ALGORITHMIC EVALUATION OF TRUST IN MULTILEVEL MODEL

Jan Samek[1], **Frantisek Zboril jr.**[1]

[1]Brno University of Technology, Faculty of Information Technology,
61266 Brno, Bozetechova 2, Czech Republic

samejan@fit.vutbr.cz (Jan Samek)

Abstract

Artificial agents are autonomous entities that should behave rationally by their own control. They should have also some social abilities that enable them to make strategic decision inside a multi-agent community. One of such decision is to select proper partner which it believes that its behaviour will be reliable and trustworthy. We address area of trust which may be important for improving agent's reasoning capabilities and by this also rationality of its behaviour within an agent or multi-agent system. This paper introduces specification of the model which is used for representation of trusts and their hierarchy in agent's beliefs. In fact this is a part of agents's belief base and it is used when decision about some system element trustworthy takes part in agent's reasoning process. For these purposes, we develop Hierarchical Model of Trust in Context which is represented by multilevel graph, where each node of the graph represents different aspects (contexts) of trustee and each edge of the graph represents correlation between different aspects. This is useful to modelling trust in respect to context-aware environment.

Keywords: trust evaluation, multilevel graph, agent reasoning, context trust

Presenting Author's Biography

Jan Samek received an MSc degree in Electrical Engineering and Computer Science in Brno University of Technology. In this same school, he is a postgradual student on Faculty of Information Technology in Brno, Czech Republic. His main interest in computer science is artificial intelligence area, especially multiagent systems and trust and reputation principles modelling for virtual agent communities.

1 Introduction

Agent and multi-agent systems are used in many computer applications and create large-scale open distributed networks. Agents as autonomous entities are nowadays used also in small integrated devices such as sensor nodes in wireless sensor network (WSN). Wireless sensor network topology is usually highly dynamic and distributed with many heterogeneous nodes [2].

Our research aims to the improvement of multi-agents effectiveness. These systems are also suitable for making modelling tools for open distributed systems like WSN [1]. In our recent research [4, 5] we study *trust* social phenomena in connection with the multi-agent systems. We address area of trust which may be important for improving agent's reasoning capabilities and by this also rationality of its behaviour within an agent or multi-agent system. We construct agent's belief base from different aspects of their possible counterparts, these aspect are connected into multi-level graph and connections between different aspects represent their relationship.

This situation can be illustrated with an example where agent relies on another agent responsibility to perform some data sensing on the one hand and also on propagation of reputation of other elements on the other hand. Malicious nodes may pretend a good sensing element of the system, but they may send false information about other nodes' reputation trying to compromise them. These two aspects are quite independent but there can be some aspects that have some dependencies. For example node's abilities to establish radio connection with other nodes are basic assumption for good and trusted coordinator and also it is necessary for a responsible network router. Then there are three aspects for which one may establish a trust attitude. We then suppose that there is a *context* in which particular trust aspects are observed [3, 4].

Context is a set of lower-level trusts that consists the considered trust. Our research of possible models which would be sufficient for the above mentioned situations has brought as to development of a model based on multilevel weighted graph. Hierarchy structure is made as a multilayer graph where individual nodes represent particular belief about trusts in some contexts and the edges between two nodes represents dependences between two context aware trusts at two adjacent levels.

We have developed two models concerning trust in hierarchy [6], one of them straight with some exact values for every particular trust aspect, the second one introduces some intervals that represent uncertainty about the aspects' value. The second case is also the case which is matter of this text.

The rest of the paper is organized as follows. Section 2 describes HMTC model structure, section 3 describes computation on this model and provide trust evaluation algorithm. Experiments with the model by using trust evaluation algorithm is provided in section 4. Conclusion and future work will be presented in section 5.

2 Structure of HMTC

Hierarchical Model of Trust in Contexts (HMTC) [6] contains nodes that represent overall trusts to the element at the highest as well as the nodes at the lower levels, that are not so context aware and they represent more specific trust aspects. Atomic aspects, which are represented by the lowest nodes of the graph, stands for functionality of hardware modules or something that does not depend on other aspects where the trust is evaluated. Edges between two nodes represent impact of the lower-level node trust to the higher-level node trust.

2.1 HMTC definition

HMTC is a multilevel graph defined as usual as a set of nodes and a set of edges. Nodes represent *trust contexts* and edges represent correlation between the contexts. The multilevel property means that the set of nodes can be split into several disjunct subsets, where each subset corresponds to one level of the graph. From other point of view, each node has exactly defined its level. Leaf node set is defined as a set of nodes which are at lower level of graph. Every edge in the model graph connects pair of nodes at two adjoining levels.

2.2 Trust function

In HMTC we define *extended trust function*, which maps every node and a time moment from a time set to power set of trust interval [6]:

$$\rho : N \times T \rightarrow 2^{<0,1>} \tag{2.1}$$

Set of nodes is denoted by N, time moments set is denoted by T and possible trust interval $< 0, 1 >$ is from $\Re$. In fact, mapping of the trust function to power set of the possible trust interval means, that trust has some *minimum* and *maximum* trustworthiness from the possible trust interval. Toward this, it is better to define trust with their minimum and maximum parts $\rho^{min} : N \times T \rightarrow < 0, 1 >$ and $\rho^{max} : N \times T \rightarrow < 0, 1 >$. With using this parts, final trust function for the node A in time t can be denoted as:

$$\rho(A, t) =< \rho^{min}(A, t), \rho^{max}(A, t) > \tag{2.2}$$

where condition $\rho^{min}(A, t) \leq \rho^{max}(A, t)$ must be valid. Initial trust value for node A, when we have no idea about its trustworthy is $\rho^{min}(A, t) = 0$ and $\rho^{max}(A, t) = 1$.

2.3 Weight function

Weight function is used to express strength of relation between different nodes. Weight function maps each edge to the *weight interval*, which is interval $< 0, 1 >$ from the set of the real numbers $\Re$.

$$w : E \rightarrow < 0, 1 > \tag{2.3}$$

Set of edges in the model is denoted by E. For every non-leaf nodes, there is a restriction that weights sum of node's incoming edges (direction from lower level to higher level) is always equal to 1. In this paper we suppose that when the model is made, each node has predefined weight function and can't change in time.

3 Trust evaluation

Behaviour of the model is driven by events, which can be either primary or causal. Each node can be updated by an event. Primary events are abstract in our model, it means that we do not deal with causes of the event, setting of size of event impact etc., but we suppose that there may occur something that change the trust interval for at least one node in the model. Result of such event is that the targeted node's trust interval is updated as the intersection of the event interval with the original node's interval. But primary event may also cause modification of other nodes trust interval. At first, all the parent nodes from the actually impacted node are the possible candidates for trust updating. And consequently all the nodes that lie on the paths from the target node to some of the highest level nodes are also the candidates. But consequently also children of the target node may be changed and then children's parents are such candidates etc. In fact when one node changes it may cause several events for the adjacent nodes. In this section, we formally discuss how a set of possibly updated nodes is determined. Then we deal with principles of parent/children nodes updating.

3.1 Up-direction computation

Up-direction computation is used in a case, when parent node trust is computed based on knowledge about all the children nodes trusts. This computation is illustrated in Figure 1. A_1 node trust is given as a sum

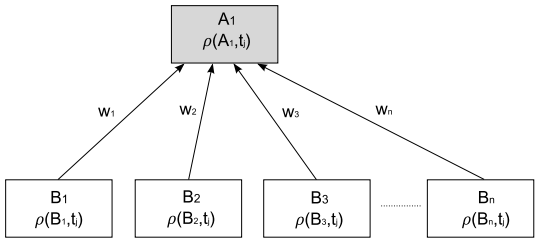

Fig. 1 Up-direction computation example

of multiplications of all its children nodes $(B_1 \dots B_n)$ trusts and their corresponding weights by the following formulas [6]:

$$\rho^{min}(A_1, t_{j+1}) = max(\rho^{min}(A_1, t_j), \sum_{i=1}^{n} w_i \rho^{min}(B_i, t_j)) \quad (3.1)$$

$$\rho^{max}(A_1, t_{j+1}) = min(\rho^{max}(A_1, t_j), \sum_{i=1}^{n} w_i \rho^{max}(B_i, t_j)) \quad (3.2)$$

$$\rho(A_1, t_{j+1}) =< \rho^{min}(A_1, t_{j+1}), \rho^{max}(A_1, t_{j+1}) > \quad (3.3)$$

By equations 3.1 and 3.2 for ρ^{min} and ρ^{max} may take the value outside the previous trust interval, therefore the final trust function ρ must be composed from ρ^{min} and ρ^{max} with using max and min functions. This will ensure normalization of ρ to the *extended trust interval*.

3.2 Down-direction computation

Down-direction computation is used in case, when child node trust is computed based on knowledge about parent(s) and neighbour(s) node(s) trust. Down-direction computation of B_1's trust is illustrated in Figure 2. Trust of node B_1 is computed trough parent node

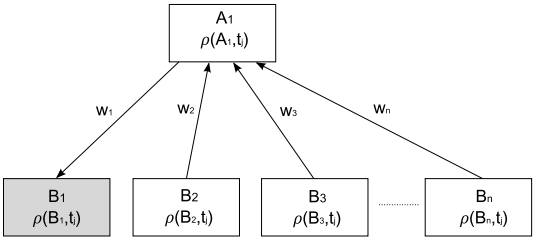

Fig. 2 Down-direction computation example

A_1 and all A_1 children $(B_2 \dots B_n)$ [6]. The following formulas demonstrate down-direction computation for node B_1:

$$\rho^{min}(B_1, t_{j+1}) = max(\rho^{min}(B_1, t_j), \frac{\rho^{min}(A_1, t_j) - \sum_{i=2}^{n} w_i \rho^{max}(B_i, t_j)}{w_1}) \quad (3.4)$$

$$\rho^{max}(B_1, t_{j+1}) = min(\rho^{max}(B_1, t_j), \frac{\rho^{max}(A_1, t_j) - \sum_{i=2}^{n} w_i \rho^{min}(B_i, t_j)}{w_1}) \quad (3.5)$$

$$\rho(B_1, t_{j+1}) =< \rho^{min}(B_1, t_{j+1}), \rho^{max}(B_1, t_{j+1}) > \quad (3.6)$$

In case when an impacted node has more than one parent, partial result trough all parent nodes are computed and aggregated with using intersection. This situation is illustrated on figure 3. B_2 is computed with using two

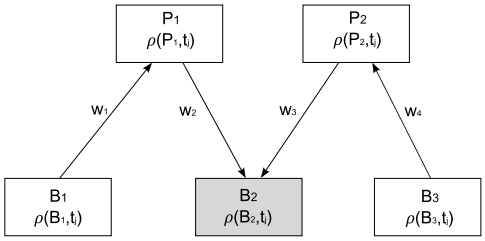

Fig. 3 Down-direction computation with two parent nodes

parts: $\rho^{P_1}(B_2, t_{j+1})$ and $\rho^{P_2}(B_2, t_{j+1})$. These parts are computed in respect to equations 3.4, 3.5 and 3.6 and combined into $\rho(B_2, t_{j+1})$ with using intersection:

$$\rho(B_2, t_{j+1}) = \rho^{P_1}(B_2, t_{j+1}) \cap \rho^{P_2}(B_2, t_{j+1}) \quad (3.7)$$

3.3 Trust evaluation algorithm

As we mentioned above, HMTC is driven by events where each node can be updated. In following text we recognize two types of updates: *direct update* and *subsidiary update*. Direct update is update which is done from environment and directly update trust interval of any node of HMTC. This update triggers process called *trust evaluation algorithm* which may consequently cause subsidiary updates. In next chapters we will use the term *update* for subsidiary update and direct update will be mentioned explicitly. This section specifies how a trust evaluation algorithm works.

Algorithm will be demonstrated on an example, when a node A in a time moment i directly updated into a trust interval $\rho(A,t_i) =< \rho^{min}(A,t_i), \rho^{max}(A,t_i) >$. Algorithm starts with adding every parent and child nodes of A into the *Open* queue. To use a queue ensures that the parent nodes are re-computed first in the next iteration and the children nodes are re-computed later. This operations order increases effectiveness of the trust update algorithm because it minimize number of iterations which goes to the final state. This optimization will be described in section 4. The algorithm works in iterations while the *Open* queue is not empty. For each `poped` node new trust interval is re-computed with using function `computeTrust`. This function accepts one argument *node*. If a *node* is a root node ($node \in N_1$), then only *up-direction* computation will progress. If this *node* is leaf-node ($node \in N_n$), only *down-direction* computation will progress. In all other cases, both up/down-direction computations will be done. Function `computeTrust` respects previous trust of the *node* and new trust interval (for time $j+1$) equals to intersection of previous (time $j-1$) and computed (time j) trust values.

$$\rho(A,t_{j+1}) = \rho(A,t_j) \cap \rho(A,t_{j-1}) \qquad (3.8)$$

If new *node* trust differs from its previous trust $\rho(A,t_{j+1}) \neq \rho(A,t_{j-1})$, then *node* trust is updated (subsidiary update) and all of *node*'s parents and children nodes are added into the *Open* queue (except nodes, which are already in the *Open* queue or except the node *A* itself).

Naturally one node may be re-computed and/or updated more than once during the algorithm run. This happens when it is added again to the *Open* queue after a previous re-computation of the some adjacent node. In next chapter, we describe how this situation may occur.

4 Experimental results

In this section, we describe case study for usage of our model. On this case study we provide example of trust update algorithm and we describe different strategies of node selection for the trust update algorithm. We also present experimental results that show how number of computations and number of updated nodes depends on position of particular nodes in the multilevel graph.

input: $updatedNode = A, time = i,$
$\rho(A,t_i) =< \rho^{min}(A,t_i), \rho^{max}(A,t_i) >$

```
1  begin
2      Open ← ∅
3      pushAll (Open, getParents (A))
4      pushAll (Open, getChilds (A))
5      while Open ≠ ∅ do
6          node ← pop (Open)
7          ρ(node, t_{i+1}) ← computeTrust (node)
8          if ρ(node, t_{i+1}) ≠ ρ(node, t_i) then
9              Parents ← ∅ ∪ getParents (node)
10             while Parents ≠ ∅ do
11                 p ← pop (Parents)
12                 if p ≠ A and p ∉ Open then
13                     push (Open, p)
14             Childs ← ∅ ∪ getChilds (node)
15             while Childs ≠ ∅ do
16                 ch ← pop (Childs)
17                 if ch ≠ A and ch ∉ Open then
18                     push (Open, ch)
19         i ← i + 1
20 end
```

Algorithm 1: Trust evaluation algorithm

4.1 Case study scenario

At the beginning, we describe our case study scenario. We provide two different examples of trust update algorithm in one HMTC. The model is created from situation where the peers in P2P network are rated in different contexts (client, server, link speed, etc.).

In the first case, we update trust in leaf node (figure 4) **Upload**. This update starts from an initial state, where each node trust interval is initialized into $< 0.0, 1.0 >$. Direct update of the **Upload** node (into interval $< 0.3, 0.5 >$) causes only up-direction re-computation (because **Upload** has no children nodes). This re-computation updates three graph nodes: **Server**, **Link speed** and **Peer**. This situation is illustrated in Figure 4.

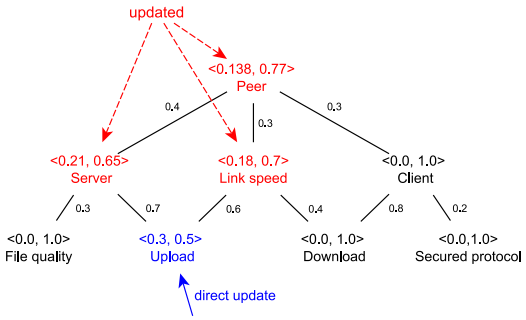

Fig. 4 Case study 1 – up-direction update

In the second case, our example starts from situation which follows the fist example after the evalua-

tion is done. Direct update of node **Client** (into interval $< 0.4, 0.6 >$) causes both up/down-direction re-computations. As a consequence of this re-computation, three other nodes are updated: **Peer**, **Link speed** and **Download** (this is illustrated on figure 5). **Download** node is re-computed and updated when parent node cause down-direction via parent nodes **Link speed** and **Client**. **Link speed** node is updated in both directions and for the **Peer node** only up-direction is used via nodes **Server**, **Link speed** and **Client**.

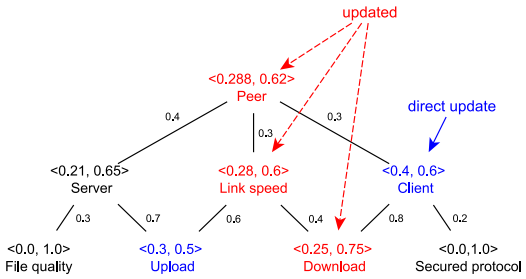

Fig. 5 Case study 1 – up/down-direction update

The **Peer** node is updated twice in this example. Firstly when the **Peer** node was added into the *Open* queue as first node (because its direct parent of updated node **Client**) and was updated from interval $< 0.138, 0.77 >$ into interval $< 0.258, 0.65 >$. Secondly the **Peer** node is updated when a **Link speed** node was updated from interval $< 0.18, 0.7 >$ to interval $< 0.28, 0.6 >$. Because its parent node is the **Peer** node and then will be added into the *Open* queue again. Finally, the **Peer** node trust is updated into interval $< 0.288, 0.62 >$ due the interval reduction of **Link speed** node.

4.2 Node selection strategies

We are investigating two different strategies in trust evaluation algorithm concerning how the nodes are inserted into the Open queue respectively selected from Open queue. Selection strategies may have impact to number of iterations until HMTC update process finishes. This does not mean that strategy selection reduce/increase number of iterations, but only reduce/increase number of iteration *when* all relevant nodes are updated.

First selection strategy is called as *first-the-child* (FtCh) strategy. This strategy ensures that children nodes of actually updated node will be inserted into the Open queue before the parent nodes and it means that they will be process before the parent nodes in future. This strategy can be presented on trust evaluation algorithm (Algorithm 1 - described in section 3) when lines 3 and 4 are swapped and block of code from line 9 to line 13 is moved after line 18.

Second selection strategy is called as *first-the-parents* (FtP). This strategy works as we illustrated in section 3.3, Algorithm 1. Parent nodes of directly updated node are re-recomputed first because they are inserted into the Open queue before the children nodes.

Tab. 1 Experimental results with different selection strategies

	Case 1		Case 2	
Strategy	*FtCh*	*FtP*	*FtCh*	*FtP*
Iterations	8	8	10	10
Iterations to achive final state	**4**	**3**	**8**	**6**
Updated nodes	3	3	4	4
Computed nodes	5	5	6	6

Experiments with these two strategies are shown in table 1. For these experiments was used case study from previous subsection 4.1. *Iterations* is number of re-computed and/or updated nodes during one evaluation run, *updated nodes* are nodes which are subsidiary updated and *computed nodes* are nodes which are re-computed but not updated.

As we can see that strategy *first-the-parents* (FtP) reduce number of iterations which are required for achieving the final state of one evaluation run. Both strategies have the same number of iterations which are needed for re-computation. Future work will be aimed to optimization of the trust update algorithm with using first-the-parents strategy. This should reduce number of iterations which are needed for re-computation of all relevant nodes.

4.3 Effect of the node level to trust updates

Our last experiment aims to investigate impact of the direct updates in different levels of the model relating to number of produced node updates. We again use HMTC from the case study scenario in subsection 4.1 where all the nodes are in their initial state at the beginning. The model contains three levels ($n = 3$), these nodes were split into three disjunctive subsets by their level:

1. root node: $L = 1$, $N_1 = \{Peer\}$;
2. non-leaf and non-root nodes: $1 < L < n$, $N_2 = \{Server, Link\ speed, Client\}$;
3. leaf-nodes: $L = n$, $N_3 = \{File\ quality, Upload, Download, Secured\ protocol\}$.

At the beginning each node is initialized to the default trust interval $< 0, 1 >$. For these three sets we simulate node trust direct updating event (for each node from each set) and we update their trust from 0.0 to 1.0 (respectively from $< 0.0, 0.0 >$ to $< 1.0, 1.0 >$ with respect to the interval representation). We do this with a step 0.05 and we observe how many nodes are consequently updated. For each nodes set and each simulation step we compute average value of nodes that had been updated.

Experimental results are presented as a graph in Figure 6. It shows some interesting aspects [6] of HMTC which follows from presented case study:

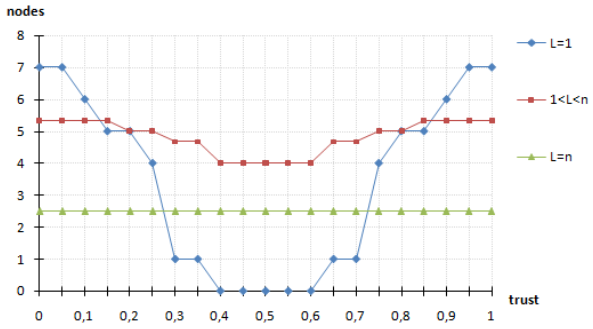

Fig. 6 Experimental results of updating nodes in different levels

1. Direct trust update in a leaf node always causes the same number of updated nodes irrespective to range of the update.
2. Updating of non-root and non-leaf node trust into values near extreme values (0 and 1) causes the greatest number of updated. The number of updates also depends on edge weights configuration.
3. In corresponding to point 2. above, we state that updating of the root node trust into values which are near to extreme values (0 and 1) causes updates of all the other nodes in the model irrespective to edge weights configuration.

4.4 Conclusion about experimental results

On the bases of the experiments presented in subsections 4.2 and 4.3 we observed that when leaf node is directly updated then only parent nodes (and all the parent nodes on the path to a highest-level node) are updated. Nodes in the same level of the model are not updated. This means that the *FtP* strategy is more effective than the *FtCh* strategy when direct updates of a node which is not in the highest level of the model are made. Both the strategies *FtP* and *FtCh* have the same effectiveness for a highest level node updates. Then we can conclude that effectiveness of the *FtP* strategy grows with the level of the node for which direct update is done.

5 Conclusion and future work

In this paper we presented the trust update algorithm for hierarchical model of trust in contexts (HMTC), which is made as a multilevel graph. The algorithm is used to infer trust from one aspect of entity for aspect another. This is useful for multiple context trust scenarios where different entities in the system should be seen as trusted in respect of an interaction context.

We also presented experiments on different scenarios which shows some interesting aspects of presented trust update algorithm. Future work will concentrate to usage of these experimental for improvement of trust update algorithm efficiency. This should consequently improve efficiency of our trust model.

Acknowledgement

This work was partially supported by the grants GACR 102/09/H042, BUT FIT-10-S-1 and the research plan MSM0021630528.

6 References

[1] Félix Gómez Mármol and Gregorio Martínez Pérez. Providing trust in wireless sensor networks using a bio-inspired technic. In *Networking and Electronic Commerce Research Conference (NAEC 08)*, 2008.

[2] Yingshu Li, My T. Thai, and Weili Wu. *Wireless Sensor Networks and Applications (Signals and Communication Technology)*. Springer-Verlag New York, Inc., 2007.

[3] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology, 2003.

[4] Jan Samek and Frantisek Zboril. Multiple context model for trust and reputation evaluating in multiagent systems. In *Proceedings of CSE 2008*, pages 336–343. Faculty of Electrical Engineering and Informatics, University of Technology Kosice, 2008.

[5] Jan Samek and Frantisek Zboril. Agent reasoning based on trust and reputation. In *Proceedings MATHMOD 09 Vienna - Full Papers CD Volume*, pages 538–544. ARGE Simulation News, 2009.

[6] Jan Samek and Frantisek Zboril. Hierarchical model of trust in contexts. In F. Zavoral et al., editor, *NDT 2010, Part II*, volume 88 of *CCIS*, pages 356–365, Berlin Heidelberg, 2010. Springer-Verlag.