A Self-organization Method for Reorganizing Resources in a Distributed Network

Xiaolong Guo and Jiajin Huang

International WIC Institute, Beijing University of Technology,
Beijing, 100124, China
{gxlvip,hjj}@emails.bjut.edu.cn

Abstract. Using bio-inspired agents to reorganize resource has been adopted to address the distributed resource optimization issue in distributed networks. This paper presents a self-organization algorithm to reorganize resources by the use of autonomous agents which can exchange their resources each other. Agents are equipped with two operations (pull and push operation) and three behaviors (best selection, better selection and random selection). And at every moment, agents probabilistically choose a behavior to perform. Experimental results indicate that the strategy has a positive influence on system performance.

1 Introduction

A distributed network (e.g. P2P networks) consists of a number of connected nodes while there is no central controller in the distributed network. So how to find the resource descriptors in the distributed network becomes a key problem because those resources of the network need be obtained and accessed effectively [1]. To discovery a number of useful descriptors in a short time, a solution is to put similar resource descriptors together effectively in restricted regions by the use of a self-organization method.

How to solve these problems in distributed environments is also a key problem in Web Intelligence [2,3]. By using bio-inspired algorithms to reorganize resource has been adopted in distributed networks. The So-grid system employs several mobile agents to reorganize resources. In So-grid, the agent travel the Grid through P2P interconnections. While the agent is traveling, it picks up the resource descriptor if the resource descriptor is different from others in the available region and it drops down the resource descriptors at appropriate nodes [1]. As same as mentioned in [4], agents in the So-grid system may make a large amount of random movements before they pick up or drop down resource descriptors, therefore some nodes may not be visited. In this case, these descriptors in non-visited nodes may not be placed at suitable nodes. The work of [5] uses an agent-view to connect the agent society. An agent-view contains agents whose contents are similar. However, these agents exchange their all contents in their agent-views with each other, which results in much more communication costs. In order to overcome the shortcoming of the above work, this paper presents a method to reorganize descriptors by using agents that reside on the nodes. These agents can (1) select descriptors which are not suitable to stay the node and then push these descriptors to other nodes until that a suitable node can keep these descriptors; (2) decide whether they keep these descriptors or not according to their local environments when they receive descriptors pushed by other agents. This paper is partly inspired by the work of [4], who proposed an ant clustering algorithm in which data are placed in a 2-Dimension Grid and an agent represents a data object. However, our work is different from the application domain of [4]. Our work faces a large-scale, dynamic and distributed network environment. So we need a method to be adapted to the network environment.

In [4], agents can be clustered dynamically by collaboration based on local information in a 2-Dimension Grid. In other words, moving behaviors of agents have autonomy features. An autonomous agent system includes the environment of agents, agent's profiles and the behavior rules of agent [6,7]. In this paper, we design a kind of agent which resides on the node of a network. An agent's local environment is the set of descriptors in a restricted region. Agents push or pull the resource descriptors to put similar resource descriptors together dynamically. By the certification of experiments, the method is valid in networks with different structures, different scales, and dynamically changing numbers of nodes.

2 The Formulation of the Self-organization Strategy

According to [6,7,8], an agent essentially has the following properties: it can live in an environment; it can sense its local environment since there is no central controller and it has some behaviors driven by certain purposes.

A network is represented as graph $G =< V, E >$, where $V = \{v_1, v_2, \cdots v_N\}$ is a set of nodes and $L = \{(v_i, v_j)|1 \leq i, j \leq N, i \neq j, v_i, v_j \in V\}$ is a set of edges. $N = |V|$ represents the total number of nodes in a network, and (v_i, v_j) indicates that an edge exists between node v_i and v_j. As a result, v_i and v_j are so-called neighbor nodes with each other.

The profile of node v is defined by $v = \{nodeID, \Gamma, D\}$, where $nodeID$ denotes the identifier of a node, Γ represents total neighbors of node v ($v.\Gamma = \{v_i|(v, v_i) \in L\}$), D represents the set of descriptors in node v.

The local environment of agent a_i is defined as E_l, which is a set of resource descriptors both in the node resided by agent a_i and the node's neighbors. At time t, the local environment of agent a_i residing on node v_i is $E_l(a_i(t)) = \{d|d \in \bigcup v.D, v \in v_i.\Gamma \bigcup \{v_i\}\}$. Corresponding to the agent's local environment, we will set the entire network G as an agent's living environment E_g. It is impossible for agents to know about the situation of E_g in distributed environments. Therefore the local environment is most important for an agent. Each agent can only sense their own local environments and the local environments E_l could be better if there are more same class of descriptors. Each agent's local environment E_l is not static, which can be modified by other agents' behaviors which are defined by the profile of agent a_i.

The profile of agent a_i is $\{ID, nodeID, behaviors, rules\}$, where ID denotes the identifier of the agent $a_i.nodeID$ represents the identifier of a node resided by agent $a_i.behaviors$ includes the pull and push behaviors. According to the local environment, an agent decides whether it pushes resource descriptors from the node resided by it to other nodes or pull the suitable resource descriptor to the node resided by it from other nodes. Thus, $rules$ are used to decide the two behaviors. By these behaviors, each agent changes their local environments. Agents interact with each other in the changing local environments.

Borrowing the concept of fitness measure in [1,9], we also let agent a decide whether it will pull or push the descriptor d at time t based on a fitness function $f_a(d,t)$ given by

$$f_a(d,t) = \frac{|E_l(a(t))|}{R_d}, \tag{1}$$

where R_d is the number of the descriptors of the whole class in the local environment $E_l(a(t))$ of agent a at time t, and $|E_l(a(t))|$ is the overall number of descriptors that are accumulated in the local environment at time t. From Equ. (1), we can see the more there are the same class of descriptors d in the local environment of agent a, the more the descriptor d is suitable in the node resided by agent a.

At time t, agent a selects descriptor $d \in a(t).nodeID.D$ with least fitness $f_a(d,t)$ and then evaluates push probability function to decide whether or not to push descriptor d out of the resided node. The push probability is shown by

$$p^a_{push}(d,t) = (\frac{k_1}{k_1 + f_a(d,t)})^2, d \in a(t).nodeID.D. \tag{2}$$

When f is much lower than the constant k_1 (i.e. there are no descriptors as same class as descriptor d in the local environment of agent a), the push probability is higher. As more descriptors as the same class as descriptor d are accumulated in the local environment of agent a, the fitness of the descriptor d increases. With the increase of the fitness (specially f is much higher than k_1), the value of the push probability for descriptor d becomes lower. So at time t, given a random value r, agent a does not select descriptor d when $p^a_{push}(d,t) < r$; agent a finds the node is not suitable to descriptor d when $p^a_{push}(d,t) > r$ and then will push descriptor d out of the node. One of three push behaviors based on the primitive behaviors of autonomous agents are probabilistically performed [6,7]. (a) agent a pushes the descriptor d to the node selected randomly from $a.nodeID.\Gamma$. (b) agent a pushes the descriptor d to the node with the most f for the descriptor from $a.nodeID.\Gamma$. (c) agent a randomly selects two nodes from $a.nodeID.\Gamma$, and then push the descriptor to the node with more f for the descriptor d between the two nodes.

When agent a receives a descriptor d from agent a' at time t, the agent evaluates pull probability function in order to decide whether it will pull descriptor d to the node resided by it or not. The pull probability is shown by

$$p^a_{pull}(d,t) = (\frac{f_a(d,t)}{k_2 + f_a(d,t)})^2, d \in a(t).nodeID.D. \tag{3}$$

As opposite to the push probability, the more same descriptors are accumulated in the agent's local environment, the higher the pull probability for this class of descriptors becomes. When $p^a_{pull}(d,t) < r$, the agent does not receive the descriptor d. When $p^a_{pull}(d,t) > r$, the agent receives the descriptor d. Let $D^a_{pull}(t)$ be the set of descriptors pulled by agent a from others nodes at time t. And at the next time $t+1$, we have $a(t+1).nodeID.D = a(t).nodeID.D \bigcup D^a_{pull}(t)$. By these behaviors, each agent's local environment are changed. At the next time, agents decide their behaviors in new local environments. From this perspective, we can see agents interact with each other in the changing local environment.

The measures of push and pull probability function are similar to the pick and drop probability function shown in [1,9]. However, agents in [1,9] can move in the enviroment, where agents in this paper need not move in the network and each agent resides in a node of the network. The push and pull operations could be embedded in the push-pull protocol of P2P [10] to implement a real-world application. According to the algorithm framework in [1,4,9], we summarize our algorithm as Algorithm 1.

Algorithm 1. A Self-organization Algorithm for Reorganizing Resources

```
01. Initialize the parameters
02. FOR $t$=1 to $T$
03.  FOR each agent $a$ DO
04.   $a(t).nodeID.D = a(t-1).nodeID.D \bigcup D^a_{pull}(t-1)$
05.   $D^a_{pull}(t) = \emptyset$
06.   IF agent $a$ receives the non-acceptance message from other agents
07.     Keep descriptors pushed by agent $a$ at time $t$-1
08.   END IF
09.   Select a descriptor $d$ in $a.nodeID.D$ with least fitness $f_a(d,t)$
10.   Compute the push probability $p^a_{push}(d,t)$
11.   $r$=random([0,1])
12.   IF $p^a_{push}(d,t) > r$
13.   Probabilistically determine a primitive behavior to select a node in $a(t).nodeID.\Gamma$
14.    Push descriptor $d$ to the node
15.   END IF
16.   FOR each descriptor $d'$ pushed by other agents
17.    Compute the pull probability $p^a_{pull}(d',t)$
18.    $r'$=random([0,1])
19.    IF $p^a_{pull}(d',t) > r'$
20.     $D^a_{pull}(t) = D^a_{pull}(t) \bigcup \{d'\}$
21.    ELSE
22.     Send the non-acceptance message to the agent pushing descriptor $d'$
23.    END IF
24.   END FOR
25.  END FOR
26. END FOR
```

3 Experiment

To prove the effectiveness of the proposed algorithm, we set a series of experiments to answer the following questions. Can the strategy remain effective in different networks? The networks include networks generated by different network generators, networks with different numbers of networks, and networks with the dynamically growing scale. Firstly, we get a network topology by the use of a network generator, store a certain number of descriptors on nodes, and then assign each agent in each node. We use three kinds of complex networks which are listed in the Table 1. The GLP [11] and WS [12] networks are used to test the effectiveness in networks generated by different network generators and networks with different numbers of networks. The active network is used to test the effectiveness networks with the dynamically growing scale. Then, we let agents operate the resource descriptors by the push or pull operation as shown in Algorithm 1.

Table 1. The structures of networks

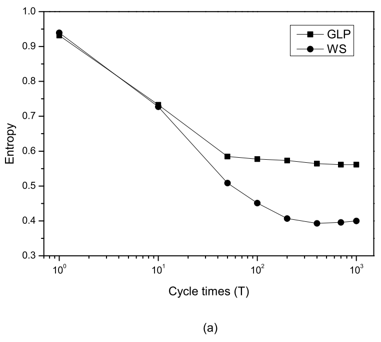
Network	Parameter
GLP	Power law network with α=2
WS	Small World network (the neighbor connection is 5)
Active	Dynamic network (increasing 20 nodes at a moment)

The effectiveness of the strategy is evaluated trough a spatial entropy function. For agent a, let $fr_a(i)$ denote the percent of descriptors of class C_i in the local environment of agent a and Nc denote the number of classes. The local entropy $En_l(a)$ [1] is given by

$$En_l(a) = \frac{-\sum_{i=1}^{N_c} fr_a(i) \lg fr_a(i)}{\lg N_c}. \tag{4}$$

Based on the local entropy, the overall entropy Eo is defined as the average of En_l of all agents. According to the well-known Shannon's formula, the more minimal is the Eo, the more effective is the strategy. In each experiment, we run the algorithm ten times and the shown results are an average value of the ten results.

The experiment about the effectiveness in different kinds of networks is done in WS and GLP networks. In each kinds of network, the numbers of nodes are 1000 and 2000 respectively. The results are shown in Fig. 1. Fig. 1(a) shows that the value of overall entropy decreases from 0.96 to 0.4 after 1000 cycle times (T=1000) and then tends to a stable state in the WS network with 1000 nodes; the value of overall entropy is stable at 0.6 in the GLP network with 1000 nodes. Fig. 1(b) shows the similar result in networks with 2000 nodes. From Fig. 1,



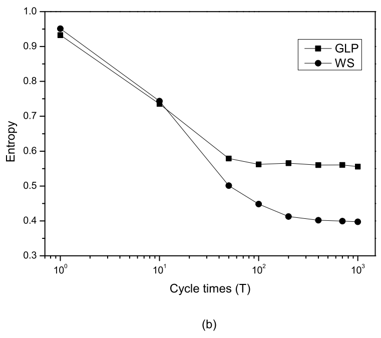
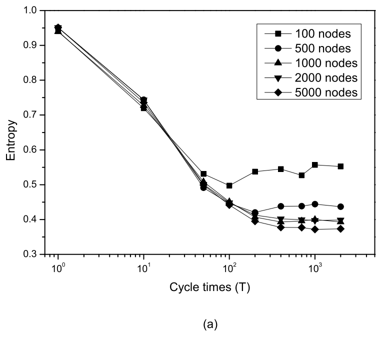


Fig. 1. Results in two kinds of networks. (a) results in networks with 1000 nodes; (b) results in networks with 2000 nodes.

we can see that the presented algorithm is useful for the WS network and the ability of clustering is better in the WS network.

The results of the experiment in the WS networks with different number of nodes are shown in Fig. 2. The number of nodes is from 100 to 5000. Fig. 2(a) shows that the value of overall entropy becomes stable with the increasing of the cycle times. And from Fig. 2(b), we know that the results of experiment could be better if the network contains more nodes.



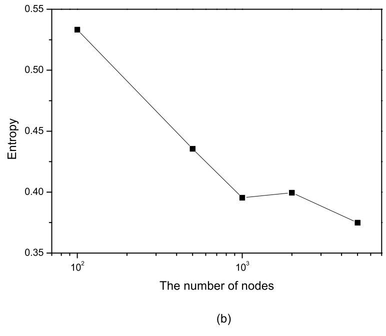


Fig. 2. Results in a kind of network with different numbers of nodes

The results of the experiment in the active network are shown in Fig. 3. The active network is a WS-based network with dynamically increasing 20 nodes at a moment. From Fig. 3, we can see that the value of overall entropy is about 0.4 at the stable state which is as same as the result in WS networks shown in Fig. 1 and Fig. 2. These results of experiments tell us that the ability of the clustering is similar whatever the network is dynamic or not.

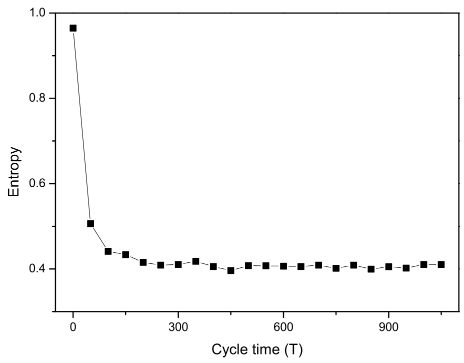


Fig. 3. Results in a dynamic network

The experimental results of relevant parameters are shown in Fig. 4. The network is a WS network with 1000 nodes. Fig. 4 shows that the value of entropy is lower and the result is better when the parameter k_1 is bigger than the parameter k_2.

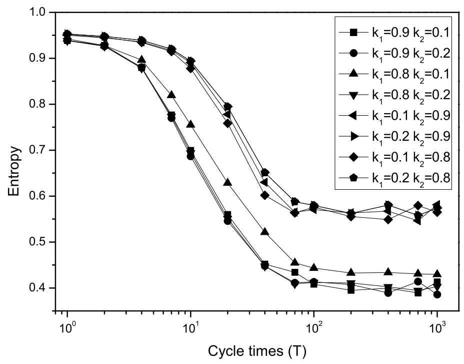


Fig. 4. Results with different combinations of k_1 and k_2 parameters

Fig. 5 shows how the random selection affects the result. We set the ratio of random selection is 0%, 5%, 10%, and 15% respectively. From Fig. 5, we can see that the random selection behavior affects the value of overall entropy and the stability of the entropy curve. When the percent of the random selection is lower, the overall entropy value is lower. But the entropy value is a little higher if there is lack of the random selection, as the strategy maybe stuck in a local optima without random selection [7].

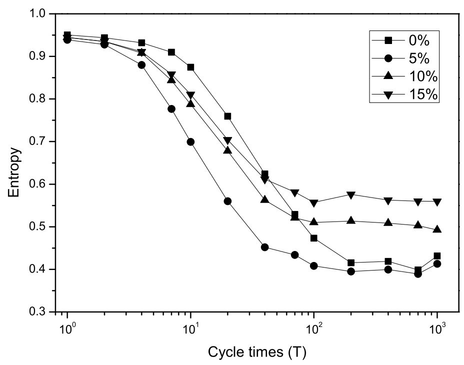


Fig. 5. Results with different assignment of random selection

4 Conclusions

This paper presents a self-organization algorithms for the resource reorganization in a decentralized network which is no central control. Inspired by the work of [9], two different kinds of agent behaviors are defined as push and pull operation. These behaviors change agents' local environments and agents exchange descriptors with each other. Simulated experiential results showed that the algorithm is effective in the controlled propagation and reorganization of information. In this primary method, we only transplant the similar data cluster method of [4] into the network environment. The future work will focus on how to balance data distribution in nodes and how to design effective positive feedback [7] and self-learning [8] principles for agents to speed up the reorganization.

Acknowledgements. This work is supported by Beijing Natural Science Foundation (4102007), the CAS/SAFEA International Partnership Program for Creative Research Teams, the China Postdoctoral Science Foundation Funded Project (2012M510298), Projected by Beijing Postdoctoral Research Foundation (2012ZZ-04), and the doctor foundation of Beijing University of Technology (X0002020201101).

References

1. Forestiero, A., Mastroianni, C., Spezzano, G.: So-Grid: a Self-organizing Grid Featuring Bio-inspired Algorithms. ACM Transactions on Autonomous and Adaptive System 3(2), 1–37 (2008)
2. Zhong, N., Liu, J.M., Yao, Y.Y.: In Search of the Wisdom Web. IEEE Computer 35(11), 27–31 (2002)
3. Zhong, N., Liu, J.M., Yao, Y.Y.: Envisioning Intelligent Information Technologies (iIT) From the Stand-Point of Web Intelligence (WI). Communications of the ACM 50(3), 89–94 (2007)

4. Xu, X.H., Chen, L., He, P.: A Novel Ant Clustering Algorithm Based on Cellular Automata. Web Intelligence and Agent Systems 5(1), 1–14 (2007)
5. Zhang, H.Z., Croft, W.B., Levine, B.N., Lesser, V.R.: A Multi-agent Approach for Peer-to-peer Based Information Retrieval System. In: Proceeding of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), pp. 456–463 (2004)
6. Gao, C., Liu, J.M., Zhong, N.: Network Immunization with Distributed Autonomy-oriented Entities. IEEE Transactions on Parallel Distributed Systems 22(7), 1222–1229 (2011)
7. Liu, J.M., Jin, X.L., Tsui, K.C.: Autonomy Oriented Computing (AOC): from Problem Solving to Complex System Modeling. Kluwer Academic Publishers (2005)
8. Liu, J., Zhong, W.C., Jiao, L.C.: A Multiagent Evolutionary Algorithm for Combinatorial Optimization Problems. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 40(1), 229–240 (2010)
9. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: from Natural to Artificial Systems. Oxford University Press (1999)
10. Gueret, C.: Nature-Inspired Dissemination of Information in P2P Networks. In: Abraham, A., et al. (eds.) Computational Social Network Analysis, pp. 267–290 (2010)
11. Bu, T., Towsley, D.: On Distinguishing between Internet Power Law Topology Generators. In: Proceeding of 2002 IEEE International Conference on Computer Communications (INFOCOM 2002), pp. 638–647 (2002)
12. Watts, D.J., Strogatz, S.H.: Collecive dynamics of 'small-world' networks. Nature 393, 440–442 (1998)