Optimizing Multi-criteria k-Shortest Paths in Graph by a Natural Routing genotype-based Genetic Algorithm

Yang Wang[1], Qing Liu[*1,2], Haipeng Ren[1,2], Xuan Ma[1,2] , Long Liu[1,2], Wenqing Wang[1,2], Jing Zhang[1,2]
1.Department of Information and Control Engineering, Xi'an University of technology
2.Shaanxi Key Laboratory of Complex System Control and Intelligent Information Processing
Xi'an, China, 710048
(liuqing@xaut.edu.cn)

***Abstract*— *k*-shortest path problem (KSP) is a more general form of the classical shortest path problem in graph. Its task is no longer to find the shortest path between two vertices, but to find the shortest *k* paths. So far, the reported KSP-algorithms only considers finding shortest *k* paths with regard to a single criterion, while far more application scenarios of the KSP require to determine *k* paths by taking more criteria into account. This article formulates the KSP as a multi-objective optimization problem. In consideration of the problem's property and genetic algorithm's great success in multi-objective optimization, the genetic algorithm is utilized as the optimization tool. The genotype is directly represented as the form of natural routing. In order to make the related genetic operators risk-free, the conception of gene-bank is introduced. And the crossover and mutation are both implemented based on the introduced gene-bank. The proposed genetic algorithm is tested on an undirected graph of 9-vertex/65-edge. The testing result shows that, our proposed algorithm is valid and outperforms some other swarm intelligence-based algorithms in terms of the obtained paths' quality.**

Keywords—*k*-shortest path problem; genetic algorithm; multi-objective optimization; gene-bank.

I. Introduction

The shortest path problem (SPP), as a classical problem in graph theory, has already been well investigated [1-4]. The task of SPP is to seek out the shortest routing between two specified vertices in a graph. These years, increasingly more researchers paid their attentions to the variant of SPP, i.e. *k*-shortest path problem (KSP), the task of which is no longer to find the shortest path between two vertices, but to find the shortest *k* paths. Many practical problems can be boiled down to the KSP, such as vehicle navigation [5], transportation science [6], and network optimization [7], etc. Up to now, a number of reported algorithms are capable of solving KSP [5-12]. Nevertheless, these reported KSP-algorithms only consider finding shortest *k* paths between two vertices with regard to a single criterion. Actually, far more application scenarios of KSP require an algorithm for finding *k* shortest paths by taking more criteria into account. Taking vehicle navigation as an example to explain, users often hope the decision support machine provide more optional routing approach when travelling such that they may make a better choice, neither the shortest path nor the one expending least cost, but a path of compromise. Unfortunately, attempts on finding *k*-shortest problem by considering multiple optimization criteria are still few.

In this article, we formulate the KSP as a multi-objective optimization problem. In consideration of genetic algorithms' great success in multi-objective optimization [13], we utilize the genetic algorithm (GA) as the optimization tool. Our research mainly focusses on the encoding representation of an arbitrary routing and how to implement the related genetic operators which act on such a representation.

The rest parts of this article are organized as follows. The problem we considered is formulated in Section II; The GA for solving the formulated problem is detailed in Section III. Section IV shows and analyzes the simulation results. At last, Section V gives a brief conclusion.

II. Problem Formualtion

Given an undirected graph *G* consisting of a vertex set *V* and an edge set *E*, in which each edge e is associated with a *d*-dimensional positive cost c=(c_1, c_2, ..., c_d), and the cost function $\boldsymbol{c}: E \rightarrow R^+$. According to the introduction, the task of the problem we intend to solve is to determine an optimal path set $P=\{p_1, p_2, \ldots, p_k\}$ by taking more criteria into account. Therefore, we formulated a multi-objective optimization model for this problem. To be specific, each potential path *p* needs to be capable of simultaneously optimizing *d* of optimization objectives, including $f_1=\sum_{e\in p} c_1(e)$, $f_2=\sum_{e\in p} c_2(e)$,..., $f_d=\sum_{e\in p} c_d(e)$.

Obviously, the final solution set to our formulated problem ought to constitute a Pareto-optimal set. In other word, as long as the Pareto-optimal solutions in regard to the above-mentioned *d* of optimization objectives can be obtained, our formulated problem can be solved via choosing *k* non-dominating solutions from the obtained Pareto front. It must be pointed out that, this article focus on the case of the optimization objective number *d*=2 only, but the problem property won't change.

III. Proposed Genetic Algroithm

A. Representation and Initialization

The key to using GA in problem-solving is to design a rational encoding representation to the problem's candidate solution. According to the property of the problem we formulated, we carry out the genotype encoding in manner of direct mapping. Namely, the natural path is directly used as the genotype in GA. Figure 1 gives a specific instance to illustrate the encoding process. In the graph shown in Fig.1, vertices 1 and 9 are separately the specified origin and destination. The natural path (1→2→4→7→9), depicted in bold, can be directly stored in the form of linear string (1|2|4|7|9). And such a linear string can be treated as the genotype, which possesses all the topological information of the natural path (1→2→4→7→9). Each element on such a linear string is referred to as a gene-chip. Obviously, such an encoding representation scheme is intuitive and easy-programming.

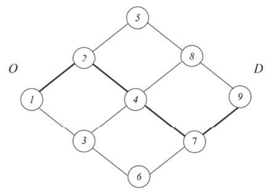

Fig.1. Natural path in an undirected graph *G* with 9-node/ 12-edge, and its corresponding genotype.

To generate an arbitrary natural path at random in a given undirected graph *G*, we specially implemented a generating algorithm and encapsulated this algorithm as a specialized operator, referred to as RP-generator, implying its function of generating random path (RP). The algorithm description to our encapsulated RP-generator is given in the form of pseudo-code as below.

```
Step 1 Input graph $G(V,E)$ ;
Step 2 Initialization:
    $v$-RP = ∅;
    $Origin = O$;
    $Destination = D$;
    $CurrVertex = Origin$;
    $N=\{V_{curr}|$ All vertices connecting with $CurrVertex\}$;
Step 3 Add Origin into the $v$-RP, i.e. $v$-RP=$\{O\}$;
Step 4 Randomly choose a vertex $V_k$ out of $N$, and then execute $CurrVertex=V_k$, where each $V_k$ can be chosen for only once;
Step 5 Update $N=\{V_{curr}|$ All vertices connecting with $CurrVertex\}$;
 /*
It must be emphasized that, once N= ∅, v-RP must be cleared and the algorithm ought to restart from the Step 3.
*/
Step 6 Add the CurrVertex into $v$-RP;
Step 7  Go back to Step 4 if $CurrVertex \neq Destination$;
Output $v$-RP as the natural routing-based genotype otherwise.
```

It needs to be emphasized that, our adopted natural routing-based genotype is length-variable, far differing from the commonly used linear string representation of fixed length. As a result, the genetic operators acting on such a length-variable genotype ought to be improved to adapt this change.

B. Crossover

The crossover operator is an important one of the genetic operators acting on the genotypes. It enables the exchange of genetic information between any two individuals. To our formulated problem, the so-called genetic information refers to the topological structure. In other words, as long as we can carry out the exchange of some local topological structures between two natural routing-based genotypes, the crossover operator is available. Not only that, after the gene-chips being recombined, the produced offspring cannot be with loops and disconnected. That requires the improved crossover operator is risk-free in terms of operating genotypes.

In order to implement the process of exchanging partial topological structures between two parent individuals, we proposed a conception of gene-bank. Figure 2 gives a simple instance for easy explanation. Two natural routing-based genotypes shown in Fig.2 (a) and (b) separately possess different genetic information. They can be treated as two parents, i.e. parent-1 and -2. We combined them both together and thus obtained a new topological structure, which is shown in Fig.2 (c). It is no hard to see that, the newly obtained topological structure contains the entire local topo-structures of the both parents. As mentioned before, topological structure implies genetic information. Therefore, we named the topological structure shown in Fig.2 (c) as gene-bank, because it possesses the entire genetic information from both genotypes of (a) and (b). We noticed that, the gene-bank is with loops, thus being an undirected graph rather than a path. By randomly eliminating different unnecessary loops of the gene-bank, we can get different paths. Since these newly got paths were derived from the gene-bank, they inevitably inherit varying genetic information from the parent-1 or -2. As thus, they can be treated as the offspring. Figure 2 (d) shows such an offspring path, which indeed inherits the topological features from the parent-1 and -2. In fact, the crossover operator has already been carried out during such a process of constructing gene-bank and eliminating loops. As for the method of eliminating loops from the gene-bank, we directly invocate the previously encapsulated RP-generator via passing the gene-bank as an argument.

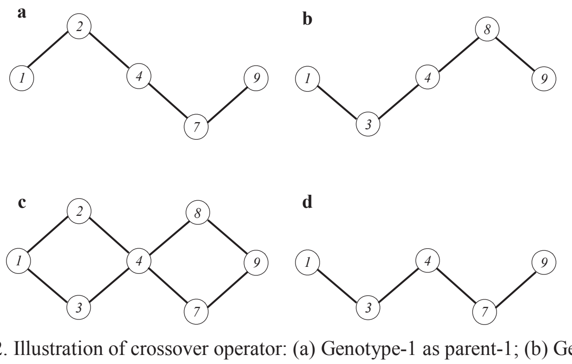

Fig. 2. Illustration of crossover operator: (a) Genotype-1 as parent-1; (b) Genotype-2 as parent-2; (c) Gene-bank obtained via combining genotype-1 and -2; (d) An offspring derived from the gene-bank in (c) at random.

C. Mutation

The mutation is another important genetic operator in GA. It enables abrupt change occurs on some partial topological structure of a genotype. However, unadvisedly alter one or more gene-chips may result in the genotype being with loops or disconnected. Therefore, the mutation operator should also be risk-free in terms of operating genotypes.

To carry out the mutation operator, the conception of gene-bank is still used. Actually, the mutation is essentially the process by which a genotype inherits some gene-chips from an unknown genotype. From this viewpoint, the mutation process can be considered as a particular case for the crossover to some extent. In consideration of this, we implement the mutation operator in a similar way to the crossover operator. To be specific, we still need to construct a gene-bank. The only difference is that, when mutating an individual, the gene-bank is built by combining its genotype and another randomly generated genotype. As a result, any natural routing derived from this gene-bank can be treated as a mutant. Figure 3 demonstrate a simple example for explaining such a process. And the specific procedure will not be detailed due to its similarity to the implementation of the crossover operator.

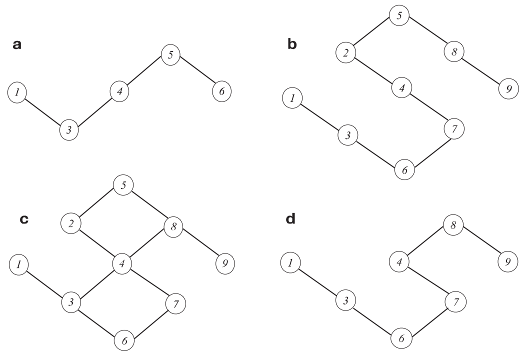

Fig. 3. Illustration of mutation operator: (a) Genotype before mutation; (b) An unknown genotype; (c) Gene-bank obtained via combining the topological structure shown in (a) and (b); (d) Mutant, the genotype after mutation.

D. *Pareto Ranking-Based Selection*

The selection operator is the core of algorithm deduction in GA. It enables the algorithm to retain the superior candidate solutions for later breeding while the inferior ones are eliminated. On the grounds that the basis for selection is based on the individual rather than the topological structure, the latter will not be changed. In other words, it is risk-free for the natural routing-based genotype. Therefore, we retained the non-dominated sorting mechanism and crowded comparison operator of NSGA-II. Readers can refer to the implementation of procedures for more details in reference [13]. The drawback of non-dominated sorting-based selection will accelerate the decrease of the population. With regards to that, we replace the repetitive individuals among the population by the newly generated ones.

Based on the previous description, this section introduces how the previously related genetic operators are organized together. During a generation, the primary procedure can be drawn as the following schematic diagram in Fig. 4. Evolving in this manner, generation by generation, the set of Pareto optimal solutions will ultimately be returned by our proposed algorithm.

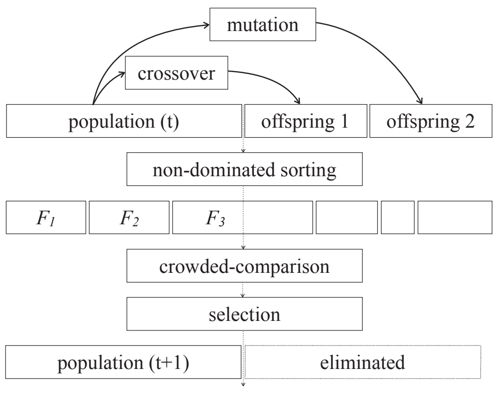

Fig. 4. The proposed algorithm primary procedure

IV. Experiments and Discussion

We implemented the proposed genetic algorithm in Matlab and carried out the simulation experiments on a computer with an Intel Core i5 @ 2.7 GHz CPU, 8 GB RAM, and Mac OS 10.13 operating system, in order to investigate its validity and performance.

A. *Experimental Results*

To verify our algorithm's validity, we compare our approach with shuffled frog leaping algorithm (SFLA), in consideration of the SFLA outperforming the particle warm optimization (PSO) [15] and the genetic algorithm (GA) [10] in terms of the obtained paths' quality [12], through the task of finding ten shortest path spanning the vertices 1 and 26 in a multi-cost undirected graph G, as shown in Fig. 5. The parameters were listed as follows: population size P_s= 50; crossover rate P_c = 0.8; mutation rate P_m = 0.1 [13]; maximum number of iterations N_{max} =100. As for the compared algorithms' parameters, the same settings as reported by reference.

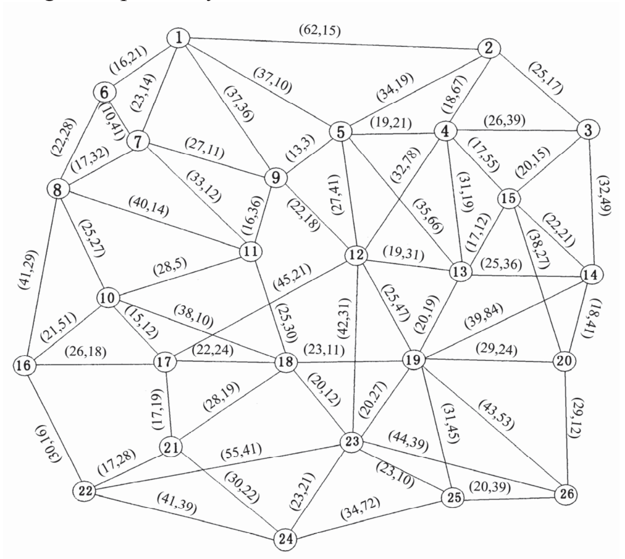
Fig. 5. An undirected Graph *G* with 9-node/ 65-edge

TABLE I. COMPARISION BETWEEN OUR PROPOSDED ALGORITHM AND SFLA[12]TEST PATHS BY SFLA

No.	Paths	Criterion-1	Criterion-2	Proposed GA	SFLA[12]	Invalid
1	1→9→12→19→26	127	136	○	○	
2	1→5→12→19→26	132	133	○	○	
3	1→5→13→19→26	135	130	○	○	
4	1→9→12→19→25→26	135	138	×	○	√
5	1→5→4→15→20→26	140	131	×	○	√
6	1→7→9→12→19→26	140	125	×	○	√
7	1→5→9→12→19→26	140	113	○	○	
8	1→5→12→19→25→26	140	182	×	○	√
9	1→9→12→13→19→26	141	139	×	○	√
10	1→9→11→18→23→25→26	141	163	×	○	√
11	1→7→11→18→23→26	145	107	○	×	
12	1→5→9→12→23→26	148	101	○	×	
13	1→7→11→18→19→26	157	102	○	×	
14	1→7→11→18→19→20→26	162	100	○	×	
15	1→7→11→10→18→23→26	186	92	○	×	
16	1→7→11→10→18→19→26	188	87	○	×	

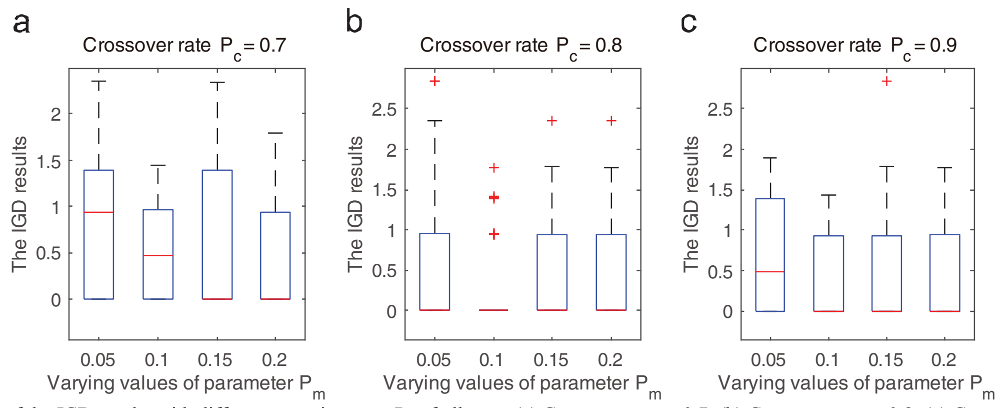

Fig .6. Box plots of the IGD results with different mutation rates P_m of all runs: (a) Crossover rate = 0.7; (b) Crossover rate = 0.8; (c) Crossover rate = 0.9.

In this test, we tested our proposed algorithm by using the problem mentioned in literature [12] and listed the results in Table I. As a comparison, the reported results obtained by the shuffled frog leaping algorithm (SFLA) [12] are also listed. According to the comparison, our proposed algorithm performed far better than the SFLA in terms of path-quality. To be specific, the intersection of the two compared algorithms' found path set is not empty. The paths No.1, 2, 3, 7 are found by both. Apart from the four paths, the rest six obtained by our proposed algorithm possess far better property in terms of criterion-2, comparing with the SFLA. It must be clearly clarified that, although the rest six found by the SFLA are also with better property in terms of criterion-1, they are all invalid because of being dominated. For example, the path No. 4 is dominated by No.3 and the paths No. 5, 6, 8, 9, 10 are all dominated by No.7. In order to make the comparison more intuitional, we depicted the paths obtained by both compared algorithm into a 2-dimensional objective space, as shown in Fig.7. It is obvious that, our proposed GA found a far better Pareto-front if we consider the paths obtained by the SFLA made up a Pseudo-Pareto-front.

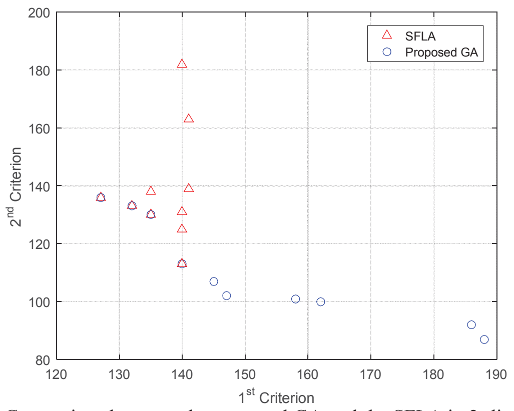

Fig .7. Comparison between the proposed GA and the SFLA in 2-dimensional objective space.

B. Convergence Investigation

To investigate the influence of algorithms' parameters. We combined different crossover rates and mutation rates, through the task of finding Pareto-front our algorithm found in Fig 6. The parameter options were listed as follows:

Maximum iteration number $N_{max} = 50$；

Crossover-rate P_c is separately set as 0.7, 0.8, and 0.9;

Mutation-rate P_m is separately set as 0.05, 0.1, 0.15, and 0.2.

As for the other parameters, they are set as the same as in the test of the previous subsection.

Inverted generational distance (IGD) [14] is an important index for judging a multi-objective optimization algorithm's convergence quality. Assume that P^* is a set of optimal solutions uniformly distributed along *PF*, while *S* represents the approximated set obtained by multi-objective algorithms. It is noted that the true Pareto-front in this test is assumed known in advance. When the size of P^* is large enough to cover the entire true PF, it can effectively measure the convergence and diversity of algorithm. Generally speaking, a lower value of the IGD metric is preferred as it indicates that the obtained set *S* is closer to the true PF and more uniformly distributed along the true *PF*.

In this test, we tested the influence of algorithm's parameters by setting different crossover rate and mutation rate. The recorded IGD of all runs is shown as in Fig. 6. The IGD results are obtained by using 50 sample points that are uniformly distributed along the true PF for the bi-objective test instances. According to the investigation, when the crossover rate is set as 0.8 and the mutation rate is set as 0.1, the IGD result in this parameter combinations outperforms other combinations. Namely, the proposed algorithm can be efficiently performed under the parameter combination and this combination is in accordance with the common optimization experience [16].

V. Conclusion

In the article, we formulated the *k*-shortest path problem as a multi-objective optimization problem and presented a natural routing-based genetic algorithm for solving it. The conception of gene-bank is proposed for improving the related genetic operators including the crossover and mutation and both of genetic operators are easy to program. Being benefiting from the introduction of gene-bank, either the crossover or the mutation becomes risk-free in terms of operating gene-chips on the genotypes and easy programming. The exhaustive simulation shows that, our proposed genetic algorithm outperforms the compared swarm intelligence-based algorithm with respect to the obtained paths' quality and also shows convergence performance.

Additional research is needed regarding the further application of the core ideas such as the risk-free genetic operators in this article for multi-criteria k-shortest paths problem in more complicated topological optimization problems. In the future, we will apply the proposed algorithm for multi-criteria path optimization in the vehicle navigation system.

Acknowledgement

The work is supported in part by National Science Foundation of China (No. 61502385, No. 61401354, No. 61503299, No.61703334), Innovation Research Team of Shaanxi Province (No. 2013KCT-04), China Postdoctoral Science Foundation (No. 2016M602942XB), Natural Science Basic Research Plan in Shaanxi Province of China (No. 2017JQ6050), Key Basic Research Fund of Shaanxi Province (No. 2016JQ6015), and Scientific Research Program Funded by Shaanxi Provincial Education Department (No. 16JK1554).

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, S1. 269–271,1959.

[2] Floyd, W. Robert, Algorithm 97, "Shortest path," Communications of the ACM, vol. 5, no. 6, pp. 345–351,1965.

[3] W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," Applied Soft Computing, vol. 8, no. 4, pp. 1643–1653, 2008.

[4] T. A. J. Nicholson, "Finding the shortest route between two points in a network," The Computer Journal, vol. 9, no. 3, pp. 275–280,Nov. 1966.

[5] W.T. Xu, S.W. He, R. Song, and S.S. Chaudhry, "Finding the k shortest paths in a schedule-based transit network," Computers & Operations Research, vol. 39, no. 8, pp. 1812–1826, 2012.

[6] F. Wang, Z.S. You, L.C. Man, Y. Gao, and L.P. Tang, "Application of Dijkstra and Dijkstra-based N-shortest-paths algorithm to Intelligent Transportation Systems," Application Research of Computers, vol. 23, no. 9, pp. 203–205, 2006.

[7] J. Y. Yen, "Finding the k shortest loopless paths in a network," Management Science, vol. 17, no. 11, pp. 712–716, Jul. 1971.

[8] I.S. Jacobs and C.P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, pp. 271–350, 1963,.

[9] D. Eppstein, "Finding the k shortest paths," SIAM J. Comput., vol. 28, no. 2, pp. 652–673, 1999.

[10] X. Ma. "A genetic algorithm for *k* optimal paths problem," Computer Engineering and Applications, vol. 42, no. 4, pp. 100–101, 2006.

[11] V. M. Jimenez and A. Marzal, "A lazy version of Eppstein's k shortest paths algorithm," in 2nd Int. Conf. on Experimental and Efficient Algorithms (WEA), pp. 179–191, 2003.

[12] X. Ma and Q. Liu, "A shuffled frog leaping algorithm for k-shortest paths problem," Information and Control, vol. 40, no. 5, pp. 614-618, 2011.

[13] Deb, K, Pratap, A., Agarwal, S, & Meyarivan, T. "A fast and elitist multiobjective genetic algorithm: NSGA-II." IEEE Transactions on Evolutionary Computation, vol 6, no. 2, pp. 182–197, 2002.

[14] Li H, Zhang Q. "Solving shortest path problem using particle swarm optimizational, MOEA/D and NSGA-II." IEEE Trans Evolut Computation, vol. 13, no. 2, pp. 284–302, 2009

[15] A. W. Mohemmed, N. C. Sahoo, T. K. Geok, "Solving shortest path problem using particle swarm optimization.", Appl. Soft Comput., vol. 8, no. 4, pp. 1643–1653, 2008.

[16] J.Q. Li and G.Z. Shi, "A study of the Relationship of Crossover Rate and Mutation Rate in Genetic Algriothm." Journal of Wuhan University of Technolog, vol. 27, no. 1, pp. 97–99, 2004.