Wireless Networks (2022) 28:979–989
https://doi.org/10.1007/s11276-019-02140-1

A heuristic search algorithm for Hamiltonian circuit problems in directed graphs

Dawei Jin[1] · QingQin Li[1] · Min Lu[1]

Published online: 15 October 2019

Abstract
A heuristic search algorithm is given that determines and resolves the Hamiltonian circuit problem in directed graphs. The heuristic information of each vertex is a set composed of its possible path length values from the starting vertex, which is obtained by the path length extension algorithm. A detailed analysis of algorithm examples shows that the heuristic algorithm can greatly reduce the number of processing nodes compared to the backtracking algorithm without information; meanwhile, the algorithm's time complexity is related to the number of loops in the directed graph. If a directed graph $G(V, E)$ only contains loops that contain the starting point, then the Hamiltonian problem of G can be determined in polynomial time. The algorithm is implemented using C++ and Python programming, and all of the test graphs are generated by the random graph algorithm. The experiments show that the heuristic algorithm can reduce the number of extended nodes by at least 10% in directed random graphs with no or only a few loops. This heuristic algorithm can save time in applications that perform tasks while extending nodes because it reduce the number of extended nodes.

Keywords Hamiltonian circuit · Directed graph · Heuristic algorithm · Random graph · Time complexity

1 Introduction

In the mathematical field of graph theory, a Hamiltonian circuit is a path in an undirected or directed graph that visits each vertex exactly once. The Hamiltonian circuit problem (HCP) is to determine whether a Hamiltonian circuit exists in a given graph, and since this problem has been proved to be NP-Complete, it has no polynomial solution. The state-of-the-art solutions are mainly classified into two categories: theoretical discussion of the laws and conjectures and discussion on algorithms and experimental results. On the one hand, the theoretical discussion includes some sufficient conditions for the determination of the Hamiltonian circuit problem, but not the necessary and sufficient conditions [1, 2], and numerous and diverse results regarding the appearance of Hamilton cycles in random graphs [3, 4]. On the other hand, algorithms and experimental results include various polynomial time heuristic algorithms and exponential time exhaustive search algorithms and related experiments. Polynomial time heuristic algorithms, such as Alon [4], Seeja [5], Frieze [6], always use greedy, rotational transformation and other heuristics and thus have polynomial worst-case complexity. Although such algorithms guarantee that the obtained solution will be found in sufficiently less time that with the exponential time exhaustive search algorithms in most of the cases,they do not guarantee that a solution will be obtained in all cases, they do not guarantee that a solution will be obtained in all cases. Therefore, in practical applications, the commonly used search algorithm is the backtracking method, which belongs to exponential time exhaustive class of methods.Works such as that of Martello [7] proposed a backtracking search algorithm that uses a low degree first heuristic for selecting the next vertex. Frank [8] proposed an algorithm by creating partial paths and making deductions that determine whether each partial path is a section of any Hamilton path, which is

✉ Min Lu
lumiousy@aliyun.com

Dawei Jin
jindawei@zuel.edu

QingQin Li
610071893@qq.com

[1] School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China

actually a non-recursive implementation of the backtracking method. Wen [9] described a hierarchical correlation method to solve the Hamiltonian circuit problem in a digraph that is also based on the backtracking method. The algorithms used to resolve the HCP in random graphs are mainly polynomial time heuristic algorithms; for example, Dana [10] describes an $O(n(logn)^2)$ algorithm, and Frieze [6] describes a polynomial $O(n^{1.5})$ algorithm DHAM for finding Hamilton cycles in digraphs. While these algorithms are all described and theoretically discussed in detail, few of them have been implemented and analyzed by performing detailed experiments using actual graphs.

Combining the characteristics of heuristic and exhaustive algorithms, this paper proposes a heuristic backtracking algorithm to resolve the Hamiltonian circuit problem in a directed graph, and the effect of this algorithm in a random directed graph is discussed. The Hamiltonian circuit problem is widely used in path planning, autonomous driving, artificial intelligence and other related applications [11, 12] and has important theoretical and practical significance.

In the following section, the related theorems and the basic idea of the algorithm are given. In Sect. 3, the implementation of the algorithm, including the generation of random graphs, is elucidated in detail, and the time complexity of the algorithm is analyzed. In Sect. 4, the experimental results are analyzed and displayed using sample cases. Finally, conclusions and future work are summarized. In this article, we use the standard graph terminology.

2 Related theorems of the algorithm

2.1 Basic idea of the heuristic search algorithm

A Hamiltonian circuit on the directed graph $G = (V, E)$ is a loop starting from the starting point S and passing through the remaining vertices in the graph once and only once and back to the starting point. Therefore, the path length of the Hamiltonian circuit is $n(n = |V|)$. The path length from the starting point to any other vertex can be $0, 1, 2, \ldots n$. Based on these basic properties, consider incrementally extending the path length from the starting point. Assign a set for each vertex to record the possible path length values from the starting vertex, and record the number of possible path length values for each vertex as heuristic information for solving the Hamiltonian circuit problem in the directed graph.

After obtaining the sets of all of the vertices' heuristic information at extension step n, it can be determined whether there is a path length value n in the set of the starting point that represents that there is a path back to the starting point. If there is no n in the sets of extended path length values of the starting point S, there is no Hamiltonian circuit in this graph.

If there exists n in the set of S, then search the vertex sequences with the path length values $\{n-1, n-2, \ldots 1, 0\}$ in descending order from the starting point according to the direction of the reverse side of the directed edges. If the sequence exists, then a Hamiltonian circuit exists; otherwise, it does not exist. Combining the existing heuristic search information with the backtracking algorithm, one or all of the Hamiltonian circuits can be found.

2.2 Solution of heuristic information

The calculation of heuristic information is as follows: First, let the current extended vertex be the starting point S, and the current extended path length be 0. At step 0, extend along the directed edge of S in the breath-first way, adding a path length value 1 to all of the adjacencies of S. Note that each vertex can be multiple path length values, which are stored in its set. Let $L_j(v_i)$ be the set of the path length values of vertex v_i at step j of the extension.

In step k, extend all of the vertices with path length k in its extended path length value set. The extension formula from the path length value k to the path length value $k+1$ is $L_{len+1}(v_j) = L_{len}(v_j) \cup \{len + 1\}$, where $len \in L(v_i)$, $v_i \in R(len)$, and $<v_i, v_j> \in E$. Note that when extending from point v_i to point v_j, there must be a directed edge from point v_i to point v_j; otherwise, it cannot be expanded.

Initially, the path length values of all vertices are set to empty set $\emptyset$, i.e. $L_0(v_i) = \emptyset (i = 0, 1, \dot{n} - 1)$. In step 0, the path length value set of the starting point has an element 0, that is, $L_0(v_0) = \emptyset$, $L_1(v_0) = L_0(v_0) \cup \{0\} = \{0\}$. The extension formula from the starting point to the node with path length value 1 is: $L_1(v_i) = L_0(v_i) \cup \{1\}$, when $<S, v_i> \in E$.

In step k, spread along the directed edges of vertices that have the path length value k in the extended path length set, adding a path length value $k+1$ to all of its adjacencies. Clearly, the vertex with path length value k may not be unique. To facilitate the search, this method uses the set $R_j(k)$ to record the set of vertices with the path length value k at step j.

After the $n-1$ step extension, the extended path length value set $L_{n-1}(v_i)$ and the number $|L_{n-1}(v_i)|$ of each vertex are obtained as the heuristic information of the point.

The algorithm specifies that in extension n, only the path length value of the starting point is extended. Note that the extended path length value of the starting point is only added in step 0 and step n.

In addition, the choice of the starting point S is based on the actual needs. The heuristic information obtained from different starting point is different, but this does not affect the Hamilton circuit problem determination and solution in directed graphs.

2.3 Judgment theorem of the Hamiltonian circuit problem

Lemma 1 *If there is no return back to the starting point at extension step $n = |V|$ from the starting point, then a Hamiltonian circuit for the directed graph $G = (V, E)$ does not exist.*

Proof (Contradiction) Assuming that there is a Hamiltonian circuit in graph G, let $v_0v_1v_2...v_{n-1}v_0$, v_1 can be reached in the first extendsion step when extending from v_0, and v_1 can be extended to v_2 in the second extension step. This process continues until it reaches v_{n-1} at step $n-1$. From the known conditions, we can see that v_{n-1} has a directed edge to the starting point v_0. Therefore, if a return back to the starting point after the n-step extension exists, this is contradictory to the known conditions. Therefore, there is no Hamiltonian circuit in graph G. □

Theorem 1 *If there are only rings containing the starting point in the graph $G = (V, E)$ and there is a return back to the starting point after the $n = |V|$ step extension, the graph G has a Hamiltonian circuit.*

Proof According to the known conditions, if there is a return to the starting point after the $n = |V|$ step extension, then search the vertex sequences with the path length values $\{n-1, n-2, ...1, 0\}$ in descending order from the starting point according to the direction of the reverse side of the directed edges back to the starting point, and set the search sequence at this time to be $v_0v_{n-1}v_{n-2}...v_2v_1v_0$.

(Contradictory) Assuming that except for the starting point, there are still repeated nodes v_i and v_j, since the extended path length value of the starting point is only added in steps 0 and n, $v_iv_{i+1}...v_{j-1}v_j$ is a circuit that does not contain the starting point, which contradicts the known conditions, and therefore, the other $n-1$ vertices are different except for the starting point.

Reverse the sequence and obtain a new vertex sequence $v_0v_1v_2...v_{n-2}v_{n-1}v_0$, which satisfies the definition of a Hamiltonian circuit. Therefore, if there are only rings containing the starting point in the graph $G = (V, E)$ and there is a return to the starting point after the $n = |V|$ step extension, then graph G has a Hamiltonian circuit. Evidence for this must be obtained. □

Theorem 2 *If there are only rings containing the starting point in the graph $G = (V, E)$ and there are m vertices with the extended path length value $n-1$ that have a directed edge return to the starting point after the $n-1(n = |V|)$ step extension, then graph G has a Hamiltonian circuit.*

Proof According to the known conditions, if there are m vertices with the extended path length value $n-1$ that have a direct edge return to the starting point after the $n-1(n = |V|)$ step extension, then search the vertex sequences with the path length values $\{n-1, n-2, ...1, 0\}$ in descending order from the starting point according to the direction of the reverse side of the directed edges back to the starting point, and set the search sequence at this time to be $v_0v_{n-1}v_{n-2}...v_2v_1v_0$.

Since there are only rings containing the starting point in the graph $G = (V, E)$, the n vertices in any sequences $v_0v_{n-1}v_{n-2}...v_2v_1v_0$ are different. Reverse the sequence, and obtain a new vertex sequence $v_0v_1v_2...v_{n-2}v_{n-1}v_0$, which satisfies the definition of a Hamiltonian circuit.

Therefore, if there are only rings containing the starting point in the graph $G = (V, E)$ and there are m vertices with the extended path length value $n-1$ that have a directed edge return to the starting point after the $n-1(n = |V|)$ step extension, then graph G has a Hamiltonian circuit. Evidence for this must be obtained. □

3 Algorithm description and analysis

The heuristic search algorithm in this paper mainly includes the following three components: the generation of random graphs, the heuristic information acquisition algorithm, and the heuristic search algorithm of the Hamiltonian circuit problem. These are described below.

3.1 Generation of random graphs (algorithm 1)

$G(n, e)$ is a random graph with n vertices and e random edges. Letting $N = \binom{n}{2}$, we obtain that each graph occurs with the same probability $\frac{1}{\binom{N}{e}}$. The number of the existing Hamiltonian circuits strongly depends on the relationship between the vertex number n and the number of directed edges e in the graph. When the number of vertices n is constant, the number of the existing Hamiltonian circuits decreases with the number of directed edges, and vice versa. If the in-degree or out-degree of a vertex is less than 1 in the generated random graph $G(n, e)$, then there is definitely no Hamiltonian circuit in G. Therefore, we avoid this circumstance by random graph reconstruction. The random graph generation algorithm is called algorithm 1.

Input: The number of vertices n, and the number of edges e in the directed graph.

Output: The random directed graph $G(n, e)$

Steps:

(1) Generate the directed graph $G(n, 0)$, and set the reconstruction variable $loop = 1$. (2) For $(i = 0; i < e; i++)$

  (a) Generate two different random numbers u and v between 0 and $n - 1$, and verify that there is no edge (u, v) in G.
  (b) Add edge (u, v) to G.

(3) Determine whether there exists a vertex for which the in-degree or out-degree is less than 1. If such a vertex exists, set $loop = 1$, clear $G(n, e)$, and return to step 2 to perform the reconstruction. Otherwise, return $G(n, e)$.

Note Using the above algorithm, we can find that neither the in-degree nor the out-degree of any vertices in graph $G(n, e)$ is 0. According to the concept of algorithm 1, when the number of vertices is 15, the range of the number of generated edges is from 15 to $15 \times 14 = 210$.

3.2 Heuristic information acquisition algorithm (algorithm 2)

The heuristic information acquisition algorithm is called algorithm 2.

Input: directed graph $G(n, e)$.

Output: the set of possible path length value of each node in G.

Note Let the list $L[i](0 \leq i \leq n)$ record the possible path length values of node i in the graph G. Let the array $R[i](0 \leq i \leq n)$ record the set of nodes with the path length value i. *curv* records the current extended vertex, and *curl* records the current extension step, which is also the current extended path length value.

Steps:

(1) Data initialization. Set $curv = 0$ and $curl = 0$. Add *curl* to $L[curv]$ and *curv* to $R[curl]$.
(2) Determine whether the path has been extended to the last node, that is, whether *curl* is equal to n. If $curl = n$, the program returns the output; otherwise, perform step (3).
(3) Determine whether the node set $R[curl]$ is empty, and if so, the program returns the output. Otherwise, extend every node j in $R[curl]$, and add $curl + 1$ to $L[j]$. Meanwhile, add the subsequent node of node j to $R[curl + 1]$. Increase the current extended length by 1, $curl = curl + 1$, and go to step (2).

3.3 Heuristic search algorithm of the Hamiltonian circuit problem in a directed graph (algorithm 3)

The heuristic search algorithm for the Hamiltonian problem in a directed graph is called algorithm 3. The algorithm can determine whether there is a Hamiltonian circuit in graph G and output a Hamiltonian circuit.

Input: directed graph $G(n, e)$.

Output: whether there is a Hamiltonian circuit in graph G; if there is, output 0, and if there is not, output 1 and a Hamiltonian circuit.

Note Visit$[n]$ records whether each node has been accessed during the search; 0 indicates that it has not yet been accessed, and 1 indicates that it has been accessed. Let k be the backtracking control variable. The remaining variables are the same as in algorithm 2. The Hamiltonian circuit is stored in the array $Hc[n]$.

Steps:

(1) Calculate the heuristic information of the graph according to algorithm 2, that is, the set of possible path length values for each node. If n does not belong to the set of path length values of the starting point S, the program returns 0. Otherwise, perform step (2).
(2) Data initialization. Set all of the nodes to be not yet visited, i.e., $visit[n] = 0$. $curl = n - 1$, $curv = S$, $visit[curv] = 1$. Let the backtracking control variable $k = 0$.
(3) If $k < 0$, the program returns 0. If $k \geq 0$, then determine whether *curl* is 0; if *curl* =0, the search successfully returns to the starting point, the current search sequence is stored in $Hc[n]$, and the program returns 1.
(4) For all of the precursor nodes i of the node *curv*, if $visit[i] = 0$ and there is path length value *curl* in the extended path length list $L[i]$, the node i is set to be accessed, i.e., $visit[i] = 1$, and the current extension node is set to i, $curv = i$. The current expansion length is reduced by 1, $curl = curl - 1$, and the backtracking control variable is incremented by 1 to examine the next node, $k++$, and perform step (3). If there is no such precursor node or all of the nodes have been accessed, the backtracking control variable $k--$ is set back to the previous layer, and the current extended node is set to not yet visited, i.e., $visit[curv] = 0$. Continue with step (3).

Supplement to the algorithm: If all of the Hamiltonian circuits are required, it is necessary to record the number of Hamiltonian circuits when the search successfully returns to the starting point in step (3), but not when the program does not return.

3.4 Time complexity analysis

The storage method for a directed graph in the computer mainly includes the adjacency matrix and the adjacency table. With a different storage structure, the time complexity of the algorithm is not very different.

In this paper, we use the adjacency matrix in algorithm 2 for acquiring heuristic information. For n extension steps, the number of vertices for each step is $O(1)$ in the best case and $O(n)$ in the worst case. Thus, when storing using an adjacency matrix, the time complexity is $O(n^2)$ in the best case and $O(n^3)$ in the worst case. Thus, the heuristic information is obtained with polynomial time complexity.

In the heuristic search algorithm (algorithm 3) of the Hamiltonian circuit problems, since the extended path length value set is stored in an $n \times n$ matrix, the number of searches for the precursor nodes of each vertex in step (4) is generally $O(n)$. In the best case, one search could locate a simple circuit, and the time complexity is $O(n^2)$.In the worst case, the algorithm needs to traverse the whole graph, and the time complexity is $O(n!)$.

3.5 Example analysis

The time complexity of the algorithm is related to the size and structure of the input directed graph. Therefore, this paper chooses three concrete examples to analyze the difference between the heuristic search algorithm and the non-information backtracking method for the Hamiltonian circuit problem. Because the time complexity of the heuristic information generation algorithm is polynomial, this paper compares the number of nodes that need to be processed by the algorithm as the benchmark. A directed graph of the Tutte subgraph is shown in Fig. 1, in which the vertex labels are shown as a, b, c and so on, and the number of vertices $n = 15$. Extend from the starting point a (path length value is 0) until the extended path length value is $n - 1$, and the extended length list of each vertex is given after the vertex label. Check whether the vertex with the path length $n - 1 = 14$ has a directed edge to the starting point a, i.e., if a node d with path length value 14 and directed edge $<d, a>$ exists. Search the vertex sequences with path lengths $n-1, n-2, \ldots 1, 0$ in order from the starting point in the reverse direction, that is, $a(15) \to d(14) \to i(13) \to n(12) \to m(11) \to l(10) \to j(9) \to e(8) \to f(7) \to p(6) \to k(5) \to h(4) \to g(3) \to c(2) \to b(1) \to a(0)$. The reverse order of this sequence constitutes a Hamiltonian circuit. As shown in Fig. 2(a, b), the directed graph and its extended length list, $n = 8$, can return to the starting point in extension step (8). However, there are no Hamiltonian circuits in graph (a). While searching from vertex a according to the depth-first algorithm with heuristic information, we obtain the sequence $a(8) \to f(7) \to d(6) \to g(5) \to e(4) \to c(3)$. At this point, only $d(2)$ can meet the requirements of vertex c, but $d(2)$ has appeared in the existing path sequence; thus, we can only backtrack to vertex a. Then, no vertices can be searched, so there is no Hamilton circuit in Fig. 2(a).

However, for Fig. 2(b) a Hamiltonian circuit does exist; while searching from vertex a according to depth-first algorithm with heuristic information, we obtain the sequence $a(8) \to f(7) \to d(6) \to g(5) \to e(4) \to h(3) \to c(2) \to b(1) \to a(0)$, or the other sequence $a(8) \to f(7) \to g(6) \to e(5) \to h(4) \to c(3) \to d(2) \to b(1) \to a(0)$. The reverse order of both sequences constitutes two different Hamiltonian circuits.

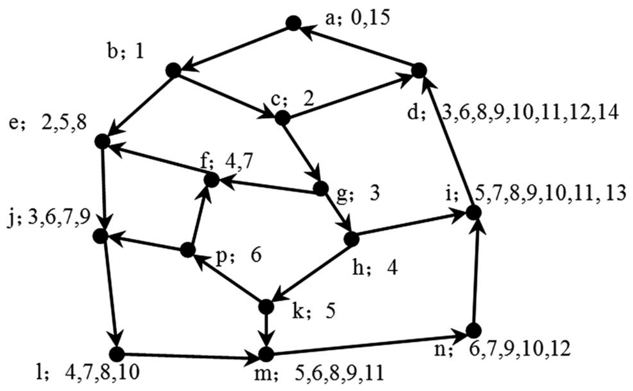


Fig. 1 A directed graph of the Tutte subgraph

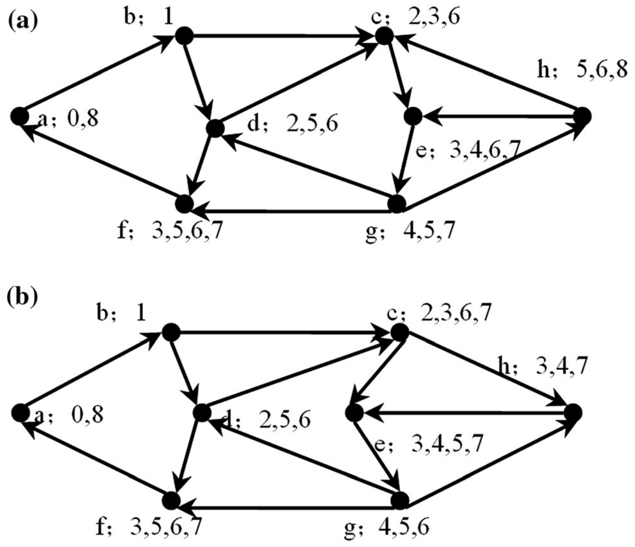


Fig. 2 A directed graph with loops

The heuristic search algorithm need to handle 15 nodes to determine whether there is a Hamiltonian circuit in Fig. 1 and 30 nodes to find all of the Hamiltonian circuits. However, using the backtracking method without heuristic information, at least 21 nodes would need to be processed to find a Hamiltonian circuit, and approximately 84 nodes would need to be processed to find all of the Hamiltonian circuits.

When determining whether there is a Hamiltonian circuit in Fig. 2(a, b), the heuristic search algorithm deals with 10 and 9 nodes, respectively, while the backtracking method deals with 30 and 17 nodes, respectively. To resolve all of the Hamiltonian circuits in Fig. 2(a, b), the heuristic algorithm needs to deal with 10 and 30 nodes, respectively, and the backtracking method without information needs to handle 30 and 47 nodes, respectively.

Therefore, the heuristic search algorithm can greatly reduce the number of processing nodes compared to the backtracking algorithm without information. For different graphs with different sizes and different structures, the amount that the search can be reduced by is also different.

From the above example, we observe that the ring is the main reason for the increase in the time complexity of the algorithm. Algorithm 2 in this paper can automatically process the ring containing the starting point. The ring without the starting point is the main reason for the increase in the time complexity. As shown in Fig. 2(a), because of the existence of the ring *dceg*, when the heuristic search program reaches node *c*, it can only backtrack to solve the impact caused by the ring. If there is no loop that does not contain the starting point, then each vertex can find a predecessor node for which the extended path length value meets the requirements and is different from the node that has been searched. The algorithm can be executed with polynomial time complexity.

In short, if a directed graph $G = (V, E)$ only contains circuits that the starting point, then the Hamiltonian problem of G can be determined in polynomial time.

4 Experiments and results

All of the algorithms in Sect. 3 are implemented using Python and C++ programming. All of the experiments are performed on an Intel(R) Core(TM) i7-4790 CPU 3.60 GHz with 8.0 GB RAM and the win7 OS.

4.1 Selection of the number of vertices and the number of edges for generating a random graph

The random graph generation algorithm (Algorithm 1) guarantees that each vertex has an out-degree and in-degree of at least 1; otherwise, a new random graph that satisfies the condition is regenerated. When the number of nodes n is fixed, the appropriate number of edges e is selected to generate a random graph.

For the 23 pairs of $n = 15$ and e from 31 to 53, algorithm 1 is repeated 1000 times, the average number of times the random graph needs to be regenerated is recorded, and the probability that the algorithm finds a Hamiltonian circuit at this time is obtained, as shown in Fig. 3.

It is observed from Fig. 3 that as the number of edges e decreases, the number of times that algorithm 1 regenerates the random graph increases, and the probability that the random graph generated at this time has a Hamiltonian circuit decreases. The average probability of the existence of a Hamiltonian circuit in Fig. 3 increases from 1.6 to 80.3%, Therefore, the following experiment will select the data according to this rule.

4.2 Experimental results

As observed from the example of Sect. 3.5, the difference between the heuristic search algorithm and the backtracking method without information depends strongly on the number of vertices, edges and graph structure of the directed graphs. Therefore, we first generate 100 random graphs with a fixed number of nodes n and a fixed number of edges e and calculate the difference in the number of nodes visited in the Hamilton circuit problem. Since the algorithm is very fast, we set a certain delay in the program when visiting the nodes so that the algorithm runs slower and we can obtain the time difference between the different algorithms.

When $n = 15$ and $e = 31$, we generate 100 random directed graphs, and the backtracking method without information and the heuristic search algorithm are used on these graphs. The experimental results of 100 pairs of the number of extended nodes are compared. The average reduction ratio for the number of extended nodes is shown in Fig. 4, and the average reduction ratio for the time consumption is shown in Fig. 5, where the average reduction ratio for the number of extended nodes is equal to the number of extended nodes obtained by the backtracking method without information subtracted by the number of extended nodes obtained by the heuristic search algorithm divided the number of extended nodes obtained by the backtracking method without information and the average reduction ratio for the time consumption is equal to the time consumption of the backtracking method without information subtracted by the time consumption of the heuristic search algorithm divided by the time consumption of the backtracking method without information.

From the original data of Figs. 4 and 5 , it can be calculated that when $n = 15$ and $e = 31$, the average

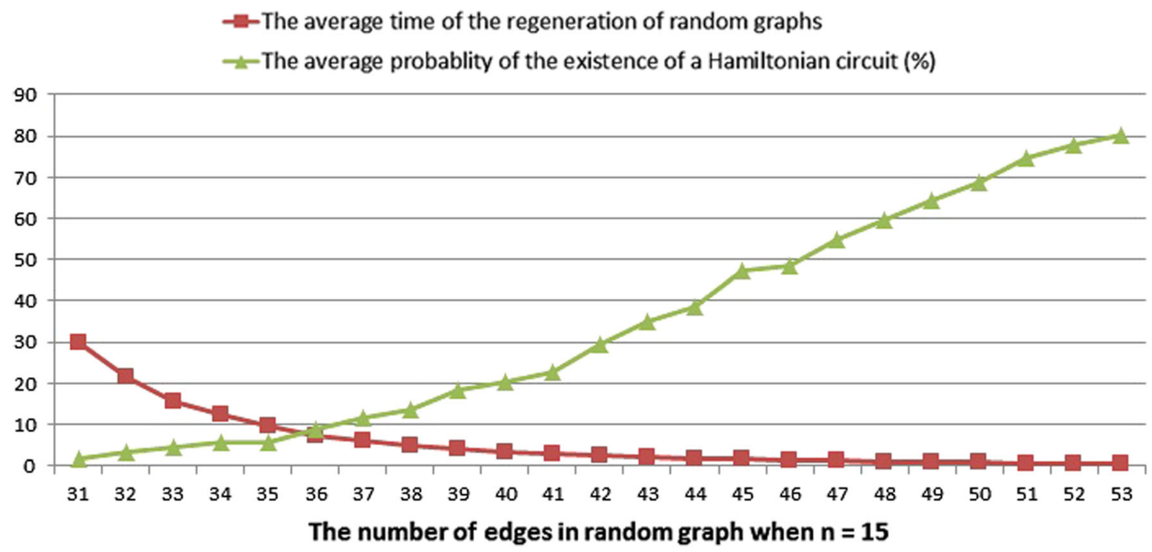
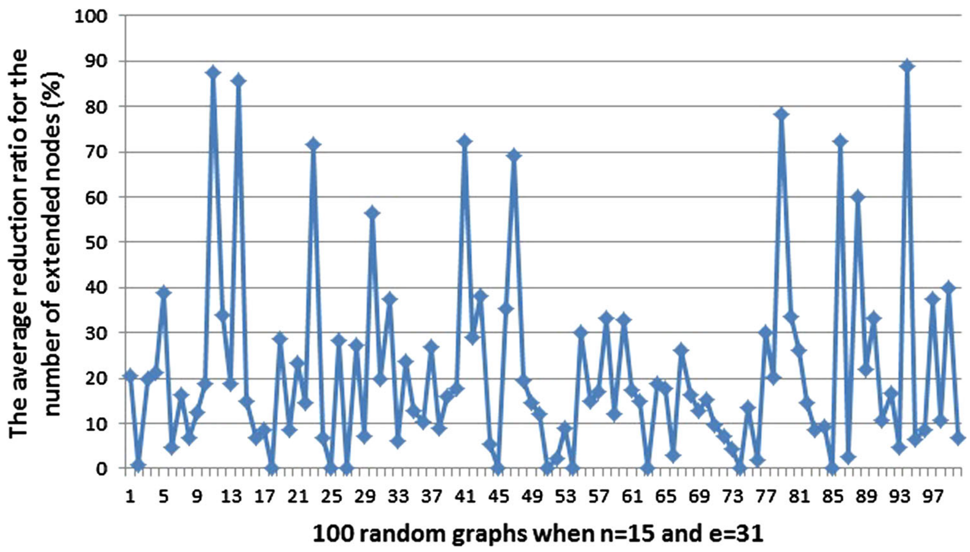


Fig. 3 Average results of 1000 random graphs

Fig. 4 Average reduction ratio for the number of extended nodes on 100 random graphs ($n = 15, e = 31$)



reduction ratio for the number of extended nodes is 21.3496%, and the average reduction ratio for the time consumption is 21.3482% on 100 random graphs. The time consumption of the algorithm is mainly the time delay set in the program when the nodes are extended, so the ratio of the number of extended nodes is almost the same. The experimental results also show that 92% of the randomly generated directed graphs ($n = 15$, $e = 31$) do not have a Hamiltonian circuit. When $n = 15$ and $e = 41$, the average reduction ratios for the number of extended nodes and time consumption are as shown in Fig. 6. As the time consumption of the algorithm is mainly the time delay set in the program when the nodes are extended, the average reduction ratio for the number of extended nodes is almost the same as the average reduction ratio for the time consumption.

From the original data of Fig. 6, it can be calculated that when $n = 15$ and $e = 41$, the average reduction ratio for the number of extended nodes is 9.6571%, and the average reduction ratio for the time consumption is 9.65% on 100 random graphs. The time consumption of the algorithm is mainly the time delay set in the program when the nodes are extended, so the ratio for the number of extended nodes is almost the same. The experimental results also show that 61% of the randomly generated directed graphs ($n = 15$, $e = 41$) do not have a Hamiltonian circuit.

4.3 Analysis of experimental results

To observe, record and analyze the experimental results more accurately and comprehensively, we carried out 100

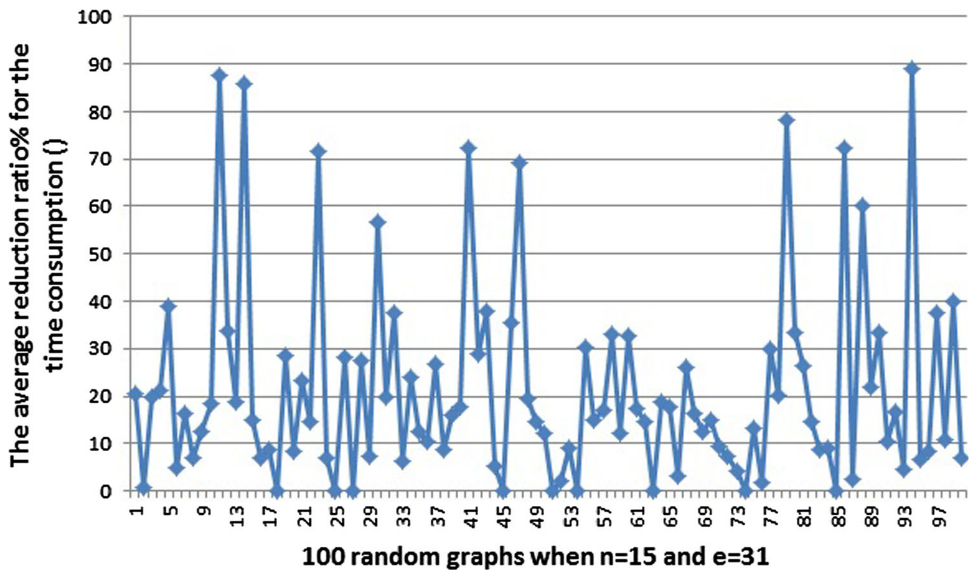


Fig. 5 Average reduction ratio for the time consumption on 100 random graphs ($n = 15, e = 31$)

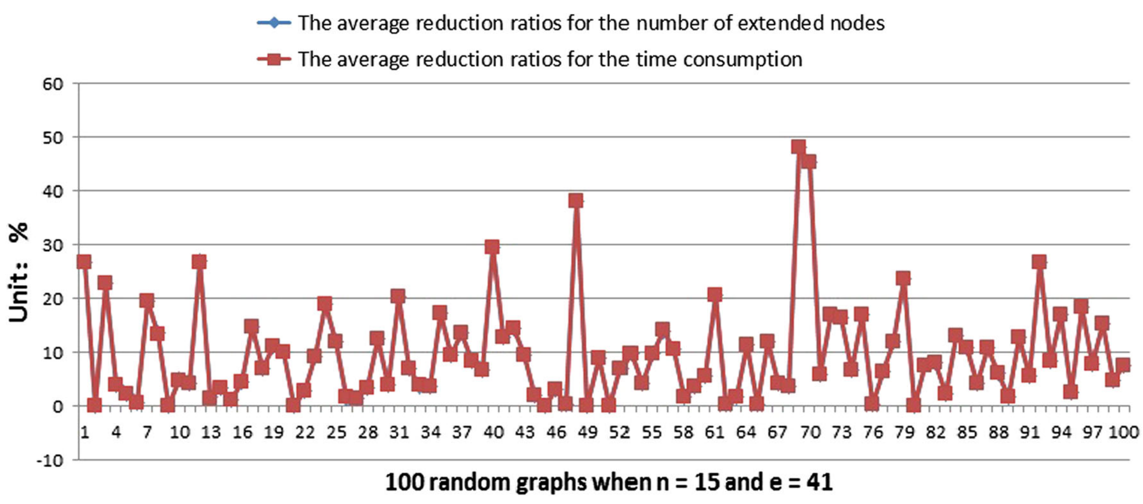


Fig. 6 Comparison of the average reduction ratios for the number of extended nodes and time consumption

experiments on 23 pairs of combinations of $n = 15$ with edge numbers varying from 31 to 53.

The 23 average reduction ratios for the number of extended nodes are shown in Fig. 6 in the diamond-shaped polyline. The square polyline in Fig. 6 shows the probability of the existence of Hamiltonian circuits in the 100 random graphs. The probability of the existence of a Hamiltonian circuit is equal to the total number of Hamiltonian circuits of the 100 random graphs divided by 100.

It can be deduced from Fig. 7 that the average probability of the existence of a Hamiltonian circuit increases with increasing number of edges in random graphs, and the average reduction ratio of the number of extended nodes decreases gradually. The average reduction ratio for the number of extended nodes in Fig. 7 is reduced from 21.34 to 3%. The average probability of the existence of a Hamiltonian circuit in Fig. 7 increases from 3 to 88%.

At the same time, we carried out 100 experiments on 23 pairs of combinations with $n = 10$ and the number of edges varying from 21 to 53. The average reduction ratio for the 23 extended nodes is shown in Fig. 8. As shown in Fig. 8, the average reduction ratio for the number of extended nodes decreases from 59.99 to 6.67%, and the probability of the existence of a Hamiltonian circuit increases from 8 to 99%.

We carried out 100 experiments on 23 pairs of combinations with $n = 20$ and the number of edges varying from 21 to 53. The average reduction ratio for the 23 extended nodes is shown in Fig. 9. In Fig. 9, the average reduction ratio for the number of extended nodes decreases from 14.49% to 4.70%, and the average probability of the existence of a Hamiltonian circuit increases from 2 to 60%.

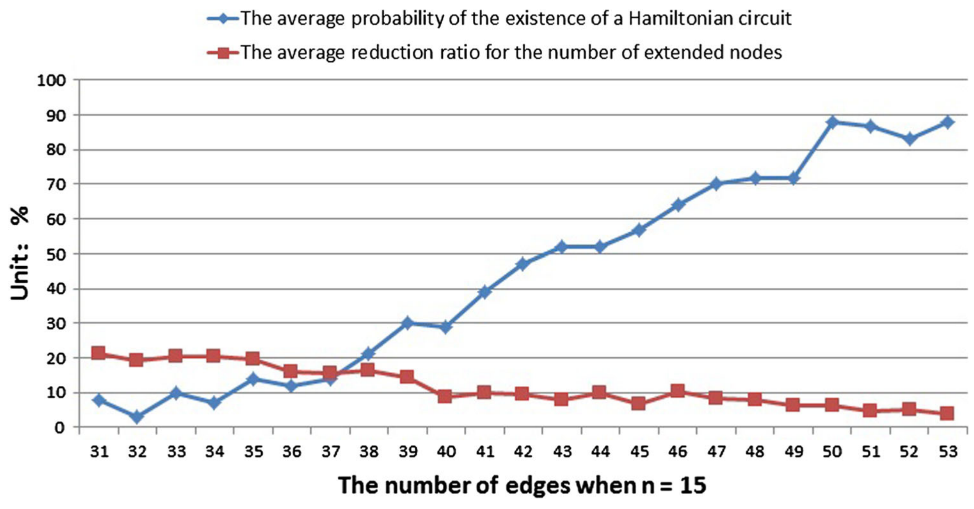


Fig. 7 Experimental results for different numbers of edges when $n = 15$

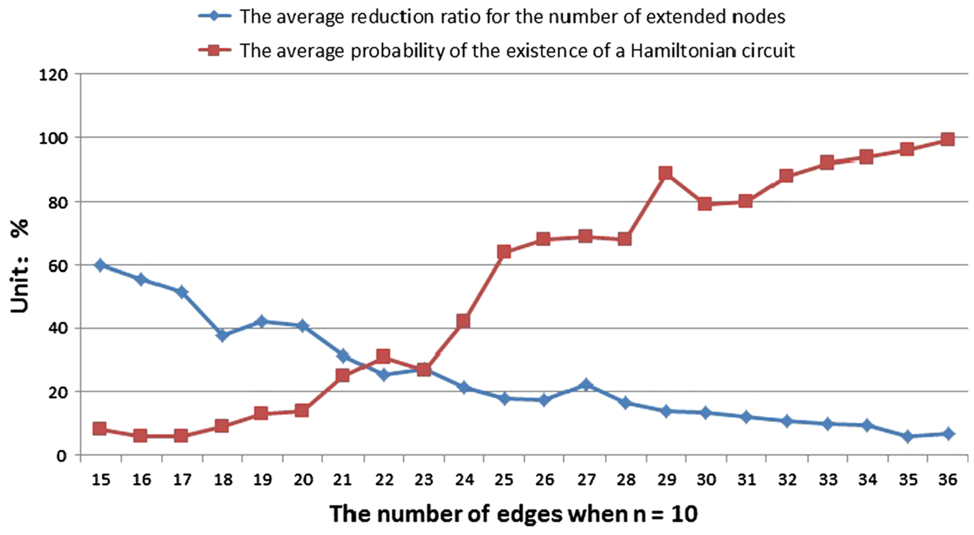


Fig. 8 Experimental results for different numbers of edges when $n = 10$

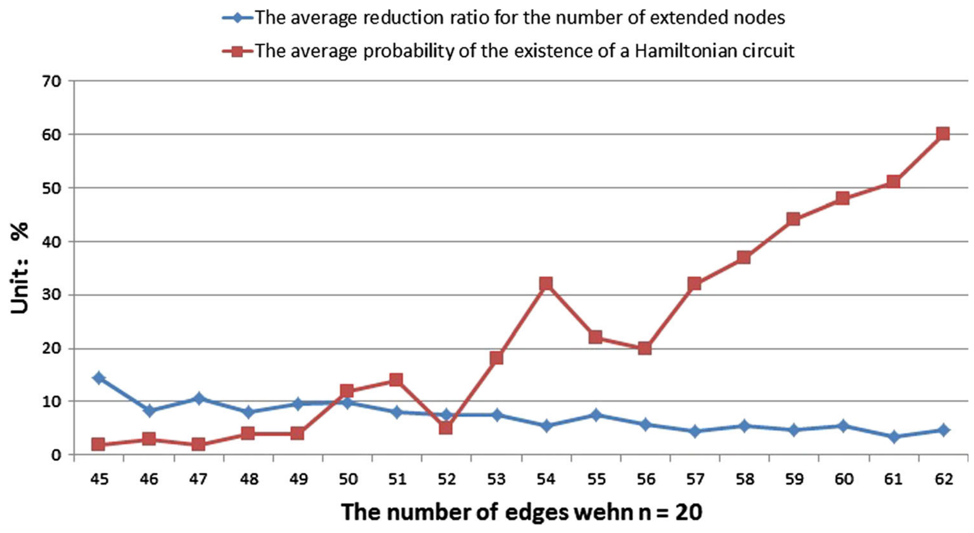


Fig. 9 Experimental results for different numbers of edges when $n = 20$

The results comparing the average reduction ratios for the number of extended nodes in different random graphs with 10, 15, and 20 vertices are shown in Fig. 10. It can be deduced from the figure that the heuristic algorithm is more effective when the number of random graph edges increases gradually and the number of vertices is smaller.

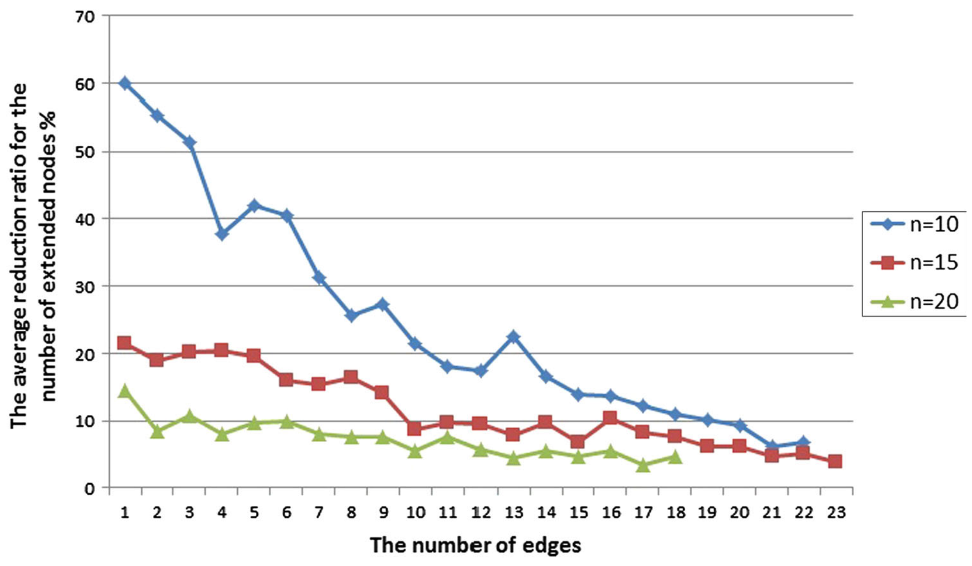


Fig. 10 Comparison of the rate of reduction in the average number of extended nodes

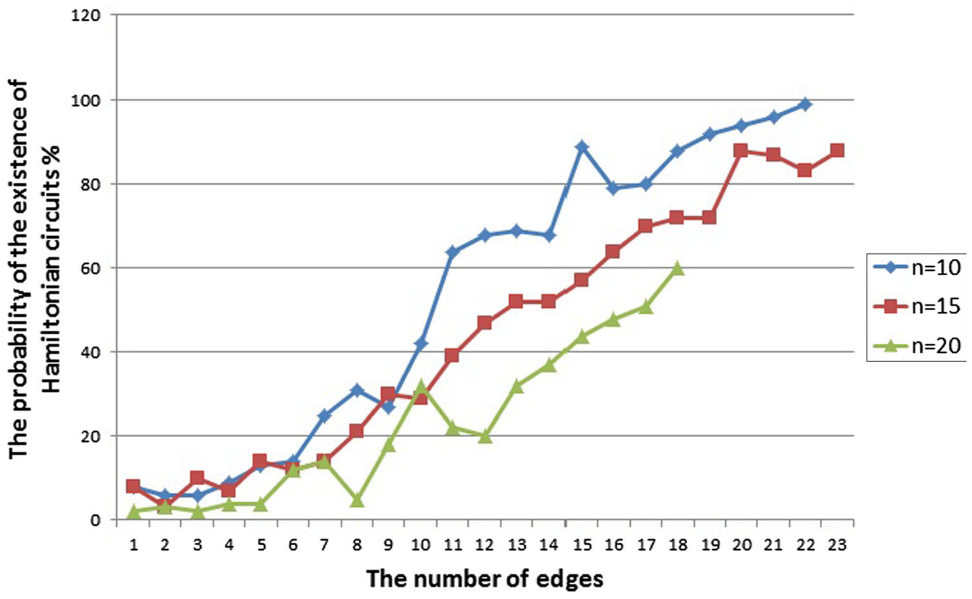


Fig. 11 Comparison of the probability of the existence of Hamiltonian circuits

The results comparing the average probabilities of the existence of a Hamiltonian circuit in different random graphs with 10, 15, and 20 vertices are shown in Fig. 11. It can be deduced from the figure that as the number of random graph edges increases and the number of vertices decreases, the probability of the existence of a Hamiltonian circuit increases, and the heuristic algorithm is more effective.

These experimental results are consistent with the theoretical analysis. From the theoretical analysis, the random graph generation algorithm requires that neither the in-degree nor the out-degree of any vertices in the graph be 0. Therefore, when the number of edges is slightly larger than the number of vertices, it is not easy to meet the needs of the algorithm, and algorithm 1 runs for a long time; thus, the number of generated random graphs that does not contain rings with the starting point is small. However, when the number of edges is larger, the probability of the existence of a Hamilton circuit in the generated random graph increases, and the number of rings without the starting point increases, reducing the efficiency of the heuristic algorithm to a certain extent.

Generally, when the number of vertices n is fixed, if the number of edges increases, the number of Hamiltonian circuits existing in the random graph increases, and the number of rings without the starting point in the random graph increases; thus, the heuristic information will gradually lose its effectiveness, and eventually, the efficiency of the heuristic algorithm will be greatly reduced.

However, when the number of vertices is fixed, if the number of edges decreases, the number of Hamiltonian circuits existing in the random graph is reduced. The number of "rings without the starting point" is reduced in the random graph, and the heuristic information utility is enhanced. Finally, the number of extended nodes when the heuristic search algorithm backtracks can be greatly reduced.

5 Conclusion and future work

This paper presents a decision and solution algorithm of the Hamiltonian circuit problem in directed graphs. Using theoretical analysis and experimental result analysis, it is found that the heuristic search algorithm can greatly reduce the number of extended nodes in the directed graph with no rings or few rings and can rapidly solve the Hamilton circuit problem. Overall, the paper has some theoretical significance and practical value. In the future if the law of variation in these experiments can be modeled in the form of mathematical formulas, the the impact will be more theoretical.

Funding The National Natural Science Foundation of China, No. 71401180; Fundamental Research Funds for the Central Universities.

References

1. Woodall, D. (1972). Sufficient conditions for cycles in digraphs. *Proceedings of the London Mathematical Society*, *24*, 739–755.
2. Kühn, D., & Osthus, D. (2012). A survey on Hamilton cycles in directed graphs. *European Journal of Combinatorics*, *33*(5), 750–766.
3. Frieze, A. (2019). Hamilton cycles in random graphs: A bibliography. arXiv:1901.07139. Accessed 5 June 2019.
4. Alon, Y., & Krivelevich, M. (2019). Finding a Hamilton cycle fast on average using rotations-extensions. arXiv:1903.03007v1. Accessed 7 Mar 2019.
5. Seeja, K. R. (2018). HybridHAM: A novel hybrid heuristic for finding Hamiltonian cycle. *Journal of Optimization*. https://doi.org/10.1155/2018/9328103.
6. Frieze, A. M. (1988). An algorithm for finding Hamilton cycles in random directed graphs. *Journal of Algorithms*, *9*(2), 181–204.
7. Martello, S. (1983). Algorithm 595: An enumerative algorithm for finding Hamiltonian circuits in a directed graph. *ACM Transactions on Mathematical Software*, *9*(1), 131–138.
8. Rubin, F. (1974). A search procedure for Hamilton paths and circuits. *Journal of Association for Computing Machinery*, *21*(4), 576–580.
9. Zhonghua, W., & Yunfei, J. (2005). An algorithm for finding all Hamiltonian cycles in digraph via hierarchical correlation. *Journal of Computer Research and Development*, *42*(10), 1809–1814. **(in Chinese)** .
10. Angluin, D., & Leslie, G. (1977). Fast probabilistic algorithms for Hamiltonian circuits and matchings. *Journal of Computer & System Sciences*, *18*(2), 155–193.
11. Lu, H., Liu, G., Li, Y., Kim, H., & Serikawa, S. (2019). Cognitive internet of vehicles for automatic driving. *IEEE Network*, *33*(3), 65–73.
12. Lu, H., Li, Y., Chen, M., Kim, H., & Serikawa, S. (2018). Brain intelligence: Go beyond artificial intelligence. *Mobile Networks and Applications*, *23*, 368–375.

Dawei Jin received the bachelor's and master's degrees in management science from the Zhongnan University of Economics and Law, Wuhan, China, and the Ph.D. degree in computer science from Wuhan University, Wuhan. He visited the School of Computer Science, Deakin University, Australia, funded by the China Scholarship Council from 2011 to 2012. He is currently a Professor and a Ph.D. Supervisor with the Zhongnan University of Economics and Law. His current research interests include financial information engineering, financial high-frequency data analysis, high-frequency trading, and algorithmic trading.

QingQin Li received the Bachelor of Administration in Nanchang Hangkong University in 2017. She is currently pursuing the Master degree at School of Information and Security Engineering, Zhongnan University of Economics and Law.

Min Lu received the Ph.D. degree in Computer Application from Institute of Software, Chinese Academy of Sciences, Beijing 2008. She is now a Lecturer at School of Information and Security Engineering, Zhongnan University of Economics and Law. Her research interests are data science, machine learning, and social network analysis.