Solving TSP with Shuffled Frog-Leaping Algorithm

LUO Xue-hui, YANG Ye, LI Xia
College of Information Engineering, Shenzhen University, Shenzhen 518060, China
Email: lixia@szu.edu.cn

Abstract

Shuffled frog-leaping algorithm (SFLA) is a new memetic meta-heuristic algorithm with efficient mathematical function and global search capability. Traveling Salesman Problem (TSP) is a complex combinatorial optimization problem, which is typically used as benchmark for testing the effectiveness as well as the efficiency of a newly proposed optimization algorithm. When applying the shuffled frog-leaping algorithm in TSP, memeplex and submemeplex are built and the evolution of the algorithm, especially the local exploration in submemeplex is carefully adapted based on the prototype SFLA. Experimental results show that the shuffled frog leaping algorithm is efficient for small-scale TSP. Particularly for TSP with 51 cities, the algorithm manages to find six tours which are shorter than the optimal tour provided by TSPLIB. The shortest tour length is 428.87 instead of 429.98 which can be found cited elsewhere.

1. Introduction

The well-known Traveling Salesman Problem (TSP) [1] is a typical combinatorial optimization problem which is commonly used in testing the efficiency of meta-heuristic algorithms. Currently, meta-heuristic algorithms mainly include Genetic Algorithm (GA), Ant Colony Optimization (ACO) and Particle Swarm Algorithm (PSO). These meta-heuristic algorithms have their respective strengths and weaknesses in solving TSP. For instance, GA may take too long in searching for the optimal solution while ACO, involving quite a number of parameters, is easy to fall in local optimum. PSO is relatively simple and has aroused attentions from researchers in different areas [2]. Recently, two researchers, Muzaffar Eusuff and Kevin Lansey brought up a new meta-heuristic algorithm — Shuffled Frog-Leaping Algorithm (SFLA) through observing, imitating and modeling the behavior of frogs searching for food laid on discrete stones randomly located in a pond [3].

Shuffled frog leaping algorithm (SFLA) is a memetic meta-heuristic that is based on evolution of memes carried by interactive individuals and a global exchange of information among the frog population. It combines the advantages of the genetic-based memetic algorithm (MA) and the social behavior-based PSO algorithm with such characteristics as simple concept, fewer parameters adjustment, prompt formation, great capability in global search and easy implementation. While proposed primarily for solving the multi-objective engineering problems such as water resource distribution [4], bridge deck repairs [5] and job-shop scheduling arrangement [6], the successful application of SFLA in solving TSP and the subsequent detailed analysis on its evolutionary mechanism may help for its widespread use in multi-objective optimization problem as well as the in-depth research in its theoretical aspect.

In this paper, we model the SFLA to deal with the traveling salesman problem. The emphasis is placed on the design of local exploration strategy within each submemeplex. Experimental results show that for small-scale benchmark TSP cases, the algorithm performs satisfactorily in terms of solution quality and processing time. In addition, for the benchmark TSP with 51 cities, the algorithm is able to seek for six tours which are shorter than the optimal tour provided by TSPLIB, the shortest of which is 428.87 instead of 429.98 which can be found cited elsewhere.

The remainder of the paper is organized as follows: the basic shuffled frog-leaping algorithm is introduced in Section 2, Section 3 describes how the SFLA is tailed to suit for the TSP. In Section 4, experimental results and discussion are presented followed by conclusions in Section 5.

2. Mathematical model of SFLA

In the SFLA, there are a number of frogs with the same structure but different adaptabilities. Each frog represents a feasible solution to an optimization problem. The entire population of frogs is divided into a number of frog memeplexes according to specific principles and each memeplex represents a type of meme. Frogs in the memeplex conduct local exploration of solution space according to specific

978-0-7695-3382-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISDA.2008.346

strategies which allow the transference of meme among local individuals.

Assume that the initial population is formed by F randomly generated frogs $U(i)$, $i = 1,2,\cdots,F$. The fitness $f(i)$ for the ith frog can be evaluated and sorted in descending order to form memeplexes. As shown in Figure 1, the entire population of F frogs is partitioned into m memeplexes. Each memeplex consists of n frogs satisfying $F = mn$.

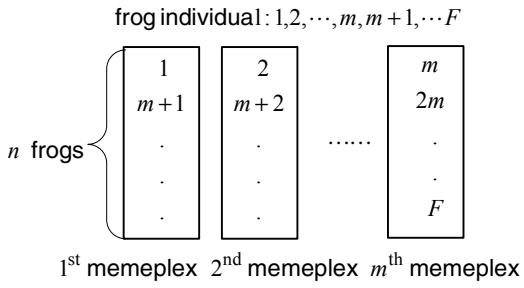


Figure1 Construction of memeplexes

To prevent local optima, a submemeplex is constructed in each memeplex, which consists of frogs chosen on the basis of their respective fitness. The better the fitness, the easier it is chosen. Suppose that the global optimal solution (position of a frog) is represented by P_X ; and for each submemeplex, P_B and P_W respectively represent the best and the worst solution. The local exploration is implemented in each submemeplex, *i.e.*, the worst performance frog in the submemeplex is updated according to the following modification rule:

$$S = \begin{cases} \min\{\text{int}[rand * (P_B - P_W)], S_{\max}\} & \text{for a positive step} \\ \max\{\text{int}[rand * (P_B - P_W)], -S_{\max}\} & \text{for a negative step} \end{cases} \quad (1)$$

$$U(q) = P_W + S \quad (2)$$

where S denotes the updated step size of the shuffled frog-leaping, $q < n$ represents the number of frogs in a submemeplex. *rand* generates a random number between (0, 1) and $S_{\max}$ is defined as the maximum step size allowed to be adopted by a frog after being affected.

After updating, if the solution $U(q)$ is better than the original worst performance solution P_W in the submemplex, P_W will be replaced by $U(q)$. Otherwise, use P_X instead of P_B to carry out the updating strategy (1) and (2) repeatedly. If there is still no improvement, randomly generate a new feasible solution to replace P_W . The calculations continue for a specific number of iterations. Then, all frogs are shuffled for global information exchange. The local exploration and the shuffling process continue until a pre-specified convergence criterion is satisfied.

3. Applying SFLA for TSP

The TSP is a standard test benchmark for meta-heuristic algorithms which attempt to find near-optimal solutions to NP-hard combinatorial optimization problem. In the following, the SFLA is modified to solve the TSP problem.

3.1. The position of individual frog

The position vector of each individual frog represents a feasible solution of TSP. The ith frog is represented by $U(i) = (e_i^1, e_i^2, \cdots e_i^N)$, of which e_i^1 , e_i^2 , ..., e_i^N represent the labels of N cities, indicating that setting from City e_i^1 and passing through cities $e_i^2, \cdots, e_i^N$ in order before finally returning to city e_i^1 . The fitness $f(i)$ is simply defined as the reciprocal of the length of the tour.

3.2. Construction of a submemeplex

The strategy for selecting individual frogs in a submemeplex is that the selection power of an individual frog depends on its corresponding fitness. It is expected that, a frog with better fitness is more prone to be selected as an element in a submemeplex. One may choose that the probability of an individual frog to be selected for the construction of a submemeplex is:

$$P_j = \frac{2(n+1-j)}{n(n+1)}, \quad j = 1,2,...n \quad (3)$$

where P_j denotes the probability of the j th frog being selected to form a submemeplex, and n is the number of frogs in a memeplex. As frogs in a memeplex are already sorted in descending order, eqn(3) implies that the selection probability for constructing the submemeplex decreases with the decrease of the fitness of individual frogs, which coincides with the assumption before.

3.3. Update the worst performance frog

Within each submemeplex, the worst performance frog is updated according to the following simple rule: the qth frog in a submemeplex $U(q)$ is obtained by randomly selecting a subsequence in P_B to replace the corresponding position in P_W , while keeping the other positions in P_W unchanged or if violating the constraint for TSP, just randomly relocate the remain

positions to form a new feasible solution. The idea is illustrated in Figure 2 with a simple numerical example. If the fitness of $U(q)$ is better than P_W's, then replace P_W with $U(q)$, otherwise replace P_B with the global best P_X and carry out the same operation as the above to generate another new feasible solution $U(q)$. If its fitness is better than P_W's, then replace P_W with this new $U(q)$, otherwise randomly generate a new feasible solution to replace P_W.

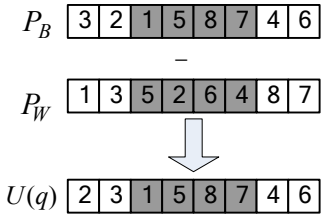


Figure2 Update $U(q)$ in a submemeplex

3.4. Parameters setting

The parameters involved in SFLA include the number of frog population F, the number of memeplex m, the number of frogs in a submemeplex q and the number of iteration for local exploration. An obvious observation is that the number of frogs in a submemeplex q is established to either make the local exploration efficient or increase the probability of frogs with good performance being selected for replacement. A large value of q provides sufficient meme transference with more time, while a small value of q indicates less time used for meme transference. Intuitively we may choose q to be dependent on the problem size.

The number of frogs n in a memeplex and the number of memeplex m satisfies the constraint $F = mn$. It is expected that the bigger the value of m, the stronger the global meme transference, yet the more the time needed. The parameter n helps to make the solution more variant in a memeplex. However, large number of n may slow down the local exploration.

As SFLA is a relatively new algorithm, there is no theoretical basis for parameters setting. We have to resort to experiments. To balance between efficiency and exploration capability, extensive experiments need to be conducted with different settings of parameters before we find effective parameters through trials and errors.

The flowchart for SFLA applied to TSP is illustrated in the following figure.

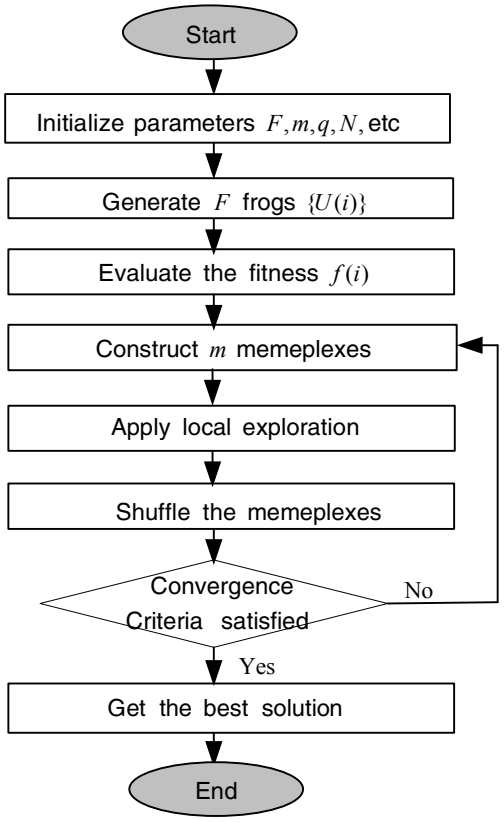


Figure3 Flowchart for SFLA applied to TSP

4. Experimental results and discussion

In the following, we use Burma14, Bays29 and Eil51 from TSPLIB [7] to conduct testing on the SFLA. The algorithm is coded in Matlab 6.5 and executed on Window XP with PIV-1.8GHz CPU and 512MB RAM. For an *N*-city TSP problem, the parameters setting is as follows: There are 10 memeplexes, the frog population is set to 10*N* and the submemeplex has 2*N*/3 frogs. The local exploration for each submemeplex is carried out *N* times. In the experiment, all results are obtained by averaging over 50 independent runs. The results are given in Table 1. In the Table, L_{opt} represents the shortest tour length provided in TSPLIB, L_{best} and L_{avg} denote the best results and the average tour length found by the SFLA. Note that for Eil51, the shortest tour length provided by TSPLIB is 426 which actually is obtained by using distances rounded to integers and it is calculated that 429.98 is the length for the same tour with fine precision. The error rate Err is a reflection of the robustness of the algorithm and is defined as the ratio of the difference between the average length and the best solution L_{best} to L_{best} in percentage.

Table1 SFLA applied to N-city TSP

Case *N*	L_{opt} [7]	L_{best}	L_{avg}	Err(%)	$T_{avg}(s)$
Burma14	30.89	30.89	30.89	0.00	1.84
Bays29	2020	2020	2044.3	1.20	6.75
Eil51	429.98	428.87	436.76	1.57	17.42

It is easy to observe that when applying SFLA to TSP, it manages to find the best routes for Burma14 with 100% successful rate. For TSP with large number of cites, the rate of success falls but the algorithm attempts to find near optimal solutions which are on average within 1.57% of the best result with an average time of 17.42s for the 51 city case Eil51, an encouraging result which implies its potential application in practical optimization problems.

An interesting observation for Eil51 is that, instead of searching for the optimal path provided by TSPLIB, the SFLA is able to find six tours that are shorter. Figure 4(a) presents the best tour provided by TSPLIB and also cited in [8~10]. Readers may refer to eil51.opt.tour [8] for the best route sequence. figure 4(b) to (g) display six shorter tours found by SFLA. Both tour lengths with fine precision as well as the corresponding integer lengths obtained by rounding distances between any two cities (shown in bracket) are given. Obviously, the shortest tour length for eil51 is 428.87 shown in figure 4(b) instead of 429.98 of figure 4(a). By comparing figure (a) and (b), the differences occur at the lower left corner (involving 4 cities) and the middle right side (involving 7 cities). It is interesting to note that all six shorter tours have the same order of sequence at the lower left corner which is different from that of figure 4(a). This undoubtedly implies that even for the lower left corner of the tour, the other algorithms used in [8~10] falls into the same local optimum and fails to find the global optimal solution. In addition, a careful examination illustrates that figure4 (c)~(e) have the same (or similar) structure in the middle while for figure4 (f) and (g), the structure is quite different but they still have tour lengths shorter than that provided by TSPLIB. Even considering the shortest integer tour length, figure 4(a) is not the only choice; we can find another route which also has the same shortest integer length of 426. This is shown in Figure 4(c).

5. Conclusions

A new memetic meta-heuristic algorithm, namely the shuffled frog-leaping algorithm (SFLA), is tailored and applied to solve TSP. The emphasis is focused on the local exploration strategy in submemeplex. Experimental results show that the proposed SFLA is efficient in finding the optimal (for small scale) or near optimal solution with a larger number of cities. Particularly for the benchmark Eil51, the algorithm manages to find six routes that are shorter than the route provided by TSPLIB and cited elsewhere. The shortest path length is 428.87 instead of 429.98.

6. Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant No.60772148.

References

[1] Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, *The traveling salesman problem: a guided tour of comibinatorial optimization*. Chichester: John Wiley & Sons, 1985.

[2] Emad Elbeltagi, Tarek Hegazy, and Donald Grierson, "Comparison among five evolutionary-based optimization algorithms", *Advanced Engineering Informatics*, 2005 (19), pp. 43-53.

[3] Muzaffar Eusuff, Kevin Lansey, and Fayzul Pasha, "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization", *Engineering Optimization*, 2006, 38 (2), pp. 129-154.

[4] Eusuff, M. M., and Lansey, K. E., "Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm", *J Water Resour Plan Manage*, 2003,129 (3), pp. 10-25

[5] Hatem Elbehairy, Emad Elbeltagi, and Tarek Hegazy, "Comparison of Two Evolutionary Algorithms for Optimization of Bridge Deck Repairs", *Computer-Aided Civil and Infrastructure Engineering*, 2006 (21), pp. 561-572.

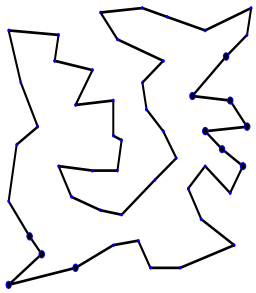
[6] Alireza Rahimi-Vahed, and Ali Hossein Mirzaei, "Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm", *Soft Computing,* Springer-Verlag, 2007.

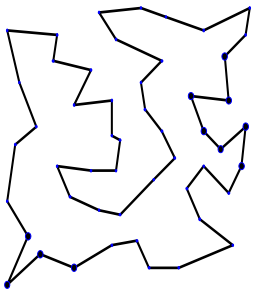
[7] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

[8] Wenliang Zhong, Jun Zhang, and Weineng Chen, "A Novel Discrete Particle Swarm Optimization to Solve Traveling Salesman Problem", *IEEE Congress on Evolutionary Computation*, 2007.

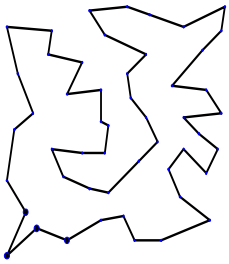
[9] X. H. Shi, Y. Zhou, and L M. Wang, "A Discrete Particle Swarm Optimization Algorithm For Traveling Salesman Problem", *Computational Methods*, 2006, pp.1063-1068.

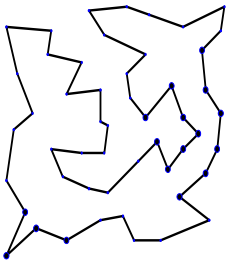
[10] Changsheng Zhang, Jigui Sun, and Yan Wang, "An Improved Discrete Particle Swarm Optimization Algorithm for TSP", *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*

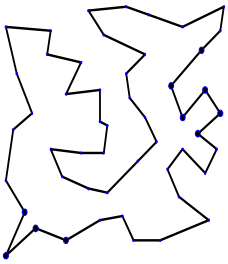




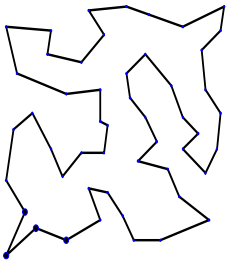
(a) best tour provide by TSPLIB, 429.98(426) **(b) best tour by SFLA, 428.87(427)**

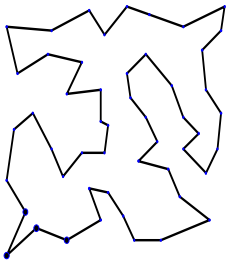






(c) 429.12(426) **(d) 429.53(428)** **(e) 429.93(427)**





(f) 428.98(427) **(g) 429.48(428)**

Figure4 Best tour provided by TSPLIB and the tours obtained with SFLA