A Role of Heuristics Miner Algorithm in the Business Process System

Saravanan .M.S
Research Scholar in R & D Centre, Bharathiar University, Coimbatore, Tamil Nadu, INDIA.
Asst. Prof. in VEL TECH Dr. RR & Dr. SR Technical University Avadi, Chennai, INDIA.
saranenadu@yahoo.co.in

Rama Sree .R.J
Professor in the Department of Computer Science
Rashtriya Sanskrit University,
Tirupati, Andhra Pradesh, INDIA.
rjramasree@yahoo.com

ABSTRACT

The main idea of process mining is to extract knowledge or information from event logs recorded by an information system. Till now, the information in these event logs was rarely used to analyze the underlying processes. Process mining aims at improving this by providing techniques and tools for discovering process, organizational, social, and performance information from event logs. Process mining has become a bright research area. In this paper we discuss the challenging process mining domain and demonstrate a heuristics driven process mining algorithm; the so called Heuristics Miner in detail. Heuristics Miner is a practical applicable mining algorithm that can deal with noise, and can be used to express the main behavior that is not all details and exceptions, registered in an event log. The business process system has a complex process system, which deals about many cases or audit trail entries and various event logs. This paper deals about the role of Heuristics Miner algorithm in the field of Business process system, also the Heuristics Miner algorithm is compared with the other mining algorithm such as α-algorithm. Hence, we analyzed that, is it possible to develop a control flow process mining algorithm such as Heuristics Miner algorithm can discover all the common control flow structures and is robust to noisy logs at once? This paper attempts to provide an answer to this question.

Keywords

Event log, Heuristics Miner, Business process, Process mining, α-algorithm.

1. INTRODUCTION

Nowadays, most organizations use information systems to support the execution of their business processes [1]. Examples of information systems supporting operational processes are Workflow Management Systems (WMS) [2] [3], Customer Relationship Management (CRM) systems and Enterprise Resource Planning (ERP) systems and so on. These information systems may contain an explicit model of the processes may support the tasks involved in the process without necessarily defining an explicit process model. All these information systems have activities and then these activities are converted into the logs.

These produced logs usually contain data about cases that is process instances, which have been executed in the organization, the times at which the tasks were executed, the persons or systems that performed these tasks, and other kinds of data. These logs are the starting point for process mining, and are usually called event logs. For instance, consider the event log in Table 1. This log contains information about four process instances that is cases of a process that handles fines. Process mining targets the automatic discovery of information from an event log. This discovered information can be used to deploy new systems that support the execution of business processes or as a feedback tool that helps in auditing, analyzing and improving already enacted business processes.

The main benefit of process mining techniques is that information is objectively compiled. In other words, process mining techniques are helpful because they gather information about what is actually happening according to an event log of an organization, and not what people think that is happening in this organization. The starting point of any process mining technique is an event log. The type of data in an event log determines which perspectives of process mining can be discovered. If the log provides the tasks that are executed in the process and it is possible to infer their order of execution and link these tasks to individual cases or process instances then the control flow perspective can be mined. The log in Table 1 has this data fields such as Case ID, Task Name and Timestamp. So, for this log, mining algorithms could discover the process in Figure 1, Basically, the process describes that after a fine is entered in the system, the bill is sent to the driver. If the driver does not pay the bill within one month, a reminder is sent. When the bill is paid, the case is archived. If the log provides information about the persons systems that executed the tasks, the organizational perspective can be discovered. The organizational perspective discovers information like the social network in a process, based on transfer of work, or allocation rules linked to organizational entities like roles and units. For instance, the log in Table 1 shows that Raja transfers work to both Saran that is case 2 and John that is cases 3 and 4, and John sometimes transfers work to Saran that is case 4. Besides, by inspecting the log, the mining algorithm could discover that Saran never has to send a reminder more than once, while John does not seem to perform as good. The managers could talk to Saran and check if he has another approach to send reminders that John could benefit from. This can help in making good practices a common knowledge in the organization. When the log contains more details about the tasks, like the values of data fields that the execution of a task modifies, the case perspective that is the perspective linking data to cases can be discovered. So, for instance, a forecast for executing cases can be made based on already completed cases, exceptional situations can be discovered etc. In our particular example, logging information about the profiles of drivers that is like age, gender, car etc. could help in assessing the probability that they would pay their fines on time. Moreover, logging information about the places where the fines were applied could help in improving the traffic measures in these places. From this explanation, the reader may have already noticed that the control flow perspective relates to the How? Question, the organizational perspective to the Who? Question and the case perspective to the What? Question. All these three perspectives are complementary and relevant for process mining. However, in this thesis we focus on the control flow perspective of process mining.

Table 1. Example of Event Log-1

Case ID	Task Name	Event Type	Originator	Timestamp
1	File Fine	Completed	Raja	22-08-10 14:00:00
2	File Fine	Completed	Raja	22-08-10 15:00:00
1	Send Bill	Completed	System	22-08-10 15:05:00
2	Send Bill	Completed	System	22-08-10 15:07:00
3	File Fine	Completed	Raja	23-08-10 10:00:00
3	Send Bill	Completed	System	23-08-10 14:00:00
4	File Fine	Completed	Raja	24-08-10 11:00:00
4	Send Bill	Completed	System	24-08-10 11:10:00
1	Process Payment	Completed	System	26-08-10 15:05:00
1	Close Case	Completed	System	26-08-10 15:06:00
2	Sent Reminder	Completed	Saran	22-09-10 10:00:00
3	Send Reminder	Completed	John	23-09-10 10:00:00
2	Process Payment	Completed	System	24-09-10 09:05:00
2	Close Case	Completed	System	24-09-10 09:06:00
4	Send Reminder	Completed	John	24-09-10 15:10:00
4	Send Reminder	Completed	Saran	24-09-10 17:10:00
4	Process Payment	Completed	System	30-09-10 14:01:00
4	Close Case	Completed	System	30-09-10 17:30:00
3	Send Reminder	Completed	John	21-10-10 10:00:00
3	Send Reminder	Completed	John	21-11-10 10:00:00
3	Process Payment	Completed	System	24-11-10 14:00:00
3	Close Case	Completed	System	24-11-10 14:01:00

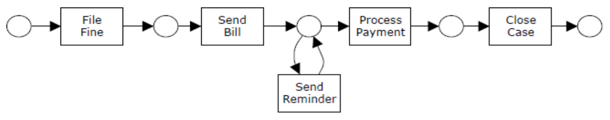

Fig 1: Petri Net illustrating the control flow perspective that can be mined from the event log in Table 1.

2. CONTROL FLOW MINING

The control flow perspective mines a process model that specifies the relations between tasks in an event log. From event logs, one can find out information about which tasks belong to which process instances, the time at which tasks are executed, the originator of tasks, etc. Therefore, the mined process model is an objective picture that depicts possible flows that were followed by the cases in the log that is assuming that the events were correctly logged. Because the flow of tasks is to be depicted, control flow mining techniques need to support the correct mining of the common control flow constructs that appear in process models. These constructs are: sequences, parallelism, choices, loops, and non free choice, invisible tasks and duplicate tasks [4]. Sequences express situations in which tasks are performed in a predefined order, one after the other. For instance, for the model in Figure 2, the tasks "*Enter Website*" and "*Browse Products*" are in a sequence. Parallelism means that the executions of two or more tasks are independent or concurrent. For instance, the task "*Fill in Payment Info*" can be executed independently of the tasks Login and "*Create New Account*" in Figure 2. Choices model situations in which either one task or another is executed. For instance, the tasks "*Remove Item from Basket*" and "*Add Item to Basket*" are involved in the same choice in the model in Figure 2. The same holds for the tasks "*Cancel Purchase*" and "*Commit Purchase*". Loops indicate that certain parts of a process can be repeatedly executed. In the model in Figure 2, the block formed by the tasks "*Browse Products*", "*Remove Item from Basket*", "*Add Item to Basket*" and "*Calculate Total*" can be executed multiple times in a row. Non free choice constructs model a mix of synchronization and choice. For instance, have a look at the non free choice construct involving the tasks "*Calculate Total*" and "*Calculate Total with Bonus*". Note that the choice between executing one of these two tasks is not done after executing the task "*Fill in Delivery Address*", but depends on whether the task Login or the task "*Create New Account*" has been executed. In this case, the non free choice construct is used to model the constraint that only returning customers are entitled to bonuses. Invisible tasks correspond to silent steps that are used for routing purposes only and, therefore, they are not present in event logs. Note that the model in Figure 2 has three invisible tasks, represented in the black rectangles. Two of these invisible tasks are used to skip parts of the process and the other one is used to loop back to "*Browse Products*". Duplicate tasks refer to situations in which multiple tasks in the process have the same label. Duplicates are usually embedded in different contexts that are surrounding tasks in a process. The model in Figure 2 has two tasks with the same label "*Calculate Total*". Both duplicates perform the same action of adding up the prices of the products in the shopping basket. However, the first "*Calculate Total*", refer top part of Figure 2 does so while the client is still selecting products and the second one refer bottom part of Figure 2, computes the final price of the whole purchase. Control flow process mining algorithms should be able to tackle these common constructs. In fact, there has been quite a lot of work on mining the control flow perspective of process models [5] [6] [7] [8] [9] [10]. For instance, the work in [8] can mine duplicate tasks, [10] can mine non free choice, [5] proves to which classes of models their mining algorithm always works, [7] mines common control flow patterns and [9] captures partial synchronization in block structured models. However, none of the current control flow process mining techniques is able to mine all constructs at once. Furthermore, many of them have problems while dealing with another factor that is common in real-life logs: the presence of noise. Noise can appear in two situations: event traces were somehow incorrectly logged that is for instance, due to temporary system misconfiguration or event traces reflects exceptional situations. Either way, most of the techniques will try to find a process model that can parse all the traces in the log. However, the presence of noise may hinder the correct mining of the most frequent behavior in the log. The first reason why these techniques have problems to handle all the constructs is that they are based on local information in the log. In other words, they use the information about what tasks directly precede or directly follow each other in a log to set the dependencies between these tasks. The problem is that some of these dependencies are not captured by direct succession or precedence. For instance, consider the non free choice construct in Figure 2.

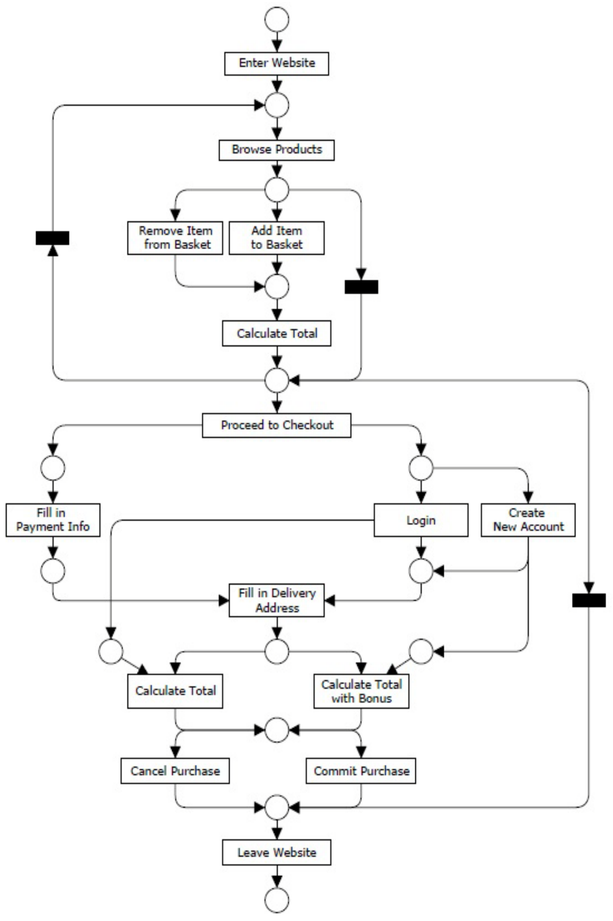

Fig 2. Example of a Petri Net that contains all the common control flow constructs that may appear in Business process (Generated from [11])

Note that the tasks involved in this non free choice constructs never directly follow of precede each other. A second reason why some of the techniques cannot mine certain constructs is because the notation they use to model the processes does not support these constructs. For instance, the notation used by the α-algorithm [5] and extensions [10] does not allow for invisible tasks or duplicate tasks. Furthermore, approaches like the ones in [8] and [9] only work over block structured processes. Finally, for many of these approaches, the number of times a relation holds in the log is irrelevant. Thus, these approaches are very vulnerable to noise because they are unable to distinguish between high frequent and low frequent behavior. Given all these reasons, we decided to investigate the following research question: Is it possible to develop a control flow process mining algorithm that can discover all the common control flow structures and is robust to noisy logs at once? This paper attempts to provide an answer to this question.

3. RELATED WORK

In process mining there are several techniques to discover process model. Each technique has different perspective and working strategy. Some algorithms work with local strategy to build model step by step and others work with global strategy to work based on a one strike search for the optimal model. And also there is differentiation between ability to extracting models dealing with noisy information, looping, duplicate tasks, incompleteness log, and also how much of information this model will be aware with. The following three different examples of process mining techniques:

3.1. Alpha Mining

This algorithm works based on local strategy technique to build model. The alpha algorithm assumes event logs to be complete and does not contain any noise. Therefore, the alpha algorithm is sensitive to noise and incompleteness of event logs. On the other hand it gives us a quick view of natural of workflow model we work on.

3.2. Genetic Mining

This algorithm works based on global strategy technique to build model. This technique can deal with noisy and duplicate tasks and can provide us with detailed model. It based on genetic algorithm so we can say the time against details.

3.3. Heuristics mining

This technique extend alpha algorithm by consider the frequency of traces in the log. Heuristics miner can deal with noise, and can be used to express the main behavior. The Heuristics Miner Plug-in mines the control flow perspective of a process model. To do so, it only considers the order of the events within a case. In other words, the order of events among cases isn't important. For instance for the log in the log file only the fields case id, time stamp and activity are considered during the mining. The timestamp of an activity is used to calculate these orderings. In Table 2 it is important that for case 1 activity A is followed by B within the context of case 1 and not that activity A of case 1 is followed by activity A of case 2. Therefore, we define an event log as follows. Let *T* be a set of activities. S ϵ T is an *event trace*, i.e., an arbitrary sequence of activity identifiers. W Ɛ T is an *event log*, i.e., a multiset that is bag of event traces. Note that since *W* is a multiset, every event trace can appear more than once in a log. In practical mining tools frequencies become important. If we use this notation to describe the log shown in Table 2 we obtain the multiset W = [ABCD; ABCD; ACBD; ACBD; AED]. To find a process model on the basis of an event log, the log should be analyzed for causal dependencies, e.g., if an activity is always followed by another activity it is likely that there is a dependency relation between both activities. To analyze these relations we introduce the following notations. Let W be an event log over T, i.e., W Ɛ T. Let a; b ϵ T: [12]

1. a $>_W$ b if and only if there is a trace S = t_1,t_2,t_3 : : : t_n and i Ɛ {1; : : : ; n -1} such that S Ɛ W and t_i = a and t_{i+1} = b,
2. a $\rightarrow_W$ b if and only if a $>_W$ b and b > not equal $_W$ a,
3. a $\neq_W$ b if and only if a > not equal $_W$ b and b > not equal $_W$ a, and
4. a $\|_W$ b if and only if a $>_W$ b and b $>_W$ a.
5. a $>>_W$ b if and only if there is a trace S = t_1,t_2,t_3 ….. t_n and i Ɛ {1; : : : ; n - 2} such that S ϵ W and t_i = a and t_{i+1} = b and t_{i+2} = a,

6. a $>>>_W$ b if and only if there is a trace $S = t_1, t_2, t_3 \ldots\ldots t_n$ and $i < j$ and $i, j \in \{1 \ldots\ldots n\}$ such that $S \in W$ and $t_i = a$ and $t_j = b$.

Consider the event log W = {ABCD; ABCD; ACBD; ACBD; AED}, that is the log shown in Table 1. The first relation $>_W$ describes which activities appeared in sequence that is one directly following the other. Clearly, A $>_W$ B, A $>_W$ C, A $>_W$ E, B $>_W$ C, B $>_W$ D, C $>_W$ B, C $>_W$ D, and E $>_W$ D. Relation $\rightarrow_W$ can be computed from $>_W$ and is referred to as the direct dependency relation derived from event log W. A $\rightarrow_W$ B, A$\rightarrow_W$ C, A $\rightarrow_W$ E, B $\rightarrow_W$ D, C $\rightarrow_W$ D, and E $\rightarrow_W$ D. Note that B $\neq_W$ C because C $>_W$ B. Relation $\|$W suggests concurrent behavior, that is potential parallelism. For log W activities B and C seem to be in parallel, that is B $\|_W$ C and C $\|_W$ B. If two activities can follow each other directly in any order, then all possible inter leavings are present, that is if for instance 10 activities are in parallel thus can be a practical problem and therefore they are likely to be in parallel [12]. Relation $\neq_W$ gives pairs of transitions that never follow each other directly. This means that there are no direct dependency relations and parallelism is unlikely. In a formal mining approach that is the Heuristics Miner algorithm these three basic relations that is A$\rightarrow_W$ B, A $\neq_W$ B, or A $\|_W$ B are directly used for the construction of a Petri net. The formal approach presupposes perfect information:

- The log must be complete that is, if an activity can follow another activity directly, the log should contain an example of this behavior and
- There is no noise in the log that is, everything is registered in the log is correct. Furthermore, the α-algorithm does not consider the frequency of traces in the log. However, in practical situations logs are rarely complete and /or noise free. Especially the differentiation between errors, low frequent activities, low frequent activity sequences, and exceptions is problematic. Therefore, in practice, it becomes more difficult to decide if between two activities say A and B, one of the three derived relations that is, A$\rightarrow_W$ B, A $\neq_W$ B, or A $\|_W$ B holds. For instance, the dependency relation used in the α-algorithm A$\rightarrow_W$ B only holds if and only if in the log there is a trace in which A is directly followed by B that is, the relation A $>_W$ B holds and there is no trace in which B is directly followed by A that is not B $>_W$ A. However, in a noisy situation one erroneous example can completely these up the derivation of a right conclusion. Even if we have thousands of log traces in which A is directly followed by B, then one B $>_W$ A, this example based on an incorrect registration, will prevent a correct conclusion. As noted before, frequency information is not used in the formal approach. For this reason in the Heuristics Miner we use techniques which are less sensitive to noise and the incompleteness of logs.

The process of mining from original process model to mined process model shown in the figure 3. Hence, normally the mining algorithm is used to generate a new fine tuned or mined model. In figure 3, we have used the Heuristics Miner algorithm to convert from original process model to mined process model. The intermediate work between the original and mined model is the main picture of process mining. Therefore the original model has the data in the form of data base file format, hence this data file is used to convert event logs the these logs are converted into mined process model with the help of control flow mining algorithm that is Heuristics Miner algorithm.

Table 2. Example of Event Log-2

case ID	Activity ID	Originator	Timestamp
case 1	activity A	Saran	09-11-2010:15.01
case 2	activity A	Saran	09-11-2010:15.12
case 3	activity A	Siva	09-11-2010:16.03
case 3	activity B	Meera	09-11-2010:16.07
case 1	activity B	Murugan	09-11-2010:18.25
case 1	activity C	Saran	10-11-2010:19.23
case 2	activity C	Murugan	10-11-2010:10.34
case 4	activity A	Siva	10-11-2010:10.35
case 2	activity B	Saran	10-11-2010:12.34
case 2	activity D	Balaji	10-11-2010:12.50
case 5	activity A	Siva	10-11-2010:13.05
case 4	activity C	Meera	11-11-2010:10.12
case 1	activity D	Balaji	11-11-2010:10.14
case 3	activity C	Siva	11-11-2010:10.44
case 3	activity D	Balaji	11-11-2010:11.03
case 4	activity B	Siva	11-11-2010:11.18
case 5	activity E	Shankar	11-11-2010:12.22
case 5	activity D	Shankar	11-11-2010:14.34
case 4	activity D	Balaji	11-11-2010:15.56

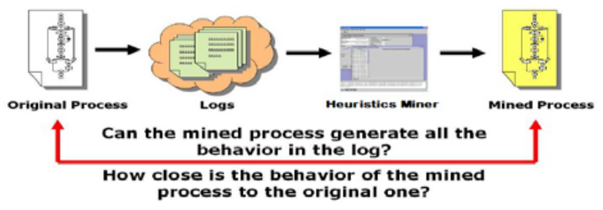
Original Process
Logs
Heuristics Miner
Mined Process
Can the mined process generate all the behavior in the log?
How close is the behavior of the mined process to the original one?

Fig 3. Overview of Process Mining using Heuristics Miner Algorithm. (Source: www.processmining.org)

4. CONCLUSION

In this paper, we have focused on the applicability of process mining in the field of Business Process System. The process mining is the process of generating new process model for the benefit of business processing system. Therefore the any process mining algorithm can do the same. But a special algorithm, called Heuristics Miner is used to generate a new model without noise that is with exception, frequent failure or partial failure, life time of the model, etc. Hence, Heuristics Miner is a most practical applicable mining algorithm that can deal with noise, and can be used to express the main behavior that is not all details and exceptions, registered in an event logs. Normally the business process system has a complex process system, which deals about many cases or audit trail entries and various event logs. Therefore this paper clearly discussed about the role of Heuristics Miner algorithm in the field of Business process system with an web page example also the Heuristics Miner algorithm is compared with the other mining algorithm such as α-algorithm, etc,. Hence, we conclude that a complex information system can be analyzed with the help of control flow process mining algorithm such as Heuristics Miner algorithm can able to discover all the common control flow structures and is robust to noisy logs at once.

5. REFERENCES

[1] M. Dumas, W.M.P. van der Aalst, and A.H. ter Hofstede. Process-Aware Information Systems: Bridging People and Software Through Process Technology, John Wiley & Sons Inc, 2005.

[2] W.M.P. van der Aalst and K.M. van Hee, Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA, 2002.

[3] Workflow Management Coalition. WFMC Home Page, http://www.wfmc.org.

[4] W.M.P. van der Aalst and A.J.M.M. Weijters. Process Mining, Special Issue of Computers in Industry. Elsevier Science Publishers, Amsterdam, Vol. 53, 2004.

[5] van der Aalst, W., Weijters, A., and Maruster, L. Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, Vol 16, No. 9, pp. 1128–1142, 2004.

[6] van der Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., and Weijters, A. Workflow Mining: A survey of Issues and Approaches. IEEE Journal of the Data and Knowledge Engineering, Vol . 47, Issue 2, pp. 237-267, 2003.

[7] J.E. Cook, Z. Du, C. Liu, and A.L.Wolf. Discovering Models of Behavior for Concurrent Workflows. Computers in Industry, Vol. 53, Issue 3, pp. 297-319, 2004.

[8] J. Herbst and D. Karagiannis. Workflow Mining with InWoLvE. Computers in Industry, Vol. 53, No.3, pp. 245-264, 2004.

[9] G. Schimm. Mining Exact Models of Concurrent Workflows. Computers in Industry, Vol. 53, No. 3, pp. 265-281, 2004.

[10] L. Wen, J. Wang, and J. Sun. Detecting Implicit Dependencies Between Tasks from Event Logs. In Xiaofang Zhou, Jianzhong Li, Heng Tao Shen, Masaru Kitsuregawa, and Yanchun Zhang, editors, APWeb, volume 3841 of Lecture Notes in Computer Science, pages pp. 591-603, Springer, 2006.

[11] Chengying Mao. Control Flow Complexity Metrics for Petri Net based Web Service Composition, Nanchang, Published in the Journal of Software, China, Vol. 5, No.11, November 2010.

[12] van der Aalst, W., van Dongen, B., Gunther,¨ C., Mans, R., de Medeiros, A. A., Rozinat, A., Rubin, V., Song, M., Verbeek, H., and Weijters, A. ProM 4.0: Comprehensive Support for Real Process Analysis. In Kleijn, J. and Yakovlev, A., editors, Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007), Lecture Notes in Computer Science, Published by Springer-Verlag, Berlin, Vol 4546, pp. 484-494, 2007.

Saravanan. M.S received B.Sc degree in computer science from Madras University in 1996, the MCA degree from Bharathidasan University in 2001, the M.Phil degree from Madurai Kamaraj University in 2004, M.Tech degree from IASE University in 2005. And now pursuing PhD degree in Bharathiar University. His current research interests include Process Mining, Business Process modeling, Workflow management systems and Exception handling etc. He is an Assistant professor in the Department of Information Technology in VEL TECH Dr, RR & Dr. SR Technical University, Avadi, Chennai, India. M.S. Saravanan has published eight international publications and presented ten research papers in international and national conferences, having 11 years of teaching experience in various institutions in India.

Rama Sree. R.J received M.S degree in computer science from BITS Pilani University in 1996 and PhD degree in S.P. Mahila University, Tirupati. She is a Reader in Department of Computer Science in Rashtriya Sanskrit University, Tirupati. Dr. Rama Sree has published three books and sixteen international publications and ten national publications, having 17 years of teaching experience.