



# Research on navigation of bidirectional A\* algorithm based on ant colony algorithm

Yu-qiang Chen<sup>1</sup> · Jian-lan Guo<sup>1</sup> · Huaide Yang<sup>1</sup> · Zheng-qin Wang<sup>2</sup> · Hong-ling Liu<sup>2</sup>

Published online: 30 May 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

## Abstract

To overcome the lengthy search time, massive space occupation, and overlong planned path of the traditional A\* algorithm, this paper integrates the bidirectional search with the intelligent ant colony algorithm to obtain the heuristic function selection factor, and uses the factor to improve the evaluation function of the algorithm. The simulation results show that the improved algorithm achieved better dynamic navigation than the traditional A\* algorithm both in search time and distance, featuring shorter path searching time and the algorithm running time. Therefore, the result of this research has effectively reduced the search time and enhanced the dynamic search.

**Keywords** Heuristic function · A\* algorithm · Path planning · Bidirectional search · Ant colony algorithm (ACA)

---

✉ Yu-qiang Chen  
chenyuqiang@126.com

Jian-lan Guo  
rachel0519@126.com

Huaide Yang  
215315201@qq.com

Zheng-qin Wang  
32017687@qq.com

Hong-ling Liu  
yjxagwy@126.com

<sup>1</sup> Department of Computer Engineering, Dongguan Polytechnic, Dongguan, China

<sup>2</sup> Department of Information, University of Guangzhou Nanyang Polytechnic, Guangzhou, Guangdong, China

## 1 Introduction

There are two main methods to alleviate urban traffic congestion: increase the traffic throughput by building more urban roads, and improve traffic efficiency by developing an intelligent transport system (ITS). Facing the increasing scarcity of urban land, the most pragmatic solution is to pursue optimal network operation and management, and better navigation in the individual aspects with intelligent learning. The solution is bound to realize obvious decline in the total cost [1]. The optimal path planning is particularly important to the optimization of urban road network. In essence, it is a shortest path algorithm based on the topology of the road network. The path planning looks for the path with minimal total cost (i.e. the shortest path, the least time, and the lowest cost) from the start point to the destination point [2].

Much research has been done on path planning at home and abroad. Based on the GIS technology, Yan put forward an improved A\* algorithm of dynamic K-shortest path by establishing a dynamic travel time estimation model and integrating the model into the shortest path algorithm [3]. Yang and Jiang proposed a path planning algorithm based on fluid neural network, pointed out the linear growth relationship between the calculation time and the number of additional nodes, and realized substantial decline in the time consumed in path calculation [4]. Zhou and Wang introduced statistics into the evaluation function to generate the automatic location searching algorithm (ALSA), which overcomes the slow calculation and poor practicability of the traditional algorithm [5]. Smiths et al. suggested to improve the search efficiency of the A\* algorithm data nodes with the open and close tables of an orderly binary tree [6]. Developed by Szczerba, the sparse A\* search (SAS) algorithm can effectively reduce the search space in combination with the constraint conditions [7]. Geng and Li presented the application method of parallel reinforcement learning based on artificial neural network after adjusting the weights based on the feedbacks of the artificial neural network, and combining the high learning efficiency of parallel reinforcement learning algorithm [8]. Pan described how to use the A\* algorithm to find the reasonable path between two points on the static grid search map [9].

At present, there are quite a number of heuristic search algorithms, such as the local optimum search, the best-first search, and the A\* algorithm first proposed by Hart, Nilsson, Raphael et al. By introducing the evaluation function, the A\* algorithm speeds up the search and elevates the precision of the local optimum search. Compared to the Dijkstra's algorithm, the A\* algorithm occupies a smaller storage space and has a lower time complexity  $O(b, d)$ , where  $b$  is the average node degree, and  $d$  is the search depth of the shortest path from the beginning to the end [10]. However, more studies are grounded on static factors and the shortest path. Based on the intelligent ant colony algorithm (ACA), this paper digs deep into the evaluation function and improves the search efficiency by bidirectional search, seeking to provide adaptive navigation under dynamic traffic conditions.

At present, many researches are based on ant colony algorithm in routing selection of wireless sensor network [11–21], and some ant colony algorithms are

applied to route planning of vehicle navigation [22–26]. A\* algorithm is widely used in route planning only considering distance [27–32], and literature [33–42] uses neural network and ant colony algorithm, particle swarm algorithm to study route planning, literature [43–56]; the artificial bee colony algorithm is used to optimize the path problem, but few literatures use the combination of ant colony algorithm and A\* in the path planning; especially in the ant colony algorithm, the actual road condition is introduced as the influence factor. In this paper, the traditional A\* algorithm is improved by two-way search. By introducing the intelligent algorithm heuristic function selection factor, the algorithm evaluation function is improved, which effectively reduces the search time and enhances the search dynamics. Finally, the feasibility of the algorithm is verified by an example.

## 2 Basic principles of the A\* algorithm

There are three goals for path planning in vehicle navigation system: the shortest path (graph theory), the least cost, and the shortest time. Considering the driver's habits, the more recent research usually pursues all three goals simultaneously. In the course of travelling, the driver is often confronted with various unexpected traffic incidents. In this case, the vehicle cannot drive along the pre-set route, calling for recalculation of the optimal route. The real-time vehicle renavigation requires fast and efficient computing by the planning algorithm [22]. The heuristic search is a good choice under this situation. This strategy guides the search along the potential directions, and speeds up the determination of the optimal solution by adding related heuristic information into the searching process. The search is knowledge-based if the heuristic strategy is integrated into a computerized search algorithm. Using a selected evaluation function, each step in the searching process aims at finding the evaluation function node with the minimum cost, and taking it as the extension node for the next step. The main advantage of the heuristic search strategy lies in that the search can be limited to a certain range. In the heuristic search, the position estimation method must be selected properly [50–54].

Currently, the A\* algorithm is the most popular and effective heuristic search algorithm for path optimization. The algorithm is innovative in that the known global information is introduced to select the next node, and the possibility of the node being in the best route is evaluated against the estimated distance between the current node and the target node, making it possible to search for the closest node, and improve searching efficiency.

- (1) The key of the A\* algorithm is to establish the heuristic function. Let  $f'(v)$  be the heuristic function of the current node  $v$  and express it as:

$$f'(v) = g(v) + h'(v) \quad (1)$$

where  $g(v)$  is the function of the actual cost from the original node  $O$  to the current node  $v$ ;  $h'(v)$  is the estimated minimum cost from the current node  $v$  to the destination node  $D$ . If there is no heuristic information, i.e.  $h'(v) = 0$ , then

the A\* algorithm is a common Dijkstra's algorithm. If  $h'(n)$  makes selections,  $h'(n)$  must be smaller than or equal to the actual minimum cost  $h^*(n)$  from node  $v$  to the destination node  $D$  so that the estimated minimum cost of the current node is not too high. The algorithm first searches for the node of the minimum  $f'(v)$  value. In the experiment,  $h'(n)$  was defined as the Euclidean distance from node  $v$  to the destination node  $D$ . The A\* algorithm is able to compute the optimal path, as long as the original problem does have the optimal solution.

Denote the coordinates of the original point  $O$  as  $(O_x, O_y)$ , the coordinates of the destination point  $D$  as  $(D_x, D_y)$ , and the coordinates of the centre point  $M$  as  $(M_x, M_y)$ . Then, the heuristic function can be expressed as:

$$f'(v) = g(v) + \sqrt{(D_x - M_x)^2 + (D_y - M_y)^2} \quad (2)$$

where  $g(v) = \sum_{t=1}^n d(t)$ .  $d(t)$  refers to the distance between the two adjacent nodes.

(2) The detailed procedure of the A\* algorithm is as follows:

- ① Suppose the initial open table only contains the original node, and make the close table empty by setting the cost of the original node to 0, i.e. the  $g$  value is 0; Set the cost of other nodes as  $\infty$ .
- ② The open table fails if it is empty. If not, identify the node of minimum  $f'(n)$  value in the open table, and denoted it as the best node. Then, move the best node into the close table to see if it is the target node. If yes, go to Step ③.
- ③ Otherwise, extend the best to generate successor nodes according to the map database and node information. Treat each successor node  $v$  with the following procedure:
  - A. Calculate the cost of node  $v$  by summing up the sum of the cost of  $g(v) = \text{best}$  and the cost from the best node to  $v$ .
  - B. If  $v$  is the same as any node in the open table, check if  $v$  has the lowest  $g$  value. If yes, replace the node cost with the cost of  $v$  in the open table, and turn the backward pointer of the matching node towards the best node.
  - C. If  $v$  is the same as any node on the close table, check if  $v$  has the lowest  $g$  value. If yes, replace the node cost with the cost of  $v$  in the open table, turn the backward pointer of the matching node towards the best node, and move the matching node to the open table [11–14, 55, 56].
  - D. If node  $v$  falls neither in the open table nor in the close table, turn the backward pointer of node  $v$  towards the best node, and put node  $v$  into the open table. Calculate the evaluation function  $f'(v)$  of node  $v$ , and repeat Step ②.
- ④ Traverse the node from the backward pointer of the best node to the original node to complete building the corresponding path. The traversal process is also called back tracing from the best node.

Rather than traverse all the nodes, the A\* algorithm introduces the heuristic information to guide the moving towards the target node, thereby facilitating and accelerating the computation. However, the deletion of numerous nodes in the searching process may exert negative effect on the computation. Due to the complexity of experience and practice, it is evitable for the cost function which introduces the heuristic information to have some mistakes. Thus, some of the deleted nodes may happen to be the optimal node. In this case, the final result may be suboptimal, or the search may fall into a “dead cycle”, failing to obtain the target node [23].

### 3 Improvement of the A\* algorithm

As a heuristic search algorithm, the A\* algorithm assesses the cost from the current node  $n$  to the target node using the heuristic function, that is, to evaluate the path length between the two nodes. According to Eq. (1) and the previous analysis, the heuristic function value of the A\* algorithm is no greater than the actual minimum cost. In the grid search map, the heuristic function  $h'(n)$  mainly measures the Manhattan distance from the current node to the target node [24]:

$$h(n) = |v_d \cdot x - v_n \cdot x| + |v_d \cdot y - v_n \cdot y| \quad (3)$$

where  $v_d \cdot x$  is the x-coordinate of the target node;  $v_n \cdot x$  is the x-coordinate for the current node;  $v_d \cdot y$  is the y-coordinate for the target node;  $v_n \cdot y$  is the y-coordinates for the current node.

In many environment models,  $H(n)$  is not necessarily smaller than or equal to  $h(n)$  in Eq. (3). Therefore, the Euclidean distance was taken as the heuristic function (e.g. Eq. (2)). The problem is that the shortest path might not be the optimal path or the least time-consuming one, under the effect of traffic, road resistance, road conditions and other factors in the routing process. Generally, path planning algorithms search from the original point to the destination at a rather slow speed. The search time may be shortened drastically by searching in two directions. Following this train of thought, the author developed the bidirectional search algorithm A\*, which searches both from the original point to the destination and from the destination to the original point, and terminates the search when the current nodes in the two search directions coincide with each other. The bidirectional search can calculate the shortest path between two nodes in a nonnegative weight network [25]. Moreover, the various dynamic influencing factors on traffic conditions should be considered in the training of the  $h(x)$  [26]. This paper suggests using the ACA to train the  $h(x)$ , aiming to improve the information content of the function.

#### 3.1 ACA

##### 3.1.1 The principle of the ACA

The ACA aims to search for an optimal path in a graph, based on the behaviour of ants seeking a path between their colony and a source of food. In the cooperative

foraging process, the path selection is heavily influenced by the pheromone, a hormone secreted by ants to send and receive feedbacks. The ants tend to choose the path with a high concentration of pheromone. The probability that a path is chosen is positively correlated with the pheromone concentration on that path. Besides, the pheromone concentration of selected path will increase after the ants move through the path. The above is the so-called positive feedback mechanism, which reduces the total pheromone amount needed on the path. The shortest path can be pinpointed solely based on the feedbacks of local pheromone. The ant colony behaviour constantly changes with the environment. When the current optimal path is blocked, the ants can quickly find another optimal path in the current environment [29–35].

### 3.1.2 The basic model

The ACA model was expounded with the classical travelling salesman problem (TSP) as an example. The TSP asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? To answer the question, the ACA-based mathematical model was built as follows:

The pheromone concentration of each path remained constant at the initial time  $t=0$ , that is,  $\tau_{ij}(t_0) = C$ , where  $\tau_{ij}(t_0)$  is the pheromone concentration of the road section  $(i, j)$  at the  $t_0$ , and  $C$  is a constant. Suppose there were  $M$  ants that choose the moving direction according to the pheromone concentration, then the state transition probability of ants at time  $t$  is:

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)\eta_{ij}^\beta(t)}{\sum_{S \in \text{allowed}_k} \tau_{is}^\alpha(t)\eta_{ij}^\beta(t)}, & S \in \text{allowed} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where  $\eta_{ij}$  is the visibility factor negatively correlated with the distance between two points;  $\alpha$  is the impact degree of the total pheromone concentration on ants behaviour;  $\beta$  is impact degree of the path length on ants behaviour;  $\text{allowed}_k = [C - ta - bu_k]$  are the cities out of the TabuList, which is a collection of cities ready for selection.

The TabuList was introduced because artificial ants possess many exclusive features that real ants do not have. To avoid repeated selection, all the cities selected in the current cycle was recorded in the TabuList, and released after the end of the cycle. At the beginning of the next cycle, the ants were able to choose from all the cities.

After  $n$  moments, when the  $M$  ants had traversed all the cities, the pheromone on each path changed according to Eqs. (5) and (6):

$$\tau_{ij}(t + 1) = (1 - \rho)\tau_{ij}(t) + \Delta\tau_{ij}(t, t + 1). \tag{5}$$

$$\Delta\tau_{ij}(t, t+1) = \sum_{k=1}^m \Delta\tau_{ij}^k(t, t+1) \quad (6)$$

where  $\rho$  is the pheromone volatility ( $\rho \in (0, 1)$ );  $\Delta\tau_{ij}^k$  is the total amount of pheromone that ant  $k$  released on the path  $(i, j)$  in the current cycle;  $\Delta\tau_{ij}$  is pheromone increment on path  $(i, j)$  in each cycle.

The pheromone can be updated by various methods. Dorio [27] presented three different models: ant-density system, ant-quantity system and ant-cycle system. The first two models, using the local information, belong to the category of local update, while the last model, using the global information, falls in the category of global update. Featuring high consistency with the behaviour of real ants, and well-demonstrated performance, the ant-cycle system was taken as the basic model [27, 28]:

$$\Delta\tau_{ij}^k(t, t+1) = \begin{cases} \frac{Q}{L_k}, & \text{(Ants pass through the path } i, j) \\ 0 & \text{Otherwise} \end{cases} \quad (7)$$

where  $Q$  is a constant about the pheromone released by each ant in the cycle;  $L_k$  is the total length of all the road sections that ant  $k$  passes through in the cycle.

### 3.1.3 The basic steps of the ACA

- Step 1 Parameter initialization: Set the cycles times  $T=0$ ; denote the maximum cycle times as Max and the pheromone constant as  $Q$ ; put  $n$  ants in the original node; initialize the pheromone value on each side  $\langle i, j \rangle$  of the directed graph as  $FI=q$ , where  $q$  is a constant; define the initial time as  $t=0$ .
- Step 2 The cycle times of the algorithm is  $T=T+1$ .
- Step 3 Initialize the TabuList of the ant population.
- Step 4 The number of ants in the algorithm is  $m=m+1$ .
- Step 5 The ants select the next node according to the next node formula.
- Step 6 Add the selected node to the TabuList.
- Step 7 If  $m < n$ , go to Step 4; otherwise, go to Step 8.
- Step 8 Update the pheromone on the topology.
- Step 9 If  $T < \text{Max}$ , go to Step 2; otherwise, terminate the algorithm.

## 3.2 ACA-based bidirectional A\* algorithm

In this research, the intelligent ACA was adopted to estimate the cost of nodes during the bidirectional search. In the A\* algorithm, the nodes need to be evaluated when it is extended. Nevertheless, the algorithm only considers the static factors, ignoring the dynamic factors of road traffic. The resultant huge gap between  $h(x)$  and  $h'(x)$  affects the computing efficiency. Since the computing efficiency is positively correlated with the proximity between  $h(x)$  and  $h'(x)$ , the ACA was adopted to train the  $h(x)$ . The ACA improves the computing efficiency, as it adds to the dynamism of the algorithm, incorporates various dynamic factors, and brings the  $h(x)$  closer to

the actual optimal value. Despite the upside, the ACA is easily trapped in the local optimum. To address the defect, the concurrent reward-based ACA was developed to prevent the trap and increase the convergence speed. The concurrent means that each ant has multiple backups. When ant  $K$  is looking for the path from the original node, its backup ants are doing the same. The reward refers to the additional pheromone rewarded to the optimal path, and the pheromone rewarded to the path adjacent to the optimal path. The global convergence of the algorithm is improved by taking the non-optimal path into account. The additional pheromone rewarded to the current optimal path is as follows [36–42]:

$$\Delta \ell_{ij}^*(v_j) = \rho' \Delta \ell_{ij}(v_j) \tag{8}$$

where  $\Delta \ell_{ij}^*(v_j)$  is the extra rewarded pheromone;  $\rho'$  is the reward factor ( $0 < \rho' < 1$ );  $v_j$  is a collection of nodes.

The pheromone update formula is as follows:

$$\ell_{ij}(v_j) = \rho \times \ell_{ij}(v_j) + \Delta \ell_{ij}(v_j) + \Delta \ell_{ij}^*(v_j) \tag{9}$$

The pheromone rewarded to the adjacent path is expressed as:

$$\Delta \ell_{ij}^*(v_j) = (\rho')^m \cdot \Delta \ell_{ij}(v_j) \tag{10}$$

where  $m$  is reward coefficient ( $m \in N$ ), which is positively correlated with the distance between the optimal path and the adjacent path.

Suppose there were  $m$  artificial ants moving at the speed of  $C_k$  in the system of  $n$  nodes, and ant  $k$  was looking for the optimal path. Then, the current nodes was evaluated and the heuristic function  $h(x)$  were trained by the bidirectional A\* algorithm based on concurrent reward ant colony system (DA\*BCRAS).

Initialize the pheromone on the edge of the network topology as  $\langle V_i, V_j \rangle$ , and we have

$$\ell_{ij}(v_j) = \text{const} \tag{11}$$

where  $\text{const}$  is a constant;  $\Delta \ell_{ij}(v_j) = 0$ . Thus,  $\eta_{ij}(v_j) = 1/d_{ij}$ , where  $d_{ij}$  is the distance from node  $V_i$  to node  $V_j$  and  $\eta_{ij}$  is the amount of local heuristic information. Then, the following equation holds:

$$\mu_{ij}(v_j) = 1 / \left( \frac{|TC_{ij}^{\text{max}} - TC_{ij}^{\text{design}}|}{|TC_{ij}^{\text{average}} - TC_{ij}^{\text{design}}|} \cdot \frac{L_{ij}^2 \rho_{ij}}{C_k} \right) \tag{12}$$

where  $\mu_{ij}(v_j)$  is the heuristic information function;  $TC_{ij}^{\text{max}}$  is the maximum hourly traffic flow;  $TC_{ij}^{\text{design}}$  is the designed hourly traffic flow;  $TC_{ij}^{\text{average}}$  is the average hourly traffic flow;  $C_k$  is the current speed;  $\rho_{ij}$  is the current traffic density;  $L_{ij}$  is the path distance. With the maximum number of cycles of the ACA denoted as  $N_{\text{max}}$ , the DA\*BCRAS was implemented through the following steps:



- (1) Let  $v_o$  be the original node and  $v_D$  be the target node. Search the network topology from the original node by the bidirectional A\* algorithm. Suppose the nodes  $v_i$  are not the target node and need to be extended for the next cycle. Denote the collection of such nodes as  $v_j$ , which is expressed as:

$$f'(v_j) = g(v_j) + h'(v_j) \tag{13}$$

- (2) Use the concurrent reward ant colony system to train the  $h'(v_j)$ , and start evaluation from the current node  $v_{j_x}$  to the target node ( $v_j \in v_j$ ). Put  $m$  ants and their  $k$  backup ants in node  $v_{j_x}$ , and perform selection with the equation below [31]:

$$p_{ij}^k(v_{j_x}) = \begin{cases} \arg \max \{ [\ell_{ij_x}^k(v_{j_x})]^\alpha [\eta_{ij_x}^k(v_{j_x})]^\beta \} \dots (1) & q \leq q_0 \\ \frac{[\ell_{ij_x}^k(v_{j_x})]^\alpha [\eta_{ij_x}^k(v_{j_x})]^\beta [\mu_{ij_x}^k(v_{j_x})]^\gamma}{\sum_{x=1}^n [\ell_{ij_x}^k(v_{j_x})]^\alpha [\eta_{ij_x}^k(v_{j_x})]^\beta [\mu_{ij_x}^k(v_{j_x})]^\gamma} \dots (2) & \text{otherwise} \end{cases} \tag{14}$$

where  $q_0$  is a fixed value;  $q$  is the random number generated by the random generator. If  $q \leq q_0$ , select the next node according to Eq. (1). Otherwise, select the next node according to Eq. (2).

- (3) Compared to its backup ants, each ant can choose a minimum cost path after one cycle. The pheromone is updated when the ants select the path:

$$\ell_{ij}(v_j) = (1 - \rho)\ell_{ij}(v_j) + \Delta\ell_{ij}^k(v_j) \tag{15}$$

where  $\rho$  is pheromone volatility;  $\Delta\ell_{ij}^k(v_j)$  is the total amount of pheromone that ant  $k$  released on the path.

$$\Delta\ell_{ij}^k(v_j) = Q/d_{ij} \tag{16}$$

- (4) When all  $m$  ants arrive at the destination node, select the optimal path, i.e. the minimum cost path obtained through global pheromone update in Step (3), and reward the additional pheromone following the equation below.

$$\ell_{ij}(v_j) = (1 - \rho)\ell_{ij}(v_j) + \Delta\ell_{ij}(v_j) + \Delta\ell_{ij}^*(v_j) \tag{17}$$

where  $\Delta\ell_{ij}^*(v_j) = \rho' \Delta\ell_{ij}(v_j)$ , where  $\rho'$  is the pheromone reward factor;  $\Delta\ell_{ij}^*$  is the amount of pheromone reward.

$$\Delta\ell_{ij} = \sum_{k=1}^m \Delta\ell_{ij}^k \tag{18}$$

where  $\Delta\ell_{ij}$  is the sum of the pheromone that  $k$  ants released on the path;  $\Delta\ell_{ij}^k$  is the pheromone that ant  $k$  released on the path.  $\Delta\ell_{ij}^k = Q/L$ , where  $L$  is the distance covered by the optimal path in a cycle at the arrival at the target node [32]. Meanwhile, it is necessary to reward additional pheromone to the adjacent path:

$$\ell_{ij_y}(v_{j_y}) = (1 - \rho)\ell_{ij_y}(v_{j_y}) + \Delta\ell_{ij_y}^*(v_{j_y}) \tag{19}$$

where  $\Delta\ell_{ij_y}^*(v_{j_y}) = (\rho')^n \Delta\ell_{ij_x}(v_{j_x})$ ;  $\Delta\ell_{ij_y}^*(v_{j_y})$  is the amount of pheromone reward;  $n$  discloses the positive correlation between the reward value and the distance from the optimal path.

- (5) When the number of cycles reaches  $N_{\max}$ , compare the optimal paths obtained in each cycle, and select the one with the minimum cost. The cost surpassing the edge  $\langle v_i, v_j \rangle$  is:

$$T = \frac{L_{ij}}{C_k} + p_{ij}T_{ij} + pT \tag{20}$$

The estimated cost of the optimal path is calculated by the following equation:

$$h(v_j) = \text{Min} \sum T \tag{21}$$

where  $T$  is the cost of each road section on the optimal path.

- (6) In order to find the optimal path of the other nodes, the estimated cost of each node is obtained through the above steps:

$$f(v_{j_{od}}) = g(v_{j_{od}}) + h(v_{j_{od}}) \tag{22}$$

where  $v_{j_{od}} \in v_j$ . Insert the nodes into the OPEN' table, and rank them in descending order by the evaluated value. Then, carry out the bidirectional A\* search as follows:

- (7) Store the original point in the forward search  $\text{Open}_1$  table, and denote the original node  $O$  as the "Least node"; then, set the cost of  $O$  as  $g_1(O) = 0$ , and the cost of the other nodes as infinitely large.
- (8) After the extension of the original node, carry out the following operations on all successor nodes  $n$ :
  - Calculate the cost of each successor node  $v$  by the concurrent reward-based ACA:

$$f'_1(v) = g_1(v) + h'_1(v) \tag{23}$$

- Set the search status of the node to "Noleast";
- Turn the backward pointer of node  $v$  to the original point  $O$ ;
- Insert node  $v$  into the forward search  $\text{Open}_1$  table, namely  $\text{insert}_1(n)$ .
- (9) If  $\text{Open}_1$  table is empty, error message pops up and the search terminates; Otherwise, remove node Min which has the minimum  $f'_1$  cost from the table  $\text{Open}_1$ , and change the search status of node  $v$  to "Least". Then, determine if node Min satisfies the two termination conditions mentioned above. If the conditions are satisfied, jump to the last step; If not, perform the following steps on all the successor nodes of Min:

- Calculate the cost of each successor node  $v$  by Eq. (23) of the concurrent reward-based ACA;
- If the search status of node  $v$  is Most, turn the backward pointer of node  $v$  to the Min node, and insert node  $v$  into  $\text{Open}_1$  table;
- If the search status of the node  $v$  is “Noleast”, calculate the new cost of node  $v$  :

$$g'_1(v) = g_1(\text{Min}) + M(\text{Min}, v) \quad (24)$$

where  $M(\text{Min}, v)$  is the function of the actual cost from node Min to node  $v$ . If  $g'_1(v) < g_1(v)$ , let  $g_1(v) = g'_1(v)$ . If the backward pointer of node  $v$  points at node Min, update  $f'_1(v)$  of node  $v$  in the  $\text{Open}_1$  table; otherwise, proceed with the following steps.

- If the search status of node  $v$  is “Least”, then:

$$g'_1(v) = g_1(\text{Min}) + M(\text{Min}, v) \quad (25)$$

If  $g'_1(v) < g_1(v)$ , let  $g_1(v) = g'_1(v)$ . If the backward pointer of node  $v$  points at node *Min*, set the search status of node  $v$  as “Nonleast”, and insert node  $v$  into  $\text{Open}_1$  table;

- Determine whether the steps of forward searching is 3. If yes, go to Step (10) (the first switch) or Step (12) (the second switch); otherwise, go to Step (9).
- (10) Handle  $\text{Open}_2$  in the same way as in Step (7);
  - (11) Handle  $\text{Open}_2$  and  $\text{Close}_2$  in the same way as in Step (8);
  - (12) Handle  $\text{Open}_2$  and  $\text{Close}_2$  in the same way as in Step (9);
  - (13) To calculate the cost of the optimal path, determine the termination node  $K$  of the bidirectional search in accordance with the above termination conditions. Furthermore, conduct two backtrackings, including the path from the original node  $O$  to the termination node  $K$  and from the termination node  $K$  to the destination node  $D$ , connect the two path, and complete the computing of the optimal path [43–49].

### 3.3 The flow chart of the DA\*BCRAS algorithm

In the research of DA\*BCRAS algorithm, the dynamic traffic parameters were introduced to the mathematical model to mirror the real-time traffic conditions, making it possible to identify the optimal path in actual navigation conditions. Besides, the algorithm convergence was accelerated with the design of backup ants, and the local optimum was avoided by rewarding extra pheromone to the non-optimal path. Capable of finding the optimal path as long as it exists, the bidirectional A\* algorithm was made more dynamic by the ACA. Next, the bidirectional search was employed to further enhance the convergence and effect of the algorithm. In the following pseudo-code as Table 1,  $G$  represents the input of the graph,  $S$  represents the starting vertex,  $T$  represents the target vertex,  $F$  represents the evaluation function of improved ant colony search,  $P$  represents the positive direction,  $N$  represents the negative direction, and  $H$

**Table 1** Ours A\* algorithm pseudo-code

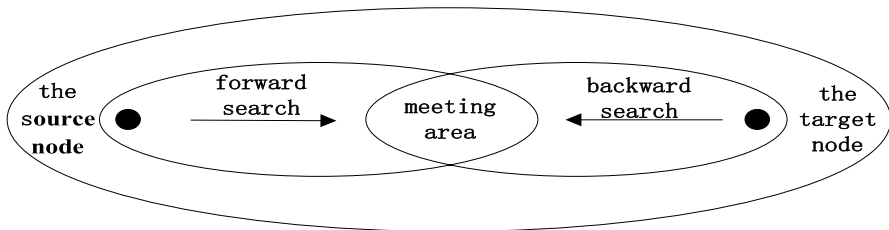
---

Algorithm: Double AStar(G,S,T,F)

---

- (1) Initialize\_Source(G,S),
- (2) CLOSE  $P \leftarrow \emptyset$ ; CLOSE  $N \leftarrow \emptyset$ ;
- (3) Insert(OPEN P, S); Insert(OPEN N,T)
- (4) while CLOSE P  $\cap$  CLOSE N =  $\emptyset$
- (5) Do if open P =  $\emptyset$  or open N =  $\emptyset$
- (6) Return FALSE
- (7) else
- (8) u1=Extract\_Min(Open P, F);
- (9) u2=Extract\_Min(Open N, F);
- (10) if H(u1)>H(u2) then // Forward search
- (11) do Insert (close P, u1)
- (12) for each vertex  $v \in Adj[u1]$
- (13) do Relax (u1,v, F)
- (14) Else
- (15) do Insert(CLOSE N, u2)// Negative direction search
- (16) for each vertex  $v \in Adj[u2]$
- (17) do Relax (u2,V,F)
- (18)  $u \leftarrow \text{Extract\_Min}(\text{CLOSE P} \cap \text{CLOSE N, F+F})$  // Optimal intersection vertex of two-way search
- (19) GetpathValue(u)
- (20) ReturnPath  $(u, G + G')$
- (21) Return TRUE

---



**Fig. 1** Bidirectional A\* schematic

represents the ant colony selection path in each two nodes. The flowchart of the proposed algorithm is shown in Figs. 1 and 2.

### 4 Simulation

In order to verify the performance of the DA\*BCRAS algorithm, the author conducted a simulation experiment in Guangzhou, China (Tables 2 and 3), and compared the results of the DA\*BCRAS algorithm and the A\* algorithm as shown in Figs. 3 and 4. Figure 5 shows the result of simulation and comparison of a map

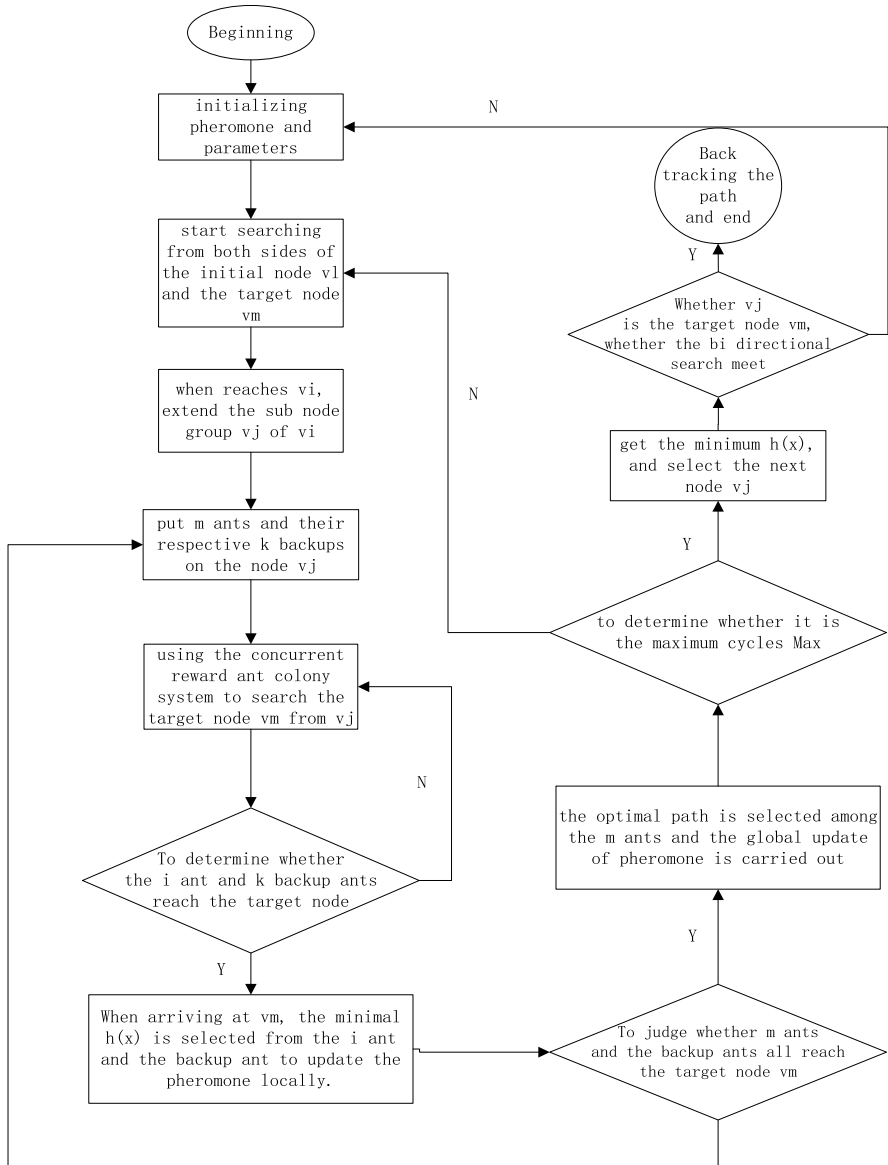


Fig. 2 Flowchart of the algorithm

topology of a certain area of Guangzhou with Dijkstra, traditional A\* and ours bidirectional A\* algorithm.

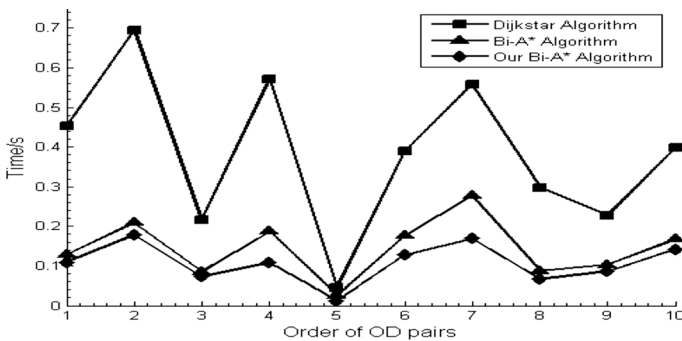
The computer configuration of the bidirectional A\* algorithm experiment in this paper is as follows: AMD Ryzen7 PRO 1700 eight-core Processor 3.0 GHz, Memory:8 GB DDR3 1 333 MHz, Disk:WDC SATA 500 GB 5 400 r/min 8 MB,

**Table 2** CPU time of ten groups OD pairs in Guangzhou city unit s

Path	CPU time		
	Dijkstra algorithm	Bi-A* algorithm	Our Bi-A* algorithm
1	0.453	0.128	0.109
2	0.694	0.209	0.178
3	0.216	0.084	0.073
4	0.572	0.187	0.108
5	0.046	0.0261	0.0128
6	0.389	0.176	0.128
7	0.559	0.278	0.169
8	0.298	0.087	0.066
9	0.227	0.103	0.086
10	0.398	0.167	0.142

**Table 3** Result of ten groups OD pairs in Guangzhou city unit km

Path	Distance		
	Dijkstra algorithm	Bi-A* algorithm	Our Bi-A* algorithm
1	1.000	1	1.000
2	2.710	2.750	2.850
3	0.520	0.560	0.660
4	4.860	4.900	4.900
5	2.600	2.730	3.030
6	1.400	1.550	1.540
7	1.600	2.000	2.200
8	1.200	1.180	1.150
9	1.800	1.920	2.120
10	1.850	1.900	2.100



**Fig. 3** Comparison of algorithm efficiency

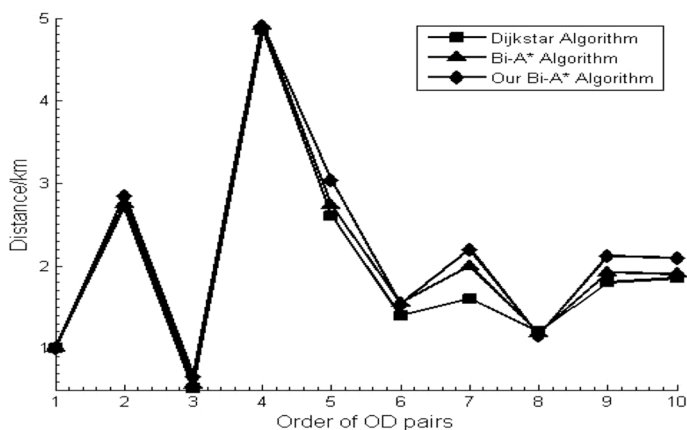


Fig. 4 Comparison of algorithm result

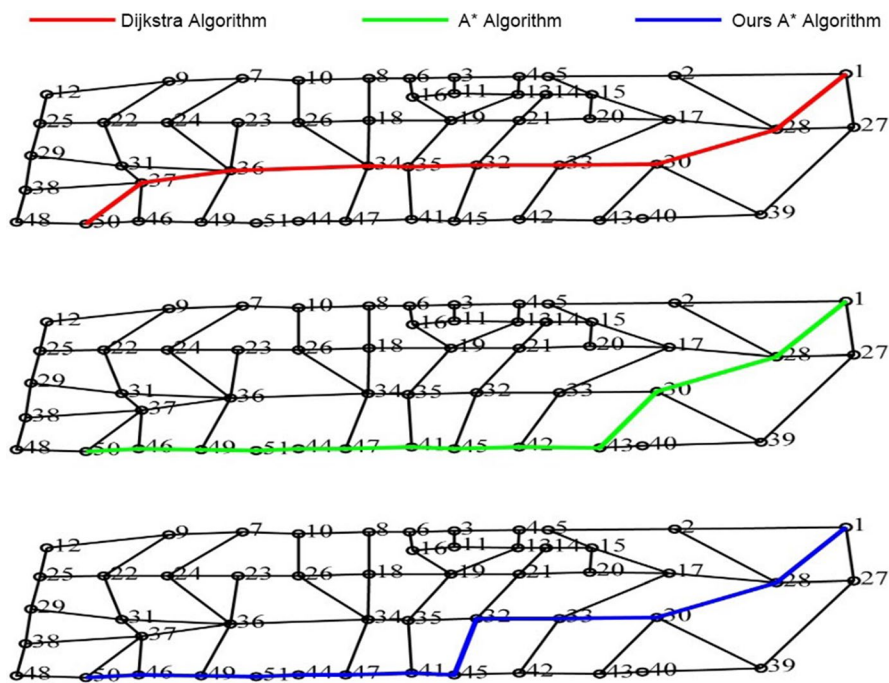


Fig. 5 Comparison of the search path among three algorithms

computer operating system is Windows 7, development environment are MATLAB 2007 and Arc GIS 10.

First, the author sets up a road traffic network diagram and established the rectangular coordinates. Taking the Euclidean distance between two points as the

road weight, the traffic network was mapped to the coordinate system on the assumption that the speed is consistent on any road. Next, the original point and the destination point were selected in the road traffic network diagram, and the path planning was performed in the current network diagram after the confirmation. Thus, the author obtained the optimal path between the original point and destination point.

## 5 Conclusions

After analysing the advantages and disadvantages of the A\* algorithm and the ACA, the author developed the DA\*BCRAS, which makes the A\* algorithm more dynamic and speeds up the convergence of the algorithm. By introducing backup ants and additional pheromone reward, the algorithm effectively jumps out of the local optimum trap and finds the global optimal path. Simulation results show that the DA\*BCRAS can accelerate path planning, shorten the search time, and increase the chance of pinpointing the optimal path. Nevertheless, the endless emergence of dynamic factors adds to the complexity of the algorithm and hinders the computing efficiency. In the future research, the algorithm will be further improved in consideration of the driver's habits, aiming to reduce the congestion rate and achieve real-time dynamic navigation.

**Acknowledgements** The work described in this paper was supported by Special Fund for Science and Technology Innovation Cultivation for College Students in Guangdong Province. The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through Research Group No. RG- 1441-331.

## References

1. Wen MF (2013) An increment searching based on multi-objective path guidance method in intelligent transportation. *J Hunan Univ (Nat Sci)* 5:55–60
2. Tian MX (2009) Application research on path planning in vehicle navigation system. Beijing Jiaotong University, Beijing, pp 32–35
3. Yan KF (2001) Algorithm for dynamic K shortest-paths in vehicle navigation system based on GIS. *J Xi'an Highw Univ* 1:64–67
4. Yang ZS, Jiang GY (1999) A study on the application of fluid neuron network under nonstandard condition. *Syst Eng Theory Pract* 8:55–57
5. Zhou HH, Wang XG (1997) A study of the automatic vehicle location algorithm. *J South China Univ Technol (Nat Sci)* 25:114–115
6. Szczerba RJ (2000) Robust algorithm for real-time route planning. *IEEE Trans Aerosp Electron Syst* 36:869–878
7. Geng XL, Li CJ (2011) Application of parallel reinforcement learning based on artificial neural network to adaptive path planning. *Sci Technol Eng* 11:756
8. Pan HB (2009) A\* algorithms to find the path. *Technol BBS* 18:21
9. Ge Y, Wang J, Meng Y, Jiang F (2010) Research progress on dynamic route planning of vehicle navigation. *J Highw Transp Res Dev* 27:113–117
10. Yang L (2008) Research and implement of route layout algorithm in vehicle navigation system. Beijing Jiaotong University, Beijing, pp 30–37



11. Tsu-Yang W, Chen C-M, Wang K-H, Meng C, Wang EK (2019) A provably secure certificateless public key encryption with keyword search. *J Chin Inst Eng* 42(1):20–28
12. Chen C-M, Wang K-H, Yeh K-H, Xiang B, Tsu-Yang W (2019) Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications. *J Ambient Intell Humaniz Comput* 10(8):3133–3142
13. Chen C-M, Xiang B, Liu Y, Wang K-H (2019) A secure authentication protocol for internet of vehicles. *IEEE Access* 7(1):12047–12057
14. Pan Jeng-Shyang, Pei Hu, Chu Shu-Chuan (2019) Novel parallel heterogeneous meta-heuristic and its communication strategies for the prediction of wind power. *Processes* 7(11):845. <https://doi.org/10.3390/pr7110845>
15. Wang Z, Li T, Xiong N, Pan Y (2012) A novel dynamic network data replication scheme based on historical access record and proactive deletion. *J Supercomput* 62(1):227–250
16. Zeng Y, Sreenan CJ, Xiong N, Yang LT, Park JH (2010) Connectivity and coverage maintenance in wireless sensor networks. *J Supercomput* 52(1):23–46
17. Lin C, He YX, Xiong N (2006) An energy-efficient dynamic power management in wireless sensor networks. In: 2006 Fifth International Symposium on Parallel and Distributed Computing
18. Sang Y, Shen H, Tan Y, Xiong N (2006) Efficient protocols for privacy preserving matching against distributed datasets. In: International Conference on Information and Communications Security, pp 210–227
19. Fang Z, Fei F, Fang Y, Lee C, Xiong N, Shu L, Chen S (2016) Abnormal event detection in crowded scenes based on deep learning. *Multimed Tools Appl* 75(22):14617–14639
20. Li J, Xiong N, Park JH, Liu C, Shihua MA, Cho SE (2012) Intelligent model design of cluster supply chain with horizontal cooperation. *J Intell Manuf* 23(4):917–931
21. Guo W, Xiong N, Vasilakos AV, Chen G, Yu C (2012) Distributed k-connected fault-tolerant topology control algorithms with PSO in future autonomic sensor systems. *Int J Sens Netw* 12(1):53–62
22. Tan BC, Wang P (2012) Improvement and implementation of A\* path planning algorithm. *J Xi'an Technol Univ* 30:325–329
23. Li WG, Su X (2015) AGV path planning based on improved A\* algorithm. *Mod Manuf Eng* 10:33–36
24. Jabbarpour MR, Zarrabi H, Jung JJ, Kim P (2017) A green ant-based method for path planning of unmanned ground vehicles. *IEEE Access* 5:1820–1830
25. Escario JB, Jimenez JF, Giron-Sierra JM (2015) Antcolony extended: experiments on the travelling salesman problem. *Expert Syst Appl* 42:390–410
26. Xu CB, Liu YM (2009) Research on dynamic route guidance algorithm. *Highw Transp Technol (Appl Technol)* 10:35–36
27. Huo ZS (2012) Research on optimal path of dynamic route guidance system. Shen Yang Aerospace University, Shenyang, pp 28–44
28. Vine SL, Zolfaghari A, Polak J (2015) Autonomous cars: the tension between occupant experience and intersection capacity. *Transp Res C Emerg Technol* 52:1–14
29. Wang ZY, Xing HL, Li TR, Yang Y, Qu R, Pan Y (2016) A modified ant colony optimization algorithm for network coding resource minimization. *IEEE Trans Evol Comput* 20:325–342
30. Zheng F, Zecchin AC, Newman JP, Maier HR, Dandy GC (2017) An adaptive convergence-trajectory controlled ant colony optimization algorithm with application to water distribution system design problems. *IEEE Trans Evol Comput* 21:773–790
31. Xia Y, Chen T, Shan J (2014) A novel iterative method for computing generalized inverse. *Neural Comput* 26(2):449–465
32. Zhong S, Chen T, He F et al (2014) Fast Gaussian kernel learning for classification tasks based on specially structured global optimization. *Neural Netw* 57:51–62
33. Zhang S, Xia Y, Wang J (2015) A complex-valued projection neural network for constrained optimization of real functions in complex variables. *IEEE Trans Neural Netw Learn Syst* 26(12):3227–3238
34. Xia Y, Wang J (2015) Low-dimensional recurrent neural network-based Kalman filter for speech enhancement. *Neural Netw* 67:131–139
35. Zhang S, Xia Y, Zheng W (2015) A complex-valued neural dynamical optimization approach and its stability analysis. *Neural Netw* 61:59–67
36. Liu G, Huang X, Guo W et al (2015) Multilayer obstacle-avoiding X-architecture steiner minimal tree construction based on particle swarm optimization. *IEEE Trans Cybern* 45(5):989–1002

37. Xia Y, Wang J (2016) A bi-projection neural network for solving constrained quadratic optimization problems. *IEEE Trans Neural Netw Learn Syst* 27(2):214–224
38. Zhang S, Xia Y (2016) Two fast complex-valued algorithms for solving complex quadratic programming problems. *IEEE Trans Cybern* 46(12):2837–2847
39. Huang Z, Yu Y, Gu J et al (2017) An efficient method for traffic sign recognition based on extreme learning machine. *IEEE Trans Cybern* 47(4):920–933
40. Guo K, Zhang Q (2013) Fast clustering-based anonymization approaches with time constraints for data streams. *Knowl-Based Syst* 46:08–95
41. Chen X, Jian C (2014) Gene expression data clustering based on graph regularized subspace segmentation. *Neurocomputing* 143:44–50
42. Ye D, Chen Z (2015) A new approach to minimum attribute reduction based on discrete artificial bee colony. *Soft Comput* 19(7):1893–1903
43. Liu G, Guo W, Li R et al (2015) XGRouter: high-quality global router in X-architecture with particle swarm optimization. *Front Comput Sci* 9(4):576–594
44. Liu G, Guo W, Niu Y et al (2015) A PSO-based timing-driven Octilinear Steiner tree algorithm for VLSI routing considering bend reduction. *Soft Comput* 19(5):1153–1169
45. Guo W, Sun X, Niu Y (2014) Multi-scale saliency detection via inter-regional shortest colour path. *IET Comput Vis* 9(2):290–299
46. Yang LH, Wang YM, Su Q et al (2016) Multi-attribute search framework for optimizing extended belief rule-based systems. *Inf Sci* 370:159–183
47. He Z (2016) Evolutionary K-means with pair-wise constraints. *Soft Comput* 20(1):287–301
48. Xia Y, Leung H, Kamel MS (2016) A discrete-time learning algorithm for image restoration using a novel L2-norm noise constrained estimation. *Neurocomputing* 198:155–170
49. Tu J, Xia Y, Zhang S (2017) A complex-valued multichannel speech enhancement learning algorithm for optimal tradeoff between noise reduction and speech distortion. *Neurocomputing* 267:333–343
50. Yu Y, Sun Z (2017) Sparse coding extreme learning machine for classification. *Neurocomputing* 261:50–56
51. Luo F, Guo W, Yu Y et al (2017) A multi-label classification algorithm based on kernel extreme learning machine. *Neurocomputing* 260:313–320
52. Wang S, Guo W (2017) Robust co-clustering via dual local learning and high-order matrix factorization. *Knowl-Based Syst* 138:176–187
53. Yuzhen N, Wenqi L, Xiao K (2018) CF-based optimisation for saliency detection. *IET Comput Vis* 12(4):365–376
54. Zhu W, Lin G, Ali MM (2013) Max-k-cut by the discrete dynamic convexized method. *INFORMS*. <https://doi.org/10.1287/ijoc.1110.0492>
55. Yang Y (2014) Broadcast encryption based non-interactive key distribution in MANETs. *J Comput Syst Sci* 80(3):533–545
56. Pan J-S, Lee C-Y, Sghaier A, Zeghid M, Xie J (2019) Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix–vector product approach. *IEEE Trans Very Large Scale Integr Syst* 27(7):1614–1622

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.