TELKOMNIKA, Vol.11, No.1, January 2013, pp. 48~56
ISSN: 2302-4046

A new Graph To Represent Complex Plan Resolution Domain

Ping LIU*, HaiFeng YU, BoYi XU, Jin LI, Qi JIN, Hong TANG
Science and Technology on Information System Security Laboratory
Beijing Institute of System Engineering
Beijing 9702 box 19# China (100101),
*corresponding author, e-mail: liuzuhan@126.com

Abstract

Taking the botnet incident response plan decision-making as an example, we construct a new graph model to represent complex plan resolution domain by using action decomposition. In this model we introduced action layer, relation between subactions and decompose the complex incident response plan into many layers. This graphic model provides a powerful representing framework on the complex plan domain. By analyzing the decomposing method and the action node structure, we prove that the store space size of this graphic representation is not increased exponentially when the action node added. We present a quantitatively analyzing algorithm on the plan resolution domain.

Keywords*: plan resolution domain, action decompose, graph theory, quantitatively analyze*

1. Introduction

More and more attacks aiming at Web service, DDoS attack is the main cause that result in Web crash. Botnet is an important technical means being used to launch DDoS attacks. Taking the botnet incident response plan decision-making as an example, we construct a new graphic model by using action decomposition. Because relations between actions and action layer are introduced in the model, so we can decompose the complex incident response plan into many layers. This graphic model also provides a representation of the complex plan resolution domain. The complex incident response plan can also be quantitatively analyzed with this model and it enables us to make incident response plan decision more scientific.

Incident response start from the current abnormal state of network system, by using of a series of commands and software tools and executing a series of actions, at last making network system to achieve its normal state. So the incident response plan decision–making is a planning process. The incident response plan is action sequence that satisfying certain constraints.

The action decomposition method being used in incident response plan representing graphic model has the same principles as that used in [2]. Some actions in the incident response plan can be executed in a free order, so the relation between the actions of incident response plan is a partial order. Using action decomposition we get subaction layers. Some subactions in the same layer have relations and form a graph. An action and subactions decomposed from it form a tree. We can use graph algorithm to optimize and analyze the incident response plan.

In section 2 we explain the research method of constructing complex plan domain representing graph; Section 3 describes the concrete method to constructing our representing graph. In section 4 we present quantitatively analyzing algorithm of the complex plan resolution domain. Section 5 is the conclusion.

2. Research Method

As described in [3] that consciousness cannot be reduced to a single neuron. For the complex incident response plan, its property cannot be reduced to a single action too, and we should analyze the total actions and the relations between these actions. Decomposing the

Received August 27, 2012; Revised November 20, 2012; Accepted November 27, 2012

action into subaction array and constructing six types of relation for these subactions, we can quantitatively analyze the complex incident response plan. The similar analyzing graphic model can be found in [4].

In the graphic modeling process an important factor of human is ignored. In fact the integration of human and machine is a key approach when analyzing complex system. We introduce expert group to our graphic model.

2.1. Equivalent

Definition 1. Equivalent relation

If a dualistic relation on P ≈ meet the following three axioms:

(1) for all x ∈P, x ≈ x (reflexivity)

(2) if x ≈ y, y≈ x (symmetry)

(3) if x ≈ y and y ≈ z then x ≈ z (transitivity)

The dualistic relation ≈ is called an equivalent relation on P.

Let E be a set of system S and its simulative model, the checked attribute set of E's is: A={ai |i=1,2,3,…,n}. An simulative model M of the system S on attribute set A is ideal, if the following relation is satisfied:

fi (S)= fi (M), i=1,2,3,…,n.

Here fi (E)= ai, i=1,2,3,…,n. are attribute extracting functions that can be used to define equivalent relation on set E as depicted in Figure 1.

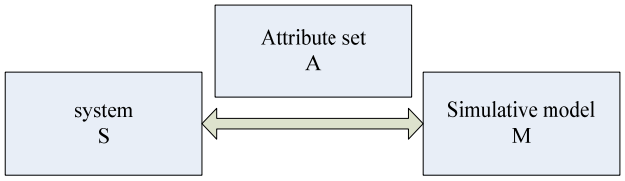

Figure 1. Equivalent relation between S and its simulative model M

2.2. Action Decompose Method

Action decomposition is a process decomposing a complex action into many subactions that satisfy certain relation constraint. For example, when facing a complex action A, different expert would provide different executing plan. In order to present this diversity executing plan we decompose A into many subactions, every subaction presents an executing plan and the relation between subactions is OR. The relation between subactions is defined as following:

Definition 2. Relation AND, OR and ORDER between subactions

The relation between subactions decomposed from one action is called AND, if we want to complete the action, we must execute all of its subactions. The relation between subactions decomposed from one action is called OR, if we want to complete the action, we must execute one and only one of its subactions. The relation between subactions decomposed from one action is called ORDER, if we want to complete the action, we must execute all of its subactions in a certain order.

Definition 3. Relation ORDERAND

The relation between subactions decomposed from one action is called ORDERAND (Figure4), if subactions have ORDER and AND relation. Relation ORDERAND is a combination of relation ORDER and AND.

In action decomposing process the participation of domain expert is very important. The data show that if we build a supper computer which has the same computing ability as human brain, it would need a nuclear power plant to support its electricity consuming.

Botnet is the primary technical means of implementing DDoS attacks. In order to describe the action decomposing method, we use IRC botnet incident response as an example. At first we according to the following characteristics of the network to determine if an attack of IRC botnet is occurred:

(1) the network connection greatly increase

(2) machine was significantly slower

(3) the launched connection from local IP to the same port of multiple targets IP appears.

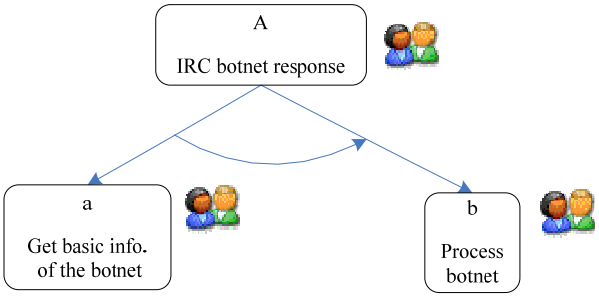

Figure 2. Subaction relation type ORDER

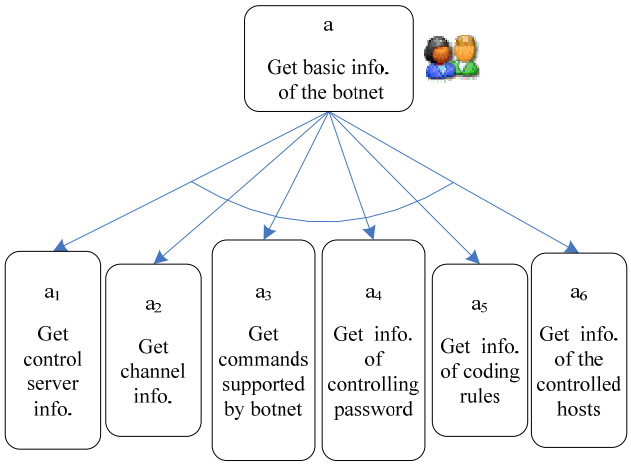

Figure 3. Subaction relation type AND

Supposing action A (IRC botnet incident response) changes the network system that attacked by IRC botnet to its normal state. The action A is put in the top-level of action decomposition. We now present the action decomposing method and how to represent the relation between subactions[5].

As depicted in Figure 2, IRC botnet incident response action A is decomposed into two ordered actions: a (get the basic information of the IRC botnet) and b (dealing with IRC botnet).

In Figure 3 action a is decomposed into a1 (get information of the control server), a2 (get channel information), a3 (get information of the command set supported by IRC botnet), a4 (get information of controlling password), a5 (get information of coding rules) and a6 (get information of the controlled hosts). Here the executing order of actions a1, a2, a3, a4, a5 and a6 is free.

Figure 4 shows the decomposition of action b (dealing with IRC botnet), which is divided into action b1, b2 and b3. Note the executing order constraints of action b1, b2 and b3. Action b1 must be executed before b2, but no order constraint of b3.

The action b1 (cut off the connection between host and controlling server) is decomposed into action b11 and b12 (shown in Figure 5). The relation between b11 and b12 is OR. This minds only one action (b11 or b12) will be executed in the incident responding process.

As depicted in Figure 3 we use an arc to represent that the relation between subactions is AND. The action a1, a2, a3, a4, a5 and a6 are named the next-layer of action a. Action decomposition would go on until all the subactions being atomic action.

Definition 4. Relation SUPPORT and SUPPORTORDER

The relation between action x and action y is called SUPPORT, if we want to complete the action x, we must beep action y is executing. The relation between action u and action set V is called SUPPORTORDER, if we want to complete the action set V, we must beep action u is executing, and the actions in V has the relation type of ORDER. As depicted in Figure 6, V={b2,b3} and u=b1.

Definition 5. Action set V, W and action s

The subactions of an action can be divided into three parts: an support action s, the supported action set V and the others W. Subactions in V and W may have relation type: AND, OR or ORDER.

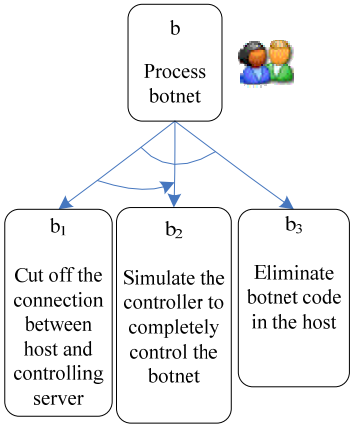

Figure 4. Subaction relation type ORDERAND

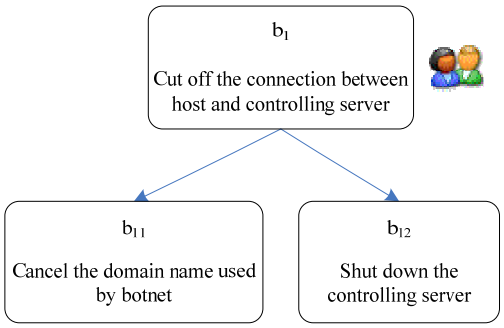

Figure 5. The subaction relation type of OR

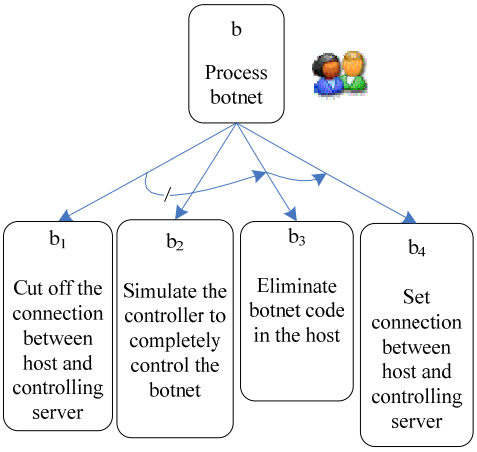

Figure 6. The subaction relation type of SUPPORTORDER

3. Representing Graph

As described in [3] in order to facilitate computer processing, first we need to build a graph to represent complex plan resolution domain that can be represented by computer. The size of the model and action nodes should have a linear relation [6], thus avoiding the exponential explosion growth of the storage space size with the increase of action nodes. The

implementing property that we need to look at of the plan resolution domain must be presented in the representing Graph.

The complex plan resolution implementing property studied in this paper is time consumed by the plan. The remaining property can be studied in the same way [7].

A complex incident response action is decomposed into several simpler subactions and we say they belonging to the same layer. Thus we have introduced a hierarchy of action.

Using the concept of action layer, we can represent the project implementation plan with different level, so we cannot only to examine the overview of the project implementation, but also to examine the different plan implementing details. Such a representation of complex project plan presents a method for us to use outside storage algorithms [8-10].

There are six types of relation between subactions: the OR, AND, ORDER, ORDERAND, SUPPORT and SUPPORTORDER as defined in definition 2, 3 and 4 respectively. The number of subactions decomposed from one action node cannot be too much, so we can observe and handle the action decomposition easily with interactive method. Of course, the relation type of subaction can be defined if you want. You have to give the corresponding quantitative calculation method.

Different subaction that belongs to the same subaction group with OR relation will result in the change of the plan resolution. Another factor that shorten the implementing time of incident response plan is executing the subactions that satisfy AND relation in parallel. So we get different plan with the longest and the shortest implementing time.

3.1. The Representation of Action Node

The action node consists of two parts, the action node attribute values, as well as a pointer to the subaction array. In the algorithm we use an integer ACTION_ID to identifier an action node.

Based on the above analysis, we give the action node representation as follows:

```
#define ACTION_ID  long  //data type
#define MAX_SUB_ACTION_NUM  16  //a constant number
struct action_node {
    //Action identifier,
    1)  ACTION_ID id;
    2)  string Action name;
    3)  Boolean bAtomic; //not to decompose atomic action
    // bFlag is TRUE only if the min_time and max_time of an action
    //have been calculated
    4)  Boolean bFlag;
    5)  Group of experts;
    6)  software tools to perform the action; // non-atomic action can be null
    7)  int layerNumber;// to indicate the action on which layer
    //if we calculate the max value of an action attribute
    8)  boolean bMaxValue;
    //if the action is an atomic action then we set min_time=max_time
    9)  int min_time; // the shortest time for performing this action
    10) int max_time; // the longest time for performing this action
    // a pointer to an array of subaction; pActionArray=NULL for the atomic action
    11) struct action_array *pActionArray;
};
```

Note that the action identifier is different to the actual action. Action identifier is globally unique.

3.2. Subaction Array

We use the following structures to define our subaction array.

```
struct action_array {
        // A pointer to the parent action: In the action decomposition hierarchy
        // use this pointer to reach the parent action layer.
        struct action_node *pParent;
        struct sub_action_relation subactionrelation; // to describe the relation type
        // to hold our subactions
```

```
        struct action_node actions[MAX_SUB_ACTION_NUM];
};

struct sub_action_relation {
        int iSupportedSubactionRelationType;//can be relation:  AND OR ORDER
        ACTION_ID SupportedSubactionID[MAX_SUB_ACTION_NUM];
        //the relation type between surport subaction and other subactions
        int iSupportSubactionRelationType; //can be relation:  AND OR ORDER
        // to hold our subactions that have relation type AND
        ACTION_ID ANDSubactionID[MAX_SUB_ACTION_NUM];
        // to hold our subactions that have relation type OR
        ACTION_ID ORSubactionID[MAX_SUB_ACTION_NUM];
        // to hold our subactions that have relation type ORDER
        ACTION_ID ORDERSubactionID[MAX_SUB_ACTION_NUM];
};
```

3.3. The Expert Group Relation

In the decision making of the complex plan resolution many groups of expert are involved. Different group proposed different plan. The relation between groups of expert can be represented by a matrix C. C (i ,j) = 1 presents expert group i and j have relation; C(i,j) = 0 present expert group i and j have no relation. The number of expert group is less than a given integer G. The store space size of matrix C is less than G*G.

3.4. The Elements in representing graph

Our representing graph provides a representating framework of the complex plan domain. If the subactions x,y,and z have OR relation type, we get three plan resolutions that are x,y,z respectively. For the same reason if the subactions x,y,and z have AND relation type, we get 7 plan resolutions that are xyz, xzy, yxz, yzx, zxy, zyx and an concurrent action [zxy] respectively. If the subactions x,y,and z have ORDER relation type, we get only one plan resolution that is xyz. So we can see that the graph is of powerful representing ability.

You can define your own subaction relation type as needed, but you should give the corresponding calculating function.

3.5. Graph Store Space Size

Let V is the store space size occupied by our plan decision-making model. By analyzing the decompose method and the action node structure, we can prove that V do not increase exponentially when the action node added.

If we use the decomposed method to get a plan P that has N atomic actions (if an atomic action appears j times, we count it j times) . An non atomic action include at least 2 subactions. The plan model has M action nodes, so we can conclude M≤2N. Since the store space size occupied by an action node or a subaction array is less than a constant K, and the number of subaction array is less than M, so we conclude that:

$V \le KM + KM + G*G \le 2KM + G*G \le 4KN + G*G$

4. Quantitatively Analyzing Algorithm

We get the plan resolution domain D by decomposing the action. Analyzing the element in D we can get the optimal resolution. According to the definition of subaction relation type an algorithm is presented in Figure 7.

The current action C is the action node that we are analyzing. At the beginning in Figure 7 the action A is defined in Figure 2. Using function R_type(C) we get the relation type of subactions that decomposed from action C. The variable of bFlag and bAtomic is defined in struct action_node.

4.1. ORDERAND_order_num

The number of subactions that have ORDER relation in the subactions which have ORDERAND relation. The array subaction_ORDERAND[ORDERAND_order_num] denotes

these subactions. If the parent action of these subactions is P, Let k= ORDERAND_order_num , then we get the P.min_time and P.max_time as formula (1) and (2):

$$P.min\_time = \max\{\sum_{i=1}^{k} sub\_action\_ORDERAND[i].min\_time,$$

$$\max\{ sub\_action\_ORDERAND[j].min\_time \mid j=k+1,\ldots,n\} \} \quad (1)$$

$$P.max\_time = \sum_{i=1}^{n} sub\_action\_ORDERAND[i].max\_time \quad (2)$$

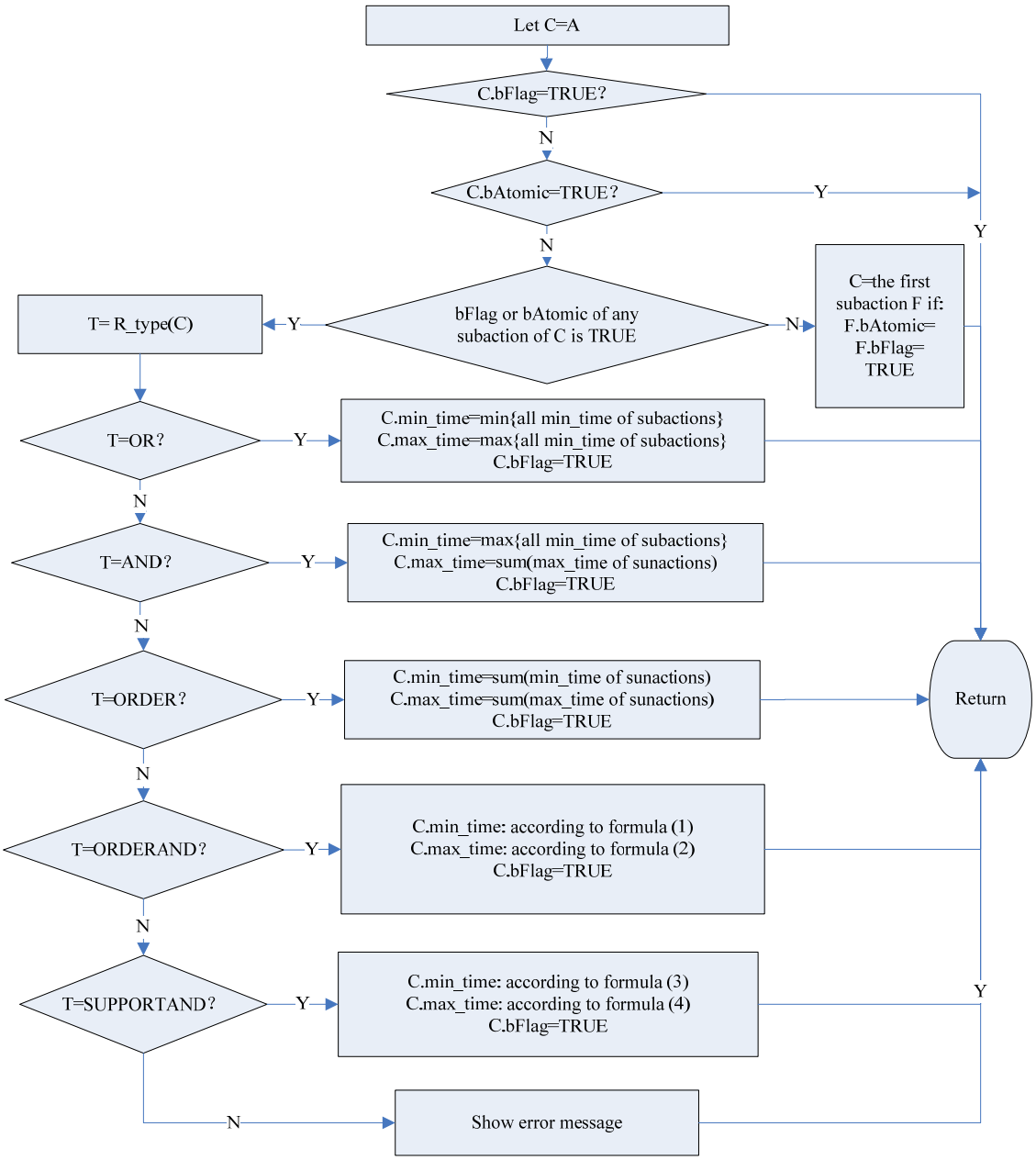

Figure 7. Quantitatively analyzing algorithm on the plan resolution domain

4.2. SUPPORTAND_support_num

Let k=SUPPORTAND_support_num: the number of supported subactions that has AND relation. The array subaction_ SUPPORTAND [SUPPORTAND_support_num] denotes these subactions. If the parent action of these subactions is P. The action s is defined in definition 5. Then P.min_time and P.max_time is as formula (3) and (4):

$$\text{P.min\_time} = \max\{\max\{\text{ subaction\_ SUPPORTAND [i].min\_time} \mid i=1,\ldots,k\}, \text{s.min\_time}\} \quad (3)$$

$$\text{P.max\_time} = \sum_{i=1}^{k} subaction\_SUPPORTAND[i].\max\_time + s.\max\_time \quad (4)$$

5. Conclusion

We use the plan decision making of responding botnet attack to describe the method of constructing represent graph. In decomposing action into subactions the relation type between subactions and the action layer are introduced. The store space size occupied by our represent graph is not increased exponentially when the action node added. We construct a quantitatively analyzing algorithm on the complex plan resolution domain. The OR relation between subactions enables the graphic model to represent different plan. The ORDER relation enables the graphic model to represent the order of actions. The SUPPORT and SUPPORTORDER relation type enable the graphic model to represent more complex actions.

Our graphic model of complex plan resolution domain is a hierarchy structure. We can display it with different level. If the graph is too larger to load into the memory, we should study the method to store our graph in external memory [11].

Every plan resolution represented by this graphic model is a partially ordered set P. Given resource constraint set how to determine P being a feasible or optimal resolution? This is the need for further research.

References

[1] Armin B, Thorsten H. *Tracking DDoS Attacks: Insights into the Business of Disrupting the Web*. Proceedings of the 5rd Intl. USENIX Workshop on Large-Scale Exloit and Emergent Threats. SAN JOSE. 2012; 1:101-108

[2] Bibai J, Saveant P, Schoenauer M, Vidal V. *An Evolutionary Metaheuristic Based on State Decomposition for Domain-Independent Satisfying Planning*. ICAPS-2010. Toronto. 2010; 18-25.

[3] Albert-László Barabási. The Network Takeover. *NATURE PHYSICS.* JANUARY 2012; 8: 14-16

[4] Mark H. *LAYERWIDTH: Analysis of a New Metric for Directed Acyclic Graphs.* Proceedings of the Ninetheenth Conference on Uncertainty in Artificial Intelligence, San Francisco. 2003; 321-328.

[5] Roberto B, Fabio G, Alberto L. *An Algebra of Hierarchical Graphs and its Application to Structural Encoding.* Scientific Annals of Computer Science. Iasi. 2010; 53-96

[6] Vitter J. Algorithms and Data Structures for External Memory. *Editors*. Series on Foundations and Trends in Theoretical Computer Science. Hanover: MA Press. *Also published of Foundations and Trends in Theoretical Computer Science*. 2008; 2(4)

[7] Alfonso G, Alessandro S, Ivan S. *Temporal Planning with Problems Requiring Concurrency through Action Graphs and Local Search.* ICAPS-2010. Toronto. 2010; 53-96.

[8] Ulrich Meyer, Vitaly Osipov. *Design and Implementation of a Practical I/O-efficient Shortest Paths Algorithm.* Proceedings of the 11th Intl. *Workshop on Algorithm Engineering and Experiments* (ALENEX). New York. 2009; 85-96.

[9] Pearce D, Kelly P. Dynamic Topological Sort Algorithm for Directed Acyclic Graphs. *ACM Journal of Experimental Algorithmics*, 2006; 11(Article No. 1.7): 1-24.

[10] Donald E Knuth. *The art of Computer Programming*. Beijing: China Machine Press; 2008; 3(2): 288-311.

[11] Moscovich T, Chevalier F, Henry N, Pietriga E, Fekete J. *Topology-Aware Navigation in Large Networks*. CHI 2009, April 9th. Boston. 2009; 2319-2328.