World Wide Web (2019) 22:885–905
https://doi.org/10.1007/s11280-018-0527-8

# Practical $k$-agents search algorithm towards information retrieval in complex networks

Pengfei Liu[1] · Minyu Feng[1] · Ming Liu[1]

Received: 27 September 2017 / Revised: 25 December 2017 / Accepted: 7 January 2018 /
Published online: 23 January 2018

**Abstract** Spanking information retrieval in large-scale Web and network has attracted increasing interest in the research community, many typical approaches have been recalled such as greedy, random-walk and high degree seeking since the search capabilities of complex networks are proved by Kleinberg in 2000. Unfortunately, the retrieval efficiency of these classic approaches is not ideal, and they are only suitable for the specific networks due to their defects. The motivation of this paper is to increase the retrieval efficiency, and we thus proposed a novel $k$-agents search approach for different types of networks which searches the networks with $k$-agents, simultaneously. Besides, to better test the efficiency of algorithms, a new evaluation method which considers search steps and query information both is put forward to measure the cost of the search algorithm. The complexity analysis also will be discussed, and the comparison with other algorithms will be displayed in detail to show its superiority. In the end, for the purpose of displaying a universe application of our algorithm, the simulations in $WS$ (proposed by Watts and Strogatz), $NW$ (proposed by Newman and Watts) small-world and $BA$ (proposed by Barabái and Albert) scale-free network model are carried out to illustrate the performance of the proposed method.

**Keywords** Information retrieval · Large-scale Web · Multi-agents · Complex network

This article belongs to the Topical Collection: *Special Issue on Deep vs. Shallow: Learning for Emerging Web-scale Data Computing and Applications*
Guest Editors: Jingkuan Song, Shuqiang Jiang, Elisa Ricci, and Zi Huang

✉ Pengfei Liu
liupengfei@uestc.edu.cn

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, People's Republic of China

## 1 Introduction

Fueled by the rapid growth of network theory and information science, the requirement of information retrieval in the large-scale Web and networks [6] raises explosively and has attracted increasing interest in both academic and industrial communities. But the current retrieval mechanism has its inherent characteristic [19], which reduces the search complexity, but there are still some defects and deficiencies. Therefore, as the improvement of the traditional search patterns, information retrieval can be more quickly and accurately identify the necessary information and suitable for a broader network structure [16, 31].

Along with the development and growing popularity of the multi-agent technology [25], and the rapid growth of computer vision research, such as video action recognition [35], image retrieval [33] and video captioning [18], etc., the information retrieval technology on the Web has entered a new era [9]. With the expansion of the Web scale, how to achieve useful information becomes very difficult, the search strategy is one of the most critical problems in the agent-based retrieval technology. Most existing search strategies work only with the search engine as an individual and do not consider the facilitate collaborative approach [15], which are potential for greatly improving search quality and efficiency. The single search engine can cover only a small portion of the Web [10], Web mining is using the data mining techniques to discover and extract information from Web and network [3], and one of the more critical technologies is Web structure mining, which studies the Web's hyperlink structure and usually involves analysis of the in-and-out links of the Web, and it has been used for search engine result ranking [30].

The research of complex networks has turned into an attractive assistance for the rapid development of Web which is a valid theoretical tool to support the Web serach [38]. Two key issues need to be solved before achieving the information retrieval on Web: how to build the real network or Web as an abstract complex network and prove its search capabilities [36]. In the late 20th century, Watts, Strogatz, and Newman created two important small-world models by reconnecting the regular network and adding path [28]. Later, Barabái and Albert found the degree distribution of complex network follows a power-law distribution, so they raised a new model called scale-free network [4, 17]. With these models, the search issue of complex networks was raised by Kleinberg [23]. He proved the searching capabilities of complex networks, and Watts did further research on this issue [37]. Recently, K Sneppen discussed the seeking in different kinds of complex networks [29].

Ever since the search capability of complex networks was proved, numerous search strategies boarded the stage. Greedy search is the most simple and straightforward one. However, this simple algorithm may not be optimal in spatial scale-free networks which have high heterogeneity in node degree [34]. Random walk ($RW$) is another commonly used search algorithm which finds a path randomly, such as the unrestricted random walk [20], no-retracing random walk [20], self-avoiding random walk [21], and clustering random walk [7]. As well known, the $RW$ is not suitable for high clustering networks such as scale-free networks for its randomness. To solve this problem, high degree seeking ($HDS$) was put forward [1, 2, 22]. Compared to $RW$, the search step of $HDS$ is quite low and more suitable for scale-free networks. And in recent years, a new search algorithm with efficient local information was proposed [39], a genetic algorithm is carried out for detecting communities in complex networks [32], detection of overlapping modules is used in complex network [5], a new survey approach was put forward [27], a fast and efficient algorithm was used for detecting community structures in complex networks [11], optimal navigation was also applied in complex networks [8] and a novel optimization method was also raised [12]. In recent, some search algorithm based on nearest neighbor [26] and local structure [40] were proposed.

However, all mentioned search strategies either search not fast enough or only apply to restricted network types. The challenge for us is to accelerate the search speed and also get rid of specific network restrictions. In this paper, we proposed a novel algorithm called $k$-agents search algorithm with low search steps and a wide range of applicable networks, where *agent* denotes a seeker, and $k$ different agents are applied simultaneously to accelerate the search speed. The existing approaches are not ideal for the reason of the downright $RW$ each agent uses, and the new proposed algorithm uses an improved $HDS$ instead of merely $RW$. On this basis, the search strategy can take full advantages of randomness and clustering characteristics of the complex networks also to expand the range of applicable networks. This method has lots of applications such as the evaluation of complex network by betweenness centrality and node degree, testing the reliability of complex networks. The key use is to detect topological structure of complex network, some important parameters such as average path length and betweenness centrality which are very important factors to learn a complex network, they are able to be produced by this proposed algorithm. Therefore, this fast search algorithm is serviceable in complex network research.

This paper is organized as follows. Related work and motivations are introduced in Section 1. The problem statement is defined in Section 2. The detailed presentation of $k$-agents searching strategy is described in Section 3. The complexity analysis of this algorithm is presented in Section 4. Simulation is displayed in Section 5. Finally, conclusions are drawn in Section 6.

## 2 Problem statement and network modeling

### 2.1 Information retrieval problem

There are $K$ agents performing a search operation in an unknown Web $N$. For convenience, in this paper, the Web $N$ is undirected and connected graphs without loops and multiple links, which is consisted of $|Q|$ nodes. $N$ can be modeled as a search space $Q \subset R^d$ is a $d$-dimensional Euclidean space. The process of information retrieval is modeled as an uncertainty probability $\phi \to [0, 1]$, i.e., with the continuous obtain of the information, the uncertain awareness of the Web $N$ is declining, and when agents acquire all the necessary information, $\phi$ becomes 0. The location distribution of agents at any given time $t$ is $L(t) = [n_1(t), n_2(t), \ldots, n_K(t), ] \in Q$, and $n_i \neq n_j$, where $n_i(t)$ is the position of the $i$th agent at time $t$. Each agent's search effectiveness is assumed to be a strictly decreasing function of $||n_i - w||$, which is the Euclidean distance between the start location $n_i$ and the node of interest $w \in Q$. The problem addressed in this paper is that of deploying $K$ agents in $Q$ to collect interest information, whereby we reduce the uncertainty density distribution over $Q$.

After $K$ agents are deployed, they perform searching to acquire information in $Q$ and update $\phi$. Usually, the search procedure continues until the probability $\phi$ reaches a threshold limit $\delta \to [0, 1]$, which is set by the user. During the search, agent $a_i$ attempts to gather information about $Q$ and reduces the uncertainty density according to

$$\phi_{n+1}(w) = \phi_n(w) min_i\{\alpha||n_i - w||\} \tag{1}$$

where $\phi_i(w)$ is the uncertainty density at the $i$th searched node; $\alpha$ is a strictly increasing function of the Euclidean distance of the agent and acts as a factor of reduction in uncertainty by the search.

At a given $w \in Q$, only $k_i$ at $w$ can reach the smallest $\alpha||n_i - w||$, that is, the agent which can reduce the uncertainty by its information acquisition. Note that $n$ represents the number of search instances and not time. Here, $n_i = n_i(t)$ is the position of the $i$th agent at the time $t$ at the end of the $n$th search step.

## 2.2 Measurement and network modeling

Search step is one of the most critical parameters for measuring search strategy. It shows the steps we need to take from the source node to the target node. It is also a standard for directly measuring the speed of the algorithm.

Query information is another important measure for measuring search algorithm, especially in the WWW or local area network. Suppose the number of the neighbors of the current node we check as query information. Let $x_i$ be query message of the current node, so the query information for just one complete path from node 1 to $m$ is

$$Q = \sum_{i=1}^{m} x_i. \tag{2}$$

The first proposed small-world model is called $WS$ model which is a transition from a regular network to a complete random graph. $WS$ model is based on the regular network and reflects the small-world property by reconnecting randomly [36]. The random process of a $WS$ model could reveal the small world effect but may destroy the network's connectivity. To prevent the destruction of connectivity, the second small-world model $NW$ model emerged which is also based on regular network, it adds edges randomly instead of reconnecting [28]. Average path length is an important substance for research of complex networks. So far, the accurate average path length of $WS$ and $NW$ are not proposed. However, an approximate formula was put forward by Newman [28]:

$$L(p) = \frac{2|V|}{K} f\left(\frac{|V|Kp}{2}\right) \tag{3}$$

where $|V|$ denotes the number of nodes, and $K$ is half of every node's neighbors.

Especially, $f(x)$ is a universal scaling function which satisfies

$$f(x) = \begin{cases} constant, & x \ll 1; \\ \dfrac{\ln x}{x}, & x \gg 1. \end{cases} \tag{4}$$

Whether $NW$ or $WS$ small-world network model, the growth and preferential attachment of the real networks are not considered. For displaying the two characteristics, another essential complex network model called $BA$ scale-free network is raised [4]. The relationship between the probability that a new node connects to an existing node $i$ and node $i$, $j$'s degrees satisfies as follows:

$$\Pi_i = \frac{k_i}{\sum_j k_j}. \tag{5}$$

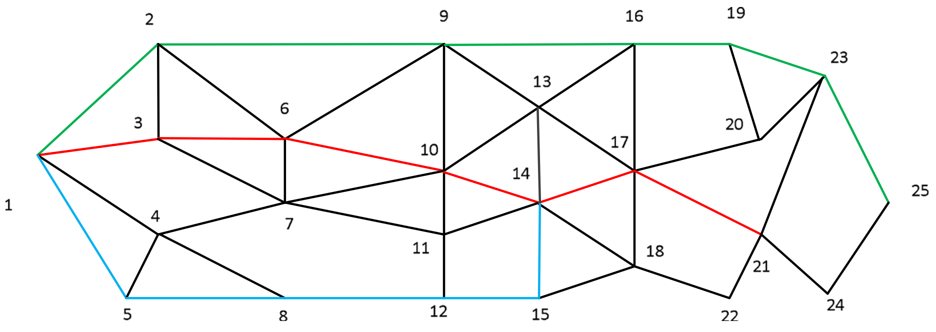


**Figure 1** The path finding of network (red-$k_1$, green-$k_2$, blue-$k_3$)

Through the construction progress, we can see the $BA$ scale-free network model has obvious clustering features. And the distribution function of the $BA$ network can be approximately described by a power-law function with a degree index of 3 which is obtained by continuum theory [4], main equation method [14] and rate equation method [24]. The Average path length of $BA$ scale-free network model is [13]:

$$L \propto \frac{log|V|}{loglog|V|}, \tag{6}$$

indicating that $BA$ scale-free network also has a small-world effect.

## 3 *k*-agents search algorithm toward complex networks

As aforementioned, the current Web search approaches also work slowly in high clustering networks as $RW$. To improve the search speed and expand the range of suitable more types of Webs, we use the complex network to abstract the real Web and proposed a novel $k$-agents search algorithm.

### 3.1 Description of proposed algorithm

$k$-agents search at the same time in our proposed algorithm, different from the traditional algorithm, the first agent $k_1$ adopts $HDS$ while the other agent $k_2$ to $k_k$ takes a few steps randomly before $HDS$. These agents search in parallel until any neighbor of the target node is found. The detailed explanation for $L$ steps of $RW$ and $k$ agents is described as below:

**Table 1** The degree of each node

| Node | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Degree | 4 | 4 | 4 | 4 | 3 | 5 | 5 | 3 | 5 | 6 | 4 | 3 | 5 |
| Node | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | |
| Degree | 6 | 3 | 4 | 6 | 4 | 3 | 3 | 4 | 2 | 4 | 2 | 2 | |

**Algorithm 1** *k*-agents search algorithm towards complex networks

```
Input:
    Adjacency matrix of the networks, the degree of each node;
Output:
    The complete path from source node to target node;
1:  Step 1: Initialization
2:  Determine the source node and target node;
3:  Denoted source node $m_0$ as the current node;
4:  Set the step of random seeking $i$ to be 0;
5:  Step 2: Beginning
6:  Start from the current node $m_0$;
7:  if $m_0$'s neighbors contain target node then
8:      goto Step 5;
9:  else if Search the max degree node in the neighbors of $m_0$ by $k_1$ then
10:     denoted as $m_1$;
11:     Search $k$-1 nodes randomly in the neighbors of $m_0$ by $k_2 \cdots k_k$;
12:     denoted as $m_2 \cdots m_k$;
13:     let $i$=$i + 1$;
14: Step 3: High Degree and Randomly Seeking;
15: Start from current nodes $m_1 \cdots m_k$;
16: if their neighbors contain target node then
17:     goto Step 5;
18: else
19:     Search the max degree node in the neighbors of $m_1$ by $k_1$;
20:     denoted as $\Gamma_1$, and set $m_1 = \Gamma_1$;
21:     Search $k$-1 nodes randomly in the neighbors of $m_2 \cdots m_k$ by $k_2 \cdots k_k$;
22:     denoted as $\Gamma_2 \cdots \Gamma_k$ and set $m_j = \Gamma_j$ (j=2 $\cdots k$);
23:     let $i$=$i + 1$;
24: if i==L then
25:     goto Step 4;
26: else
27:     goto Step 3;
28: Step 4: High Degree Seeking
29: Start from current nodes $m_1 \cdots m_k$;
30: if their neighbors contain target node then
31:     goto Step 5;
32: else
33:     Search the max degree node in the neighbors of $m_1 \cdots m_k$ by $k_1 \cdots k_k$;
34:     denoted as $\Gamma_1 \cdots \Gamma_k$;
35:     set $m_j = \Gamma_j$ (j=1 $\cdots k$);
36:     goto Step 4;
37: Step 5: Termination
38: Generate the final path.
```

There are three points in this method to be noted:

- the same edge can be visited several times, but each node can only be visited only once;
- if agent $k_x$ and $k_y (x < y)$ visit the same current node, then stop agent $k_y$;

– if all of the current node's edges have been visited, then return to the previous one to re-select the current node.

For a clearer understanding, the progress of this algorithm above is discussed step by step. In **Step 1**, the source and target node should be determined first and denoted as the current node. In **Step 2**, the current node's neighbors should be checked first, and if the target node is in its neighbors, then go to **Step 5**. Else, the agent $k_1$ selects the max degree neighbor as $HDS$ while agent $k_2$ to $k_k$ selects the neighbors randomly, and set them as the current nodes, marked as $m_1 \cdots m_k$, then go to **Step 3**. In **Step 3**, the neighbors of current nodes $m_1 \cdots m_k$ should be checked first, and if the target node is in its neighbors, then go to **Step 5**. Else, the agent $k_1$ selects the max degree neighbor as $HDS$ while agent $k_2$ to $k_k$ selects the neighbors randomly, set them as the current nodes, marked as $m_1 \cdots m_k$. If this is the $L - 1$ times to run **Step 3**, then go to **Step 4**. Else, repeat **Step 3**. In **Step 4**, the neighbors of current nodes $m_1 \cdots m_k$ should be checked first as **Step 2**, and if the target node is in its neighbors, then this algorithm goes to **Step 4**. Else, the agent $k_1$ to $k_k$ selects the max degree neighbor, and set them as the current nodes marked as $m_1 \cdots m_k$. Repeat this step until any neighbor of the target is found. In **Step 5**, the target node is found and the final path is written.

It is noteworthy that agent $k_1$ is very special, it just uses $HDS$ until any neighbor of target node is found which ensures this algorithm won't take more steps than $HDS$. The other agents $k_2$ to $k_k$ take a few steps in the random walk but not exactly same to traditional $RW$ for it won't inquiry current node's neighbors in each step, we do this to drop the excess query message. And then agents $k_2$ to $k_k$ go on with $HDS$. The steps of the random walk are decided by the type of the network and number of network's nodes. For the regular and random networks, random walk takes more steps while for small world and scale-free networks, random walk always takes less. For a large network, we need more random walks to show its effectiveness. Take the size of the network of 100 nodes as an example, two or three steps are reasonable. In each step, agents $k_1$ to $k_k$ work simultaneously which means this algorithm works in parallel. However, after the random walk, any two or more agents visit the same node indicate that they will take the same path in the future. Therefore, we stop the excess agent. Each node can be visited only once to prevent superfluous query information. When the target node is reached in any agent and any step, every agent stops the search at the same time.

### 3.2 Evaluation in real network

The above sections introduced the algorithm design and information retrieval in the abstract complex network in detail, in this section, we analyze the performance in a normal real network, as shown in Figure 1, each node denotes a station. The proposed algorithm is used in this specific network to find a path from source node 1 to target node 25. In order to express clearly, we just take three agents, denoted as $k_1$, $k_2$ and $k_3$. agent $k_1$ is very distinctive because it uses $HDS$ all the time until the target node is found, agents $k_2$ and $k_3$ first take four steps randomly then start to run in $HDS$.

The detailed progress of $k$-agent search algorithm for this real network is described as follows: First, the degree of each node $d_i$ is kept first for calculating query information conveniently, and the degree of each node is shown in Table 1. Node 1 is determined to be the source node and 25 is the target node as described in algorithm's **Step 1**. In the first four steps, agent $k_1$ uses $HDS$ while $k_2$ and $k_3$ search randomly as described in **Step 2** and **Step 3**. From the fifth step, agents $k_1$, $k_2$ and $k_3$ all run in $HDS$ as described in **Step 4**.

Finally, the agent $k_2$ reaches node 25 first, so this algorithm stops, and the final path is written as described in **Step 5**.

The results include the detailed path and query information both are given in Table 2.

It is to be noted that in step 5, agent $k_3$ goes into the same node 14 as $k_1$, then they will search the same path in the future. Therefore, it stops for preventing additional query message. At last, agent $k_2$ gets the target node, and the whole query process ends. In this progress, the same node 14 which is visited by agent $k_1$ and $k_3$ is called **'collision node'**. To compare with Kim's $HDS$, Table 3 shows the search steps, query information and each agent's path in $HDS$.

The results show that the proposed algorithms need fewer search steps and query information. However, the additional memory is opened up to save agent $k_1$ and $k_2$. The detailed discussion of the efficiency of the algorithm will be given in the simulation section below. To be more intuitive, Figure 1 gives the path for agent $k_1$, $k_2$ and $k_3$ in the color of red, green and blue. In this figure, we can see that 14 is a collision node clearly.

## 4 Complexity analysis

In this section, the complexity analysis of $k$-agents search algorithm will be made in best, worst and average situation respectively.

### 4.1 Analysis in best situation

The best case for the proposed algorithm is that the source node's neighbors include the target node, that is to say, the target node is found directly. We just need to search source node's neighbors which means to traverse the whole network only once, so the time cost is

$$T(|V|) = |V|. \tag{7}$$

Figure 2a shows an example of path from node 1 to its neighbor node 2 in the red line.

### 4.2 Analysis in worst situation

Opposed to the best case, the worst situation always needs the highest search steps and query message which means we search every node in the network, and the target node is the last one to be found. In the graph, agents search all nodes, so the time cost is

$$T(|V|) = = \sum_{k=1}^{|V|} |V| = |V|^2. \tag{8}$$

**Table 2** The search steps, query information and each agent's path of 3-agents search algorithm

| Step | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Query information | 0 | 4 | 7 | 11 | 16 | 23 | 31 |
| Agent $k_1$ | $1 \rightarrow$ | $3 \rightarrow$ | $6 \rightarrow$ | $10 \rightarrow$ | $14 \rightarrow$ | $17 \rightarrow$ | 21 |
| Agent $k_2$ | $1 \rightarrow$ | $2 \rightarrow$ | $9 \rightarrow$ | $16 \rightarrow$ | $19 \rightarrow$ | $23 \rightarrow$ | 25 |
| Agent $k_3$ | $1 \rightarrow$ | $5 \rightarrow$ | $8 \rightarrow$ | $12 \rightarrow$ | $15 \rightarrow$ | 14 | |

**Table 3** The search steps, query information and each agent's path of $HDS$

| Step | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Query information | 0 | 4 | 7 | 11 | 16 | 21 | 26 | 29 | 32 |
| Search path | 1 → | 3 → | 6 → | 10 → | 14 → | 17 → | 21 → | 23 → | 25 |

Figure 2b shows an example of the worst situation. agent $k_1$ in blues uses $HDS$. Agent $k_2$ in red lines first takes two steps of random walk node 4, then start $HDS$ to reach node 5. However, node 5 is the collision node, so the agent $k_2$ is terminated. At last the whole seven nodes have been reached and node 7 is the last one to be found.

## 4.3 Analysis in average situation

Differs from other algorithms, $k$-agents search strategy usually goes into the average situation for its random characteristic. In fact, the average situation is closer to the best which will be demonstrated in the calculating below.

Here is the detailed discussion of the average case. Suppose the number of agents as $i$, the steps of the random walk of $k_2$ to $k_i$ are $m$, the average path length also considered as the average search steps $L$ follows

$$L = \frac{1}{\frac{1}{2}N(N+1)} \sum_{a \geq b} d_{ab}, \tag{9}$$

where $d_{ab}$ denotes the distance between node $a$ and $b$. So the time cost for agent $k_1$ is

$$T(|V|_{k_1}) = \sum_{n=1}^{L} |V| = L|V|. \tag{10}$$

Suppose the time for each random walk without querying is just 1, and each time cost for agents $k_2$ to $k_i$ is

$$T(|V|_{k_s}) = m + \sum_{n=1}^{L-m} |V| = m + (L-m)|V|, s = 2, 3, \cdots, i. \tag{11}$$

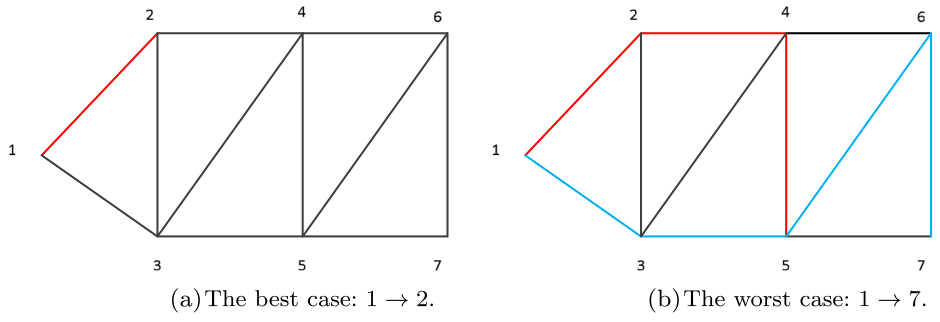


**Figure 2** A sample of best and worst case

Since only one agent can reach the target, and suppose each agent has the probability $\rho_1, \rho_2, \ldots, \rho_i$ to reach the target, therefore, the anticipant time cost is

$$\begin{aligned} T(|V|) &= \rho_1 T(|V|_{k_i}) + \rho_2 T(|V|_{k_2}) + \ldots + \rho_i T(|V|_{k_i}) \\ &= \rho_1 L|V| + (\rho_2 + \ldots + \rho_i)(m + (L-m)|V|) \\ &= L|V| + (\rho_2 + \ldots + \rho_i)(1 - |V|)m. \end{aligned} \tag{12}$$

In this part, we assume $\rho_i = 1/i$ which means each agent has the same chance to search successfully, and for simplicity, 3 agents and 2 random steps are taken, that is to say, $i$ equals to 3 while $m$ equals to 2, and the equation changes into:

$$T(|V|) = L|V| + \frac{4}{3}(1 - |V|). \tag{13}$$

So far, the time cost for the average situation is carried out. However, different networks have different average path lengths, so we need to discuss them one by one in the following.

## 4.4 The average situation in different networks

The detailed discussion for different kinds of networks in average case is listed in this subsection.

### *4.4.1 Regular network*

Different regular networks also have different average path lengths, we take the star coupled network as an example. In this kind of network, there exists a center node connects to other $|V| - 1$ nodes. However, any two of them is disconnected, so the average path length is

$$L_{star} = 2 - \frac{2(|V|-1)}{|V|(|V|-1)} \to 2, \quad (|V| \to \infty). \tag{14}$$

and the average time we cost is

$$T(|V|) = 2|V| - \frac{4}{3} \in O|V|. \tag{15}$$

### *4.4.2 Random graph*

The random graph is the first kind of network with randomness. As an example with 10 nodes and the probability of connection 0.25, the average path length of random graph is

$$L_{ER} \propto \frac{\ln|V|}{\ln\langle k\rangle}, \tag{16}$$

where $\langle k\rangle$ is the average degree of the network which follows

$$\langle k\rangle = p(|V| - 1) \to |V|, \quad (|V| \to \infty). \tag{17}$$

Therefore, the time cost for random graph is

$$T(|V|) \propto \frac{\ln|V|}{\ln\langle k\rangle}|V| + \frac{4}{3}\left(1 - \frac{\ln|V|}{\ln\langle k\rangle}\right) \in O(|V|\ln|V|). \tag{18}$$

#### 4.4.3 Small-world model

Small-world networks are built from the regular network then reconnecting randomly or adding path randomly separately. The specific tectonic processes will be shown in simulation section. Unfortunately, the accurate expressions for $WS$ and $NW$ haven't been worked out. However, $WS$ model's average path length has been given by renormalization group analysis as (3):

$$L_{WS} = \frac{2|V|}{K} f\left(\frac{|V|Kp}{2}\right), \tag{19}$$

where $K$ is an even, and each node connects to its left and right neighbors totally $K/2$ nodes, $p$ is the probability of reconnecting, $f(x)$ is a universal scaling function. So the time cost for $WS$ model is

$$T(|V|) = \begin{cases} \left(\frac{2|V|}{K} - \frac{4}{3}\right) \in O(|V|), \quad |V| \ll \frac{2}{Kp}; \\ \left(\frac{2|V|}{K} - \frac{4}{3}\right) \frac{2\ln\frac{|V|Kp}{2}}{|V|Kp} + \frac{4}{3} \in O(\ln|V|), \quad |V| \gg \frac{2}{Kp}. \end{cases} \tag{20}$$

where $C$ is a constant.

From this equation, we can see that the time spends for the small-world network is close to the best situation, revealing that our proposed method works efficiently in $WS$ small-world model. In the second situation, the cost is less than the best case which means the average path length is not sufficiently precise.

#### 4.4.4 Scale-free network model

The $BA$ scale-free network is a better model for its characteristics of growth and preferential attachment. The tectonic processes will be shown in the simulation section. The average path length of $BA$ scale-free model is

$$L_{BA} \propto \frac{\log|V|}{\log\log|V|}. \tag{21}$$

It shows this kind of network also has the small-world property because $\log|V|$ increase slowly while $|V|$ becomes vast. In other words, the average path length is minimal even the network itself is exceptionally vast. Therefore, the time cost for $BA$ scale-free network is

$$T(|V|) \propto \frac{|V|\log|V|}{\log\log|V|} + \frac{4}{3}(1 - |V|) \in O\left(\frac{|V|\log|V|}{\log\log|V|}\right). \tag{22}$$

## 5 Simulation and results analysis

In this section, we will describe the building of the networks, the measure of parameters and the simulation studies, respectively. All programs, codes were run in Matlab $R2010B$.

### 5.1 Small-world network model

The construction algorithm for the $WS$ small-world model has been stated in Section 2. We start from a nearest-neighbor coupled network which is surrounded as a ring with 100 nodes, and each node is connected to its every two neighbors on both the left and the right.

The probability of reconnecting $p$ is equal to 0.2. Figure 3 shows the result. The average degree of this model is 8 while the max degree is 21.

## 5.2 Building of networks

To show the results of the proposed algorithm, the models of complex networks are built first.

To build the $NW$ small-world network, we also start from a nearest-neighbor coupled network which is surrounded as a ring with 100 nodes and each node is connected to its every two neighbors on both the left and the right. The probability of randomly connecting is equal to 0.01, here we choose a low probability of adding edges to prevent too many paths that ensures visibility of the figure. Finally, the result is given in Figure 4. The average degree of this network is 10, and its max degree is 20.

From the Figures 3 and 4, we can see that the structure of $WS$ is similar to the structure of $NW$. The reason is that $NW$ is equal to $WS$ essentially when $p$ is low enough and $|V|$ is huge enough.

### *5.2.1 BA scale-free network*

The $BA$ scale-free model is different from $WS$ or $NW$ model since it starts from a few nodes instead of a nearest-neighbor coupled network and the probability of adding path is calculated by formula.

To build the $BA$ scale-free model, we start from a complete graph of just 10 nodes, and then a new node is added to existing nodes each time. Repeat the progress 90 times, we

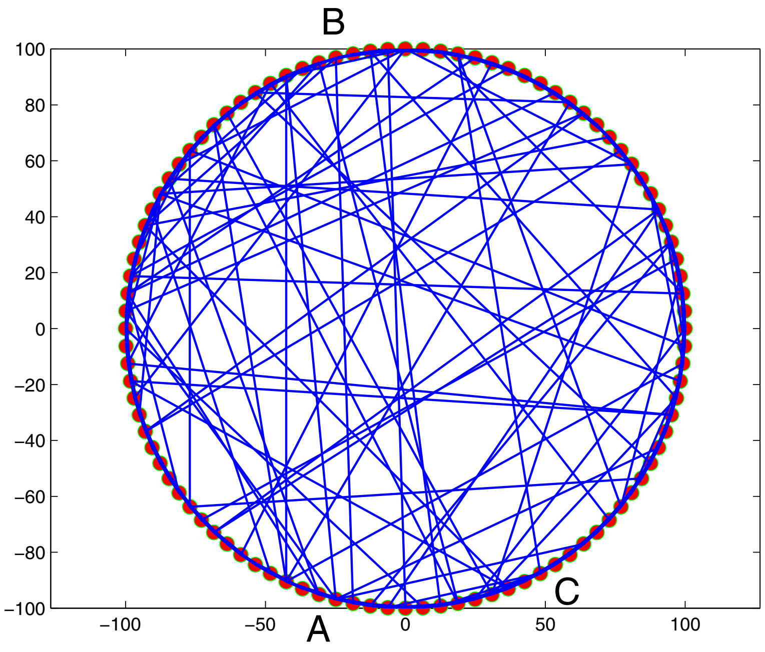


**Figure 3** $WS$ small-world model made by matlab

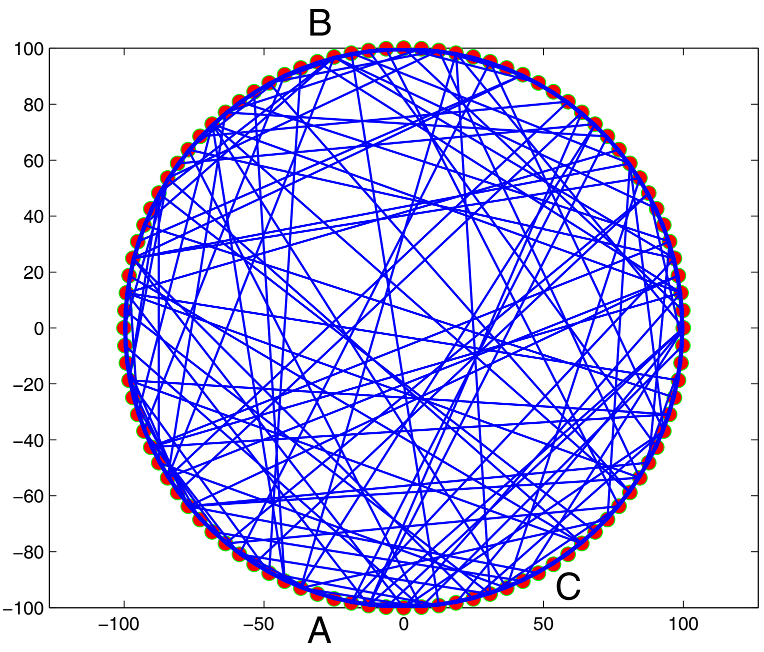


**Figure 4** *NW* small-world model made by matlab

obtain a network with 100 nodes. In the end, Figure 5 shows the result. The average degree of it is 4 while the max degree is 28.

## 5.3 Measuring parameters

Before we carry out the results, some important parameters should be introduced for measuring the effectiveness of the search algorithms.

### *5.3.1 Average search steps*

As we discuss in the Section 4, the average search step is the most important parameter for measuring search strategy.

We randomly choose $n$ source nodes repeatedly, applying some kinds of search algorithms to the network of $|V|$ node. To each source node $i$, the search steps for simple one target node are denoted as $t_j$, the total search steps to other all $|V|-1$ nodes are

$$T_i = \sum_{j=1, j\neq i}^{|V|} t_j. \tag{23}$$

So the average search steps between any two nodes are

$$\overline{T} = \sum_{i}^{n} \frac{T_i}{n(|V|-1)}. \tag{24}$$

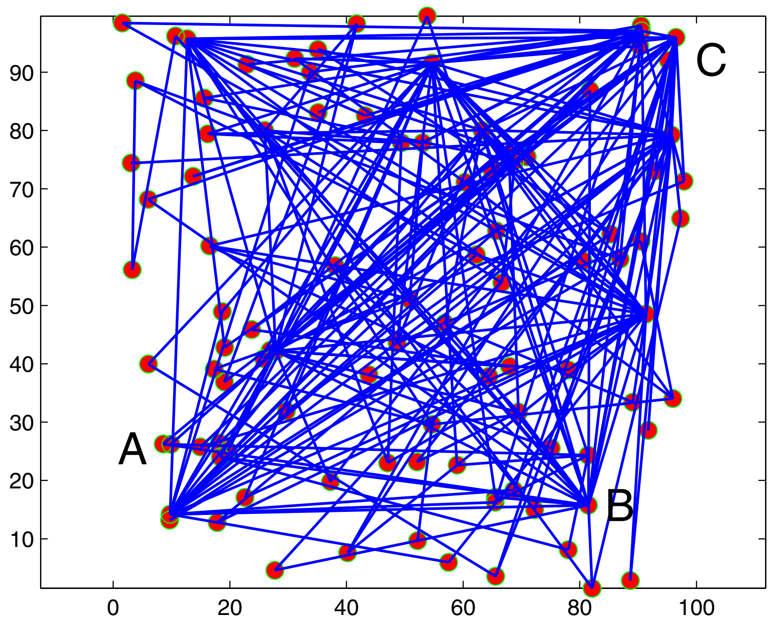


**Figure 5** *BA* scale-free model made by matlab

### *5.3.2 Average query information*

Average query information is another important parameter for measuring search algorithm, especially in the WWW or local area network.

Similar to average search steps, we choose $n$ different source nodes, to each source node $i$, the total query information to other $|V|-1$ nodes is

$$Q_i = \sum_{j=1, j\neq i}^{|V|} Q_j. \tag{25}$$

So the average query information between any two nodes is

$$\overline{Q} = \sum_{i}^{n} \frac{Q_i}{n(|V|-1)}. \tag{26}$$

### *5.3.3 Search cost*

Search steps and query information are both important, but sometimes the degree of importance for them is not exactly same. Take social network as an example, more asking is the faster way to reach the one you want to find. Therefore, search steps of the social network have more weights than query information. But in some networks such as $P2P$, too much query information cause the network congestion, so we prefer to take more steps than more inquiries.

Here we display a variable parameter to measure both search steps and query message, we called it search cost, denoted as

$$C = a\overline{T} + b\frac{\overline{Q}}{k_{max}}, \tag{27}$$

where $\overline{T}$ denotes the average search steps, $\overline{Q}$ denotes the average query information, $k_{max}$ is the max degree and $a$, $b$ are variable coefficients.

In (27), $\overline{Q}/k_{max}$ denotes a value close to $\overline{T}$ since the characteristic of $HDS$ is that almost every path we find passes the node with the max degree. It ensures that two parts of the (10) are possibly in the same weights if coefficient $a$ and $b$ are equal. For convenience, we take $a$ as 1 and $b$ as 1 in the discussion below.

Equation (28) shows a more general expression of search cost. However, the values of $\lambda_t$ and $\lambda_q$ depend on the algorithms, networks themselves and human factors.

$$C = \lambda_t \overline{T} + \lambda_q \overline{Q}. \tag{28}$$

The value of search cost shows the efficiency of the search strategy: the lower search cost means search faster and inquiry lower while the more search cost denotes search slower and inquiry higher.

## 5.4 Simulation studies

Based on the $WS$, $NW$ and $BA$ models, we compare $k$-agent search algorithm with other search strategies, analyze the successful search agents and run the proposed algorithm in different sizes of networks one by one as follows.

### *5.4.1 Comparison of different search strategies*

In this subsection, a comparison of different search strategies of $k$-agents search ($k$-agent), high degree seeking ($HDS$) and high degree seeking with $k$ steps of random walk ($k$-$HDS$) is made to show the effective and efficient performance of the proposed approach.

As previously described, nodes $A$, $B$ and $C$ from every three models are selected as source nodes to obtain the search steps, query information, search costs and their average values. $A$, $B$ and $C$ are nodes 6, 48 and 92 in the adjacency matrixes which is generated by three models, respectively. Four agents are used to search in $WS$, $NW$ and $BA$ model, denoted as $k_1$, $k_2$, $k_3$ and $k_4$. agents $k_2$ to $k_4$ take 2 steps of random search before $HDS$.

We run $k$-agent, $HDS$ and $k$-$HDS$ in three models one by one and total data of each node's search steps and query information are shown in Table 4.

In Table 4, $T$ stands for the total search steps while $Q$ stands for the total query information. Take node $A$ as an example, the total search steps of $HDS$ in $WS$, $NW$ and $BA$ models are 1113, 1268 and 1173, the total search steps of $k$-$HDS$ are 1113, 1151 and 1134, while the total search steps of $k$-agent are only 694, 514 and 522. The total query information of $HDS$ in $WS$, $NW$ and $BA$ models are 10500, 10014 and 10521 while the total query information of $k$-$HDS$ are 10362, 10793 and 10885. However, the total query information of $k$-agent are 20518, 19440 and 27698.

From the results of total search steps, we can see that the $k$-agent search strategy is an effective algorithm since its total search steps are much lower than $HDS$ and $k$-$HDS$ especially in $NW$ and $WS$ model. In other words, the path of any two nodes is shorter in k-agents search algorithm than other two algorithms.

From the results of query information, we could see the shortcoming of proposed algorithm obviously. The total query information is higher than other two algorithms due to its extra agents.

However, the total data are not easy to check the result, therefore, the (24), (26) and (28) are used to calculate the average search steps, average query information and search cost of

**Table 4** The total search steps and query information of each node

| Model \ Cost | $T_{HDS}$ | $Q_{HDS}$ | $T_{k-HDS}$ | $Q_{k-HDS}$ | $T_{k-agents}$ | $Q_{k-agents}$ |
|---|---|---|---|---|---|---|
| (a) Node *A* | | | | | | |
| WS | 1113 | 10500 | 1131 | 10362 | 694 | 20518 |
| NW | 839 | 10014 | 937 | 10793 | 514 | 19440 |
| BA | 607 | 10521 | 691 | 10885 | 522 | 27698 |
| (b) Node *B* | | | | | | |
| WS | 1268 | 11876 | 1151 | 10563 | 691 | 20146 |
| NW | 941 | 10898 | 900 | 10262 | 556 | 21262 |
| BA | 660 | 10125 | 687 | 10418 | 593 | 15573 |
| (c) Node *C* | | | | | | |
| WS | 1173 | 10970 | 1134 | 10478 | 710 | 22985 |
| NW | 913 | 10726 | 900 | 10262 | 507 | 18814 |
| BA | 710 | 9942 | 760 | 9790 | 672 | 11957 |

$WS$, $NW$, and $BA$ models which show a bunch of more intuitive data for comparison. The detailed comparisons of different average data is listed in Table 5.

In Table 5, $\overline{T}$ denotes the average search steps and $\overline{Q}$ denotes the average query information. The average search steps and query information are rounded to the nearest integers. In this way, the data reflect much more reasonable in reality because we cannot take half step. However, the search cost is kept two decimal places to ensure their accuracy since they don't need to be rounded. We take $WS$ as an example, the average search steps of $HDS$ and $k$-$HDS$ are both 12, and the average search steps of $k$-agent are barely 7. The average query information of $HDS$ and $k$-$HDS$ are 112 and 106, and the average query information of $k$-agent is 214. The search cost of $HDS$, $k$-$HDS$ and $k - agents$ are 17.31, 16.52 and 17.26 respectively, which are nearly equal.

Through the comparison of average search steps, we can figure out that $k$-agent takes lower steps to reach the target nodes than $HDS$ and $k$-$HDS$ in the condition of nearly same search cost. In other words, our proposed method is much faster than other two algorithms.

Through the comparison of average query information, we can figure out that $k$-agent search algorithm needs to inquiry more messages than other two algorithms for its additional agents. In theory, four agents are used in total means four times query information. In fact, there are only twice. The reason for this is the collision nodes we proposed before. When a collision node is found, an agent is terminated. Therefore, there will only be a few agents in the end, and the average query information is much lower than the theoretical value.

In Table 6, we list the results of search costs of different algorithms, our method is between the other two algorithms and shows a good performance. It is obvious that all three algorithms display a better result in BA networks. The reason is that this kind of model is

**Table 5** The comparison of average search steps, query information and search cost in differen algorithms

| Model \ Cost | $\overline{T}_{HDS}$ | $\overline{Q}_{HDS}$ | $\overline{T}_{k-HDS}$ | $\overline{Q}_{k-HDS}$ | $\overline{T}_{k-agents}$ | $\overline{Q}_k$ |
|---|---|---|---|---|---|---|
| WS | 12 | 112 | 12 | 106 | 7 | 214 |
| NW | 9 | 107 | 9 | 105 | 5 | 200 |
| BA | 7 | 103 | 7 | 105 | 6 | 186 |

**Table 6** The comparison of search cost in differen algorithms

| Model \ Cost | $C_{HDS}$ | $C_{k-HDS}$ | $C_{k-agent}$ |
|---|---|---|---|
| WS | 17.31 | 16.52 | 17.26 |
| NW | 14.39 | 14.49 | 15.33 |
| BA | 10.33 | 10.93 | 12.66 |

highly nonuniform, in other words, there exist nodes with a very high degree increasing the probability for a $k$-agent to search the useful information in a short time.

Figure 6 shows the results in histogram more clearly. $k$-agent in blue lines is nearly twice faster than other two algorithms especially in $WS$ and $NW$ models.

### *5.4.2 Successful agents*

Namely, one agent reaches the target node successfully in the final path. We call it "successful agent". Figure 7 shows the proportion of successful search times of each agent in the simulation above. Figure 7a is the total proportion of successful agent. Agent $k_1$ occupies 55 percent, $k_2$, $k_3$ and $k_4$ occupies 19, 16 and 10 percent respectively. Figure 7b, c and d display the detailed proportion of each agent in $WS$, $NW$ and $BA$ model.

Theoretically, each agent has the same probability to be the successful agent when the network is big enough, and the collision nodes don't exist. However, from Figure 7, we can see that agent $k_1$ is the most successful one to reach the target node. There are three reasons for this:

- The first reason is that the network itself isn't big enough. The target node is found within a few steps, $k_1$ has an advantage than other agents with $k$ steps of the random walk especially in $BA$ model;
- The other one is that we set $k_1$ to be the first agent to reach the target node in the best case. In fact, other agents get the target node at the same time;
- The final one is that the existence of collision nodes. When two agents meet the same collision node, the agent with smaller subscript is kept while the others are stopped. In this way, agent with higher smaller subscript has fewer chances to be the successful agent. Figure 7 confirms this view for $k_4$ is always the smallest part of the pie chat.

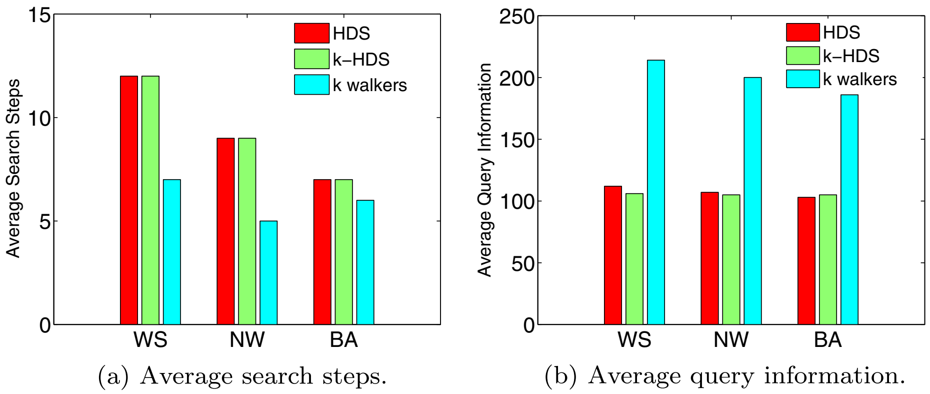
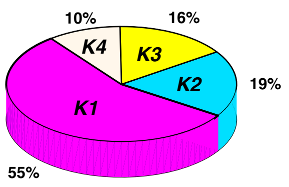
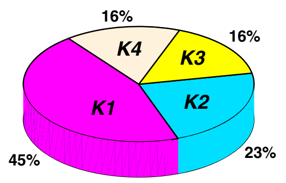


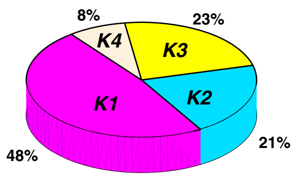
**Figure 6** The comparison of average search steps and query information in different algorithm



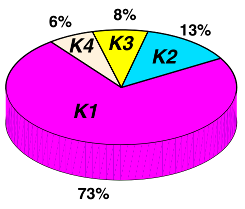
(a) The agents of all models.



(b) The agents of $WS$ model.



(c) The agents of $NW$ model.



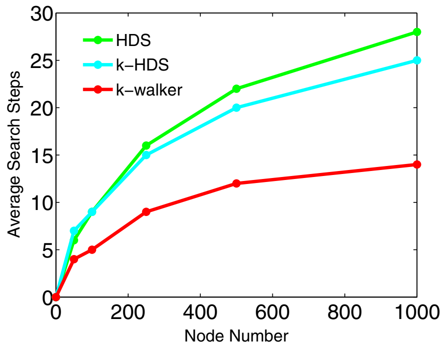
(d) The agents of $BA$ model.

**Figure 7** The proportion of successful agent in different models

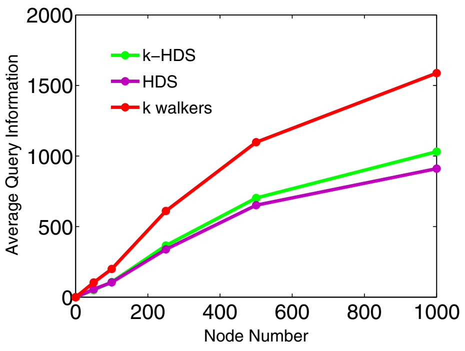
We study the successful frequency for each agent for the purpose of studying the value $\rho_i$, which is the probability of selecting each agent. It is mentioned in Section 4, we consider each agent has the same $\rho_i$, after this experiment, the empirical value for each model is obtained.

### *5.4.3 Scale analysis of the growth network*

In the last part of simulation, the comparison of average search step and query information among $k$ agents, $HDS$, $k$-$HDS$ with the growing of nodes $|V|$ is carried out.



(a) Trends of the average search steps.



(b) Trends of the average query information.

**Figure 8** The average search steps and query information of different nodes in $NW$ model

The $NW$ model is selected to be the background network. $HDS$, $k$-$HDS$ and $k$ agents with four agents will run in 50, 100, 250, 500 and 1000 nodes of the network in proper order, and the results are shown in Figure 8.

Figure 8a and b show the curves of average search steps and average query information in different algorithms, respectively. Green line denotes $HDS$, blue one denotes $k$-$HDS$ and red one denotes $k$-agent. The abscissa is the number of nodes while the ordinate stands for the value of average search step and query information.

From the Figure 8a, we can see that the average search steps increase smoothly with the growth of node number while. We can judge from the trend of the figure that the average search will finally go to a fixed value, and be much smaller than the other two algorithms for their rapid growth with the node number.

However, we can see the average query information will still grow from Figure 8b. The reason is that when average search step is stationary, the grow of nodes will make the degree increase and query information will grow. In theory, the agents are four times than $HDS$, hence the average search steps should be about one quarter, and the average query information should be about four times. However, the collision node terminates some of the agents, so the actual average search steps are nearly half and the average query information is nearly two times. After all, the effective and efficient is satisfactory.

Finally, the number of agents is discussed. Theoretically speaking, $k$ agents will reduce average search steps by $k$ times, for this reason, the more agents is taken and the fast the search is. Unfortunately, the existence of collision node makes it much less than $k$ times. And more importantly, the query information will rapidly increase with more agents. When the number of agents is large enough to the max degree of the network, this algorithm changes into broadcast search which searches all the network quickly but causes the network congestion. Therefore, if the query information is not important, that is to say, the $\lambda_q$ in (28) is quite small even tend to be 0, we could increase the number of agents to lower the search step.

## 6 Conclusions and future work

With the continuous rise of the network theory, network search gradually attracted people's attention. In this paper, a new search strategy based on $WS$, $NW$ and $BA$ models is proposed. Extra agents are added to accelerate the speed of search, and high degree seeking is used by every agent to fully take its advantage of the fast search in the high degree of aggregation networks. The complexity analysis is mathematically carried out respectively in best, worst and average situation. A new comprehensive parameter search cost for measuring search algorithm is also proposed. Finally, simulation results show the effectiveness and efficiency of the proposed methods.

However, additional agents require more query messages. This proposed algorithm appears to be unsuitable when network is easy to occur congestion. In fact, in the simulations, the average query information is below twice than high degree seeking, and the search cost is nearly same as other algorithms. If the query information is not important, such as social network, the proposed search strategy is quite efficient.

In the future work, how to determine the suitable number of agents, reduce the query information in order to cut the search cost, eliminate collision node for the purpose of faster search and if there exists a more suitable search strategy than high degree seeking for each agent are still issues worth exploring.

**Acknowledgments** This work is partially supported by the National Natural Science Foundation of China (Grant No.61602093).

## References

1. Adamic, L.A., Lukose, R.M., Puniyani, A.R., Huberman, B.A.: Search in power-law networks. Phys. Rev. E **64**, 046135 (2001)
2. Adamic, L.A., Lukose, R.M., Huberman, B.A.: Local search in unstructured networks. In: Bornholdt, S., Schuster, H.G. (eds.) Handbook of Graphs and Networks, pp. 295–317. Wiley-VCH (2003)
3. Al-asadi, T.A., Obaid, A.J., Hidayat, R., et al.: A survey on web mining techniques and applications. International Journal on Advanced Science Engineering and Information Technology **7**, 1178–1184 (2017)
4. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
5. Bennett, L., Liu, S., Papageorgiou, L.G., Tsoka, S.: Detection of disjoint and overlapping modules in weighted complex networks. Adv. Complex Syst. **15**, 1150023 (2012)
6. Berger, A., Lafferty, J.: Information retrieval as statistical translation. ACM SIGIR Forum **51**, 219–226 (2017)
7. Cai, B., Wang, H.Y., Zheng, H., Wang, H.: An improved random walk based clustering algorithm for community detection in complex networks. In: 2011 IEEE International Conference on Systems, Man, and Cybernetics, pp. 2162–2167 (2011)
8. Cajueiro, D.: Optimal navigation for characterizing the role of the nodes in complex networks. Physica A **389**, 1945–1954 (2010)
9. Cao, Y., Yu, W., Ren, W., et al.: An overview of recent progress in the study of distributed multi-agent coordination. IEEE Trans. Ind. Inf. **9**, 427–438 (2013)
10. Chau, M., Zeng, D., Chen, H., et al.: Design and evaluation of a multi-agent collaborative Web mining system. Decis. Support. Syst. **35**, 167–183 (2003)
11. Chen, D., Fan, Y., Shang, M.: A fast and efficient heuristic algorithm for detecting community structures in complex networks. Physica A **388**, 2741–2749 (2009)
12. Chen, L., Chen, J., Guan, Z., Zhang, X., Zhang, D.: Optimization of transport protocols in complex networks. Physica A **391**, 3336–3341 (2012)
13. Cohen, R., Havlin, S.: Scale-free networks are ultrasmall. Phys. Rev. Lett. **90**, 01 1–4 (2003)
14. Dorogovtsev, S.N., Mendes, J.F.F., Samukhin, A.N.: Structure of growing networks with preferential linking. Phys. Rev. Lett. **85**, 4633–4636 (2002)
15. Drias, Y., Pasi, G.: A collaborative approach to Web information foraging based on multi-agent systems. In: Proceedings of the International Conference on Web Intelligence, pp. 365–371 (2017)
16. Feng, M., Qu, H., Yi, Z.: Highest degree likelihood search algorithm using a state transition matrix for complex networks. IEEE Trans. Circuits Syst. Regul. Pap. **61**, 2941–2950 (2014)
17. Feng, M., Qu, H., Yi, Z., et al.: Evolving scale-free networks by poisson process: modeling and degree distribution. IEEE Transactions on Cybernetics **46**, 1144–1155 (2016)
18. Gao, L., Guo, Z., Zhang, H., Xu, X., Shen, H.: Video captioning with attention-based LSTM and semantic consistency. IEEE Trans. Multimedia **19**, 2045–2055 (2017)
19. Gao, L., Song, J., Liu, X., Shao, J., Liu, J., Shao, J.: Learning in high-dimensional multimedia data: the state of the art. Multimedia Systems **23**, 303–313 (2017)
20. Hughes, B.D.: Random Walks and Random Environments. Clarendon Press, Oxford (1996)
21. Jasch, F., Blumen, A.: Target problem on small-world networks. Phys. Rev. E **63**, 041108 (2001)
22. Kim, B.J., Yoon, C.N., Han, S.K., Jeong, H.: Path finding strategies in scale-free networks. Phys. Rev. E **65**, 027103 (2002)
23. Kleinberg, J.M.: Navigation in a small world. Nature **406**, 406–845 (2000)
24. Krapivsky, P.L., Redner, S., Leyvraz, F.: Connectivity of growing random networks. Phys. Rev. Lett. **85**, 4629–4632 (2000)
25. Liu, M., Xu, Y., Mohammed, A.W.: Decentralized opportunistic spectrum resources access model and algorithm toward cooperative ad-hoc networks. PloS one **11**, e0145526 (2016)
26. Liu, X., Li, Z., Deng, C., Tao, D.: Distributed adaptive binary quantization for fast nearest neighbor search. IEEE Trans. Image Process. **26**, 5324–5336 (2017)
27. Lu, L., Zhou, T.: Link prediction in complex networks: a survey. Physica A **390**, 1150–1170 (2011)
28. Newman, M.E.J., Watts, D.J.: Renormalization group analysis of the small-world network model. Phys. Lett. A **263**, 341–346 (1999)
29. Rosvall, M., Trusina, A., Minnhagen, P., Sneppen, K.: Hide-and-seek on complex networks. Europhys. Lett. **69**, 853–859 (2005)

30. Saini, S., Pandey, H.M.: Review on Web content mining techniques. Int. J. Comput. Appl. **18**, 118 (2015)
31. Sharma, D.K., Sharma, A.K.: Deep Web information retrieval process: a technical survey. IJITWE **5**(1), 1–22 (2010)
32. Shi, C., Yan, Z.Y.: A genetic algorithm for detecting communities in large-scale complex networks. Adv. Complex Syst. **13**, 3–17 (2010)
33. Song, J., Gao, L., Liu, L., Zhu, X., Sebe, N.: Quantization-based hashing: a general framework for scalable image and video retrieval. Pattern Recogn. **75**, 175–187 (2018)
34. Thadakamalla, H.P., Albert, R., Kumara, S.R.T.: Search in spatial scale-free networks. New J. Phys. **9**, 190 (2007)
35. Wang, X., Gao, L., Wang, P., Sun, X., Liu, X.: Two-stream 3D convNet fusion for action recognition in videos with arbitrary size and length. In: IEEE Transactions on Multimedia. In press (2018)
36. Watts, D.J., Strogatz, S.H.: Collective dynamic of 'small-world' networks. Nature **393**, 440–442 (1998)
37. Watts, D.J., Dodds, P.S., Newman, M.E.J.: Identity and search in social networks. Science **296**(5571), 1302–1305 (2002)
38. Witten, I.H., Frank, E., Hall, M.A., et al.: Data mining: practical machine learning tools and techniques, p. 223. Morgan Kaufmann, Burlington (2016)
39. XinLing, S., LiJun, Z.: Search in complex networks with local efficient information. In: International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), pp. 359–362 (2011)
40. Zhu, X., Zhang, S., Hu, R., Zhu, Y., Song, J.: Local and global structure preservation for robust unsupervised spectral feature selection. In: IEEE Transactions on Knowledge and Data Engineering. In press (2018)