Pertanika J. Sci. & Technol. 25 (S): 199 - 210 (2017)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Water Flow-Like Algorithm Improvement Using K-Opt Local Search

Wu Diyi, Zulaiha Ali Othman*, Suhaila Zainudin and Ayman Srour

Centre of Artificial Intelligence Techonology (CAIT) Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Selangor, 43600 Malaysia

ABSTRACT

The water flow-like algorithm (WFA) is a relatively new metaheuristic algorithm, which has shown good solution for the Travelling Salesman Problem (TSP) and is comparable to state of the art results. The basic WFA for TSP uses a 2-opt searching method to decide a water flow splitting decision. Previous algorithms, such as the Ant Colony System for the TSP, has shown that using k-opt (k>2) improves the solution, but increases its complexity exponentially. Therefore, this paper aims to present the performance of the WFA-TSP using 3-opt and 4-opt, respectively, compare them with the basic WFA-TSP using 2-opt and the state of the art algorithms. The algorithms are evaluated using 16 benchmarks TSP datasets. The experimental results show that the proposed WFA-TSP-4opt outperforms in solution quality compare with others, due to its capacity of more exploration and less convergence.

Keywords: Combinatorial optimization, Nature-inspired metaheuristics, Traveling Salesman Problem, Water flow-liked algorithm

INTRODUCTION

The Travelling Salesman Problem (TSP) is one of the most widely studied problems in combinatorial optimization (Martin & Otto, 1996). The TSP aims to find the shortest trip from a starting point, visiting all cities in a route exactly once and returning to the start point. The total number of possible routes is (n-1)!/2 for n number of cities. For example, if the TSP is applied on a large dataset with thousands of cities, the number of possible solutions is huge (2.01e+2567 for 1000 cities). The Water Flow-like Algorithm (WFA) is a dynamic-based solution to solve object grouping problems from the start (Yang & Wang, 2007). The number of solution agents deployed dynamically changes throughout

ARTICLE INFO

Article history:
Received: 15 August 2016
Accepted: 18 May 2017

E-mail addresses:
zzwudiyi@126.com (Wu Diyi),
zao@ukm.edu.my (Zulaiha Ali Othman),
suhaila@ukm.edu.my (Suhaila Zainudin),
aymansrour@gmail.com (Ayman Srour)
*Corresponding Author

ISSN: 0128-7680 © 2017 Universiti Putra Malaysia Press.

the water flow, splitting and merging. (Srour, Othman, & Hamdan, 2014) applied the WFA to solve the TSP and demonstrated that the solution outperformed the Ant Colony Optimization (ACO) in terms of solution quality and computation time (Srour et al., 2014).

Local search with k-change neighbourhoods, k-opt, is one of the most effective methods for improving the metaheuristic algorithm solution quality (Helsgaun, 2006). This research aims to improve the WFA-TSP by using 3-opt (WFA-TSP-3opt) and 4-opt (WFA-TSP-4opt). The performance of the algorithms will be compared in terms of solution quality and computation time.

This paper will be organized as follows: section 2 review past related work in this domain, while section 3 presents the proposed WFA using k-opt for the TSP. Section 4 discusses the experiment setups, followed by the experiment results divided into solution quality, computation time and proposed algorithm behaviour discussion. The last section concludes the results and future works.

RELATED WORKS

Two popular methods use for solving the TSP are known as the exact method and approximate method. The exact method aims to obtain optimal solutions and guarantees their optimality, whereas the approximate method generates "good" quality solutions (Martin & Otto, 1996). Approximate algorithms are usually faster than other approaches, but produces worse solutions in most cases (Mo, 2010).

Metaheuristic is an approach that produces efficient, near-optimal solutions without any guarantee of finding the global optimal or even bounded solution (Talbi, 2009). A metaheuristic algorithm aims for a balance between diversification (exploration) and intensification (exploitation). The term diversification refers to exploring a larger search space. Intensification refers to the exploitation of the accumulated search experience (Talbi, 2009). The most popular classification is known as the single solution based metaheuristic and population based metaheuristic solution (Talbi, 2009). A single solution based metaheuristic, such as Simulated Annealing (Kirkpatrick, Gelatt, & Vecchi, 1983) and Tabu Search (Glover, 1989), works on improving based on a single solution agent; it is easy to implement but lacks exploration and easily stagnates at the local optima, whilst the population based metaheuristic performs searches with multiple initial starting points in parallel, such as the Ant Colony Optimization (Dorigo & Gambardella, 1997) and Genetic Algorithm (Goldberg, 1989). However, it suffers from a quick convergence and unavoidable redundant searches, and uses much memory during a search (Talbi, 2009). Recently, a few researches inspired by nature, i.e. an animal, the cuckoo (Ouaarab, Ahiod, & Yang, 2014) and the invasive weed phenomenon in agriculture (Zhou, Luo, Chen, He & Wu, 2015), have been applied to the TSP.

A novel metaheuristic algorithm proposed by Yang and Wang (2007) known as the Water Flow Algorithm (WFA) has shown an ability of balancing between exploration and exploitation, due to the idea of dynamic population searching agents. The WFA has been successfully adapted and applied in different Combinatorial Optimization (CO) problems including bin-packing (Yang & Wang, 2007), manufacturing cell formation problems (Wu, Chung, & Chang, 2010), and Nurse Scheduling (Shahrezaei, Moghaddam, Azarkish, & Sadeghnejad, 2011). The WFA for solving the Travelling Salesman Problem (TSP) (Srour et al., 2014) is shown to outperform the state of the art metaheuristics in this paper.

The basic operations of the WFA for solving the Travelling Salesman Problem (TSP) include initialization, flow splitting and moving, flow merging, water evaporation, and water precipitation (Srour et al., 2014). Initialization includes parameter setting and initial solution generation. The original WFA is adopted for the parameter setting (Yang & Wang, 2007). The initial solution is generated by the Nearest Neighbour (NN). Normally, the process of enhancement of the WFA only starts after the initialization process. The flow splitting operation is conducted and depends on the flow momentum value. After that, the moving operation is conducted where the design is based on the type of target problem being solved. The algorithm for the flow moving process combines two types of neighbourhood structures, namely the insertion move and k-opt.

The flow merging operation is the operation that combines more than two flow moves to the same location. This operation merges more than one flow into a single flow. Masses and momentums accumulate to compose an integrated flow to reinforce the solution search. This accumulation helps the stagnant flow to escape from the trapped location. This operation checks a flow with others whether they share the same location, and if it does, the latter flow will merge into the former one. The merging operation is executed to eliminate redundant flows.

The water evaporation operation is performed after the flow merging operation. This operation aims to simulate the natural evaporation of water into the air. Water evaporation is executed when evaporation conditions are met. The WFA uses the concept of water evaporation for preparing regeneration flows to increase the wideness of a solution search. When the evaporated water accumulates to a certain amount, it will return to the ground. This process is known as the precipitation operation, and it is the natural rainfall behaviour. This operation achieves redistribution of flows that escape from the local optima and spread the solution search range. Two types of precipitation are used in the WFA known as the enforced precipitation and regular precipitation.

There are many methods that can improve the metaheuristic solution quality such as a hybrid of different meta-heuristics, adding heuristics and using some memory to avoid redundant searches. k-opt is the most popular local search algorithm and has been applied in most meta-heuristic algorithms. k-opt move changes a route by replacing the k edges from the route, with the k edges denoting a visit to each city exactly once; it then picks a shorter route (Helsgaun, 2006). A higher k-opt will obtain better solutions. However, the most popular k is 2 and 3 (Helsgaun, 2006). This is because of the complexity of the algorithm is increased in O (n^k) (Helsgaun, 2006). Therefore, k>3 is considered too complex. 3-opt and 4-opt local search designed based on basic concept of k-opt, apply the sequential checking all cities. The sequential exchange criterion is a necessary condition that the exchange of link in a tour is that the chain is closed (Helsgaun, 2000). Basic concept of k-opt, apply the sequential checking all cities as the example shows in Figure1 below at line 13 to 26. The sequential exchange criterion is a necessary condition that the exchange of link in a tour is that the chain is closed. And it shown a sequential exchange is not possible for 4-opt (O(n^4)).There are some methods to speed up the 2-opt and 3-opt algorithm and reducing the complexity to O(mn) (Helsgaun, 2000; Nilsson, 2003).

Proposed WFA using k-opt for TSP

The Water Flow-like Algorithm (WFA) for the TSP (Srour et al., 2014) consists of five basic operations mentioned in the previous section, and 9 equations and 11 steps described by the proposed WFA. This section proposed the WFA for the TSP with a higher k-opt: 3-opt (WFA-TSP-3opt) and 4-opt (WFA-TSP-4opt). Figure 1 shows the proposed WFA for the TSP with 3-opt and 4-opt, respectively.

Firstly, this operation starts with the initialization, where the initial solution is generated by using the Nearest Neighbour (NN) which similar in (Srour et al., 2014) (line 2). The process of enhancement of the WFA only starts after the initialization process.

Next, the process continues on to the flow splitting and moving operation (line 5 to 16). The flow splitting operation is conducted and depends on the flow momentum value. The moving operation is conducted where the design is based on the type of target problem being solved. The algorithm for the flow moving process combines two types of neighbourhood structures, namely the insertion move and k-opt. The insertion move is used to determine the new locations of the sub-flows while the 3-opt and 4-opt are used to find the best neighbour solution for each sub-flow. This algorithm used the full sequential checking for both 3-opt and 4-opt, which cause complexity *N* (maximum iteration number), where *N*=(*n*-*k*-1) x (*n*-*k*-1). However, its time consuming. Therefore, the WFA algorithm is improved by reducing the iterations. The 10% is used full checking based on the k-opt (line 8 to 13), while the other 90% uses fixed 50 iterations (line 14 to 16). This cause reducing the complexity into 10% *N* x (*n*-*k*-1) x (*n*-*k*-1) and the 90% N uses 50 iterations to random select k city point.

```
1  WFA_Algorithm ()
2  Generate an initial solution using NN
   (nearest neighbor) //refer step1 in (Srour et al., 2014)
3  WHILE (stop criterion is false) {
4  Cal. no. of sub-flows; //refer equation 1 in (Srour et al., 2014)
5  Assign sub-flows new locations using insertion move //refer step 3 in (Srour et al., 2014)
6  r=random.nextDouble(1)
7  Define L as tour length;
8  IF (r<0.1){
9  FOR firstPoint from 0 to L-2
10 WHILE LastPoint < L {
11 LeftPoin(k-2) random pick between
12 firstPoint+1 and LastPoint-1
13 Do k-opt local search     }
14  ELSE {
15  FOR iteration from 1 to 50
16  random pick up k point do k-opt local search }
17  Distribute mass of flow to its sub-flows; //refer equation 2 and 3 in (Srour et al., 2014)
18  Calculate the improvement in objective function;
19  Flow merging. //refer step 6 in (Srour et al., 2014)
20  Water evaporation. //refer step 7 in (Srour et al., 2014)
21  IF (rainfall required) {
22  Precipitation. //refer step 8 and step 9 in (Srour et al., 2014)
23  Flow Merging. //refer step 6 in (Srour et al., 2014)
24  }
25  IF (new best solution found)
26  Update best solution record.
27  }
```

Figure 1. Pseudo-code of proposed WFA for TSP with k-opt

The next steps below are follow the (Srour et al., 2014) algorithm which the flow merging operation (line 19) is the operation that combines more than two flow moves that share the same location to the same location. The water evaporation (line 20) operation is performed after the flow merging operation. This operation aims to simulate the natural evaporation of water into the air. After the evaporated water accumulates to a certain amount, it will return to the ground. This process is known as the precipitation operation (line 21 to 24), and it is a natural rainfall behaviour.

RESULT AND DISCUSSION

The performance of the proposed WFA-TSP-3opt and WFA-TSP-4opt is compared with algorithm in (Srour et al., 2014) and evaluated using 16 TSP benchmark datasets from TSPLIB ranging from 51 to 3795, which represented as small, medium, and large sized datasets. All algorithms are implemented in a Java environment JDK 1.6. Each dataset is based on 10 independent operations with 10,000 iterations used to obtain the statistical test results.

The experiment was conducted using an Intel Pentium i3 2.4GHz processor and 4GB RAM. The setting of the parameters for the algorithm testing was adopted from (Srour et al., 2014) where the base momentum is 20, initial mass is 8, initial velocity is 5 and sub flow number limit is 3.

A statistical descriptive analysis is used to obtain the mean, standard deviation, best solution, average computation time, average iteration number, percentage deviation of the average with the best-known solution of the tested algorithms, and based on the values of the mean in terms of solution quality and computation time, the improvement percentage could be calculated. Additionally, a t-test is performed in order to obtain the significant difference of solution quality among those algorithms. The results are analysed in three types of measurements: solution quality, computation time taken to get the best solution and proposed algorithm behaviour.

Quality Solution

Table 1 shows that the WFA-TSP-4opt and WFA-TSP-3opt indicate optimum performance in terms of solution quality for 5 datasets (eil51, eil76, kroA100, eil101, bier127) from a total of 16 datasets and PDavg=0 (The percentage of deviation of the mean values regarding the best-known solution), Meanwhile, for the other 11 datasets the WFA-TSP-4opt showed an accuracy improvement percentage from 0.01%-1.06% compared with the WFA-TSP-3opt. Furthermore, the WFA-TSP-4opt reached optimum performance for 10 datasets (eil51, eil76, kroA100, eil101, bier127, ch130, ch150, kroA150, kroA200, kroB200) from a total of 16 datasets and PDavg=0. The results also show that the WFA-TSP-4opt algorithm gives a more stable solution performance compared to the WFA-TSP-3opt, where most of the standard deviation values for WFA-TSP-4opt were less or equal than` WFA-TSP-3opt. The dataset fl1400 showed a unique result where the standard deviation for the WFA-TSP-4opt value was slightly larger compared to the WFA-TSP-3opt value. Besides, the WFA-TSP-4opt had significant improvement in terms of the solution quality of the datasets including ch130, kroA200, kroB200, lin318, rat575, rat783, fl1400, d1655 and fl13795, with p-value < 0.05 as follow standard t-test for TSP (Srour et al., 2014). The (~) denotes the p-value is equal to one; both of the compared datasets are totally same.

Table 2 shows the mean and standard deviation (SD) of the WFA-TSP-3opt and WFA-TSP-4opt versus the WFA-TSP (Srour et al., 2014), depending on 10 independent runs. It can be seen that the solution quality of the WFA-TSP-4opt is generally better than the WFA-TSP-3opt and WFA-TSP; the WFA-TSP-4opt obtained the best mean solution in all datasets highlighted with bold font in Table 1. The WFA-TSP-4-opt and WFA-TSP-3opt outperformed the WFA-TSP (Srour et al., 2014) in 13 out of 15 datasets in terms of the mean results they established. These datasets are eil51, eil101, bier127, ch130, ch150, kroA150, kroA200, lin318, rat575, rat783, fl1400, d1655 and fl3795; for the other 2 datasets, all these methods got the best known results. The WFA-TSP-4opt obtained the best known results for 9 out of 14 datasets including eil51, eil76, kroA100, eil101, bier127, ch130, ch150, kroA150, kroA200, highlighted in bold. For the other 6 datasets (dataset size larger than 318), the WFA-TSP-4opt still could not obtained the best known results marked which as grey background shown in Table 2. The SD of the WFA-TSP-4opt was lower than the WFA-TSP except for dataset d1655, and the SD of the WFA-TSP-3opt was lower than the WFA-TSP except for dataset rat575 and d1655. It shown WFA-TSP-4opt has more stable solution compare to WFA-TSP-3opt and WFA-TSP (Srour et al., 2014), respectively.

Furthermore, Table 2 contains the experimental results of the latest research work solving TSP: improved Discrete Cuckoo Search (DCS) (Ouaarab et al., 2014) and discrete invasive weed optimization (Zhou et al., 2015), respectively. It can be obviously seen that the WFA-TSP-4opt and WFA-TSP-3opt outperforms the improved DCS in terms of the mean of 10 over 12 datasets including eil76, eil100, bier127, ch130, ch150, kroA150, kroA200, lin318, rat575 and rat783. Both the WFA-TSP-4opt WFA-TSP-3opt and the improved DCS obtained 8 over 12, including eil51, eil76, kroA100, eil100, bier127, ch130 and ch150 with best solutions, while for the other 4 datasets the WFA-TSP-4opt got the better best solutions. WFA-TSP-3opt got only one worse solution (lin318) compared with Improved DCS (Ouaarab et al., 2014), others 3 solutions WFA-TSP-3opt obtained better result. Meanwhile, except for two datasets, both methods got the optimal mean; for the other datasets the WFA-TSP-4opt Standard Deviation results were less than the Improved DCS (Ouaarab et al., 2014). The Standard Deviation of WFA-TSP-3opt generally less than Improved DCS (Ouaarab et al., 2014) except for dataset rat575, WFA-TSP-3opt SD larger than Improved DCS (Ouaarab et al., 2014) (39.184 vs 35.45). Higher Standard Deviation result means not stable result. On the other hand, the invasive weed optimization only had 3 datasets the same with our picked datasets. The results of the comparison show that the WFA-TSP-4opt, WFA-TSP-3opt outperforms in terms of best solution and mean. In addition, the WFA-TSP-4opt and WFA-TSP-3opt obtained its results with 10 runs, the discrete invasive weed optimization with 20 runs, but the improved DCS with 30 runs.

Compared with Srour et al. (2014) paper, WFA-TSP-3opt and WFA-TSP-4opt applied insertion move to determine the new locations of the sub-flows, new algorithm applied more complex k-opt to find better neighbour solution.

Computation Time

Table 1 shows the average computation time of the WFA-TSP-4opt for all datasets, from small to large datasets, which are higher between 5.4% to 596% compared with WFA-TSP-3opt, but the relation between the computation time and city size revealed that even 4-opt can represent time as k x n (k not larger than 6, computation time increased 596% at most), where n is the city size and k is a constant number.

Table 1
Results of the comparison between WFA-TSP-4opt and WFA-TSP-4opt in small, medium, and large datasets

Dataset	WFA-3opt						WFA-4opt						p-value	Performance%	
	Mean	SD	Best	AVG. time	Avg. iteration	Pdavg	Mean	SD	Best	AVG. time	Avg. iteration	Pdavg	Solution quality	Accuracy	Time
eil51	**426**	0	426	4	834.3	0	**426**	0	426	9.4	828.3	0	~	0.00	-135.1
eil76	**538**	0	538	2.1	385.8	0	**538**	0	538	8.7	385.8	0	~	0.00	-315.7
kroA100	**21282**	0	21282	1.2	95.2	0	**21282**	0	21282	3.7	95.2	0	~	0.00	-211.8
eil101	**629**	0	629	15.3	871.1	0	**629**	0	629	32.1	871.1	0	~	0.00	-109.6
bier127	**118282**	0	118282	50.6	959.3	0	**118282**	0	118282	53.3	959.3	0	~	0.00	-5.4
ch130	6121.7	14.7	6110	21.9	518.5	0.19	**6110**	0	6110	33.4	518.5	0	0.033	0.19	-52.3
ch150	6532.2	8.9	6528	54.2	1767.4	0.06	**6528**	0	6528	142.3	1767.4	0	0.168	0.06	-162.5
kroA150	26527.2	8	26524	50	3418.8	0.01	**26524**	0	26524	244.6	3418.8	0	0.239	0.01	-389.3
kroA200	29409	31.6	29368	150.9	3519.3	0.14	**29368**	0	29368	534.7	3523.8	0	0.003	0.14	-254.3
kroB200	29482.2	44.4	29438	143.8	5673	0.15	**29437**	0	29437	997.5	6428.7	0	0.011	0.15	-593.7
lin318	42393	134.8	42203	356.4	6982.5	0.87	**42212**	131.48	42082	2483.3	6982.5	0.435	0.008	0.43	-596.8
rat575	6955.6	39.2	6874	1489.6	7689.1	2.7	**6881.9**	35.449	6822	7815.9	7552.1	1.608	0.005	1.06	-424.7
rat783	9043.9	37	8981	3309.4	8311.9	2.7	**8986.1**	23.976	8962	17573.4	8442.8	2.045	0.002	0.64	-431.0
fl1400	20307.3	33.8	20269	12536.3	7185.3	0.9	**20261**	52.532	20173	63146.8	6987.4	0.665	0.009	0.23	-403.7
d1655	64409.3	453.2	63685	16501.4	8682	3.67	**63903**	321.16	63460	76466.7	7757.1	2.856	0.006	0.79	-363.4
fl3795	29367.8	154.3	29155	123808	9462	2.07	**29219.7**	93.86	29134	644380	9554.3	1.56	0.035	0.5	-420.5

Table 2 indicates the average computation time of the proposed algorithm WFA-TSP-3opt and WFA-TSP-4opt versus the WFA-TSP (Srour et al., 2014) in small datasets. The three different algorithms' computation time changed slightly, including for datasets eil51, eil76, kroA100, kroA200, lin318 and rat575. While the other datasets changed significantly between the WFA-TSP-3opt and WFA-TSP with the WFA-TSP-4opt and WFA-TSP-3opt. Therefore, Figure 2 reveals that the WFA-TSP with higher k-opt require a longer computation time to process, but the relationship between computation time and city size revealed that even 4-opt can represent time as k x n, where n is the city size and k is a constant number. However, slope k value for WFA-TSP-kopt less than applying k-opt into Tabu Search (Tsubakitani & Evans, 1998).

Table 2
Comparison of the experimental WFA-TSP-3opt, WFA-TSP-4opt results with the results of WFA-TS

		WFA-TSP [12]			WFA-TSP-3opt			WFA-TSP-4opt			Improved discrete Cuckoo Search[20]			Discrete invasive weed Optimization [21]		
TSP instance	BKS	Best	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best	Mean	SD
eil51	426	426	426.6	0.52	426	426	0	426	426	0	426	426	0	428	429	/
eil76	538	538	538	0	538	538	0	538	538	0	538	538.03	0.17	/	/	/
kroA100	21282	21282.0	21282	0	21282	21282	0	21282	21282	0	21282	21282	0	21285	21290	/
eil101	629	629	630.7	2.83	629	629	0	629	629	0	629	630.43	1.14	/	/	/
bier127	118282	118282	118714.94	177.62	118282	118282	0	118282	118282	0	118282	118358.63	12.73	/	/	/
ch130	6110	6110	6133.9	29.29	6110	6121.7	14.728	6110	6110	0	6110	6135.96	21.24	/	/	/
ch150	6528	6528	6542.2	14.22	6528	6532.2	8.8544	6528	6528	0	6528	6549.9	20.51	/	/	/
kroA150	26524	26524	26568.4	58.45	26524	26527.2	8.025	26524	26524	0	26524	26569.26	56.26	26720	26730	/
kroA200	29368	29368	29438.2	114.84	29368	29409	31.556	29368	29368	0	29382	29446.66	95.68	/	/	/
kroB200	29437	42334	29555.2	96.9	29438	29482.2	44.4	29437	29437	0	/	/	/	/	/	/
lin318	42029	42278	42490.1	208.54	42203	42393	134.77	42082	42211.7	131.5	42145	42434.73	185.43	/	/	/
rat575	6773	6971	7000.8	21.73	6874	6955.6	39.184	6822	6881.89	35.45	6896	6956.73	35.74	/	/	/
rat783	8806	9126	9180.2	40.79	8981	9043.9	36.979	8962	8986.11	23.98	9043	9109.26	38.09	/	/	/
fl1400	20127	20365	20449	50.46	20269	20307.3	33.828	20173	20260.9	52.53	/	/	/	/	/	/
d1655	62128	64313	64878.6	344.25	63685	64409.3	453.15	63460	63902.5	321.2	/	/	/	/	/	/
fl3795	28772	29400	29578.8	135.7	29155	29219.7	93.86	29134	29219.7	93.86	/	/	/	/	/	/

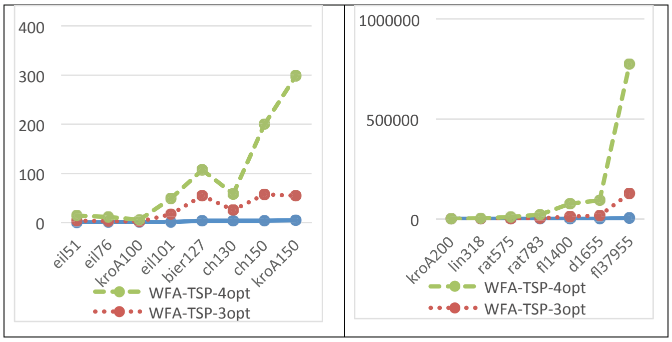

Figure 2. Comparison of average computation time of WFA-TSP, WFA-TSP-3opt and WFA-TSP4-opt for small, medium and large dataset

Proposed Algorithm Behavior. Appendix I shows the general search behaviour of the WFA-TSP-3opt and WFA-TSP 4-opt and a comparison with the WFA-TSP (Srour et al., 2014) in three size of datasets which represented as small, medium and large to show the exploration behaviour. For small (eil101) datasets, all methods showed the minimum fluctuation (the shadow portion), which means few exploration. For medium (lin318) datasets, all methods showed more fluctuations than the WFA-TSP, and for large (d1655) datasets, all methods showed the maximum fluctuation, which means more exploration in search behaviour. Along with the increased dataset, there was more extensive exploration search behaviour in finding a good solution. The figure also indicates that for medium (lin318) and large (d1655) datasets, the WFA-TSP-4opt has a larger exploration. But for the small datasets, the WFA-TSP-2opt and WFA-TSP-3opt revealed stronger exploration in the search behaviour. Therefore, the figure shows a higher k-opt of Water Flow Algorithm applied shows a much stronger exploration in medium and large datasets, but less exploration in small datasets. This behaviour causes obtain quality solution.

Figure 3 and Figure 4 show the log graph best solution of datasets lin318 and fl3795, obtained within 10 independent runs, with iterations below 1000, and between 1000 and 10000, respectively. The speed of the WFA-TSP-3opt in reaching the optimal solution was faster compared to the WFA-TSP-4opt. But the WFA-TSP (Srour et al., 2014) got the optimal solution in later iterations compared with WFA-TSP-3opt and WFA-TSP-4opt. Therefore, the WFA-TSP-3opt is an earlier convergence compared with the WFA-TSP-4opt. The WFA-TSP-4opt has shown diversity of solution because of the slower convergence compared with the WFA-TSP-3opt, but earlier than WFA-TSP (Srour et al., 2014).

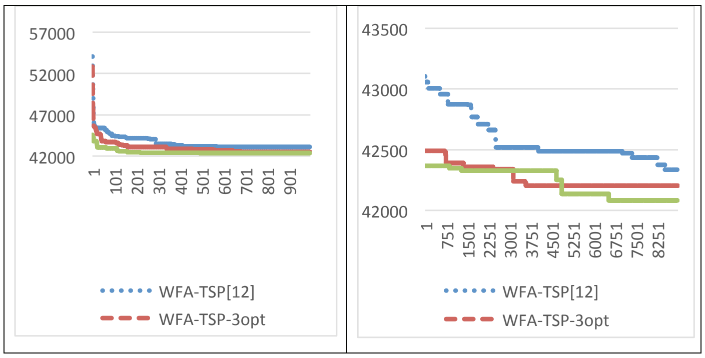

Figure 3. Comparison of the best objective value during the optimiztion process of WFA-TSP-3opt and WFA-TSP-4opt for "lin318" TSP dataset

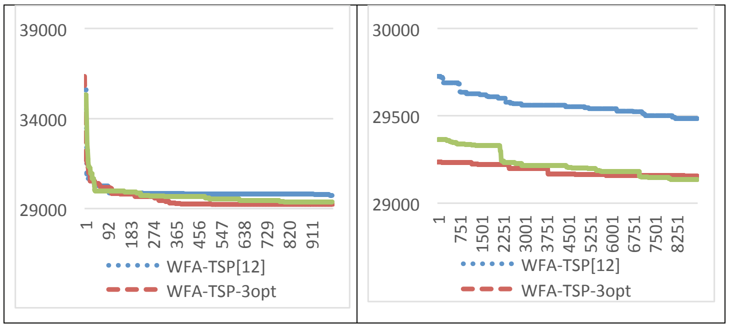

Figure 4. Comparison of the best objective value during the optimiztion process of WFA-TSP-3opt and WFA-TSP-4opt for "fl3795" TSP dataset

CONCLUSIONS

This paper has presented an improvement of the WFA-TSP using the k-opt local search strategy. The higher the k, the better quality solution is obtained. With such results, the WFA with the highest k-opt, i.e. WFA-TSP-4opt, has become the benchmark algorithm for TSP as it outperforms the improved discrete cuckoo search (Ouaarab et al., 2014) and the discrete invasive weed optimization (Zhou et al., 2015) in terms of best, mean and standard deviation. However, the difference is the fact that the use of a higher k-opt has increased the computation time, but the ratio increment is far less compared with other algorithms linear form (k x n). The higher k means higher exploration and slower convergence. This is due to the dynamic exploration behaviour and convergence speed of the WFA. Therefore, it can be concluded that the WFA has potential for more improvement using a more complex local search algorithm, ch as simulated annealing or tabu search.

REFERENCES

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation, 1*(1), 53-66.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing, 1*(3), 190-206.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Reading: Addison-Wesley.*

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research, 126*(1), 106-130.

Helsgaun, K. (2006). *An effective implementation of K-opt moves for the Lin-Kernighan TSP heuristic* (Doctoral dissertation), Roskilde University. Department of Computer Science.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by simulated annealing Sci 220* (4598): 671-680.

Martin, O. C., & Otto, S. W. (1996). Combining simulated annealing with local search heuristics. *Annals of Operations Research, 63*(1), 57-75.

Mo, Y. B. (2010). The advantage of intelligent algorithms for TSP. In *Traveling Salesman Problem, Theory and Applications*. InTech.

Nilsson, C. (2003). *Heuristics for the traveling salesman problem*. Tech. Report Linköping University Sweden.

Ouaarab, A., Ahiod, B., & Yang, X. S. (2014). Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications, 24*(7-8) 1659-1669.

Shahrezaei, P. S., Moghaddam, R. T., Azarkish, M., & Sadeghnejad-Barkousaraie, A. (2011). Water flow-like and differential evolution algorithms for a nurse scheduling problem. *American Journal of Scientific Research, 34*, 12-32.

Srour, A., Othman, Z. A., & Hamdan, A. R. (2014). A water flow-like algorithm for the travelling salesman problem. *Advances in Computer Engineering*, 2014.

Talbi, E. G. (2009). *Metaheuristics: From design to implementation* (Vol. 74). John Wiley & Sons.

Tsubakitani, S., & Evans, J. R. (1998). Optimizing tabu list size for the traveling salesman problem. *Computers and Operations Research, 25*(2), 91-97.

Wu, T. H., Chung, S. H., & Chang, C. C. (2010). A water flow-like algorithm for manufacturing cell formation problems. *European Journal of Operational Research, 205*(2), 346-360.

Yang, F. C., & Wang, Y. P. (2007). Water flow-like algorithm for object grouping problems. *Journal of the Chinese Institute of Industrial Engineers, 24*(6), 475-488.