SCIENCE CHINA
Technological Sciences

December 2010 Vol.53 No.12: 3258–3263
doi: 10.1007/s11431-010-4105-2

Solving the constrained shortest path problem using random search strategy

LI KePing*, GAO ZiYou , TANG Tao & YANG LiXing

State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China

Received March 12, 2010; accepted July 23, 2010

In this paper, we propose an improved walk search strategy to solve the constrained shortest path problem. The proposed search strategy is a local search algorithm which explores a network by walker navigating through the network. In order to analyze and evaluate the proposed search strategy, we present the results of three computational studies in which the proposed search algorithm is tested. Moreover, we compare the proposed algorithm with the ant colony algorithm and *k* shortest paths algorithm. The analysis and comparison results demonstrate that the proposed algorithm is an effective tool for solving the constrained shortest path problem. It can not only be used to solve the optimization problem on a larger network, but also is superior to the ant colony algorithm in terms of the solution time and optimal paths.

constrained shortest path, deterministic random walk, optimization

Citation: Li K P, Gao Z Y, Tang T, et al. Solving the constrained shortest path problem using random search strategy. Sci China Tech Sci, 2010, 53: 3258–3263, doi: 10.1007/s11431-010-4105-2

1 Introduction

The shortest path problem (SP) is one of the basic network optimization problems. Its importance is mainly due to finding applications in many areas, such as the train scheduling, internet communication and traffic transportation etc. The constrained shortest path problem (CSP) refers to the constrained problem that includes additional constraints for the paths in a network [1, 2]. CSP is often used as a sub-problem in some applications, for example, the crew scheduling, optimizing picking and multi-commodity flow problems [3—5]. In general, the additional constraints are to establish some limits on the sum of some arc costs. The addition of these constraints to the SP problem results in that CSP becomes NP-hard [6, 7].

The theory of random walk has a long history and has been applied to solve numerous theoretical and practical problems [8, 9]. The dynamics of random walk on networks represents a powerful tool. In general, the random walk model represents a link between the microscopic dynamics and macroscopic observation. As a simple model of diffusion processes, it can be used to explore the connections between network topologies and functional properties of networks [9]. The random walk is also interesting since it could be a mechanism of transport and search on networks [10, 11]. One of the random walk models is the deterministic random walk model, which combines deterministic and probabilistic features. Recently, deterministic random walk has presented many interesting results in different areas, such as the regular and disordered media [12, 13].

Based on deterministic random walk, in this paper, we propose a new algorithm for solving the constrained shortest path problem. The proposed algorithm is a kind of simulation analysis approaches that do not have to find the analytical and mathematical solutions. Since the time is discrete,

*Corresponding author (email: rtkpli@188.com)

© Science China Press and Springer-Verlag Berlin Heidelberg 2010

numerical implementation of the proposed search algorithm is relatively simple. To our knowledge, this work explicitly shows this effect for walk model for the first time. The paper is organized as follows. We introduce the background in Section 2. Section 3 outlines the random walk model. The proposed approach and numerical computation results are respectively presented in Sections 4 and 5. Finally, conclusions of this approach are presented.

2 Constrained shortest path problem

Researchers have developed many methods to solve the CSP problem. In 1980s, Handler and Zang proposed two basic methods for solving the CSP problem [6]. One is the k shortest paths algorithm and the other is based upon Lagrangian relaxation method. Based on Lagrangian relaxation method, Beasley and Christofides used sub-gradient optimization to get lower and upper bounds for the optimal solution of CSP [14]. Mehlhorn and Ziegelmann proposed a hull method for solving the linear-relaxation of CSP [15]. Other methods based on dynamic programming were proposed. Dumitrescu and Boland presented the label-setting algorithm and an exact algorithm based on the weight-scaling method [7]. In addition, Wilhelm, Damodaran and Li reported a pseudo-polynomial method for solving CSP on an acyclic graph [16].

Let $G=(V, E)$ be an undirected graph (or network), where $V=1, \ldots, n$ is the set of nodes and $E=(i, j)$: $i, j \in n$, $i \neq j$ is the set of edges. Each edge has two non-negative weights c_{ij} and t_{ij}. c_{ij} represents the generalized cost, and t_{ij} represents the constrained variable, such as travel time. The mathematical formulation of the CSP problem can be written as follows:

$$\min \sum_{i,j} c_{ij} x_{ij}, \tag{1}$$

$$\text{s.t.} \quad \sum_{ij} x_{ij} - \sum_{jk} x_{jk} = \begin{cases} 1, & j = 1, \\ 0, & j = 2, \cdots, n-1, \\ -1, & j = n, \end{cases} \tag{2}$$

$$\sum\nolimits_{i,j} t_{ij} x_{ij} \leqslant T, \tag{3}$$

$$x_{ij} \in \{0,1\}, \quad \forall i, j, \tag{4}$$

where x_{ij} is binary variable, which is defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if } x_{ij} \text{ is in optimal path,} \\ 0 & \text{others.} \end{cases} \tag{5}$$

The parameter, T, represents the maximum value allowed for the sum of t_{ij}. The shortest path problem is formulated by eqs. (1), (2) and (4). The addition of constraint (3) results in that CSP problem belongs to the set on NP-hard problem.

3 Deterministic random walk

The classical random walk model was proposed by Patlak in 1950s, which deals with a particle moving in a series of steps [17]. Considering an arbitrary finite network, a random walk is defined as that a walker moves along the edges of such a given network. At each time step, the walker is restricted to move to its neighbor node. During each step, the direction of a walk is either independent of or dependent on the direction taken at its preceding step. If the probability for the walker to step in one direction is greater than that in other directions, then the walker propagates in such a direction with higher probability. This is called deterministic random walk.

One deterministic random walk is called tourist walk [18, 19]. Here a walker begins his path from an initial node. At each time step, the walker goes to his neighbor node according to the following deterministic rule: go to the neighbor node that has not been visited in the preceding τ time steps. Here, τ is called the limited memory range. The walker performs a deterministic partially self-avoiding walk. The rotor router model proposed by Jim Propp is another deterministic random walk model that simulates a random walk on a graph [20–22]. In rotor router model, the walker goes to his neighbor node in a fixed order. Recently, this model has attracted considerable attention. Many studies turn out that the Propp model is a very good simulation of a random walk.

Random walk search strategy is a well-known technique, which forwards a query message to a randomly selected neighbor at each time step until the object is found. In ref. [23], the authors used local next-near-neighbor search algorithm to explore the complex networks, where the independent random walk and interacting random walk were considered. Adamic et al. proposed a random walk search algorithm that can be efficiently used to search through power-law networks [11]. They demonstrated the utility of such a search strategy on the GNUTELLA peer-to-peer network. Noh and Rieger used the random walk search strategy to find the mean first passage time (MFPT) in arbitrary networks [24]. Here the authors derived an exact formula for the MFPT from one node to another node. Random walk search processes would be optimal if one follows the optimal path between two nodes under considerations.

4 The search algorithm based on deterministic random walk

A common approach to solve the shortest path problem is to use the graph search algorithm. This search algorithm finds an optimal solution from all possible solutions. However, since the number of network nodes is large, this approach leads to that the amount of work and its need for memory

increase. In this paper, we use the deterministic walk search strategy to solve the shortest path problem. The aim is that the proposed algorithm can be executed with shorter computation time, and can provide good solutions.

In the proposed approach, we introduce a parameter f_i, which is defined as the fitness value of node i. At one time, the value of the fitness function of one node is equal to the cost of the shortest path among the paths passing through this node up to now. That is to say, for each OD, some possible paths, which pass through some nodes and correspond to different costs, can be obtained at one time. Then, for a given node, there exist several possible paths passing through this node. We shall use the cost of the shortest path as the value of the fitness function f_i.

At the beginning, the fitness values of all nodes are initialized. When a search process is finished, these fitness values possibly are updated. The updated rule is as follows. On a selected path, the fitness values of all nodes must be smaller than the search cost J, where J is the sum of c_{ij} on such a selected path. It is very important to select the initial value of fitness function in the proposed algorithm. If such an initial value is selected improperly, we shall not get the optimal solution, even a feasible solution. In this paper, based on the size of network, we regard the maximum possible search cost "J" as the initial value.

Using purely local information, such as the number of nodes and edges, we implement the walk search algorithm that monitors the neighbor nodes. Usually, from a given departure node to its objective node, there are a number of possible paths. The search process for walker will be repeated many times. In the search process, the walker moves according to the following deterministic rules: (a) Starting from the node i, the walker selects one of its neighbor nodes to move. Moreover, the edge e_{ij} connecting the node i and the selected neighbor node j should not be visited in the preceding time steps. If all such edges have been visited in the preceding time steps, the walker randomly selects one of them; (b) The probability the walker moving from the node i to its jth neighbor node is $P_j=\eta_j/\Sigma_r\eta_r$, where $\eta_r=1/f_r$, and r represents the rth edge connecting the node i and its neighbor nodes.

The executed process of the proposed algorithm can be described below.

Step 1. Initialization parameters. Circle time Mc=0. $Mc_{\max}$ is the maximum circle time. Let $f_i=f_{\text{initial}}$, i=1,..., n.

Step 2. Displace one walker on departure node.

Step 3. Circle time $Mc=Mc+1$.

Step 4. According to the deterministic walk rules described above, the walker moves from the departure node to its objective node. As the walker arrives its objective node, a selected path is found. If the sum of t_{ij} on the selected path is smaller than or equal to the parameter T, a possible solution is obtained.

Step 5. Update the fitness values of all nodes which are on the selected path. For the ith node, if $f_i>J$, then $f_i=J$, otherwise $f_i=f_i$.

Step 6. If $Mc<Mc_{\max}$, go to Step 2.

Step 7. Record the best solution which has the smallest cost J.

In step 7, we can range all the obtained solutions. Using this step, one (or more) optimization solution(s) can be found from a specified departure node to a specified objective node. The initialization parameter f_{initial} is reasonably selected in advance. In order to compare simulation results to reality observations, one iteration roughly corresponds to 1 s, and the length of a unit is about 1 m.

5 Numerical computations

The algorithm proposed in this paper is suitable for computer executing. In order to test the power of the proposed search strategy, we apply the proposed algorithm to find optimal paths on an assumed network. This network is randomly generated, which contains 16 nodes. Figure 1 depicts the structure of such a network. Here the lengths of the network edges are given in Table 1. The value of t_{ij} of each edge (i, j) is labeled by a number shown in Figure 1, which is located close to the edge (i, j). The speed at which walker moves is taken as v=2 m/s. In search process, c_{ij} is calculated by $c_{ij}=l_{ij}/v$. The initialization fitness values of all network nodes are all set to be the largest possible c_{ij} by estimation.

Figure 2 presents the processes of the proposed algorithm executed. Here, the horizontal axis indicates the step that walker moves, and the vertical axis indicates the visited node. The searched paths for the walker are between nodes 2 and 15. In Figure 2, the optimal path between nodes 2 and 15 is a solid line with symbol "o", and the other dash lines represent different searched paths. The numbers shown in Figure 2

Table 1 Lengths of the network edges (m)

Edge	Length	Edge	Length	Edge	Length	Edge	Length	Edge	Length
l_{12}	30	l_{210}	42	l_{712}	24	l_{913}	15	l_{1516}	32
l_{16}	27	l_{34}	27	l_{89}	21	l_{1011}	38	l_{1015}	39
l_{17}	35	l_{45}	21	l_{812}	34	l_{1014}	33		
l_{23}	24	l_{410}	25	l_{813}	20	l_{1114}	35		
l_{27}	28	l_{511}	40	l_{910}	8	l_{1215}	37		
l_{28}	24	l_{67}	20	$l_{12\,16}$	29	l_{1315}	20		

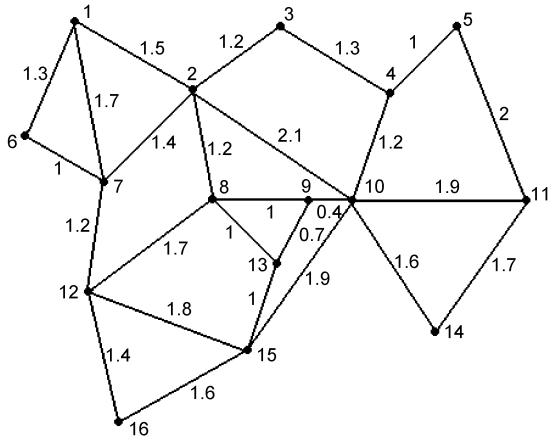

Figure 1 An example of the assumed networks.

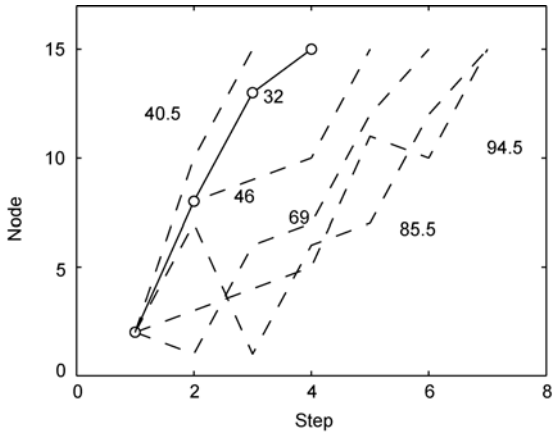

Figure 2 The search processes for walker.

represent the search cost J for each search processes. The searched paths shown in Figure 2 all meet the constraint (3), where T is set to be T=10. From Figure 2, it is obvious that the optimal path has the smallest search cost J among the possible paths which are shown in Figure 2.

In the search process, if one node has smaller fitness value, passing through such a node, there exists one searched path which has smaller search cost. Many nodes which have smaller fitness values consist of an optimal path. According to the definition of the probability P_j, the node having smaller fitness value will be visited with higher probability. In other word, the nodes on an optimal path will be visited with higher probability. This results in that the optimal path can be found with shorter computation time. Figure 3 shows the visited times of all nodes depicted in Figure 1. Here the visited times of one node is the number that walker visits such a node through different paths. This search process is repeated for many times, and the final visited times are obtained by averaging them. From Figure 3, it is obvious that 5 nodes have larger visited times, i.e. 2, 8, 10, 13 and 15. As shown in Figure 2, two optimal paths, which have smaller search cost, are respectively the 2 → 8 → 13 → 15 and 2 → 10 → 15. The result indicates that using the

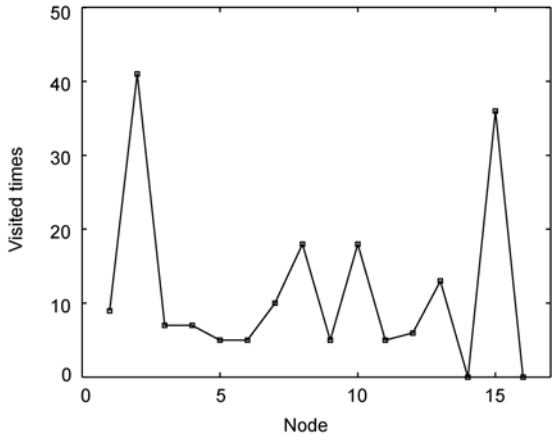

Figure 3 The visited times of all network nodes.

proposed algorithm, some nodes, which are visited for many times, consist of an optimal path.

The parameter T is an important parameter, which is determined in advance according to the studied problem. For example, in train scheduling problem, T is set to be the largest allowed traveling time from one station to another station. In the computation process, if the value of T is too small, it usually is difficult for us to find possible solution. We program the proposed algorithm on a personal computer which has one processor. When the proposed algorithm is executed, we record the solution time, i.e. CPU time for finding the possible solution. Let $T'=\Sigma_{i,j}\ t_{ij}x_{ij}$. One possible solution corresponds to one T'-value. Table 2 gives the solution time for finding different possible solutions. Here the walker searches between nodes 2 and 15. The computation experiment is made 10 times, and the final result is obtained by averaging them. In Table 2, it is obvious that the smaller the T'-value is, the larger the solution time is. Since $T' < T$, when T-value is smaller, the solution time is longer.

So far, many algorithms have been used to solve CSP problem. The ant colony algorithm is one of them [25]. In order to test the performance of the proposed algorithm, in this paper, we compare the proposed algorithm with ant colony algorithm. The comparisons are tested on the same network shown in Figure 1. Here the number of ants in ant

Table 2 CPU time for finding possible solution

T'-value	Solution time (s)
3.2	0.075
3.5	0.068
4.0	0.063
4.7	0.050
4.9	0.040
5.1	0.045
8.4	0.035
8.7	0.035
8.8	0.032
10.2	0.030

colony algorithm is equal to the size of the considered network N. This network includes 240 different OD pairs. For each OD pair, we compare the results of the searched paths found by the algorithm proposed in this paper and the ant colony algorithm. In addition, we compare these 240 searched paths separately found by the two algorithms with real optimal paths. There are 61 paths obtained by the ant colony algorithm which are not the real optimal paths, for example, the searched paths between 10 and 15. However, there are only 3 searched paths found by the proposed algorithm which are not real optimal paths. Table 3 gives part of the comparison results. From Table 3, it is clear that the proposed algorithm is better than the ant colony algorithm.

We consider a larger network size which has 50 nodes. The lengths of all network edges are set to be 20 m. The t_{ij}-value of each edge (i, j) is taken as t_{ij}=1.5. T-value is set to be too large so that all possible paths are possible solutions from a specified departure node to a specified arrival node. The comparison test is made 10 times, and the final result is obtained by averaging them. Table 4 gives part of the comparison results obtained by the proposed algorithm and the ant colony algorithm. In Table 4, the average solution time for finding optimal solutions is presented in columns 3 and 4. From Table 4, we can see that the solution time listed in column 3 is smaller than that in column 4. These comparison results mean that the proposed algorithm is superior to the ant colony algorithm. The proposed algorithm can be executed with shorter computation time.

Table 3 Optimal paths

Departure node	Arrival node	Proposed algorithm	Ant colony algorithm
1	10	1→2→10	1→2→8→9→10
2	14	2→10→14	2→8→9→10→14
3	14	3→4→10→14	3→2→8→9→10→14
4	14	4→10→14	4→5→11→14
5	15	5→4→10→15	5→4→10→9→13→15
9	2	9→8→2	9→10→4→3→2
10	15	10→15	10→9→13→15
10	3	10→4→3	10→9→8→2→3
14	16	14→10→15→16	14→10→9→13→15→16
15	1	15→12→7→1	15→13→8→2→1

Table 4 Solution time for finding optimal solution

Departure node	Arrival node	Proposed algorithm	Ant colony algorithm
1	31	0.011	0.050
2	15	0.009	0.029
2	37	0.001	0.034
3	36	0.024	0.053
4	25	0.017	0.040
8	20	0.009	0.056
11	18	0.003	0.012
17	33	0.007	0.012
19	40	0.014	0.029
26	37	0.014	0.050

One of the important indexes for evaluating an algorithm is the time complexity. We compute the time complexity of the proposed algorithm and then compare it with that of ant colony algorithm. For our proposed algorithm, the scale of the considered CSP problem is equal to the number of network nodes N, and the maximum time step is $Mc_{\max}$. The time complexity of the proposed algorithm is $O(Mc_{\max}\cdot N^2)$. However, as the number of ants in ant colony algorithm is set to be N, the time complexity of ant colony algorithm is $O(Mc_{\max}\cdot N^3)$. This means that the proposed new algorithm is more suitable for solving CSP.

The k shortest paths algorithm is a commonly used algorithm in practical applications [26]. In order to further investigate the performance of the proposed algorithm, we compare the proposed algorithm with the classical k shortest paths algorithm. Here, in k shortest paths algorithm, the Dijkstra's algorithm is used to find the shortest route between all OD pairs [27]. We consider a new network which has 24 nodes. The lengths of all network edges are set to be 20 m. The t_{ij}-value of each edge (i, j) is taken as t_{ij}=1.5. In the considered network, there are totally 552 possible OD pairs. For each OD pair, we made one test respectively using the k shortest paths algorithm and the proposed algorithm. Among 552 measurements, we finally find 33 second optimal routes by the proposed algorithm. The results indicate that using the proposed algorithm, 94% solutions are global optimal solutions. The comparison results demonstrate that the performance of the proposed algorithm is very close to the classical k shortest paths algorithm. However, unlike the k shortest paths algorithm, the proposed algorithm can be used when weights have negative values.

6 Conclusions

In conclusions, this paper introduces a new optimal algorithm to solve the CSP problem. The proposed optimal algorithm is based on deterministic random walk model. Since the proposed algorithm consists of some simple rules, it can be executed with shorter computation time. Remarkably, it can be integrated into a decision support system used by operators, such as the train dispatch system.

In addition, we compare the proposed algorithm to the ant colony algorithm. The results of these comparisons indicate that the proposed algorithm can solve CSP problems in reasonable time and has advantages over the ant colony algorithm in terms of solution time. This means that the proposed algorithm is an effective tool for solving CSP problem.

It should be pointed out that the proposed algorithm may provide an optimal solution, but there is no guarantee that such an algorithm will provide a global optimal solution over all time. It is in fact an improved heuristic search

strategy. In the process of selecting an initial value of optimal algorithm, if we initialize the algorithm by using other heuristic algorithms, there is no doubt that its searching performance can be improved. We believe that the proposed search strategy will open new perspectives for path select modeling. It can be used to solve many problems in practical applications, such as the railway transportation, the city traffic and internet.

The project was supported by the National Natural Science Foundation of China (Grant Nos. 60634010 and 60776829) and the State Key Laboratory of Rail Traffic Control and Safety (Contract No. RCS2008ZZ001), Beijing Jiaotong University.

1 Deo N, Pang C. Shortest-path algorithms: taxonomy and annotation. Networks, 1984, 14(2): 275–323
2 Current J R, Pirkul H, Rolland E. Efficient algorithms for solving the shortest covering path problem. Transp Sci, 1994, 28(4): 317–327
3 Mingozzi A, Boschetti M A, Ricciardelli S, et al. A set partitioning approach to the crew scheduling problem. Oper Res, 1999, 47(6): 873–888
4 Wilhelm W E, Arambula I, Choudhry N N. A model to optimize picking operations on dual-head placement machines. IEEE Trans Autom Sci Eng, 2006, 3(1): 1–15
5 Holmberg K, Yuan D. A multicommodity network-flow problem with side constraints on paths solved by column generation. Informs J Comp, 2003, 15(1): 42–57
6 Handler G E, Zang I. A dual algorithm for the constrained shortest path problem. Networks, 1980, 10(4): 293–310
7 Dumitrescu I, Boland N. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. Networks, 2003, 42(3): 135–153
8 Spitzer F. Principles of Random Walk. New York: Springer-Verlag, 1976
9 Jespersen S, Sokolov I M, Blumen A. Relaxation properties of small-world networks. Phys Rev E, 2000, 62(3): 4405–4408
10 Guimerà R, Díaz Guilera A, Vega-Redondo F, et al. Optimal network topologies for local search with congestion. Phys Rev Lett, 2002, 89(24): 248701
11 Adamic L A, Lukose R M, Puniyani A R, et al. Search in power-law networks. Phys Rev E, 2001, 64(4): 046135
12 Freund H, Grassberger P. The red queen's walk. Physica A, 1992, 190(3-4): 218–237
13 Bunimovich L A. Deterministic walks in random environments. Physica D, 2004, 187(1-4): 20–29
14 Beasley J E, Christofides N. An algorithm for the resource constrained shortest path problem. Networks, 1989, 19(4): 379–394
15 Mehlhorn K, Ziegelmann M. Resource Constrained shortest paths. In: Paterson M, ed. 7th Annual European Symposium on Algorithms. Berlin Heidelberg: Springer-Verlag, 2000. 326–337
16 Wilhelm W E, Damodaran P, Li J. Prescribing the content and timing of product upgrades. IIE Trans, 2003, 35(7): 647–664
17 Patlak C S. Random walk with persistence and external bias. Bull Math Biol, 1953, 15(3): 311–338
18 Lima G F, Martinez A S, Kinouchi O. Deterministic walks in random media. Phys Rev Lett, 2001, 87(1): 010603
19 Stanley H E, Buldyrev S V. Statistical physics: The salesman and the tourist. Nature (London), 2001, 413 (6854): 373–374
20 Kleber M. Goldbug variations. Math Intel, 2005, 27(1): 55–63
21 Levine L, Peres Y. The rotor-router shape is spherical. Math Intel, 2005, 27(3): 9–11
22 Cooper J, Spencer J. Simulating a random walk with constant error, Combinatorics, Probability and Computing, 2006, 15(6): 815–822
23 Tadic B. Exploring Complex Graphs by Random Walks. In: Garrido P L, Marro J, eds. Modeling Complex Systems. Melville: American Institute of Physics, 2002. 24–26
24 Noh J D, Rieger H. Random walks on complex networks. Phys Rev Lett, 2004, 92(11): 118701
25 Dorigo M, Maniezzo V, Colorni A, Ant system:optimization by a colony of cooperating agents. IEEE Trans Systems, Man, and Cybemeties-Part B, 1996, 26(1): 29–41
26 Eppstein D. Finding the k shortest paths. SIAM J Comp, 1998, 28(2): 652–673
27 Dijkstra E W. A note on two problems in connexion with graphs. Numerische Matematik, 1959, 1(1): 269–271