Journal of Computer Science 10 (3): 393-402, 2014
ISSN: 1549-3636

doi:10.3844/jcssp.2014.393.402 Published Online 10 (3) 2014 (http://www.thescipub.com/jcs.toc)

DETECTING ABNORMAL BEHAVIOR IN SOCIAL NETWORK WEBSITES BY USING A PROCESS MINING TECHNIQUE

Mahdi Sahlabadi, Ravie Chandren Muniyandi and Zarina Shukur

Software Technology and Management, Faculty of Information Science and Technology,
University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

Received 2013-10-03, Revised 2013-11-19; Accepted 2013-11-22

ABSTRACT

Detecting abnormal user activity in social network websites could prevent from cyber-crime occurrence. The previous research focused on data mining while this research is based on user behavior process. In this study, the first step is defining a normal user behavioral pattern and the second step is detecting abnormal behavior. These two steps are applied on a case study that includes real and syntactic data sets to obtain more tangible results. The chosen technique used to define the pattern is process mining, which is an affordable, complete and noise-free event log. The proposed model discovers a normal behavior by genetic process mining technique and abnormal activities are detected by the fitness function, which is based on Petri Net rules. Although applying genetic mining is time consuming process, it can overcome the risks of noisy data and produces a comprehensive normal model in Petri net representation form.

Keywords: Anomaly Detection, Genetic Algorithm, Social Network, Process Mining, Petri Net

1. INTRODUCTION

There is no standard way to define user interactions formally. Knowing the ways users act in a multi-user interaction environment, like social networking websites, helps to identify the behavior pattern (Priambodo and Satria, 2012). Behavioral pattern can be used to analyze the user's activities. Social networks offer several activities which baffles analyst to identify privacy issues, like how user's profile information should be protected from other clients through these activities, moreover, activities in social networking websites are not transparent (Ahmed, 2011).

This study defines the behavior pattern from the process mining perspective to detect user's abnormal behaviors. For achieving this goal, these actions should be followed: creating a user's activities log file (process mining techniques runs on log files), extracting process models, defining the normal model and detecting abnormal activities. The study has been substantiated by presenting a case study that includes real and syntactic data sets of Facebook. There are some tools fitting the proposed approach, such as ProM (Aalst, 2011a) and CPN (Jensen and Kristensen, 2009). The comprehensive dataset is needed to evaluate the approach; we use both syntactic and real datasets. The data in the syntactic set are replicated intellectually by Color Petri Nets (CPN) tool. The syntactic dataset is a combination of possible behaviors of the users rooted from the real dataset.

Conventionally, data mining tools are used to produce behavior pattern from datasets of social networking website. There are two problems in this approach. First, databases normally are huge and the social network databases are always growing. Most machine learning algorithms run with difficulties whenever they encounter gigantic datasets. Second, databases are dynamic where databases are often exposed and can be altered many times, so data mining systems cannot perform fit classification.

The alternative technique is process mining. Pattern discovery is conducted by process mining; it produces an explicit process model that goes through event logs to make a compatible model for dynamic behavior (Aalst *et al*., 2012).

Corresponding Author: Mahdi Sahlabadi, Software Technology and Management, Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

This study develops a control system for user's activities monitoring to detect threats. The system performs like an intrusion detection system. By logging activities and comparing the actions of the users against predefined model, the system can detect suspicious activities. Knowing the threat in such a dynamic domain is difficult, while, in the intrusion detection system, the discovered intrusion can be as simple as system boundary security intruded by malicious users. Normally, the security border is distinct in such a system. But, in social network websites, the margin between what the users intend to preserve and to share to the public is blurred (Berg and Leenes, 2010). Abnormal behaviors can be detected when compared to normal behaviors. Normal behavior needs norms. Norm recognition is difficult because of the immaturity of social networks and radical variation of users. Norms are changing as time passes by. The heart of the system is norm recognition. After knowing the norms, the abnormal activity can be identified by measuring the norm deviation of the users. There are three algorithms that are commonly used for anomaly detection in process mining.

First, the algorithm based on sampling (Bezerra and Waine, 2013) makes a sample population from log file based on sampling factor. Then, it models M as the normal model. If a trace out of population can be parsed by M, then it is normal, otherwise, it is abnormal. The algorithm exposes some deficiencies. Assuming the sampling model is contaminated by abnormal traces, the M model does not represent norms. With regard to sampling factor, failure is possible as the sampling model can include abnormal traces in different numbers. It is time-consuming as the algorithm runs for each trace. In addition, the traces are labeled abnormal might be parsed partially. Therefore, the algorithm reduces the measurability of abnormal trace recognition since there is no way to determine the parsing ability.

Second, the algorithm based on threshold (Bezerra and Waine, 2013) divides the log file into two sections: frequent and infrequent. The frequent log file is more numerous than the infrequent one. By randomly picking from the infrequent set and inclusion cost of anomalous candidate measurement, algorithm ensures that the deviation never exceeds the threshold. By dividing a log in terms of frequency, contamination problems are partially alleviated. Inclusion cost is a metric for evaluating the differences of process models vertices.

Third, the algorithm based on iteration (Bezerra and Waine, 2013) is similar to the threshold algorithm, although it picks all anomalous traces once. It works with the threshold to categorize the traces into anomalous or normal. Therefore, it saves more time. There is function called select () that returns trace with highest inclusion cost.

2. MATERIALS AND METHODS

2.1. Anomaly Detection using Process Mining

Process mining techniques elicits process models through log files consisting traces. The way how these models have been explored by process mining is out of our scope. In this part we review all the existing methods and algorithms to find abnormalities which have been explored so far. Traditionally, the frequent traces are considered normal. The hypothesis seems awkward because the results tremendously depend on a training set. Abnormal traces are categorized with respect to not being parsed completely by the frequent process model (norms) while the normal activity being parsed completely. This is black and white solution to the problem. By the intervention of cost inclusion factor, the solution gets more realistic. Bezerra (2009) propose a method to detect anomalies by using a process mining technique. First, the log gets trimmed by scoping. Scoping means just focusing on data stored in a specific range of time. The time-scoping concept looks at specific data at a specific range of time. The scope set is a subset of all log traces $t' \subseteq T$, in which T is a set containing all log file traces. Then, scoped-data segregates by fitness and appropriateness metrics. Fitness factor makes two separate sets and considers the fittest model as norm based on appropriateness. Although the inclusion cost can boost abnormality measurement, it is still awkward because it just works based on differentiation of vertices. Given A is a set of activities, trace t represents a sequence of activities such that $t \in A*$ in which $A*$ is power set of A. The fitness function runs for an instance of log L. f_M: L → B for a trace from log L in which B is behavior. If trace t is instance of a model M, t can be completely parsed by M:

$$fM(t)\begin{cases}\text{True, if t can be replayed by modle M}\\ \text{False, otherwise}\end{cases}$$

Fitness function is f:{(M,L)|M is a model ∧ L is a log} → [0, 1] that illustrates the level of fitness model M to a log L to find how the M model complies with a specific log. Formula determines the extent M is able to fit a Log. The function is illustrated as dividing the number of all possible instances of M in log L by the number of all traces in log L:

$$f(M,L) = \frac{\left|\left\{t \in L \middle| fM(t)\right\}\right|}{|L|}$$

Accordingly, models that exceed fitness minimum values are candidates for testing for M. If equals 100%, it means that the model is able to parse all log traces because it is capable of getting the M model as a norm. Nevertheless, 100% of fitness does not affirm an appropriate model. A general model called the "flower model" that is able to parse extra traces even if the traces are not placed in the log file. All subsets of activities {A, B, C, D} could be parsed by the model even if the activities do not exist in the real world (Accorsi and Stocker, 2012).

a: {(M,L)|M is a model ∧ L is a log} → [0, 1] is a function that measures the level of appropriateness of model M against L. Here, appropriateness is interpreted as a simple model instead of a complex one. Too much extra behavior is undesirable. As a result, the function represents a balance between structural complexity and extra-behavior support. In anomalous traces recognition, assume that log L and p ∈ [0, 1] is the thresholds of fitness in log L and M∗ is an appropriate model:

$$f(M*,L) \geq p; \forall M' f(M',L) \geq p \Rightarrow a(M',L) \leq a(M*,L)$$

Then, an anomalous trace t′∈ L is defined as follows: $\neg f_{M*}(t)$, i.e.,$\{t \in L | \neg f_{M*}(t)\}$ is the set of anomalous traces.

2.2. ProM and CPN Tool

ProM is plug-able tool using Extensible Event Stream (XES)/ Mining Extensible Markup Language (MXML) as input format to provide a common basis for all kinds of process mining techniques, supporting the loading and filtering of event logs and the visualization of results in Petri Net, Event-driven Process Chain (EPCs). It can distribute the execution of plug-ins over multiple computers (Ahmed, 2011). CPN Tools is a simulating, analyzing and editing tool for untimed and timed Hierarchal Colored Petri nets. It supports incremental syntax checking and code generation, whenever, net is being constructed. A fast simulator provides state spaces generated and analyzed, a standard state space report contains information (boundedness and liveness properties). It also provides CPN ML language programing feature for variable definition and function execution (Aalst, 2011b).

2.3. Proposed Method

Anomaly detection algorithms in process mining are concerned with three facts: (1) Process mining algorithm that used to discover the model prior to anomaly detection. Study applies genetic algorithmic as it mines mains patterns and structures. (2) Metric quantifying the manipulation of model structure in order to measure the extent to which the trace fits the model. Study applies formula based on Petri Net presentation. (3) Process model presentation or the notion used for process modeling. Here we use Petri Net to show user behavior. Petri Net has flexible structure to represent many actions (Barghash *et al*., 2011; Athamena, 2012).

All reviewed approaches in previous sections suffer from lack of model mutations to achieve norms. The approaches recognize norms as most frequent traces model. Process model mutation is needed to manipulate process model, proposed approach aims to revise the extracted models. In this case, genetic mining is the key to this problem. The genetic mining algorithm is able to mine all major types of pattern structures, although it has low performance (Accorsi and Stocker, 2012).

Step1. Event Recording to Collect Relevant Data

The events are made of three elements: Object(s) [x], a property [P] and time or a temporal interval [t]. Events can be represented in a 3-tuple, like [x, P, t]. Two principles define a unique event, namely, the existence and the identity condition. The existence condition states that [x, P, t] exists if and only if object x exemplifies the n-adic P at time t. This condition means a unique event exists if the above is met. The identity condition states that [x, P, t] is [y, Q, t`] if and only if x = y, P = Q and t = t` (Berg and Leenes, 2010).

The raw data should be mapped to a log file. For database D, each tuple d ∈ D has the form of d =<a,t,s>, where d.a identifies an actor uniquely and d.s indicates the location of this actor at time d.t. Although in reality time runs seamlessly, it is expressed as a discrete value at a particular granularity. Furthermore, each data unit may be generated anytime, rather than only at strictly regular intervals as found in a time series. Meanwhile, we model space as a collection of semantic locations, which may be physical locations such as rooms and buildings, or cyber locations such as web addresses and domains. A semantic rather than a more refined coordinate space is more practical to assume because the latter is more difficult to record accurately and requires mapping to correlate a coordinate to an actual location. Small-scale efforts to track locations, such as within building complexes, will likely settle for semantic location because it will be easier and more useful to

know that a person is at a particular room than at a given xyz coordinate. A semantic location may be expressed at several levels of granularity (e.g., room or building, web address, or domain) and may also have a natural meaning. These conditions indicate the purpose of the location, which may render a co-occurrence and hence, become even more meaningful.

We proposed new tuple which has two extra fields, include r and m; d.r is the role of an actor in an interaction and r $\in$ R*, which means the actor can have more than one role:

- R$\in$ {initiator, sender, receiver, participant}
- m is the event type and m $\in$ M

M$\in$ {Investigate/view profiles, Investigate/view groups, Investigate/view events, Investigate/view pictures, Investigate, view notes or posted items, View the news feed, Search for people (profiles), search for groups, search for events, Search for pictures, Check messages, Reply to messages, Send messages, Read wall posts, Respond to wall posts, Make wall posts, Poke others (initiate),Return pokes (reciprocate), Create groups, Create events, Post pictures, Check out advertisements}.

A = <a, t, s, r, m> is a new tuple used in our method. The r field addresses the role in interaction and the m field is intended for knowing the event type. r and m sets has been elicited mostly from our direct observation of social networks websites activities. After providing a log file, we need to extract all user activities to model normal behaviors.

Step2. Making Log File

The data which has been collected in first step is converted to log file. The data should contain the six items. Process mining work is based on the log file that typically contains six properties of data: (1) Case: represents a process instance, (2) Activity: identifies an activity instance for the process, (3) Event: specifies the event type of an activity instance (i.e., START or COMPLETE), (4) Repository: specifies the data repositories an activity instance writes to or reads from, (5) Participant: specifies who performs an activity instance and (6) Time: specifies the time of occurrence of an event (Aalst, 2011c).

Process-mining enables analyzer to look at process model in three different views: (1) Control flow perspective: the produced process model shows how the process is within the system, thus rendering a more manageable process. (2) Organizational perspective: Workflow mining extracts social networks that are hidden in the workflow and defines the role of a participant and “who” the actor is (3). Case perspective: The view considers finding which data are linked to instances and “what” data are related to the process for further analysis.

The MXML event log format meets the process mining requirements for process mining tools such as ProM. Process mining technique and data stored in log file produced multi-user interaction model. Users have role (e.g., sender, receiver, viewer, follower) while they involve an activity (e.g., logging in, sharing an image and so on).

Step3. Synthesize Log File

A data log can be simulated with respect to a small real log file. ProM mines real log file and makes process models. These models are initial models of user behaviors. The study applies CPN tool to simulate this models and amplifies data, as available datasets are more applicable for data mining.

Step4. Genetic Process Mining

The discovery of the most appropriate model is crucial to the study because the rest of the definitions revolve around the norm. Genetic process mining is based on genetic algorithm. In this study, the algorithm performs among primitive population of individuals, discovers the process model of each log trace. A value generated by the fitness function is assigned to the individual, which allowed for the assessment of individual quality. Iteratively, algorithms evolve with the populations and select the fittest individual. The next step involves operator intervention crossover and mutation to produce higher quality generation. Finally, the most appropriate model is selected when the stop condition is satisfied. Moreover, the genetic algorithm can mine most process structures required in the control-flow model (Aalst, 2011a).

This step is to produce appropriate model so called norm. In a mathematical perspective, a process is composed of a set of tasks, while an activity is a group of tasks and a trace is a sequence of tasks. A set of activities containing visible tasks constitutes a log file. Individuals, who are similar in terms of activities, comprise a population. An algorithm randomly creates a population by using different individuals as inputs and outputs. In this study, the fitness value is based on the number of traces that be parsed correctly, in the event log.

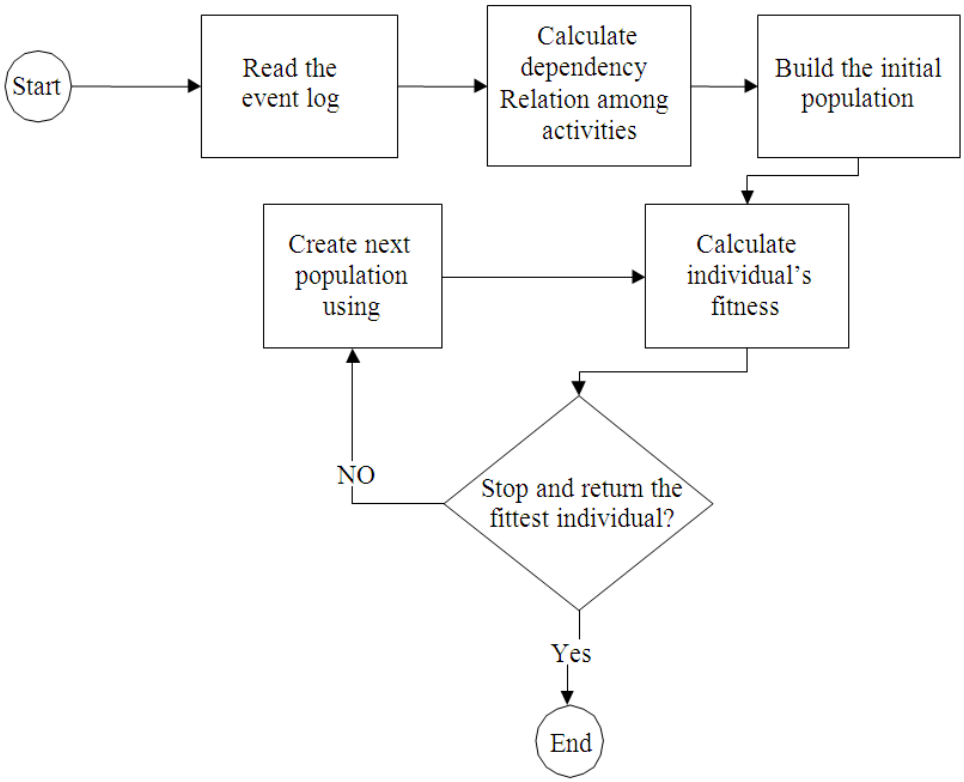

Fig. 1. Main steps of the genetic algorithm

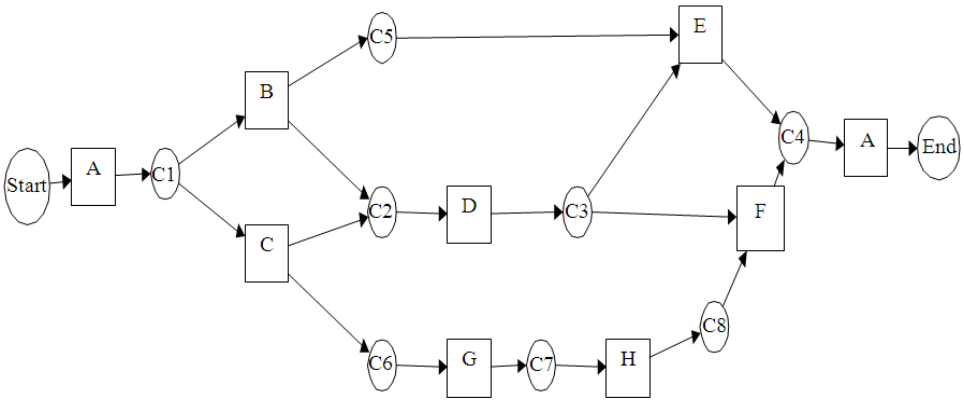

Fig. 2. Appropriate model

When an individual describes the behavior of a log correctly, the fitness value becomes higher. The fitness value is 1 in a normal, noise-free situation, but ranges from 0 to 1 in practical situations (Aalst, 2011b). **Figure 1** presents the genetic algorithm showing the process mining steps.

Step5. Anomaly Detection Method (Petri Net)

The method describes the mismatches between the log traces and the model based on the deviation of event log traces from appropriate process models. Formerly, once the model could replay the traces completely, the function returns "1," otherwise "0."

Table 1. Log traces for 1333 and 1222

No. of instance	Log traces	Produced	Consumed	Missed	Remained
1331	ABDEA	1	0	0	0
1331	ABDEA	2	1	0	0
1331	ABDEA	4	2	0	0
1331	ABDEA	5	3	0	0
1331	ABDEA	6	5	0	0
1331	ABDEA	7	7	0	0
1222	ACHDFA	1	0	0	0
1222	ACHDFA	2	1	0	0
1222	ACHDFA	4	2	1	0
1222	ACHDFA	5	3	1	0
1222	ACHDFA	6	4	1	0
1222	ACHDFA	7	6	1	0
1222	ACHDFA	8	8	1	1

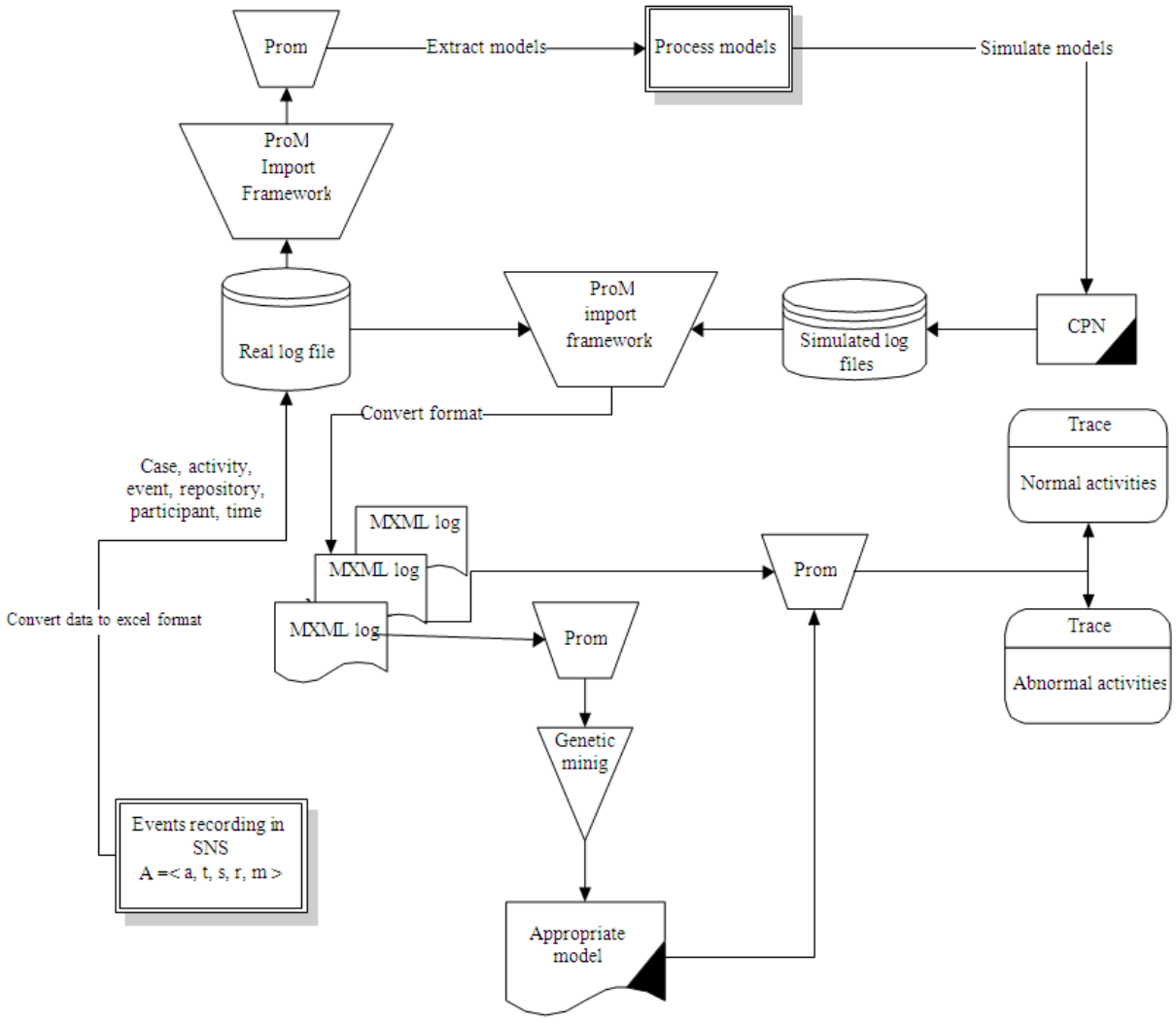

Fig. 3. Proposed solution overview

With respect to Petri net token firing rule (a transition fires when all incoming arcs carry token(s)), if there is not token for a trace to replay, artificial token will be placed, so the trace could be parsed. At the end, artificial and left tokens are taken into account for mismatch measurement.

Equation (1) calculates the abnormalities (Aalst, 2011c):

$$f = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} p_i}\right) \quad (1)$$

For all i, $m_i < c_i$ and $r_i < p_i$ and therefore $0 \leq f \leq 1$.

Table 1 show the logs traces for two instances against a model which is indicated in **Fig. 2**. Trace 1331 (i = 1) can be replayed without any problems, which means that no tokens are missing (m = 0) or remaining (r = 0). Meanwhile, log trace 1222 replays for trace i = 2 of event log L in process model M. The replay of this trace requires the artificial creation of one token (m = 1) and one token left behind (r = 1) (Accorsi and Stocker, 2012).

2.4. Case Study

The real log file was produced by Brightsoft Company located in Malaysian Technology Park, the events were recorded among the colleagues community in Facebook. The real log file instances are limited to six tasks. There are two simulated log files: log A and Log B. Log file A is a syntactic log file simulated based on a predefined process elicited from real log file. For log B simulation, the elicited process model from real log file is manipulated to have duplicated and invisible tasks. The stop criteria is thousand times of generations, crossover rate of 0.2, mutation of 0.2 and elitism of 0.2 are applied for all log files. The model could parse all traces that already exist in the log file. A diagnostic diagram finds dependencies or causality dependencies.

2.5. Experiment

Figure 3 shows the proposed solution overview for handling abnormal behaviors. System input is events recorded that makes real log file. Based on real log file models, the syntactic log file is made. CPN is a simulation tool for providing a simulated log file. ProM is a pluggable tool that to discover processes, verifies conformance and adherence to genetic mining. The ProM Import Framework is a tool for converting many kinds of log files to MXML format identifiable by ProM. The system mines the log file and creates an appropriate model. Next, a conformance checker separates normal from abnormal activities.

3. RESULTS

The result of study analysis is presented in figures and **Table 2**. Results are shown briefly. **Figure 4** indicates log B process model resulted from Alpha algorithm, **Fig. 5** indicates Log B model resulted from genetic mining. **Figure 6** shows diagnostic results, these are for finding deficiencies in a model. **Figure 7** is for finding dependency in model.

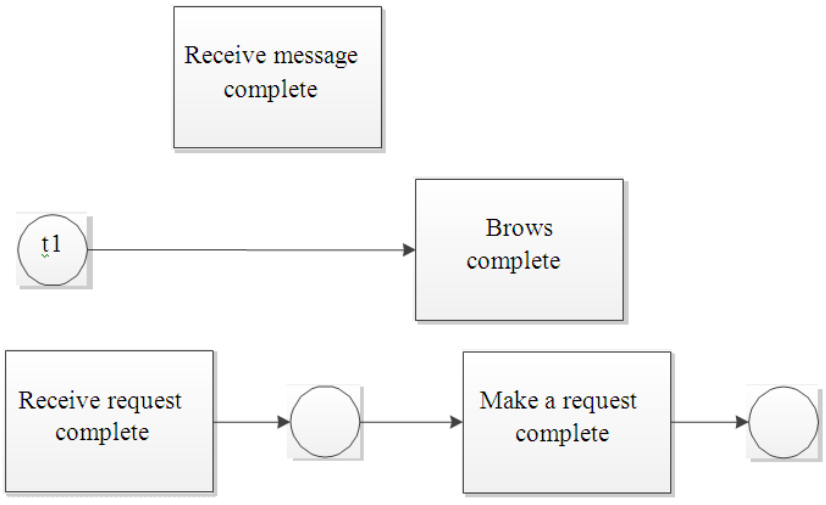

Fig. 4. Alpha algorithm result of log B

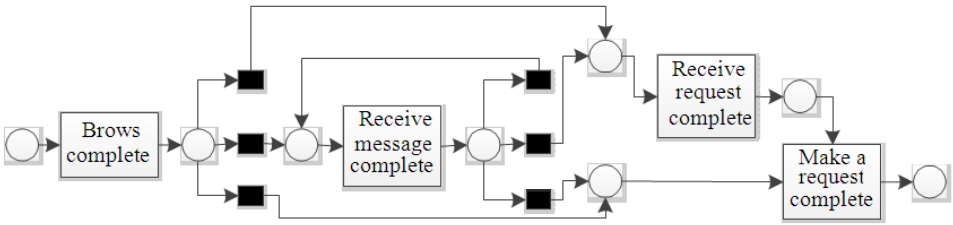

Fig. 5. Appropriate model log B: Petri net result

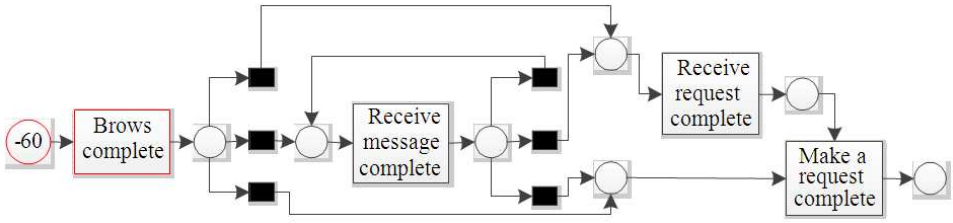

Fig. 6. Diagnostic results

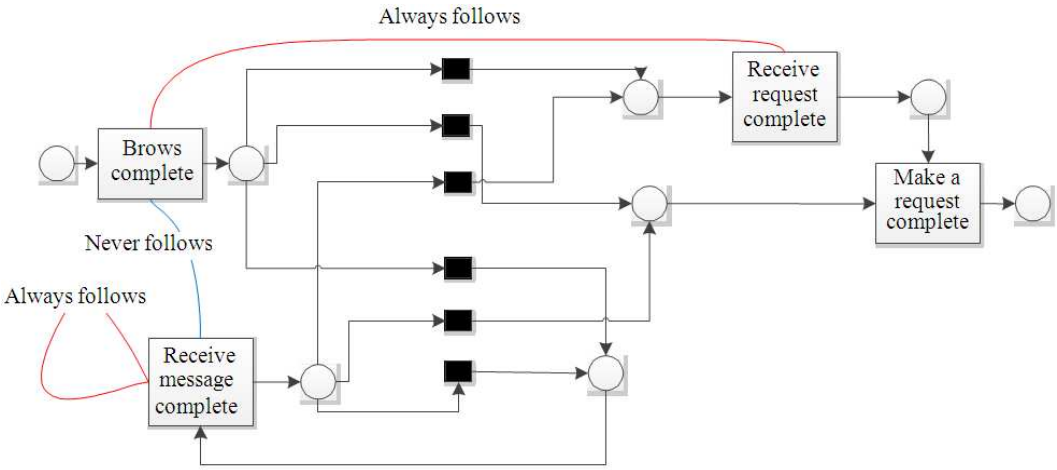

Fig. 7. Log trace 263; Fitness = 0.93

Table 2. Comparison of results

Dataset and noise	Process instances	Task number	Computation time (minute)	Fittest value
Real log and rare	6	42	3.34	1.000
Log A and rare	141	710	24.13	0.980
Log B and duplicate, hidden task	150	720	33.47	0.568

First, results demonstrate user behavior pattern, for instance a post action always follows another post action, whereas making requests never comes immediately after post. A study of the diagnostic diagram reveals causality dependencies. Second, Genetic process mining makes understandable model rather than other techniques, **Fig. 4** indicates ambiguous and disconnected model while **Fig. 5** shows comprehensible model with duplicate and invisible tasks. Third, abnormalities could be detected; the **Fig. 6** shows abnormalities in BROWS task. The 60 tokens remained, it seems, user when received message, ignores message and keeps on browsing on Facebook.

4. DISCUSSION

Table 2 compare case study's log files in terms of noise, size (process instances, task number) and computation time. The fittest value indicates model power to pars all traces in a log file. According to **Table 2**, the genetic process-mining algorithm overcomes these issues by using genetic operators to search for the fittest solution from all possible process models. The main disadvantage of using genetic algorithms is the required computation time but genetic algorithms are resilient to noisy data, capable of discovering complex constructs, such as loops and handle duplicated tasks very well.

The current algorithm needs several iterations to get good process models with high fitness. This problem makes the algorithm process time consuming and sometimes, makes the results not accepted (Li *et al*., 2011).

5. CONCLUSION

Social networking websites are currently exposed to threats by unknown user behavior. For example, Facebook seems very naïve in managing data privacy in user interaction (Ahmed, 2011). The proposed solution identifies user behavior pattern for better control and to analyze user activity. This study defines process model for users as norm and the training dataset consists of abnormal and normal instances. The system norm is affected by the abnormalities existing in the datasets, but possibility of norm contamination to abnormalities become less as study uses genetic algorithm to produce elites.

The idea of using genetic process mining for finding appropriate model in social network websites introduced first time in this study. For future work, we extend our method to handle, norms change with time (Aalst *et al*., 2012) and analysis is offline. The system currently requires online analysis to adapt with changes of behaviors.

6. ACKNOWLEDGEMENT

This study is supported by FRGS, Ministry of Higher Education (Malaysia). Grant code: FRGS /1/2012/SG05/UKM/02/3.

7. REFERENCES

Aalst, W.M.P.V.D., 2011a. Tool Support. In: Process Mining: Discovery, Conformance and Enhancement of Business Processes, Aalst, W.V.D. (Ed.), Springer, Berlin, ISBN-10: 3642193447, pp: 265-270.

Aalst, W.M.P.V.D., 2011b. Advanced Process Discovery Techniques. In: Process Mining: Discovery, Conformance and Enhancement of Business Processes, Aalst, W.V.D. (Ed.), Springer, Berlin, ISBN-10: 3642193447, pp: 265-270.

Aalst, W.M.P.V.D., 2011c. Tool Support. In: Process Mining: Discovery, Conformance and Enhancement of Business Processes, Aalst, W.V.D. (Ed.), Springer, Berlin, ISBN-10: 3642193447, pp: 265-270.

Aalst, W.V.D., A. Adriansyah, A.K.A.D. Medeiros, F. Arcieri and T. Baier *et al*., 2012. process mining manifesto. Bus. Process Manage. Workshops, 99: 169-194. DOI: 10.1007/978-3-642-28108-2_19

Accorsi, R. and T. Stocker, 2012. On the exploitation of process mining for security audits: The conformance checking case. Proceedings of the 27th Annual ACM Symposium on Applied Computing, Mar. 26-30, ACM Press, Riva (Trento), Italy, pp: 1709-1716. DOI: 10.1145/2245276.2232051

Ahmed, J., 2011. Privacy issues in social networking platforms: Comparative study of facebook developers platform and opensocial. Proceedings of the International Conference on Computer Networks and Information Technology, Jul. 11-13, IEEE Xplore Press, Abbottabad, pp: 179-183. DOI: 10.1109/ICCNIT.2011.6020927

Athamena, Z.H., 2012. A petri net based agent behavioral testing. Am. J. Applied Sci., 9: 1876-1883. DOI: 10.3844/ajassp.2012.1876.1883

Barghash, M.A., O.M. Abuzeid, A.N. Al-Rabadi and A.M. Jaradat, 2011. Petri nets and ladder logic for fully-automating and programmable logic control of semi-automatic machines and systems. Am. J. Eng. Applied Sci., 4: 252-264. DOI: 10.3844/ajeassp.2011.252.264

Berg, B.V.D. and R. Leenes, 2010. Audience segregation in social network sites. Proceedings of the 2010 IEEE Second International Conference on Social Computing, Aug. 20-22, IEEE Xplore Press, Minneapolis, MN., pp: 1111-1116. DOI: 10.1109/SocialCom.2010.165

Bezerra, F. and J. Waine, 2013. Algorithms for anomaly detection of traces in logs of process aware information systems. Inform. Syst., 38: 33-44. DOI: 10.1016/j.is.2012.04.004

Bezerra, F.A., 2009. Anomaly detection using process mining. In: Enterprise, Business-Process and Information Systems Modeling, Halpin, T.A. (Ed.), Springer Berlin Heidelberg, ISBN-10: 3642018629, pp: 149-161.

Jensen, K. and L.M. Kristensen, 2009. Introduction to Modeling and Validattion. In: Coloured Petri Nets: Modelling and Validation of Concurrent Systems, Jensen, K. and L.M. Kristensen, (Eds.), Springer, Dordrecht, ISBN-10: 3642002846, pp: 8-10.

Li, J., J. Ouyang and M. Feng, 2011. A heuristic genetic process mining algorithm. Proceedings of the 7th International Conference on Computational Intelligence and Security, Dec. 3-4, IEEE Xplore Press, Hainan, pp: 15-19. DOI: 10.1109/CIS.2011.12

Priambodo, R. and R. Satria, 2012. User behavior pattern of mobile online social network service. Proceedings of the International Conference on Cloud Computing and Social Networking (ICCCSN), Apr. 26-27, IEEE Xplore Press, Bandung, West Java, pp: 1-4. DOI: 10.1109/ICCCSN.2012.6215732