A Two-Stage Swarm Optimizer with Local Search for Water Distribution Network Optimization

Ya-Hui Jia, *Member, IEEE,* Yi Mei, *Senior Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Evolutionary computation algorithms have been successfully applied to the small-scale water distribution network optimization problem. However, due to the city expansion, the network scale grows at a fast speed so that the efficacy of many current evolutionary computation algorithms degrades rapidly. To solve the large-scale water distribution network optimization problem effectively, a two-stage swarm optimizer with local search is proposed in this paper. To address the issues caused by the large-scale and multi-modal characteristics of the problem, the proposed algorithm divides the optimization process into an exploration stage and an exploitation stage. It first finds a promising region of the search space in the exploration stage. Then it searches thoroughly in the promising region to obtain the final solution in the exploitation stage. To search effectively the huge search space, we propose an improved level-based learning optimizer and use it in both the exploration and exploitation stages. Two new local search algorithms are proposed to further improve the quality of the solution. Experiments on both synthetic benchmark networks and a real-world network show that the proposed algorithm has outperformed the state-of-the-art meta-heuristic algorithms.**

***Index Terms*—Water Distribution Network Optimization, Large-Scale Global Optimization, Evolutionary Computation, Level-based Learning Swarm Optimizer, Meta-heuristic.**

I. Introduction

WATER distribution network (WDN) is an important infrastructure of a modern city to transfer water from sources to consumers [1]. The World Bank reports that there is still an urgent need for the infrastructure constructions like the electricity network and WDN in developing countries [2]. Building new WDNs and rehabilitating existing WDNs are very expensive and difficult tasks since they are related to many aspects [1], [3]. How to reduce the building or rehabilitating cost while guaranteeing a stable water supply capacity is a challenging problem.

A WDN usually consists of water sources (reservoir or tank), pipes, nodes (consumers), pumps, and some other components. Since the sizes of pipes largely influence the water supplying capacity and pipes usually occupy most of the expenditure of constructing a WDN, the WDN optimization problem is usually defined as the minimization of the total cost of all pipes, subject to a finite set of hydraulic and standardized constraints [1], [4]. The WDN optimization problem has been proven NP-hard [5]. In the past several decades, many algorithms were proposed to solve this problem. Roughly, these algorithms can be classified into three categories [1]: mathematical programming algorithms, individual-based meta-heuristic algorithms, and population-based meta-heuristic algorithms. The mathematical methods like linear programming [6], nonlinear programming [7], integer programming [4], were widely applied to small-scale problems. However, when the network scale grows, the high computational complexity makes these methods inefficient to generate good solutions within a limited time budget. The individual-based meta-heuristic algorithms such as simulated annealing (SA) [8], tabu search (TS) [9], and iterated local search (ILS) [10] showed good performance. However, due to the sensitivity to the predefined neighborhood structure and the initial solution, they are easy to be trapped into poor local optima. In contrast, the population-based meta-heuristic algorithms, mainly evolutionary computation (EC) algorithms such as particle swarm optimization (PSO) [11], [12], differential evolution (DE) [13], [14], genetic algorithm (GA) [15], [16], and ant colony optimization (ACO) [17], [18] are less likely to be trapped into poor local optima compared with the individual-based meta-heuristic algorithms, and have been proven effective for WDN optimization problems with less than 200 pipes. Nevertheless, as the problem scale keeps growing, the performance of the aforementioned algorithms including both the individual-based and population-based meta-heuristic algorithms degrades rapidly [19].

To reduce the complexity of the problem and shrink the search space, Zheng *et al.* [20] and Chen *et al.* [19] incorporated the divide-and-conquer strategy and proposed two different methods to divide a large-scale WDN that has more than 200 pipes into small sub-networks, but these two methods are only applicable to some specific types of multi-source networks. They cannot handle many other common network types such as the large-scale single-source networks and the multi-source networks that are hard to decompose.

In the research field of EC for large-scale global optimization (LSGO), scholars have found that insufficient exploration ability is a key reason of the incapability of many traditional EC algorithms in handling large-scale problems [21]. Accordingly, several new algorithms have been proposed by increasing the exploration ability of traditional algorithms, such as success-history based adaptive DE (SHADE) [22], competitive swarm optimizer (CSO) [23], and level-based learning swarm optimizer (LLSO) [24], [25]. These algorithms bring new opportunities to the WDN optimization, but they have not been widely investigated on the WDN optimization. Previously, we tried LLSO for the first time and found that

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509 and VUW1614, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, and the MBIE SSIF Fund under Contract VUW RTVU1914.

The authors are all with School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand (email: jia.yahui@foxmail.com; yi.mei@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

LLSO performed better than some traditional EC algorithms on the WDN optimization problem, but there is still much room for improvement [26].

Goal: In general, because of the multi-modal characteristic of the WDN optimization problem, the growth of the network scale increases the number of local optima exponentially. Some EC algorithms that were effective for small-scale networks with less than 200 pipes gradually lose their effectiveness on the large-scale networks due to the lose of population diversity. Considering this situation, we propose a two-stage swarm optimizer with local search (TSOL) to solve the large-scale WDN optimization problem effectively in this paper. Specifically, the contributions of the proposed TSOL are shown as follows:

1) A two-stage optimization process that consists of an exploration stage and an exploitation stage is proposed. TSOL in the exploration stage first finds a promising region in the huge search space. Then, in the exploitation stage, the solution is exploited in the promising region.
2) An improved LLSO, denoted as ILLSO, is proposed as the population-based optimization algorithm in TSOL. The velocity updating rule of LLSO is modified by considering the characteristics of the WDN optimization problem so that ILLSO can have a better exploration ability to search the huge search space effectively without being trapped into poor local optima.
3) Two new local search methods are proposed as the individual-based optimization methods to further refine the solutions.

Through the two-stage mechanism, we can fully take advantage of the good exploration ability of the newly proposed EC algorithm ILLSO in finding promising region in the huge search space, and the setting of the exploitation stage can further improve the exploitation ability of the algorithm in searching such a multi-modal region.

The rest of this article is organized as follows. First, the WDN optimization problem is defined in Section II, and the related works are reviewed in Section III. Then, Section IV describes TSOL in detail. Experiments are set up in Section V and conducted in Section VI. Finally, Section VII draws the conclusions.

II. Water Distribution Network Optimization

The construction of a WDN consists of many steps. From the basic investigation of the daily water consumption to the final construction, every step requires high expertise [27]. In this paper, we only focus on a single step, i.e. network optimization. Considering different objectives, scholars have proposed several models for this optimization problem [28], [29]. Since this paper focuses on studying the optimization algorithms rather than the problem models, we adopt the most basic and widely studied problem model that considers the construction expenditure as the objective [1]. Assuming that the investigation steps are already completed and the layout of the network is designed, the objective of the WDN optimization problem is to choose suitable type for each pipe so that the building expenditure can be minimized under three

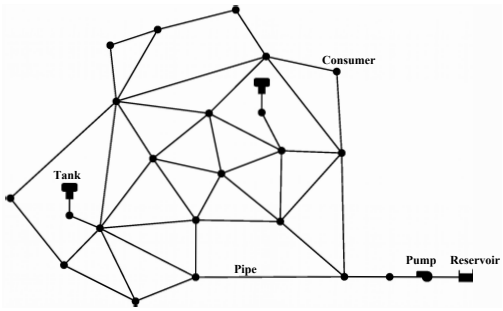


Fig. 1. Anytown water distribution network.

kinds of constraints [8], [19]. 1) Commercial constraint: every pipe should be chosen from the commercially available types. 2) Hydraulic and physical constraints: the law of conservation of energy and mass. 3) Pressure constraint: the pressure head of every consumer should be higher than a standardized value.

Among these three constraints,the commercial constraint defines the value range of variables (pipes) that is related to the encoding scheme. Hydraulic and physical constraints can be handled by the simulation tool EPANET2 [30]. The pressure head of each consumer under a specific configuration of the WDN is also estimated by the simulation, and the pressure constraint should be handled by the optimization algorithms.

A famous WDN, i.e. Anytown, is shown in Fig. 1 to facilitate description. Suppose there are Np pipes that each pipe i has a fixed length l_i, Nn consumers that the pressure head of each consumer j is H_j, and Nt commercially available pipe types $\{\zeta_1, \ldots, \zeta_{Nt}\}$. Denote the EPANET2 simulation as a function g to map a solution $\mathbf{x}$ (the configuration of all pipes) to the pressure heads of consumers $(H_1, \ldots, H_{Nn})$ under the other configuration of the network Ω. The problem can be formulated as:

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

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

$$H_{\min} \leq H_j, j = 1, \ldots, Nn, \tag{3}$$

$$(H_1, \ldots, H_{Nn}) = g(\mathbf{x}, \Omega). \tag{4}$$

The objective is defined by (1) where x^i represents the type of the ith pipe and $\mu(x^i)$ represents its unit price. The commercial constraint is shown in (2). The pressure constraint is shown in (3) where $H_{\min}$ represents the standardized pressure head value. Physical and hydraulic constraints are handled in (4) where the pressure heads are also calculated. It should be noted that the aforementioned model is not a mathematical programming model. It only shows the objective and the constraints from the the perspective of the optimization algorithm assisted by the simulation tool EPANET2.

In general, a pipe type with a larger diameter has a higher capability and a higher price than a smaller one. Sorting the pipe types by diameter or price, the values of variables are actually sequential rather than categorial. Thus, both combinatorial optimization algorithms like ACO [17] and continuous optimization algorithms like PSO [11] can be applied.

III. Related Work

In this section, the meta-heuristic algorithms that were proposed for WDN optimization are reviewed. Also, the PSO-based EC algorithms for LSGO are introduced to build the foundation of the proposed ILLSO.

A. Individual-based Meta-heuristic Algorithms

The basic idea of individual-based meta-heuristic algorithms is to keep refining a solution by local search and perturbation methods. Loganathan [31] first used the concept of "annealing" to design a heuristic algorithm for WDN optimization but they did not call their algorithm a SA. Afterwards, Cunha and Sousa [8] used SA to optimize two relatively small networks. Another popular algorithm is TS. Fanni *et al.* [9] first applied this algorithm. Later, Cunha [32] and Sung [33] also applied this algorithm to different WDNs. TS showed better performance than SA but for large-scale networks, it required a very large tabu list. Reca [34] tried several individual-based meta-heuristic algorithms and their combinations, including SA, ILS with SA, and SA with TS. The combinations of different individual-based meta-heuristic algorithms showed superiority than using each algorithm individually, and this work might be the first time that ILS was applied to WDN optimization. However, they did not show how the ILS is designed. De Corte and Sörensen [10], [35] recently proposed an ILS algorithm. The effectiveness of this algorithm is verified on several large-scale networks.

Generally, individual-based meta-heuristic algorithms have the potential to deal with large-scale WDN optimization problems, but, they usually need high expertise to design the local search and perturbation methods. If the applied local search or perturbation methods were not effective, the algorithm is easy to be trapped into poor local optima since it operates on a single solution. Also, they are usually sensitive to the quality of the initial solution [10], [34], [35].

B. Population-based Meta-heuristic Algorithms

Population-based meta-heuristic algorithms maintain a population containing a set of solutions rather than a single solution. The representative algorithm is GA [15]. Simpson *et al.* [16] first introduced how to use GA with a penalty function to solve the WDN optimization problem. After that, GA has been sufficiently studied from different aspects [36], such as the initialization method [37], evolutionary operator [38], and hybrid with other algorithms [39]. These modifications improved the performance of the canonical GA a lot, but GA still has some drawbacks like slow convergence speed.

PSO and DE become more and more popular nowadays. At the beginning, the canonical PSO was applied to this problem with different penalty functions [12], [40], [41]. Then, some works also considered to change the real-number encoding to integer encoding [42] and to use some variants of PSO [43]. Different DE algorithms were also applied to the WDN optimization problem [14], [44], [45]. Sedki [46] even tried to hybrid DE with PSO. These algorithms have shown great success in optimizing some small-scale WDNs, but when the scales of the networks grow larger, their effectiveness degrades greatly. This situation happens not only to PSO and DE, but also to some other EC algorithms [47]–[52]. In the research filed of EC for LSGO, scholars have found that these traditional EC algorithms suffer from the curse of dimensionality, and one important reason is that their exploration abilities are not good enough to work in the huge search space [21].

To reduce the complexity of optimizing large-scale WDNs, Zheng *et al.* [20] and Chen *et al.* [19] proposed different decomposition methods to divide a large-scale WDN into small sub-networks. However, both methods can only be applied to some multi-source networks since the sub-networks are generated according to the water sources.

C. Particle Swarm Optimization Variants for Large-scale Global Optimization

The WDN optimization problem is highly related to LSGO. In the research of EC for LSGO, several new EC algorithms have been proposed [22], [23], [53]. For the large-scale 'single-modal' problems, the algorithms that were designed based on DE [22] and evolutionary strategy (ES) [53] showed better results. However, for the large-scale 'multi-modal' problems, PSO variants like CSO [23] and LLSO [24] have shown competitive or even better performance [54] than DE and ES algorithms.

The canonical PSO maintains a swarm of particles representing the solutions. Each particle has its position and velocity. In each generation, every particle's velocity and position are updated according to:

$$v_i^d \leftarrow \omega v_i^d + r_1 c_1 (x_{gb}^d - x_i^d) + r_2 c_2 (x_{pb_i}^d - x_i^d), \quad (5)$$

$$x_i^d \leftarrow x_i^d + v_i^d, \quad (6)$$

where $\mathbf{x}_{gb}$ and $\mathbf{x}_{pb_i}$ are two exemplars for $\mathbf{x}_i$ to learn from, which are the best solution that the whole swarm has ever found and the best solution that the particle $\mathbf{x}_i$ has ever found, respectively. r_1 and r_2 are two random values generated within $(0, 1)$. c_1 and c_2 are two weight coefficients. ω is the inertia weight. d represents the dth dimension.

To enhance the exploration ability of the canonical PSO, new variants like CSO [23] and LLSO [24] were proposed by increasing the learning diversity of individuals and discarding the historical information. Specifically, through replacing $\mathbf{x}_{gb}$ and $\mathbf{x}_{pb}$ by the predominant individuals in the population as the learning exemplars, the velocity updating rule of the canonical PSO was changed to [23], [24]:

$$v_l^d \leftarrow r_1 v_l^d + r_2 (x_{e1}^d - x_l^d) + r_3 \varphi (x_{e2}^d - x_l^d), \quad (7)$$

where $\mathbf{v}_l$ represent the velocity of the learner $\mathbf{x}_l$. $\mathbf{x}_{e1}$ and $\mathbf{x}_{e2}$ are two exemplars taken or generated from the current population. $\varphi \in [0, 1)$ is a coefficient that balances the influence of the two exemplars. In different algorithms, there are different ways to generate these two exemplars.

In CSO, individuals will be pairwise coupled in each generation. Between each pair of individuals, the one with worse fitness value is the learner $\mathbf{x}_l$, and the one with better fitness value is the first exemplar $\mathbf{x}_{e1}$. The second exemplar $\mathbf{x}_{e2}$ is set as the central point of the whole population. Only

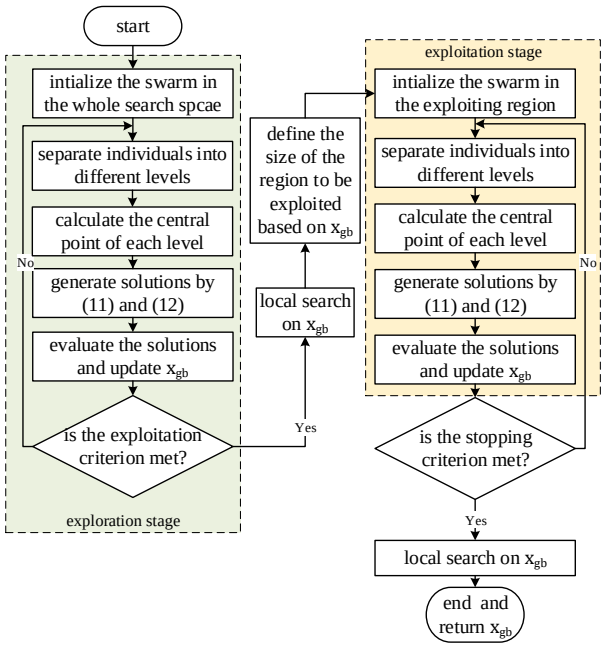


Fig. 2. Flowchart of TSOL.

the learners are updated in each generation. In this way, an individual can learn from different exemplars in different generations. Thus, the learning diversity of individuals is greatly increased. Meanwhile, moving towards the central point of the population, not only helps the population converge but also protects each individual from being trapped into poor local optima induced by its couple.

In LLSO, individuals are evenly separated into different levels in each generation according to their fitness values. Each individual at a specific level, denoted as the learner $\mathbf{x}_l$, can choose two better individuals in two higher levels as exemplars. The better one between the two exemplars is considered as $\mathbf{x}_{e1}$, the other one is taken as $\mathbf{x}_{e2}$. The individuals at the top level will be directly reserved for the next generation. By using this level-based learning strategy, the learning diversity and efficiency are increased. Meanwhile, learning from the second exemplar diversifies the moving direction of the learner in the search space, which is helpful for jumping out from local optima.

Recently, we tried LLSO in our previous work [26] and found that it was indeed more effective than some traditional EC algorithms. However, directly using it without adjustments would cause unstable performance. Due to the large-scale and multi-modal characteristics of the problem, both the exploration and the exploitation abilities of the existing algorithms need to be strengthened.

IV. Two-stage Swarm Optimizer with Local Search

Considering the drawbacks of the existing algorithms that were mentioned in Section III.A and Section III.B, we propose a new algorithm called TSOL. The flowchart of TSOL is shown in Fig. 2. The algorithm first comes into the exploration stage and the individuals in the swarm are first initialized randomly in the whole search space. Then, in each generation, new solutions are generated by ILLSO. These new solutions are evaluated to update the global best solution $\mathbf{x}_{\text{gb}}$. If the criterion to go to the exploitation stage is met, the local search algorithm is applied to refine $\mathbf{x}_{\text{gb}}$, and the promising region of the search space is defined based on the refined $\mathbf{x}_{\text{gb}}$. The exploitation stage is similar to the exploration stage, except that the individuals are re-initialized in the promising region. Finally, when the stopping criterion is met, $\mathbf{x}_{\text{gb}}$ is refined by the local search method and returned as the final solution.

In the following, the fitness function, the ILLSO algorithm, the method to define the size of the exploitation region, and the two local search algorithms are described in detail. Due to the page limit, the time complexity of TSOL and the difference between this work and our preliminary work [26] are discussed in the supplementary material.

A. Solution and Fitness Function Representation

The pipe types are sorted and indexed in the ascending order of diameter in TSOL, meaning that ζ_i has a smaller size and cheaper price than ζ_j if $i < j$. In TSOL, ILLSO still uses the real number coding rather than the integer number coding. We take the type of the ith pipe as $\lfloor x^i \rfloor$. $x^{14} = 3.2$ means that we choose ζ_3 for the 14th pipe.

The fitness function proposed in our previous work is applied to handle the constraint [19], where the pressure constraint is transferred into a penalty function:

$$F(\mathbf{x}) = f(\mathbf{x})/f(\mathbf{x}_{max}) + P(\mathbf{x}), \tag{8}$$

$$P(\mathbf{x}) = \sum_{i=1}^{Nn} \psi_i \cdot (1 + H_{min} - H_i), \tag{9}$$

$$\psi_i = \begin{cases} 1 & \text{if } H_{min} > H_i \\ 0 & \text{otherwise} \end{cases}. \tag{10}$$

$\mathbf{x}_{max}$ represents the solution in which the most expensive pipe type (is also the largest pipe type) is chosen for each pipe. H_{min} represents the threshold value of the minimum pressure constraint. $\mathbf{x}_{max}$ has the largest objective value, and it must be feasible. If $\mathbf{x}_{max}$ is infeasible, there will be no feasible solutions. Thus, for any solution, the first part of the fitness function $f(\mathbf{x})/f(\mathbf{x}_{max})$ is not greater than one. If a solution is feasible, the value of the penalty function will be zero. Therefore, the fitness of a feasible solution is always equal to or smaller than one. If a solution is infeasible, the value of the penalty function is equal to the number of nodes having water deficits plus the amount of deficits. This value is greater than one. Thus, for an infeasible solution, its fitness value is always greater than one. This fitness function perfectly obeys the constraint tournament selection rule [55]. By using this fitness function, we can simply compare any pair of solutions by their fitness values. The one with lower fitness value is better. Using (8) as the fitness function, the fitness landscape of one variable is shown in Fig. 3.

From Fig. 3, one may intuitively think that the integer coding scheme is more suitable to TSOL than the real-number coding since only the integer part of the variable is useful to represent the pipe type and the real-number coding brings plateaus to the search space. However, the real-number coding is actually a better choice because of two reasons. 1) Globally,

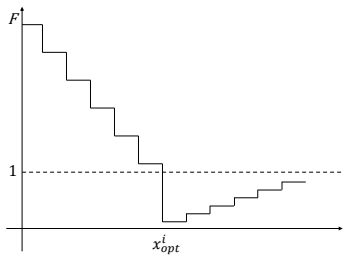


Fig. 3. Fitness landscape of one variable in the optimal solution, assuming that the other variables are held fixed.

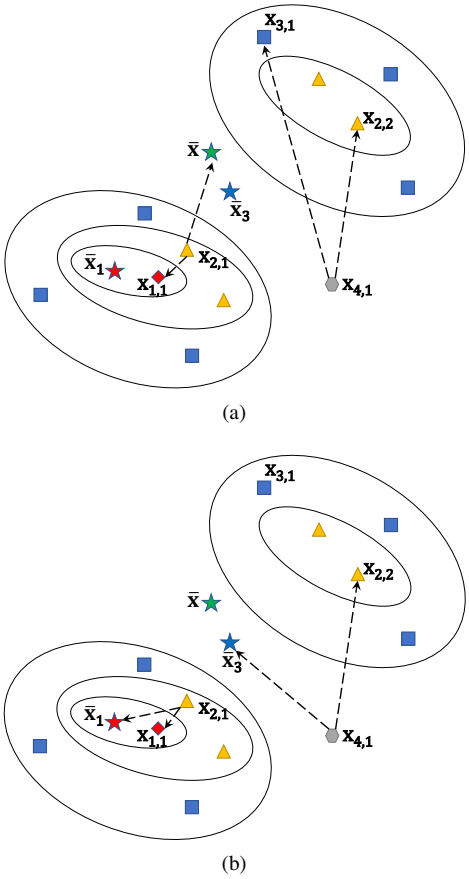


Fig. 4. Different ways of choosing exemplar of CSO, LLSO, and ILLSO. (a) CSO and LLSO. (b) ILLSO. The individuals are divided into four levels. The red rhombus represents the solution at the first level. The yellow triangles represent the solutions at the second level. The blue squares represent the solutions at the third level. The gray hexagon represents the solution at the lowest level. The green star represents the central point of the whole population. The blue star represents the central point of the individuals in the third level. The red star represents the central point of the individuals in the first level.

there is still approximate gradient information that can be utilized in the search space just like Fig. 3 showing. 2) Locally, using real-number coding can keep the individuals moving even in a small area of the search space, so that the algorithm can maintain a good exploitation ability. If the integer coding is applied, most individuals will be identical to each other in the later stage of optimization so that individuals will stop moving and the exploitation ability of the algorithm in a local area becomes poor. The experimental results in previous studies also showed that using the real-number coding in DE or PSO was a good choice for WDN optimization [56].

B. Improved Level-based Learning Swarm Optimizer

In TSOL, we propose a new EC algorithm called ILLSO to conduct the search during both the exploration and exploitation stages. In the previous section, we have introduced the reason of using PSO variants for WDN optimization and introduced two representative algorithms that were proposed for LSGO problems, CSO and LLSO. Before describing ILLSO, the advantages and disadvantages of these two algorithms are further demonstrated to motivate the design of ILLSO. Since the real fitness landscape of a large-scale WDN optimization problem is hard to display, to facilitate our explanation, a two-dimension multi-modal function is shown in Fig. 4 to illustrate the idea. $\mathbf{x}_{i,j}$ represents the jth solution in the ith level, where $F(\mathbf{x}_{i_1,j_1}) < F(\mathbf{x}_{i_1,j_1})$ if $i_1 < i_2$ or $i_1 == i_2 \wedge j_1 < j_2$. $\bar{\mathbf{x}}_i$ represents the central point of the solutions in the ith level. $\bar{\mathbf{x}}$ represents the central point of the population.

In CSO, the population evolves in a competitive way. Except for the best individual, every individual in the population has a certain chance to learn from others at each generation. This competitive learning strategy has a very strong learning diversity. However, because of the second exemplar, i.e. the central point of the population, there is a chance that a very promising solution will learn from a worse exemplar than itself. As shown in Fig. 4(a), a very promising individual $\mathbf{x}_{2,1}$ is coupled with the best individual $\mathbf{x}_{1,1}$ in the current population. The central point $\bar{\mathbf{x}}$ is much worse than $\mathbf{x}_{2,1}$. Since the distance between $\mathbf{x}_{2,1}$ and the central point $\bar{\mathbf{x}}$ is much longer than the distance between $\mathbf{x}_{2,1}$ and $\mathbf{x}_{1,1}$, according to (7), $\mathbf{x}_{2,1}$ tends to move towards $\bar{\mathbf{x}}$, a position that has a large probability to be worse than its current position.

LLSO on the other hand, applies a level-based learning strategy. By dividing the population into different levels, an individual in LLSO will never learn from the exemplars that are worse than itself. However, the two exemplars may be chosen from the same basin or adjacent basins so that the learner is guided to a poor local optimum, which is not good for exploration. As shown in Fig. 4(a), if the two exemplars $\mathbf{x}_{2,2}$ and $\mathbf{x}_{3,1}$ are chosen for $\mathbf{x}_{4,1}$, $\mathbf{x}_{4,1}$ will directly go to the upper basin whose local optimum is worse than that of the lower one.

To overcome the limitations of both algorithms, the ILLSO algorithm is proposed. The velocity and position updating rules of ILLSO are designed as follows:

$$v^d_{i_1,j_1} \leftarrow r_1 v^d_{i_1,j_1} + r_2(x^d_{i_2,j_2} - x^d_{i_1,j_1}) + r_3\varphi(\bar{x}^d_{i_3} - x^d_{i_1,j_1}), \tag{11}$$

$$x^d_{i_1,j_1} \leftarrow x^d_{i_1,j_1} + v^d_{i_1,j_1}, \tag{12}$$

where $\mathbf{x}_{i_1,j_1}$ represents the j_1th individual at the i_1th level and $\mathbf{x}_{i_2,j_2}$ represents the j_2th individual at the i_2th level. $\bar{\mathbf{x}}_{i_3}$ represents the central point of the individuals at the i_3 level. The relationship among i_1, i_2, and i_3 is $i_2 < i_3 < i_1$. d represents the dth dimension. In the extreme case when $i_1 = 2$, we allow that $i_2 = i_3 = 1$. After updating, if $x^d \geq Nt + 1$, we set $x^d = Nt$. If $x^d < 1$, we set $x^d = 1$.

To ensure the quality of the first exemplar, the level-based learning strategy is adopted in ILLSO. To prevent from being trapped into poor local optima, the central point of the

Algorithm 1 ILLSO

Input: The initialized population with M individuals $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, number of levels NL, fitness function $F(\mathbf{x})$, the best solution $\mathbf{x}_{\text{gb}}$.
Output: the best solution $\mathbf{x}_{\text{gb}}$.

1: **while** the stopping criterion and the exploitation criterion is not met **do**
2: rank the individuals according to their fitness values;
3: $ls = M/NL$;
4: **for** $i = 1 \rightarrow NL - 1$ **do**
5: $start = (i-1) \cdot ls + 1$; $end = i \cdot ls$;
6: $LS_i = \{\mathbf{x}_{start}, \ldots, \mathbf{x}_{end}\}$;
7: **end for**
8: $LS_{NL} = \{\mathbf{x}_{(NL-1)\cdot ls+1}, \ldots, \mathbf{x}_M\}$;
9: **for** $i = 1 \rightarrow NL - 1$ **do**
10: calculate $\bar{\mathbf{x}}_i$ of LS_i;
11: **end for**
12: **for** $i_1 = NL \rightarrow 3$ **do**
13: **for all** $\mathbf{x}_{i_1,j_1} \in LS_{i_1}$ **do**
14: find i_2 and i_3 that $i_2 < i_3 < i_1$;
15: choose the first exemplar $\mathbf{x}_{i_2,j_2}$ in LS_{i_2};
16: update $\mathbf{x}_{i_1,j_1}$ according to (11) and (12);
17: **end for**
18: **end for**
19: **for** each $\mathbf{x}_{2,j_1} \in LS_2$ **do**
20: choose the first exemplar $\mathbf{x}_{1,j_2}$ in LS_1;
21: set the second exemplar as $\bar{\mathbf{x}}_1$;
22: update $\mathbf{x}_{2,j_1}$ according to (11) and (12);
23: **end for**
24: calculate the fitness values of the updated solutions;
25: $\mathbf{x}_{\text{ib}} = \text{argmin}_{\mathbf{x}\in\{\mathbf{x}_1,\ldots,\mathbf{x}_M\}} F(\mathbf{x})$;
26: **if** $F(\mathbf{x}_{\text{gb}}) > F(\mathbf{x}_{\text{ib}})$ **then**
27: $\mathbf{x}_{\text{gb}} = \mathbf{x}_{\text{ib}}$;
28: **end if**
29: **end while**
30: **return** $\mathbf{x}_{\text{gb}}$;

individuals in the i_3th level are used as the second exemplar rather than a single individual in the i_3th level. From Fig. 4(b), we can see that $\mathbf{x}_{2,1}$ will learn from $\bar{\mathbf{x}}_1$ and $\mathbf{x}_{1,1}$. Compared with CSO, individuals in ILLSO will never learn from exemplars that are even worse than themselves. $\mathbf{x}_{4,1}$ will learn from $\mathbf{x}_{2,2}$ and $\bar{\mathbf{x}}_3$. Compared with LLSO, individuals in ILLSO will be less likely to be attracted into poor local optima.

The main evolutionary process of ILLSO is shown in Algorithm 1. Taking the initialized population containing M individuals, at the beginning of each generation, ILLSO ranks the individuals based on their fitness values (line 2). Then, the individuals are divided into NL levels (lines 3-8). Except for the last level, the central point of the individuals at each level is calculated (lines 9-11). Afterwards, from bottom to up, each individual is updated according to (11) and (12). For the individuals in the second level, since there are no two better levels that can be chosen, the first exemplar is randomly chosen from the first level and the second exemplar is taken as the central point of the first level (lines 12-23). All the updated individuals are evaluated to update $\mathbf{x}_{\text{gb}}$ (lines 24-28). Whenever the exploitation criterion or the stopping criterion is met, ILLSO returns the best solution it finds, i.e. $\mathbf{x}_{\text{gb}}$. If the stopping criterion is met, according to Fig. 2, after applying a local search algorithm to $\mathbf{x}_{\text{gb}}$, the whole TSOL will end. If the exploitation criterion is met, according to Fig. 2, TSOL will define a region to exploit and re-initialize the individuals in the region, and ILLSO will continue to run.

C. Exploitation Stage

Two questions should be considered when TSOL goes to the exploitation stage. 1) When does TSOL go to the exploitation criterion? 2) How to define the size of the exploitation region?

1) Exploitation Criterion: The exploitation stage is triggered by the exploitation criterion. If the exploration stage ends too early, ILLSO may not find a promising region for exploitation yet. If the exploration stage ends too late, computing resources will be wasted, making TSOL inefficient.

Intuitively, it is reasonable to stop the exploration when the algorithm has almost converged. In TSOL, considering the discrete characteristic of the variable, we propose a new measurement in this paper, denoted as ρ^g_{max}, to measure whether ILLSO has converged:

$$\rho^g_{\text{max}} = \max(\rho^g_1, \ldots, \rho^g_i, \ldots, \rho^g_{Np}), \tag{13}$$

$$\rho^g_i = \max(x^i_1, x^i_2, \ldots, x^i_M) - \min(x^i_1, x^i_2, \ldots, x^i_M), \tag{14}$$

where ρ^g_i represents the value range of the ith pipe in gth generation. For example, if in the third generation, the maximum value and the minimum value of the fourth pipe among all the individuals are 5.4 and 1.2, ρ^3_4 will be 5.4-1.2=4.2. Due to the discrete characteristic of the variable in WDN optimization, when the individuals in the population gather in a small area of the search space, ρ^g_{max} will fluctuate, which indicates the convergence of the algorithm. Thus, we judge whether ILLSO has converged by detecting the fluctuation of ρ^g_{max}. Due to the randomness of EC algorithms, we use the mean value of ρ^g_{max} in successive I generations to make the judgment. At the gth generation, if the following criterion is met, the exploration stage ends and we start the exploitation stage:

$$\left(\sum_{i=g-I}^{g} \rho^i_{\text{max}}/I \geq \sum_{i=g-2I}^{g-I} \rho^i_{\text{max}}/I\right) \wedge \left(\sum_{i=g-I}^{g} \rho^i_{\text{max}}/I < \epsilon\right) \tag{15}$$

where ϵ is a threshold value of ρ^g_{max}. To keep the algorithm simple, according to the observation of a related but different measurement used in our preliminary experimental study [26], I and ϵ are set to 30 and 5, respectively.

2) Exploitation Region: The exploitation region is defined as the neighborhood of $\mathbf{x}_{\text{gb}}$. Specifically, the value range of the dth variable is defined as [26]:

$$x^d \in [lb^d, ub^d), d = 1, 2, \ldots, Np, \tag{16}$$

$$lb^d = \max(x^d_{\text{gb}} - \max(Nt/8, 2), 1), \tag{17}$$

$$ub^d = \min(x^d_{\text{gb}} + \max(Nt/8, 2), Nt + 1), \tag{18}$$

where Nt is the number of commercially available pipe types. Basically, around the value of each variable in $\mathbf{x}_{\text{gb}}$, an upper bound and a lower bound are set to define the size of the neighboring region. For each dimension, the size of the neighboring region is at least four (± 2) to cover the potential better solutions in the region. There are two reasons of setting 2 as the minimum value instead of 1. 1) $\mathbf{x}_{gb}$ is refined by the local search method after the exploration stage. Most of the

Algorithm 2 BFLS

Input: the best solution $\mathbf{x}_{\text{gb}}$ and the fitness function $F(\mathbf{x})$.
Output: a refined best solution $\mathbf{x}_{\text{gb}}$.
1: $\Pi = \emptyset$; //list of tuple (saving, pipe)
2: **for** $i = 1 \rightarrow Np$ **do**
3: **if** $\lfloor x^i_{\text{gb}} \rfloor > 1$ **then**
4: $s = l_i \cdot (\mu(x^i_{\text{gb}}) - \mu(x^i_{\text{gb}} - 1))$;
5: add (s, i) to Π;
6: **end if**
7: **end for**
8: sort Π according to $saving$ in descending order;
9: **while** Π is not empty **do**
10: $\Pi' = \emptyset$;
11: **for all** $tuple \in \Pi$ **do**
12: $i = tuple.pipe$;
13: $x^i_{\text{gb}} = x^i_{\text{gb}} - 1$;
14: **if** $F(\mathbf{x}_{\text{gb}}) > 1$ **then**
15: $x^i_{\text{gb}} = x^i_{\text{gb}} + 1$;
16: **else if** $\lfloor x^i_{\text{gb}} \rfloor > 1$ **then**
17: $s = l_i \cdot (\mu(x^i_{\text{gb}}) - \mu(x^i_{\text{gb}} - 1))$;
18: add (s, i) to Π';
19: **end if**
20: **end for**
21: sort Π' according to $saving$ in descending order;
22: $\Pi = \Pi'$;
23: **end while**
24: **return** $\mathbf{x}_{\text{gb}}$;

Algorithm 3 DFLS

Input: the best solution $\mathbf{x}_{\text{gb}}$ and the fitness function $F(\mathbf{x})$.
Output: a refined best solution $\mathbf{x}_{\text{gb}}$.
1: $\Pi = [(saving, pipe)]$ //priority queue whose first element always has the biggest saving
2: **for** $i = 1 \rightarrow Np$ **do**
3: **if** $\lfloor x^i_{\text{gb}} \rfloor > 1$ **then**
4: $s = l_i \cdot (\mu(x^i_{\text{gb}}) - \mu(x^i_{\text{gb}} - 1))$;
5: add (s, i) to Π;
6: **end if**
7: **end for**
8: **while** Π is not empty **do**
9: $i = \Pi.firstElement.pipe$;
10: remove the first element from Π;
11: $x^i_{\text{gb}} = x^i_{\text{gb}} - 1$;
12: **if** $F(\mathbf{x}_{\text{gb}}) > 1$ **then**
13: $x^i_{\text{gb}} = x^i_{\text{gb}} + 1$;
14: **else if** $\lfloor x^i_{\text{gb}} \rfloor > 1$ **then**
15: $s = l_i \cdot (\mu(x^i_{\text{gb}}) - \mu(x^i_{\text{gb}} - 1))$;
16: add (s, i) to Π';
17: **end if**
18: **end while**
19: **return** $\mathbf{x}_{\text{gb}}$;

local search methods for WDN optimization reduce the size of pipes by one each time including the two local search methods proposed in this paper [10]. If 1 is used as the minimum value, there will be a big overlap between the space already searched by the local search method and the space going to be exploited by ILLSO. 2) (± 1) is a very small region that even there is a better solution, the superiority of that solution against $\mathbf{x}_{\text{gb}}$ is trivial. Meanwhile, the size of the neighboring region is at most a quarter $(\pm Nt/8)$ of the whole searching space to ensure the exploiting efficiency. ub^d is excluded in (16) since it may equal to $Nt + 1$, which is not in the value range.

D. Local Search Algorithms

Local search algorithms are used to make further refinement on $\mathbf{x}_{\text{gb}}$ after ILLSO converges. In this paper, we propose two local search algorithms to reduce the size of each pipe. The first one is the breadth-first local search (BFLS) and the second one is the depth-first local search (DFLS). The pseudo-codes of these two algorithms are shown in Algorithm 2 and Algorithm 3. It should be noted that it is usually not difficult for TSOL to find feasible solutions. Thus, $\mathbf{x}_{\text{gb}}$ is assumed to be feasible in the local search. If the input $\mathbf{x}_{\text{gb}}$ is not feasible, the local search algorithms would directly stop.

In BFLS, firstly, every pipe whose size can be reduced is stored in a list Π with its corresponding saving (line 1-7). The saving of a pipe represents how much money can be saved if we reduce the size of the pipe by one. Then, the list Π is sorted in the descending order of the saving (line 8). In the main loop (line 9-23), the pipes in the list are checked one by one. The size of each pipe is reduced by one (line 11-13). If $\mathbf{x}_{\text{gb}}$ becomes infeasible after size reduction, we revert the size of the pipe back (line 14-15). Otherwise, the saving of this pipe is updated and saved to a new list Π' (line 16-18). After all the pipes in Π have been checked, the new list Π' is sorted to start a new round (line 21-22). When there is no pipe can be further optimized, BFLS stops and returns $\mathbf{x}_{\text{gb}}$.

In DFLS, we do not check the pipes in the list one by one. Instead, a priority queue is maintained. The first pipe with the largest saving in the priority queue is always checked. As the two names suggest, the difference between BFLS and DFLS is that in BFLS we go through all the potential pipes in each iteration, while in DFLS we always choose the pipe whose saving is the largest. Thus, DFLS is greedier than BFLS. In DFLS, if there is a pipe that keeps being the first one in the priority queue, it will be always chosen. However, this may lead to a situation of "over decrease" that spares no room for the other pipes to be optimized. For BFLS, the fundamental idea is to avoid the "over decrease" of a specific pipe. In each iteration, we still optimize the pipes according to their savings but we give every pipe in the list a chance to be optimized in each iteration.

V. Experiment Setup

A. Test Cases

The WDN benchmark set created in [19] is adopted. It contains 15 WDNs. In addition, a well-know real-world network, i.e. Balerma Network [57], is tested. The information of the WDN instances is shown in Table I. There are 26 pipe types prepared for the benchmark WDNs [19] and 10 pipe types prepared for the Balerma Network [57]. The minimum pressure constraints of the synthetic networks and Balerma are 16 meters [19] and 20 meters [57], respectively. The structures of B400 and the Balerma network are shown in Fig. 5.

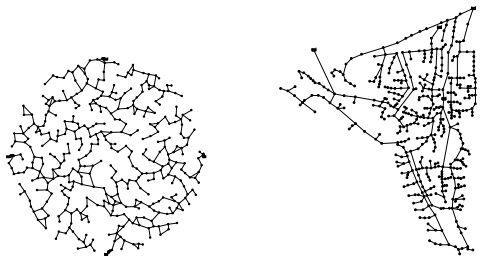
B. Algorithm Configuration

1) Population Size: According to the empirical studies in [41], [45], when using population-based algorithms to solve the WDN optimization problems, setting the population size

TABLE I
INFORMATION OF THE BENCHMARK INSTANCES.

Name	Nn	Np	Res	Sup
S200	200	226	1	single
B200	200	226	2	balanced
I200	200	226	2	imbalanced
S300	300	328	1	single
B300	300	328	3	balanced
I300	300	328	3	imbalanced
S400	400	452	1	single
B400	400	452	4	balanced
I400	400	452	4	imbalanced
S500	500	546	1	single
B500	500	546	5	balanced
I500	500	546	5	imbalanced
S600	600	661	1	single
B600	600	661	6	balanced
I600	600	661	6	imbalanced
Balerma	443	452	4	imbalanced

'single' means there is only one water source. 'balanced' means there are multiple water sources with balanced supplying capacity. 'imbalanced' means there are multiple water sources with imbalanced supplying capacity.



(a) (b)

Fig. 5. Structures of B400 and Balerma.

approximately equal to the scale of the problem is a simple yet effective strategy. This strategy is in line with the analysis of LLSO on large-scale function optimization problems [24]. Since the edge-per-node ratio Np/Nn is small for the test instances, we set the population size to Nn to facilitate the implementation of the algorithm for the sake of simplicity. If Np/Nn is larger than 1.2, we recommend to set the population size based on Np rather than Nn.

2) Weight of the Second Exemplar: Besides the population size and the parameters in (15)-(18) that we have already set based on our preliminary work, another important parameter that should be considered, which is φ in the velocity updating rule of ILLSO (11). To avoid the influence of the local search algorithm, a pure ILLSO is tested with five candidate φ values $\{0, 0.05, 0.1, 0.15, 0.2\}$, which means that we directly use the algorithm showed in Algorithm 1 and used the stopping criterion in line 1.

To see whether this parameter is sensitive to the scale of the problem and the type of the network, six instances {S200, S400, S600, B200, B400, B600} are selected. ILLSO is tested on each instance with each candidate value for ten independent times. Following the experimental setting in [19], [26], $Nn \cdot 4000$ times of fitness evaluations are given as the stopping criterion. The experimental results of the objective value are shown in Fig. 6 in the form of box plot.

TABLE II
COMPARISON OF THE OBJECTIVE VALUE BETWEEN BFLS AND DFLS WHEN THEY ARE USED IN TSOL.

Instance	Index	TSOL-B	TSOL-D
S200	mean	6.5927E+06	6.5921E+06
	std.	2.2467E+05	2.2921E+05
B200	mean	3.8958E+06	3.8943E+06
	std.	4.5867E+04	4.9647E+04
S400	mean	3.7125E+07	3.7148E+07
	std.	3.1539E+05	3.1493E+05
B400	mean	2.0646E+07	2.0637E+07
	std.	2.3171E+05	2.0827E+05
S600	mean	1.4105E+08	1.4101E+08
	std.	1.2254E+06	1.2787E+06
B600	mean	6.2493E+07	6.2508E+07
	std.	4.5744E+05	4.1998E+05

As can be seen from the figure, this parameter is sensitive to the scale of the problem, but it is not sensitive to the type of the network. According to the patterns shown in Fig. 6, we can come up with a simple adaptive setting strategy of φ as follows:

$$\varphi = \max((Nn/200 - 1) \cdot 0.05, 0). \tag{19}$$

3) Local Search Selection: Since two local search algorithms have been proposed in TSOL, we need to examine which one is more suitable for TSOL through the experiment. TSOL-B and TSOL-D are used to represent TSOL using BFLS and DFLS, respectively. They are also tested on the selected six instances. Each algorithm is executed 30 independent times. The experimental results are shown in Table II.

From the table, we can see that TSOL-B and TSOL-D achieved very similar results. The reason for this phenomenon is that we only applied the local search algorithm after the algorithm converges. Under this circumstance, $\mathbf{x}_{\text{gb}}$ is already well-optimized although it is not a local optimum in most cases. Using BFLS and DFLS will improve the quality of $\mathbf{x}_{\text{gb}}$ but the two local search algorithms will push $\mathbf{x}_{\text{gb}}$ to similar local optima. For the sake of simplicity, we use TSOL-B to represent TSOL in the following experiment. The effectiveness of local search in TSOL is also further discussed in the following experiment.

Overall, to facilitate the usage of TSOL to other WDN optimization problems, for the parameters in TSOL, either we have given fixed values like I and ϵ in (15) or an adaptive strategy is offered like (19).

VI. COMPARISONS AND ANALYSES

In this section, TSOL is compared with the state-of-the-art meta-heuristic algorithms proposed for the WDN optimization problem. The mathematical programming methods are not compared because they requires extremely high computational budget to converge on large-scale WDNs, which is not efficient [1]. Then, the components of TSOL are investigated individually. Due to the page limit, some experimental analyses are provided in the supplementary material including the success rate of finding feasible solutions and how long each algorithm takes to outperform heuristic methods.

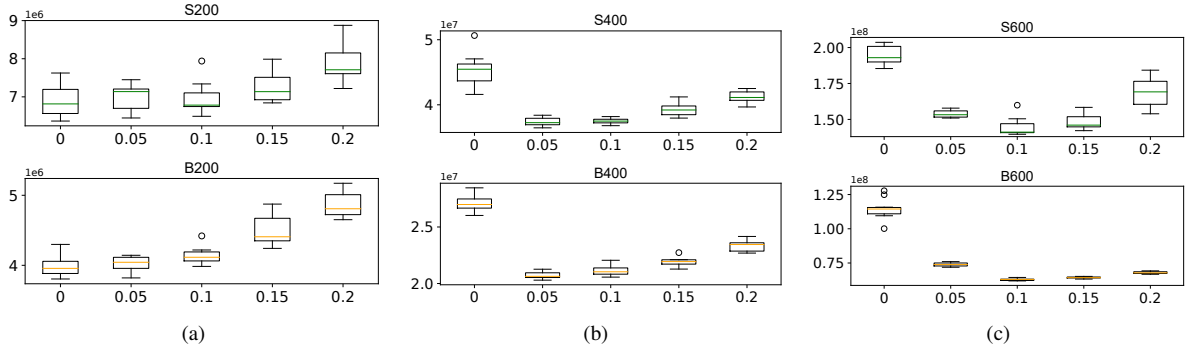


Fig. 6. Parameter selection of φ. x-axis represents the candidate φ value. y-axis represents the objective value. (a) S200 and B200, (b) S400 and B400, (c) S600 and B600.

A. Comparison with Meta-heuristic Algorithms for WDN Optimization

The following algorithms are compared, including ILS, self-adaptive DE (SADE), WDNCC, and LLSORL. The recently proposed ILS [10], [35] is chosen as the representative individual-based meta-heuristic algorithm. SADE [45] is chosen to represent the classical EC algorithm that is used to solve WDN optimization problems. It has significantly outperformed the canonical PSO and showed competitive performance compared with the min-max ant system [19], [45]. WDNCC [19] is taken as the representative algorithm that adopts the divide-and-conquer strategy. Since the dividing method of WDNCC can only be applied to multi-source networks, we will not test it on the single-source networks. LLSORL [26] is the algorithm we proposed in our preliminary work. The parameters of these compared algorithms are set following their original settings. In addition, BFLS and DFLS are directly applied to $\mathbf{x}_{max}$ as the baselines.

$Nn \cdot 4000$ times of fitness evaluations are given as the stopping criterion. Each algorithm is executed 30 independent times on each WDN. The experimental results are shown in Table III including the mean objective value, the standard deviation, and the result of the Friedman test with the Holm-Bonferroni post-hoc test in which the significance level is set to 0.05. From Table III, we can get the following observations:

- TSOL has never been defeated by any compared algorithm on any instance. According to the significance test, it either significantly outperforms the compared algorithms or shares similar performance with them. According to the mean fitness value, it is better than all the compared algorithms.
- Compared with ILS, TSOL is significantly better on all the instances except S200, on which they achieved similar results. Compared with SADE, TSOL outperforms it on all instances. It is obvious that the performance of SADE degrades rapidly as the scale of the network grows. These two comparisons show that TSOL is better than the individual-based algorithms and classical EC algorithms for large-scale WDN optimization problems.
- Compared with WDNCC, TSOL is significantly better on the multi-source networks, which indicates that TSOL is effective enough even without using the divide-and-conquer strategy.
- Compared with LLSORL, TSOL showed significant advantages on most of the instances. Although the significance test shows that they have similar performance on the three largest networks, the mean fitness values show that TSOL is still better.

The convergence curves of the compared algorithms are drawn in Fig. 7 to compare the efficiency of the algorithms. Due to the page limit, only the 400-scale and 600-scale instances are shown. From the figures, we can see that the rank of the convergence speeds of the compared algorithms is ILS>LLSORL>TSOL>WDNCC>SADE.

- The efficiency of SADE is far worse than the other algorithms. As we have explained that the traditional EC algorithms' exploration ability is not enough to explore such a big searching space. Thus, it takes a long period for SADE to find feasible solutions and after that its converging speed is still slower than the others.
- Since ILS starts from a feasible solution that is generated from $\mathbf{x}_{max}$ and the EC algorithms are initialized randomly, its starting point in the figure is always better than the others. Then, the local search algorithm provides a fast convergence speed. However, due to the lack of global search ability, ILS is easy to be trapped into poor local optima.
- Among all the algorithms except for SADE, WDNCC has the slowest converging speed since after decomposition, every time it can only optimize a part of the network. To improve the quality of the whole solution, it requires more fitness evaluations than the other algorithms.
- The convergence speed of LLSORL is faster than TSOL in general. The reason is that the original LLSO has a faster convergence speed than ILLSO. Using (11) indeed slows the converging speed a little but it brings performance improvement.

Overall, from the results shown in Table III and Fig.7, we can see that TSOL is effective to find good solutions with a moderate convergence speed.

After the comparison, a question rises that can we use the initial solution of ILS in EC algorithms as an initial individual to accelerate the convergence speed? It is certainly easy to adopt a feasible solution in the initial population, but this strategy will impair the learning diversity of the algorithm. Take TSOL as an example. In the early stage of the

TABLE III
COMPARISONS OF THE OBJECTIVE VALUE BETWEEN THE STATE-OF-THE-ART META-HEURISTIC ALGORITHMS THAT WERE PROPOSED FOR WDN OPTIMIZATION AND TSOL ON 15 SYNTHETIC WDNS AND THE BALERMA WDN.

Instance	Index	BFLS	DFLS	ILS	SADE	WDNCC	LLSORL	TSOL
S200	mean	1.3511E+07(+)	1.2571E+07(+)	6.7881E+06(=)	7.0374E+06(+)	NA	6.7737E+06(=)	**6.5927E+06**
	std.	0	0	3.5648E+05	2.5063E+05	NA	4.3788E+05	2.2467E+05
B200	mean	5.7963E+06(+)	6.1664E+06(+)	4.0878E+06(+)	4.3893E+06(+)	4.0980E+06(+)	4.0629E+06(+)	**3.9151E+06**
	std.	0	0	7.1068E+04	7.7747E+04	1.2802E+05	1.3432E+05	4.5867E+04
I200	mean	6.7201E+06(+)	6.6560E+06(+)	4.0060E+06(+)	4.3954E+06(+)	4.5329E+06(+)	4.0756E+06(+)	**3.9124E+06**
	std.	0	0	7.4218E+04	7.5100E+04	3.6378E+05	1.0673E+05	6.1870E+04
S300	mean	4.5130E+07(+)	4.1650E+07(+)	2.7492E+07(+)	2.8019E+07(+)	NA	2.4476E+07(+)	**2.3721E+07**
	std.	0	0	9.2725E+05	4.0753E+05	NA	6.2838E+05	4.0499E+05
B300	mean	2.2405E+07(+)	2.0653E+07(+)	1.3439E+07(+)	1.6827E+07(+)	1.3550E+07(+)	1.3181E+07(+)	**1.2846E+07**
	std.	0	0	3.5662E+05	4.2044E+05	4.9871E+05	3.7207E+05	3.1103E+05
I300	mean	2.2503E+07(+)	2.3306E+07(+)	1.4267E+07(+)	1.6629E+07(+)	1.3825E+07(+)	1.3232E+07(+)	**1.2854E+07**
	std.	0	0	3.9765E+05	3.2546E+05	5.2606E+05	4.3792E+05	2.0262E+05
S400	mean	9.3958E+07(+)	1.1575E+08(+)	4.0064E+07(+)	5.3281E+07(+)	NA	3.7754E+07(+)	**3.7125E+07**
	std.	0	0	9.9122E+05	9.8846E+05	NA	7.7928E+05	3.1539E+05
B400	mean	4.5127E+07(+)	4.3382E+07(+)	2.1997E+07(+)	3.6483E+07(+)	2.1797E+07(+)	2.1000E+07(+)	**2.0646E+07**
	std.	0	0	3.7654E+05	1.3986E+06	7.3008E+05	4.4458E+05	2.3171E+05
I400	mean	5.0355E+07(+)	4.9814E+07(+)	2.9670E+07(+)	4.0109E+07(+)	2.9122E+07(+)	2.5498E+07(+)	**2.4381E+07**
	std.	0	0	1.4340E+06	1.7659E+06	2.5861E+06	1.7857E+06	3.5952E+05
S500	mean	2.1570E+08(+)	2.1854E+08(+)	1.2094E+08(+)	1.7045E+08(+)	NA	1.1244E+08(+)	**1.1031E+08**
	std.	0	0	3.0723E+06	6.1290E+06	NA	3.4776E+06	1.0702E+06
B500	mean	1.3739E+08(+)	1.6715E+08(+)	8.4167E+07(+)	1.3121E+08(+)	7.7616E+07(=)	7.5577E+07(=)	**7.4309E+07**
	std.	0	0	1.5369E+06	3.5297E+06	7.0974E+06	3.5778E+06	3.4349E+06
I500	mean	1.5885E+08(+)	1.7740E+08(+)	7.7815E+07(+)	1.7351E+08(+)	8.2374E+07(+)	7.4511E+07(+)	**7.1872E+07**
	std.	0	0	4.2566E+06	8.6856E+06	5.7135E+06	3.5499E+06	2.2856E+06
S600	mean	3.1838E+08(+)	2.6361E+08(+)	1.4966E+08(+)	2.5626E+08(+)	NA	1.4420E+08(=)	**1.4105E+08**
	std.	0	0	3.5925E+06	8.4597E+06	NA	5.5228E+06	1.2254E+06
B600	mean	2.1465E+08(+)	2.0515E+08(+)	7.2097E+07(+)	1.7104E+08(+)	6.7755E+07(+)	6.2705E+07(=)	**6.2493E+07**
	std.	0	0	2.1610E+06	6.2474E+06	2.8038E+06	5.2842E+05	4.5744E+05
I600	mean	1.8516E+08(+)	2.1026E+08(+)	1.2984E+08(+)	2.0185E+08(+)	1.2932E+08(+)	1.0839E+08(=)	**1.0553E+08**
	std.	0	0	3.8814E+06	4.1387E+06	1.8604E+07	7.6799E+06	7.4946E+06
Balerma	mean	3.0749E+06(+)	3.9560E+06(+)	2.1428E+06(+)	2.0856E+06(+)	2.0495E+06(+)	2.0267E+06(+)	**2.0106E+06**
	std.	0	0	8.6004E+04	9.8686E+03	2.3784E+04	3.1662E+04	3.2657E+04
w/t/l	–	16/0/0	16/0/0	15/1/0	16/0/0	10/1/0	11/5/0	–

'NA' represents 'not available'. (+) represents that TSOL is significantly better than the compared algorithm. (-) represents that TSOL is significantly worse than the compared algorithm. (=) represents that there is no significant difference between TSOL and the compared algorithm. 'w/t/l' represents on how many instances TSOL has won, tied, or lost the comparison.

optimization, this individual on a large probability would be the only feasible one especially for large-scale WDNs. Thus, this individual will not move. Once the other individuals are coupled with this feasible one, they will move towards a same position in the search space, which is not good for exploration. We leave this question of how to utilize the feasible solution to accelerate the convergence speed without losing learning diversity as an open issue for future study.

B. Investigation of ILLSO

In TSOL, ILLSO is proposed and used as the optimizer to search the huge search space. To verify the effectiveness of ILLSO as the optimizer, we compare it with CSO, LLSO, SHADE, and IPOP-CMA-ES. CSO [23] and LLSO [24] are the representatives of PSO variants for LSGO. Although SHADE as a DE variant was originally proposed for small-scale function optimization, it has been applied to LSGO and achieved competitive results recently [22]. CMA-ES is usually used as the optimizer of cooperation co-evolutionary algorithms for LSGO [54]. However, it is very hard to use it within WDNCC because it cannot fit the periodic decomposition method of WDNCC. Since there is no guideline of how to use CMA-ES for LSGO directly, considering the multi-modal characteristic of WDN optimization, we use the famous CMA-ES variant IPOP-CMA-ES [58] to compare. The population size for CSO, LLSO, and SHADE is also set to Nn following the same scheme [41], [45]. IPOP-CMAES can adaptively increase its population size. Other parameters of the algorithms are the same as their original settings. We remove the exploitation part of TSOL and directly used stopping criterion in line 1 of Algorithm 1 to get a pure ILLSO. No local search method is used. Each of them is executed 30 independent times.

Table IV shows the experimental results. For some large-scale WDNs like S600, sometimes SHADE cannot find feasible solutions within the given computational budget. Since the objective values of infeasible solutions are meaningless, we only used feasible solutions to calculate its mean objective values. From Table IV, we can get the following observations:

- Compared with IPOP-CMA-ES, ILLSO shows dominant superiority. Since the search space is not smooth, CMA-ES cannot approximate the Hessian Matrix of the search space well, which leads to its poor performance.
- Compared with SHADE, ILLSO either has equivalent performance or significantly outperforms it. When the problem scale increases, the performance of SHADE degrades rapidly, which is similar to SADE in Table III.
- Compared with CSO, ILLSO performs worse on rela-

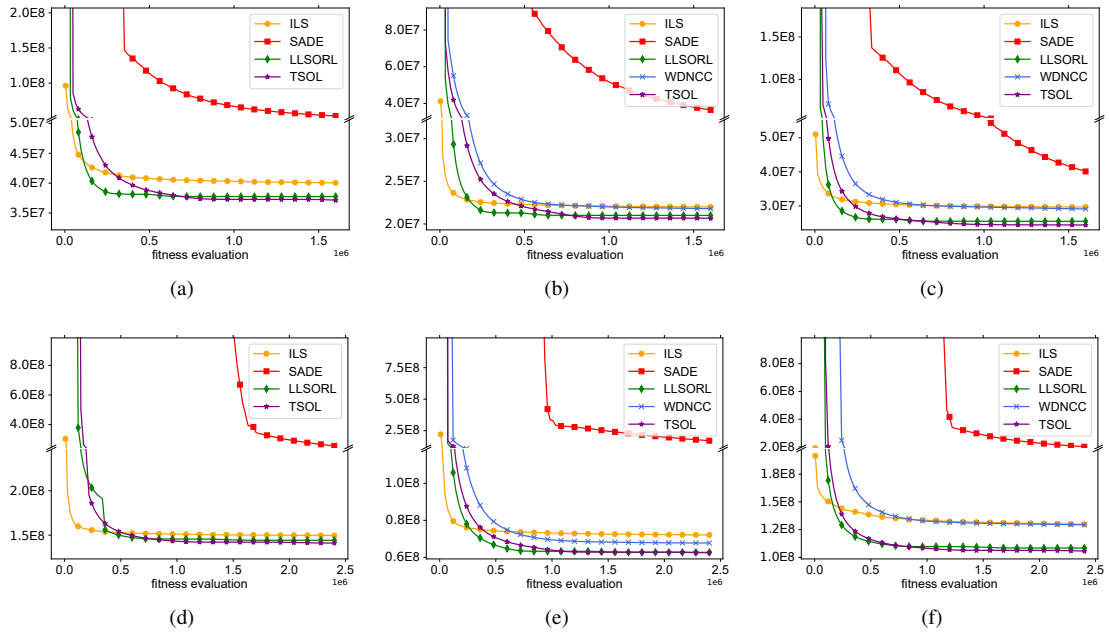


Fig. 7. Convergence curves of ILS, WDNCC, LLSORL, and TSOL. (a) S400, (b) B400, (c) I400, (d) S600, (e) B600, (f) I600.

TABLE IV
COMPARISON BETWEEN IPOP-CMA-ES, SHADE, CSO, LLSO, AND ILLSO.

Inst.	IPOP-CMA-ES	SHADE	CSO	LLSO	ILLSO
S200	5.94E+07(1.92E+07)(+)	6.88E+06(2.24E+05)(=)	**6.60E+06(2.76E+05)(-)**	7.07E+06(4.79E+05)(+)	6.72E+06(3.18E+05)
B200	4.55E+07(1.87E+07)(+)	4.32E+06(1.71E+05)(+)	**3.90E+06(1.11E+05)(-)**	4.25E+06(1.57E+05)(+)	3.95E+06(6.98E+04)
I200	4.12E+07(2.09E+07)(+)	4.31E+06(1.91E+05)(+)	**3.93E+06(1.09E+05)(=)**	4.28E+06(1.24E+05)(+)	3.95E+06(7.31E+04)
S300	1.06E+08(1.17E+07)(+)	2.61E+07(1.14E+06)(+)	2.42E+07(6.31E+05)(=)	2.49E+07(6.62E+05)(+)	**2.38E+07(4.52E+05)**
B300	1.44E+08(5.12E+07)(+)	1.42E+07(3.86E+05)(+)	1.30E+07(2.89E+05)(=)	1.34E+07(4.24E+05)(+)	**1.30E+07(3.39E+05)**
I300	1.40E+08(5.82E+07)(+)	1.39E+07(4.00E+05)(+)	**1.27E+07(1.49E+05)(-)**	1.35E+07(4.92E+05)(+)	1.29E+07(2.87E+05)
S400	1.81E+08(2.74E+07)(+)	4.10E+07(1.20E+06)(+)	3.81E+07(5.82E+05)(+)	3.81E+07(8.73E+05)(+)	**3.74E+07(4.61E+05)**
B400	2.87E+08(7.73E+07)(+)	2.38E+07(5.80E+05)(+)	2.17E+07(3.74E+05)(+)	2.13E+07(5.40E+05)(+)	**2.07E+07(3.01E+05)**
I400	2.68E+08(9.68E+07)(+)	2.92E+07(2.07E+06)(+)	2.54E+07(8.40E+05)(+)	2.61E+07(1.95E+06)(+)	**2.44E+07(4.60E+05)**
S500	2.01E+08(3.28E+07)(+)	1.38E+08(1.35E+07)(+)	1.22E+08(1.21E+06)(+)	1.15E+08(3.91E+06)(+)	**1.11E+08(1.42E+06)**
B500	4.96E+08(1.58E+08)(+)	8.88E+07(2.59E+06)(+)	8.65E+07(2.23E+06)(+)	7.65E+07(3.37E+06)(=)	**7.55E+07(3.25E+06)**
I500	4.83E+08(1.42E+08)(+)	8.39E+07(4.37E+06)(+)	8.41E+07(4.34E+06)(+)	7.58E+07(3.69E+06)(+)	**7.25E+07(2.70E+06)**
S600	2.72E+08(3.89E+07)(+)	1.79E+08(1.16E+07)(+)	1.70E+08(3.72E+06)(+)	1.45E+08(5.90E+06)(=)	**1.42E+08(2.04E+06)**
B600	6.51E+08(1.84E+08)(+)	7.45E+07(2.58E+06)(+)	9.02E+07(1.48E+06)(+)	6.32E+07(6.06E+05)(=)	**6.28E+07(4.97E+05)**
I600	6.81E+08(1.46E+08)(+)	1.32E+08(1.29E+07)(+)	1.33E+08(5.84E+06)(+)	1.10E+08(7.53E+06)(=)	**1.07E+08(7.77E+06)**
Bal	4.53E+06(8.33E+05)(+)	2.14E+06(3.16E+04)(+)	2.02E+06(3.12E+04)(=)	2.04E+06(3.21E+04)(+)	**2.01E+06(3.18E+04)**
w/t/l	16/0/0	15/1/0	9/4/3	12/4/0	–

tively small WDNs but got better results on the WDNs with over 400 pipes. This situation illustrates the effectiveness of ILLSO for large-scale WDN optimization.

- Compared with LLSO, ILLSO achieved significantly better results on the WDNs with less than 500 pipes. Although on S600, B600, and I600, the results of the significance test show that the superiority of ILLSO is not significant, the mean objective values show that ILLSO is still better than LLSO.

Overall, the results show that the proposed ILLSO is generally better to be used to solve WDN optimization problems than the other EC algorithms proposed for LSGO problems.

C. Investigation of the Exploitation Stage

1) Effectiveness of the Exploitation Stage: Comparing the results of ILLSO in Table IV with the results of TSOL in Table III, we can find that the mean objective values of TSOL are smaller than the mean objective values of ILLSO. Although ILLSO can already achieve very promising results, the exploitation stage of TSOL is important to further improve the performance. To show the effectiveness of the exploitation stage, the results of TSOL and ILLSO are compared in Fig. 8 on the 200-scale, 400-scale, and 600-scale WDNs.

From Fig. 8, we can find that:

- On some WDNs like B200 and S400, the exploitation stage of TSOL has successfully decreased the deviation of the algorithm's performance by improving the qualities of the relatively bad solutions. For the solutions that are already good enough, the effect is marginal.
- On B600, we can see that the exploitation stage has improved the qualities of most solutions. Thus, the deviation does not decrease but the overall performance is better.
- On I600, the effectiveness of the exploitation stage is not remarkable. There is only a little improvement.

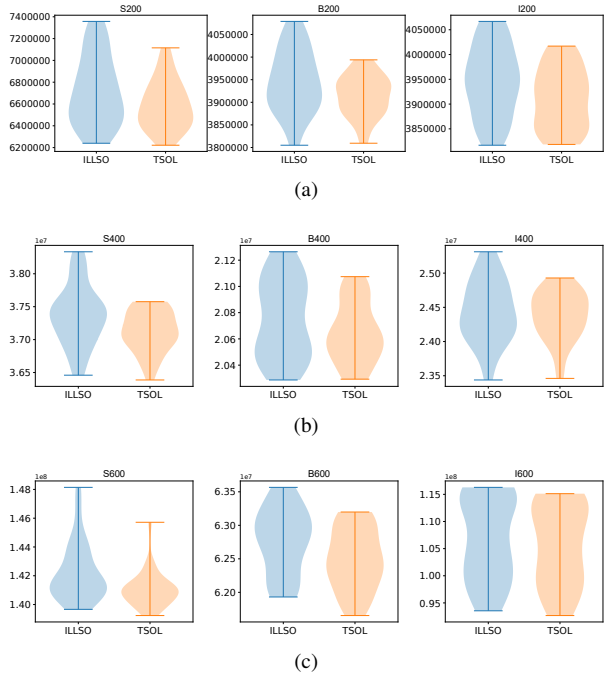


Fig. 8. Comparison between the results of TSOL and ILLSO. (a) S200, B200, and I200; (b) S400, B400, and I400; (c) S600, B600, and I600.

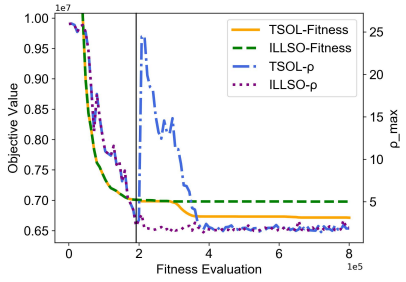


Fig. 9. Convergence curves of the fitness value and $\rho^g_{\max}$ of TSOL and ILLSO on S200. The black vertical line shows the starting point of the exploitation stage.

Generally, we can see that setting an exploitation stage is useful for obtaining better solutions.

2) Exploitation Criterion: The timing when TSOL goes to the exploitation stage is checked to verify the validity of the exploitation criterion. S200 is selected as the representative instance to conduct the investigation. Basically, the convergence behavior of TSOL has the same pattern on different instances, and for a relatively small WDN, the given number of fitness evaluations is usually sufficient to let the algorithm fully converge. The convergence curves of ILLSO without the exploitation stage and the convergence curves of TSOL with the exploitation stage are drawn. Also, the $\rho^g_{\max}$ of ILLSO and the $\rho^g_{\max}$ of TSOL are drawn to show when the exploitation stage starts. Since the average performance of 30 runs would flatten the fluctuation of $\rho^g_{\max}$ and the convergence curve of the objective value, we choose a single run where there is a clear improvement of the objective value in the exploitation stage to display. Fig. 9 shows the detailed convergence behavior.

From Fig. 9, we can get the following observations:

- It is clear that when the decreasing trend of the fitness value just stopped, the exploitation stage of TSOL begins. Under this circumstance, there is no computing resource wasted in the exploration stage. The validity of the exploitation criterion is thus verified.
- The objective curves show the effectiveness of the exploitation stage again that the objective value is further decreased after the exploitation stage starts.
- Also, from the convergence curves of $\rho^g_{\max}$, we can see the fluctuations clearly after the algorithm converges. This has verified our judgment made in Section IV.C that $\rho^g_{\max}$ is a good convergence indicator for the WDN optimization problem.

TABLE V
THE AVERAGED AMOUNT OF MONEY SAVING BY USING LOCAL SEARCH.

Inst.	LS in Exploration	LS in Exploitation
S200	5265.01	891.29
B200	7912.85	578.37
I200	4017.96	796.79
S300	2554.74	537.80
B300	3774.70	834.98
I300	3163.45	540.33
S400	4585.24	40822.76
B400	7300.83	26924.51
I400	7215.01	34985.46
S500	6808.24	359464.97
B500	7263.85	76128.09
I500	7165.48	140177.36
S600	15938.17	19092.94
B600	14390.36	5264.52
I600	14216.19	172789.46
Balerma	82.38	15.37

For synthetic WDNs, the unit of money saving is CNY. For Balerma, the unit of money saving is EUR.

An interesting phenomenon showed in Fig. 9 is that $\rho^g_{\max}$ increases rapidly after the exploitation stage starts. From Section IV.B, we know that the individuals of TSOL are actually re-initialized in a small exploitation region after the algorithm goes to the exploitation region. The reason for the growth is that even in the small region, the search space still has the multi-modal characteristic that the individuals do not have a consistent moving direction to a specific optimum in the beginning. After checking the values of $(\rho^g_1, \ldots, \rho^g_i, \ldots, \rho^g_{Np})$, we find that only a few of them increase that rapidly which is different from the situation of the first initialization where very ρ^g_i is very big. It means that after going into the exploitation stage, TSOL focus on only some of the variables, instead of re-exploring all of them, which is inline with our expectation.

D. Effectiveness of Local Search in TSOL

As Fig. 2 shows, in TSOL, we have applied the local search algorithm twice, once at the end of the exploration stage and once at the end of the exploitation stage. We check each time how much money was saved in average of 30 runs to examine the effectiveness of the local search method. The results are shown in Table V.

From Table V, we can see that the local search algorithm has successfully improved the qualities of the solutions by decreasing the sizes of some pipes on all WDNs. Generally, the improvement increases with the scale of the WDN. Since it is harder for a population-based meta-heuristic algorithm to

optimize a larger WDN than a smaller WDN, there will be more room for local search to show its effectiveness.

The effectiveness of the local search algorithm on the Balerma WDN is very little since there are only 10 commercially available pipe types that can be chosen for Balerma. Compared with the 26 pipe types for the synthetic WDNs, the search space of Balerma is much smaller than the synthetic WDNs. Under such circumstances, the solutions found by ILLSO are already very close to the local optima, leaving a very small room for the local search algorithm to optimize.

It should be noted that in TSOL, the final local search procedure is executed after the stopping criterion is met. Thus, in the above experiments, TSOL used a little more computing sources than the other algorithms. According to the experimental results, it used several hundred more evaluations in the final local search procedure. Compared with the $Nn \cdot 4000$ times of fitness evaluations, these extra evaluations are small enough to be ignored, and the extra execution time of this local search procedure is usually smaller than five seconds.

VII. Conclusions and Future Work

The goal of this paper was to solve large-scale WDN optimization problems effectively. This goal has been successfully achieved by proposing a new algorithm called TSOL. The newly proposed ILLSO that is used as the optimizer in TSOL, has been proven more effective than several representative EC algorithms that were proposed for LSGO, including LLSO, CSO, and SHADE. By setting an exploitation stage and applying new local search algorithms, TSOL has further improved the performance of ILLSO. The comparisons with the state-of-the-art algorithms for WDN optimization, including ILS, SADE, WDNCC, and LLSORL, have demonstrated the significant advantage of TSOL.

Although TSOL has achieved very good results in this paper, there are still several directions to continue the research. First, from the perspective of the problem, extending TSOL to multi-objective context that can be applied to more practical WDNs with more objectives such as network resilience, operation cost, and security is a good research direction to follow. Second, from the perspective of the optimization algorithm, every EC algorithm has its upper limit of capability that when the scale of the problem reaches a level, it will lose its functionality. Developing divide-and-conquer methods that can be applied to different types of WDNs, even single-source WDNs, is a promising research direction.

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, 2015.

[2] S. V. Lall, J. V. Henderson, and A. J. Venables, *Africa's cities: Opening doors to the world*. The World Bank, 2017.

[3] B. Coelho and A. Andrade-Campos, "Efficiency achievement in water supply systems—a review," *Renew. Sust. Energ. Rev.*, vol. 30, pp. 59–84, 2014.

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

[5] D. Yates, A. Templeman, and T. Boffey, "The computational complexity of the problem of determining least capital cost designs for water supply networks," *Eng. Optimiz.*, vol. 7, no. 2, pp. 143–155, 1984.

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

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

[8] M. d. C. Cunha and J. Sousa, "Water distribution network design optimization: simulated annealing approach," *J. Water Resour. Plan. Manag.*, vol. 125, no. 4, pp. 215–221, 1999.

[9] A. Fanni, S. Liberatore, G. Sechi, M. Soro, and P. Zuddas, "Optimization of water distribution systems by a tabu search metaheuristic," in *Computing Tools for Modeling, Optimization and Simulation*. Springer, Boston, MA, 2000, pp. 279–298.

[10] 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, 2016.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'1995*, vol. 4. IEEE, 1995, pp. 1942–1948.

[12] C. Suribabu and T. Neelakantan, "Design of water distribution networks using particle swarm optimization," *Urban Water J.*, vol. 3, no. 2, pp. 111–120, 2006.

[13] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimiz.*, vol. 11, no. 4, pp. 341–359, 1997.

[14] 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, 2010.

[15] L. Davis, *Handbook of genetic algorithms*. Van Nostrand Reinhold New York, 1991.

[16] 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, 1994.

[17] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. M.*, vol. 1, no. 4, pp. 28–39, 2006.

[18] H. R. Maier, A. R. Simpson, A. C. Zecchin, W. K. Foong, K. Y. Phang, H. Y. Seah, and C. L. Tan, "Ant colony optimization for design of water distribution systems," *J. Water Resour. Plan. Manag.*, vol. 129, no. 3, pp. 200–209, 2003.

[19] W.-N. Chen, Y.-H. Jia, F. Zhao, X.-N. Luo, X.-D. Jia, and J. Zhang, "A cooperative co-evolutionary approach to large-scale multisource water distribution network optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 842–857, 2019.

[20] 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, 2013.

[21] Q. Yang, W.-N. Chen, T. Gu, H. Zhang, J. D. Deng, Y. Li, and J. Zhang, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2896–2910, 2016.

[22] D. Molina, A. LaTorre, and F. Herrera, "Shade with iterative local search for large-scale global optimization," in *Proc. CEC'2018*. IEEE, 2018, pp. 1–8.

[23] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, 2014.

[24] Q. Yang, W.-N. Chen, J. Da Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 578–594, 2017.

[25] F.-F. Wei, W.-N. Chen, Q. Yang, J. Deng, X.-N. Luo, H. Jin, and J. Zhang, "A classifier-assisted level-based learning swarm optimizer for expensive optimization," *IEEE Trans. Evol. Comput.*, 2020.

[26] Y.-H. Jia, Y. Mei, and M. Zhang, "A memetic level-based learning swarm optimizer for large-scale water distribution network optimization," in *Proc. GECCO'2020*, 2020, pp. 1107–1115.

[27] A. C. Twort, D. D. Ratnayaka, and M. J. Brandt, *Water supply*. Elsevier, 2000.

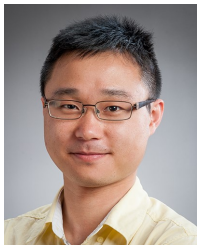
[28] F. Zheng, A. C. Zecchin, H. R. Maier, and A. R. Simpson, "Comparison of the searching behavior of nsga-ii, samode, and borg moeas applied to water distribution system design problems," *J. Water Resour. Plan. Manag.*, vol. 142, no. 7, p. 04016017, 2016.

[29] D. G. Eliades and M. M. Polycarpou, "A fault diagnosis and security framework for water systems," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 6, pp. 1254–1265, 2009.

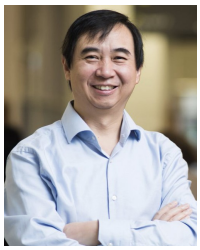
[30] L. A. Rossman *et al.*, "Epanet 2: users manual," 2000.

[31] G. Loganathan, J. Greene, and T. Ahn, "Design heuristic for globally minimum cost water-distribution systems," *J. Water Resour. Plan. Manag.*, vol. 121, no. 2, pp. 182–192, 1995.
[32] M. da Conceicao Cunha and L. Ribeiro, "Tabu search algorithms for water network optimization," *Eur. J. Oper. Res.*, vol. 157, no. 3, pp. 746–758, 2004.
[33] Y.-H. Sung, M.-D. Lin, Y.-H. Lin, and Y.-L. Liu, "Tabu search solution of water distribution network optimization," *J. Environ. Eng. Manag.*, vol. 17, no. 3, p. 177, 2007.
[34] J. Reca, J. Martínez, C. Gil, and R. Baños, "Application of several meta-heuristic techniques to the optimization of real looped water distribution networks," *Water Res. Manag.*, vol. 22, no. 10, pp. 1367–1379, 2008.
[35] A. De Corte and K. Sörensen, "An iterated local search algorithm for multi-period water distribution network design optimization," *Water*, vol. 8, no. 8, p. 359, 2016.
[36] J. Nicklow, P. Reed, D. Savic, T. Dessalegne, L. Harrell, A. Chan-Hilton, M. Karamouz, B. Minsker, A. Ostfeld, A. Singh *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, 2010.
[37] W. Bi, G. C. Dandy, and H. R. Maier, "Improved genetic algorithm optimization of water distribution system design by incorporating domain knowledge," *Environ. Modell. Softw.*, vol. 69, pp. 370–381, 2015.
[38] F. Zheng, A. C. Zecchin, A. R. Simpson, and M. F. Lambert, "Non-crossover dither creeping mutation-based genetic algorithm for pipe network optimization," *J. Water Resour. Plan. Manag.*, vol. 140, no. 4, pp. 553–557, 2014.
[39] J. Reca, J. Martínez, and R. López, "A hybrid water distribution networks design optimization method based on a search space reduction approach and a genetic algorithm," *Water*, vol. 9, no. 11, p. 845, 2017.
[40] K. Vairavamoorthy and Y. Shen, "Least cost design of water distribution network using particle swarm optimisation," in *Hydroinformatics: (In 2 Volumes, with CD-ROM)*. World Scientific, 2004, pp. 834–841.
[41] 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, 2008.
[42] R. Ezzeldin, B. Djebedjian, and T. Saafan, "Integer discrete particle swarm optimization of water distribution networks," *J. Pipeline Syst. Eng. Pract.*, vol. 5, no. 1, p. 04013013, 2014.
[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, p. 04016005, 2016.
[44] C. Suribabu, "Differential evolution algorithm for optimal design of water distribution networks," *J. Hydroinforma.*, vol. 12, no. 1, pp. 66–82, 2010.
[45] 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, 2013.
[46] A. Sedki and D. Ouazar, "Hybrid particle swarm optimization and differential evolution for optimal design of water distribution systems," *Adv. Eng. Inform.*, vol. 26, no. 3, pp. 582–591, 2012.
[47] 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, 2007.
[48] F. Zheng, A. C. Zecchin, J. P. Newman, H. R. Maier, and G. C. Dandy, "An adaptive convergence-trajectory controlled ant colony optimization algorithm with application to water distribution system design problems," *IEEE Transs Evol. Comput.*, vol. 21, no. 5, pp. 773–791, 2017.
[49] 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, 2003.
[50] D. Mora-Melia, P. L. Iglesias-Rey, F. J. Martínez-Solano, and P. Muñoz-Velasco, "The efficiency of setting parameters in a modified shuffled frog leaping algorithm applied to optimizing water distribution networks," *Water*, vol. 8, no. 5, p. 182, 2016.
[51] Z. W. Geem, "Optimal cost design of water distribution networks using harmony search," *Eng. Optimiz.*, vol. 38, no. 03, pp. 259–277, 2006.
[52] Y. H. Choi, H. M. Lee, D. G. Yoo, and J. H. Kim, "Self-adaptive multi-objective harmony search for optimal design of water distribution networks," *Eng. Optimiz.*, vol. 49, no. 11, pp. 1957–1977, 2017.
[53] Y. Sun, X. Li, A. Ernst, and M. N. Omidvar, "Decomposition for large-scale optimization problems with overlapping components," in *Proc. CEC'2019*. IEEE, 2019, pp. 326–333.
[54] Y.-H. Jia, Y. Mei, and M. Zhang, "Contribution-based cooperative co-evolution for non-separable large-scale problems with overlapping subcomponents," *IEEE Trans. Cybern.*, 2020, early access.
[55] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Method. Appl. Mech. Eng.*, vol. 186, no. 2-4, pp. 311–338, 2000.
[56] F. Zheng, A. R. Simpson, and A. C. Zecchin, "A combined nlp-differential evolution algorithm approach for the optimization of looped water distribution systems," *Water Resour. Res.*, vol. 47, no. 8, 2011.
[57] J. Reca and J. Martínez, "Genetic algorithms for the design of looped irrigation water distribution networks," *Water Resour. Res.*, vol. 42, no. 5, 2006.
[58] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in *Proc. CEC2005*, vol. 2. IEEE, 2005, pp. 1769–1776.

Ya-Hui Jia (M'20) received the B.Eng. and Eng.D. degrees from Sun Yat-sen University, China, in 2013 and 2019, respectively. He is currently a postdoctoral research fellow at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. His research interests include evolutionary computation, combinatorial optimization, software engineering, cloud computing, and intelligent transportation.



Yi Mei (M'09-SM'18) received the BSc and PhD degrees from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively. He is currently a Senior Lecturer at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. His research interests include evolutionary scheduling and combinatorial optimisation, machine learning, genetic programming, and hyper-heuristics. He has over 100 fully referred publications, including the top journals in EC and Operations Research such as IEEE TEVC, IEEE TCYB, Evolutionary Computation Journal, European Journal of Operational Research, ACM Transactions on Mathematical Software. He serves as a Vice-Chair of the IEEE CIS Emergent Technologies Technical Committee, and a member of Intelligent Systems Applications Technical Committee. He is an Editorial Board Member/Associate Editor of three International Journals, and a guest editor of a special issue of the Genetic Programming Evolvable Machine journal. He serves as a reviewer of over 30 international journals.



Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively. He is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. Prof. Zhang is a Fellow of Royal Society of New Zealand, a Fellow of IEEE, and an IEEE Distinguished Lecturer. He was the chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, the chair for the IEEE CIS Emergent Technologies Technical Committee, the chair of Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a vice-chair of the Task Force on Evolutionary Computer Vision and Image Processing, and the founding chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a committee member of the IEEE NZ Central Section.