxon	Mean	Median	std.	Wilcoxon
	200-B-S				200-I-S			
WDNCC	3.3652E-02	3.3834E-02	1.0268E-03		3.6022E-02	3.5792E-02	1.3269E-03	
PSO	2.8655E+03	1.0829E+03	4.5366E+03	better	2.0139E+03	1.4024E+03	2.0300E+03	better
DSO	1.2870E+03	9.7430E+02	1.5364E+03	better	9.6826E+02	9.4399E+02	5.4823E+02	better
SADE	4.3416E-02	4.3488E-02	1.1341E-03	better	4.3142E-02	4.3061E-02	1.0511E-03	better
TSDE	3.3260E-02	3.3180E-02	5.0123E-04	equal	4.9481E-02	4.8975E-02	3.5118E-03	better
	300-B-S				300-I-S			
WDNCC	5.3679E-02	5.3490E-02	1.5063E-03		5.5204E-02	5.4957E-02	1.3598E-03	
PSO	8.6064E+05	1.5613E+05	1.4661E+06	better	1.7897E+05	7.7815E+04	3.0046E+05	better
DSO	9.6201E+04	9.3046E+04	3.1665E+04	better	9.0660E+04	8.1189E+04	3.4867E+04	better
SADE	9.5420E-02	9.4102E-02	2.9784E-03	better	9.4625E-02	9.5761E-02	2.0837E-03	better
TSDE	8.2546E-02	8.1811E-02	5.6678E-03	better	1.3572E-01	1.3366E-01	7.5306E-03	better
	400-B-S				400-I-S			
WDNCC	4.9017E-02	4.9163E-02	2.1759E-03		6.5552E-02	6.2745E-02	6.8369E-03	
PSO	2.7549E+06	2.4535E+06	2.2300E+06	better	8.6846E+05	2.5061E+05	1.0242E+06	better
DSO	1.2546E+05	1.3051E+05	3.7425E+04	better	1.2788E+05	1.2020E+05	3.0464E+04	better
SADE	1.3577E-01	1.3690E-01	7.3808E-03	better	1.4738E-01	1.4730E-01	3.7960E-03	better
TSDE	7.3318E-02	7.3266E-02	1.3853E-03	better	1.6652E-01	1.6752E-01	8.9224E-03	better
	500-B-S				500-I-S			
WDNCC	1.0937E-01	1.0931E-01	5.2789E-03		1.0772E-01	1.0403E-01	7.6870E-03	
PSO	1.7679E+07	1.7691E+07	4.0542E+06	better	1.4857E+07	1.3985E+07	5.4435E+06	better
DSO	5.5798E+05	5.4011E+05	1.1679E+05	better	5.7066E+05	5.7549E+05	1.4523E+05	better
SADE	2.8422E-01	2.8517E-01	1.1478E-02	better	3.1661E-01	3.1572E-01	6.2233E-03	better
TSDE	4.4517E+00	4.5833E+00	2.3837E+00	better	2.0596E+01	1.9986E+01	7.3345E+00	better
	600-B-S				600-I-S			
WDNCC	6.8924E-02	6.8482E-02	1.9064E-03		1.3503E-01	1.3445E-01	1.7360E-02	
PSO	4.1326E+07	4.0649E+07	1.2463E+07	better	4.2180E+07	3.8429E+07	9.6462E+06	better
DSO	1.4038E+06	1.3637E+06	3.5038E+05	better	1.5268E+06	1.3625E+06	4.7148E+05	better
SADE	2.7641E-01	2.7790E-01	9.4394E-03	better	4.5789E-01	3.5450E-01	3.0575E-01	better
TSDE	2.6956E+02	2.5110E+02	6.8400E+01	better	1.5678E+03	1.5723E+03	3.4470E+02	better
	Balerma							
WDNCC	9.3096E-02	9.2396E-02	2.7814E-03					
PSO	1.2305E+02	3.5566E+01	1.4834E+02	better				
DSO	1.1535E+03	1.1556E+03	3.2693E+02	better				
SADE	9.6860E-02	9.7025E-02	7.1512E-04	better				
TSDE	9.6139E-02	9.6393E-02	1.0946E-03	better				

'better' represents that WDNCC is significantly better than the compared algorithm according to a Wilcoxon rank sum test at level 0.05.
'equal' represents that WDNCC gets similar results with the compared algorithm according to a Wilcoxon rank sum test at level 0.05.

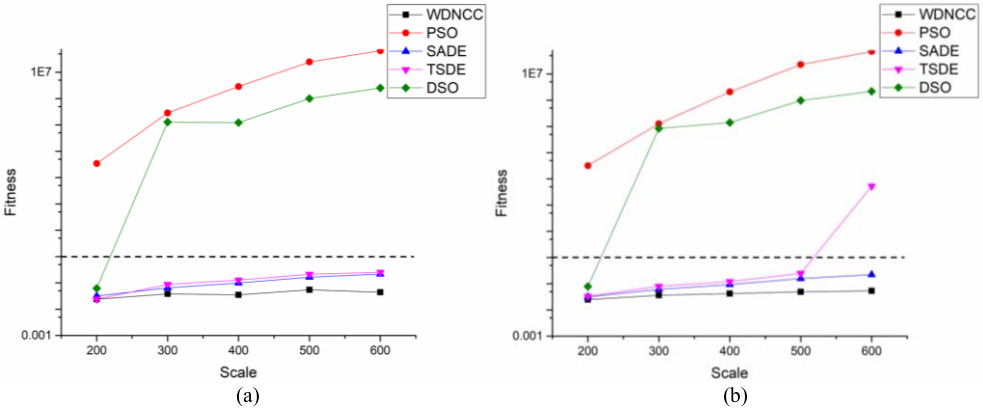


Fig. 8. Median values of fitness of the five algorithms on multiple-loading cases. (a) On balanced cases. (b) On imbalanced cases. The black dotted line is the feasible reference line which represents the fitness value 1. The area above the reference line represents the infeasible zone. The area below represents the feasible zone.

The water source of a node may also change in different time slices. Therefore, for some multiple-loading WDNs, theoretically, the perfect partition does not exist. Based on such a fact, here we do not arbitrarily discuss the accuracy of the decomposition. The changing of the decomposition is shown to demonstrate that the redecomposition strategy works well on multiple-loading cases.

First, the 200-B-M network and the 200-I-M network are used as the test cases. On 200-B-M, TSDE and WDNCC are well-matched, but on 200-I-M, WDNCC performs better. Thus, through comparing the decompositions generated by these two methods, we hope to see the difference and then discussing the accuracy. Still, the decomposition results in three different stages of WDNCC are shown. The partitions

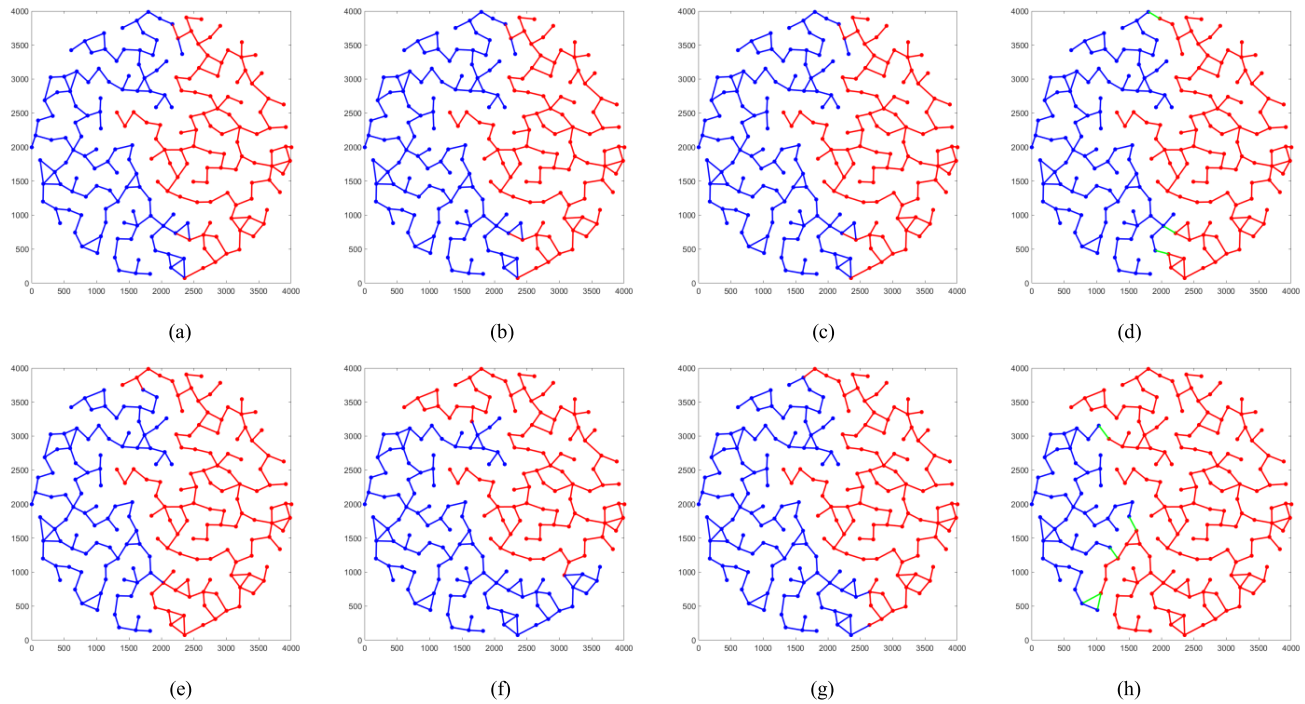


Fig. 9. Decomposition of 200-B-M and 200-I-M. (a) First decomposition of WDNCC on 200-B-M. (b) Second decomposition of WDNCC on 200-B-M. (c) Final decomposition of WDNCC on 200-B-M. (d) Decomposition of TSDE on 200-B-M. (e) First decomposition of WDNCC on 200-I-M. (f) Second decomposition of WDNCC on 200-I-M. (g) Final decomposition of WDNCC on 200-I-M. (h) Decomposition of TSDE on 200-I-M.

TABLE VII
RESULTS ON MULTIPLE-LOADING CASES

Algorithm	Mean	Median	std.	Wilcoxon	Mean	Median	std.	Wilcoxon
	200-B-M				200-I-M			
WDNCC	2.4179E-02	2.3838E-02	1.1766E-03		2.5067E-02	2.4749E-02	1.4630E-03	
PSO	3.0132E+03	3.4102E+03	2.3055E+03	better	7.3137E+03	3.1899E+03	1.1725E+04	better
DSO	6.6901E-02	6.2436E-02	1.7648E-02	better	7.7493E-02	7.8712E-02	2.4055E-02	better
SADE	3.1689E-02	3.1640E-02	9.2464E-04	better	3.1861E-02	3.1790E-02	1.0743E-03	better
TSDE	2.4113E-02	2.4057E-02	2.7593E-04	equal	3.3398E-02	3.3190E-02	1.6843E-03	better
	300-B-M				300-I-M			
WDNCC	3.8345E-02	3.8321E-02	1.2849E-03		3.6043E-02	3.6367E-02	1.1857E-03	
PSO	8.5174E+05	2.8598E+05	1.2346E+06	better	7.1812E+05	1.2662E+05	1.5351E+06	better
DSO	1.3255E+05	1.2807E+05	4.6344E+04	better	7.8815E+04	8.3806E+04	2.7600E+04	better
SADE	6.5330E-02	6.4668E-02	1.5162E-03	better	6.1708E-02	6.0802E-02	2.6575E-03	better
TSDE	8.8431E-02	8.7520E-02	5.1632E-03	better	8.0755E-02	7.9575E-02	4.2418E-03	better
	400-B-M				400-I-M			
WDNCC	3.5284E-02	3.5045E-02	1.2400E-03		4.2668E-02	4.1960E-02	4.0539E-03	
PSO	2.6421E+06	2.8733E+06	2.3177E+06	better	2.6283E+06	2.0662E+06	2.5572E+06	better
DSO	1.4087E+05	1.2293E+05	6.0095E+04	better	1.3843E+05	1.4098E+05	6.1639E+04	better
SADE	9.9773E-02	9.9011E-02	4.3418E-03	better	9.4120E-02	9.3012E-02	4.2408E-03	better
TSDE	1.2790E-01	1.2791E-01	3.7467E-03	better	1.2113E-01	1.2011E-01	4.0674E-03	better
	500-B-M				500-I-M			
WDNCC	5.3704E-02	5.4206E-02	2.3033E-03		4.9665E-02	4.9093E-02	2.2603E-03	
PSO	2.5382E+07	2.4570E+07	4.7127E+06	better	2.3674E+07	2.2771E+07	8.5467E+06	better
DSO	1.0461E+06	1.0125E+06	2.9654E+05	better	1.0262E+06	9.7244E+05	2.3507E+05	better
SADE	1.6488E-01	1.6471E-01	6.7703E-03	better	1.5644E-01	1.5868E-01	7.8023E-03	better
TSDE	2.1477E-01	2.1311E-01	9.0525E-03	better	2.4559E-01	2.4746E-01	7.5382E-03	better
	600-B-M				600-I-M			
WDNCC	4.4199E-02	4.4244E-02	8.0915E-04		5.4380E-02	5.3579E-02	2.8151E-03	
PSO	6.4793E+07	6.6122E+07	1.8689E+07	better	7.3405E+07	7.1712E+07	2.1869E+07	better
DSO	3.0257E+06	2.5291E+06	1.1471E+06	better	2.2373E+06	2.1663E+06	4.1228E+05	better
SADE	2.1601E-01	2.1717E-01	8.7728E-03	better	2.2518E-01	2.2324E-01	7.9592E-03	better
TSDE	2.5284E-01	2.5155E-01	8.4628E-03	better	6.5750E+02	5.2828E+02	3.2329E+02	better

'better' represents that WDNCC is significantly better than the compared algorithm according to a Wilcoxon rank sum test at level 0.05.
'equal' represents that WDNCC gets similar results with the compared algorithm according to a Wilcoxon rank sum test at level 0.05.

generated by TSDE are also displayed as reference. Other experimental settings are kept unchanged. Partitions are shown in Fig. 9. It should be noted that the partitions generated by TSDE are not ideal. They are displayed just to show the difference among partitions generated by the two methods. The green pipes shown in Fig. 9(d) and (h) are links between

subnetworks. In TSDE, they are not considered in the first stage of optimization.

First, seeing Fig. 9(a)–(d), we can find that the decomposition generated by WDNCC on 200-B-M is always the same, and it is quite similar to the decomposition generated by TSDE. Only on few links between the subnetworks, they differ from each other. Combining the results shown in Table VII, where they achieve similar performances on 200-B-M, we can say that these two methods both get accurate enough decomposition.

However, on 200-I-M, we can see significant change from Fig. 9(e) to (g). First, it means that the redecomposition process of WDNCC really functions well. Then, comparing these three figures with Fig. 9(h), we can find that no one is similar to the partitions generated by TSDE. Although, for the partitions generated by WDNCC on 200-I-M, the red subnetwork does own more nodes and links compared with 200-B-M, the water trace results show that the difference between the water supply capacities of the two reservoirs is not as big as shown in Fig. 9(h). The results in Table VII on 200-I-M also indicate that the decomposition generated by WDNCC is more accurate.

VI. Conclusion

In this paper, we have proposed a novel approach called WDNCC to handle the WDN optimization problem with multiple sources. With the help of the simulation tool, EPANET, an effective decomposition method is designed, which needs little hydraulic domain knowledge. Experimental results have proved the accuracy of the decomposition generated by WDNCC. Meanwhile, the cooperation of the decomposition process and the optimization process makes the algorithm efficient enough to find near-optimal solutions. Experimental results on both single-loading cases and multiple-loading cases have verified the effectiveness of WDNCC. Besides, in this paper, we have designed a series of large-scale WDNs which may fill the vacancy of the benchmarks.

In future research, besides proposing more effective and efficient algorithms, there are still some points which are worth studying.

1) Considering different structures of the WDNs, more decomposition methods should be designed specifically. This is also a key point to promote the usage of EAs on the problems with higher dimensionality.
2) Another big part of the urban water infrastructure is wastewater network. Optimizing the networks of wastewater is more challenging since pressure-free networks are more difficult to design than pressured networks.

References

[1] W. Zhao, T. H. Beach, and Y. Rezgui, "Optimization of potable water distribution and wastewater collection networks: A systematic review and future research directions," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 5, pp. 659–681, May 2016.

[2] B. Ceolho and A. Andrade-Campos, "Efficiency achievement in water supply systems—A review," *Renew. Sustain. Energy Rev.*, vol. 30, pp. 59–84, Feb. 2014.

[3] M. C. Cunha and L. Ribeiro, "Tabu search algorithms for water network optimization," *Eur. J. Oper. Res.*, vol. 157, no. 3, pp. 746–758, Sep. 2004.

[4] H. M. V. Samani and A. Mottaghi, "Optimization of water distribution networks using integer linear programming," *J. Hydraul. Eng.*, vol. 132, no. 5, pp. 501–509, May 2006.

[5] S. Atkinson, R. Farmani, F. A. Memon, and D. Butler, "Reliability indicators for water distribution system design: Comparison," *J. Water Resour. Plan. Manag.*, vol. 140, no. 2, pp. 160–168, Feb. 2014.

[6] D. F. Yates, A. B. Templeman, and T. B. Boffey, "The computational complexity of the problem of determining least capital cost designs for water supply networks," *Eng. Optim.*, vol. 7, no. 22, pp. 143–155, 1984.

[7] E. Alperovits and U. Shamir, "Design of optimal water distribution systems," *Water Resour. Res.*, vol. 13, no. 6, pp. 885–900, Dec. 1977.

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

[9] V. V. Sonak and P. R. Bhave, "Global optimum tree solution for single-source looped water distribution networks subjected to a single loading pattern," *Water Resour. Res.*, vol. 29, no. 7, pp. 2437–2443, Jul. 1993.

[10] K. E. Lansey and L. W. Mays, "Optimization model for water distribution system design," *J. Hydraul. Eng.*, vol. 115, no. 10, pp. 1401–1418, Oct. 1989.

[11] N. Duan, L. W. Mays, and K. E. Lansey, "Optimal reliability-based design of pumping and distribution systems," *J. Hydraul. Eng.*, vol. 116, no. 2, pp. 249–268, Feb. 1990.

[12] D. Savic and M. da Conceição Cunha, "Discussion of 'optimization of water distribution networks using integer linear programming' by Hossein M. V. Samani and Alireza Mottaghi," *J. Hydraul. Eng.*, vol. 134, no. 7, pp. 1024–1025, Jul. 2008.

[13] M. da Conceição Cunha and J. Sousa, "Water distribution network design optimization: Simulated annealing approach," *J. Water Resour. Plan. Manag.*, vol. 125, no. 4, pp. 215–221, Jul. 1999.

[14] A. De Corte and K. Sörensen, "An iterated local search algorithm for water distribution network design optimization," *Networks*, vol. 67, no. 3, pp. 187–198, Feb. 2016.

[15] E. Keedwell and S.-T. Khu, "Novel cellular automata approach to optimal water distribution network design," *J. Comput. Civil Eng.*, vol. 20, no. 1, pp. 49–56, Jan. 2006.

[16] W. Bi, G. C. Dandy, and H. R. Maier, "Improved genetic algorithm optimization of water distribution system design by incorporating domain knowledge," *Environ. Model. Softw.*, vol. 69, pp. 370–381, Jul. 2015.

[17] J. W. Nicklow *et al.*, "State of the art for genetic algorithms and beyond in water resources planning and management," *J. Water Resour. Plan. Manag.*, vol. 136, no. 4, pp. 412–432, Jul. 2010.

[18] A. C. Zecchin, H. R. Maier, A. R. Simpson, M. Leonard, and J. B. Nixon, "Ant colony optimization applied to water distribution system design: Comparative study of five algorithms," *J. Water Resour. Plan. Manag.*, vol. 133, no. 1, pp. 87–92, Jan. 2007.

[19] I. Montalvo, J. Izquierdo, R. Pérez, and M. M. Tung, "Particle swarm optimization applied to the design of water supply systems," *Comput. Math. Appl.*, vol. 56, no. 3, pp. 769–776, Aug. 2008.

[20] A. Vasan and S. P. Simonovic, "Optimization of water distribution network design using differential evolution," *J. Water Resour. Plan. Manag.*, vol. 136, no. 2, pp. 279–287, Mar. 2010.

[21] F. Zheng, A. R. Simpson, and A. C. Zecchin, "A decomposition and multistage optimization approach applied to the optimization of water distribution systems with multiple supply sources," *Water Resour. Res.*, vol. 49, no. 1, pp. 380–399, Jan. 2013.

[22] F. Zheng, A. C. Zecchin, and A. R. Simpson, "Self-adaptive differential evolution algorithm applied to water distribution system optimization," *J. Comput. Civil Eng.*, vol. 27, no. 2, pp. 148–158, Mar. 2013.

[23] Y. Zhang, *Water, Woodland to Dominate Xiongan New Area Landscape*, China Daily USA, Shijiazhuang, China, Oct. 2017. [Online]. Available: http://usa.chinadaily.com.cn/epaper/2017-10/03/content_32798731.htm

[24] Y.-J. Gong *et al.*, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Appl. Soft Comput.*, vol. 34, pp. 286–300, Sep. 2015.

[25] Q. Yang *et al.*, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 578–594, Aug. 2018.

[26] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000.

[27] S. Alvisi and M. Franchini, "A procedure for the design of district metered areas in water distribution systems," *Procedia Eng.*, vol. 70, pp. 41–50, Apr. 2014.
[28] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.
[29] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Apr. 2014.
[30] Q. Yang, W.-N. Chen, and J. Zhang, "Evolution consistency based decomposition for cooperative coevolution," *IEEE Access*, vol. 6, pp. 51084–51097, 2018.
[31] G. Ochoa, E. Lutton, and E. K. Burke, "The cooperative royal road: Avoiding hitchhiking," in *Proc. Int. Conf. Evol. Artificielle*, 2007, pp. 184–195.
[32] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer, "Polar IFS+ parisian genetic programming= efficient IFS inverse problem solving," *Genet. Program. Evol. Mach.*, vol. 1, no. 4, pp. 339–361, 2000.
[33] A. Tonda, E. Lutton, and G. Squillero, "A benchmark for cooperative coevolution," *Memetic Comput.*, vol. 4, no. 4, pp. 263–277, 2012.
[34] N. R. Sabar, J. Abawajy, and J. Yearwood, "Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 315–327, Apr. 2017.
[35] A.-M. Farahmand, M. N. Ahmadabadi, C. Lucas, and B. N. Araabi, "Interaction of culture-based learning and cooperative co-evolution and its application to automatic behavior-based system design," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 23–57, Feb. 2010.
[36] F. P. Vidal, E. Lutton, J. Louchet, and J.-M. Rocchisani, "Threshold selection, mitosis and dual mutation in cooperative co-evolution: Application to medical 3D tomography," in *Proc. PPSN*, 2010, pp. 414–423.
[37] L. A. Rossman, *EPANET 2: Users Manual*, Nat. Risk Manag. Res. Lab. Office Res. Dev., U.S. Environ. Protect. Agency, Cincinnati, OH, USA, 2000.
[38] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE CEC*, 2008, pp. 1110–1116.
[39] T. M. Walski *et al.*, "Battle of the network models: Epilogue," *J. Water Resour. Plan. Manag.*, vol. 113, no. 2, pp. 191–203, Mar. 1987.
[40] R. Farmani, G. A. Walters, and D. A. Savic, "Trade-off between total cost and reliability for anytown water distribution network," *J. Water Resour. Plan. Manag.*, vol. 131, no. 3, pp. 161–171, May 2005.
[41] A. R. Simpson, G. C. Dandy, and L. J. Murphy, "Genetic algorithms compared to other techniques for pipe optimization," *J. Water Resour. Plan. Manag.*, vol. 120, no. 4, pp. 423–443, Jul. 1994.
[42] C. R. Suribabu and T. R. Neelakantan, "Design of water distribution networks using particle swarm optimization," *Urban Water J.*, vol. 3, no. 2, pp. 111–120, 2006.
[43] R. Sheikholeslami and S. Talatahari, "Developed swarm optimizer: A new method for sizing optimization of water distribution systems," *J. Comput. Civil Eng.*, vol. 30, no. 5, Sep. 2016.
[44] C. R. Suribabu, "Differential evolution algorithm for optimal design of water distribution networks," *J. Hydroinformat.*, vol. 12, no. 1, pp. 66–82, Jan. 2010.
[45] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods Appl. Mech. Eng.*, vol. 186, nos. 2–4, pp. 311–338, Jun. 2000.
[46] F. Zheng, A. C. Zecchin, J. P. Newman, H. R. Maier, and G. Dandy, "An adaptive convergence-trajectory controlled ant colony optimization algorithm with application to water distribution system design problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 773–791, Oct. 2017.
[47] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *J. Water Resour. Plan. Manag.*, vol. 129, no. 3, pp. 210–225, May 2003.
[48] M.-D. Lin, Y.-H. Liu, G.-F. Liu, and C.-W. Chu, "Scatter search heuristic for least-cost design of water distribution networks," *Eng. Optim.*, vol. 39, no. 7, pp. 857–876, 2007.
[49] S. Sankaranarayanan, G. Swaminathan, N. Sivakumaran, and T. K. Radhakrishnan, "A novel hybridized grey wolf optimization for a cost optimal design of water distribution network," in *Proc. Conf. IEEE Comput.*, 2017, pp. 961–970.
[50] A. Villagra, D. Pandolfi, G. Leguizamón, and E. Alba, "Optimization of potable water networks with hybrid metaheuristics," in *Proc. RPIC*, 2017, pp. 1–6.
[51] M. A. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. 3rd Conf. Parallel Problem Solving Nat.*, 1994, pp. 249–257.
[52] Y.-H. Jia *et al.*, "Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization," *IEEE Trans. Evol. Comput.*, to be published.
[53] Y.-J. Gong *et al.*, "Optimizing the vehicle routing problem with time window: A discrete particle swarm optimization approach," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 2, pp. 254–267, Mar. 2012.
[54] J. Reca and J. Martínez, "Genetic algorithms for the design of looped irrigation water distribution networks," *Water Resour. Res.*, vol. 42, no. 5, May 2006.

Wei-Neng Chen (S'07–M'12–SM'17) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has co-authored over 90 papers in international journals and conferences, including over 30 papers in IEEE TRANSACTIONS journals. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering.

Dr. Chen was a recipient of the National Science Fund for Excellent Young Scholars in 2016 and the IEEE Computational Intelligence Society Outstanding Dissertation Award in 2016 for his doctoral thesis. He is the Vice-Chair of the IEEE Guangzhou Section.

Ya-Hui Jia received the bachelor's degree from Sun Yat-sen University, Guangzhou, China, in 2013, where he is currently pursuing the Ph.D. degree.

His current research interests include evolutionary computation algorithms and their applications on software engineering, cloud computing, and intelligent transportation.

Feng Zhao received the B.S. degree from the Guilin University of Electronic Technology, Guilin, China, in 1997 and the Ph.D. degree in communication and information system from Shandong University, Jinan, China, in 2007.

From 2008 to 2011, he was a part time Post-Doctoral Fellow with the Beijing University of Posts and Telecommunications, Beijing, China. In 2013, he was a Visiting Scholar with the University of Texas at Arlington, Arlington, TX, USA, for six months. He is currently a Professor with the Guangxi Colleges and Universities Key Laboratory of Complex System Optimization and Big Data Processing, Yulin Normal University, Yulin, China. He has published over 60 papers in journals and international conferences. His research has been supported by the National Science Foundation of China. His current research interests include cognitive radio networks, MIMO wireless communications, cooperative communications, and smart antenna techniques.

Dr. Zhao was a recipient of the second prize of the Shandong Province Science and Technology Progress, in 2007, 2012, and 2017.

Xiao-Nan Luo received the B.S. degree from Jiangxi Normal University, Nanchang, China, in 1983, the M.S. degree from Xidian University, Xi'an, China, in 1985, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 1991.

He is currently a Professor with the School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China. His current research interests include computer graphics, CAD, image processing, and mobile computing.

Dr. Luo was a recipient of the National Science Fund for Distinguished Young Scholars Granted by the National Nature Science Foundation of China, and was the Director of the National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou, China.

Xing-Dong Jia received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 1998.

He was a part-time Professor with Graduate School, Shenzhen Tsinghua University and Sun Yat-sen University, Guangzhou, China. He was also a Committee Member of the Scientific and Technological Commission of the Ministry of Industry and Information Technology of the People's Republic of China. He is currently the President of Shenzhen Polytechnic, Shenzhen, China. He has authored over 50 papers in international journals and conferences. His current research interest includes city informatization.

Jun Zhang (M'02–SM'08–F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

From 2004 to 2016, he was a Professor with Sun Yat-sen University, Guangzhou, China. Since 2016, he has been with the South China University of Technology, Guangzhou, where he is currently a Cheung Kong Chair Professor. He has authored seven research books and book chapters, and over 100 technical papers in his research areas. His current research interests include computational intelligence, cloud computing, big data, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits.

Prof. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and Current Chair of the IEEE Guangzhou Subsection and the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters. He is the Founding and Current Chair of the ACM Guangzhou Chapter.