Supplementary Material of "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*

I. Algorithm Analysis

A. Time Complexity

The time complexity of TSOL can be divided into two parts $\mathrm{O}(X+Y)$ where $\mathrm{O}(X)$ and $\mathrm{O}(Y)$ represent the time complexities of generating and evaluating solutions, respectively. Since evaluating a solution is mainly related to the simulation rather than the optimization algorithm, we will focus on analyzing $\mathrm{O}(X)$ here instead of $\mathrm{O}(Y)$.

TSOL contains an EC algorithm ILLSO and a local search algorithm. Regarding ILLSO, suppose there are M individuals and G generations. In each generation, there are three operations: 1) ranking the individuals, 2) calculating the central point of every level, and 3) generating new solutions. The time complexity of ranking is $\mathrm{O}(M \cdot \log(M))$ where M is the population size. The time complexity of calculating the central point of every level is $\mathrm{O}(M \cdot Np)$, where Np is the number of pipes. The time complexity of generating solutions in each generation is also $\mathrm{O}(M \cdot Np)$. Adding them together, the time complexity of each generation is $\mathrm{O}(M \cdot Np + M \cdot \log(M))$. Thus, the overall time complexity of ILLSO is $\mathrm{O}(M \cdot G \cdot (Np + \log(M)))$. $M \cdot G$ is approximately equal to the number of fitness evaluations. Denoting the total number of fitness evaluations as $\hat{F}$, the time complexity of ILLSO can be also presented as $\mathrm{O}(\hat{F} \cdot (Np + \log(M)))$.

Regarding the local search algorithm, BFLS or DFLS, their time complexities highly depends on the quality of $\mathbf{x}_{\mathrm{gb}}$. Theoretically, the time complexity of BFLS and DFLS can be presented as $\mathrm{O}(\hat{Np} \cdot T)$, where $\hat{Np}$ represents the number of pipes whose sizes can be still reduced in $\mathbf{x}_{\mathrm{gb}}$ and T represents how much these pipes can be reduced on average. In TSOL, $\mathbf{x}_{\mathrm{gb}}$ has been well-optimized by ILLSO before it is refined by the local search method. According to the empirical observation, the time complexity of the local search algorithm in TSOL is usually smaller than $\mathrm{O}(Np)$, which is much smaller than the time complexity of ILLSO so that it can be ignored. Adding the time complexities of generating and evaluating solutions together, the time complexity of TSOL is $\mathrm{O}(\hat{F} \cdot (Np + \log(M)) + Y)$.

B. TSOL vs LLSORL

In our previous work [1], we have acknowledged the superiority of the newly proposed PSO variants, i.e. CSO and LLSO, against the traditional EC algorithms, and proposed LLSORL that applied LLSO to the WDN optimization problem. To compensate for the lack of exploitation ability of LLSO, a random-descent heuristic was proposed to refine the solution when LLSO converged. Also, we found that setting a exploitation stage was helpful to improve the performance.

Although TSOL shares a similar structure with LLSORL that an EC algorithm for large-scale global optimization cooperates with a heuristic algorithm, the core components are completely updated. In particular, the difference between TSOL and our previous work LLSORL [1] is embodied in three aspects:

- A new EC algorithm ILLSO is proposed in TSOL. LLSORL [1] directly used LLSO without any change. In TSOL, we have proposed a new EC algorithm ILLSO. The velocity updating rule is modified so that ILLSO has better exploratio ability than LLSO. The advantages of ILLSO compared with CSO and LLSO have been analyzed in Section IV.B of the paper in detail. Experimental results in Section VI.B also show that ILLSO is significantly better than LLSO and CSO.
- Two new heuristic methods to further refine the solutions are proposed in TSOL. In the previous work [1], we proposed a random-descent heuristic that the order of the variables to be decreased is randomly shuffled each time. In TSOL, we proposed BFLS and DFLS that used a fixed order according to the saving of each pipe. In most cases, guided by the heuristic information, the performance of BFLS and DFLS is better than the random-descent heuristic, and also more stable than the random-descent since no randomness is adopted.
- A new exploitation criterion is proposed TSOL. In the previous work [1], we switch to the exploitation stage if LLSO did not find any better solution for a fixed number of successive generations. However, we found that the parameter of the fixed number of successive generations was both algorithm-dependent and instance-dependent. Due to the stochastic characteristic of EC algorithms, the previous criterion may often cause the situation that LLSO is not converged when entering the exploitation stage. The new criterion proposed in TSOL is more adaptive than the previous one in [1]. It considers the value range of the variables and the behaviors of the individuals in different optimization stages. It measures whether the algorithm has converged according to the distance between individuals which is more reasonable. It is more suitable to be applied to different algorithms and different WDN instances.

TABLE A
SUCCESS RATE OF SHADE, IPOP-CMA-ES, CSO, LLSO, AND ILLSO IN TERMS OF FINDING FEASIBLE SOLUTIONS

instance	CMA-ES	SHADE	CSO	LLSO	ILLSO
S200	1.0	1.0	1.0	1.0	1.0
B200	1.0	1.0	1.0	1.0	1.0
I200	1.0	1.0	1.0	1.0	1.0
S300	1.0	1.0	1.0	1.0	1.0
B300	0.93	1.0	1.0	1.0	1.0
I300	0.97	1.0	1.0	1.0	1.0
S400	1.0	1.0	1.0	1.0	1.0
B400	1.0	1.0	1.0	1.0	1.0
I400	1.0	1.0	1.0	1.0	1.0
S500	1.0	0.77	1.0	1.0	1.0
B500	0.87	1.0	1.0	1.0	1.0
I500	0.9	1.0	1.0	1.0	1.0
S600	1.0	0.5	1.0	1.0	1.0
B600	0.83	1.0	1.0	1.0	1.0
I600	0.8	0.77	1.0	1.0	1.0
Bal.	1.0	1.0	1.0	1.0	1.0

II. EXPERIMENTAL RESULT ANALYSIS

This section provides more analyses about the experimental results including 1) the success rate of each algorithm finding feasible solutions, 2) how long each algorithm takes to find a feasible solution, 3) the success rate of each algorithm outperforming the heuristic methods, and 4) how long each algorithm takes to outperform the heuristic methods.

A. Feasibility

1) Success Rate of Finding Feasible Solutions: Here, we show the success rate of each algorithm of finding feasible solutions. The success rate of the algorithms in Table III of the paper can always find feasible solutions, including ILS, SADE, WDNCC, LLSORL, and TSOL. Thus, only the success rate of the algorithms in Table IV of the paper are shown in Table A, including SHADE, IPOP-CMA-ES, CSO, LLSO, and ILLSO.

From Table A, it is clear that the PSO variants, i.e. CSO, LLSO, and ILLSO, have shown better performance than SHADE and CMA-ES. Starting from randomly initialized solutions, SHADE and CMA-ES sometimes cannot find feasible solutions. Although we know that it is easy to get a feasible solution by setting all pipes to the largest type, the results reveal that the exploration ability of SHADE and CMA-ES for WDN optimization is poor.

2) Computational Load to Find Feasible Solutions: How many fitness evaluations each population-based meta-heuristic algorithm costs to find a feasible solution is reported in Table B. LLSORL shares the same results with LLSO since LLSORL uses LLSO as the optimizer. Also, TSOL shares the same results with ILLSO due to the same reason.

From the results, we can see that LLSORL and LLSO are the quickest algorithms to find feasible solutions among these population-based meta-heuristic algorithms since they have the fastest converging speed. TSOL and ILLSO are in the second place. CSO is slower than LLSO and ILLSO. This phenomenon indicates that the rank of converging speed of these three algorithms is LLSO $>$ ILLSO $>$ CSO. Referring the final objective values shown in Table IV of the paper, we think that ILLSO has found a better balance between exploration and exploitation than the other two algorithms for WDN optimization so that it is more effective.

B. Effectiveness

1) Success Rate of Finding Better Solutions Than Heuristic: We take the best results of the two heuristics, i.e. BFLS and DFLS, as the baseline to check whether the other algorithms have found better solutions than the heuristic methods. On a specific instance, if the performance of BFLS is better than DFLS, we take BFLS as the baseline. Otherwise, DFLS is taken as the baseline. The results are shown in Table C. From the results we can get following observations:

- For a large-scale WDN optimization problem, using CMA-ES is clearly not a good idea since it is even worse than heuristic methods.
- Two DE-based algorithms, i.e. SADE and SHADE, are effective when the scale of the network is small. However, when the scale of the WDN grows, their effectiveness degrades rapidly which is similar to the phenomenon that scholars have found from other applications.
- CSO, LLSO, and ILLSO perform very well that they have found better solutions than the two heuristics every time, which demonstrates their effectiveness.

2) Computational Load to Outperform Heuristics: The results of how long each algorithm takes to outperform the two heuristics are reported in Table D. Generally, the pattern shown in the results of Table D is similar to the patter shown in the results of Table B. LLSO is still the quickest algorithm that finds better solutions than the two heuristic algorithms. ILLSO on S200-I300 WDNs is the slower than CSO, but it surpasses CSO when the scale of the network grows to 400. For the other algorithms, only CMA-ES struggles to outperform the heuristic methods. Other algorithms all have the capability to find better solutions than the heuristic methods. Overall, from the four tables, we can achieve the following conclusions:

- Among CSO, LLSO, and ILLSO, the rank of the converging speed is LLSO $>$ ILLSO $>$ CSO. These three algorithms can always outperform heuristic methods. Thus, LLSORL and TSOL are also efficient and effective.
- Two DE variants, SADE and SHADE may work well on relatively small networks. However, when the scale of the network grows larger and larger, their performance degrades faster than the PSO variants for WDN optimization.
- CMA-ES is not suitable to be applied to WDN optimization. Theoretically, CMA-ES optimizes the problem by approximate the Hessian Matrix [2], but the search space of the WDN optimization problem is not differentiable. Thus, although its performance for function optimization is good, it does not fit WDN optimization well.

Basically, the results have further demonstrated the advantages of ILLSO and TSOL.

REFERENCES

[1] 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.

[2] N. Hansen, "The cma evolution strategy: a comparing review," *Towards a new evolutionary computation*, pp. 75–102, 2006.

TABLE B
THE NUMBER OF FITNESS EVALUATIONS EACH ALGORITHM TAKES TO FIND A FEASIBLE SOLUTION.

instance	SADE	WDNCC	LLSORL	TSOL	CMA-ES	SHADE	CSO	LLSO	ILLSO
S200	4.6E4	–	**5.5E3**	7.6E3	2.0E4	1.4E4	9.3E3	**5.5E3**	7.6E3
B200	3.4E4	1.1E4	**3.3E3**	4.3E3	8.9E3	6.3E3	4.4E3	**3.3E3**	4.3E3
I200	3.5E4	1.4E4	**3.6E3**	5.0E3	9.0E3	7.6E3	5.5E3	**3.6E3**	5.0E3
S300	1.7E5	–	**1.9E4**	2.6E4	1.1E5	6.1E4	3.4E4	**1.9E4**	2.6E4
B300	1.4E5	3.3E4	**1.3E4**	1.9E4	3.9E4	3.0E4	2.2E4	**1.3E4**	1.9E4
I300	1.4E5	4.2E4	**1.3E4**	1.8E4	3.7E4	3.1E4	2.3E4	**1.3E4**	1.8E4
S400	3.3E5	–	**2.7E4**	3.7E4	1.1E5	7.9E4	5.2E4	**2.7E4**	3.7E4
B400	2.6E5	3.6E4	**1.7E4**	2.4E4	4.8E4	3.9E4	3.1E4	**1.7E4**	2.4E4
I400	3.0E5	5.7E4	**2.3E4**	3.1E4	4.7E4	5.8E4	4.2E4	**2.3E4**	3.1E4
S500	7.3E5	–	**6.6E4**	9.0E4	2.0E5	3.5E5	1.3E5	**6.6E4**	9.0E4
B500	7.0E5	1.1E5	**1.9E4**	6.3E4	7.2E4	2.0E5	9.5E4	**1.9E4**	6.3E4
I500	7.5E5	1.2E5	**5.1E4**	6.6E4	6.3E4	2.0E5	1.0E5	**5.1E4**	6.6E4
S600	1.5E6	–	**1.2E5**	1.4E5	2.7E5	5.2E5	2.2E5	**1.2E5**	1.4E5
B600	9.2E5	9.4E4	**5.3E4**	6.6E4	1.1E5	1.8E5	1.1E5	**5.3E4**	6.6E4
I600	1.1E6	1.8E5	**7.6E4**	9.6E4	1.1E5	3.5E5	1.6E5	**7.6E4**	9.6E4
Bal.	2.4E4	1.2E4	**5.0E3**	6.0E3	8.7E2	1.2E4	7.9E3	**5.0E3**	6.0E3

TABLE C
SUCCESS RATE OF FINDING BETTER SOLUTIONS THAN BFLS AND DFLS.

instance	SADE	WDNCC	LLSORL	TSOL	CMA-ES	SHADE	CSO	LLSO	ILLSO
S200	1	–	1	1	0	1	1	1	1
B200	1	1	1	1	0	1	1	1	1
I200	1	1	1	1	0	1	1	1	1
S300	1	–	1	1	0	1	1	1	1
B300	1	1	1	1	0	1	1	1	1
I300	1	1	1	1	0	1	1	1	1
S400	1	–	1	1	0.03	1	1	1	1
B400	1	1	1	1	0	1	1	1	1
I400	1	1	1	1	0	1	1	1	1
S500	1	–	1	1	0.6	0.77	1	1	1
B500	0.93	1	1	1	0	1	1	1	1
I500	0.03	1	1	1	0	1	1	1	1
S600	0.9	–	1	1	0.37	0.5	1	1	1
B600	1	1	1	1	0	1	1	1	1
I600	0	0.97	1	1	0	0.77	1	1	1
Bal.	1	1	1	1	0	1	1	1	1

TABLE D
THE NUMBER OF FITNESS EVALUATIONS EACH ALGORITHM TAKES TO FIND BETTER SOLUTIONS THAN BFLS AND DFLS.

instance	SADE	WDNCC	LLSORL	TSOL	CMA-ES	SHADE	CSO	LLSO	ILLSO
S200	1.9E5	–	**1.4E4**	2.4E4	NA	2.4E4	2.2E4	**1.4E4**	2.4E4
B200	3.6E5	8.8E4	**2.5E4**	4.5E4	NA	4.9E4	4.1E4	**2.5E4**	4.5E4
I200	2.9E5	8.6E4	**2.1E4**	3.7E4	NA	3.6E4	3.3E4	**2.1E4**	3.7E4
S300	4.4E5	–	**2.7E4**	4.7E4	NA	6.8E4	4.9E4	**2.7E4**	4.7E4
B300	7.5E5	1.1E5	**3.8E4**	7.5E4	NA	6.1E4	7.4E4	**3.8E4**	7.5E4
I300	6.5E5	1.1E5	**3.4E4**	6.5E4	NA	5.1E4	6.3E4	**3.4E4**	6.5E4
S400	6.3E5	–	**2.7E4**	4.2E4	3.6E5	7.9E4	5.2E4	**2.7E4**	4.2E4
B400	1.2E6	1.1E5	**4.3E4**	7.6E4	NA	5.3E4	9.2E4	**4.3E4**	7.6E4
I400	1.2E6	1.4E5	**4.8E4**	8.0E4	NA	7.3E4	9.4E4	**4.8E4**	8.0E4
S500	1.2E6	–	**6.7E4**	9.4E4	4.4E5	3.5E5	1.3E5	**6.7E4**	9.4E4
B500	1.9E6	1.9E5	**6.7E4**	9.8E4	NA	2.0E5	1.4E5	**6.7E4**	9.8E4
I500	1.8E6	1.6E5	**5.9E4**	8.5E4	NA	2.0E5	1.3E5	**5.9E4**	8.5E4
S600	2.3E6	–	**1.2E5**	1.4E5	7.7E5	5.2E5	2.3E5	**1.2E5**	1.4E5
B600	1.9E6	1.0E5	**5.3E4**	6.6E4	NA	1.8E5	1.1E5	**5.3E4**	6.6E4
I600	NA	2.8E5	**1.0E5**	1.4E5	NA	4.0E5	2.5E5	**1.0E5**	1.4E5
Bal.	2.0E5	5.1E4	**1.9E4**	2.4E4	NA	4.1E4	2.9E4	**1.9E4**	2.4E4