

Social Network Models for the TDR System

YingJie Tang, HaoRan Zhang, ZhiJian Liang and Shi-Kuo Chang
Department of Computer Science

University of Pittsburgh, Pittsburgh, PA 15260, USA
yit20@pitt.edu, colinzhang@cs.pitt.edu, zhijianliang@163.com, chang@cs.pitt.edu

Abstract—The TDR system is an experimental multi-level slow intelligence system for personal health care. The TDR system can be used by a single user or a group of users who will interact to understand, maintain and improve each other's state of health. In this paper, we simulate social networks by applying the Abstract Machine model. Two social network models are described: the Circulated Model and the Teacher Student Model. We have incorporated the Teacher Student Model into the Chi super-component of the TDR system and obtained positive experimental results.

Keywords— *Personal health care system, slow intelligence system, social network, abstract machine model.*

1. Introduction

An experimental multi-level slow intelligence system for personal health care, called the TDR system, was developed as a test bed for exploring and integrating different applications in personal health care, emergency management and social networking [1]. The TDR system mainly consists of three super-components: Tian, Di and Ren. According to the Chinese philosophy these three super-components are the essential ingredients of a human-centric psycho-physical system. They can be thought of as human beings (Ren) interacting with the environment consisting of heaven (Tian) and earth (Di). For personal health care, there is a fourth higher level super-component called Chi (or Qi), which in this context represents the state of health of a person (or persons).

Decision making in TDR system is through multiple computation cycles involving the super-components to increase the chances of survival and well being of human beings (or groups). Any action based on only one aspect of the environment without considering the other aspects could reduce the chances of survival, thus iterative, multiple computation cycles are crucial for the TDR system.

The TDR system can be used by a single user, or a group of users who will interact to understand, maintain and improve each other's state of health. In this paper, we investigate the social network models for the TDR system.

The paper is organized as follows. Section 2 presents the GUI interface of the TDR system so that the reader can understand how the TDR system works in practice. In Section 3 an Abstract Machine model for the computation cycles is

presented. In Sections 4 and 5 we show how and what part of the Abstract Machine model is extended to incorporate the social network model into interaction models. Based upon this approach, we describe two social network models: The Circulated Model (Section 6) and the Teacher Student Model (Section 7). We have incorporated the Teacher Student Model into Chi component in the TDR system and obtained positive experimental results (Section 8). In Section 9 we discuss the implications and further research.

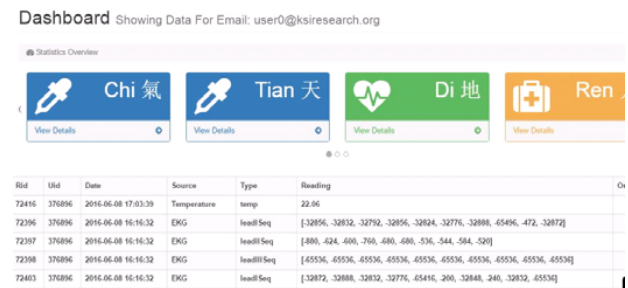
2. The Web GUI of the TDR System

In this section, we describe the Web GUI of the TDR system so that the reader can understand how the system works in practice.

The dashboard is the main GUI interface of the TDR system. As illustrated by Figure 1 it provides a high-level overview of the data in the system. On the left side, it has a menu panel that contains all the actions the user can perform, including activating and deactivating super-components. For the admin user, this menu will also include addition, deletion and modification of regular users.



Figure 1. The dashboard for Web GUI of TDR system.

The image shows a carousel interface for the TDR system. It is titled 'Dashboard Showing Data For Email: user0@ksiresearch.org'. The carousel displays four cards: 'Chi 氣' (blue), 'Tian 天' (blue), 'Di 地' (green), and 'Ren' (orange). Below the carousel is a table with columns: 'Rid', 'UId', 'Date', 'Source', 'Type', and 'Reading'.

Rid	UId	Date	Source	Type	Reading
72416	376896	2016-06-08 17:03:39	Temperature	temp	22.86
72396	376896	2016-06-08 16:16:32	EKG	leadII Seq	[32856, 32832, 32752, 32856, 32824, 32776, 32888, 45496, 472, 32872]
72397	376896	2016-06-08 16:16:32	EKG	leadIII Seq	[480, 424, 400, 760, 480, 480, 536, 544, 584, 520]
72398	376896	2016-06-08 16:16:32	EKG	leadIII Seq	[45536, 45536, 45536, 45536, 45536, 45536, 45536, 45536, 45536, 45536]
72403	376896	2016-06-08 16:16:32	EKG	leadII Seq	[32872, 32888, 32832, 32776, 45416, 200, 32848, 240, 32832, 45536]

Figure 2. The Carousel.

There is a carousel that displays all super-components in rotation, four at a time for the PC screen and only one for the smart phone screen. This vividly demonstrates the idea of computation cycles in the TDR system. A component's banner is in *tranquil* state (blue or green color) until an alert is received and then it changes to *elevated* state (orange or red color).

As shown in Figure 2, when the user clicks on the “**View Details**” button at the lower right part of the dashboard, a table will appear beneath the carousel panel to display all records that belong to the current user. For each entry, it contains the date and time of a record, the sensor type, the data type, the actual reading of the data, and the originator. This scheme allows flexibility and scalability, as in the future there might be more and more sensors added to the TDR system. In Figure 2, the first record is the room temperature from the Earth (Di) super-component, and the other records are the EKG recordings from the Human (Ren) super-component.

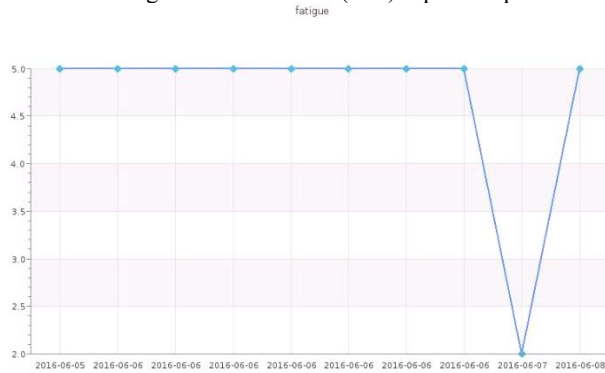


Figure 3. Visualization of fatigue in time.

If the user is communicating remotely with his/her doctor, a user might want to specify the record ID so that the doctor knows exactly what entry he/she is referring to. As shown in Figure 3, by clicking on the “**Draw Graphs**” button a graph showing the data-to-day changes of a selected data item can also be displayed by the GUI for visualization by the user or the doctor.

By clicking on the “**Analyze Data**” button at the lower right part of the dashboard, a user's records will be analyzed to evaluate his/her state of health such as the total-Chi of a person. Clicking on the “**Find Similar**” button will retrieve other user's records similar to the current user's record so that the user or the doctor can make a comparison to enhance his/her understanding. The details on data analysis and similarity retrieval are explained in the Appendix.

Interestingly, the user in a social group can use the same TDR GUI to retrieve other user's records, analyze them to evaluate their state of health, etc. If a particular user is a teacher or a 'master' with deep knowledge, this user can influence others to understand, maintain and improve one's state of health.

Therefore the TDR system can also be used by social groups for collective personal health care.

3. The Abstract Machine Model for Computation Cycles

A slow intelligence system SIS typically possesses at least two decision cycles [2]. The first one, the *quick decision cycle*, provides an instantaneous response to environmental changes. The second one, the *slow decision cycle*, tries to follow the gradual changes in the environment and analyze the information acquired from the environments or peers or past experiences. The slow/quick decision cycles enable the SIS to both cope with the environment and meet long-term goals.

To model such decision cycles we introduce an Abstract Machine model of multiple computation cycles in Section 3.1, and then specify the computation cycles for the TDR system in Section 3.2. In Section 3.3 we describe multi-level computation cycles and how to apply the abstract machine model to social networks.

3.1. The Abstract Machine Model

The Abstract Machine Model is specified by: $(P, S, P_0, \text{Cycle}^1, \dots, \text{Cycle}^n)$, where

- P is the non-empty problem set,
- S is the non-empty solution set, which is a subset of P_0 ,
- P_0 is the initial problem set, which is a subset of P,
- $\text{Cycle}^1, \dots, \text{Cycle}^n$ are the computation cycles.

Each computation cycle will start from an initial problem set and apply different operators such as $+adap_{Aij}$, $-enum$, $>elim$, $=prop_{Aij}$ + and $>conc$ successively to generate new problem sets from old problem sets until a non-empty solution set is found. If a non-empty solution set is found, the cycle is completed and later the same computation cycle can be repeated. If on the other hand no solution set is found, a different computation cycle is entered.

As an example the problem set P consists of problem elements $p_1, p_2, p_3, \dots, p^n$, and each problem element p_j is specified by a vector consisting of attributes A_{ij} . A computation cycle x will attempt to find a solution set by first adapting based upon input from the environment: $P^x_0 +adap_{Aij} = P^x_1$ that is to adapt based on attribute A_{ij} , for example, by appending A_{ij} to each element in P^x_0 to form P^x_1 .

Then it may try to find related problem elements: $P^x_1 -enum < P^x_2$ where $P^x_2 = \{y: y \text{ is related to some } x \text{ in } P^x_1, \text{ e.g. } d(x,y) < D\}$. Next it may try to eliminate the non-solution elements: $P^x_2 >elim - P^x_3$ where $P^x_3 = \{x: x \text{ is in } P^x_2 \text{ and } x \text{ is in } S\}$.

Finally the solution elements (or alert messages if there are no solutions) may be propagated to peers: $P^x_3 =prop_{Aij} + P^x_4$ that is to export/propagate attribute A_{ij} to peers.

Therefore this computation cycle can be specified succinctly as follows: Cycle^x [guard x,y]: P^x0 +*adap*_{Aij}= P^x1 -*enum*< P^x2 >*elim*- P^x3 =*prop*_{Aij}+ P^x4

The above expression is a specification of the computation cycle, not a mathematical equation. This expression should be read and interpreted from left to right.

If P^x4 is non-empty, the Abstract Machine will complete this cycle of computation and terminate at the end of Cycle^x, and it may later resume at the beginning of Cycle^x. Otherwise P^x4 is empty and the Abstract Machine will jump to a different Cycle^y. This is specified by [guard x,y] where x is the current computation cycle if a solution set is found (P^x4 is non-empty), and y is the computation cycle to enter if no solution set is found (P^x4 is empty). Before an Abstract Machine completes its current computation cycle, it will propagate the solution set (or alert messages) to its peers.

In the above, the elimination operator can be replaced by the concentration operator, whenever the solution set is not known a priori. The concentration operator applies a predefined threshold to filter out problem elements below the threshold: P^x1 >*conc*= P^x2 where P^x2 = {x: x is in P^x1 and th(x) above a predefined threshold t}.

3.2. Multiple Computation Cycles of TDR System

For the TDR system, a problem element is a combination of Tian, Di and Ren attributes. Those problem elements that are favorable for human survival are in the solution set S. The problem set P consists of problem elements p1, p2, p3, ..., pⁿ, and each problem element is specified by a vector consisting of the attributes from Tian (heaven), Di (earth) and Ren (human being), i.e.,

$$p^j = (t1j, t2j, \dots, d1j, d2j, \dots, r1j, r2j, \dots)$$

For example, the Tian attributes tij are atmospheric variables such as amount of sunlight and water level, the Di attributes dij are residential variables such as ambient temperature and humidity, and the Ren attributes rij are personal health indicators such as blood pressure, EKG reading, heart rate, etc.

$$p^j = (\text{sunlight}_j, \text{waterlevel}_j, \text{temp}_j, \text{humidity}_j, \text{bloodpressure}_j, \text{spo2value}_j, \text{heartrate}_j)$$

Initially some attributes may not be assigned any value and some may already have pre-assigned values. After most attributes have been assigned values one can decide whether the problem element is in the solution set. (The simplest case is that each attribute Aij has a solution range Rj, and if every attribute Aij falls within the solution range Rj then the problem element pj is in the solution set S).

In the TDR system, there are continuous interactions among the three super-components Tian, Di and Ren. Each super-component has its own computation cycle, which is basically

the following: Starting from some problem set P0, the super-component first adapts to the input from the environment as well as from other peer super-components. It then tries to find related problem elements by enumeration. After those problem elements not in the solution set have been eliminated either using the elimination operator or using the concentration operator, the termination condition can be tested. The termination condition is expressed by [guard x, y] where Cycle x is the current cycle and Cycle y is the cycle to jump to. Whenever one super-component completes its computation cycle, if a solution is found the computation ends, otherwise the control is transferred to the next super-component. Since there are three super-components, we will have three computation cycles.

The Tian super-component has computation Cycle1:

$$\text{Cycle1 [guard1,2]: } P^1_0 + \textit{adap}_{Aij} = P^1_1 - \textit{enum} < P^1_2 > \textit{elim} - P^1_3 = \textit{prop}_{Aij} + P^1_4$$

Likewise, the Di super-component has computation Cycle2:

$$\text{Cycle2 [guard2,3]: } P^2_0 + \textit{adap}_{Aij} = P^2_1 - \textit{enum} < P^2_2 > \textit{elim} - P^2_3 = \textit{prop}_{Aij} + P^2_4$$

Finally, the Ren super-component has computation Cycle3:

$$\text{Cycle3 [guard3,1]: } P^3_0 + \textit{adap}_{Aij} = P^3_1 - \textit{enum} < P^3_2 > \textit{elim} - P^3_3 = \textit{prop}_{Aij} + P^3_4$$

Notice the three computation cycles together form a higher-level computation cycle. High-level computation cycles are essential for a complex human-centric psycho-physical system such as the TDR system.

3.3. Multi-Level Computation Cycles

The three computation cycles together form a higher-level computation cycle. High-level computation cycles are essential for a complex human-centric psycho-physical system such as the TDR system. In the above specification, we can replace [guard1,2], [guard2,3] and [guard3,1] by [guardX,2], [guardX,3] and [guardX,1], respectively.

When computations in Cycle1, Cycle2 or Cycle3 is unsuccessful and solution set is empty, control is transferred to the next Cycle in cyclic order, i.e. first Cycle1, then Cycle2, then Cycle3 and then returning to Cycle1. On the other hand, when computations in Cycle1, Cycle2 or Cycle3 are successful and solution set is non-empty, control is transferred to the Chi super-component computation CycleX at the next higher level. CycleX is specified as follows:

$$\text{CycleX [guardX,1]: } P^X_0 + \textit{adap}_{Aij} = P^X_1 - \textit{enum} < P^X_2 > \textit{elim} - P^X_3 = \textit{prop}_{Aij} + P^X_4$$

In the above, the +*adap*_{Aij}= may be the input propagated from the lower-level super-components, or from the super-components of other human observers (see below).

If computation in CycleX is unsuccessful, control is returned to the Tian computation Cycle1 (or the Di computation Cycle2, the Ren computation Cycle3, respectively). If computation in

CycleX is successful, then the computation terminates in CycleX and the Dashboard will display the results, i.e., the estimated total-Chi values.

The subjective evaluations can be entered by the principal user himself/herself based upon his/her subjective feelings. For example if he/she feels “sweaty at night”, he/she will enter a value close to 10 (on a scale of 1 to 10) for the “sweaty-at-night” attribute for Chi.

It is also possible to formulate CycleX so that other human observers who are “friends” of the principal user can fill in the subjective Chi attributes. This social network of human observers can also vote on updating the Chi attributes for the principal user. These human observers may even be allowed to fill in the objective Chi attributes as if they were sensors. Thus this TDR system is an iterative slow intelligence system, or what we call the *Sentient Net*.

4. Social Network Modeling

A social network can be modeled by a graph $G = (P, A)$ where persons p_i in the social network are modeled by nodes P and their relations are modeled by arcs A . A node can be attributed. For example, an attribute value 0.7 could mean a 0.7 probability of propagating its influence. An arc can also be attributed. For example, an attribute value 5 could mean 5 units of interaction between two persons p_i and p_k represented as nodes.

A person p_i is I-related to p_k if there is an arc between p_i and p_j and there are at least I units of interaction between them. A person p_i is (I,D)-related to p_k if there exists an I-related path from p_i to p_j and $d(x,y) \leq D$ is the shortest distance or minimum path length.

The slow intelligence operators are as follows:

Operator $enum_D$: $P_1 -enum_D P_2$ where $P_2 = \{y: y \text{ is } (I,D)\text{-related-to some } x \text{ in } P_1, \text{ e.g. } d(x,y) \leq D\}$

Operator $elim$: $P_1 >elim P_2$ where $P_2 = \{x: x \text{ is in } P_1 \text{ and } x \text{ is in } S\}$

Operator $conc_t$: $P_1 >conc_t P_2$ where $P_2 = \{x: x \text{ is in } P_1 \text{ and } th(x) \geq t \text{ where } t \text{ is a predefined threshold}\}$

Operator $adap_q$: $p_j +_{A_{ij}} adap_q p_k$ is to adjust p_j based on input attribute A_{ij} , for example, by appending A_{ij} to p_j with probability q , and not appending A_{ij} with probability $1-q$.

Operator $prop_q$: $p_j =_{prop_{A_{ij}}} p_k$ is to output/propagate attribute A_{ij} to p_k with probability q .

We focus on the enumeration operator and with inputs of arguments, the output will be a system with super-components where enumeration will be performed in each cycle. In order to better formulate the problem, the following assumptions are made:

Assumption 1: All the elements in the problem set are connected to each other. That is, for every pair of elements x and y , $d(x, y) < D$ is true.

Assumption 2: For every cycle in the system, each element having a probability less than 1 may be explored. In other words, not all the elements satisfying the condition $d(x, y) < D$ will be explored.

Assumption 3: The solution set is achieved when the problem set is *stable* at cycle t where cycle i and cycle j in the predefined range cycle $t-k$ to cycle k satisfies the following condition: the difference between every attribute of a certain element in cycle i and the attribute in cycle j is smaller than a predefined threshold d_{ij} .

The first assumption simplifies the distance computation between elements. The second assumption restricts ourselves to real social networks. The third assumption is the definition for a stable solution for computation cycles.

We are formulating a social network by observing the influence of opinions. Thus picking a reasonable solution set is important. Opinions tend to be more steady when the profiles are fully interacted.

Our elements are P_1, P_2, \dots, P_n , where each P_i represents a person in a group. Each person is defined by their [id, opinion, influencePercentage]. The id is a unique identifier for that person, and the opinion is the subject that would change during the computational cycles. Finally, the influence percentage is the indicator of how influential is that person (i.e. how capable is that person in changing others' opinions).

After knowing the nature of our elements, now we describe the problem set that contains those elements. Each problem set is divided into two sub sets: *High Influence Group*, and *Low Influence Group*.

The interaction happens in a sub-set basis. That is, H is interacting with L and vice versa. As expected, the result of this interaction is a potential change in the opinion of the element having interacted with the group. This change might happen, or it might not. It is a result of the average influence of all the elements in the interacting group.

Assuming that a change happened, there are two possible scenarios: the change in the opinion is either *influential* or *mind changing*.

The influential model is changing the element having interacted with group opinion based on the opinion of the

majority of the interacting group. For example, if the majority of the interacting group has the opinion X, then the change in element of the interacted group would be to the opinion X with some probability.

The other model causes the interacted group to change its opinion to the other one (in case of binary domain of opinions) regardless of what is the opinion of the interacting group.

Now we present the formal definitions of the model. From the abstract machine, we have the definition of the enumerator operator:

$P1 -enum < P2$ where $P2 = \{ y: y \text{ is related-to some } x \text{ in } P1, \text{ e.g. } d(x,y) < D\}$

This definition is extended in our model to the following:

$P1 -enum < P2$ where $P2 = \{ y: y \text{ is } x \text{ in } P1, \text{ with a chance that its opinion is changed}\}$

Each P_i is represented as a vector:

$P_i = [id, opinion, influence]$, where id is a unique integer, $opinion$ is a binary variable that represent an opinion, and $influence$ is a percentage.

The set of elements is divided into two subsets (Figure 1):

$H = \{P1, P2, \dots, P_i\}$

$L = \{P_j, \dots, P_n\}$

Where $H \cup L = P$

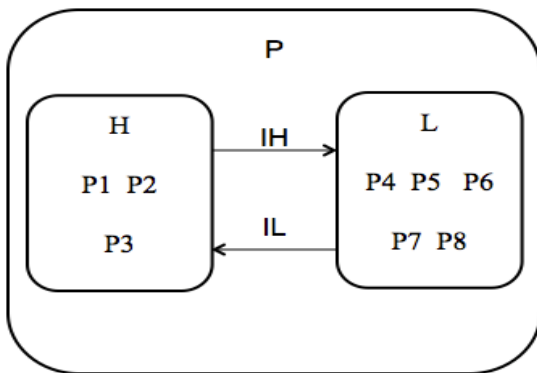


Figure 4. An example of a problem set with 8 elements.

During each interaction cycle, the interaction between the groups is represented by IH and IL , where IH is the percentage with which the group H affects the L group. The opposite holds for IL (see Figure 4). The probability with which an interaction will change an opinion of an element in the other group is calculated as follows:

$$IH = \frac{\sum_{i=1}^{|H|} P_i \text{ influence}}{|H|}$$

$$IL = \frac{\sum_{i=1}^{|L|} P_i \text{ influence}}{|L|}$$

Thus, each interaction cycle is a two-way interaction. From H to L ($H \rightarrow L$) and from L to H ($L \rightarrow H$). However, the percentage of changing the opinion of elements is different in each interaction.

5. The Interaction Models

When the the people element P_i is to have its opinion changed, there are two ways to change as mentioned above and discussed in [3]:

The Influencing Model: In this mode, if the majority of the interacting group has the opinion X , then the element that is interacted with will have the opinion changed to X (even if it is already has X as its opinion).

The Mind Changing Model: In this model, the opinion of the interacting group is irrelevant. Hence, if the element is going to change its opinion, it will change it to an opinion that is (not its current one) (e.g. flipping the opinion in binary domain opinions).

Some scenarios will now be presented. A video demonstration is available at: <http://screencast.com/t/n48teU5c>

We first present the influential model with two examples:

- 1) **Example 1:** $|H| = 20, |L| = 180$
 H group influencing percentage = [7%, 10%]
 L group influencing percentage = [1%, 3%]
 5% in H have their opinion for '0'
 95% in L have their opinion for '0'
 Number of interactions is: 30 (bidirectional)
 (See Figure 5).

- 3) **Example 2:** $|H| = 100, |L| = 100$
 H group influencing percentage = [80%, 90%]
 L group influencing percentage = [10%, 20%]
 5% in H have their opinion for '0'
 95% in L have their opinion for '0'
 Number of interactions is: 30 (bidirectional)
 (See Figure 6).

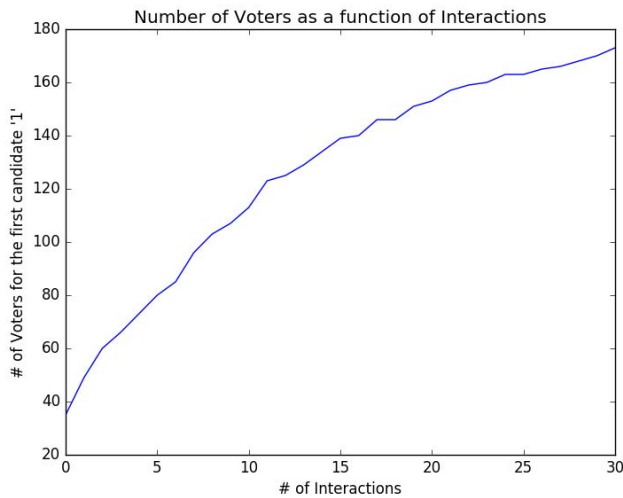


Figure 5. # of voters over # of interactions for Example 1.

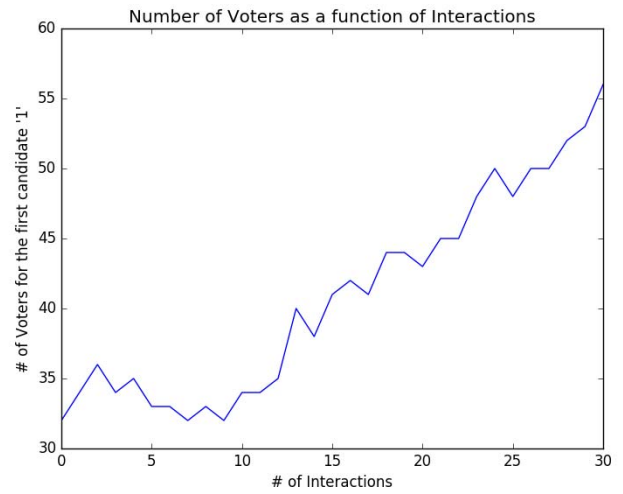


Figure 7. # of voters over # of interactions for Example 3.

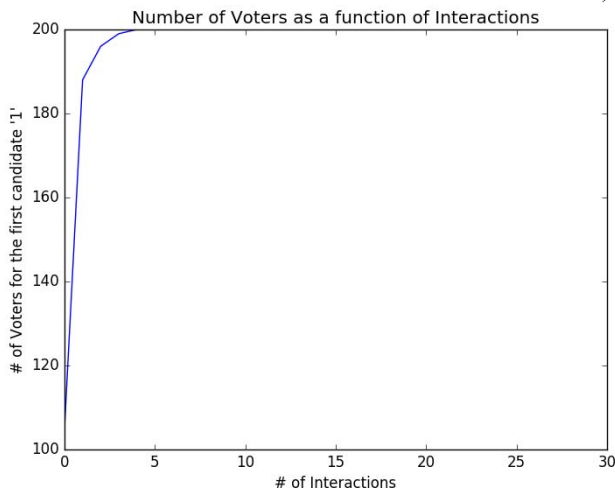


Figure 6. # of voters over # of interactions for Example 2.

From the above two examples, we see that the influence percentage controls the speed of convergence towards the “unified opinion”.

Next we will show the behavior of the Mind Changing Model with the same two examples using the same parameters:

Example 3: $|H| = 20, |L| = 180$
 H group influencing percentage = [7%, 10%]
 L group influencing percentage = [1%, 3%]
 5% in H have their opinion for ‘0’
 95% in L have their opinion for ‘0’
 Number of interactions is: 30 (bidirectional)
 (see Figure 7)

1) **Example 4:** $|H| = 100, |L| = 100$
 H group influencing percentage = [80%, 90%]
 L group influencing percentage = [10%, 20%]
 5% in H have their opinion for ‘0’
 95% in L have their opinion for ‘0’
 Number of interactions is: 30 (bidirectional)
 (See Figure 8).

The interesting observation is that the behavior is not predictable. It changes due to people changing their minds, hence the fluctuation. The codes are available at this URL:

<https://dl.dropboxusercontent.com/u/19443460/code.z>

ip

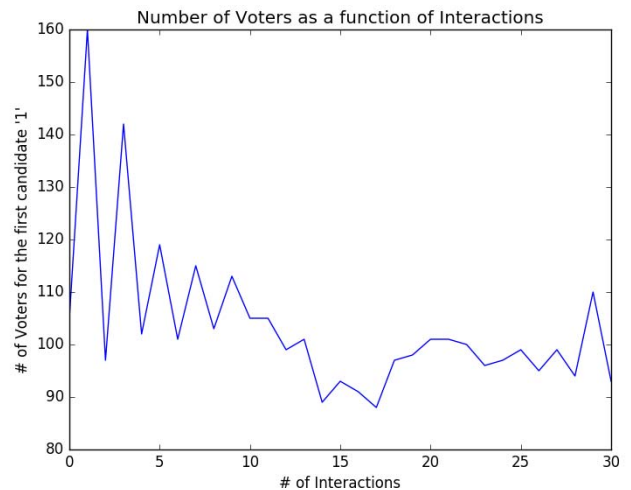


Figure 8. # of voters over # of interactions for Example 4.

6. Circulated Model

The Circulated model aims to weaken the Assumption 1 in the previous assumption in the social network model. The Assumption 1 made the social network model simpler by assuming that all nodes in the problem set are connected. However, this is nearly impossible in real social network models. Distance matrix is one of the most important features in the model and in this section, we introduce a distance matrix in our model to weaken the previous assumption.

In a circulated Model, communication is possible among all nodes albeit with different distance matrix. Adjacent nodes are connected with distance equal to 1 and the communication is bilateral, i.e. every node can influence its adjacent node with distance equal to 1. The influence is transitive, i.e. node1 can only influence node3 by passing the influence to node2 if node1 and node3 are not adjacent. A Circulated Model with n nodes is called Circulate-n model and a Circulate-5 model is shown in Figure 9.

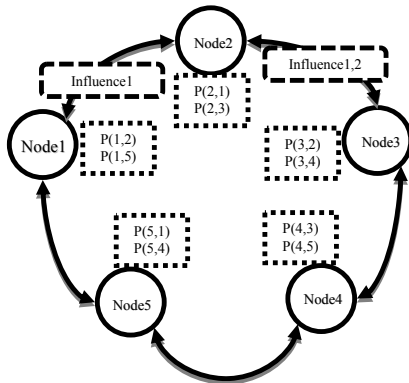


Figure 9. Circulate-5 model.

Node1 initiates an opinion influence and passes the influence to Node2, and Node2 passes the influence to Node3 by adding its' own views. In this case, Node1 influences Node3 by Node2 and it is one of the two ways that Node1 can influence Node3 whereas another ways is through:

Node1 -> Node5 -> Node4 -> Node3

Compared to real life social network models like Facebook or Twitter, the Circulated Social Network Model simplifies the relations of nodes and assumes that each node only interacts with its neighbors and the interaction to other nodes in the same group can be accomplished by interacting with its neighbors.

In this case, each node should store the probability of interacting with its neighbors. The distance matrix of each pair of nodes can be derived in the Circulate Social Network Model by the minimum distance of each path. In the above example, the distance between Node1 and Node3 is:

$$\begin{aligned} \text{dist}(\text{Node1}, \text{Node3}) &= \min\{d(\text{Node1}, \text{Node2}, \text{Node3}), \\ &d(\text{Node1}, \text{Node5}, \text{Node4}, \text{Node3})\} = \min\{2, 3\} = 2 \end{aligned}$$

7. Teacher Student Model

Figure 10 illustrates not only the teacher student model but also how the TDR system actually works in social networking. Obviously the teacher student model fits the TDR system very well. In Figure 10 as well as the following scenario, message types are indicated by M9, M10, etc.

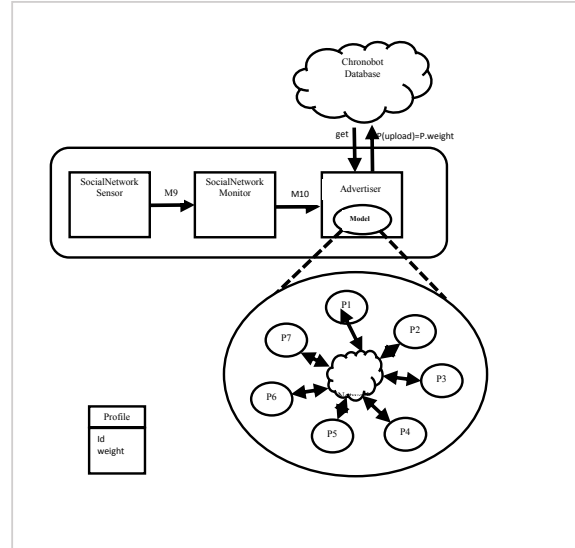


Figure 10. Teacher Student Model.

A scenario on how social network works in TDR system is as follows:

1. The SocialNetwork Sensor gathers information of Chi attributes (tongue, fatigue, weakBreath, pulse, sweaty) as well as the id and originator and send it to SocialNetwork Monitor through message M9.
2. SocialNetwork Monitor receives message M9 and calculates the total-Chi attribute using the same algorithm as from Chi attribute.
3. SocialNetwork Monitor sends the 6 attributes as message M10 to SocialNetwork Advertiser.
4. SocialNetwork receives message M10 and passes it to the SocialNetwork Model.
5. SocialNetwork Model gets the most recent value of the total-Chi attribute from Chronobot Database and compares it with the total-Chi from message M10 to evaluate the correctness of this record.
6. The SocialNetwork model adjusts the weights of each profile in the model and uploads the Chi attributes to Chronobot Database.

A profile in the group with higher weight is more reliable in judging others and we call this kind of profile *teacher profiles*. In the group model, there is one or more profile which is

called teacher profile who has a higher weight while the other profiles have lower weights. In order to propagate the weights to the Advisor component that uploads data to the Chronobot Database, we transfer the weight of each profile to the probability of uploading $P(\text{upload}) = P.\text{weight}$. In this case, whether a record of evaluations from a specific profile should be uploaded or not is associated with its weight attribute. The evaluation from a teacher is most likely to be uploaded to the database while an evaluation from a student is less likely to be uploaded. And an opinion from a student whose weight is small is less likely to be trusted and we upload this judgment to the database with a smaller probability. And thus all the records uploaded to the Chronobot Database are reliable. In this way, we transfer the weight of the profile to the trustworthiness of a judgment from a specific profile.

After a few computation cycles, students can learn the skills of n from the teacher and their weights will be increased so that they can produce more reliable evaluations. And it is the same as in the teacher-student relationship in the real life. As students learn from the teacher, the students will have gain the same power as the teacher. Once we make an assumption that the teacher's evaluations are always correct which means the teacher acts as an Oracle, the whole system will converge to a self-learning system and every profile in the system will be knowledgeable about the evaluation. And as the system improves, the evaluations will become more objective.

One important assumption we made in this model is that we restrict the number of profiles to be a small number, usually less than 10. A person is not likely to have a large number of friends that he can trust to evaluate his state of health. Since a small size group is easier to manipulate, this assumption makes the Teacher Student Model more robust.

8. Experimental Results for Teacher Student Model

In order to prove that the Teacher Student Model will converge to a stable status, we set up the following offline experiment. There are 7 members in the group including a teacher while the rest of the members are students. Teacher owns the weight of 1 which means every evaluation originated by the teacher is for sure reliable. On the other hand, students owns weights = 0.3 which means 3 in 10 evaluations originated by a student are correct and the rest are false. And with the hops forward, students can learn from the teacher and to gain the skill of judging correctly. If a student generates an evaluation that matches the teacher's evaluation, we increase his/her weights by a small value. And if he/she made a mistake, we decrease his/her weights in making evaluation. Thus only the weight changes in students worth tracing because the teacher is assumed to always generate correct evaluations. In other words in this Teacher Student Model, teacher acts as an Oracle and all the evaluations by the teacher are assumed to be correct.

Before starting the experiment, we create an *evaluations pool* where all the evaluations come from the teacher. The evaluations originated by students will be compared with those in the evaluations pool.

The experimental results show that as the hops (cycles) go on, the weights of each student increase steadily and all converge to 1 at the 14th hop. It successfully simulates the learning steps in real life. As the students learn from the teacher, they will gain the ability to evaluate state of health that the teacher has.

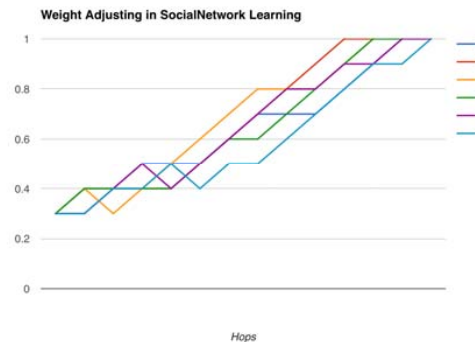


Figure 11. Experimental results.

In order to show how the learning converges, we did the following experiments: Three users, user0, user2 and user4 are incorporated in this experiment. User0 acts as the master in the group which means every made by user0 is set to be the standard. User2 act as a student and he should learn from the master how to make evaluations. Once user2 made an evaluation which is similar to the master, the reliability of user2 should be increased. Weight of user2 is the measurement of his/her reliability. Theoretically, as time goes by and user2 keeps on learning from the master, the weight of user2 should converge to 100 which is the same as the master. The last user incorporated in the experiment is user4, who is a patient. Both users0 and user2 make evaluations about user4. If the evaluation from user2 is similar to that of user0, we expect to see that the weight of user2 increases. The initial weight of user2 is set to 73. The master first made evaluations about user4 and the scores are (5, 5, 5, 5, 5), and the student later made an evaluation (5, 5, 5, 4, 5) about user4. We checked back the weight of user2, it increased to 74. This showed the ability of convergence of the Social Network Model in TDR system.

9. Discussion

One of our main goals is to expand the TDR system for the computation of Chi (also spelled as Qi in Chinese transliteration system HanYu PinYin). The Chi super-component is regarded as at a higher level. It has attributes including both objective measurements and subjective

evaluations [4]. This makes the Chi super-component both pro-active and adaptive at multiple levels.

The dashboard for TDR system can be further refined. When user clicks on “view details” for the Chi super-component, a list of attributes for Chi is shown. The objective measurements in this list is filled by the multi-level computation cycles based upon actual measurements. The subjective evaluations are entered by the principal user himself/herself based upon his/her subjective feelings, or his/her friends, teachers and a master with deep knowledge.

The convergence of the opinion shows that a steady opinion of the group is met which means the solution set is achieved under the three assumptions. The Teacher Student model successfully resolves the peer evaluation problem in TDR system and can play an important role in future generations of the TDR system. In the future, we plan to adjust the model to having more than one teacher, so that user can learn both from the master and from other teachers or selected friends.

References:

[1] Shi-Kuo Chang, JunHui Chen, Wei Gao and Qui Zhang , “TDR System - A Multi-Level Slow Intelligence System for Personal Health Care”, Proceedings of 2016 International Conference on Software Engineering and Knowledge Engineering, Hotel Sofitel, Redwood City, USA, July 1-3, 2016, 183-190.

[2] Shi-Kuo Chang, "A General Framework for Slow Intelligence Systems", *International Journal of Software Engineering and Knowledge Engineering*, Volume 20, Number 1, February 2010, 1-16.

[3] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, Ben Y. Zhao, User interactions in social networks and their implications, Proceedings of the 4th ACM European conference on Computer systems, April 01-03, 2009, Nuremberg, Germany.

[4] Ming-Feng Chen, Hsi-Ming Yu, Shu-Fang Li and Ta-Jung You, “A Complementary Method for Detecting Qi Vacuity”, *BMC complementary and alternative medicine*, Vol. 9, No. 12, 2009.

Appendix: Data Analysis and Similarity Retrieval

A slow intelligence system SIS includes many kinds of original data from Tian, Di, Ren and Chi sub-systems. In the TDR system, those data are used to judge a person’s health status. Two important functions provided by the TRD system are data analysis and similarity retrieval. In data analysis, data will be analyzed so that alert conditions are detected. In similarity retrieval, data records similar to the current data record are found.

1. Data Analysis

The Analyze Model is specified by:(V, V1, V2, V3, VT, VF, VP, VS, VW), where

V is final result, which may be “Normal”, ”Abnormal” or ”N/A”,

V1 is the result of Chi, which may be is “Normal”, ”Abnormal” or ”N/A”,

V2 is the result of bloodpressure, which may be is “Normal”, ”Abnormal” or ”N/A”,

V3 is the result of SPO2, which may be is “Normal”, ”Abnormal” or ”N/A”,

VT is the result of Tongue,

VF is the result of Tongue,

VP is the result of Tongue,

VS is the result of Tongue,

VW is the result of Tongue,

VT, VF, VP, VS, VW have the same value range: 0, 1, 2, 3 or 4.

$$V1 = \begin{cases} \text{Normal} & (VT + VF + VP + VS + VW \leq 6) \\ \text{Abnormal} & (VT + VF + VP + VS + VW > 6) \\ \text{N/A} & (VT, VF, VP, VS, VW = \frac{N}{A}) \end{cases}$$

$$V2 = \begin{cases} \text{Normal} & ((\text{RegularLow} - 5 \leq \text{Dia} \leq \text{RegularLow} + 5) \text{ and } (\text{RegularHigh} - 5 \leq \text{Sys} \leq \text{RegularHigh} + 5)) \\ \text{Abnormal} & ((\text{RegularLow} - 5 \geq \text{Dia} \text{ or } \text{Dia} \geq \text{RegularLow} + 5) \text{ or } (\text{RegularHigh} - 5 \geq \text{Sys} \text{ or } \text{Sys} \geq \text{RegularHigh} + 5)) \\ \text{N/A} & (\text{Dia, Sys} = \frac{N}{A}) \end{cases}$$

Where Dia is test result for low pressure, and Sys is for high pressure.

RegularLow and RegulaHigh has a definition as following table.

age	Regula rHigh (male)	Regula rLow (male)	Regular High (female)	Regular Low (female)
16-20	115	73	110	70
21-25	115	73	110	71
26-30	115	75	112	73
31-35	117	76	114	74
36-40	120	80	116	77
41-45	124	81	122	78
46-50	128	82	128	79
51-55	134	84	134	80
56-60	137	84	139	82
61-65	148	86	145	83

$$V3 = \begin{cases} \text{Normal} & (SPO2 \in (93\% \sim 98\%)) \\ \text{Abnormal} & (SPO2 \text{ not } \in (93\% \sim 98\%)) \\ \text{N/A} & (SPO2 = \frac{N}{A}) \end{cases}$$

V=V1&V2&V3, where ‘&’ is operator, which is defined as follows.

“Normal” & ”Normal”=”Normal”

“Normal” & ”Abnormal”=”Abnormal”

“Normal” & ”N/A”=“Normal”

”Abnormal” & ”Abnormal”=”Abnormal”

"Abnormal" & "N/A"="Abnormal"
 "N/A" & "N/A"="N/A"

The Data Analysis Algorithm is as follows:

- (1) Get records from database according to received message,
- (2) Get information which includes VT、VF、VP、VS、VW、bloodpressure、SPO2 from all records of the day,
- (3) If can't get enough information at last step, get latest information from the week,
- (4) After finished the (3) step, records for some test items still does not exist, those items should be ignore when computing final result.
- (5) If there is not any test record in the database, the result is defined as N/A, and insert it into the database, end of algorithm,
- (6) According to the formula $V=V1&V2&V3$, we can get the result("Normal" or "Abnormal") and insert into the database, end of algorithm.

2. Similarity Retrieval

This function will retrieve other users' records, and find out few records that most similar to the current user's records. This is helpful for user or doctor to make a comparison to enhance his/her understanding of their own situation. For example, if your record is similar to a patient in the last few days, you would better to be aware of your own health condition. In contrast, if your record is similar to a healthy people, it gains your confidence of your health condition.

To find out similar records, the system uses a simplified K Nearest Neighbor algorithm. The system treats each record as a 5 dimensions' data. The dimensions are Chi factors defined in the system, they are sweaty, pulse, weakBreath, fatigue, and tongue. Each dimension represented by an integer from 1 to 5, that is the score for a specific Chi factor.

After the system retrieve all others records, it calculates the L2 distance (Euclidean distance) between current user's record and each other's records using

$$\text{dist}(c, o) = \sqrt{(s_c - s_o)^2 + (p_c - p_o)^2 + (w_c - w_o)^2 + (f_c - f_o)^2 + (t_c - t_o)^2}$$

where c is current user's record, o is one of other's record, s is score of sweaty, p is score of pulse, w is score of weakBreath, f is score of fatigue, and t is score of tongue.

When the system knows the distances between current user's record and others' records, it can find out the most similar records easily by looking at the distances. Since find similar function do not need to perform any further prediction among current data, thus this can be the end of the nearest neighbor algorithm.

The reason why nearest neighbor algorithm is suitable for this system. One of the reason is, there is almost no assumptions about the data, the only assumptions implied by distance function. In addition, this algorithm is a non-parametric approach, which is mean the data will tell us everything and the system do not need to have any prior knowledge about the data. Although, nearest neighbor algorithm seems fit the system perfectly, it has 2 main disadvantages. One is the algorithm sensitive to irrelevant attributes, also known as dimension curse. Which is mean, high dimension data makes the distance meaningless. Fortunately, the data in the system is 5-dimension data, thus the system do not have this problem. Another disadvantage is the algorithm is computationally expensive, either space wise and time wise. For space wise, the algorithm need to store all examples. However, the system has to keep all the records anyway, therefore, this is not a problem for the system. For time wise, the system has to compute distance to all records with time complexity $O(nd)$, where n is number of existing records, d is cost of computing distance. It is easy to know that, with increasing of n, the system will become slower. The solution in this system is, it only uses recent data so that to control the total number of records involve in the calculation. Due to property of slow intelligence system, data were collected continuously in each cycle, thus missing data will appear in the records. To deal with missing data, the system just simply ignores the incomplete records. To sum up, the nearest neighbor algorithm is one of the most suitable for this system.