Research on Path Planning Algorithm for Two-Dimensional Code Guidance Model of Automated Guided Vehicle

Wei-Dong Zheng[1], Ben Yan[1(✉)], Zhi-Xian Li[1], Hua-Ping Yao[1], Li-Li Wei[1], and Masahide Nakamura[2]

[1] LuoYang Institute of Science and Technology, No. 90 Wangcheng Road, Luolong District, Luoyang 471023, Henan, China
yanbenjp@gmail.com
[2] Kobe University, 1-1 Rokkodai, Nada, Kobe, Hyogo 657-8501, Japan
masa-n@cs.kobe-u.ac.jp

Abstract. With the development of industrial automation technology, the automated guided vehicle (AGV) has been widely used in industrial transport, warehouse logistics and other fields. As a new guidance model for AGV, the two-dimensional code guidance model has been discussed in many fields. Two-dimensional code guidance model is a type of free path guidance technology. Under this guidance model, the work area of AGV can be divided into multiple rectangle regions, a two-dimensional code is deployed at each intersection of these regions, which carries the guidance information for its neighborhood regions. The AGV walks along a path which is obtained from the central server, and by scanning the two-dimensional code it encountered, it acquires the way to the next node in the path, until it reaches the endpoint. This guidance model has many advantages, such as low-cost, high re-usage, high flexibility etc., it is a competitive technology to replace the traditional track bedding and laser guidance technology.

Although the use of the two-dimensional code technology avoids or solves a lot of problems compared to the traditional guidance technologies, there are still many potential problems to be tackled, such as matching the actual environment, arranging AGV in place etc. Especially, because there is no track for the AGV to follow, a new path planning algorithm should be designed to solve the above problems.

In this paper, we propose a path planning algorithm under the two-dimensional code guidance model. It allows us to find an optimal path for the AGV to follow from the start point to the end point. To demonstrate the feasibility and advantages of this solution, a simple simulation platform is built and the algorithm is tested and verified by experiments.

Keywords: AGV · Path planning · Dijkstra's algorithm · Shortest path · Two-dimensional code guidance technology

V. G. Duffy (Ed.): HCII 2019, LNCS 11581, pp. 355–365, 2019.
https://doi.org/10.1007/978-3-030-22216-1_26

1 Introduction

Automated guided vehicle (AGV) is a kind of wheeled autonomous mobile robot, which is widely used in automation production line, warehousing logistics etc. It is an important unit of flexible manufacturing system, automated storage and retrieval system of logistics transport. At present, there're several mature guidance technologies, such as are electromagnetic guidance, magnetic guidance, optical guidance, laser guidance, inertial guidance and vision guidance, but each of the above guidance technologies has its drawbacks (shown in Table 1).

Table 1. The type and weakness of guidance technologies

Type	Guidance model	Technology	Weakness
Electromagnetic guidance	Fixed path guidance	Electromagnetic	Difficult to change guiding, easy to wear, susceptible to interference
Guidance magnetic	Fixed path guidance	Magnetic Rubber strip belt	
Optical guidance	Free path guidance	Image recognition	High cost, Strict external requirements
Laser guidance	Free path guidance	Laser	
Inertial guidance	Free path guidance	Ground assisted location facility	Accuracy instability
Vision guidance	Free path guidance	Image recognition	Technical bottleneck

As the table above, since the limitations of traditional AGV guidance model, such as difficult to change guiding path, easy wearing, susceptible to interference, high cost, strict external requirements and accuracy instability etc., it can not meet the needs of modern electricity suppliers to discrete heterogeneous picking, then a new guidance model, which employs the two-dimensional code technology is proposed recently.

Two-dimensional code guidance model is a type of free path guidance technology. It does not have strict external environment requirement, and have the advantages of low cost, flexibility, and high efficiency. What it requires is just the network access of AVG cars and the two-dimensional code for each small region, which carries the guidance information for its neighborhood regions [23].

Our research aims to present a complete technical solution of "the path planning and guidance of AGV cars based on two-dimensional code dispersion visual identity model", which can be effectively applied to the logistics robots of the modern storage centers, and enhance logistics efficiency and productivity.

In order to achieve the above research objectives, first of all, we need to solve the optimal path selection problem for AGV, then the two-dimensional guidance system should be designed and implemented.

For this purpose, we propose a path planning algorithm under the two-dimensional code guidance model. It allows us to find an optimal path for the AGV to follow from the start point to the end point. And to demonstrate the feasibility and advantages of this solution, a simple simulation platform is built and the algorithm is tested and verified by experiments.

2 Preliminaries

2.1 Dijkstra's Algorithm

The Dijkstra's algorithm is a classic single-source shortest path algorithm proposed by Dutch computer scientist Edsger W. Dijkstra, which is mainly used to calculate the shortest path form one node in a connected graph to all other nodes. It is used in the breadth of the first search ideas from the inside to the outside of the traversal search, until traversing to the end [1].

The core idea of this algorithm is to record the distance from the source point to any point in real time by means of the edge expansion method. The working process of Dijkstra's Algorithm are described as following steps:

Let the node at which we are starting be called the initial node. Let **the distance of node Y** be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

Step 1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.

Step 2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.

Step 3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.

Step 4. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.

Step 5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.

Step 6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3 [2].

Figure 1 is an example of Dijkstra's algorithm.

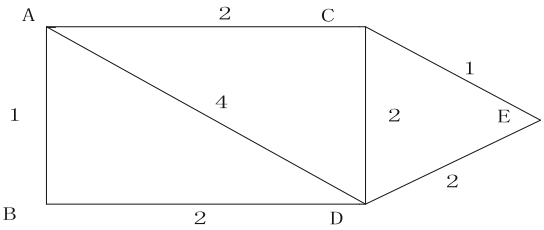

Fig. 1. Dijkstra's algorithm example

The above figure (Fig. 1) requires us to use the Dijkstra algorithm to find the shortest distance from point A to any other point (Where S = {A(source)}, U = {points other than A}). Proceed as follows (number in parenthesis denotes the distance from point A to other points):

STEP 1. Initialize the set S: S = {A (0)}

Set U: U = {B (1), C (2), D (4), E (∞)}

STEP 2. Since there's a direct path from A to B, and the distance is the shortest, move B to the set S,

Set S: S = {A (0), B (1)}

Set U: U = {C (2), D (4), E (∞)}

STEP 3. Update the distance values for each point in the U collection with B as the middle point,

Set S: S = {A (0), B (1)}

Set U: U = {C (2), D (3), E (∞)}

STEP 4. Since there's a direct path from A to C, and the distance is the shortest, movie C to the set S,

Set S: S = {A (0), B (1), C (2)}

Set U: U = {D (3), E (∞)}

STEP 5. Update the distance values for each point in the U set with C as the middle point,

Set S: S = {A (0), B (1), C (2)}

Set U: U = {D (3), E (3)}

STEP 6. There're three paths from A to D, and the path with B as middle point is the shortest, then move C to the set S,

Set S: S = {A (0), B (1), C (2), D (3)}

Set U: U = {E (3)}

STEP 7. Update the distance value for each point in the U collection with D as the middle point,

S set: S = {A (0), B (1), C (2), D (3)}

U set: U = {E (3)}

STEP 8. Move the updated E to S

S set: S = {A (0), B (1), C (2), D (3), E (3)}

U set: U = {}

Finally, we get the shortest distance form source A to other points through the set S.

2.2 Two-Dimensional Code Guidance Model for AGV

Two-dimensional code guidance technology is a free guidance technology which is proposed recently. There're two important parts in the two-dimensional code guidance model, one is a central server, which is a computer which computes and send the path information to each AGV car whenever it requires; another are the AGV cars, which have wireless network access, could communicate with the central server, and walk along given path to perform actual actions.

In the Two-dimensional Code Guidance Model, the work area of AGV is usually divided into many rectangle regions, and a two-dimensional code, which contains the location information and path information to the adjacent regions, is deployed at the intersection of each region. When the AGV walks along a certain path (code sequence) given by the central server, it just need to reach the next point in the path, recognize the two-dimensional code it encountered, acquires the working information (corner degree, speed etc.), obtain the path to the next node, and repeat the same action of the last step, until it reaches the end point.

Compare with traditional guidance model, two-dimensional code guidance model has a lot of advantages, such as low cost, flexible layout, easy controlling of density, easy to modify, having a wider range of application prospects etc.

In view of these advantages, assuming that there is an irregular warehouse, in the cargo-intensive place, the code can be placed more closely, and in the place where the goods cannot put the two-dimensional code, they can be arranged more relaxed, so that the cost of laying down can be controlled, and the working efficiency can be higher.

3 Research Goal and Approach

3.1 Research Goal

As mentioned above, the traditional guidance models for AVG cars have many dis-advantages. So our research goal is to design a novel guidance model based on two dimensional code technology which has the following features: low cost, dynamic path planning, low requirement for external environment.

The core points for the above idea are:

Design of the dynamic path planning algorithm;
Combined with above algorithm and network technology, design the guidance model based on two-dimensional code technology.

3.2 Key Idea

Based on Dijkstra's Algorithm, a new path planning algorithm is proposed to better meet the needs of AGV cars, to better match the actual working environment, and to improve work efficiency.

According to the path determined by the above algorithm, the two-dimensional guidance model is designed so that the AGV cars could walking along the given path with low cost, high flexibility and high efficiency.

3.3 The Path Planning Algorithm

The two-dimensional code guidance technology is a free guidance technology, the travel route can be very flexible, which is different from the traditional guidance technology. Therefore, the path planning algorithm of AVG cars plays a vital role as it should choose a suitable path from the initial point to the target point so that both the total cost and the efficiency can be optimized.

The so-called suitable path selection, not only refers to the length of the path, but also means that the time spent should be as short as possible. Usually the walking path of AGV cannot be a straight line, there may be many factors which influence the final result, such as the speed of AGV, the time to slow down and speed up, the rotation speed and times, all the above factors should be taken into account during the design of the path planning algorithm.

The proposed algorithm is based on the Dijkstra's algorithm, which is a classic single-source shortest path algorithm proposed by Edsger W. Dijkstra. The core idea of Dijkstra's algorithm is to record the distance from the source point to any point in real time by means of the edge expansion method, and it is mainly used to calculate the shortest path from one node in a connected graph to all other nodes. The Dijkstra's algorithm only focused on the sum of the edge weight or the length of the path between the start point and end point, so it is not suitable for AGV application.

In our proposal, we not only consider the length of the path, but also consider the whole time spent which is influenced by the cruising speed of AGV, the turn angle and frequency of AGV, the time of speedup and slowdown of the AGV etc.

The key idea of our proposal algorithm is defined as follow:

For a start point SP(X,Y) and a goal point GP(X,Y), the path between SP and GP can be defined as a set of coordinate PATH = $\{sp(x_1,y_1), pp(x_2,y_2)\ldots pp(x_n,y_n), gp(x_m, y_m)\}$.

STEP 1. In the initial state, the AGV is at SP(X,Y), and PATH = $\{sp(x_1,y_1)\}$.

STEP 2. Set current start point as $sp(x_1,y_1)$, find appropriate points from its adjacent points, which can make the AGV walk close to GP(X,Y), either from the X-axis, or from the Y-axis, or both. If only one point is found, add it to the PATH as pp (x_2,y_2), then PATH = $\{sp(x_1,y_1), pp(x_2,y_2)\}$; If multiple suitable points are found, name the paths as $PATH_1$, $PATH_2$,……, $PATH_n$ respectively, and add the corresponding point to the corresponding path set.

STEP 3. Set the current start point as the last point for each path set, repeat the same behavior as STEP 2, until GP(X,Y) is reached, add it to the path set, and the path can be denotes as $PATH_i = \{sp(x_1,y_1), pp(x_2,y_2)\ldots pp(x_n,y_n), gp(x_m,y_m)\}$.

STEP 4. Repeat the behavior of STEP 3, until all the path reaches GP(X,Y).

STEP 5. For each path found in above steps, compute the time spent for the AGV to run from SP(X,Y) to GP(X,Y) along each found path, then select the path with the shortest time as the final result. In this step, the whole time not only consider the time spent walking along the given path, but also consider the cruising speed of AGV, the turn angle and frequency of AGV, the time of speed up and slowdown of the AGV etc.

STEP 6. The AGV walk along the selected path PATH f = $\{sp(x_1,y_1), pp(x_2,y_2)\ldots pp(x_n,y_n), gp(x_m,y_m)\}$. If the AGV is at $pp(x_i,y_i)$, and encounters an obstacle, then set $pp(x_i,y_i)$ as the new start point, obtain a new path to GP(X,Y), then continue to walk

along the new path until GP(X,Y) is reached; if no path can be found, set $pp(x_{i-1},y_{i-1})$ as new start point and repeat the same action, until a path is found, and then continue to walk along the new path.

In a warehouse deployed with a two-dimensional code, the shortest path from the initial point to the target point can be derived from the Dijkstra algorithm without considering other factors except for the distance problem. However, taking into account the characteristics of two-dimensional code guidance technology, this study will be based on our proposed algorithm as the main algorithm for path planning. First of all, the core of Dijkstra algorithm is the comparison of weights. The weight of our proposed algorithm is no longer a simple distance problem. Instead, it joins the parameters such as vehicle speed and corner time, that is T[S] = T[run timing] + T[rotate]. These parameters together determine the weight problem between two points. And the choice of the optimal path is determined by comparing the weights.

Our Path planning algorithm differs from the Dijkstra algorithm for the following reasons:

The Goal is Different. Our algorithm only needs to find the optimal path from the start point to the end point, but the Dijkstra algorithm aims to find the shortest path from one start point to all other points in a connected graph.

The requirement is Different. In our algorithm, there's a rule, that is, for every step forward, the AGV should be closer to the endpoint either from the x-axis, or the y-axis, or both. But the Dijkstra algorithm doesn't have this requirement.

The Measurement of Cost is Different. In Dijkstra algorithm the total length of edges in the path is its cost. But in our algorithm, the speed of AGV, the time of speed up and slow down, the rotating angle, rotating speed, time of rotation, all the above factors should be taken in account, and is a part of the total cost.

4 Case Study

In practical applications, there may exist two conditions for the AGV in two-dimensional guidance model, one is the AGV get a path without any obstacle, another is that there is one or more obstacles in the path.

For the first condition, once the AGV gets its path from the central server, it just needs to walk along the given path until it reaches the end point.

For the second condition, each time the AGV encounter an obstacle along the given path, it returns to the last point in the path, and asks the central server for a new path, which set the current position of AGV as the start point, and the end point as the initial end point, and walks along the new path, until it reaches the end point.

4.1 Case 1: Path with no Obstacle

In the absence of obstacles, we set the trolley's initial point (3,1) and the target point coordinates (26,14), according to the optimized Dijkstra's Algorithm. The AGV car is simulated with three successive points, the red one denotes the front, blue for the rear,

and black for the body. This can reflect the characteristics of the car in the steering. The working process of the algorithm is simulated by setting the running parameters in Table 2. The horizontal point time in the Table 2 means the time needed for the car between adjacent point in horizontal direction.

Table 2. Setting of AVG parameters

Endpoint coordinate	Horizontal point time	45°	90°	135°	180°	225°	270°	315°	360°
(26,14)	1	1	2	3	4	5	6	7	8

As can be seen from Fig. 2:

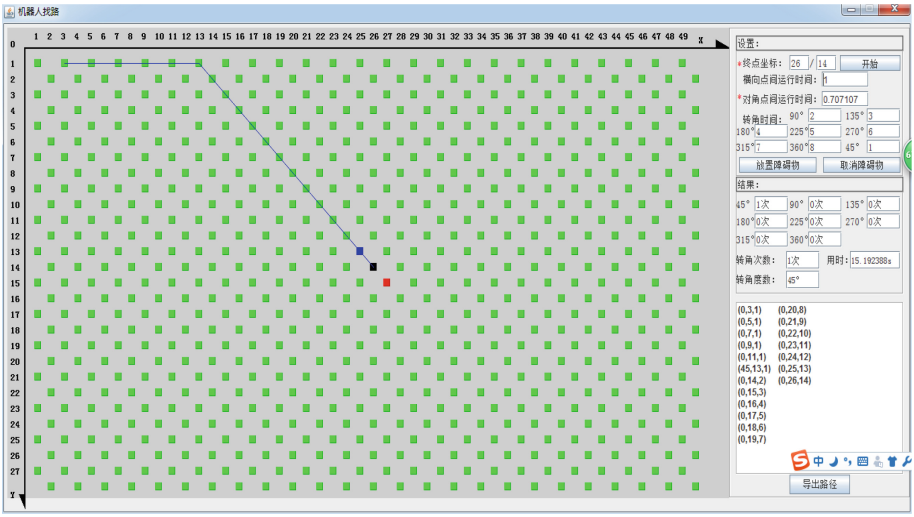

Fig. 2. Simulation of path without obstacle

Starting point: (3,1)

Target: (26,14)

Optimal path: (3,1), (5,1), (7,1), (9,1), (11,1), (13,1), (14,2), (15,3), (16,4), (17,5), (18,6), (19,7), (20,8), (21,9), (22,10), (23,11), (24,12), (25,13), (26,14)

There is one corner in the path, and the turning angle is 45°. If for any reason there exists another corner in the path, the total time spent must be greater than the previous path. Therefore, the path shown in the figure is the optimal path for the car.

4.2 Case 2: Path with Obstacles

The path planning algorithm has the obstacle avoidance function, which could simulate the more actual case. Because in practical case, there may be many goods or shelves

placed here and there in the warehouse, and there may be many obstacles for a given path.

In the case of obstacles, we set the initial point (3,1) and the target point coordinates (23,15), and simulate the real environment by setting different obstacles on the simulation platform. The optimal path of the car is simulated by setting the run parameter the same as in Table 2. The blue line represents the optimal path without obstacle, and the red line denotes the optimal path with one obstacle.

It can be seen from the above Fig. 3 that when there is an obstacle, the car:

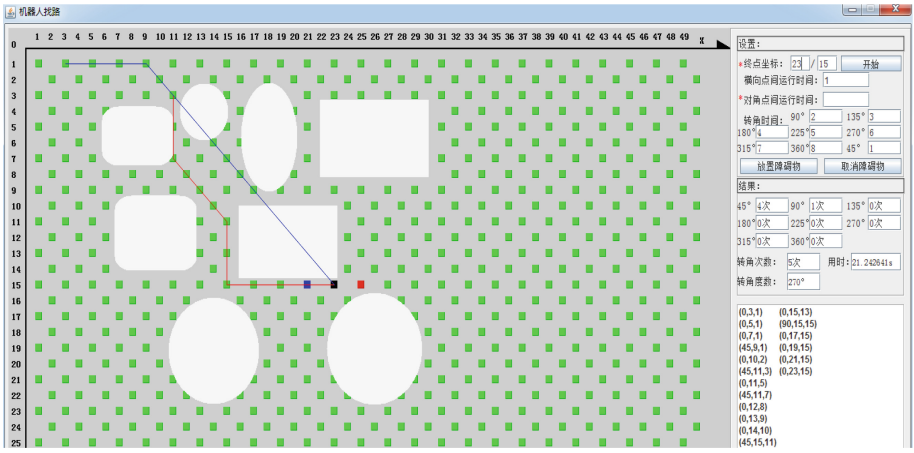

Fig. 3. Simulation of path with obstacles

Starting point: (3,1)

Target: (23,15)

Optimal path: (3,1), (5,1), (7,1), (9,1), (10,2) (11,3), (11,7), (12,8), (13,9), (14,10), (15,11), (15,13), (15,15), (17,15), (19,15), (21,15), (25,13), (23,15).

There're five corners in the path, of which four corner is 45°, and one is 90°. Therefore, it can be seen from the figure that the car has successfully avoided the obstacle and has chosen the shortest path between the running time and the turning time.

4.3 Case Analysis

As show in the figure, case 1 and case 2, we set the distance between the coordinate points in the matrix, including the horizontal distance and the diagonal distance, as well as the setting of the corner time. The purpose of above settings is trying to simulate the actual behavior of the real AGV.

According to the case study, we can obviously find that in the case of with or without obstacles, the proposed algorithm makes the correct judgments, and find the suitable path in both conditions.

Therefore, in practical application, the proposed algorithm can be tested in certain cases, for the logistics industry or large warehouse, to provide more efficient transport tools, improve work efficiency, and acquire more data to further improve the algorithm.

5 Discussion

5.1 Summary

In our previous study and experiment, we successfully proved the operability and practicability of the proposed algorithm. As the experiment data shows, the path planning algorithm could meet the requirement of path planning, and it makes it easier to achieve the previously complicated path planning of AGV under two-dimensional guidance model, and it has the function of obstacle avoidance, which means it could avoid the fixed or mobile obstacles and select alternative paths for the AGV to run along with.

5.2 Further Work

For next step, we plan to further study the influence of different combination of speed, rotation times, laying down method on the algorithm, and find out better combinations for different circumstance. Testing and evaluating the performance of the algorithm in case of obstacles under more complex situation is another topic.

To achieve the above goals, a more complicated and fully functional simulation platform should also be built to help demonstrate and verify the path planning algorithm for AGV, and to deal with more complicated conditions.

Acknowledgements. This work was supported by the National Natural Science Foundation of China under Grant U1604149; and the International Science and Technology Cooperation Project (2015) (152102410053) supported by Science and Technology Department of Henan Province, China.

References

1. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
2. Description of Dijkstra's Algorithm on Wikipedia.org. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm. Accessed 20 Jan 2019
3. Yan, W.: Data Structure (C Language Edition), 1st edn. Tsinghua University Press, Beijing (1997)
4. Wang, H., Huang, Q., Li, C.: Graph Theory and Its MATLAB Implementation, 1st edn. Beijing University of Aeronautics and Astronautics Press, Beijing (2010)
5. Zhou, X.: The shortest path problem and its solution. Comput. Knowl. Technol. **06**, 1403–1405 (2010)
6. Chaorui, W.: Graph Theory, 2nd edn. Beijing Institute of Technology Press, Beijing (1997)
7. Feng, L.: Shortest path algorithm classification system and research progress. J. Surv. Mapp. **3**, 269–275 (2001)

8. Chen, X., Cai, X., Tang, D.: Shortest path algorithm analysis and its application to bus route query. J. Eng. Graph. **3**, 20–24 (2001)
9. Song, X., Yu, L., Sun, H., Li, J.: Advanced path search algorithm in the case of congestion path in traffic network. J. Shenyang Jianzhu Univ. (Nat. Sci. Edn.) **25**(4), 776–780 (2009)
10. Zhang, C., Yang, Y., Zhao, H.: Improvement and implementation of shortest path algorithm based on path dependence. Comput. Eng. Appl. **25**, 56–58 (2006)
11. Li, Q.: An adaptive genetic algorithm. J. Univ. Sci. Technol. Beijing **11**, 1082–1086 (2006)
12. Xie, B., She, X.: Algorithm and Data Structure, 1st edn. Higher Education Press, Beijing (2001)
13. Chen, Y., Lu, X., Ding, H.: Study on optimization strategy of Dijkstra algorithm. Comput. Technol. Dev. **16**(9), 73–75, 78 (2006)
14. Zhang, Y.: Dijkstra shortest path algorithm optimization. J. Nanchang Inst. Technol. **25**(3), 30–33 (2006)
15. Hu, S., Zhang, X., Zhao, Y.: Sector optimization Dijkstra algorithm. Comput. Technol. Dev. **16**(12), 49–51 (2006)
16. Wang, P., Sun, L., Huang, B.: Ground mobile robot system research status and key technology. J. Mech. Des. **23**(7), 1–4 (2006)
17. Wang, S.: Graph Theory, 2nd edn. Science Press, Beijing (2009)
18. Yin, J., Wu, K.: Graph Theory and its Algorithm, 1st edn. Press of University of Science and Technology of China
19. Yao, E., He, Y., Chen, S.: Mathematical Planning and Combinatorial Optimization, 1st edn. Zhejiang University Press, Hangzhou (2001)
20. Zhu, J., Zhang, G.: Mathematical Modeling Method, 1st edn. Zhengzhou University Press, Zhengzhou (2003)
21. Yang, Yu.: Bar code and automatic identification technology. Enterp. Stand. **6**, 35–36 (2002)
22. Jiao, Y., Zhang, C.: Two-dimensional Bar Code Technology, 1st edn. China Price Press, Beijing (1996)