AMELIORATION OF BUG TRIAGE PROCESS IN OPEN SOURCE SYSTEMS

A Thesis

submitted to Pondicherry University in partial fulfillment of the requirements for the award of the Degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

V. AKILA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PONDICHERRYENGINEERINGCOLLEGE

PUDUCHERRY – 605 014

INDIA

MAY 2015

Dr. G. ZAYARAZ, M.E., Ph.D.
Associate Dean (Student Affairs) and Professor (CSE)
Pondicherry Engineering College
(A Govt. of Puducherry Institution)
Puducherry–605014.

CERTIFICATE

Certified that this thesis entitled "**AMELIORATION OF BUG TRIAGE PROCESS IN OPEN SOURCE SYSTEMS**" submitted for the award of the degree of **DOCTOR OF PHILOSOPHY** in **COMPUTER SCIENCE AND ENGINEERING** of the Pondicherry University, Puducherry is a record of original research work carried out by **Ms. V. AKILA** during the period of study under my supervision and that the thesis has not previously formed the basis for the award to the candidate to any Degree, Diploma, Associateship, Fellowship or other similar titles. This thesis represents independent work on the part of the candidate.

Place : Puducherry	**Dr. G. ZAYARAZ**
Date :	**Supervisor**

DECLARATION

I hereby declare that this thesis entitled **“AMELIORATION OF BUG TRIAGE PROCESS IN OPEN SOURCE SYSTEMS”** submitted to the Pondicherry University, Puducherry, India for the award of the degree of **DOCTOR OF PHILOSOPHY** in **COMPUTER SCIENCE ENGINEERING** is a record of bonafide research work done by me under the supervision of **Prof. G. ZAYARAZ,** Associate Dean (Students Affairs) and Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry and that the work has not been submitted either in whole or in part for any other degree or at any other university.

Counter Signed Signature

Dr. G. ZAYARAZ **V. AKILA**

ACKNOWLEDGEMENT

First and foremost, my deepest gratitude is to my supervisor, **Dr. G. Zayaraz**, Associate Dean (Student Affairs) and Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry. His continuous encouragement and support helped me complete my research.

I register my gratitude to my Doctoral Committee members, **Dr. N. P Gopalan**, Professor, Department of Computer Applications, National Institute of Technology, Tiruchirappali and **Dr. K. Vivekanandan**, Dean (Student Affairs) and Professor, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry for their guidance and helpful suggestions.

I am grateful to **Dr. D. Govindarajulu**, Principal, Pondicherry Engineering College, Puducherry and **Dr. V. Prithiviraj**, Former Principal, Pondicherry Engineering College, Puducherry for their support to pursue the research work.

I thank **Dr. D. Loganathan**, Professor and Head, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry and **Dr. N. Sreenath**, Professor and former Head, Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry for the academic, technical and logistical support provided by the Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry.

I register my special thanks to all the faculty members and non-teaching members of Department of Computer Science and Engineering and Department of Information Technology, Pondicherry Engineering College, Puducherry for their encouragement and support.I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture.

V. AKILA

ABSTRACT

Critical and objective decisions made during the corrective maintenance of proprietary software rely on the innate knowledge, experience, judgment and wisdom of the software practitioners. The decision making process during corrective maintenance pertains to evaluating the validity of a bug, duplicate bug detection, bug triage, bug reassignment, patch testing, etc. The retention of tacit knowledge of the software practitioners has been proven to be complicated for Open Source Software Maintenance. Bug management is one of the core constituent of Open Source Software Maintenance. Bug Triage is an integral element of bug management. Bug Triage is concerned about evaluating the validity of a reported bug, assigning severity, priority and the most critical exercise is to determine an appropriate developer to solve the bug. Bug Triage when performed manually is susceptible to error and delay. This principal problem instigates the research work to develop techniques that provide automated support for Bug Triage and thereby, aid the Open Source Software Maintenance.

This research work presents techniques that automate and thereby ameliorate the human driven bug triage process by eliciting information from the bug repository. The Open Source Software development process generates massive data that are stored in software repositories. This data embeds in it the patterns, practices, behavior and social context of the developers involved in the Open Source Software.

The amelioration of the bug triage process is achieved by routing the new bug report through an optimal referral chain of developers who can add value towards solving of the bug. To this end, the research work makes several contributions. To begin with, the bug tossing relation among developers is elicited from the bug repository and the collaboration in terms of bug tossing is modeled as a directed weighted Bug Toss Graph. The Bug Toss Graph is formalized as an Actual Path Model. The Bug Toss Graph is further refined to an Enriched Collaboration Graph by augmenting features that capture the dynamism of relationship among the developers. The next step is to design and administer adaptive techniques based on ant systems to query and learn the referral chain of developers from the progressively evolving Enriched Collaboration Graph. The adaptive techniques gain information about a developer's contribution by passively observing the developers bug tossing activity. The adaptive learning techniques employ probabilistic strategy to select outgoing links for bug report forwarding. Further, to bolster the recommendation metrics of Precision and Recall, additional Path Similarity metric is introduced to evaluate the effectiveness of the retrieved referral chain of developers in terms of position and order. The Path Similarity metric is based on Levenshtein distance. The Path Similarity metric computes the number of edit operations needed to convert the retrieved referral chain of developers to the original path of developers. This computation is performed based on dynamic programming. Finally, to further demonstrate the overarching usefulness of the Ameliorated Bug Triage System, an Holistic Evaluation Framework with Key Performance Indicators (KPI) based on developer's performance is introduced. The set of KPIs inducted are Developer Time Index, Developer Effectiveness Index and Developer Productivity. The KPIs serve as the user metrics to evaluate the Ameliorated Bug Triage System.

Thus, in this research work, the Ameliorated Bug Triage System is developed in a progressive manner. The proposed Ameliorated Bug Triage System is based on an Enriched Collaboration Graph. The adaptive Co-Ant algorithm is utilized as the learning model in the Enriched Collaboration Graph. The existing Bug Triage System is based on the Bug Toss Graph modeled as a Goal Oriented Path model. The Weighted Breadth First Search algorithm is utilized as the learning model in the existing system.

The experimental results prove that the developed Ameliorated Bug Triage System shows significant improvement over the existing Bug Triage System in terms of the three conventional metrics viz., Path Length, Precision and Recall. In addition to the conventional metrics, a new metric - Path Similarity is proposed. The Ameliorated Bug Triage System outperforms the existing system in terms of the proposed Path Similarity metric as well. Further, the developed Holistic Evaluation Framework is used to validate the proposed Ameliorated Bug Triage System using the three KPIs viz., Developer Time Index, Developer Effectiveness Index and Developer Productivity. The outcome of the evaluation using the Holistic Evaluation

Framework further proves that the proposed Ameliorated Bug Triage System is superior to the existing Bug Triage System. ANOVA is used to perform the statistical analysis of experimental results. The ANOVA based statistical analysis concludes that the proposed Ameliorated Bug Triage System performs significantly better than the existing system. Thus, the Ameliorated Bug Triage System aids to automate the Bug Triage process.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
6.34	Co-Ant: Weighted Average Precision	128
6.35	Co-Ant: Recall for Study 1	129
6.36	Co-Ant: Recall for Study 2	129
6.37	Co-Ant: Recall for Study 3	130
6.38	Co-Ant: Weighted Average Recall	131
6.39	Co-Ant: Path Similarity for Study 1	132
6.40	Co-Ant: Path Similarity for Study 2	132
6.41	Co-Ant: Path Similarity for Study 3	133
6.42	Co-Ant: Weighted Average Path Similarity	134
7.1	Developer Contribution Distribution	137
7.2	Developer Time Distribution w.r.t Bug field	138
7.3	Developer Distribution w.r.t Time	138
7.4	Developer Distribution w.r.t Bug Resolution Time	139
7.5	Dependency in the Key Performance Indicators	140
7.6	A Holistic Evaluation Framework with Developer Performance	143
7.7	Developer Time Index	144
7.8	Developer Effectiveness Index	144
7.9	Developer Productivity	145

LIST OF TABLES

LIST OF ABBREVIATIONS

ACRONYMS	**ABBREVIATIONS**
AP	Actual Path Model
AP-WBFS	Actual Path Model Based on Weighted Breadth First Search
BTL	Bug Tagging Language
BT-ANT	Bug Triage Based on Ant System
Bi-Objective	Bug Triage based on Bi-Objective Optimization
Co-Ant	Cooperative Ant Algorithm
DETROM	DEveloper Recommendation based on TOpic Models
Drex	Developer Recommendation with k-nearest-neighbor search and Expertise ranking
DECOBA	DEvelopersCOmmunities in Bug Assignment
ES	Enterprise System
GP	Goal Oriented Path Model
GP-WBFS	Goal Oriented Path Model Based on Weighted Breadth First Search Based
LDA	Latent Dirichlet Allocation
LSI	Latent Semantic Indexing
MSTR	Mean Steps to Resolve
MDN	Multiple software Developer Network
OSS	Open Source Software
ONM	Optimized Network Model
P	Path Length
PS	Path Similarity
PR	Precision
R	Recall
SVM	Support Vector Machine
TRAM	TRiaging Approach using bug reports Metadata
VMS	Variable order Multiple active state Search
VSM	Vector State Model
WBFS	Weighted Breadth First Search

CHAPTER - 1

INTRODUCTION

1.1 PREAMBLE

This chapter introduces the concept of Bug Triage. It also presents the broader context of Bug Triage in the milieu of Open Source Software. The key differences between Open Source Software and Proprietary Software are highlighted. The relevance of Software repositories in Open Source Software Maintenance is presented. Finally, the Bug Triage and bug tossing are introduced.

1.2 OPEN SOURCE SOFTWARE

Open Source Software (OSS) is commercial software where full access to the code for viewing, modification and redistribution is granted to all the users by agreeing to a free of cost license. Over the years, there is a deep proliferation of OSS in every walk of life in terms of DNS servers, web proxy caches, Apache web server, etc. Gartner has reported that OSS will be part of 99% of the mission-critical systems by the year 2016. Further, to reiterate the wide adoption of OSS, even the stock exchanges of the New York, London and Tokyo are based on Linux [1].

OSS systems have graduated from being an alternative to a necessity in today's environment. The foremost trigger for OSS adoption is the global recession in the year 2009. The recession in the year 2009 resulted in tightened budgets, which has forced companies to look into the expenditure of the software procurements. With tighter budgets, there is a growing need for the experienced developer with strong mathematical leanings and prove-it-as-you-code model of software development[2].

The key player of OSS environment is the open source community. The open source community comprises a set of developers who assist in development, implementation, deployment and maintenance of the software. The developers collaborate in a decentralized and distributed manner. The software developers participate voluntarily and are connected via the Internet [3].

OSS banks on a wiki-like coding environment where the developers are more often scrutinized by their peers. The uniqueness of OSS is that, there is a blurring of boundaries between the software users and the software developers. The motivation for contribution of code mostly is peer respect, ego gratification and future prospects. These factors contribute in building high quality and robust software that are secure, flexible and stable. The other benefit of OSS is the avoidance of vendor lock-in.

Proprietary software scores over OSS in documentation, mature ease of use, accountability and vendor handholding, even after deployment.Bugs eventually arise in deployed software, irrespective of the software being proprietary or open. There is continued vendor support for the procured proprietary software in terms of upgrades, patches, etc. This vendor support is guaranteed in the case of proprietary software by Service Level Agreements.This guaranteed support, after deployment,is amiss in OSS. The maintenance support to deployed software in OSS is offered in terms of mailing list, live chat, issue tracking system and web forums. The very nature of OSS is voluntary, collaborative and geographically distributed. The challenge to OSS is there is no guaranteed support.

The solution to this challenge lies in the software repositories. While there are very many obvious advantages of OSS, the not-so-obvious benefit of OSS is the availability of software repositories in the public domain. These repositories hold in them information pertaining to the OSS development. The one thing that is possible in this scenario is to exploit the software repositories that are the fringe benefit of OSS development model. With the information in the software repositories though accountability of support is a distant possibility, accelerated support is a reality.

1.3 SOFTWARE REPOSITORIES

Software repositories are repositories referring to artifacts that are products of software evolution. These data refer to the vast amount of information that is accumulated during the period of software evolution [4].Software repositories were primarily used for record keeping purposes. Currently, these repositories are used to derive actionable knowledge from the static records stored.These repositories were unavailable in the public domain when software development was confined within enterprises. Now, with the landscape changing with the proliferation of OSS, access to these repositories is possible.

Software practitioners employ strategies based on instinct and past experience. For example, managers use experiences from past project to allocate resources, Testers prioritize the testing features based on past experience [5]. In an enterprise system, there is minimal attrition of the software developers. As a result of this, the tacit knowledge of the developer can be tapped into. This tacit knowledge is not readily available in OSS where the software developers are in a state of flux. The software repositories bridge this gap in providing implicit knowledge and past experience in the OSS context.

The few thrust areas, where software repositories are currently used are:(i) source code mining, (ii) study of issues and patches, (iii) bug prediction, (iv) communication artifacts mining, (v) text mining and (vi) developer modeling [6]. The following section throws light on the popular software repositories.The popular repositories are versioning repositories, email repositories and bug repositories.

1.3.1 Version Control Repository

Revision control refers to tracking of changes to source code. Maintaining the documents and configuration files are part of version control. Often, a version control system consists of source code with monitored access. The changes made to the code along with the accounting information of who made the change, date, time why the change was made is maintained. This version control is more important in a distributed, collaborative and virtual environment which is the signature of OSS systems. Revision control is important in comparing the performance of the current version with any of the previous version of software. It is also useful in tracking all the modifications made to the source code. These modifications are made regularly in OSS. Revision control systems assist in monitoring the modifications carried on a source code and to locate contaminations if any.

Revision control plays an important role in finding the role of developers in the evolution of the OSS system. It helps to identify the number of authors for a particular source code and the behaviors of the developers when developing the code [7].

1.3.2 E-mail Repository

E-mail repositories contain artifacts that comprise design choices and issues encountered in software development. The mailing list is used for technical discussions, reporting of bugs, announcement of releases etc. This archive can serve as part of the institutional memory. The analysis pertaing to e-mail repository are,selection of e-mails to be used, data pre-processing techniques to be employed, applicability of techniques applied in small e-mail collections to large collections, etc[8],[9].

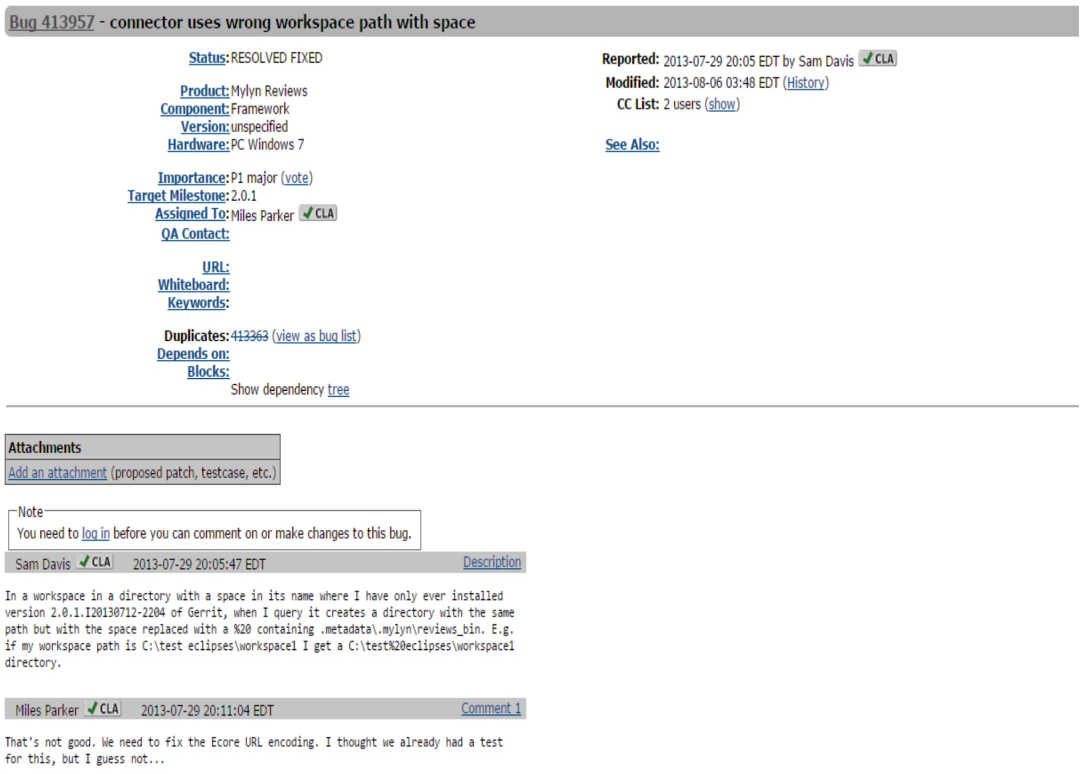
1.3.3 Bug Repository

Bug repositories are where the users of OSS are encouraged to report the bugs. The OSS system uses an open bug repository like www.bugzilla.org or www.jira.org. The use of the open bug repository propels the identification of more problems in the software with relative ease and encourages more developers to engage in the discussions and deliberations on a bug.

1.3.3.1 Anatomy of a Bug Report

A bug report consists of pre-defined fields, free-form text, attachments and dependencies. The pre-defined field consists of fields such as BugID, the reporter of the bug, the fields product, the component to which the bug belongs to, the version, the operating system, the priority and the severity. The fields like assignee, status of the bug etc are subject to frequent change [10].

The free form text is the textual summary part of the bug report. It comprises of the one line summary of the bug and the detailed description of the effects of the bug. The comments parts details the comments made by the public as well as the developers during the course of discussion over the bug. Additional attachments like screen shot of the bug and the activity log of the bug are also provided.



Bug 413957 - connector uses wrong workspace path with space

Status:	RESOLVED FIXED
Product:	Mylyn Reviews
Component:	Framework
Version:	unspecified
Hardware:	PC Windows 7
Importance:	P1 major (vote)
Target Milestone:	2.0.1
Assigned To:	Miles Parker ✓ CLA
QA Contact:	
URL:	
Whiteboard:	
Keywords:	
Duplicates:	~~413363~~ (view as bug list)
Depends on:	
Blocks:	
	Show dependency tree

Reported:	2013-07-29 20:05 EDT by Sam Davis ✓ CLA
Modified:	2013-08-06 03:48 EDT (History)
CC List:	2 users (show)
See Also:	

Attachments
Add an attachment (proposed patch, testcase, etc.)

Note
You need to log in before you can comment on or make changes to this bug.

Sam Davis ✓ CLA 2013-07-29 20:05:47 EDT Description

```
In a workspace in a directory with a space in its name where I have only ever installed
version 2.0.1.I20130712-2204 of Gerrit, when I query it creates a directory with the same
path but with the space replaced with a %20 containing .metadata\.mylyn\reviews_bin. E.g.
if my workspace path is C:\test eclipses\workspace1 I get a C:\test%20eclipses\workspace1
directory.
```

Miles Parker ✓ CLA 2013-07-29 20:11:04 EDT Comment 1

```
That's not good. We need to fix the Ecore URL encoding. I thought we already had a test
for this, but I guess not...
```

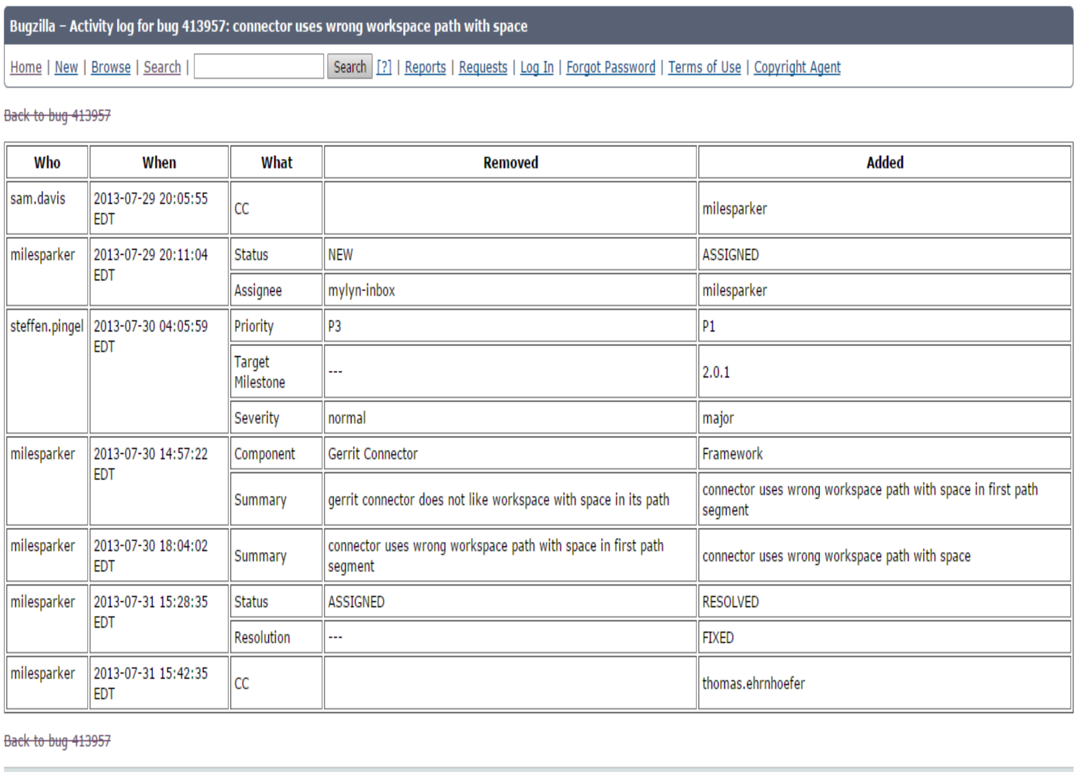
Figure 1.1: Bug Report of Bug ID – 413917

A sample bug report from www.bugzilla.org is presented in the Figure 1.1. The bug report ID is Bug ID - 413917. The status is 'Resolved' and

‘Fixed’. The component to which the bug belongs to is ‘Framework’ and it is assigned to a developer named Miles Parker. The bug was reported on 29th July 2013 by Sam Davis. The comments posted on the bug are displayed on the lower part of the bug report. The hyperlink ‘History’ links the activity log of that particular bug.

1.3.3.2 Activity Log

The activity log of the Bug ID-413917 is depicted in the Figure 1.2. The activity log provides a gist of the history of the bug. From the activity log, it can be inferred that the bug was reported by Sam Davis on 29th July 2013. On the same day, it was assigned to Miles Parker. Following this, the status of the bug has been modified from ‘New’ to ‘Assigned’. The developer Miles Parker has assigned it to Steffen Pingel on 30th July 2013. The developer Steffen Pingel has modified the priority of the bug from ‘P3’ to ‘P1’, added the Target Milestone field and the severity of the bug has been modified to ‘Major’ from ‘Normal’. The bug is tossed back to Miles Parker on the same day.



Bugzilla – Activity log for bug 413957: connector uses wrong workspace path with space

Home | New | Browse | Search | Search [?] | Reports | Requests | Log In | Forgot Password | Terms of Use | Copyright Agent

~~Back to bug 413957~~

Who	When	What	Removed	Added
sam.davis	2013-07-29 20:05:55 EDT	CC		milesparker
milesparker	2013-07-29 20:11:04 EDT	Status	NEW	ASSIGNED
		Assignee	mylyn-inbox	milesparker
steffen.pingel	2013-07-30 04:05:59 EDT	Priority	P3	P1
		Target Milestone	---	2.0.1
		Severity	normal	major
milesparker	2013-07-30 14:57:22 EDT	Component	Gerrit Connector	Framework
		Summary	gerrit connector does not like workspace with space in its path	connector uses wrong workspace path with space in first path segment
milesparker	2013-07-30 18:04:02 EDT	Summary	connector uses wrong workspace path with space in first path segment	connector uses wrong workspace path with space
milesparker	2013-07-31 15:28:35 EDT	Status	ASSIGNED	RESOLVED
		Resolution	---	FIXED
milesparker	2013-07-31 15:42:35 EDT	CC		thomas.ehrnhoefer

~~Back to bug 413957~~

Figure 1.2: Activity Log of BugID - 413917

The developer Miles Parker now has changed the component field to 'Framework' from 'Gerrit Connecter' and modified the summary field of the bug. The following day i.e. 31st July 2013 the status of the bug is changed to 'Resolved' from 'Assigned' and the resolution is made as 'Fixed' by Miles Parker. This log shows that the bug may not be solved by the first assignee itself, it may need to be changed hands in order to be solved. This changing of developers is called as tossing of the bug. The tossing of the bug contributes to the time taken to resolve the bug.

For example, the tossing activity underwent by Bug ID - 413917 is {Sam Davis, Mike Parker, Steffen Pingel,Mike Parker}. Fortunately, the bug has been resolved in two days which is not the case always.

The tossing path of another bug with Bug ID-320934 is {cgold, markus_keller, cgold, markus_keller, cgold, spektom, markus_keller, pwebster}. The bug was reported on 2010-07-29 and finally, resolved on 2013-08-14.

The tossing path of yet another bug of Bug ID - 402560 is {pascal.rapicault, peter, martin.oberhuber, markus_keller, Joerg.Thoennes, pascal, hamdan.msheik, pascal.rapicault}. The bug was reported on 2013-03-06 and finally, resolved on 2013-08-27. It has been reported that 90% of the bugs gets tossed at least once. It takes 40 days to assign a bug to a developer and another 90 days to reassign a bug [11], [12].

1.4 BUG TRIAGING

Bug Triaging process addresses the validation and then, the distribution of bug reports to the most appropriate developers in order to resolve the bugs. The conceptual view of the Bug Triaging process is given in the Figure 1.3.Once the bug report is submitted to the bug repository, the triager has to evaluate, prioritize, categorize and assign the bug to software developer. The bugs may be categorized according to the content in the summary or by the component to which the given bugs belong to. The bug is assigned to a software developer based on the expertise of the software developer.

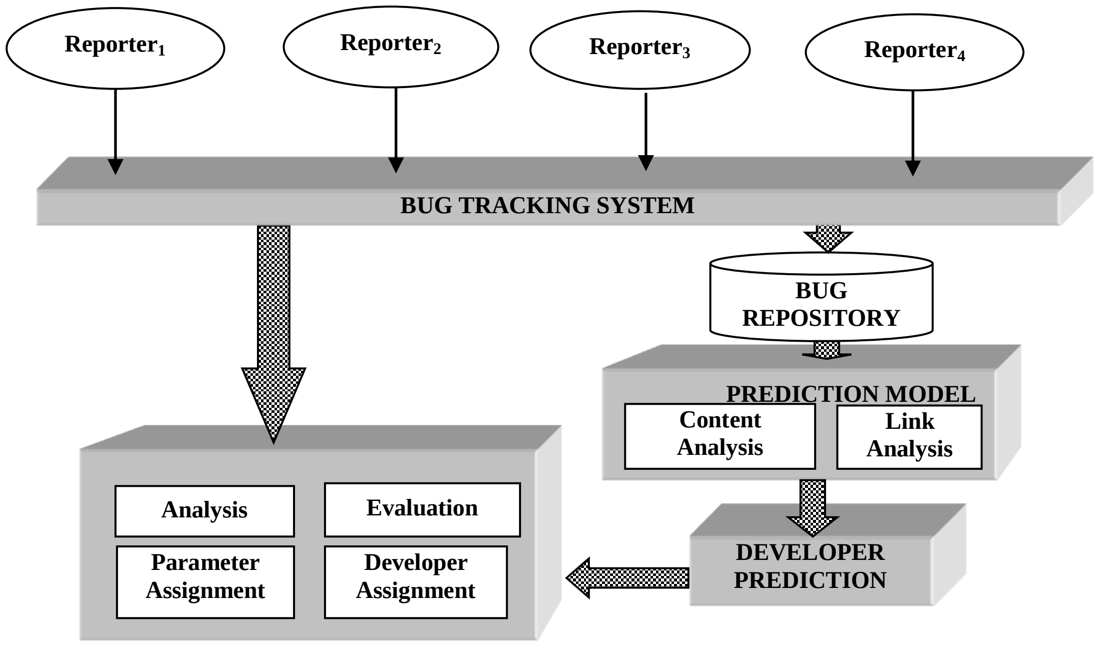


Figure 1.3: Conceptual View of a Bug Triage System

The bug assigned to a software developer may get reassigned if the concerned software developer is not able to solve the bug. This process of reassignment is termed as bug tossing. After a few bug tosses, the bug may get resolved. The bug tossing relation among the software developers can be modeled as a Bug Toss Graph. In the Bug Toss Graph, the nodes represent the software developers and the arcs represent the tossing activity. The goal of the triager is to find a software developer in a manner such that the number of tosses a bug undergoes before being resolved is minimized. Each toss takes up some lead time for the software developer to familiarize himself with the bug, thereby contributing to the bug fix time. The goal of any Automated Bug Triage System is to minimize the bug resolution time by minimizing the number of unwarranted reassignments in the life time of the bug.

1.5 MOTIVATION

OSS development is here to stay. It is all the more relevant in the current period of global economic slump. OSS has moved on from being a low cost alternative to the in house software development, to a necessity. Examples of OSS in every walk of life include Android, Chrome, Wordpress blogging platform, Openoffice, 7-Zip, PDFCreator etc. The list is endless.

While OSS may have no proprietary rights and there is no cost incurred while acquiring an OSS per say, there are hidden costs when using OSS. The cost incurred via adopting OSS is due to the limited handholding to the service provided by the OSS after its deployment, limited warranty for the software and lack of documentation. Or, in other words, maintaining an OSS, post its deployment is tricky. This is because: OSS development does not conform to the traditional software development practices. It is independent and more agile in nature. This is due to the fact that the participation of the developers in OSS development is voluntary. Since the participation is voluntary, the developers can shift from one project to another, there may be expertise evolution, or more so the developers may drop out of disinterest.

Bug management is a central component of the software maintenance of the OSS. Bug management comprises of the following three activities: (i) Bug Triaging, (ii) bug assignment to the software developer for solution and (iii) solving of the bug. Software maintenance expenditure is about 50% of the overall expenditure of the software project. In OSS development, the expenditure translates to time. Bug Triaging comprises of checking for validity of the bug, assigning priority, severity and assigning the bug to a correct software developer. Manual Bug Triaging is time consuming and fault prone [11], [12], [13]. So, it is imperative that automated support is essential for Bug Triaging. It is postulated that automated support for Bug Triaging will accelerate the bug management process which in turn will percolate to the OSS maintenance and thereby slash down the Total Cost of Ownership of the OSS.

1.6 THESIS ORGANIZATION

The rest of this thesis is organized as follows. Chapter 2 presents a pervasive analysis of the state of the art techniques of Bug Triage in OSS. Besides that, it presents an analysis of Ticket Resolution techniques in Enterprise Systems. The shortcomings in the state of the art techniques in Bug Triaging are identified.

Chapter 3 records the Problem Statement, Research Objectives of the thesis. Further, it proceeds to present the high level conceptual view of the Ameliorated Bug Triage System. The research methodology, the data preparation and metrics used for evaluation are documented in this chapter.

Chapter 4 documents the analysis of techniques that were considered for serving as Path Similarity metric. The techniques are evaluated by computing the Correlation Coefficient values.

Chapter 5catalogues the comparison of Actual Path Model with the Goal Oriented Path Model. The enhanced bug triage algorithm based on Bi-Objective Optimization is presented in the chapter.

Chapter 6 records the formal model for the Enriched Collaboration Graph. Moreover, it offers the Adaptive Techniques employed on the Collaboration Graph. The adaptive techniques are based on the self organizing and adaptive behavior of ant systems.

Chapter 7 archives the Holistic Evaluation Framework of Bug Triage System by integrating the developer performance in the evaluation. It inducts the three Key Performance Indicators based on the developer’s performance.

Chapter 8 presents a comprehensive conclusion of the thesis. It also provides an extensive indication of the future directions of the research.

CHAPTER - 2

REVIEW OF THE LITERATURE

2.1 PREAMBLE

This chapter offers a literature appraisal of the existing research approaches of the ongoing research in the field of Bug Triaging as well as its counterpart in the Ticket Resolution techniques in Enterprise System (ES). The survey has analyzed all the recent leading research publications. Taxonomy of the techniques that are adhered in the field of Bug Triage is derived. The gaps in the literature are highlighted.

2.2 STATE OF THE ART BUG TRIAGE TECHNIQUES IN OSS

The survey reveals that the techniques employed for Bug Triaging use the textual summary in the bug report and use the text mining approach on them. The other set of techniques exploits the tossing relation that the developers indulge in.

It can be noticed from the taxonomy depicted in the Figure 2.1 that the majority of techniques fall under content based techniques which use the textual summary from the bug report or link based which use the tossing relation. The content based techniques can further be classified as activity based or location based. In activity based techniques, the text based approaches are used over the summary field of the bug report, whereas in the location based techniques, the bug has to be localized. Bug localization pertains to locating the source code module to which the bug belongs to. Following bug localization, the textual contents linked with the source code module such as comment, versioning information serve as fodder to the location based techniques. The textual content, be it activity based or location based, are predominantly used with techniques based on machine learning, soft computing and information retrieval.

The link based techniques use the links that exist among the developers. The link may be in terms of bug reassignments i.e. tosses or comments made by the developers on the bug reports. The graph that is derived from the links can either be a Goal Oriented Path Model (GP)or Actual Path Model (AP).

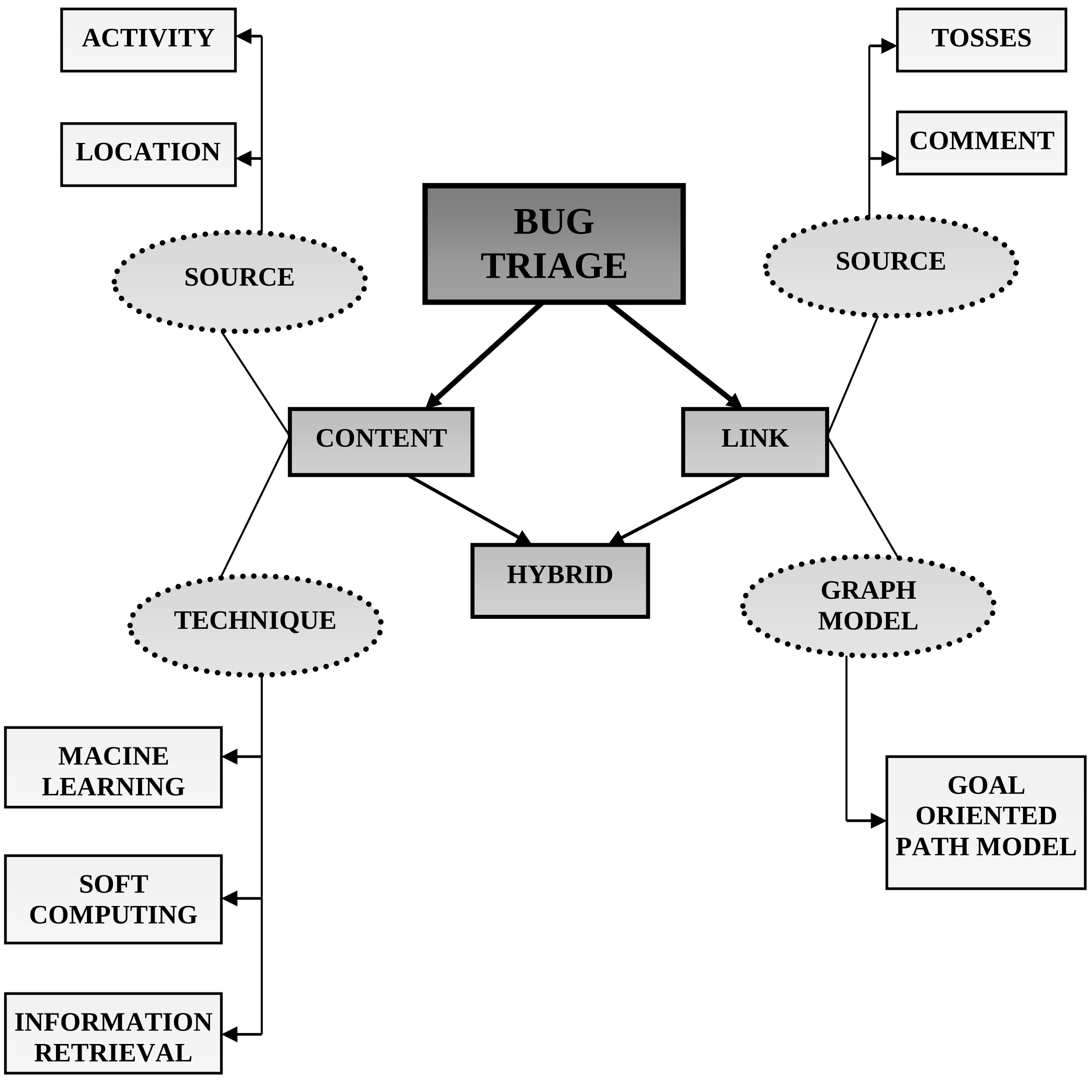


Figure 2.1: Taxonomy of the Techniques for Bug Triage in OSS

The state of the art existing Bug Triaging techniques have been analyzed along these dimensions:

i) Data

ii) Algorithm

iii) Tossing Graph Model

iv) Evaluation Parameters

2.2.1 Content Based Techniques

2.2.1.1 Activity Based Techniques

In activity based techniques for Bug Triaging, the bug reports similar to the new incoming bug report are identified. From the similar bug reports, the potential software developers are inferred. The similar bug reports are identified by machine learning, soft computing or information retrieval methods.

Machine Learning

Machine learning treats the textual summary in the bug report as instances. Instances may also have a label that indicates the category, or class, to which they belong. A supervised machine learning algorithm takes as input a set of instances with labels and produces a classifier as output. The classifier can then be used to assign a label to an unknown instance. There are a considerable number of techniques that are based on machine learning for Bug Triage. They are listed below.

Reducing efforts in making developer recommendations is done in a three pronged manner [14]based on machine learning techniques. There are three kinds of recommenders viz., (i) software developer oriented recommender, (ii) component recommender and (iii) interest recommender. The software developer oriented recommender suggests the developer who may fix the bug, the component recommender localizes the component the bug belongs to and the interest recommender suggests developer who may be interested in resolving the bug. The algorithms explored are Naïve Bayes, Support Vector Machine (SVM), C4.5, Expectation Maximization, Conjunctive Rules and Nearest Neighbor. The parameters that were used for evaluation are Precision, Recall and F-Measure. SVM and Naïve Bayes performed better with regard to the parameter - Precision to make one recommendation, but, when more than two recommendations had to be done, SVM performed better. The bug reports are pre-processed and only the summary and description fields are utilized.

Machine learning algorithms are used to model the expertise of the developers [15]. This model is then utilized in making recommendations. The recommendation is based on the following eight types of information: the textual description of the bug, the component, the operating system, the hardware, the version, the developer associated with the source code, the work load of the developer and a list of active developers. SVM algorithm was used as the machine learning algorithm. The Precision of the system was 86% for the data from Eclipse project.

TRiaging Approach using bug reports Metadata (TRAM) system uses the discriminating terms in the bug report including reporter of the bug and the component to triage a bug [16]. The Naïve Bayes classifier is used to build the predictive model. The chi-square method is used for feature selection during data preparation. The system was evaluated using the parameters: (i) Precision (ii) Recall and (iii) F-Measure.

The developer preferably works at most only on two to three components [17]. This information is exploited with unsupervised learning techniques to make developer recommendations. It is observed that the filtering module of supervised learning techniques greatly affect the diversity of the developers recommended. The developer’s expertise is leveraged using a Vector Space Model (VSM). The cosine similarity metric is used to compute the similarity between any two developers. The system is evaluated using Accuracy and Diversity metrics. Diversity in recommendation is used to break monotony in developer recommendation. Data preparation considering efficient feature selection and instance selection are used to reduce the training set for Bug Triage [18]. The feature selection is done based on chi-squared test. The instance selection is based on iterative case filter algorithm. The system is validated using the parameters Accuracy, Precision and Recall. The Bug Triaging task is visualized as a text classification problem [19]. Specifically, it is treated as a multi-class, single label classification problem where the software developer corresponds to a class and each bug report is assigned to a class using Naïve Bayes classification. The system achieves 30% Accuracy. A theoretical framework based on SVM to recommend software developers considering preference elicitation and load balancing for triaging of bugs is proposed [20].

Social Network Metrics are integrated into the developer recommendation in Developer Recommendation with k-nearest-neighbor search and Expertise ranking (DREX) [21]. The system makes contribution in two aspects. First, the similar bug reports are searched with respect to the incoming bug report with K-Nearest-Neighbor search. Then, the developers from the similar bugs are ranked according to their contribution in the discussion of the bugs. This contribution is measured in terms of social network metrics such as in-degree, out-degree, page rank, betweenness and closeness. The system is evaluated based on the parameters: Precision and Recall.

A semi automated approach based on machine learning is presented [22]. The novelty of the approach is that it identifies trace between the bug repository and the source code. The approach comprises of four predominant steps :(i) characterizing the bug report(ii) labeling the bug report(iii) choosing reports for the training set and (iv) applying the classifier algorithm. The characterizing of the bug report involves converting the free-form text to a feature vector. The labeling of the bug reports is performed based on heuristics. The label is assigned as per the developer who submitted the last patch or by the developer who changed the status as resolved or the developer who assigned duplicate status, etc. The choosing of bug reports for training is done by filtering approach. The filtering operation is performed by avoiding developers who no longer work in the project and developer with minor contributions. The SVM classifier is used to classify the bug. The system is evaluated with Precision and Recall parameters.

DevRec is a composite method that performs triage with bug report based analysis and software developer based analysis [23]. The bug reports based analysis finds the bug reports that are similar to an incoming bug report based on k-nearest neighbors. The second part consists of determining a software developer's affinity to bug reports in terms of the following parameters: topic, components and product. The Recall@5 and Recall@10 values are computed for performance evaluation. An auction algorithm based Bug Triage is advocated [24]. A Port Stemming algorithm is used to extract the keywords from an incoming bug. The keywords extracted are then compared with the keywords from the history database. A keyword extraction algorithm is used to classify the remaining bug reports. As a final step, the candidate software developers are chosen based on history, preference, experience, credit, efficiency and workload.

The textual summary is once again used with ann-gram based algorithm and approximate string matching to recommend developers [25]. The evaluation metrics are Precision and Weighted Average Recall.This free-form text is converted to a feature vector. The resulting data is then labeled according to the software developer. Summarizing the data in the report using path group and determining the activity levels of the software developers are the two techniques implemented. The accuracy percentage of the recommender is 75%.

For any content based method, the natural language content of the bug report is converted to a bag of words. Term selection is an important aspect of data preparation in bag of words method. Selection of terms affects classification accuracy [26]. The three term selection methods that are examined are based on Logs Odd Ratio score. These methods are compared against Information Gain Methods and Latent Semantic Analysis (LSI). The Bayesian Network Classifier is used for triaging of the bugs in the bug repository. The results are compared based on Precision and Recall. Logs Odd Ratio provided 30% improvement in Precision and 5% improvement in Recall.Data preparation can also be viewed as a dimension reduction problem [27]. The report title and report summary from the bug report are extracted. The extracted text data is then converted to term document matrix using parsing, filtering and term weighting. The dimension of the term document matrix is reduced using feature selection and LSI. The reduced term frequency matrix is used along with seven different classifiers. It has been observed that SVM with LSI produces the best results. The Accuracy, Precision and Recall values of SVM and LSI are 44.4%, 30% and 28% respectively.

The summary of the survey in Bug Triaging based on machine learning in Open Source System is presented in the Table 2.1.

Table 2.1: Summary of Machine Learning Techniques

Paper	**Machine learning**			
	Accuracy	**Precision**	**Recall**	**F-Measure**
[14]	-	✓	✓	-
[15]	-	✓	-	-
[16]	-	✓	✓	✓
[17]	✓	-	-	-
[18]	-	✓	✓	✓
[19]	✓	-	-	-
[21]	-	✓	✓	-
[22]	-	✓	✓	-
[23]	-	-	✓	-
[25]	-	✓	✓	-
[26]	✓	✓	✓	-
[27]	-	✓	✓	✓

Soft Computing

The Bugzie system is based on fuzzy set approach [**28**]. The set of technical terms are collected from the software artifacts. A fuzzy set 'Ct' is used to represent a set of software developers to fix a bug with term 't'. The system has three stages. They are (i) training(ii) recommending and (iii) updating. The system

Bugzie outperforms SVM, Naïve Bayes, C4.5 and Bayesian Network in terms of the Parameters: (i) Prediction(ii) Accuracy and (iii) Time Efficiency.

Information Retrieval

Over specialization is frequently associated with recommender systems. This is handled by content boosted collaborative filtering [29]. The content boosted collaborative filtering is the combination of content based recommendation and collaborative based recommendation. The recommendation problem is reformulated as an optimization problem of accuracy and cost. The software developer profiles are created by a cost aware triage algorithm based on Latent Dirichet Analysis (LDA). A cost profile for each software developer is prepared based on the time taken by the software developer to solve the bug. The performance evaluation was a trade-off between cost in terms of bug fix time and accuracy. The bug fix time efficiency was improved by 30%.

Knowledge management in terms of activity profile for the software developers present in the bug tracking system is reported to enhance the Bug Triage process [30]. The activity profile consists of the user role (review, assign, resolve) and user’s topic association. For an incoming bug, the topic association is derived based on LDA using the title, description and system component. A list of software developers who match the new bug’s topic are then ranked. The ranking is based on the software developers experience and the number of topics the software developers is associated. The new system produces an average hit ratio of 88%.

A prototype for Bug Triage which uses the resolved bug reports as well as the change sets has been developed [31]. Information retrieval method is used to find the bug reports that are similar to an incoming bug report based on cosine similarity. Then, the software developers who provided the change set for the similar bug reports are retrieved. A score that takes into account the number of change set provided by the software developer and the similarity score of the bug report is calculated for every software developer. The software developers are ranked using this score. Further, to provide context for the assigned software developer, the files that were examined while the similar bug was resolved is provided. To foster collaboration, the prototype has been developed as a multi touch table.

Time aware bug assignment is presented in FixTime [32]. The method is based on topic modeling. A log normal regression model is used profile bug resolution time across topics. The system is validated using Prediction Accuracy.

Porch Light is the prototype for a tool for Bug Triaging [33]. Software developers deal with grouped bug reports effectively. The Porch Light provides a tag based grouping of bugs. The grouping may be with respect to the component to which the bug belong or any dependency among the previously resolved bugs etc. A specialized Bug Tagging Language (BTL) is used for creating new tag sets. The predominant tags that are labeled are:

i) Popular – Bugs with more than three comments
ii) Missing details – Bugs with no screen shot or stack trace
iii) Hot Potato – Bugs that are reassigned more than two times
iv) Zombie – Bugs with no activity for more than a month

Software DEveloper Recommendation based on TOpic Models (DETROM) is modeled on a bipartite graph to make software developer recommendations [34]. As a first step, the software developers who contributed towards the resolution of the bug from the history are extracted from the activity log associated with each bug. A bipartite graph is built with nodes as software developers and bug reports. Topic modeling is applied over the natural language contents of the resolved bugs and the conditional probability that a bug belongs to a particular topic is derived. The association between the software developer and the bug reports is calculated as a probability. For an incoming bug report, the topic of the bug is determined. The probability of a software developer towards resolving a bug is calculated. The software developer with the highest probability is chosen for resolving the bug. The DETROM system is validated using Precision, Recall and F-Measure.

The time meta data is used to improve the bug assignment process [35]. Any text data can be converted to a tf-idf data based on the frequency of occurrence in the text documents. In the time based tf-idf, the time of use in the project is also considered. There are three modules in this approach. They are: (i) corpus creation (ii) expertise determination and (iii) developer recommendation. In the corpus creation, the data is collected from the versioning repository. From the source code, the identifiers are extracted and the relationship between the identifier and the developer who contributed is established. The source code identifiers are usually the name of classes, methods etc as they indicate the functionality of the object. The extracted identifiers are divided through tokenization and are converted to decomposed identifiers. For each identifier, the developer associated with the identifier and the time stamp the source code was committed is taken into account for term weighting. Only the noun terms are used. The terms are lemmatised. Finally, an index of terms with each unigram associated with a developer is created. Term weighting is performed by calculating the frequency of the term. The inverse of term used by a developer and the date the bug was reported is used to calculate the time based tf-idf. The calculated term weight determines a developer's expertise. A simple ranking method is used to rank the developers by combining the term weights of the terms occurring in the new bug report as well as in the developer expertise. Three approaches based on SVM classifier and Term Frequency - Inverse Document Frequency, LDA with SVM and LDA with Kullback Leibler divergence are comprehensively investigated [36]. The LDA with Kullback Leibler divergence was found to provide better consistency across components. The bug reports in the training set are assigned components. From this information, the average topic probability of all reports in the component category is computed. This average acts as the centroid of topics. The new bug report is categorized by computing the divergence of the new bug report's document topic probability from the average topic probabilities of each component in the training set. The system is evaluated for Recall parameter.The summary of the survey in Bug Triaging based on information retrieval in OSS is presented in the Table 2.2.

Table 2.2: Summary of Information Retrieval based Techniques

Paper	Information Retrieval					
	Accuracy	Precision	Recall	F-Measure	Fix Time	Average hit Ratio
[29]	✓	-	-	-	✓	-
[30]	-	-	-	-	-	✓
[34]	-	✓	✓	✓	-	-
[32]	✓	-	-	-	-	-
[36]	-	-	-	✓	-	-

2.2.1.2 Location based Techniques

Location based methods scrutinizes the source code file that may be altered to handle the new bug. The probable software developers for resolving the bug are identified from the natural language content of the source code file or by identifying the software developers who contributed to the source code file.

Location based bug assignments are an alternative to activity based bug assignment [37]. The source code files that need to be modified for the bug are identified in the first phase. The software developers associated with the source code files are identified in the second phase. Parts of speech method is used for nouns extraction from four different sources, namely (i) identifiers such as name of classes' methods (ii) commit messages (iii) the comments present in the source code and (iv) previously fixed bugs for the source code files that the new bug has identified itself with. The evaluation parameter used is Accuracy. The proposed system has achieved an Accuracy of 89.41% and 59.76% for eclipse and Mozilla projects.

Change request handling comprises of bug reports processing as well as feature requests processing[38]. In the first step, a corpus for the software system is created. Identifiers and comments in the source code are utilized to create the corpus. The indexing of the corpus is done using LSI. For a new change request, a query is formed. The query is a set of terms formed from the change request or bug report. The similarity between the query and the source code files in the corpus are determined and the ranking is performed. The software developers who contributed to the source code files are extracted according to the commits submitted by the software developer in the version control repository. The software developer's Euclidian distance from the source code file is used to rank the software developers. The system produces accuracy in the range of 47% to 96% for bug reports, between 43% and 60% for feature requests.

Source code contributions also form a basis for Bug Triage [39]. The vocabulary found in the "diffs" of a software developer is lexically compared with an incoming bug for making recommendations. The expertise of each author is modeled as a term author matrix. The activity decay of a software developer and vocabulary decay are modeled. The proposed system achieved 33.6% Precision and 71% Recall. Another location based method develops a Developer-Component-Bug network for assigning bugs. In the Developer-Component-Bug network, the bug are localized in accordance with the component they belong to and developers are localised with respect to the location of the bug they have fixed [40]. The VSM is applied on the incoming new bug report and the similarity between the new bug report and other fixed bug reports are calculated. The Developer-Component-Bug network is then referred to and the ranked set of developers based on the relevance score between the developers and the new bug report is listed. The system was evaluated with Recall. The summary of the survey in Bug Triaging based on location in Open Source System is presented in Table 2.3.

Table 2.3: Summary of Location based Techniques

Paper	**Information Retrieval**					
	Accuracy	**Precision**	**Recall**	**F-Measure**	**Fix Time**	**Average hit Ratio**
[37]	✓	-	-	-	-	-
[38]	✓	-	-	-	-	-
[39]	-	✓	✓	-	-	-
[40]	-	-	✓	-	-	-

2.2.2 Link based Techniques

Only the link data is used here. The importance of non-committers known as catalysts is acknowledged [41]. The catalysts may not actually solve the bug, but may significantly contribute in terms of commenting or committing a bug. The commenting activities and the tossing of bugs are captured in a directed tossing graph based on AP model. A minimal essential graph algorithm has been designed to identify the essential catalyst in solving a bug. The system is evaluated as per the number of nodes reduced.

2.2.3 Hybrid Techniques

A Bug Triaging System which combines machine learning with Bug Toss Graph based on GP model is advocated [12]. Many machine learning methods were analyzed and it was inferred that Naïve Bayes gives the best result. To handle the occurrence of inactive software developers and modification of software developer expertise, they have modeled the Bug Toss Graph as a multi-featured graph. Each arc in the graph is labeled with the bug class that is the product/component and software developer activity count. The multi-featured graph is modeled as a GP. A Weight Based Breadth First Search (WBFS) algorithm is employed to find the best tossing relationship. The whole system is incorporated in an incremental learning framework so that recent bugs have more impact on the recommendations made by the system. The technique achieved 83.62% Prediction Accuracy and 86.31% reduction in tossing lengths.Bug tossing relationships modeled as Bug Toss Graph based on GP model to make software developer recommendations were first employed in OSS systems [11]. A GP model was made based on first order Markov model. WBFS algorithm was used to maximize the path reduction. It has been proved that the usage of WBFS algorithm reduces the tossing length from that of the original path length. The GP model also helps to increase the prediction accuracy by finding the best tossing relation. Machine learning techniques, Naïve Bayes and Bayesian Networks were used in combination with the Bug Toss Graph to make the recommendations. The method increases the accuracy by 70% and reduces the tossing length by 72%.

The Bug Toss Graph can also be modeled using a socio-technical approach [42]. The model exploits the number of comments made by a software developer in bug resolution. The software developers are prioritized as per the out-degree. This information is leveraged with SVM and Naïve Bayes Classification results. An incremental learning framework with 10 folds is used to recommend software developers. A new ranking function based on classifier probability, software developer prioritization score based on product and software developer prioritization based on component is computed. The software developer prioritization based on SVM produced the best results. An approach that consists of three aspects: Bug Toss Graph, textual similarity and sub graph generation to narrow down the search space is presented [43]. The textual similarities between the new bug and the existing bugs were calculated using VSM. Then, the sub graph was generated. Over the sub graph, WBFS algorithm was deployed to identify the top software developers. This approach reduced the tossing length by 76.25%.

The bug repository consists of multiple types of entities. A heterogeneous network was formed from the bug repository[44], [45]. The heterogeneous network consists of four types of entities: (i) software developer(ii) bug(iii) comments and (iv) component. Further, each software developer can make four types of contributions: (i) reporting bugs (ii) commenting (iii) reopening bugs and (iv) fixing bugs. The network schema of the objects and their relations in the bug repository is developed. Based on the network schema, the heterogeneous network is evolved. For an incoming bug report, the list of potential software developers is extracted by using SVM and Naïve Bayes Classifier. For each software developer, the best collaborating partner is identified by using the collaboration probability metric. The top five software developers along with their partners are listed for resolving the bug.

A hybrid bug triage algorithm is one which combines a probability model and experience model to recommend software developers [46]. For any new bug, the bugs that are similar to it are shortlisted by an unigram model. From the shortlisted bugs, a social network based on probability model is constructed using the comments posted by the software developer. An experience model is extracted again from the shortlisted bugs using the number of fixed bugs and activity factor.The hybrid bug triage algorithm then utilizes both models to recommend software developers. The unigram based method is enhanced by combining it with component information to identify bug reports that are similar to the new bug report [47]. From the short listed bug report, a Multiple software Developer Network (MDN) is constructed. In MDN, the nodes represent the software developers and edges are labeled as vector. The vector consists of two components: (i) the number of commits made by the software developer and (ii) the number of comments made by the software developer. A software developer ranking score is used to rank the software developers. The idea of toss graphs for OSS was introduced in 2009 [13]. The Bug Toss Graphs are modeled after GP model. The metric used for evaluation is Prediction Accuracy.

Social networks are utilized to build the concept profile of the developer [48]. Using the concept profile a ranked list of developers are retrieved according to their expertise and fixing cost. The concept profile development consist of two parts i) categorizing bugs ii) extracting Topic terms. K-means clustering algorithm is used to categorize the bug. The cosine similarity measure is applied in finding the similarity among the bugs. Extracting Topic terms involves extracting terms with high frequency. A social network of developers as nodes and their commenting activity as links is formed. Based on the number of links for a developer and the number of bugs resolved by a developer, a probability score for each developer is calculated. Finally, a ranking algorithm is used to rank the developers according to the developers experience and fixing cost which is the function of fixing time.

The crashes that occurred in software were captured in a crash graph method [49].The crash graph is used to provide a high level view of the crashes reported as bug. The evaluation parameter used is graph similarity. Social networks are developed based on the developers commenting activities to detect developer communities. The developers in each community are ranked based on their experience. When a bug report is assigned as ‘new’ the relevant community for the bug is identified. The DEvelopers COmmunities in Bug Assignment (DECOBA) has five steps. The first step creates a term matrix of bug reports. The next step is to create the social network of developers. The third step identifies communities using a greedy optimization algorithm that detects dense sub graphs. A predictive model is built as a fourth step in determining the appropriate community for the incoming bug. The predictive model is built on Naive Bayes and Random Forest algorithm. The system is evaluated using centrality measures.

The summary of the Bug Triage techniques based on Hybrid Techniques is listed in the Table 2.4.

Table 2.4: Summary of Hybrid Bug Triaging Techniques

Paper	Content									Link		
	Machine learning				Information Retrieval					Metric	Graph Model	
	Accuracy	Precision	Recall	F-Measure	Accuracy	Precision	Recall	F-Measure	Mean Reciprocal Rank	No of steps	Tosses	Comments
[12]	✓	-	-	-	-	-	-	-	-	✓	✓	-
[11]	✓	-	-	-	-	-	-	-	-	✓	✓	-
[13]	✓	-	-	-	-	-	-	-	-	✓	✓	-
[42]	✓	-	-	-	-	-	-	-	-	✓	✓	-
[44]	✓	-	-	-	-	-	-	-	-	-	-	✓
[45]	✓	-	-	-	-	-	-	-	-	-	-	✓
[46]	-	-	-	-	-	-	-	✓	✓	-	-	✓
[47]	-	-	-	-	-	✓	✓	✓	-	-	-	✓
[48]	-	-	-	-	-	✓	✓	-	-	-	-	✓
[49]	-	-	-	-	-	-	-	-	-	-	-	✓
[50]	-	-	-	-	-	-	-	-	-	✓	✓	-

2.2.4 Inference

From the literature survey, it is inferred that machine learning techniques have been broadly applied on the summary field of the bug report. The widely applied techniques are Naïve Bayes and SVM [11] - [16], [19], [27], [44], [45].These two techniques give better results than the other machine learning techniques.The tossing relations that exist among the software developer are captured predominantly as Bug Toss Graph using GP model [11], [12], [13], [42], [50]. WBFS algorithm is broadly used to search the Bug Toss Graph.

2.3 STATE OF THE ART TICKET RESOLUTION TECHNIQUES IN ENTERPRISE SYSTEM

The survey pertaining to ticket resolution has been done along these dimensions:

i) Data

ii) Algorithm

iii) Ticket Resolution Graph Model

iv) Evaluation Parameters

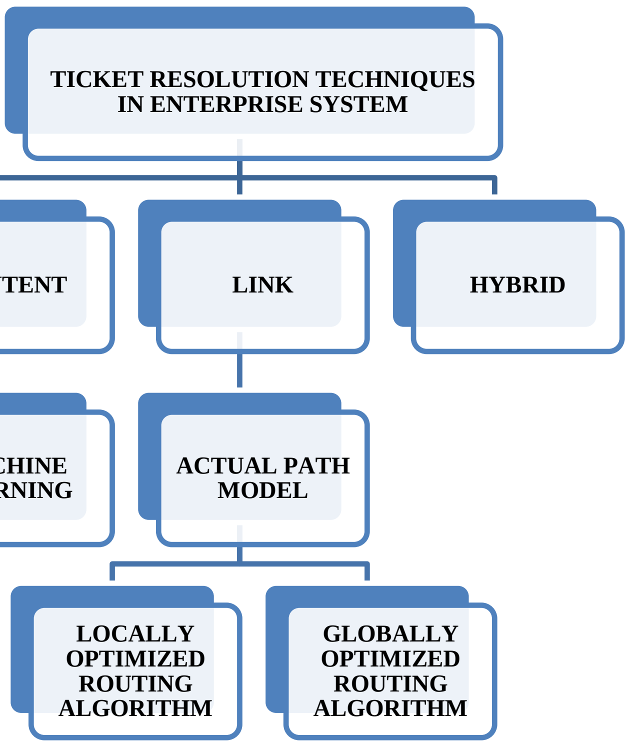

TICKET RESOLUTION TECHNIQUES IN ENTERPRISE SYSTEM
CONTENT
LINK
HYBRID
MACHINE LEARNING
ACTUAL PATH MODEL
LOCALLY OPTIMIZED ROUTING ALGORITHM
GLOBALLY OPTIMIZED ROUTING ALGORITHM

Figure 2.2: Classification of Ticket Resolution Techniques in Enterprise System

The Figure 2.2 gives the classification of Ticket Resolution Techniques in the literature. Ticket resolution is mainly done by content based, link based or a combination of content and link based methods. The dominant method that is applied on the content is machine learning. In link based methods, the ticket transfer relations among the software developers are captured using a ticket resolution graph based on an AP model.

2.3.1 Content Based Techniques

2.3.1.1 Machine Learning

The Trouble Miner system uses document clustering methods to obtain related trouble tickets [51]. The hierarchical clustering algorithm produces a binary tree of un-labeled nodes which capture the relation between trouble tickets. An algorithm based on depth first search labels the unlabeled Binary tree. The labeled tree is then converted to a hierarchy H of n-ary tree.

2.3.2 Link Based Techniques

A statistical model is used to capture the ticket resolution sequences [52]. The order of the Markov model is fixed according to the conditional entropy obtained from the ticket data. A Variable order Multiple active state Search (VMS) algorithm generates the ticket transfer recommendations. The robustness of the algorithm was evaluated by taking three factors: (i) the training set size (ii) time variability of the data and (iii) variety of problem categories. The algorithm performed consistently when the training size was increased as well as across different problem categories.The performance of the VMS algorithm increased when ticket resolutions from recent windows were taken. Expertise awareness is determined in terms of transfer effectiveness [53]. A novel exclude algorithm has been designed to calculate the transfer effectiveness. The exclude algorithm helps to assess the weakest components in the resolution system. It has been inferred that Collaborative Networks have truncated power law node degree distributions. The routing in Collaboration Networks is also task driven. A network model which captures the static connectivity is proposed. A stochastic routing algorithm has been developed to simulate human dynamics in Collaboration Networks [54].

2.3.3 Hybrid Techniques

The Optimized Network Model (ONM) is a generative model which uses both content as well as link information [55] , [56]. The model captures the transition probability and the reason for which the ticket was transferred between two groups. Ranked Resolver algorithm, Greedy transfer algorithm and Holistic Routing algorithm has been designed. The ranked resolver algorithm is based on the probability a resolver will be able to solve ticket. The greedy algorithm makes one step transfer predictions. The holistic routing approach finds the best resolver within ‘k’ steps. The holistic algorithm is globally optimized. A hybrid model uses the ticket content and the resolution sequences to make ticket routing recommendations.

The system analyses the ticket content and groups a set of semantically similar tickets. A cosine similarity based weight function is used for model generation. The weighted Markov model is created from these set of tickets. The VMS algorithm is used to retrieve the resolution sequences. The algorithm is locally optimized. A unified generative model that portrays a lifecycle of a ticket using both content and routing sequence is presented [57]. The ONM applies maximum likelihood estimation to capture the transfer profiles. The algorithm is globally optimized.The summary of the Ticket Resolution techniques is given in Table 2.5.

Table 2.5: Summary of Ticket Resolution Techniques

Classification	Paper	Content	Ticket Resolution Graph	
		Machine learning	Actual Path Model	
			No of steps	Influence
Content Based	[51]	✓	-	-
Link Based	[52]	-	✓	-
	[53]	-	-	✓
Hybrid	[55]	✓	✓	-
	[56]	✓	✓	-
	[57]	✓	✓	-

2.3.4 Inference

It has been observed from the survey that AP models have been employed to model the ticket resolution graph[52] - [57]. The locally optimized ticket routing algorithm is the VMS algorithm. The globally optimized holistic algorithm outperforms the VMS algorithm [52] , [53], [57].The summary of the ticket resolution techniques followed are highlighted in Table 2.5.

2.4 EXTRACT OF THE LITERATURE SURVEY

2.4.1 Graph Models

Bug resolution is about collaboration among software developers to solve a bug. In order to solve a bug, it is necessary to find a set of software developers who can collaborate on a bug. It is also observed that not all reassignments are detrimental in nature. Some reassignments are necessary to determine the root cause of the bug[58] , [59], [60]. Thus, it becomes essential that the underlying software developer structure has to be preserved in the Bug Toss Graph. From the survey, it has been observed that GP models are extensively researched for Bug Triage in OSS. The GP model may be useful in determining the best toss relationship with a software developer. The GP models do not capture the software developer structure in the network. GP model loses knowledge pertaining to intermediate tosses.

The knowledge about the software developer structure is important to retrieve a set of collaborators who can deliberate on a bug. The AP model based Bug Toss Graphs need to be explored in the realm of OSS. Further, the current Bug Toss Graph model captures the presence or absence of tossing activities. The attribute assigned to this tossing activity feature is the cardinality of the tossing activity. Multiple attributes can be appended to the cardinality to enrich the graph model.

2.4.2 Concept Drift

Software projects evolve through time. In Open Source projects, the software developers collaborate to develop software. The software developers may become inactive after sometime and their area of expertise may change. The problem of concept drift is present in issue tracking systems. Software projects go through periods of stability and instability [61]. Concept drift has been explored to some depth by using Window Frequency Baseline, SVM Minibatch, Perceptron, modified Bugzie and Regression with Stochastic Gradient Descent [62]. The concept drift handling mechanism discussed here takes into account only the textual contents of the bug report. In the literature, concept drift handling techniques when taking the textual content as well as the tossing relations are based on two methods (i) Heuristics of software developer activity count [11], [12](ii) Incremental learning framework[11], [12], [42]. The incremental learning framework used in the literature uses fixed size window method, which is too rigid to capture the software evolution process.

The Bug Triaging process is a learning process where the Automated Bug Triaging system has to learn from past experience to recommend the best set of collaborators. Recent data will have more weight in determining the set of developers. In order to make the triaging process robust, methods which exploit both the contents as well as the tossing relations have to be explored. Attention may be given to additional techniques like adaptive techniques in learning model to perform effective Bug Triage. Further, the performance loss due the use of GP model needs to be analysed.

2.4.3 Metrics

The metrics that are used in the Bug Triaging and ticket resolution systems are: (i) Accuracy(ii) Precision(iii) Recall and (iv) Mean Steps To Resolve. Precision is a better parameter for software developer recommendation because the cost of false recommendation is much higher than in search engine. Further, the Mean Steps to Resolve parameter codifies only the number steps in the predicted path. While the reduction in the number of steps to resolve is required, it is also vital to compare how far the predicted path is similar to the original path.

The structure and the ordering of nodes in the predicted path need to be compared with that of the original path. Metrics based on graph edit distance need to be introduced. Moreover, it is observed from the survey that the Bug Triage System is evaluated only with recommender metrics. In order to build confidence in the Bug Triage System, it is necessary to corroborate the results evaluated by recommendation metrics with evaluation by user metrics.

2.5 SUMMARY

Bug Triaging is an important aspect of OSS maintenance. A broad study is made on the state of art practices in Bug Triaging in OSS. The state of art practices are compared with the practices in enterprise system. It has been understood that many concepts of OSS have been adapted from the ES. The most important adaptation is the modeling of the Bug Toss Graph based on Markov model.

From the survey, the classification of techniques handling Bug Triage has been derived. It is inferred from the survey that Bug Triage techniques use the textual contents in the bug report or the tossing relations that exists among the software developers or both. The performance evaluation of the Automated Bug Triage System is based on the parameters (i) Precision (ii) Recall(iii) Accuracy and (iv) Mean Steps To Resolve. There are a few open areas of research that need to be explored in the area of Bug Triage. At present, GP models are widely adapted in Bug Triage. The performance of GP model needs to be compared with that of AP model in a quantitative manner. Concept drift is present in bug tracking. An efficient technique to handle concept drift that uses both the textual content as well as the tossing relations is the need of the hour. Effective knowledge management techniques are necessary to convert the raw information in the bug repository to intelligence which may be exploited by the software practitioners.

CHAPTER - 3

PROBLEM STATEMENT AND RESEARCH OBJECTIVES

3.1 PREAMBLE

This chapter presents the overarching goal of the research framework that is endorsed and the evaluation that is used to appraise the Ameliorated Bug Triage Process. The review of the Literature clearly asserts that there is a need to upgrade the Bug Triage System along three directions. They are: (i)the improvements that are imperative in the graph model(ii) the learning algorithm to be adaptive and (iii) the extensive metrics mandatory to enhance the performance of the Bug Triage System.

3.2 PROBLEM STATEMENT

The intent of this research is to ameliorate the Bug Triage process in Open Source Systems. The basic hypothesis is that, Bug Triage can be significantly improved by assimilating the propinquity among the developers engaged in bug resolution in the enriched collaboration graph and by deploying the adaptive learning techniques on the graph.

Research Question1: Does the social context of the developers make an impact in the Bug Triage Process?

Research Question 2: Does the adaptive techniques make a difference in the Bug Triage Process?

3.2.1 Research Objectives

Three Research Objectives are derieved from the Problem Statement. They are as follows:

Collaboration Graph Model

- To scrutinize and determine the observable difference in the performance of the existing Goal Oriented Path Model against the Actual Path Model.
- To determine the factors influencing the performance of the Actual Path Model.
- To integrate the propinquity among the developers to formalize the Enriched Collaboration Graph.

Learning Model

- To deploy adaptive learning over the Enriched Collaboration Graph to engage the dynamic, self organizing Bug Triage environment.
- To model the adaptive learning complaint with Ant System.

Evaluation Framework

- To augument the Recommender metrics with Path Similarity metric.
- To develop an Holistic Evaluation Framework with Key Performance Indicators based on developer performance.

3.2.2 Research Methodology

The experimental research methodology was embraced to investigate the research questions in the previous section. The data from the www.bugzilla.org was utilized to perform the experiments. The Bug Toss Graph was concocted as a GP model. The GP model with the WBFS algorithm is designated as the baseline system.The experiments were repeated for three studies, with data for two years, three years and four years to evaluate the consistency of the results. The results from the three studies were statistically analyzed using Two Tailed T-test and ANOVA for significance against a p value < .05.

3.2.2.1 Data Preparation

The bug reports from Eclipse project were used for the experiments because of their stability factor. The schematic of the data preparation process is given in Figure 3.1. The activity data from the bug reports were used. First, the CSV file of the report was downloaded. Following that, a web crawler was designed to download the activity data of the Bug IDs in the CSV file from www.bugzilla.org . Further, the regex tool was used to resolve the name of the developers in the activity data. The extracted activity data were stored in Oracle database for further use. For every record in the activity data for a particular bug, there is a corresponding tuple created in the Oracle database. Each tuple in the Oracle database consisted of {Bug ID, Date, Whom, What, Status}. The 'date field' accords information about when the bug was assigned to the developer, the 'whom field' consists of the unique ID of the developer, the 'what field' consists of the contribution made by the developer. The next tuple contains information about to whom the bug was reassigned, the 'date reassigned', etc.

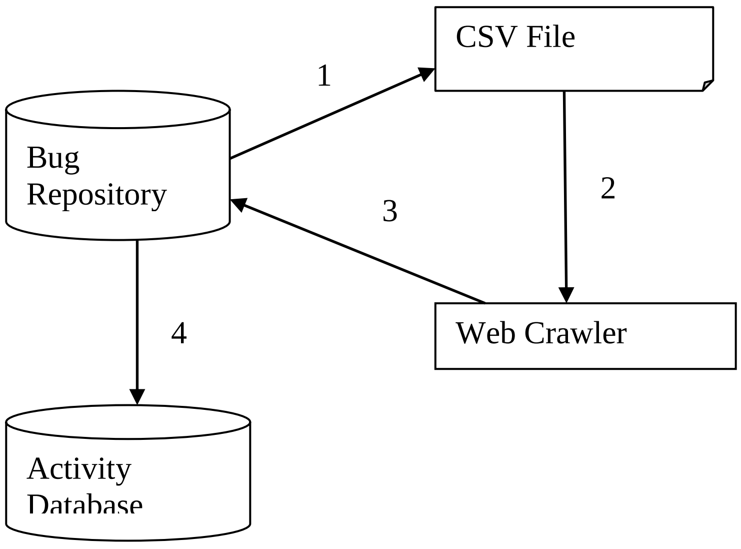


Figure 3.1: Schematic Diagram for Data Preparation

3.2.3 High Level Conceptual View of the Ameliorated Bug Triage Process

The High Level Conceptual View of the Ameliorated Bug Triage Process is illustrated in the following Figure 3.2.

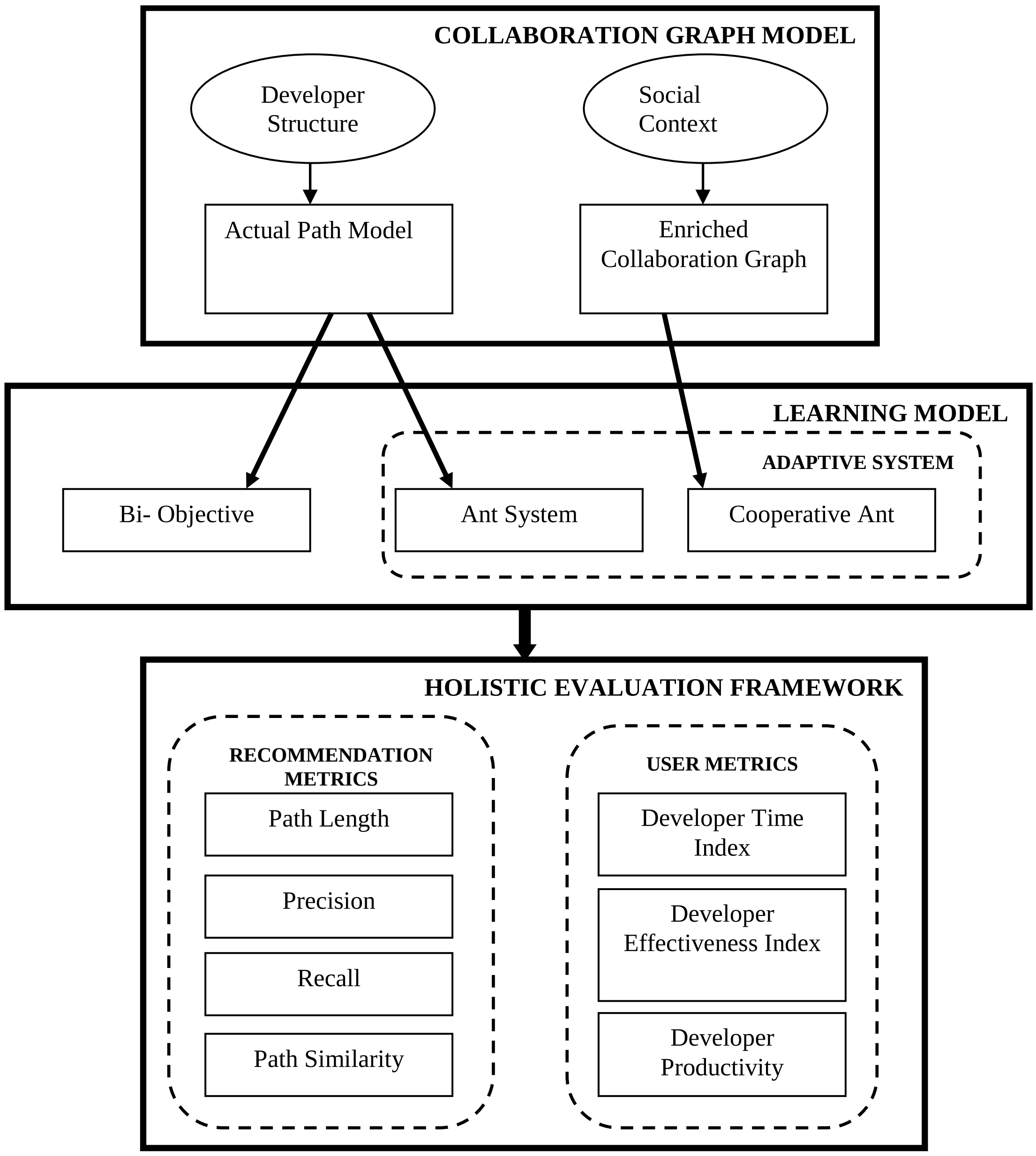


Figure 3.2: High Level Conceptual View of the Ameliorated Bug Triage Process

The overarching goal of the dissertation is to effectively extract patterns in the OSS maintenance to produce information that can be utilized in the Bug Triage process. This information can steer the Bug Triage practitioners to prescribe the developer recommendations based on historical data, rather than rely on gut and experience. The underlying principle of the dissertation is that software repositories contain dormant information that can assist in decision making. Based on this principle, the dissertation makes a three fold contribution:

- The underlying Bug Toss Graph model is analyzed. A new Bug Toss Graph model based on Actual Path model is designed. This graph model is enriched by augmenting the social context of the developers. This graph model is the Enriched Collaboration Graph.
- Adaptive techniques are formalized for learning from the Enriched Collaboration Graph.
- A two phase evaluation framework is designed to accommodate recommendation metrics as well as user metrics. This will aid in the rigorous evaluation of the Bug Triage System.

3.2.4 Evaluation Metrics

The outcome of the proposed system is a set of paths encompassing developers who can substantially contribute towards the resolution of the bug. The retrieved paths are appraised by an Holistic Evaluation Framework. The Holistic Evaluation Framework comprises the common recommendation metrics and the user metrics to evaluate the system. The recommendation metrics are Precision and Recall.

The Path Length metric compares the length of paths of the retrieved path and that of the corresponding original path.

$$\text{Path Length}_j = \frac{1}{n_j} \sum_{0}^{n_j} L_i$$

Where n_j - Number of paths of length 'j'

L_i - Length of the ith path of length 'j'

The Precision metric compares the number of developers in the retrieved path who match with the developers in the original path. The Precision value is in the range of {0..1}.

$$\text{Precision} = \frac{\text{Number of Relevant Developers Retrieved}}{\text{Number of Retrieved Developers}}$$

The Recall metric compares the number of developers in the retrieved path who match with the total number of relevant developers. The Recall value is in the range of {0..1}.

$$\text{Recall} = \frac{\text{Number of Relevant Developers Retrieved}}{\text{Total Number of Relevant Developers}}$$

The Path Similarity metric compares the developers in the retrieved path to the developers in the original path in terms of position and ordering. This metric is computed based on Levenshtein Similarity.

3.3 SUMMARY

This chapter catalogues a succinct view of the Problem Statement. The Research Objectives derived out of the research questions are given in a comprehensive manner. The Research Methodology and the data preparation module are highlighted in detail. In addition, the high level conceptual view of the dissertation is presented. Further, the performance metrics applied to investigate the Ameliorated Bug Triage System are also identified.

CHAPTER - 4

PATH SIMILARITY METRIC FOR BUG TRIAGE

4.1 PREAMBLE

This chapter divulges an exploratory analysis of Path Similarity metrics applicable for Bug Triage. As discussed earlier, the OSS Development is divergent from the traditional software development. Here, the developers are not collocated.They communicate mainly in the virtual environment. This fact propelled the need for Automated Bug Triage System so as to accelerate the Bug Triage process. Any Automated Bug Triage System discerns the optimal set of developers to whom a bug has to be sent, so that the bug can be resolved. To resolve the bug, first, its severity has to be determined, following which the component to which the bug belongs to, need to be identified. The root cause of the bug needs to be perceived, following which, the source code is modified so that the bug is resolved. There exists a definite ordering of tasks, that the bug should undergo before being resolved. Meanwhile, it has also been argued that all developers are not equal. There is a definite role each developer plays in the process of bug resolution. The different roles played by the developers vary among Triagers, Bug analysts, Assists, Patch tester, Patch-quality improvers, Core developers and Bug fixers [63].Triagers are contributors who examine a bug report and short list the potential fixers. Bug Analyst decides on bug prioritization, duplicate bug identification and valid bug confirmation. Contributors play the role of brokers. They identify to whom the bug has to be tossed. Once a probable solution for a bug has been submitted, a Patch tester tests the submitted patch. The Patch-quality improver improves the quality of the patch. Core developers are differentiated from bug fixers as developers who have contributed to the source code other than the bug fixes. Bug fixers are developers who contributed towards the bug fixing. Intuitively, the ordering of the developers as per their roles is expressed in the Figure 4.1.

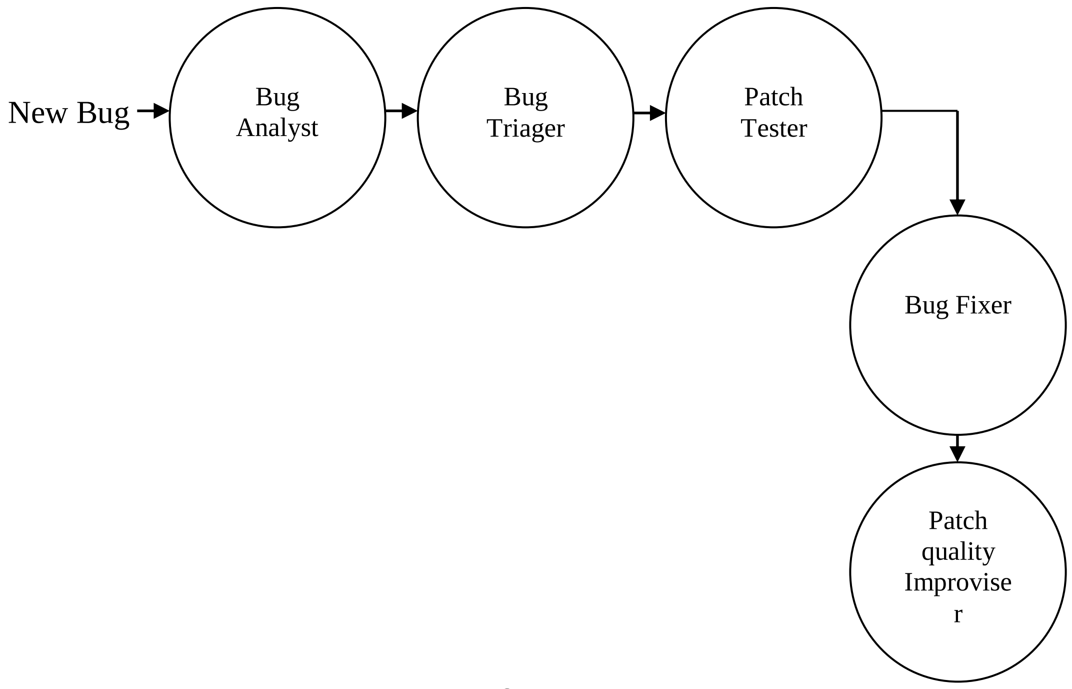


Figure 4.1: Roles of Developers in Bug Fixing

From the figure, it can be witnessed that there is a need for the bug to be passed along the developers in a certain order according to the developer role for resolving the bug effectively. It can be envisaged that there is a referral chain of developers through which the bug needs to be passed, so that it can be resolved.

The current evaluation techniques for assessing the referral chain rely on classical metrics from the recommendation systems. The recommendation metrics like Precision, Recall in essence validates whether the developers who actually collaborated on the bug are being retrieved by the Bug Triage System. In addition, there is a need for metrics that quantify the performance of the Bug Triage System based on the Path Similarity of the retrieved path of the developers with the original path of the developers. The metric should measure the ordering, position of developers in the retrieved path. To this end, techniques from bioinformatics and approximate string matching are explored to find the technique that is most suitable to serve as Path Similarity metric for Bug Triage System.

4.2 OVERVIEW OF PATH SIMILARITY

The task of pair wise sequence alignment is comparing the underlying structure of a path with another path. Sequences of DNA, RNA or protein can be matched and regions of similarity can be identified based on path or sequence alignment. This alignment may be because of structural, functional or evolutionary relation among the sequence.

Pair wise sequence alignment methods are used to find alignment between any two sequences. The three main methods for implementing pair wise sequence alignment are: (i) dot-matrix method, (ii) dynamic programming and (iii) word method. The pairwise alignment methods are applied in different fields like web page prediction, subsequence matching in time series databases using dynamic time warped measure, plagiarism detection and modeling network traffic in intrusion detection systems[64], [65], [66], [67], [68]. The pair wise sequence methods are broadly classified as local alignment method and global alignment method [69], [70].

4.2.1 Local Alignment

The local alignment of two paths or sequence deals with finding segment of one path that is close with the segment of another path.The most popular algorithms for local alignment are Smith-Waterman algorithm and Needle-Wunch algorithm [71]. Dynamic programming is mainly applied in finding the alignment of sequences. The original problem is subdivided into sub problems and a solution is found for the sub problems. The matches and mismatches between the two paths are given by computing a score in the scoring matrix. Smith-Waterman algorithm for sequence alignment is defined by dynamic programming approach.

4.2.2 Global Alignment

The alignment of two paths or sequences can be obtained by computing the distance between the two paths or sequences. Closely related sequences which are of same length are very much appropriate for global alignment. Here, the alignment is carried out from the beginning till the end of the sequence to find out the best possible alignment. Needleman-Wunsch algorithm is a popular global sequence alignment algorithm [72]. The number of mismatches and gaps are awarded a cost value to find the distance between the two paths.

4.3 AN ANALYSIS OF PATH SIMILARITY METRICS FOR BUG TRIAGE

This section records the analysis of the different pairwise sequence alignment methods for the evaluation of the retrieved path. The methods used for the exploration are Levenshtein similarity, Smith Waterman similarity and Jaro Wrinkler similarity[73]. The flow of the system is given in the Figure 4.2.

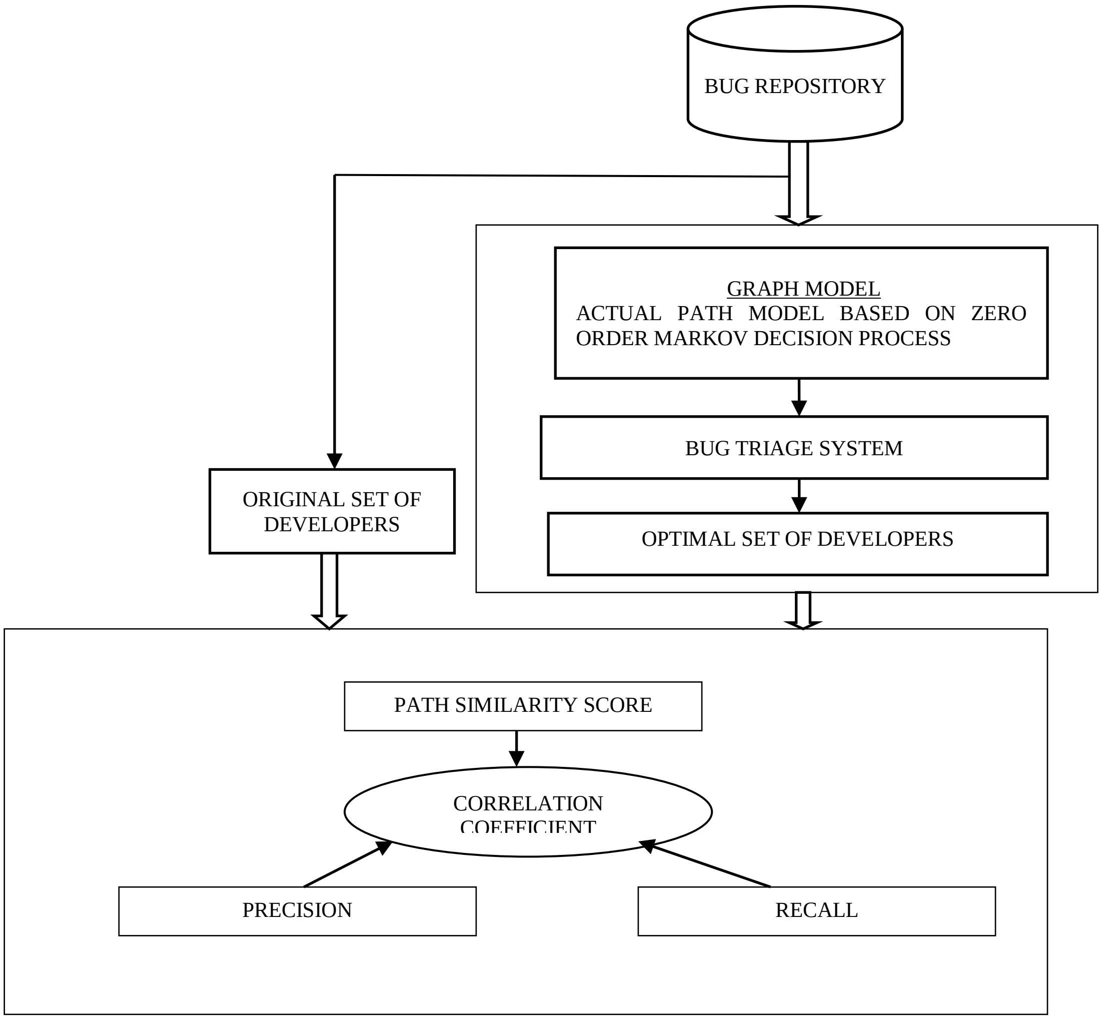

BUG REPOSITORY
GRAPH MODEL
ACTUAL PATH MODEL BASED ON ZERO ORDER MARKOV DECISION PROCESS
BUG TRIAGE SYSTEM
ORIGINAL SET OF DEVELOPERS
OPTIMAL SET OF DEVELOPERS
PATH SIMILARITY SCORE
CORRELATION COEFFICIENT
PRECISION
RECALL

Figure 4.2: Flow diagram of Analysis of Path Similarity Metrics for Bug Triage

The Automated Bug Triage System is used to retrieve the optimal path through which the bug has to be tossed. The retrieved paths are compared against the original path. The similarity values based on Levenshtein similarity, Smith Waterman similarity and Jaro Wrinkler similarity are computed. The correlation coefficient of each of the similarity values are computed against the Precision and Recall values. Based on the value of the correlation coefficient, the most suitable algorithm that can be applied as Path Similarity metric is chosen.

4.3.1 Problem Formulation

The bug reports from the repository are retrieved. The activity data from the bug reports are taken into consideration. The activity report consists in it, the bug tossing relation that exists in the history of a bug. This tossing relation that exists in a bug is assigned as a tossing path. For a set of bugs, a set of tossing paths can be composed. The union of all tossing paths is the Bug Toss Graph where the nodes are the developers and links are the tossing relation that exists among the developers. This relation that exists among the developers is captured as a zero order Markov model [12].The transition probabilities among the developers record the local decisions made by the developer. The Bug Toss Graph is defined as:

$G = \{V,E\}$: Bug Toss Graph

$V=\{v_1,v_2,\dots v_n\}$: set of states which represents the developer

$E=\{e_1,e_2,\dots\dots,e_n\}$: set of edges where each edge represents the previous bug tossing relation among the developer

$P(v_i|v_k)$: transition probability of an edge or link from v_i to v_k

$$P(v_i|v_k) = \begin{cases} N(v_i,v_k)/N(v_i) & \text{If } N(v_i) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

$N(v_i,v_k)$: number of transfers from v_i to v_k

$N(v_i)$: total number of transfers from v_i

The Bug Triage System is deployed over the Bug Toss Graph. The Bug Triage System proposes a set of developers to collaborate on the bug. The set of developers are represented essentially as an optimal path. The optimal path of developers ‘A’ is an array of nodes from ‘V’ that are contiguous and occupy a unique position.

4.3.2 Levenshtein Similarity

The Levenshtein distance is a global alignment method that compares two paths or sequences from end to end. The distance between any two paths is calculated based on the number of edit operations that needs to be performed on one path so that it becomes equal to the other. The edit operations are the number of insertions, deletions and number of substitutions [74]. The cost of each edit operation is

Insertion :1

Deletion:1

Substitution: 0.5

```
LevenshteinDistance( A[1..m],B[1..n])
Begin
1:      for i=1 to m
2:          d[i,0]=i
3:      for j=1 to n
4:          d[0,j]=j
5:      for j=1 to n
6:          for i=1 to m
7:              if A[i]==B[j]
8:                  d[i,j]=d[i-1,j-1]
9:              else
10:                 d[i,j]=Min(
11:                 d[i-1,j]+1,//Deletion
12:                 d[I,j-1]+1,//Insertion
13:                 d[i-1,j-1]+0.5)//Substitution
14:             endif
15:         endfor
16:     endfor
17:     return d[m,n]
End
```

Figure 4.3: Levenshtein Distance Algorithm

The underlying assumption behind the assignment of costs is that, if a developer has to be inserted, then a valuable developer has been missed by the Bug Triage System, therefore, the cost is 1.

If the edit operation is deletion, then, the underlying intuition is that an undesirable developer has been selected by the Bug Triage System that may, in turn be detrimental to the solving of the bug. So, the cost of deletion is 1. The cost of substitution is kept at 0.5, the reason being the developer has been retrieved, but, is in the wrong position.

The Levenshtein distance algorithm is given in the Figure 4.3. The input to the algorithm is the retrieved set of developers as path ‘A’ and the corresponding original set of developers as path ‘B’. First, a matrix ‘d’ is constructed with ‘M’ rows and ‘N’ columns. The first row and column are initialized to 0. Each developer in ‘A’ and that of ‘B’ are examined. If A[i] is equal to B[j], then, the developers match in position, so the cost is 0. The d[i,j] is made as the minimum of the cells in the immediate vicinity of d[i,j], i.e. d[i-1,j]+1,d[i,j-1]+1 and d[i-1,j-1]+0.5. Finally, the cost of matching the retrieved path to the original path is found in d[M,N].The Levenshtein similarity is obtained from Levenshtein distance by:

$$\text{Levenshtein Similarity} = \text{Levenshtein Distance}/\text{Max}(|A|, |B|\,)$$

Where $\text{Max}(|A|, |B|\,)$ gives the maximum of the absolute length of the retrieved path and the original path.

4.3.3 Smith Waterman Similarity

Smith Waterman Similarity computes the local similarity of retrieved path with that of the original path. In the milieu of Bug Triage, this means that sections of developer chain matching with the original path have been retrieved by the Bug Triage System.

The algorithm ignores the beginning and the terminal parts of the path. It returns the well preserved parts of the path connected by poorly connected segments.

Smith Waterman algorithm is based on dynamic programming principles. A scoring function 'SIM' is used to record the various alignments of the two paths relative to the number of gaps and number of matches in the alignment. The largest scoring alignment is taken to be the optimal alignment [68]. The retrieved path is $A=(a_1,a_2,\ldots\ldots a_m)$. The original path is $B=(b_1,b_2,\ldots\ldots b_n)$. The scoring matrix $SIM:\{1, 2, . . . , m\} \times \{1, 2, . . . , n\} \rightarrow R$ in which SIM(i,j) is the best alignment of the prefixes from the retrieved path and the original path is represented by $(a_1, a_2, . . . , a_i)$ and $(b_1, b_2, . . . , b_j)$. The matrix SIM is computed recursively. The value of 'g' is 0.5. The value of 'SIM' is set to 0 if all values that can be assigned to SIM[i,j] at position (i,j) are less than zero. The value of 0 at F(i, j) indicates that a new alignment has to commence at position (i,j). The algorithm for Smith Waterman Similarity is given in Figure 4.4.

```
SmithWatermanSimailarity(A[1..m],B[1..n])
Begin
1:      S=0
2:      for i=0 to m
3:              SIM[i,0] = 0
4:      for j=1 to n
5:              SIM[0,j] = 0
6:      for i=1 to m
7:              for j=1 to n
8:                      SIM[i,j] = max(
9:                                      0,
10:                                     SIM[i-1,j-1] + s(A[i],B[j]),
11:                                     SIM[i-1,j]+g,
12:                                     SIM[i,j-1]+g)
13:                     S=max(S,SIM[i,j])
14:             endfor
15:     endfor
16:     return S
End
```

Figure 4.4: Smith Waterman Similarity Algorithm

4.3.4 Jaro Wrinkler Similarity

Jaro Winkler similarity is an improvisation over Jaro Similarity. Jaro Similarity is a technique applied for approximate string matching. The core approximation done by this algorithm is that a developer in the retrieved path is not searched for in the exact position in the original path. Instead, the developer is searched in a segment of path which is equal to half the path length of the minimum of the original or the retrieved path. The algorithm Jaro Winkler similarity is given in Figure 4.5.

```
JaroWinklerDistance(A[1..m],B[1..n])
Begin
1:  g= Min(length(A),length(B))
2:  CommonA[]={}//Common Developers in Retrieved Path
3:  CommonB[]={}// Common Developers in Original Path
4:  l=0
5:  p=0
6:  trans=0 // Transposition
7:  LCP=0 // Longest Common Prefix
8:  for i=0 to m
9:              for j=0 to n
10:             for k=j-g to j+g // Common Computation
11:                     if (A[i]==B[j])
12:                             CommonA[l]=A[i]
13:                             CommonB[p]=B[j]
14:                             l++
15:                             p++
16:                     endif
17:             endfor
18:     endfor
19: endfor
20: for i=0 to l //Transposition Computation
21:     if  (CommonA[l]!= CommonB[l])
22:             Trans++
23:     endif
24: endfor
25: l=0
26: while (CommonA[l] == CommonB[l])) // Longest Common Prefix Computation
27: LCP++
28: l++
29: endwhile
30: JaroDistance=0.333 *((p/ length(A) ) +(p/ length(B) + 0.5*(Trans*p))
31: return(JaroWinklerDistance= (1-LCP*0.1) * JaroDistance + LCP * 0.1)
End
```

Figure 4.5: JaroWinkler Similarity Algorithm

Formally, the retrieved path is denoted as A=(a_1,a_2,……a_m) and original path as B=(b_1,b_2,…….b_n). A developer 'a_i' is a match with 'b_j' provided (i-g <= j <= i+g) where g= Min(|A|,|B|). The common developers from the retrieved and the original path are retrieved and compared. The number of transpositions required to transform one path to another is calculated. Finally, the Jaro distance is calculated as

$$\text{Jaro Distance} = \frac{1}{3}\left(\frac{\text{Common}}{|\text{A}|} + \frac{\text{Common}}{|\text{B}|} + \frac{1}{2} * \frac{\text{Transpositions}}{\text{Common}}\right)$$

$$\text{Jaro Winkler Distance} = (1 - |\text{LCP}| * \text{LCPwt}) * \text{Jaro Distance} + |\text{LCP}| * \text{LCPwt}$$

Where |LCP| is the length of the common prefix path and LCPwt is the weight assigned to the length of the common prefix path.

The JaroWinkler Similarity is computed by

$$\text{Jaro Winkler Similarity} = \frac{\text{Jaro Winkler Distance}}{|\text{A}| * |\text{B}|}$$

4.3.5 Performance Evaluation

The performance of the three algorithms was analyzed using the bug reports from the Eclipse project of www.bugzilla.org. The bug reports for the period from 2009/01/09 to 2013/01/09 were used to conduct the experiments. The experiments were conducted as two studies to test the consistency of the results obtained. The data considered for the first study were for the bug reports of the range 2009/01/09 to 2011/01/09,i.e. two years data set(Study 1). The data considered for the second study were from 2009/01/09 to 2013/01/09,i.e four years data set(Study 2). The experiments were run on a Pentium4 processor with 320 GB hard disc. Netbeans 7.2 was used as the frontend and Oracle as the backend. The JDK tool was employed in conducting the experiments.

The results from the experiments are presented in the Figure 4.6 and Figure 4.7.

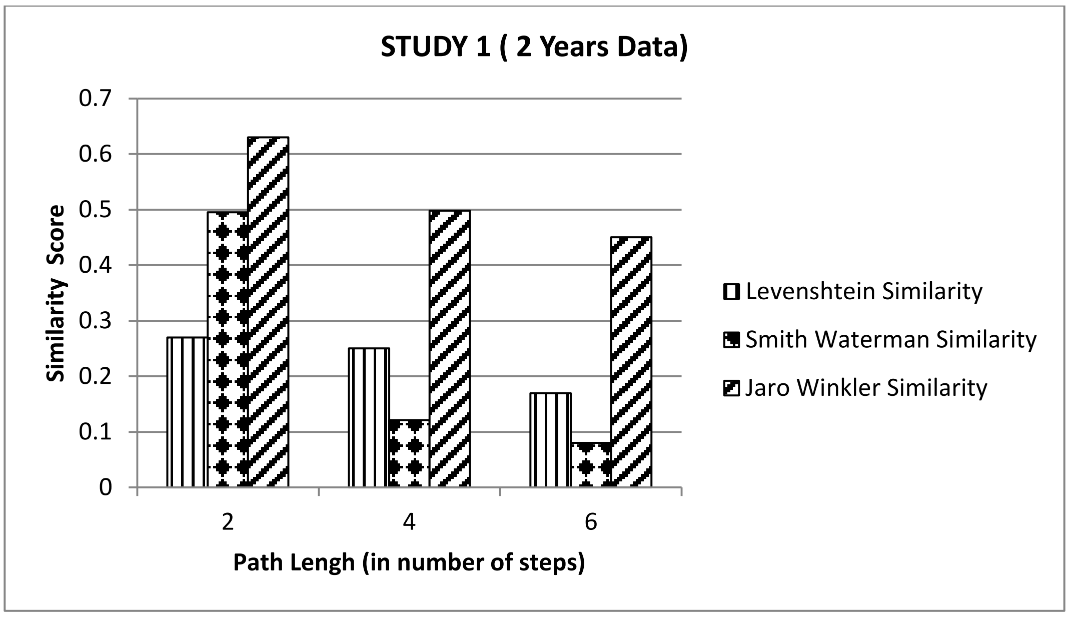


Figure 4.6: Similarity for Study 1

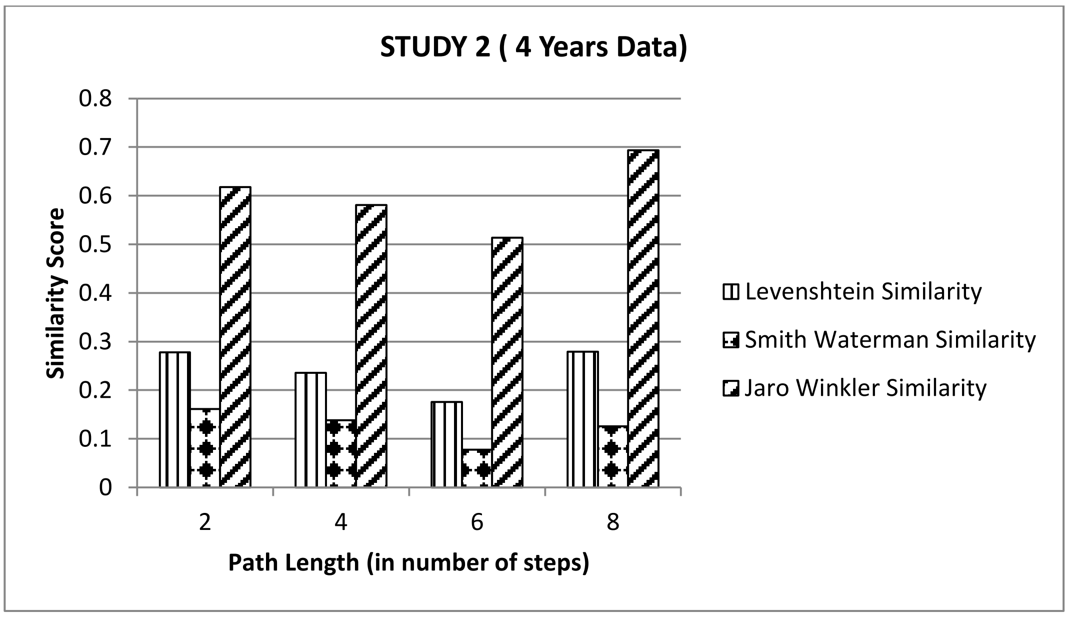


Figure 4.7: Similarity for Study 2

From the results presented, it can be deduced that JaroWinkler algorithm outperforms in similarity values when compared with the other two algorithms. But, to determine which algorithm can be used as a Path Similarity metric, the correlation of the similarity values with that of the Precision and Recall values are calculated.

The Precision and Recall values for the Study1 and Study2 are presented in the Figure 4.8 and Figure 4.9.

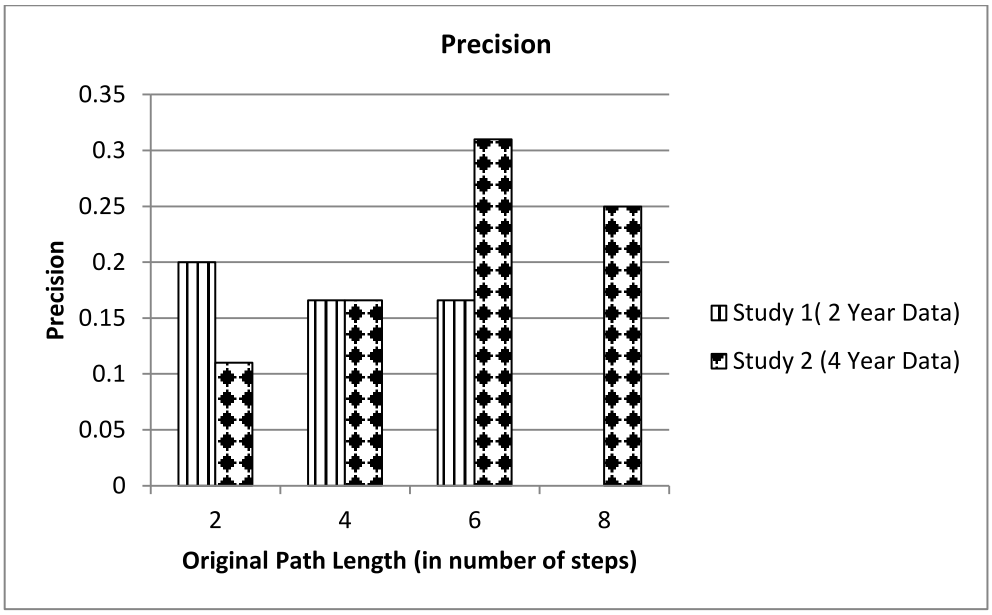


Figure 4.8: Precision for Study 1 and Study 2

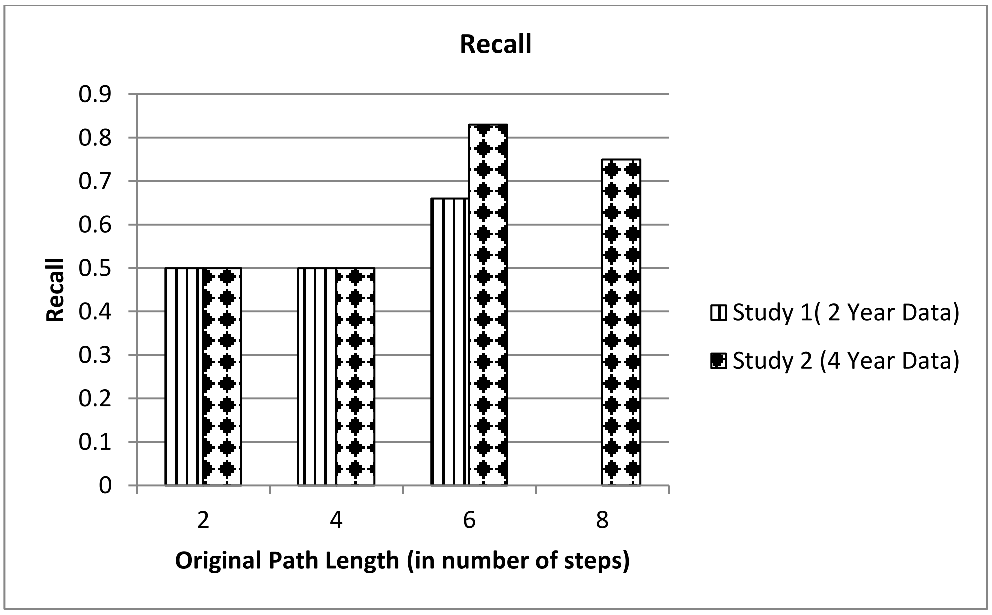


Figure 4.9: Recall for Study 1 and Study 2

The results of the Correlation Coefficients for Levenshtein, Smith Waterman and Jaro Winkler Similarity with the corresponding Precision and Recall values are tabulated in the Table 4.1.

Table 4.1: Correlation Coefficients

	Study 1 (2 Years Data)		**Study 2 (4 Years Data)**	
	Correlation w.r to Precision	**Correlation w.r to Recall**	**Correlation w.r to Precision**	**Correlation w.r to Recall**
Levenshtein Similarity	0.9714	0.9532	0.9671	0.8949
Smith Waterman Similarity	0.9714	0.9349	0.9412	0.9245
Jaro Winkler Similarity	0.9532	0.8921	0.9595	0.9068

From the results presented in the Table 4.1,it is observed that though Jaro Winkler Similarity values are higher than the other algorithms, when it comes to correlation with the other Precision and Recall values, Levenshtein Similarity is consistently outperforming the Smith Waterman Similarity algorithm and Jaro Winkler Similarity algorithm.Hence, Levenshtein Similarity is confirmed as the method that will serve as Path Similarity metric for the Bug Triage System.

4.4 SUMMARY

This chapter recounts the analysis of Path Similarity algorithms to determine the most suitable algorithm that can be used to compute the Path Similarity of the retrieved path from the Bug Triage System with that of the original path. The results and the correlation coefficient of the similarity values and that of Precision and Recall values of the Bug Triage System were analyzed. From the correlation coefficient values, it is concluded that Levenshtein Similarity is the most consistent in performance; thereby,Levenshtein Similarity is used as the Path Similarity metric.

CHAPTER - 5

ANALYSIS OF PATH MODELS FOR BUG TOSS GRAPH

5.1 PREAMBLE

The decision making in software organizations is based on experience and intuition [75]. When a software development spans a long period of time and developers are geographically distributed, exploiting the past experience of a developer becomes a problem. This issue is more attenuated in OSS Development. In this juncture, Bug Triaging is time consuming and error prone. The bugs are reported from a deployed software system. The reported bugs are accumulated in a bug repository. The reported bugs are to be assigned to a developer who may solve the bug. This assignment of a bug to a developer in an Open Source environment is intricate. The Automated Bug Triage Systems that are currently prevalent exploit two issues: one is the summary part of the bug report using machine learning techniques and other is the bug tossing relations. If a developer is not able to solve a bug assigned to him, he may toss the bug to another developer. These tosses are captured as Zero Order Markov process in a Bug Toss Graph.

The Bug Toss Graphs that are used in the OSS are modeled as a GP model [12], [13]. In a GP model, the relation between any assignee and the final resolver of the bug is captured. This model is effective in eliciting the final resolver. While Automated Bug Triaging System should try to minimize the number of tosses, it should be understood that not all tosses are bad. OSS Development is unstructured. It does not conform to typical software development environment where a bug can be deliberated upon, so it can be assumed that some of the bug tosses that occur at the initial stages are group level discussions on the bug. These initial discussions are deemed to be crucial. The discussion may pertain to identifying the root cause of the bug, the component fixing, the severity field fixing, etc [50], [58], [59]. In this conjuncture, the Bug Toss Graph modeled as a GP model does not capture the underlying structure of the bug resolution. On the contrary, in the ticket resolution techniques applied in ES, the developer collaboration is established as an AP model [52], [56]. This chapter presents a comparison of the performance of Bug Toss Graphs based on GP model and AP model.Further, the chapter advocates for an enhanced Bug Triage System based on Bi-Objective optimization.

5.2 ANALYSIS OF PATH MODELS FOR BUG TRIAGE

5.2.1 Formal Representation of Goal Oriented Path Model

The Bug Toss Graph based on GP model is a directed weighted graph, such that, the developer involved in the bug resolution acts as the node and the relation between each developer and the final resolver is the edge. Formally stated, the Bug Toss Graph G=(D,E) is a directed weighted graph with weight function '*w'*. The weight function w(u, x) of the edge (u, x) $\in E$ is the total number of tosses from developer '*u'* to final resolver '*x'* during bug resolution.

For example, the bug toss paths are given in the Table 5.1.

Table 5.1: Sample Bug Toss Paths

S.No	Tossing Paths
1	a→b→c→d
2	b→a→g→d
3	a→c→e

The individual decomposed paths as per GP model are depicted in the Table 5.2.

Table 5.2: Decomposed Paths as per GP model

S.No	Individual Paths	Weights
1	a→d	2
2	b→d	2
3	c→d	1
4	g→d	1
5	a→e	1
6	c→e	1

The transition probability values matrix for the given decomposed paths is given in the Table 5.3.

Table 5.3: Transition Probability Values of GP model

	a	**b**	**c**	**d**	**e**	**g**
a				0.67	0.33	
b				1		
c				0.5	0.5	
d						
g				1		

5.2.2 Formal Representation of Actual Path Model

The Bug Toss Graph based on AP model is a directed weighted graph such that each developer involved in the bug resolution acts as a node and the bug tossing relation among them are the edges. Each bug after being reported is liable to being tossed through a number of developers before getting resolved. Formally stated, the Bug Toss Graph G=(D,E) is a directed weighted graph with weight function ‘w’. Each edge in ‘E’ is the toss that exists between any two developers who shared a toss when solving a bug. The weight function w(u, v) of the edge (u, v) ∈E is the total number of tosses from developer ‘u’ to developer ‘v’ during bug resolution. In the AP model, each individual toss that exists between the developers is captured. For example, the bug tossing paths are given in the Table 5.4.

Table 5.4: Sample Bug Toss Paths

S.No	Tossing Paths
1	a→b→c→d
2	a→b→g→d
3	b→c→e

The individual decomposed paths as per AP model are depicted in the Table 5.5.

Table 5.5: Decomposed Paths as per AP model

S.No	Individual Paths	Weights
1	a→b	2
2	b→c	2
3	c→d	1
4	b→g	1
5	g→d	1
6	c→e	1

The transition probability values matrix for the given decomposed paths is given in the Table 5.6.

Table 5.6: Transition Probability Values of AP model

	a	**b**	**c**	**d**	**e**	**g**
a		1				
b			0.67			0.33
c				0.5	0.5	
d						
g				1		

5.2.3 Goal Oriented Path Model Vs Actual Path Model

The flow of the analysis of GP model against the AP model is illustrated in the Figure 5.1.

The bug reports from the bug repository are collected. The activity data is used to construct the Bug Toss Graph. The Bug Toss Graph is constructed based on GP model as well as AP model. The algorithm WBFS is deployed on the constructed graph to retrieve the optimal path the bug may traverse. The retrieved paths are evaluated based on the parameters: (i) Path Length (ii) Precision (iii) Recall and (iv) Path Similarity.

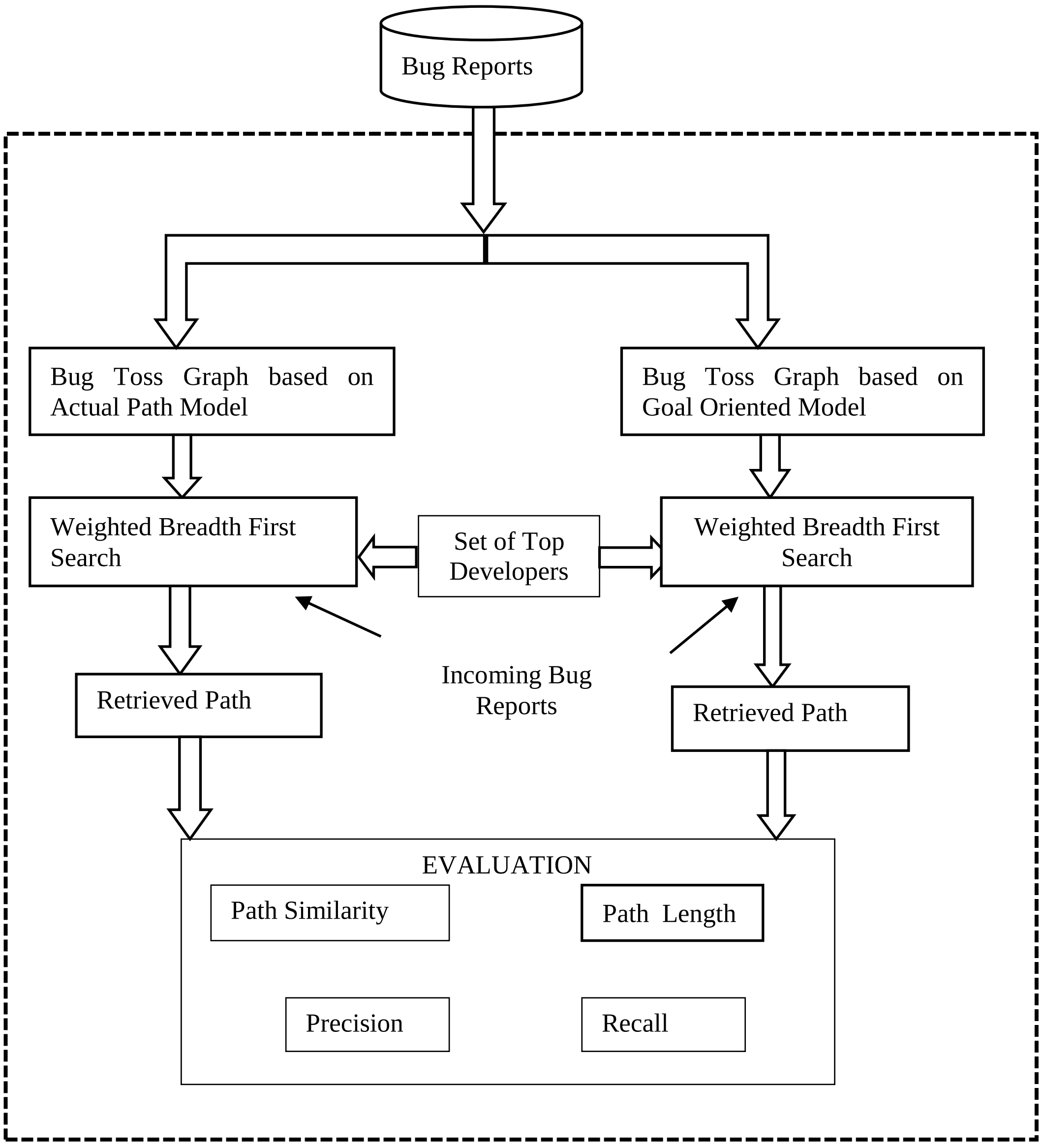

Bug Reports
Bug Toss Graph based on Actual Path Model
Bug Toss Graph based on Goal Oriented Model
Weighted Breadth First Search
Set of Top Developers
Weighted Breadth First Search
Incoming Bug Reports
Retrieved Path
Retrieved Path
EVALUATION
Path Similarity
Path Length
Precision
Recall

Figure 5.1: Flow of Analysis of Path Models for Bug Triage

5.2.4 Performance Evaluation

The activity report of the bug reports from the eclipse project from the year 2009 to 2013 was used for the experiments. The bug reports with status fixed and resolved with vote count 2 were utilized for the experiments. The experiments were conducted as a total of three studies to test the consistency of the expriments. The three studies were conducted for bug data from Eclipse project 2 years data, 3 years data and 4 years data. The experiments were run in a system with Core i3 processor, 2GB RAM and 320 GB hard disk. The environment in which the experiments were conducted comprises of Windows 7 Operating System, Oracle 10g database and Netbeans 7.2. The retrieved paths from the GP model and AP model were compared with parameters: Path Length, Precision, Recall and Path Similarity.

5.2.4.1 Path Length

Hypothesis with respect to the result of the parameter P: (Path Length)

Null hypothesis H0: P1 = P2 where P1= Path Length obtained in GP-WBFS (GP model with WBFS), P2 = Path Length obtained in AP-WBFS (AP model with WBFS)

(There is no significant difference between the two systems that is GP-WBFS and AP-WBFS in terms of Path Length obtained)

Alternate hypothesis H1: Path Length mean values are not equal for at least one pair of the result mean values of the parameter P.

(There is a significant difference between the two systems in terms of Path Length obtained)

The Path Length comparison is given in Figure 5.2, Figure 5.3 and Figure 5.4. From the graphs, it is evident that the existing GP model gives superior results than the proposed AP model.

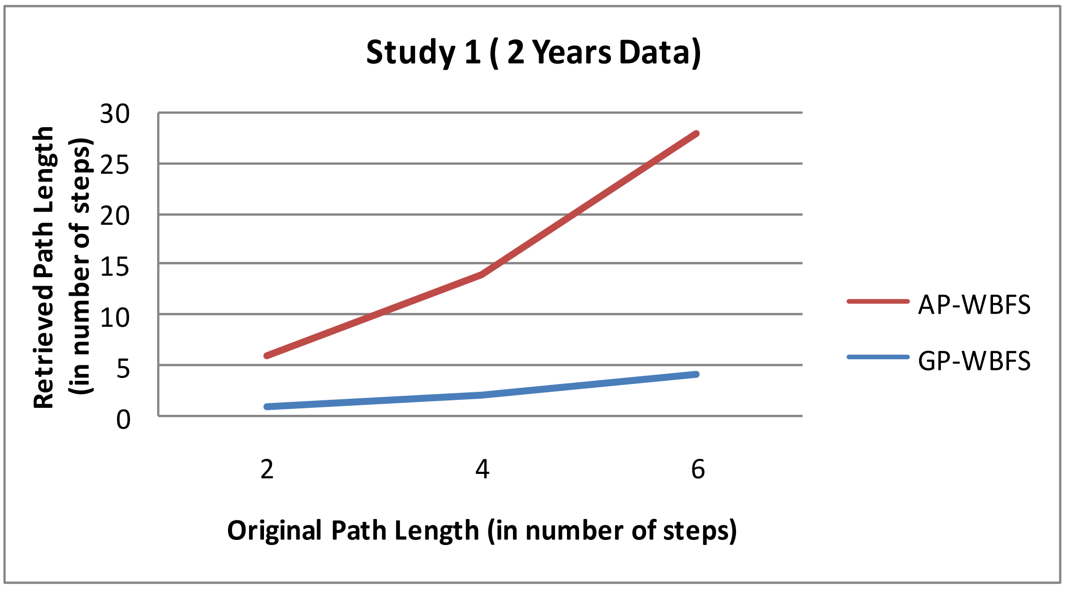


Figure 5.2:AP-WBFS: Path Length Comparison for Study 1

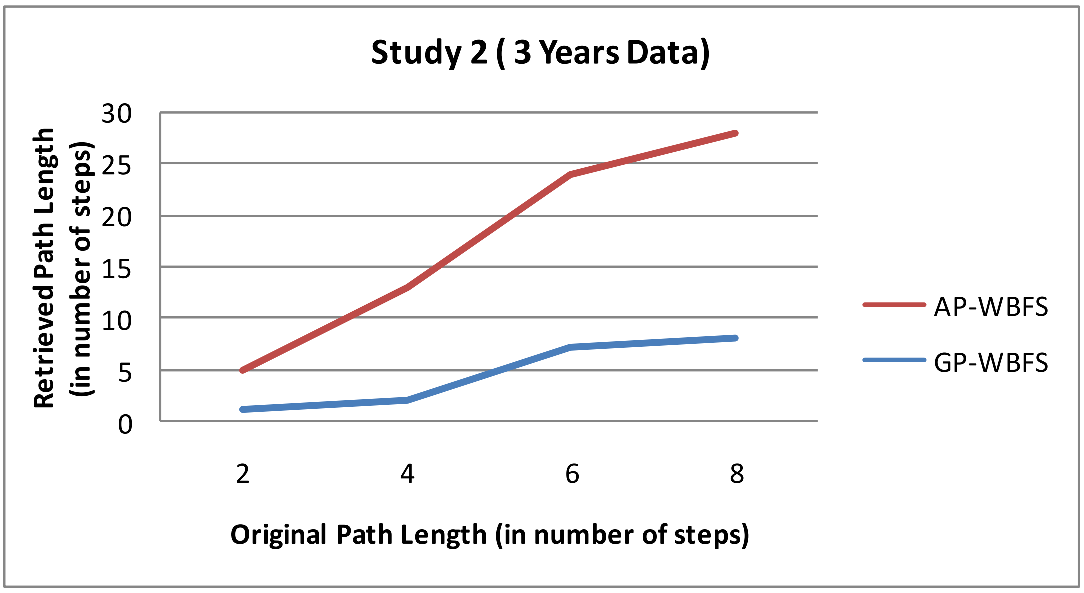


Figure 5.3: AP-WBFS: Path Length Comparison for Study 2

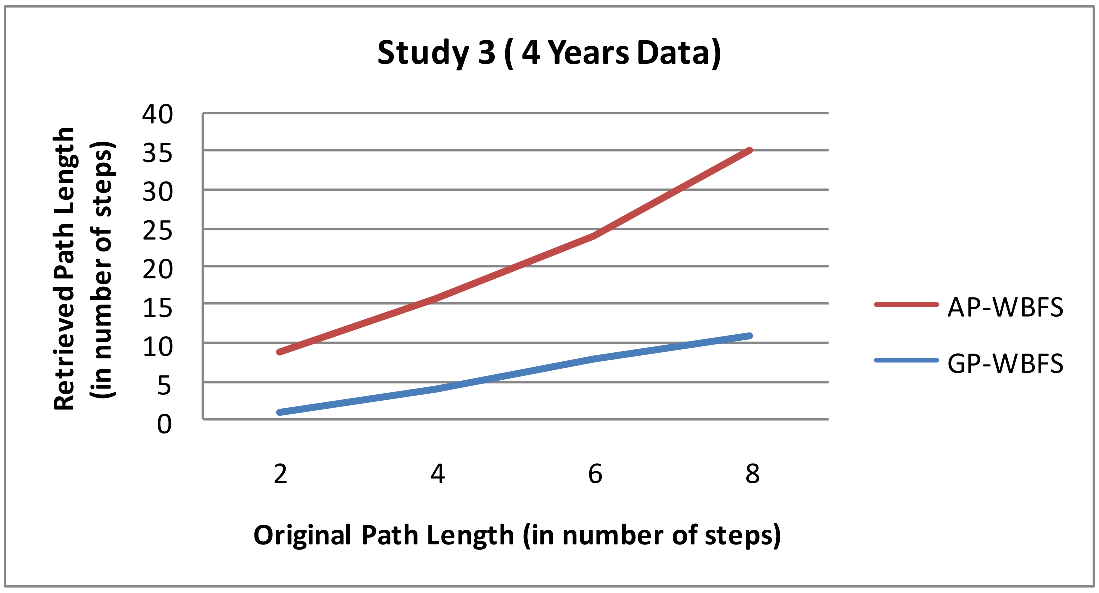


Figure 5.4: AP-WBFS: Path Length Comparison for Study 3

Hypothesis Evaluation with respect to P (Path Length) :

Table 5.7: T-test for GP-WBFS and AP-WBFS based on Path Length

Hypothesis	**Technique**		**Study**	**Path Length**
	I	**II**		**p value**
H1	GP-WBFS	AP-WBFS	Study 1	<0.0001
			Study 2	<0.0001
			Study 3	<0.0001

From the Table 5.7, it is concluded that the calculated significance level of the parameter Path Length by comparing the two systems namely, GP-WBFS, AP-WBFS satisfies the condition (p value<0.05) for all the three studies. There is a significant difference between the results for different Path Length values of the above two systems namely, GP-WBFS and AP-WBFS. Hence, the null hypothesis for H1 may be rejected. Further, since GP model captures the relation between any developer and the final resolver, the Path Length, or to be more precise, the number of developers returned by GP model is fewer than that of the AP model. The results obtained from the three studies are consolidated using the Weighted Average method. Based on which, the results are collated in Figure 5.5. The rate of increase in the Path Length in the AP model is 15.71 with respect to the GPmodel.

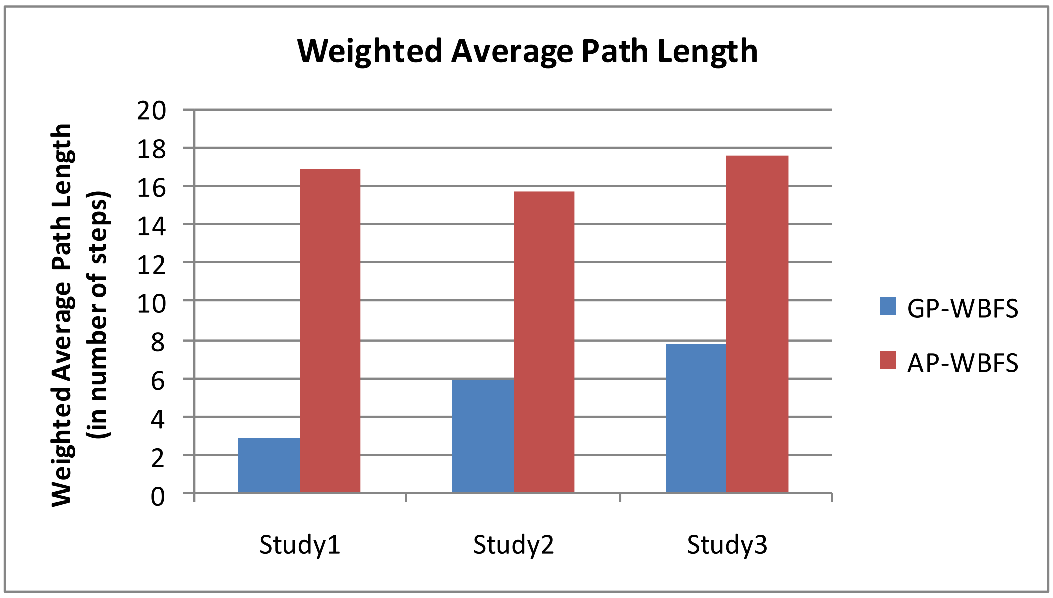


Figure 5.5: AP-WBFS: Weighted Average Path Length

5.2.4.2 Precision

Precision is the ratio of the number of matched developers in the retrieved path compared with the developers in the original path against the number of retrieved developers.

Hypothesis with respect to the result of the Precision PR: (Precision)

Null hypothesis H0: PR1 = PR2 where PR1= Precision obtained in GP-WBFS, PR2 = Precision obtained in AP-WBFS

(There is no significant difference between the two systems that is GP-WBFS and AP-WBFS in terms of Precision obtained)

Alternate hypothesis H2: Precision means values are not equal for at least one pair of the result mean values of the parameter PR.

(There is a significant difference between the two systems in terms of Precision obtained)

The Precision comparison from the experimental results is enumerated in Figure 5.6, Figure 5.7 and Figure 5.8. From the graphs, it is evident that the existing GP model gives superior results than the proposed AP model for longer original path length.

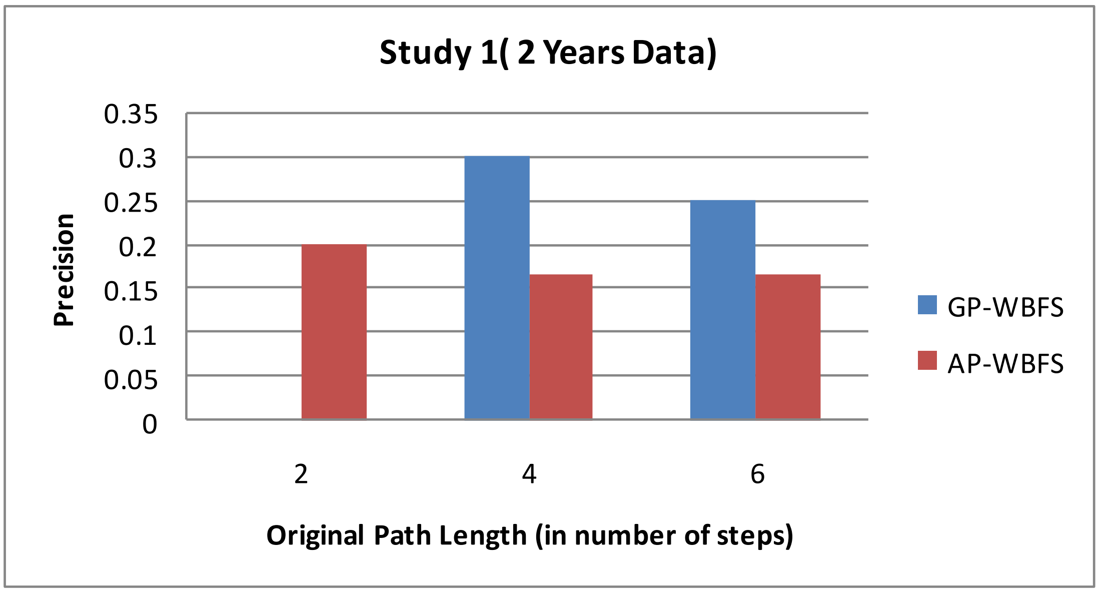


Figure 5.6: AP-WBFS: Precision Comparison for Study 1

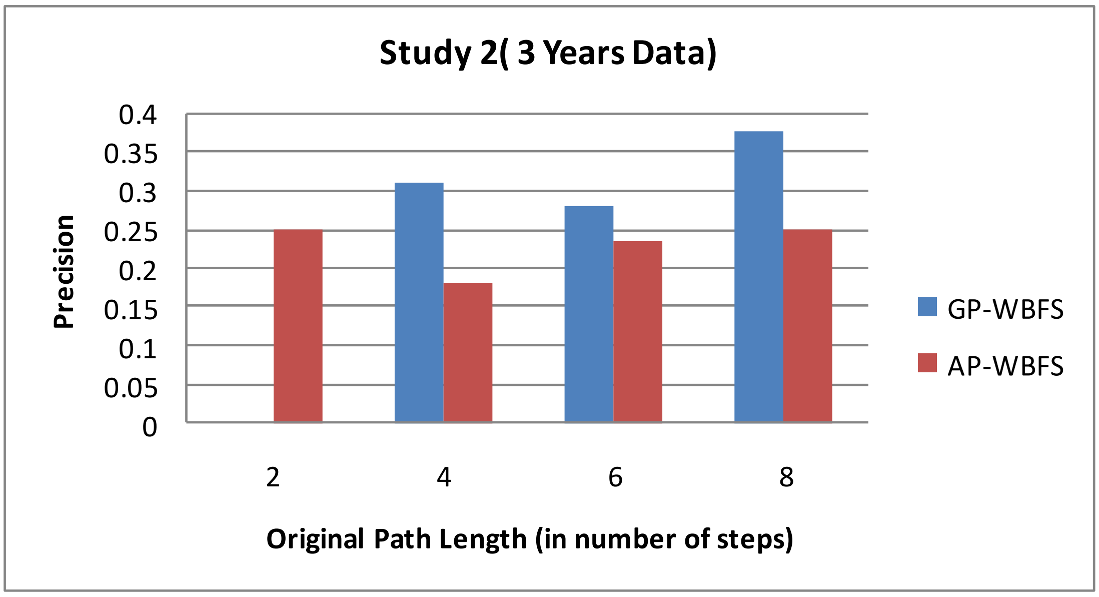


Figure 5.7: AP-WBFS: Precision Comparison for Study 2

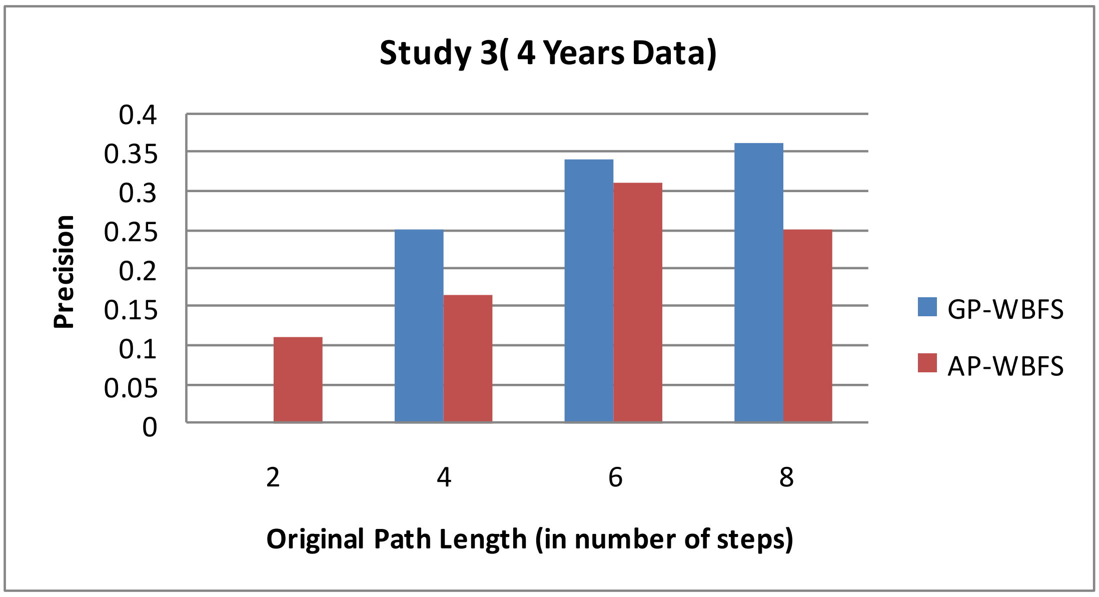


Figure 5.8: AP-WBFS: Precision Comparison for Study 3

Hypothesis Evaluation with respect to PR (Precision)

Table 5.8: T-test for GP-WBFS and AP-WBFS based on Precision

Hypothesis	**Technique**		**Study**	**Precision**
	I	**II**		**p value**
H2	GP-WBFS	AP-WBFS	Study 1	0.7709
			Study 2	0.5060
			Study 3	0.0973

From the Table 5.8, it is concluded that the calculated significance level of the parameter Precision by comparing the two systems namely, GP-WBFS, AP-WBFS does not satisfy the condition (p value<0.05) for all the three studies. Hence, the null hypothesis for H2 may be rejected.

The Weighted Average Precision of the three studies is specified in the Figure 5.9. Since the number of developers retrieved by the AP model is higher and the false positive rate to AP model is higher than the GP model, the Precision of AP model is less than that of GP model by -0.216.

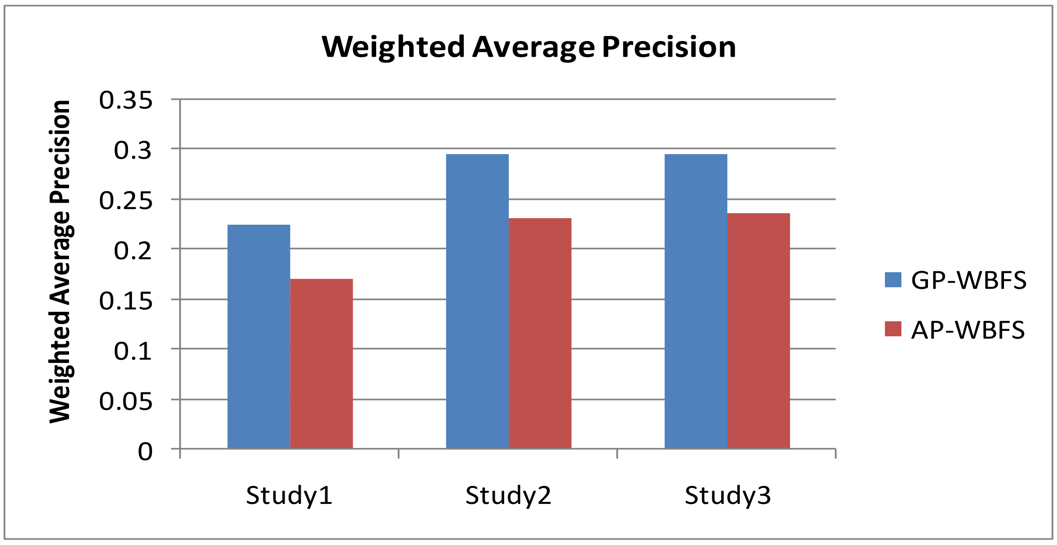


Figure 5.9: AP-WBFS: Weighted Average Precision

5.2.4.3 Recall

Recall is the ratio between the numbers of relevant developers retrieved to the total number of relevant developers.

Hypothesis with respect to the result of the Precision R: (Recall)

Null hypothesis H0: R1 = R2 where R1= Recall obtained in GP-WBFS, R2 = Recall obtained in AP-WBFS

(There is no significant difference between the two systems that is GP-WBFS and AP-WBFS in terms of Recall obtained)

Alternate hypothesis H3: Recall mean values are not equal for at least one pair of the result mean values of the parameter R.

(There is a significant difference between the two systems in terms of Recall obtained)

The Recall comparison from the experimental results is given in Figure 5.10, Figure 5.11 and Figure 5.12. From the graphs, it is evident that the existing AP model gives superior results than the proposed GP model.

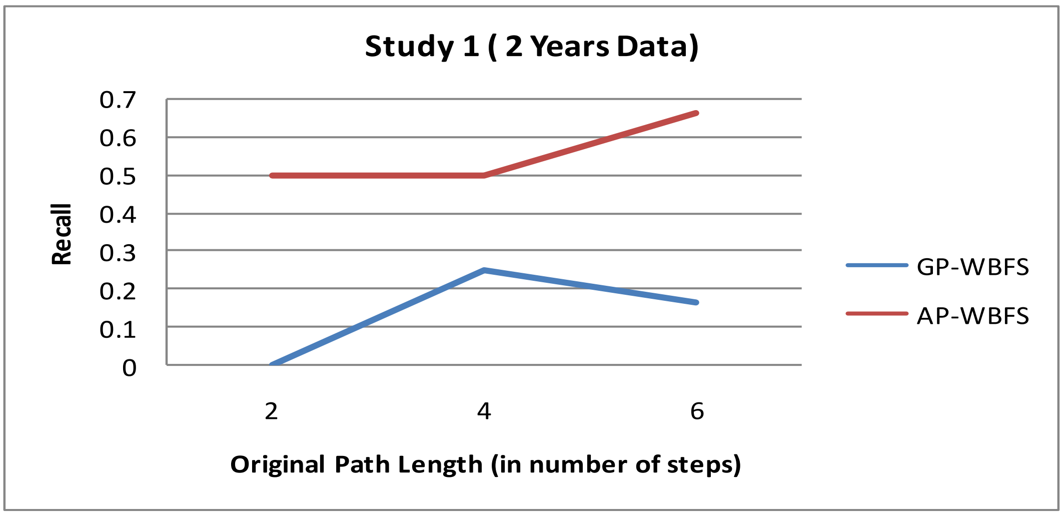


Figure 5.10: AP-WBFS: Recall Comparison for Study 1

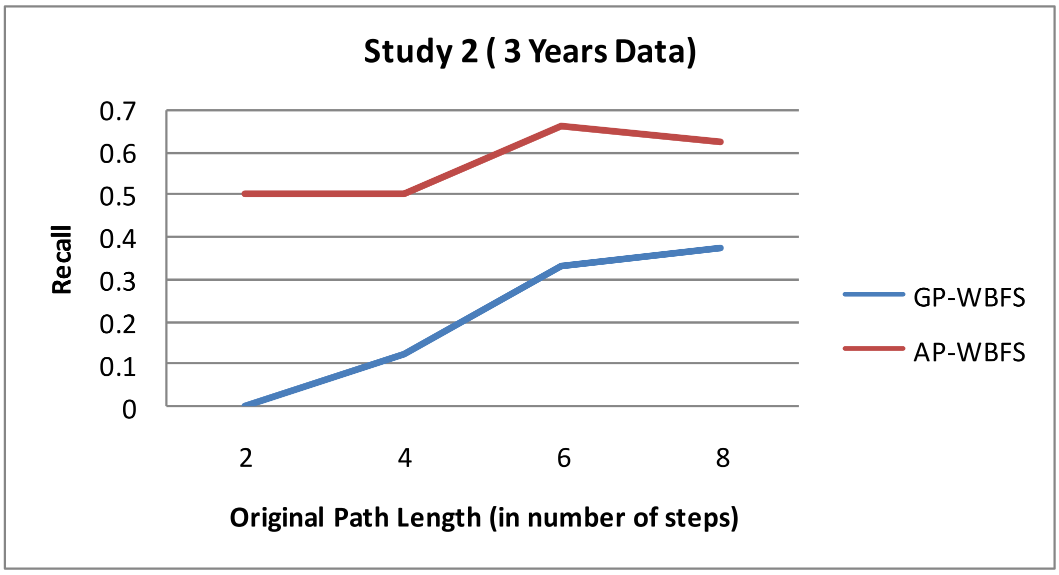


Figure 5.11: AP-WBFS: Recall Comparison for Study 2

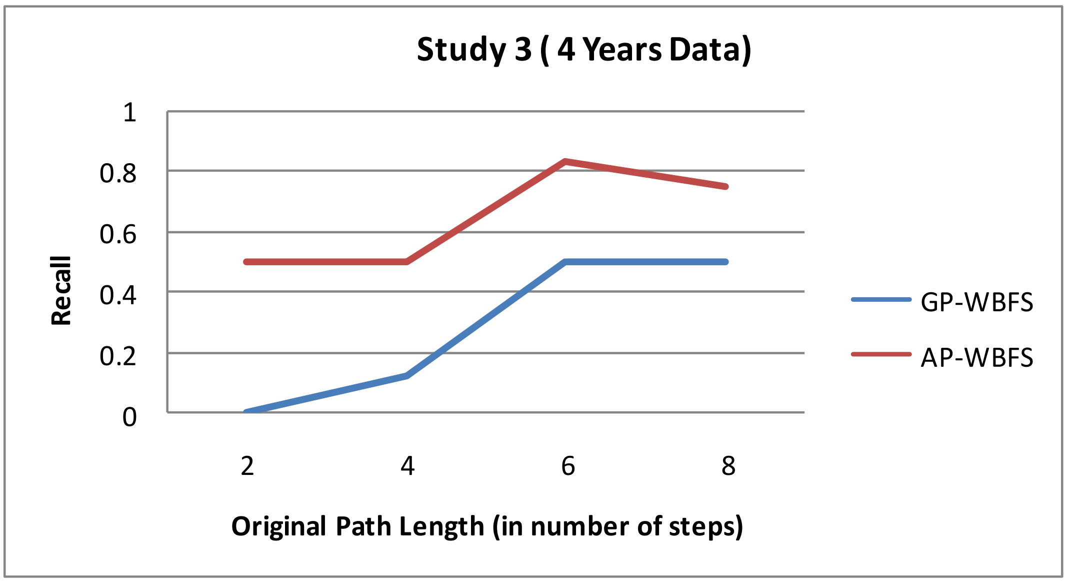


Figure 5.12: AP-WBFS: Recall Comparison for Study 3

Hypothesis Evaluation with respect to R (Recall)

Table 5.9: T-test for GP-WBFS and AP-WBFS based on Recall

Hypothesis	**Technique**		**Study**	**Recall**
	I	**II**		**p value**
H3	GP-WBFS	AP-WBFS	Study 1	<0.0001
			Study 2	<0.0001
			Study 3	<0.0001

From the Table 5.9, it is concluded that the calculated significance level of the parameter Recall by comparing the two systems namely, GP-WBFS and AP-WBFS satisfies the condition (p value<0.05) for all the three studies. There is a significant difference between the results for different Recall values of the above two systems namely, GP-WBFS and AP-WBFS for all the three studies. Hence, the null hypothesis for H3 may be rejected.

The Weighted Averaged Recall of the three studies is presented in the Figure 5.13. The performance of AP model when compared with GP model is increased by 1.302.

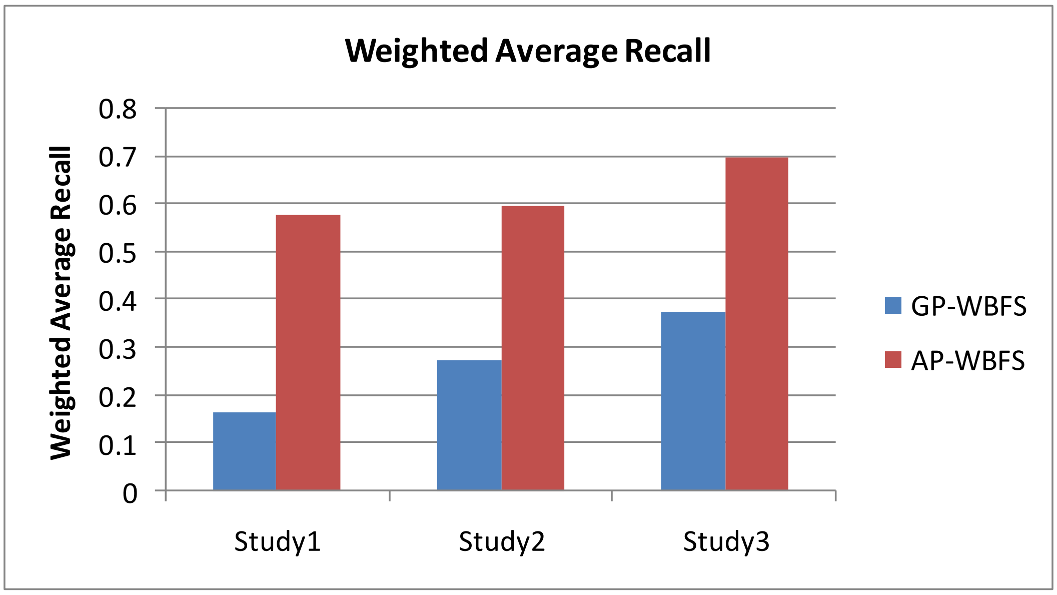


Figure 5.13: AP-WBFS: Weighted Average Recall

5.2.4.4 Path Similarity

Path Similarity encodes the retrieved developers in position and order when compared to the developers in the original path.

Hypothesis with respect to the result of the Precision PS: (Path Similarity)

Null hypothesis H0 : PS1 = PS2 where PS1= Path Similarity obtained in GP-WBFS, PS2 = Path Similarity obtained in AP-WBFS

(There is no significant difference between the two systems that is GP-WBFS and AP-WBFS in terms of Path Similarity obtained)

Alternate hypothesis H4: Path Similarity mean values are not equal for at least one pair of the result mean values of the parameter R.

(There is a significant difference between the two systems in terms of Path Similarity obtained)

The Path Similarity comparison from the experimental results is revealed in Figure 5.14, Figure 5.15 and Figure 5.16. From the graphs, it is evident that the existing AP model gives superior results than the proposed GP model.

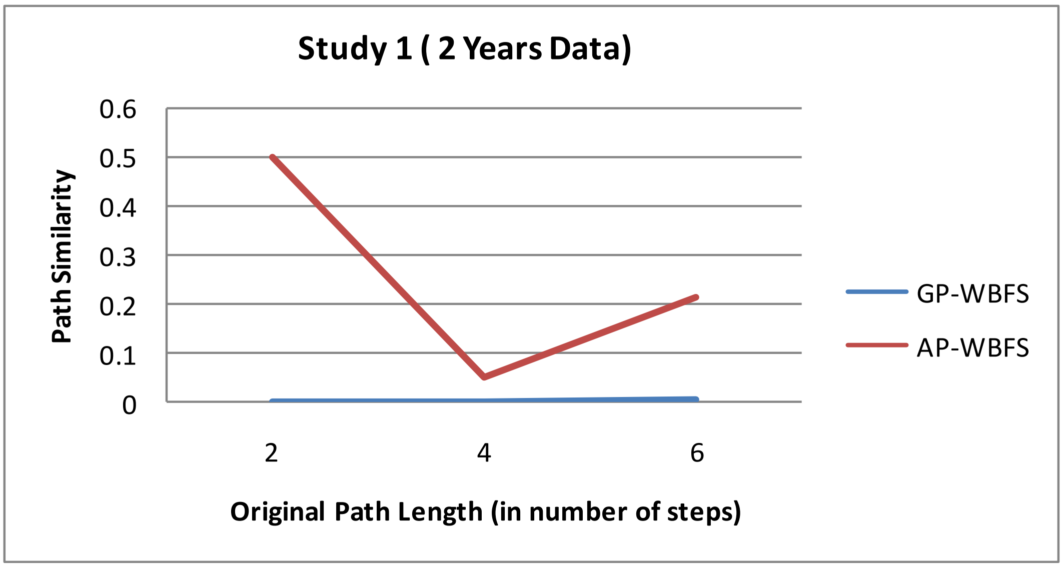


Figure 5.14: AP-WBFS: Path Similarity Comparison for Study 1

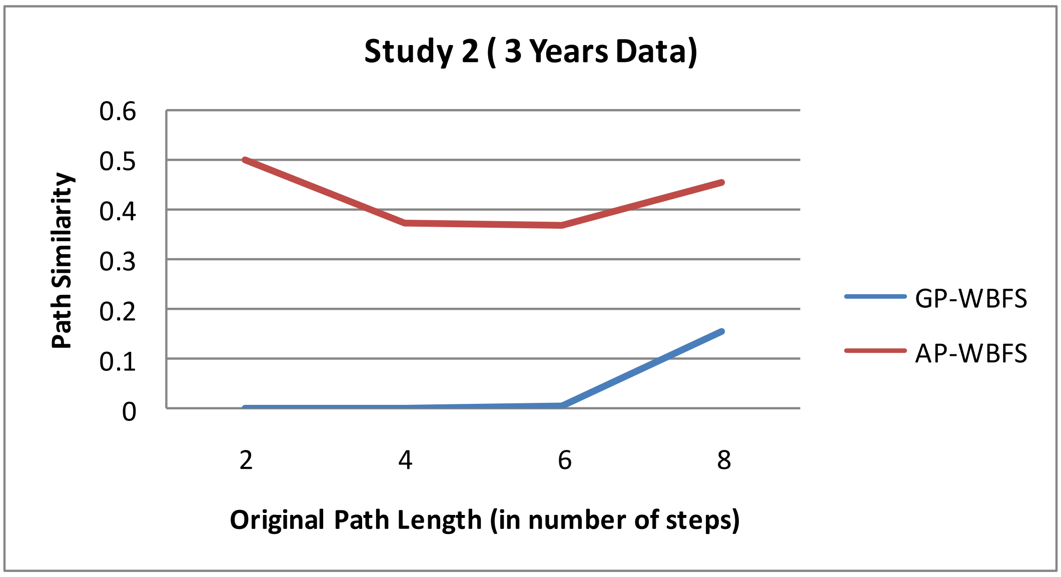


Figure 5.15: AP-WBFS: Path Similarity Comparison for Study 2

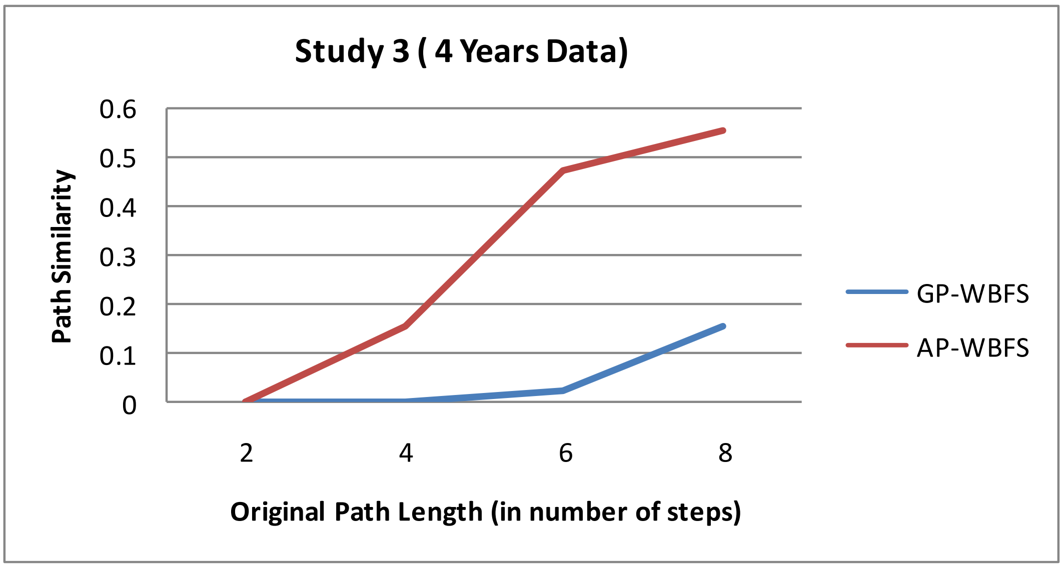


Figure 5.16: AP-WBFS: Path Similarity Comparison for Study 3

Hypothesis Evaluation with respect to PS: (Path Similarity)

Table 5.10: T-test for GP-WBFS and AP-WBFS based on Path Similarity

Hypothesis	Technique		Study	Path Similarity
	I	II		p value
H4	GP-WBFS	AP-WBFS	Study 1	<0.0001
			Study 2	<0.0001
			Study 3	<0.0001

From the Table 5.10, it is ascertained that the calculated significance level of the parameter Path Similarity by comparing the two systems namely, GP-WBFS and AP-WBFS satisfies the condition (p value<0.05) for all the three studies. There is a significant difference between the results for different Path Similarity values of the above two systems namely, GP-WBFS and AP-WBFS in all the three studies. Hence, the null hypothesis for H4 may be rejected. The Weighted Average Path Similarity of the three studies is apprised in the Figure 5.17. The increase in the performance of AP model when compared to GP model with respect to Path Similarity is5.939.

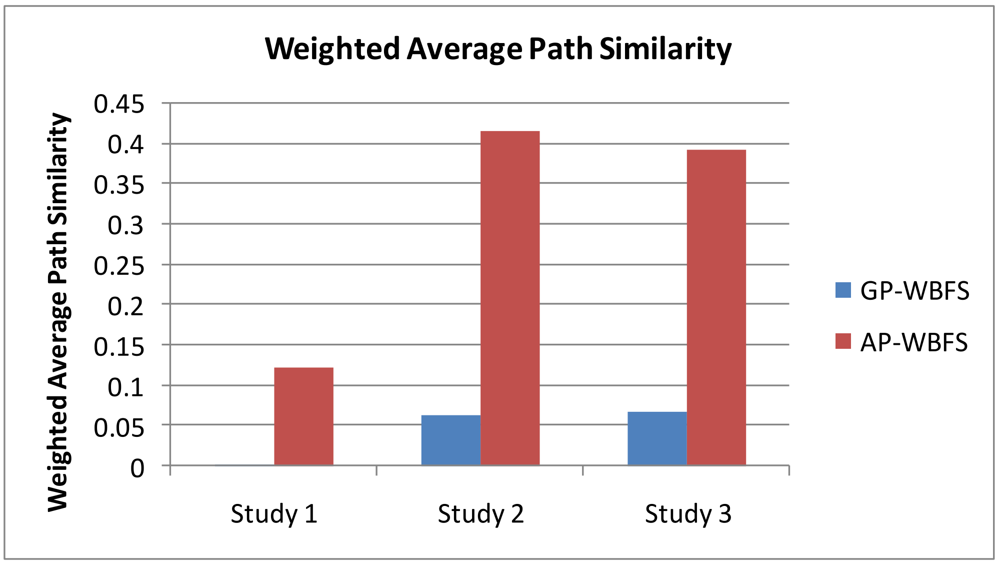


Figure 5.17:AP-WBFS: Weighted Average Path Similarity

The inference from the results is that the performance of the AP model can be improved with respect to Path length and Precision by improving the algorithm deployed on the Bug Toss Graph based on AP model. AP model captures the underlying developer structure which is invariably lost in the GP model. The WBFS algorithm uses the Bug Toss Graph as a static graph. But, since OSS itself is a volatile model of software development, the volatility percolates to the Bug Triaging also. Using a static graph in this milieu, presents undesirable results. On the other hand, updating the Bug Toss Graph for every new change in the underlying graph structure is also a costly affair. A nice middle road will be to employ adaptive learning techniques over the AP model based on Bug Toss Graph.

5.3 AN ENHANCED BUG TRIAGE SYSTEM BASED ON BI-OBJECTIVE OPTIMIZATION

In the existing work, the algorithm deployed on the Bug Toss Graph is based on WBFS algorithm. The Bug Toss Graph is a weighted directed graph, where the weights on each edge are the number of tosses. The objective of the WBFS algorithm is to retrieve a path with minimum number of tosses.

The Enhanced Bug Triage System formulates the Bug Triage problem as a bi-objective optimization problem. The two objectives are to minimize the distance between the first assignee and the final resolver and to include well connected developers in the retrieved path. The cost on each edge of the retrieved path is taken to be 1. The well connected developers are defined by their degree of connections. The degree of a developer subsumes the in-degree and out-degree. The in-degree indicates the number of developer who has made a toss to the developer and the out-degree indicates the number of developers to whom a toss has been made. The cost of traversing each edge is taken to be 1. Each developer who serves as a node is associated with a profit function, depending on the developer’s connectivity with the neighboring developers. The profit function depends on two values - the number of connections and the number of times a bug was tossed through the connection. The objective of the bi-objective optimization algorithm is to maximize the profit and minimize the cost [76], [77]. The underlying intuition is that, the algorithm should include as many well connected developers as possible, as well as the numbers of tosses the bug undergoes in order to be minimized. This is done by aggregating both the objectives into a single scalar function [78].

5.3.1 Formal Definition of the Problem

The Bug Toss Graph is a directed weighted graph G = (V,E) where n=|V| is the number of developers. The set ‘E’ is a set of edges .The cost ‘c_{ij}’ to traverse on each edge is taken to be 1. A profit ‘p_i’ is associated with each vertex in ‘V’ [76].

$\forall v_i \in V$

$y_i=0$ - All nodes are unvisited

$\forall v_i \in V$

$p_i = \sum_{i=0}^{indegree} \text{Tosses To}_i + \sum_{i=0}^{outdegree} \text{Tosses From}_i$- Initialize Profit values for each developer

$\forall e_{ij} \in E$

$x_{ij}=0$ - All edges are unvisited

Objective Function1 - Max $\sum c_{ij}x_{ij}$)

Objective Function2 - Min ($\sum p_i y_i$)

Aggregated Objective Function –Max($\sum p_i y_i$ - $\sum c_{ij}x_{ij}$)

5.3.2 Bug Triage with Bi-Objective Optimization Algorithm(Bi-Objective)

The algorithm for bi-objective optimization is portrayed in the Figure 5.18. The set ‘A’ is initialized to the set of first assignees. The set ‘R’ is initialized to the set of final resolvers. A developer from the set ‘A’ is chosen as the starting node. The starting node is assigned as the current node. The next developer is chosen from the neighborhood of the current node. The next developer is selected subject to the condition that the developer should be previously unvisited and the edge from the current developer to the next developer should be unvisited. The next developer with maximum profit is chosen among all the neighbors. The profit for that partial solution is updated. The developer is added to the solution set. The process is repeated until a developer belonging to set ‘R’ is reached.

```
Procedure Bi-Objective(BTG[0..i][0..j],N[0..i])
//BTG- Bug Toss Graph
//N-Contains the profit values for each developer
{A}- Set of first assignee
{R}-Set of top resolver
{S}- Set of Solutions
Neighbor{CURR}- List of nodes in the neighborhood of CURR
Begin
1:  E[0.k] ={0} //All edges are unvisited
2:  V[0..i]={0}// All nodes are unvisited
3:  while {A} !=NULL
4:        choose ST from A // Starting node from set of first assignees
5:        add ST to S
6:        CURR=ST
7:        PROFIT=0
8:        while CURR ∉ R
9:              i=0
10:             while(Neighbor{CURR}!=NULL)
11:                   max=0
12:                   $x_i$∈Neighbor{CURR}
13:                   if (V[i]==0 &&  E[i]==0) && max< N[i])
14:                         NEXT= xi
15:                         max=N[i]
16:                   endif
17:                   increment i
18:             endwhile// Neighbor{CURR}!=NULL
19:             PROFIT=PROFIT +N[i]
20:             add NEXT to S
21:             CURR=NEXT
22:       endwhile // CURR ∉ R
23:       CUMPROFIT=PROFIT – LEN(Solution)
24:endwhile //{A} !=NULL
25:return the Solution with Max(CUMPROFIT)
End
```

Figure 5.18: Bug Triage with Bi-Objective Optimization Algorithm

The cumulative profit of the solution is computed by deducting the solution path length from the profit accumulated. The same process is repeated for all the nodes from set ‘A’. The solution that has the maximum cumulative profit is chosen to be the best set of developers who can collaborate on a bug.

5.3.3 Performance Evaluation

The performance of the Bug Triage System based on Bi-objective optimization is compared with the existing GP-WBFS as well as AP-WBFS considering the parameters: Path length, Precision, Recall and Path Similarity. The experiments were repeated for three runs. The experimental performance of the three systems was statistically analyzed using ANOVA.

5.3.3.1 Path Length

The experimental results of parameter Path Length is shown in the Figure 5.19, Figure 5.20 and Figure 5.21. It can be observed from the experimental results that Bi-objective outweighs the proposed AP-WBFS. But still, the existing GP-WBFS produces the best results with respect to Path Length. This is due to the fact that, GP-WBFS takes into consideration only the tosses from any developer to the final resolver. This is the reason for the fact that the connectivity to the final resolver is much dense in the GP model. This connectivity contributes to the reduced path length for GP-WBFS.

Hypothesis with respect to the result of the parameter P: (Path Length)

Null hypothesis H0: P1 = P2=P3 where P1= Path Length obtained in GP-WBFS, P2 = Path Length obtained in AP-WBFS, P3= Path Length obtained in Bi-Objective.

(There is no significant difference among the three systems: that is GP-WBFS, AP-WBFS and B-Objective in terms of Path Length obtained)

Alternate hypothesis H5: Path Length mean values are not equal for at least one pair of the result mean values of the parameter P.

(There is a significant difference among the three systems in terms of Path Length obtained).

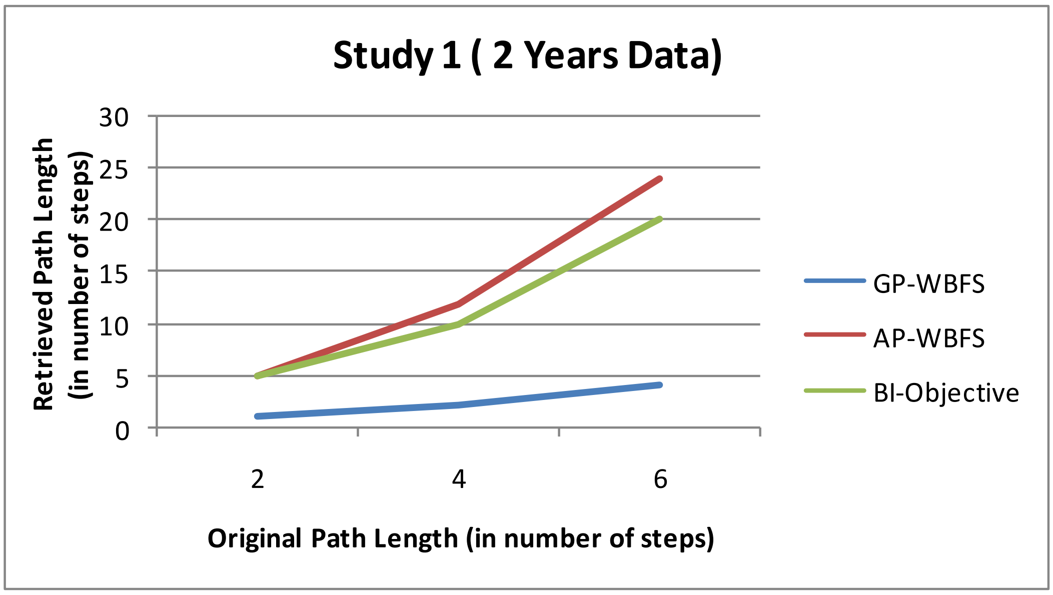


Figure 5.19: Bi-Objective: Path Length of Study 1

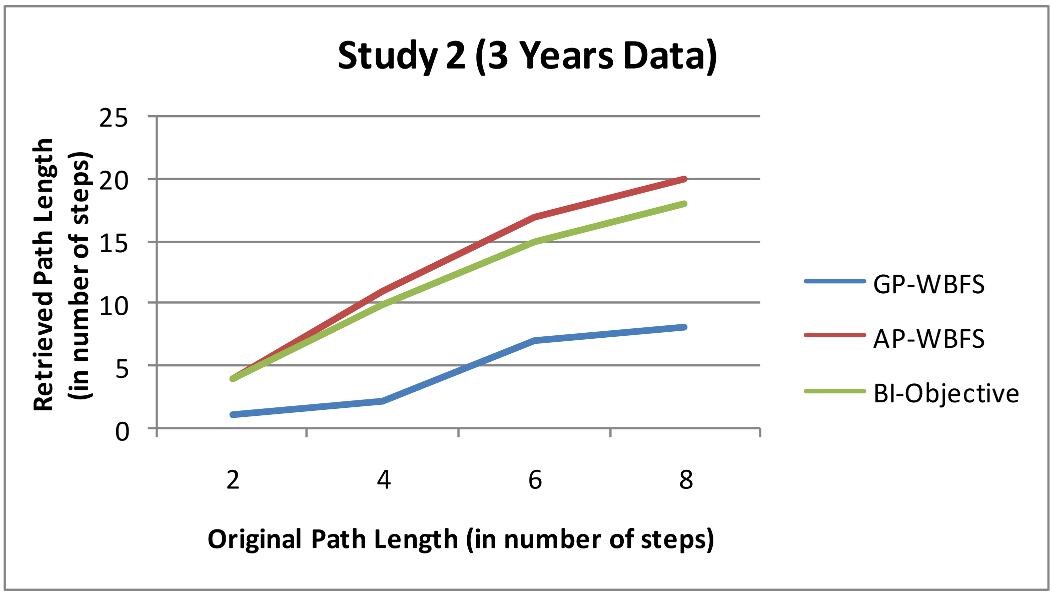


Figure 5.20: Bi-Objective : Path Length Study 2

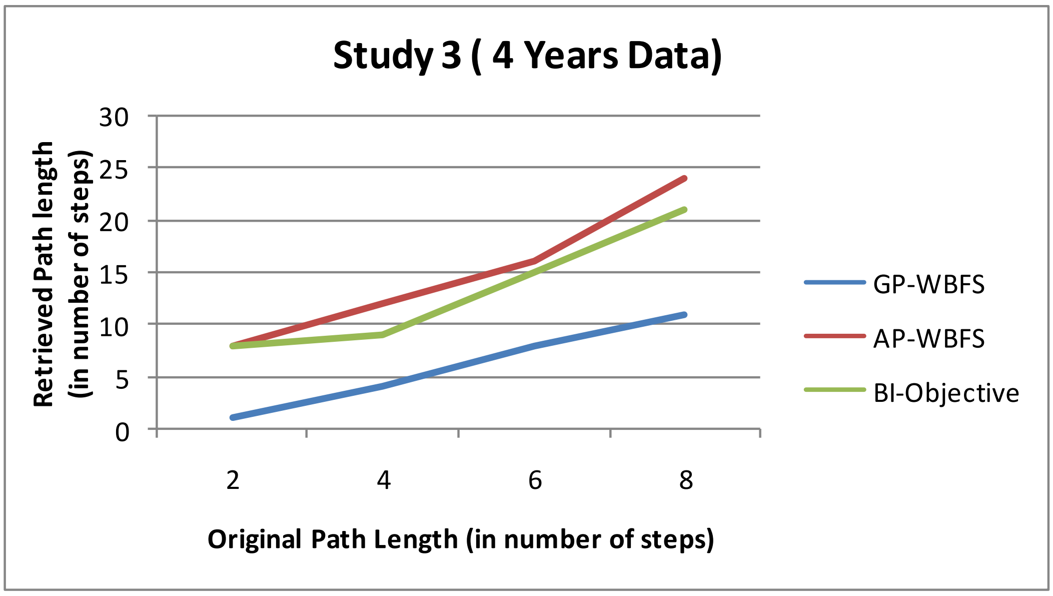


Figure 5.21: Bi-Objective : Path Length Study 3

Hypothesis Evaluation with respect to P (Path Length)

Table 5.11: ANOVA for GP-WBFS,AP-WBFS and Bi-Objective based on Path Length

Hypothesis	**Technique**			**Study**	**Path Length**
	I	**II**	**III**		**p value**
H5	GP-WBFS	AP-WBFS	Bi-Objective	**Study 1**	<0.0001
				Study 2	<0.0001
				Study 3	<0.0001

From the Table 5.11, it is concluded that the calculated significance level of the parameter-Path Length by comparing the three systems namely, GP-WBFS,AP-WBFS and Bi-Objective always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Path Length values of the above three systems namely, GP-WBFS, AP-WBFS and Bi-Objective. Hence, the null hypothesis for H5 may be rejected. The results obtained from the three studies are combined using the Weighted Average method, based on which the results are consolidated in the Figure 5.22.

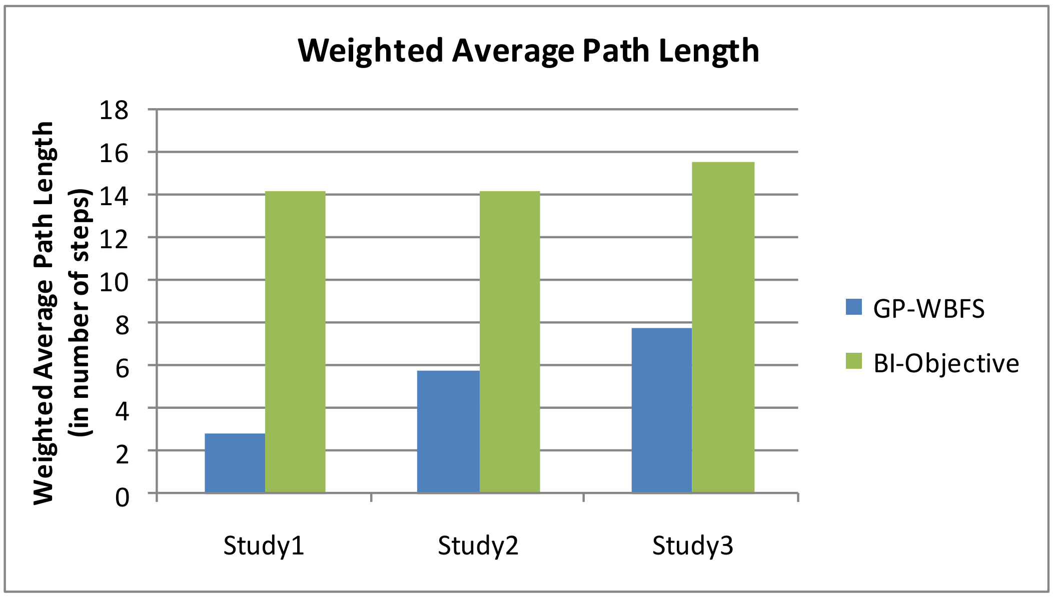


Figure 5.22: Bi-Objective : Weighted Average Path Length

The rate of increase in the Path Length in the Bi-Objective with respect to the existing GP-WBFS is 13.588.

5.3.3.2 Precision

The experimental results of parameter Precision is displayed in the Figure 5.23, Figure 5.24 and Figure 5.25.It can be observed from the experimental results that Bi-Objective performs better than the proposed AP-WBFS. But still, the existing GP-WBFS produces the best results with respect to the parameter: Precision in particular for original Path Lengths 6 and 8. This is because, Precision is the ratio between the matched developers and the number of retrieved developers. Since, the numbers of retrieved developers are more in number for Bi-Objective and AP-WBFS, the parameter: Precision is less when compared to GP-WBFS.

Hypothesis with respect to the result of the Precision PR (Precision)

Null hypothesis H0: PR1 = PR2 = PR3 where PR1= Precision obtained in GP-WBFS , PR2 = Precision obtained in AP-WBFS , PR3 = Precision obtained in Bi-Objective.

(There is no significant difference among the three systems, that is GP-WBFS, AP-WBFS and Bi-Objective in Precision obtained)

Alternate hypothesis H6: Precision mean values are not equal for at least one pair of the result mean values of the parameter PR.

(There is a significant difference among the three systems in terms of Precision obtained)

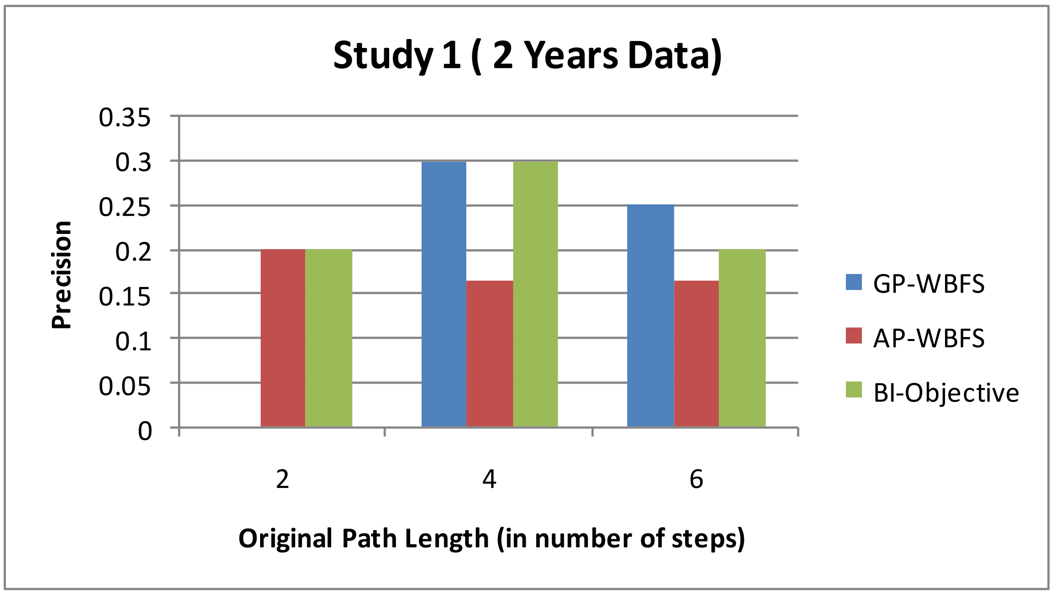


Figure 5.23: Bi-Objective: Precision of Study 1

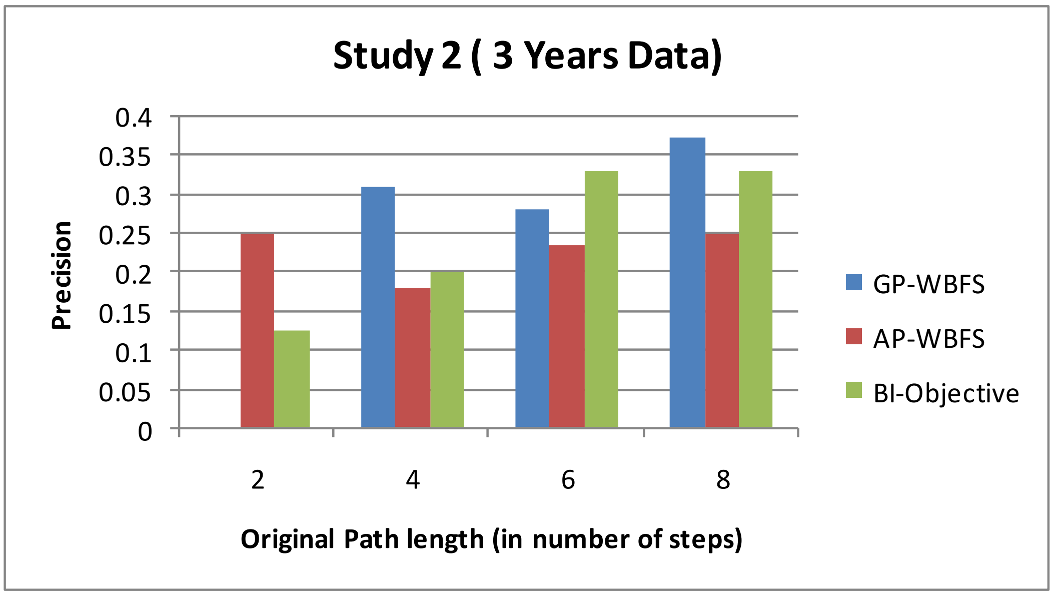


Figure 5.24: Bi-Objective: Precision of Study 2

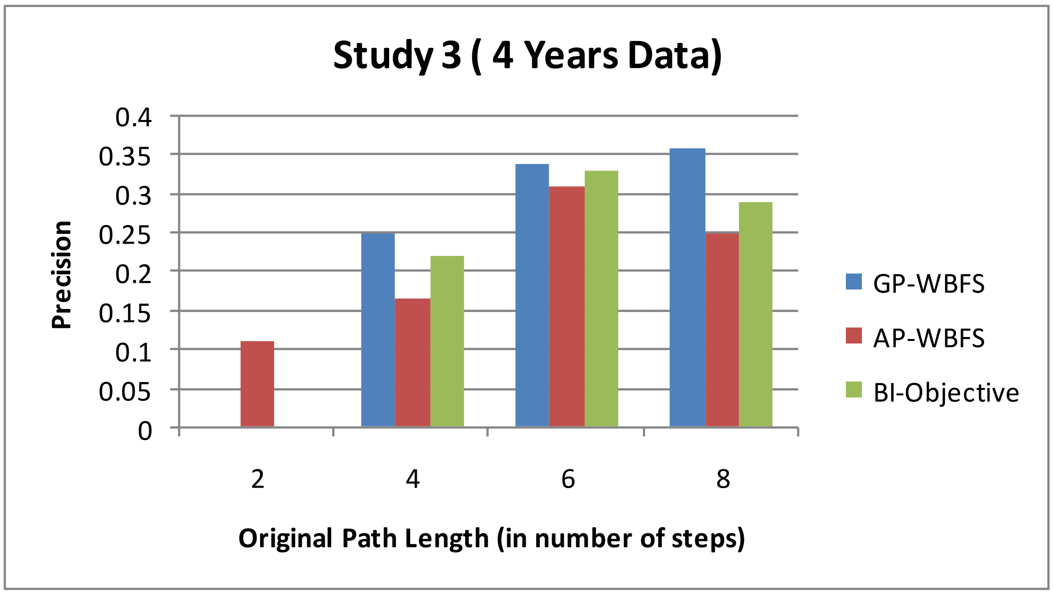


Figure 5.25: Bi-Objective: Precision of Study 3

Hypothesis Evaluation with respect to PR (Precision)

Table 5.12: ANOVA for GP-WBFS, AP-WBFS and Bi-Objective based on Precision

Hypothesis	Technique			Study	Precision
	I	II	III		p value
H6	GP-WBFS	AP-WBFS	Bi-Objective	**Study 1**	<0.0001
				Study 2	<0.0001
				Study 3	<0.0001

From the Table 5.12, it is asserted that the calculated significance level of the parameter-Precision by comparing the three systems namely, GP-WBFS, AP-WBFS and Bi-Objective always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Precision values of the above three systems namely, GP-WBFS, AP-WBFS and Bi-Objective for the three studies. Hence, the null hypothesis for H6 may be rejected.

The results obtained from the three studies are consolidated using the Weighted Average method. Based on which, the results are consolidated in the Figure 5.26.

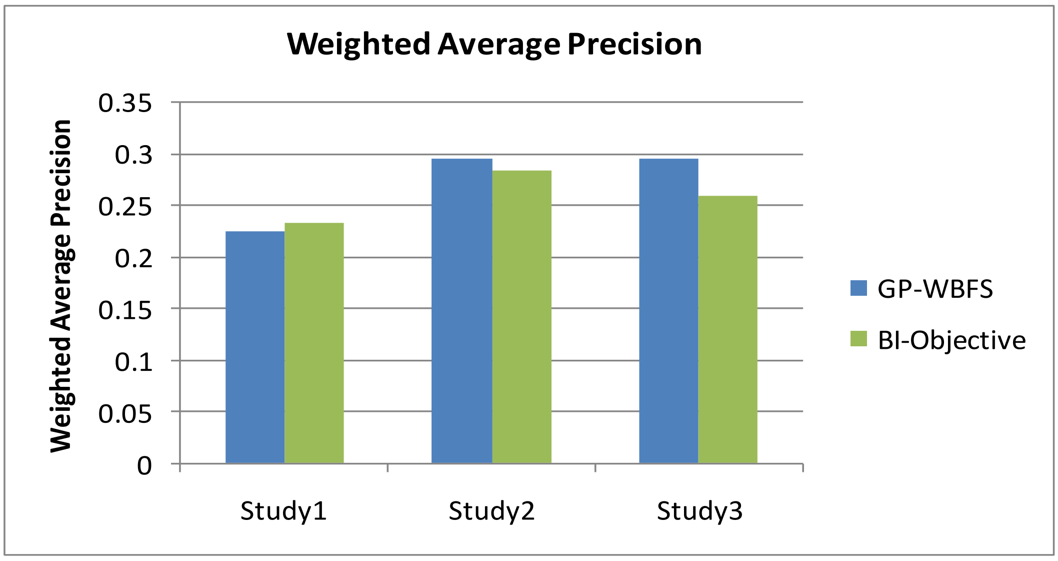


Figure 5.26: Bi-Objective: Weighted Average Precision

The rate of increase in Precision in the Bi-Objective with respect to the existing GP-WBFS is -0.05.

5.3.3.3 Recall

The experimental results of parameter - Recall are divulged in the Figure 5.27, Figure 5.28 and Figure 5.29. It can be observed from the experimental results that Bi-Objective performs better than the proposed AP-WBFS and the existing GP-WBFS.This is due to the following reason: Recall is the ratio between the matched developers and the number of developers in the original path.Since, the numbers of retrieved developers are more in number for Bi-Objective, the Recall value is higher for Bi-Objective.

Hypothesis with respect to the result of the Precision R (Recall)

Null hypothesis H0: R1 = R2 = R3 where R1= Recall obtained in GP-WBFS, R2 = Recall obtained in AP-WBFS, R3 = Recall obtained in Bi-Objective

(There is no significant difference among the three systems, that is GP-WBFS, AP-WBFS Recall and Bi-Objective in terms of Recall obtained)

Alternate hypothesis H7: Recall mean values are not equal for at least one pair of the result mean values of the parameter R.

(There is a significant difference among the three systems in terms of Recall obtained)

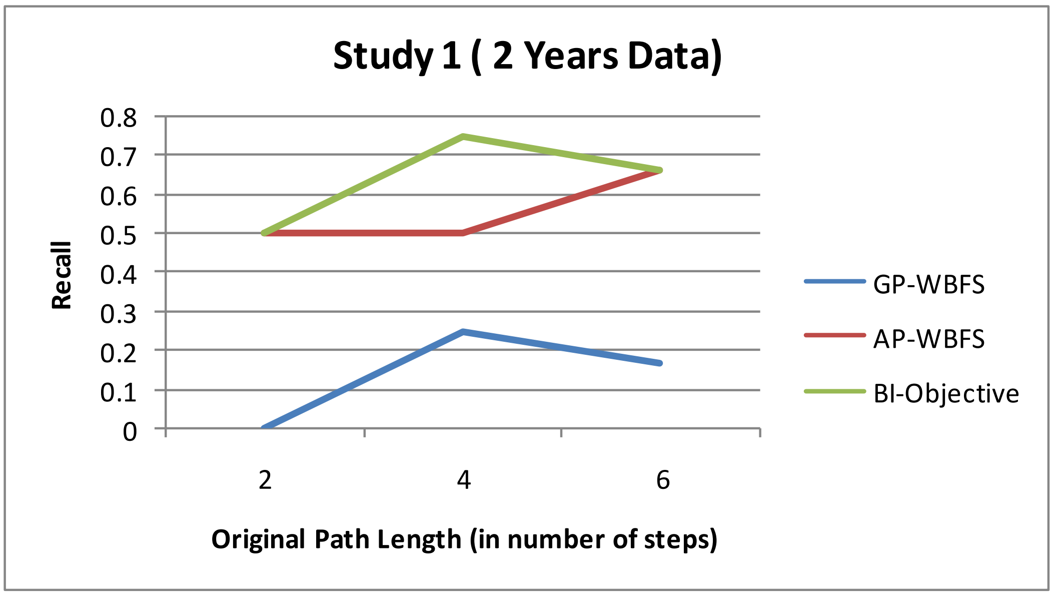


Figure 5.27: Bi-Objective: Recall of Study 1

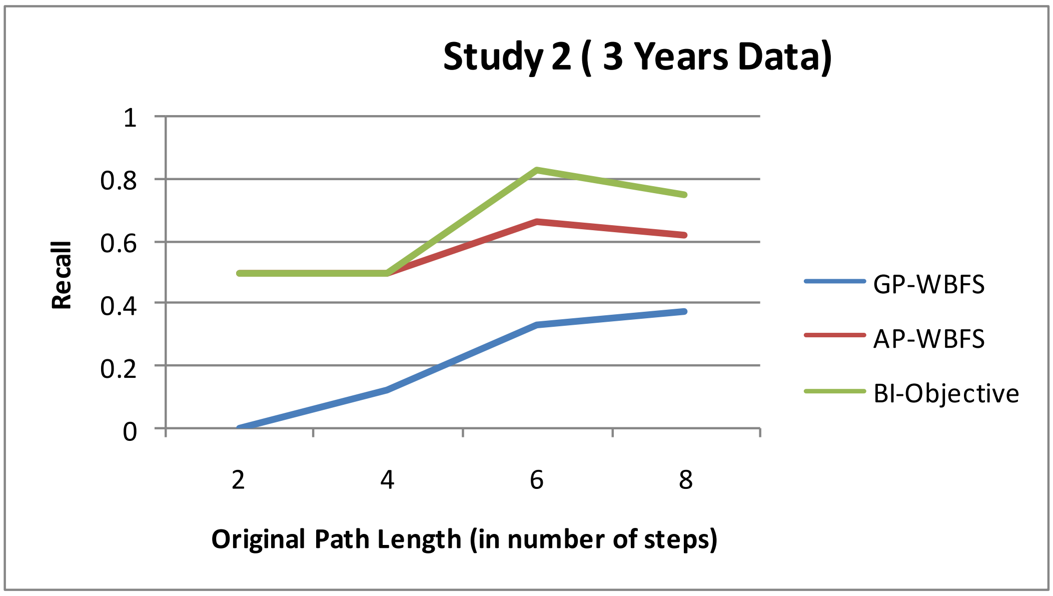


Figure 5.28: Bi-Objective: Recall of Study 2

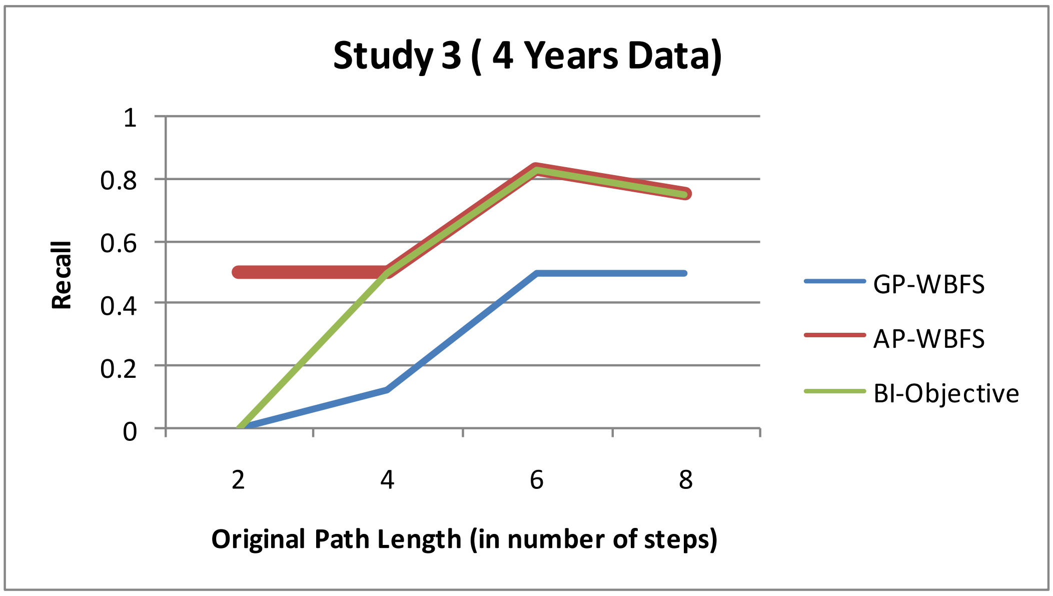


Figure 5.29: Bi-Objective: Recall of Study 3

Hypothesis Evaluation with respect to R: (Recall)

Table 5.13: ANOVA for GP-WBFS, AP-WBFS and Bi-Objective based on Recall

Hypothesis	**Technique**			**Study**	**Recall**
	I	**II**	**III**		**p value**
H7	GP-WBFS	AP-WBFS	Bi-Objective	**Study 1**	<0.0001
				Study 2	<0.0001
				Study 3	<0.0001

From the Table 5.13, it is concluded that the calculated significance level of the parameter- Recall by comparing the three systems namely, GP-WBFS, AP-WBFS and Bi-Objective always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Recall values of the above three systems namely, GP-WBFS, AP-WBFS and Bi-Objective. Hence, the null hypothesis for H7 may be rejected.

The results obtained from the three studies are consolidated using the Weighted Average method. Based on which the results are consolidated in the Figure 5.30.

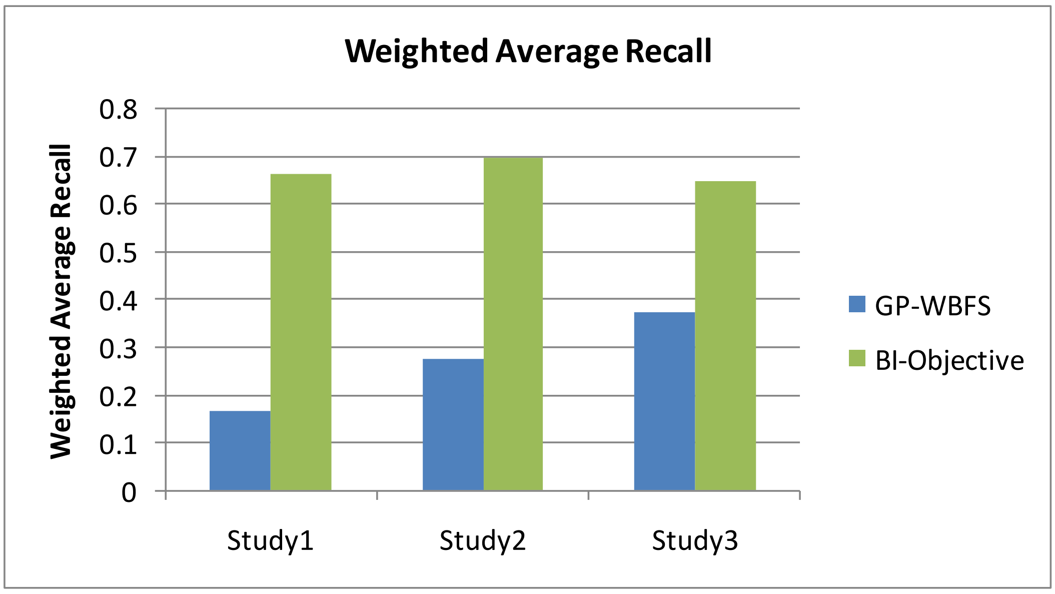


Figure 5.30: Bi-Objective: Weighted Average Recall

The rate of increase in Recall in the Bi-Objective with respect to the existing GP-WBFS is 1.469.

5.3.3.4 Path Similarity

The experimental results of parameter- Path Similarity is given in Figure 5.31, Figure 5.32 and Figure 5.33. It can be observed from the experimental results that Bi-Objective performs better than the proposed AP-WBF and the existing GP-WBFS.

Hypothesis with respect to the result of the Precision PS (Path Similarity)

Null hypothesis H0: PS1 = PS2= PS3where PS1= Path Similarity obtained in GP-WBFS, PS2 = Path Similarity obtained in AP-WBFS , PS3 = Path Similarity obtained in Bi-Objective.

(There is no significant difference among the three systems that is GP-WBFS, AP-WBFS and Bi-Objective in terms of Path Similarity obtained)

Alternate hypothesis H8: Path Similarity mean values are not equal for at least one pair of the result mean values of the parameter R.

(There is a significant difference among the three systems in terms of Path Similarity obtained).

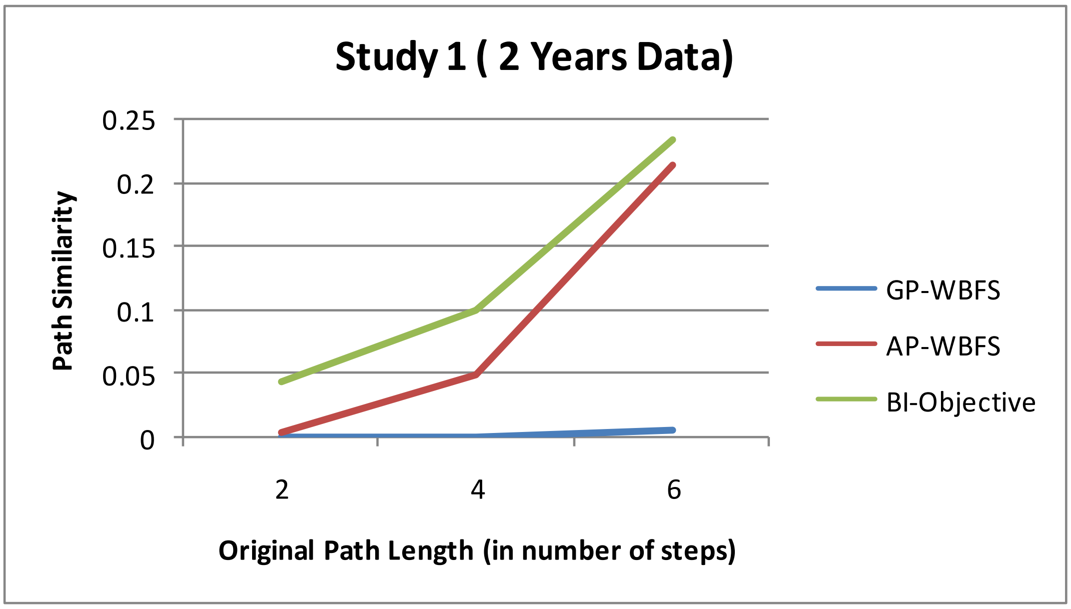


Figure 5.31: Bi-Objective: Path Similarity of Study 1

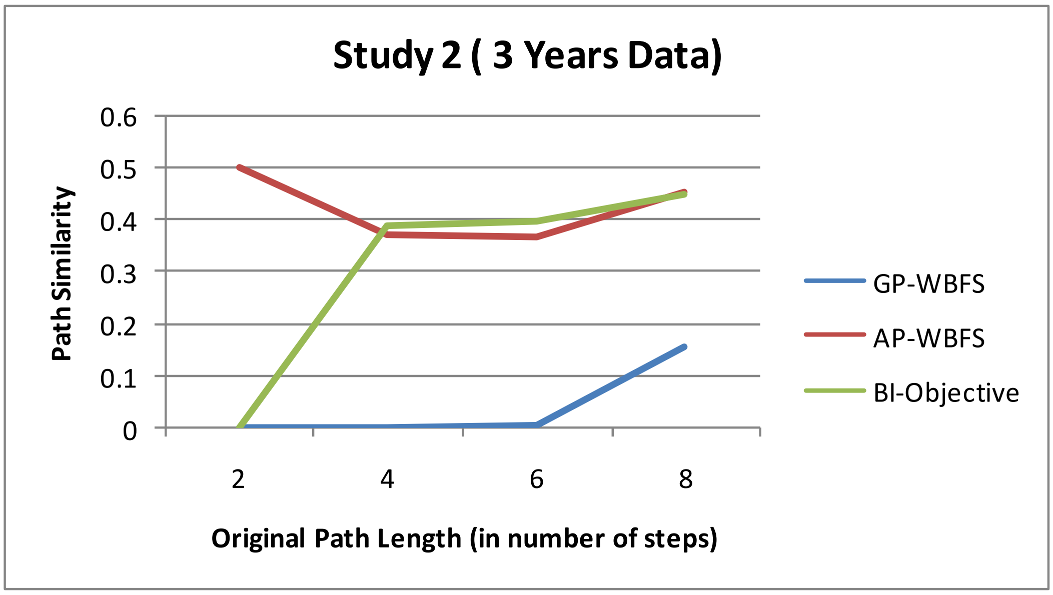


Figure 5.32: Bi-Objective: Path Similarity of Study 2

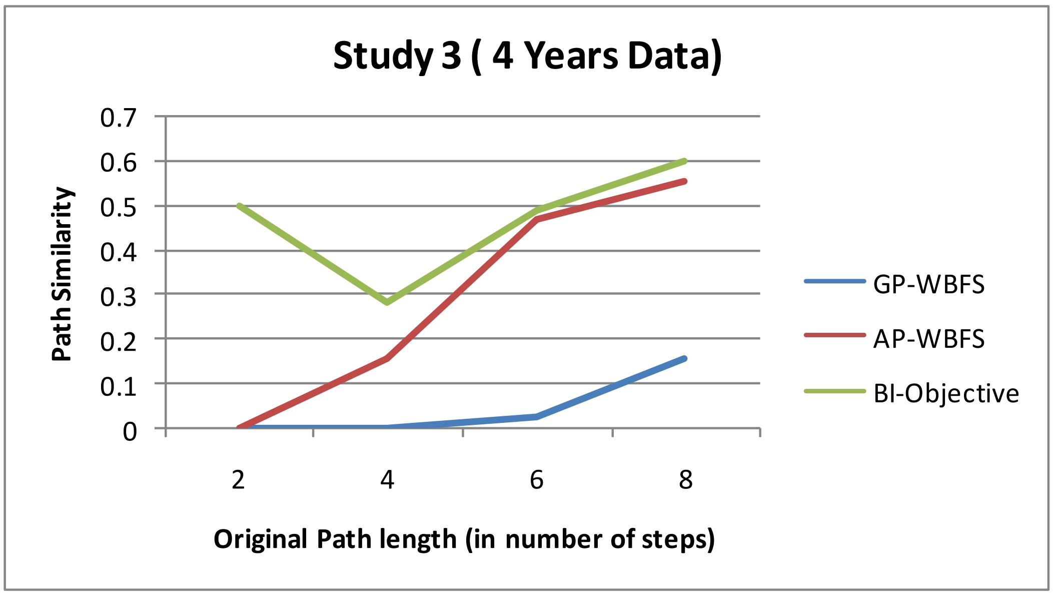


Figure 5.33: Bi-Objective: Path Similarity of Study 3

Hypothesis Evaluation with respect to PS (Path Similarity)

Table 5.14: ANOVA for GP-WBFS, AP-WBFS and Bi-Objective based on Path Similarity

Hypothesis	**Technique**			**Study**	**Recall**
	I	**II**	**III**		**p value**
H8	GP-WBFS	AP-WBFS	Bi-Objective	**Study 1**	<0.0001
				Study 2	<0.0001
				Study 3	<0.0001

From the Table 5.14, it is concluded that the calculated significance level of the parameter-Path Similarity by comparing the three systems namely, GP-WBFS,AP-WBFS and Bi-Objective always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Path Similarity values of the above three systems namely, GP-WBFS, AP-WBFS and Bi-Objective. Hence, the null hypothesis for H8 may be rejected.

The results obtained from the three studies are consolidated using the Weighted Average method. Based on which, the results are consolidated in the Figure 5.34.

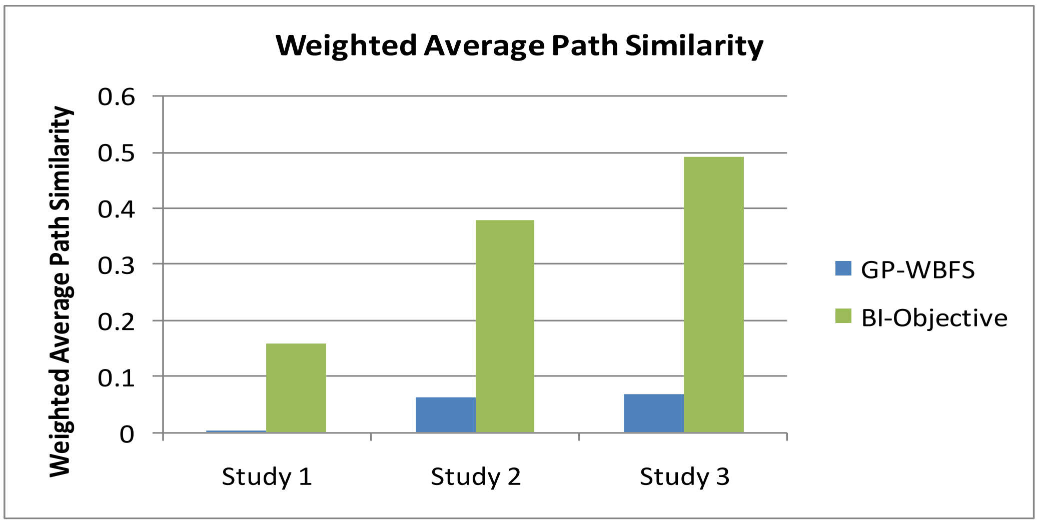


Figure 5.34: Bi-Objective: Weighted Average Path Similarity

The rate of increase in Path Similarity of Bi-Objective when compared with GP-WBFS is 6.657.

5.4 SUMMARY

This chapter chronicles a comprehensive analysis of AP model over the GP model with respect to four parameters viz.,Path length, Precision, Recall and Path Similarity. The AP model codifies in it every intermediate toss that happens in the lifetime of a bug. When the target of the Bug Triage System is to retrieve the referral chain of developers rather than only one developer who can solve the bug, intuitively, AP model is a promising candidate. To corroborate the intuition, experimental analysis was necessary to examine the performance of the AP model. The results presented after the experimental evaluation clearly indicate that the existing GP model outperforms the proposed AP model for the parameters: Path Length and Precision, whereas the AP model outperforms the GP model for the parameters: Recall and Path Similarity. The statistical analysis of the results obtained confirms that there is a significant difference in the performance of GP-WBFS from AP-WBFRS with respect to the parameters: Path Length, Recall and Path Similarity.

This chapter also describes an enhanced Bug Triage System based Bi-objective optimization. The Bi-Objective method exhibits improved performance over the existing GP-WBFS with respect to the parameters of Recall and Path Similarity. To improve the performance of the Bug Triage System based on AP model with respect to the parameters: Path length as well as Precision, adaptive techniques need to be employed.

CHAPTER - 6

ADAPTIVE TECHNIQUES WITH ENRICHED COLLABORATION GRAPH

6.1 PREAMBLE

This chapter makes a two fold contribution. On one hand, it presents adaptive techniques to be deployed on the Bug Toss Graph, on the other hand, it visualizes the Bug Triage as a social activity. The adaptive techniques are based on Ant Systems. Since,Bug Triage is a social activity; the underlying graph that captures the relation among the developers is modeled as an Enriched Collaboration Graph which captures the social context of the developers involved in resolving the bug. Over this Enriched Collaboration Graph, the multi objective ant systems are deployed to retrieve the developers who can form the referral chain.

6.2 NEED FOR ANT SYSTEM IN BUG TRIAGE

The growing size of the OSS and the unique attributes of OSS development system make it as arduous for software maintenance.This is due to the following facts [79]:

i) There is no centralized control.

ii) The participation and contribution is voluntary

iii) The organizational structure in the software development is informal, complex, self-organizing and not explicit.

iv) The collaboration between the developers is in the form of discussion forums and emails.

v) The motivation is peer respect.

Bug Triage is a vital facet of OSS corrective maintenance. Interestingly, the bugs reported in an OSS garners more respect than its counterparts in enterprise system. This is because the bugs reported in OSS are by the fellow developers and not the users of the system. As discussed earlier, the currently available Bug Triage System exploits the textual contents present in the bug report and the link information due to the tossing of bugs between the developers. The tossing activity that exists among the developers is embodied in the Bug Toss Graph. The current techniques use the Bug Toss Graph as a static graph. But, the static Bug Toss Graph loses the dynamicity inherent to the OSS development system. In addition, online learning techniques that call for updating the graph structure for every change in the developer is also not practical due to the cost concerns. In this scenario, ant systems are suitable for capturing the evolving nature of the developers in Bug Triage in OSS. Ant systems are applied widely in domains which are of adaptive nature.

6.2.1 Overview of Ant System

Swarm intelligence is an area of research where swarm of primitive animal's exhibit intelligent behavior.A swarm of ants foraging for food shows remarkable intelligence and capability in finding shortest path to a food source. A collection of ants cooperate in finding food using a chemical pheromone. This trail of pheromone is used for indirect communication among the ants. Ants also exhibit the property of adapting to environments. They are able to tackle obstacles on their path. This emergent property of ants has propelled the creation of artificial ant colonies [80].

Further, there is no supervisor to coordinate the ants. The self organizing concept is provided by the feedback mechanism. There are two types of feedback- positive and negative feedback. The positive feedback mechanism is provided by the deposition of the pheromone trails. Ants can detect the pheromone trail and exhibit a tendency to follow the pheromone trail laid by the other ant agents. The new set of ants again deposits pheromone on their behalf on the same path. The new set of pheromone reinforces the pheromone trail which in turn encourages other new ants to follow the same trail. Negative feedback counters the positive feedback. The negative feedback is provided by the evaporation of the pheromone trail. This helps the ants in converging towards the most often used paths and disregard the bad solutions that are longer and infrequently used. Stigmergy is the property that describes the communication among the ants. This communication among the ants is indirect and asynchronous in nature. The ants change the environment to interact with each other. They need not be at the same location at the same time to interact with each other.

Ant algorithms are typically Meta heuristic algorithms that are inspired by the foraging behavior of the ants. Ant algorithms are self organizing where each ant agent functions independently. Artificial ant agents are modeled after real ants. Artificial ants maintain memory about where they have been. Self organization of the real ants is achieved by the stochastic state transitions rule. The ant agents construct solution by moving in the graph through stochastic steps. The decision on which node is to be traversed next is based on a probability distribution. The stochastic walks are performed until the terminal condition is reached. Stigmergy with respect to artificial ants refer to pheromone trail. Artificial pheromone is deposited either on the edge or on the node or on both as the artificial ants traverse. The artificial ants can construct the solution path and then, deposit the pheromone over the solution path. This type of pheromone deposit is called delayed trail update. The other type of trail update is a step by step trail update. Here, the artificial ants deposit pheromone every time a node is added to the solution path. The amount of pheromone deposited on a path helps in strengthening of the path. Earlier converging of the artificial ants to a path results in stagnation and is called snowball effect. To avoid the ants from prematurely converging to a solution, negative feedback is used. Evaporation of the pheromone is implemented by the reduction in the pheromone deposited.

Every ant algorithm contains a stochastic state transition rule. This rule utilizes two components - the local pheromone value and the heuristic value. The choice of the next node is made on a probability distribution [81]. The self organizing behavior and feedback mechanism of the ant algorithms can be capitalized for Bug Triaging.

6.2.2 Applications of Ant System in Other Domains

Ant routing has been applied in solving vehicle routing problems. Here, vehicles need to be routed in the most optimal route so that the fuel consumption can be minimized and maximum number of customers can be serviced. Each vehicle is modeled as an ant agent and the route is formed by incrementally visiting each customer, starting from the depot [82]. Ant systems have also been widely adapted for scalable link prediction in Social Networks. Predicting a new link from an existing link or characteristic is called link prediction. The social network is modeled as a graph where each user acts as a node and the relation between the user acts as a link. From the social network, the sub graphs are determined, and based on their evolution, new links are predicted [83]. The other application domains of ant systems are in community structure detection in email communication. The community structure detection takes into account the structure, direction, weight and semantics of the email communication [84]. Web usage patterns can also be predicted using ant colony systems. The web structure, web content and web usage are used for the prediction. Continuous learning strategy is used to train the ants. The trained ants are then deployed in the new web graph. A web site is modeled as a graph, with webpages as the nodes and the hyperlinks as the edges [85].

Ad hoc networks have proliferated due to the explosion of mobile communication. An ad hoc network is constructed by a mobile terminal that is enabled to communicate with other nodes anytime anywhere. The network topology changes dynamically due to the mobility of the node. Stochastic routing algorithms based on ant routing are used to route packets in these dynamic environments. Robust routing is performed by acquiring information from routing history. The routing algorithms are particularly stable when there are many packets and mobility of nodes is intense etc [86]. Co-authorship network is an emerging area in social network. Behavior pattern of the entities in the co-authorship network is studied based on ant colony optimization. The nodes in the co-authorship network are the author and the link is the joint publication among them. The relation among the entities is categorized to social ties and professional relation. Social ties are long term relation formed on the basis of joint studies, joint work assignments, etc.

Professional ties are more dynamic. The dynamic professional relation is modeled as ant colony optimization [87] , [88].

It can be inferred that ant colony systems have been widely adapted in diverse fields such as web pattern mining, vehicle routing, co-authorship network, etc. The common factor in all these domains is that they are evolving and dynamic in nature.

6.3 BUG TRIAGE WITH ANT SYSTEM

Bug Triage System based on ant system to make developer recommendation is presented. Bug Triage encompasses the developer recommendations in a volatile environment. Any Bug Triage System uses the past reassignments of bugs in forging making recommendations. The reassignments of the bug are captured in a Bug Toss Graph. The Bug Toss Graph is a dynamic graph. This is because new developers may be added at anytime. The area of expertise of the existing developers may mutate with time. Also, the language of the text in the bug report may also metamorphise. Overall, the entire OSS development cycle goes from period of stability to instability.

6.3.1 Formal Definition of the Problem

The bug tossing relations are captured as a graph G ={V,E}. Here, $V=\{v_1,v_2,\dots v_n\}$ is the set of vertices which exemplify the developers. $E=\{e_1,e_2,\dots\dots,e_n\}$ embodies the set of edges where each edge represents the previous bug tossing relation among the developers. The transition probabilities among the developers capture the local decisions made by the developer. The transition probability of an edge or link from v_i to v_k is $P(v_i|v_k)$.

$$P(v_i|v_k) = \begin{cases} N(v_i,v_k)/N(v_i) & \text{If } N(v_i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Here, $N(v_i,v_k)$ is the number of transfers from v_i to v_k. $N(v_i)$ is the total number of transfers from v_i. As discussed earlier, the ant algorithm contains a local variable pheromone and a heuristic value to compute the probability distribution based on which the next state is identified. The ants stochastically construct solutions by making walks on the Bug Toss Graph. Each edge 'e' has an associated pheromone trail value 't'. The initial trail value for all paths is 't_0'.Each edge 'e' also has a heuristic value 'h' that is the transition probability P(vi|vk) of that edge. Each ant also possesses a Tabu memory 'Mk' which holds in it the set of visited vertices.

The stochastic transition rule for ant 'k' for edge between 'v_i' and 'v_j' is given by

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[t_{ij}(T)]^{\alpha} \cdot [h_{ij}]^{\beta}}{\sum_{l \in J_i^{M_k}} [t_{il}(T)^{\alpha} \cdot [h_{ij}]^{\beta}} & \text{if } j \in J_i^{M_k} \\ \text{otherwise zero} \end{cases}$$

h_{ij}	-	transition probability of edge e_{ij}
t_{ij}	-	pheromone value of edge e_{ij}
α,β	-	parameters used to scale the pheromone trail value t_{ij} and the heuristic value h_{ij}.
$J_i^{M_k}$	-	the set of developers in the neighborhood of 'i' which the ant has not yet visited

The pheromone forgetting function is given by:

$$t_{ij}(T+1) = (1-\rho)t_{ij}(T) + N$$

ρ	-	evaporation Factor
N	-	number of ants traversed

6.3.2 Bug Triaging based on Ant Systems(BT-ANT)

The overall flow of the BT-ANT is portrayed in the Figure 6.1. Incremental learning framework is used to incorporate the evolution of the Bug Toss Graph.

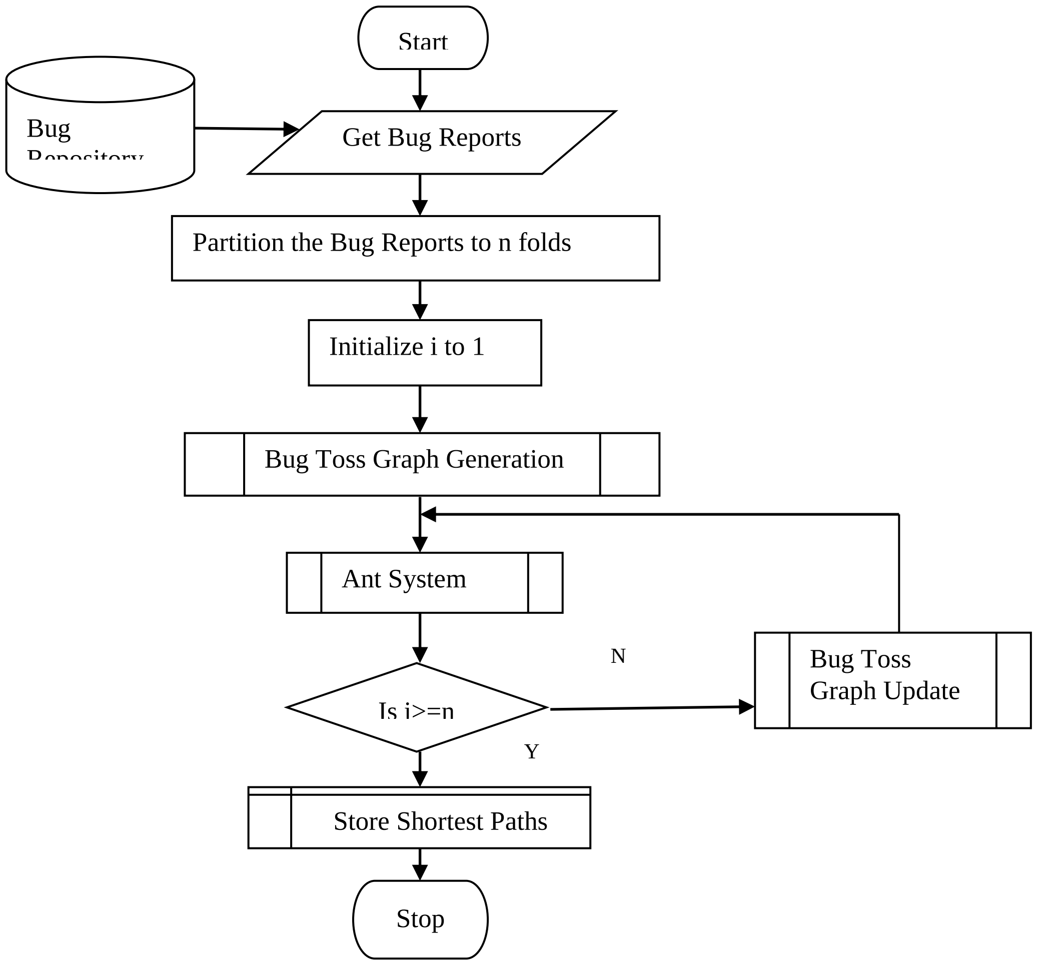
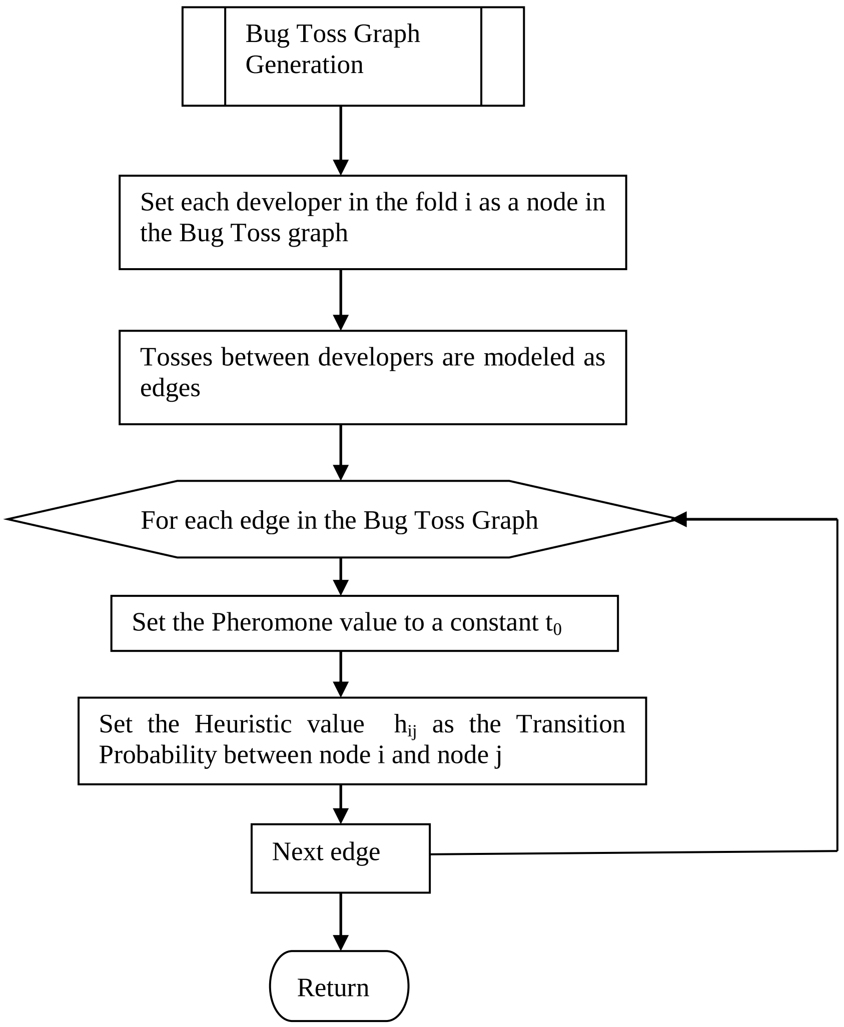


Figure 6.1: Flow Diagram of the BT-ANT

The bug reports are segregated to ‘n’ folds. The modules in the BT-ANT are i) Bug Toss Graph Generation ii) Ant Routing and iii) Bug Toss Graph Update. The Bug Toss Graph for the first fold is generated. The ant agents are deployed over the Bug Toss Graph. The ants lay pheromone trail to compute the shortest path to a node from the final resolvers. The initial state for the ant agents is a node from the set of first assignee. After the ants converge paths, the links and nodes from the next fold is updated in the Bug Toss Graph. After the update from the 'n^th' fold, the shortest paths are retrieved from the paths where the ants converged.

6.3.2.1 Bug Toss Graph Generation

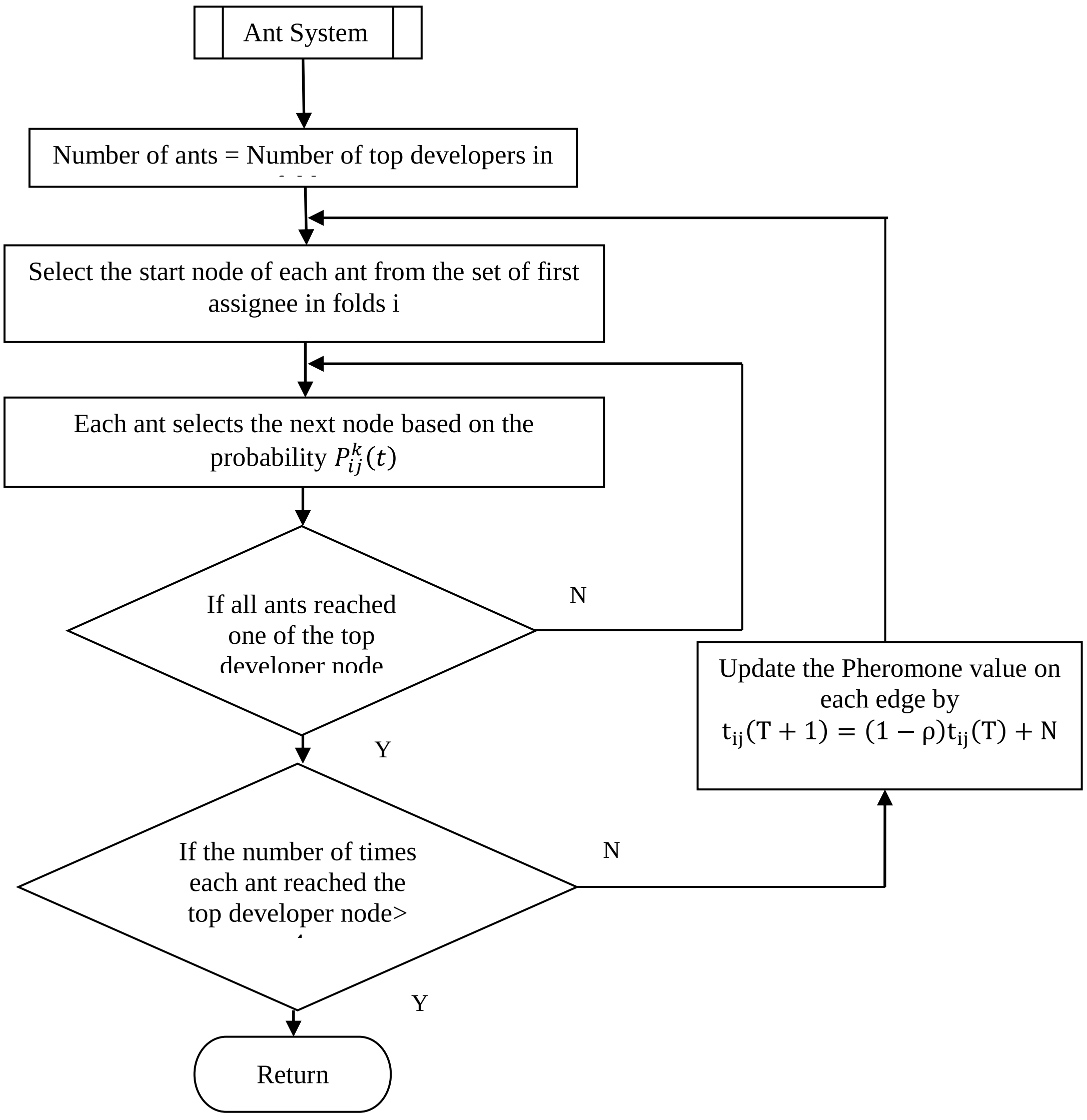


Bug Toss Graph Generation
Set each developer in the fold i as a node in the Bug Toss graph
Tosses between developers are modeled as edges
For each edge in the Bug Toss Graph
Set the Pheromone value to a constant t_0
Set the Heuristic value h_{ij} as the Transition Probability between node i and node j
Next edge
Return

Figure 6.2:Bug Toss Graph Generation

The flow diagram of the Bug Toss Graph Generation is rendered in Figure 6.2. The activity data is extracted from the bug reports. The activity data consists of the history of the bug. It contains data like who reported the bug, to whom it was first assigned, what was modified by the assignee, to whom it was tossed and who changed the status of the bug to resolve. The tosses or the reassignments a bug went through in its lifetime are synthesized in the Bug Toss Graph. The nodes in the Bug Toss Graph are the developers and the edges are the tosses between them. The Bug Toss Graph is a weighted directed graph. The weights on the links are a heuristic value and a pheromone value. The heuristic value h_{ij} on any edge between node 'i' and node 'j' is the transition probability of tosses between them. Initially, a constant pheromone value 't_0' is assigned for the edges traversed in the first fold.

6.3.2.2 Ant System



Ant System
Number of ants = Number of top developers in
Select the start node of each ant from the set of first assignee in folds i
Each ant selects the next node based on the probability $P_{ij}^{k}(t)$
If all ants reached one of the top developer node
N
Y
Update the Pheromone value on each edge by $t_{ij}(T+1) = (1-\rho)t_{ij}(T) + N$
If the number of times each ant reached the top developer node>
N
Y
Return

Figure 6.3: Ant System

The Ant System adapted for Bug Triaging is illustrated in the Figure 6.3. The number of ants deployed is equal to the number of top developers in that fold.

The top developers are identified according to their bug resolution frequency. The starting node of the ant is from the set of first assignee. The ant agent selects the next node based on the probability$P_{ij}^{k}(t)$. An iteration is considered completed when all the ants reach one of the top developers. After an iteration is completed, the negative feedback of the ant system is enforced by the pheromone update. The update comprises of two factors: a reinforcement factor and a forgetting factor. The reinforcement factor increases the pheromone content of an edge proportional to the number of ants that have traversed that edge. The forgetting factor decrements the pheromone value. When each ant reaches the same top developer more than four times, the iteration for that fold is concluded.

6.3.2.3 Bug Toss Graph Update

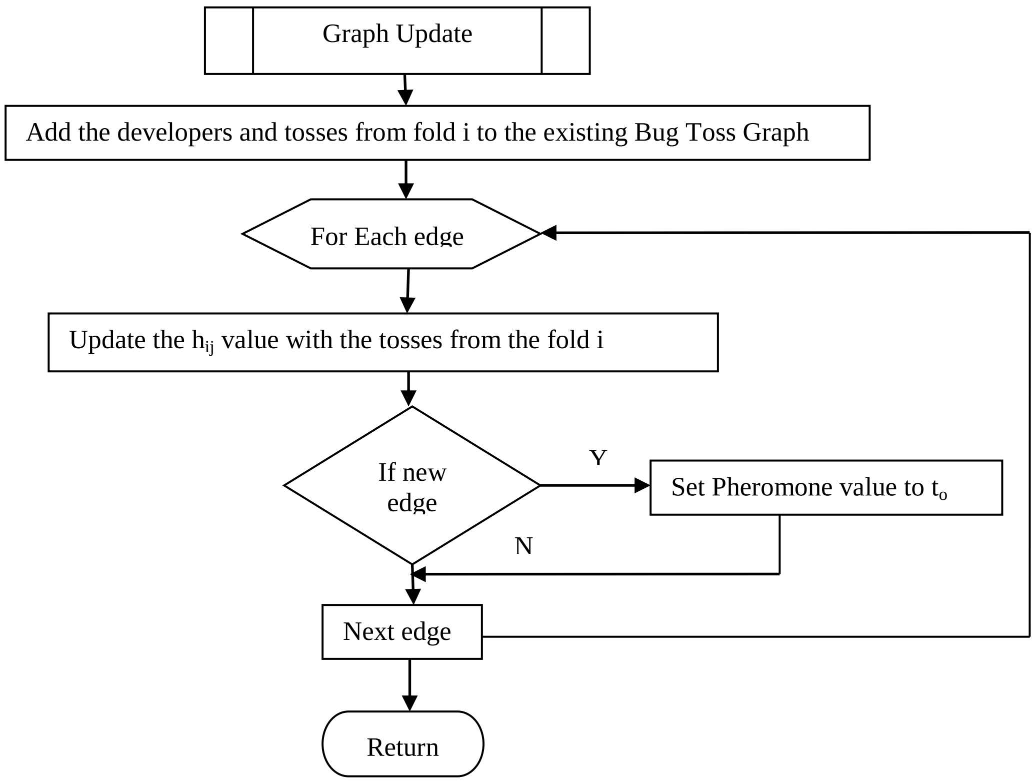

Graph Update
Add the developers and tosses from fold i to the existing Bug Toss Graph
For Each edge
Update the h_{ij} value with the tosses from the fold i
If new edge
Y
Set Pheromone value to t_o
N
Next edge
Return

Figure 6.4: Bug Toss Graph Update

The flow of the Bug Toss Graph Update is shown in the Figure 6.4. The new developers from the next fold and the new links among the existing developers are amended in the existing Bug Toss Graph. The heuristic values on the existing edges are adjusted to reflect any new tosses added to the same edge. For new edges, the new heuristic values are determined. For the new edges introduced in the fold, the pheromone values assigned is the constant 't_0'.

6.2.3.4 Algorithm for Bug Triage based on Ant System

The algorithm is given in the Figure 6.5. There are various assumptions made in implementing the BT-ANT.

```
Procedure BT-ANT
F={Set of First Assignees in a fold}, Initial Pheromone value $t_0$=0.1,
BTG: Bug Toss Graph
Begin
1:  get Bug reports from Bug Repository
2:  partition the Bug reports to 'n' folds
3:  initialize i=1
4:  call Generate Bug Toss Graph
5:  repeat Step 6 to 7 Until (i<n)
6:      call Ant System
7:      call Update Bug Toss Graph
8:  increment  i
End
Procedure  Generate Bug Toss Graph
1:  set each developer in $fold_i$  as a node
2:  set the tosses between developers as edges
3:  for each edge in the BTG do
4:       initialize pheromone value  to '$t_0$'
5:       initialize  Heuristic value  $h_{ij}$ as the Transition   Probability between node i and node j
6:  return
Procedure  Ant System
1:  initialize Number of ants = Number of top  developers in $fold_i$
2:  repeat Step 3 to 6 Until each ant reaches the same Top developer > 4 times
3:       select the start node of each ant from the set F for '$fold_i$
4:       repeat  Step 5 Until all deployed ants  reach  one of the Top developer
5:           Each ant selects the next node based on the  probability  $P_{ij}^{k}(t)$
6:  update the Pheromone value on each edge by $t_{ij}(T+1)$
7 :  return
Procedure  Update Bug Toss Graph
1: insert the developers  and tosses  from  $fold_{i+1}$ to  BTG
2:for each edge inserted  from $fold_{i+1}$ do
3:  update the $h_{ij}$ value with tosses from  $fold_{i+1}$
4:  if  new edge
5:          initialize Pheromone value to $t_o$
6:  return
```

Figure 6.5: Algorithm for BT-ANT

The value assumed for 'α' and 'β' is 0.5 so as to give equal weight to both the transition probability and the pheromone value. 'ρ' is assumed to be 0.6. The value 't_0' which is the initial pheromone value is assumed as 0.1.

6.3.3 Performance Evaluation

The bug reports of Eclipse project during the period from 2009/01/09 to 2013/01/09 were extracted from www.bugzill.org. The bugs with vote count as 2 and with status as "RESOLVED" and resolution as "FIXED" were only considered for extraction. The data set was divided into training set and testing set. The first 70% Eclipse bugs were used as a training set and the remaining bugs were used as a testing set. The experiments were conducted as three studies to test the robustness of the proposed system. The first study comprises of bug reports for two years for the period from 2011/01/09 to 2013/01/09), the second study for three years for the period from 2010/01/09 to 2013/01/09) bug reports and the third study for four years from 2009/01/09 to 2013/01/09) bug reports. The proposed system BT-ANT was compared with the base line system of WBFS algorithm deployed on Bug Toss Graph modeled as GP model. The proposed and the baseline systems were compared based on the four parameters – (i) the Length of the Retrieved Path (ii) Precision(iii) Recall and (vi) Path Similarity. The experiments were run on a Pentium 4 processor with 320 GB hard disk. Netbeans 7.2 was used as the front end and Oracle as the back end. The JDK tool was employed in conducting the experiments.

6.3.3.1 Path length

The performance of the BT-ANT with respect to Path Length parameter, compared to the existing GP-WBFS, proposed AP-WBFS and Bi-Objective are manifested in Figure 6.6, Figure 6.7 and Figure 6.8 for all the three data sets. From the results, it is discernible that the existing system based on GP model is superior to the AP-WBFS, Bi-Objective as well as the BT-ANT with respect to Path Length. When compared with the other Bug Triage Systems based on AP model, BT-ANT surpasses in performance.

Hypothesis with respect to the result of the parameter P (Path Length)

Null hypothesis H0 : P1 = P2 = P3 = P4, where P1= Path Length obtained in GP-WBFS, P2 = Path Length obtained in AP-WBFS,P3 = Path Length obtained in Bi-Objective and P4=Path Length obtained in BT-ANT.

(There is no significant difference among the four systems in terms of Path Length obtained)

Alternate hypothesis H9: Path Length mean values are not equal for at least one pair of the result mean values of the parameter P.

(There is a significant difference among the four systems in terms of Path Length obtained)

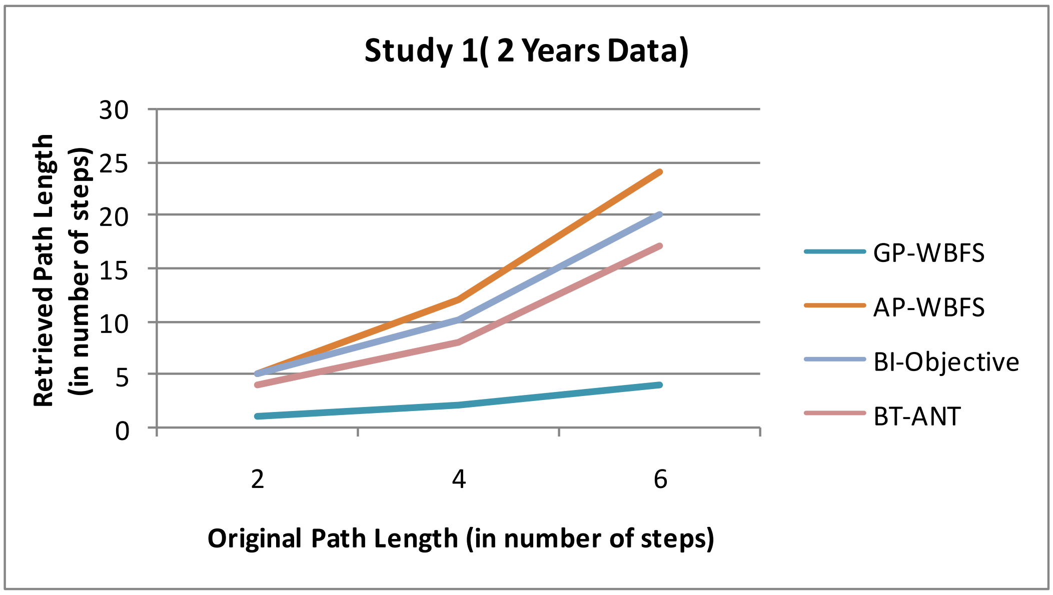


Figure 6.6: BT-ANT: Path Length Performance for Study 1

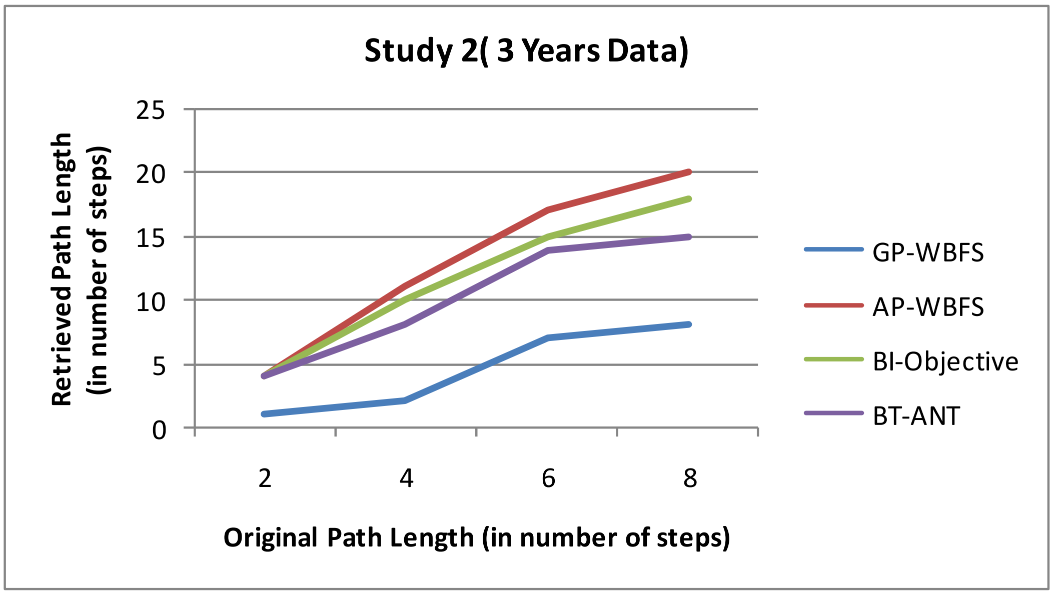


Figure 6.7: BT-ANT: Path Length Performance for Study 2

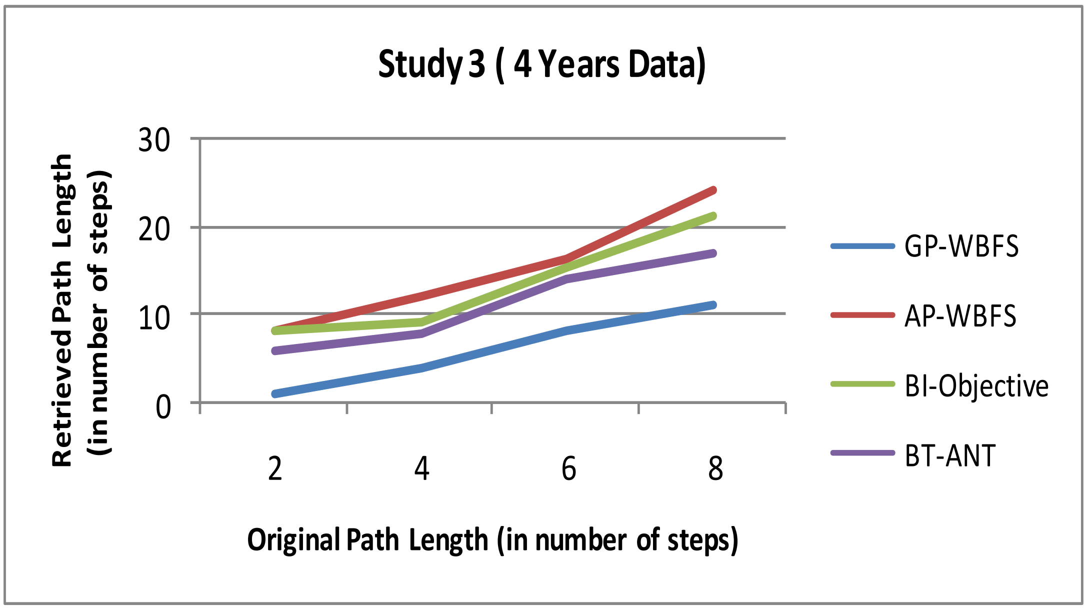


Figure 6.8: BT-ANT: Path Length Performance for Study 3

Hypothesis Evaluation with respect to P (Path Length)

Table 6.1: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective and BT-ANT based on Path Length

Hypothesis	**Technique**				**Study**	**Path Length**
	I	**II**	**III**	**IV**		**p value**
H9	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	**Study 1**	<0.0001
					Study 2	<0.0001
					Study 3	<0.0001

From the Table 6.1, it is concluded that the calculated significance level of the parameter-Path Length by comparing the four systems namely, GP-WBFS,AP-WBFS,Bi-Objective and BT-ANT always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Path Length values of the above four systems namely, GP-WBFS,
AP-WBFS,Bi-Objective and BT-ANT. Hence, the null hypothesis for H9 may be rejected.

The results obtained from the three studies are consolidated using the Weighted Average method. Based on which the results are consolidated in the Figure 6.9.

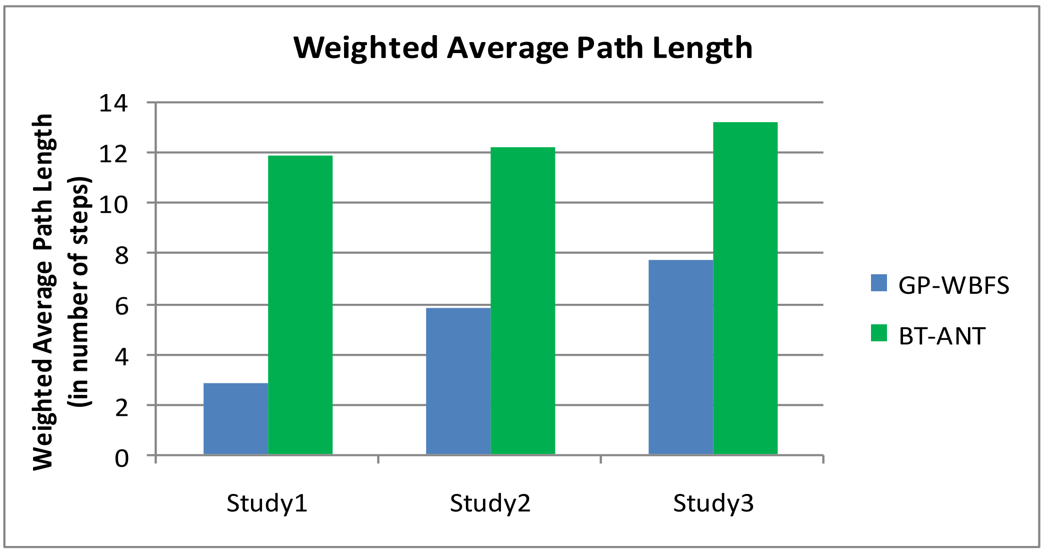


Figure 6.9: BT-ANT: Weighted Average Path Length

The rate of increase in the Path Length in the BT-ANT when compared to the existing GP-WBFS is 11.41.

6.3.3.2 Precision

The performance of the BT-ANT with respect to Precision parameter,compared with the existing GP-WBFS, proposed AP-WBFS and Bi-Objective are manifested in the Figure 6.10, Figure 6.11 and Figure 6.12 for all the three data sets. From the results, it is indisputable that the BT-ANT based on AP model is superior consistently over the three test runs to the AP-WBFS, Bi-Objective as well as the GP-WBFS.

Hypothesis with respect to the result of the parameter PR (Precision)

Null hypothesis H0 : PR1 = PR2 = PR3 = PR4, where PR1= Precision obtained in GP-WBFS, PR2 = Precision obtained in AP-WBFS,PR3 = Precision obtained in Bi-Objective and PR4=Precision obtained in BT-ANT.

(There is no significant difference among the four systems in terms of Precision obtained)

Alternate hypothesis H10: Precision mean values are not equal for at least one pair of the result mean values of the parameter PR.

(There is a significant difference among the four systems in terms of Precision obtained)

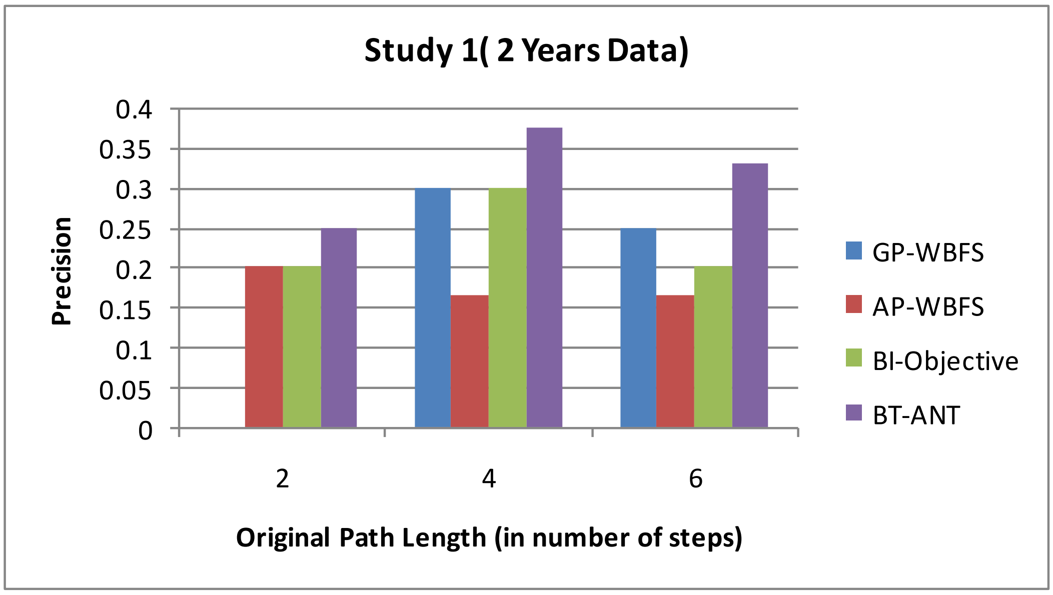


Figure 6.10: BT-ANT: Precision for Study 1

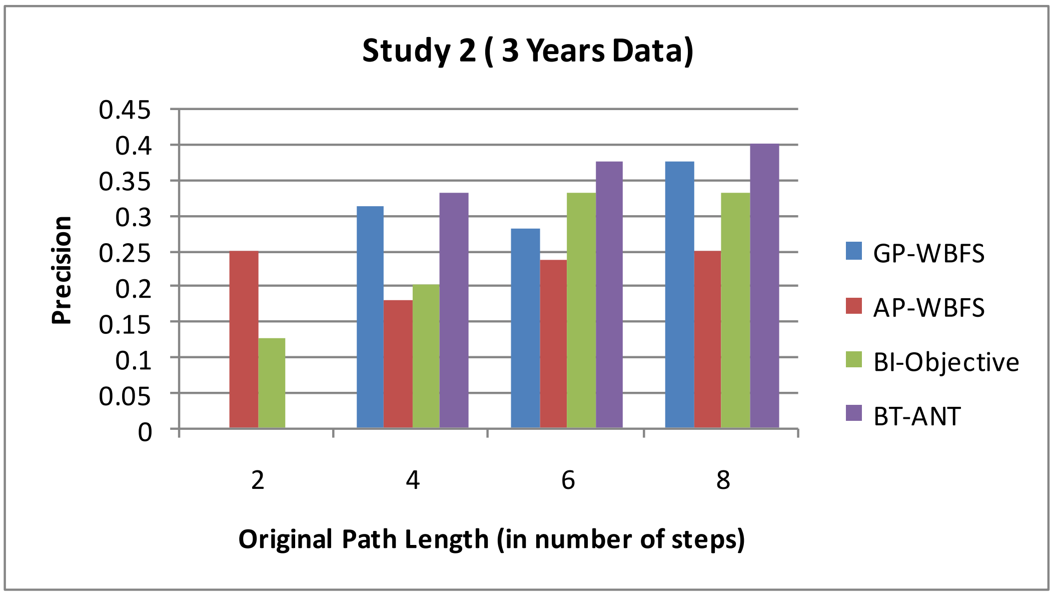


Figure 6.11: BT-ANT: Precision for Study 2

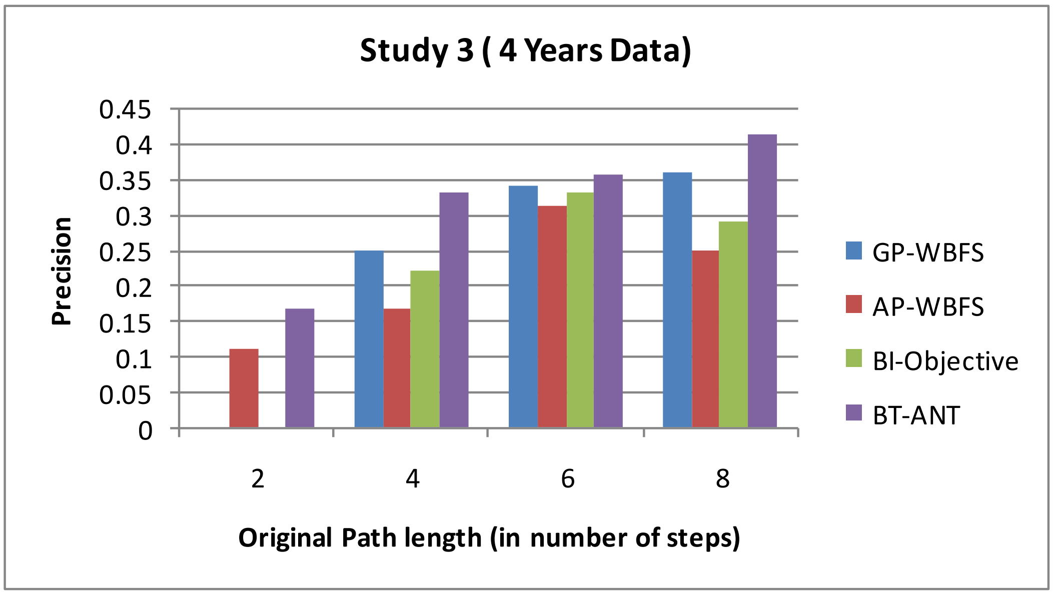


Figure 6.12: BT-ANT: Precision for Study 3

Hypothesis Evaluation with respect to PR: (Precision)

Table 6.2: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective and BT-ANT based on Precision

Hypothesis	**Technique**				**Study**	**Precision**
	I	**II**	**III**	**IV**		**p value**
H10	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	**Study 1**	<0.0001
					Study 2	0.127
					Study 3	<0.0001

From the Table 6.2, it is established that the calculated significance level of the parameter Precision by comparing the four systems namely, GP-WBFS, AP-WBFS,Bi-Objective and BT-ANT always satisfy the condition (p value<0.05) for study 1 and study 3. There is significant difference among the results for different Precision values of the above four systems namely, GP-WBFS,AP-WBFS,Bi-Objective and BT-ANT in two of the three studies. Hence, the null hypothesis for H10 may be rejected.

The proposed BT-ANT outweighs the existing system in terms of the parameter Precision. This is because the number of false positives extracted by the BT-ANT is significantly fewer than the existing systems. This was possible because the Bug Toss Graph is better revised due to the incremental learning framework. The consolidated Weighted Average Precision is presented in the Figure 6.13. The increase in Precision of BT-ANT with respect to the existing GP-WBFS is 0.253.

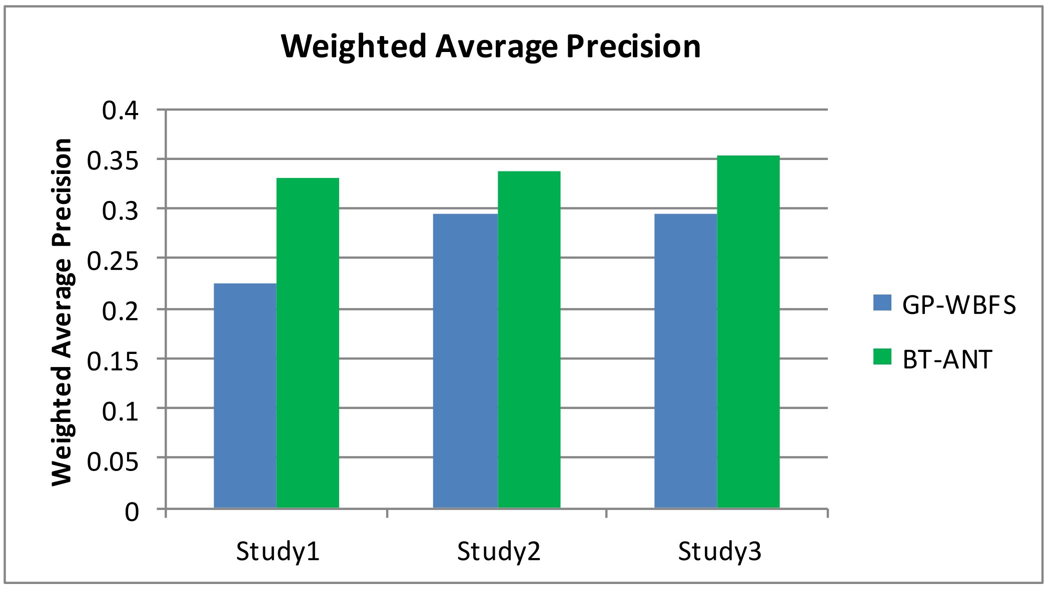


Figure 6.13: BT-ANT: Weighted Average Precision

6.3.3.3 Recall

The performance of the BT-ANT with respect to Recall parameter,compared with the existing GP-WBFS, proposed AP-WBFS and Bi-Objective is manifested in the Figure 6.14, Figure 6.15 and Figure 6.16 for all the three data sets. From the results, it is evident that the BT-ANT based on AP model is superior consistently over the three test runs to the AP-WBFS, Bi-Objective as well as the GP-WBFS.

Hypothesis with respect to the result of the parameter R (Recall)

Null hypothesis H0: R1 = R2= R3= R4, where R1= Recall obtained in GP-WBFS, R2 = Recall obtained in AP-WBFS, R3 = Recall obtained in Bi-Objective and R4=Recall obtained in BT-ANT

(There is no significant difference among the four systems in terms of Recall obtained)

Alternate hypothesis H11: Recall mean values are not equal for at least one pair of the result mean values of the parameter R.

(There is a significant difference among the four systems in terms of Recall obtained)

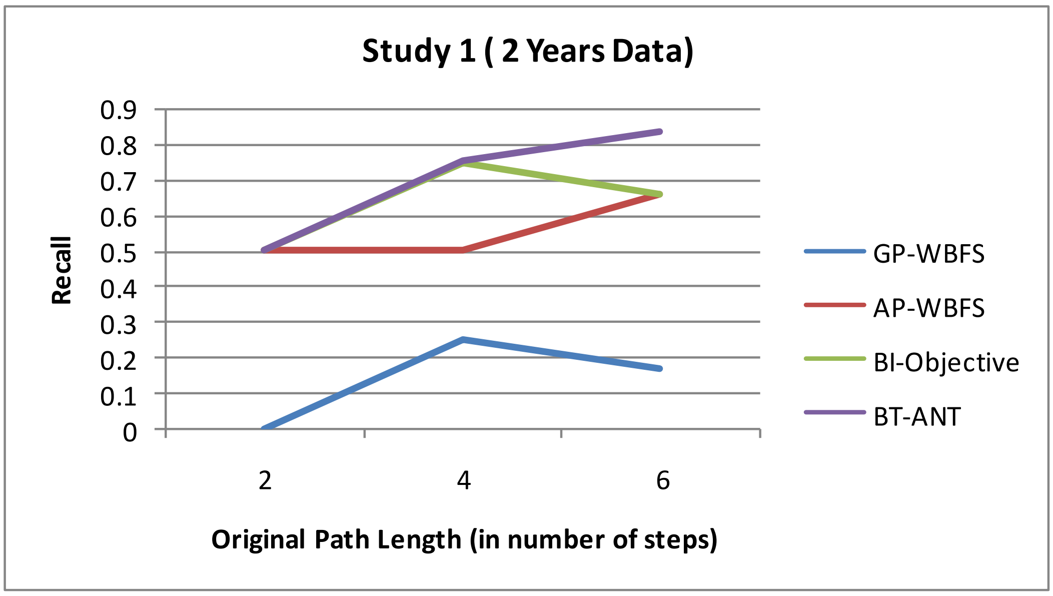


Figure 6.14: BT-ANT: Recall for Study 1

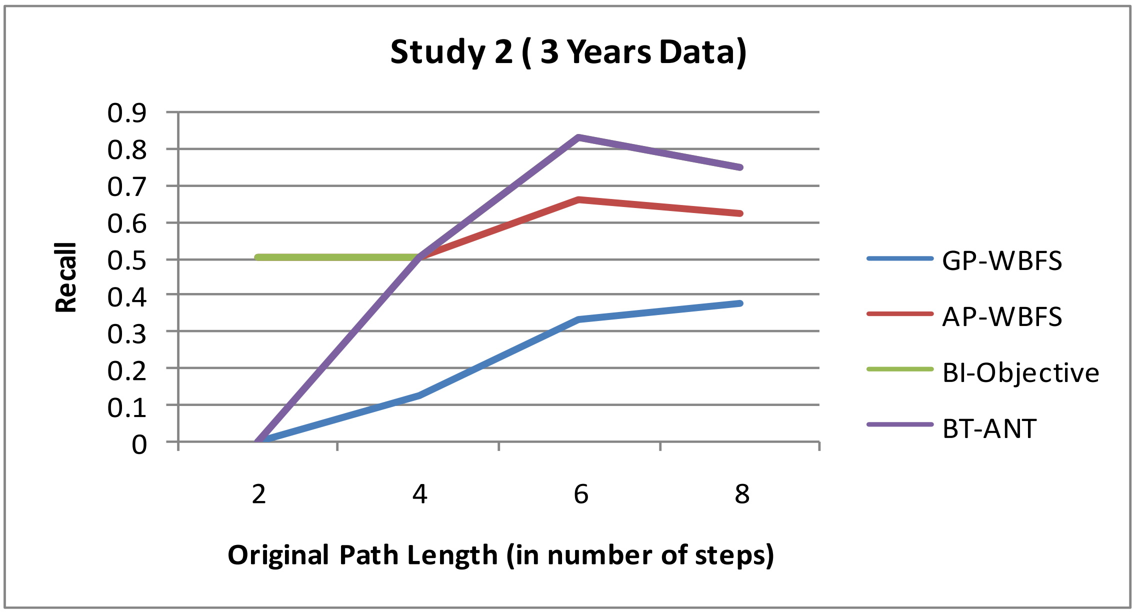


Figure 6.15: BT-ANT: Recall for Study 2

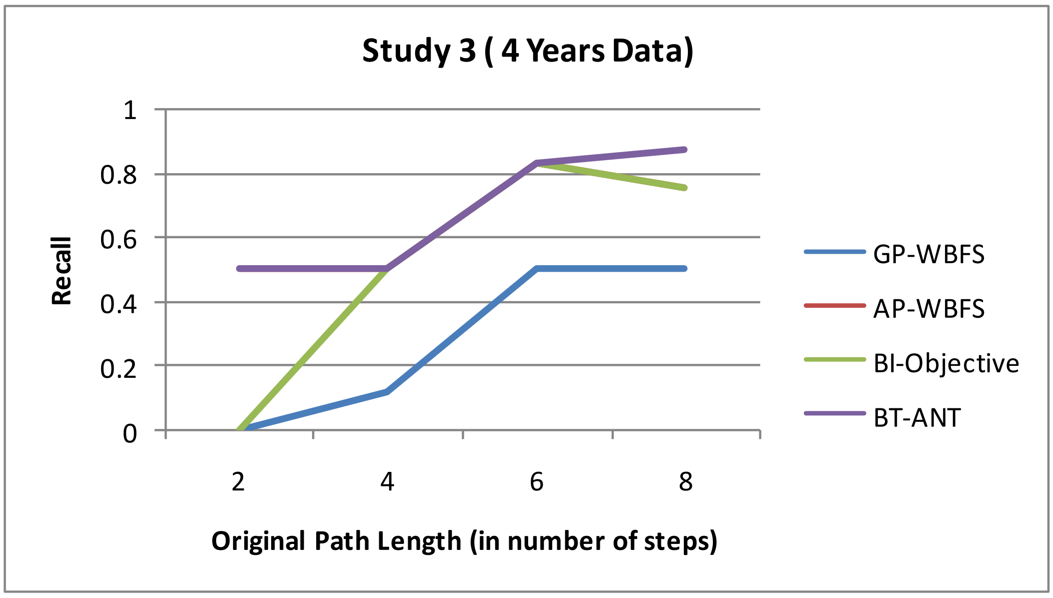


Figure 6.16: BT-ANT: Recall for Study 3

Hypothesis Evaluation with respect to R(Recall)

Table 6.3: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective and BT-ANT based on Recall

Hypothesis	**Technique**				**Study**	**Recall**
	I	**II**	**III**	**IV**		**p value**
H11	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	**Study 1**	<0.0001
					Study 2	<0.0001
					Study 3	<0 .0001

From the Table 6.3, it is realized that the calculated significant level of the parameter-Recall when comparing the four systems namely, GP-WBFS, AP-WBFS,Bi-Objective and BT-ANT always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Recall values of the above four systems namely, GP-WBFS,AP-WBFS, Bi-Objective and BT-ANT. Hence, the null hypothesis for H11 may be rejected. The proposed BT-ANT performs better than the existing system in terms of the parameter - Recall. This is because the number of true positives extracted by the BT-ANT is significantly greater than the existing system. The increase in Recall of BT-ANT when compared to the existing GP-WBFS is 1.635 as shown in Figure 6.17.

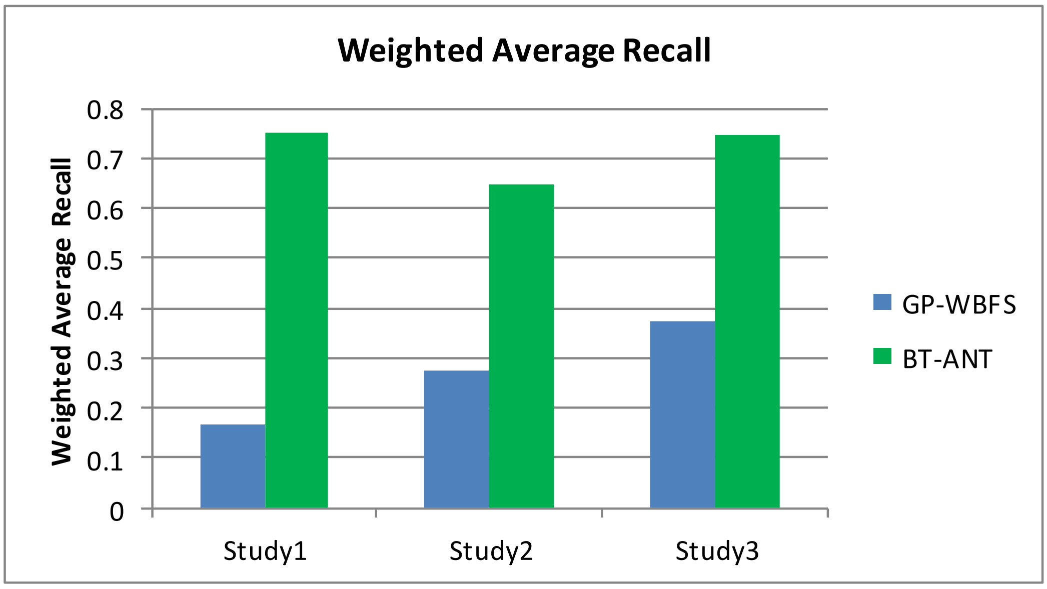


Figure 6.17: BT-ANT: Weighted Average Recall

6.3.3.4 Path Similarity

The performance of the BT-ANT with respect to Path Similarity parameter, compared with the existing GP-WBFS, proposed AP-WBFS and Bi-Objective is manifested in Figure 6.18, Figure 6.19 and Figure 6.20 for all the three data sets. From the results, it is obvious that the BT-ANT based on AP model dominates the AP-WBFS, Bi-Objective as well as the GP-WBFS in all the three runs with respect to Path Similarity.

Hypothesis with respect to the result of the parameter PS(Path Similarity)

Null hypothesis H0 : PS1 = PS2 = PS3 = PS4, where PS1= Path Similarity obtained in GP-WBFS, PS2 = Path Similarity obtained in AP-WBFS , PS3 = Path Similarity obtained in Bi-Objective and PS4=Path Similarity obtained in BT-ANT.

(There is no significant difference among the four systems in terms of Path Similarity obtained)

Alternate hypothesis H12: Path Similarity mean values are not equal for at least one pair of the result mean values of the parameter.

(There is a significant difference among the four systems in terms of Path Similarity obtained)

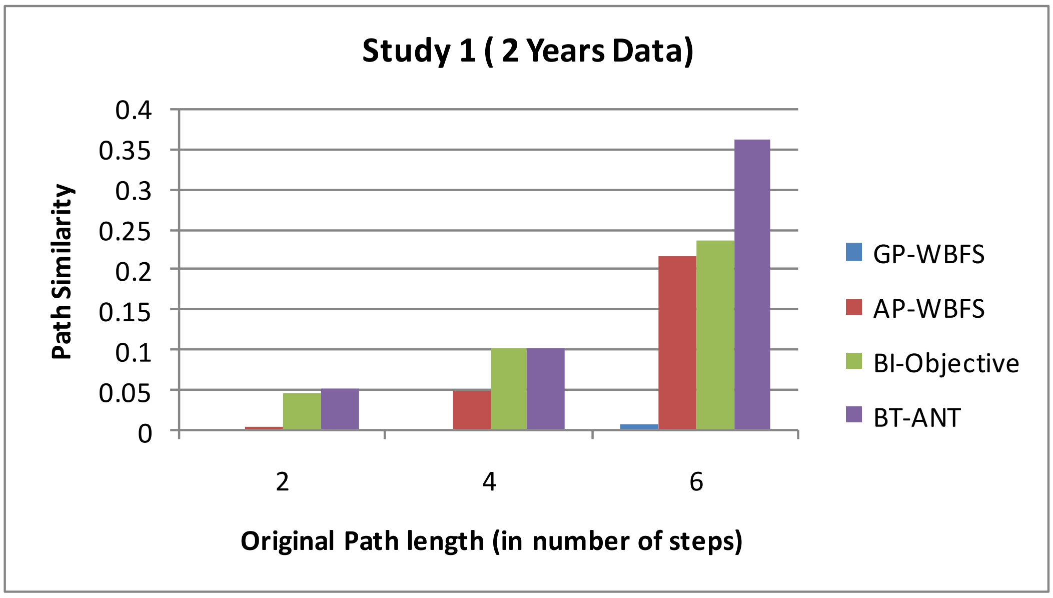


Figure 6.18: BT-ANT: Path Similarity for Study 1

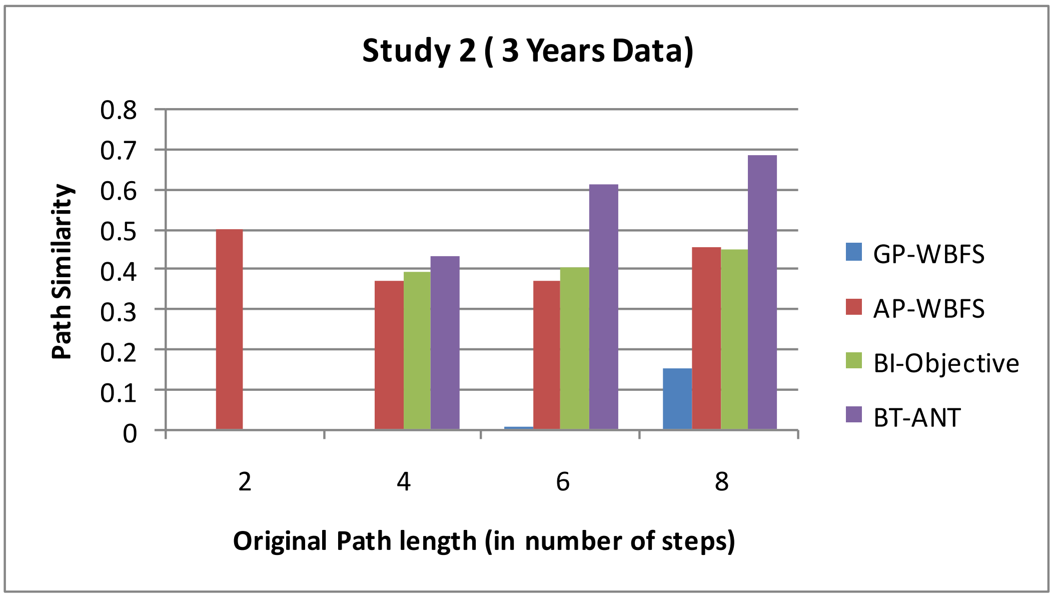


Figure 6.19: BT-ANT: Path Similarity for Study 2

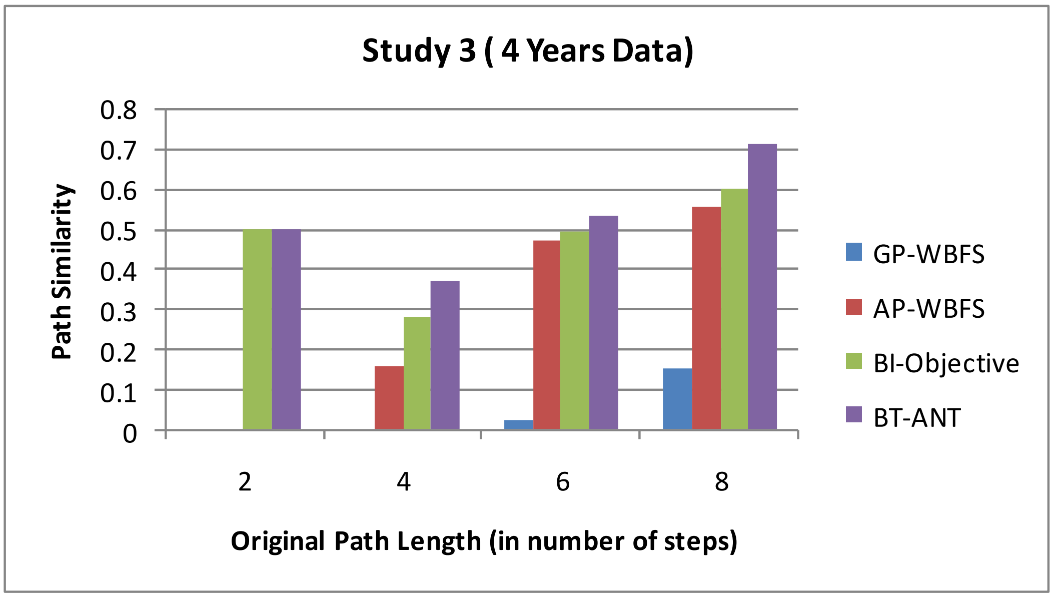


Figure 6.20: BT-ANT: Path Similarity for Study 3

Hypothesis Evaluation with respect to PS (Path Similarity)

Table 6.4: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective and BT-ANT based on Recall

Hypothesis	**Technique**				**Study**	**Recall**
	I	**II**	**III**	**IV**		**p value**
H12	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	**Study 1**	<0.0001
					Study 2	<0.0001
					Study 3	<0.0001

From the Table 6.4, it is confirmed that the calculated significance level of the parameter-Path Similarity by comparing the four systems namely, GP-WBFS,AP-WBFS,Bi-Objective and BT-ANT always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Path Similarity values of the above four systems namely, GP-WBFS, AP-WBFS, Bi-Objective and BT-ANT. Hence, the null hypothesis for H12 may be rejected.

The proposed BT-ANT performs better than the existing system in terms of the parameter - Path Similarity. The reason is due to the number of true positives extracted by the BT-ANT, are significantly greater than the existing system and the number of false positives reported is fewer. This is the consequence of the fact that the underlying tossing relation is better adapted in the Bug Toss Graph due to the pheromone evaporation. The increase in Path Similarity of BT-ANT when compared to the existing GP-WBFS is 8.897 as depicted in Figure 6.21.

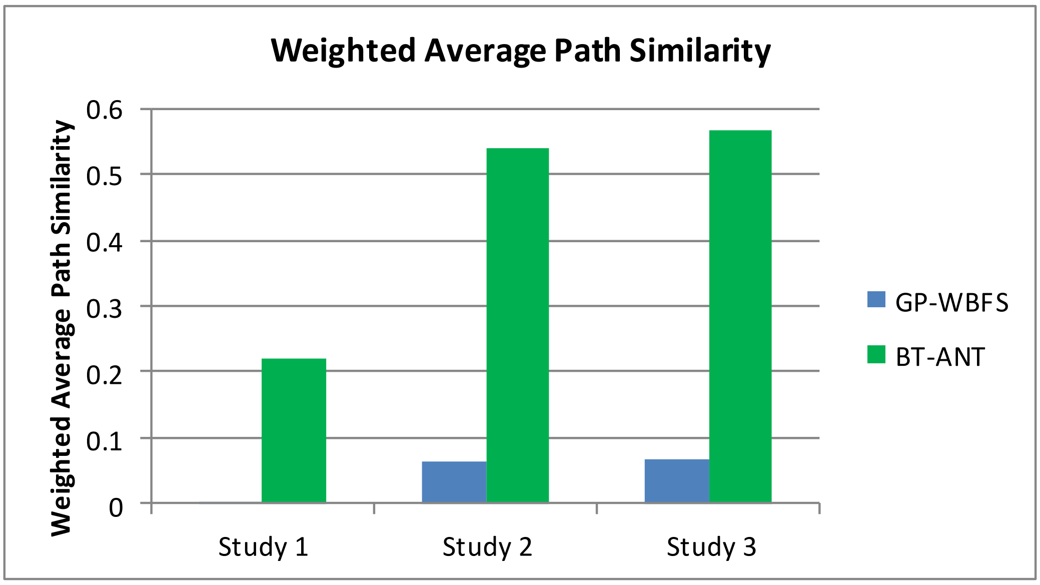


Figure 6.21: BT-ANT : Weighted Average Path Similarity

6.4 COLLABORATION GRAPH

Collaboration Graphs encode in them who works with whom in a specific setting.There are various kinds of collaboration graphs including citation graph, co-authorship among scientist graph, co-appearance of movie actors, web graph etc. Collaboration graph is more relevant in the distributed software development setting.The collaboration occurs mostly in virtual environment. The collaborations among the developers engaged in the software development can be modeled as a collaboration graph. The communication among team members are based on email, chat rooms and software artifacts [89]. The collaboration graph is essentially a dynamic graph [90]. The graph is modified for every alteration that occurs in the corresponding domain.

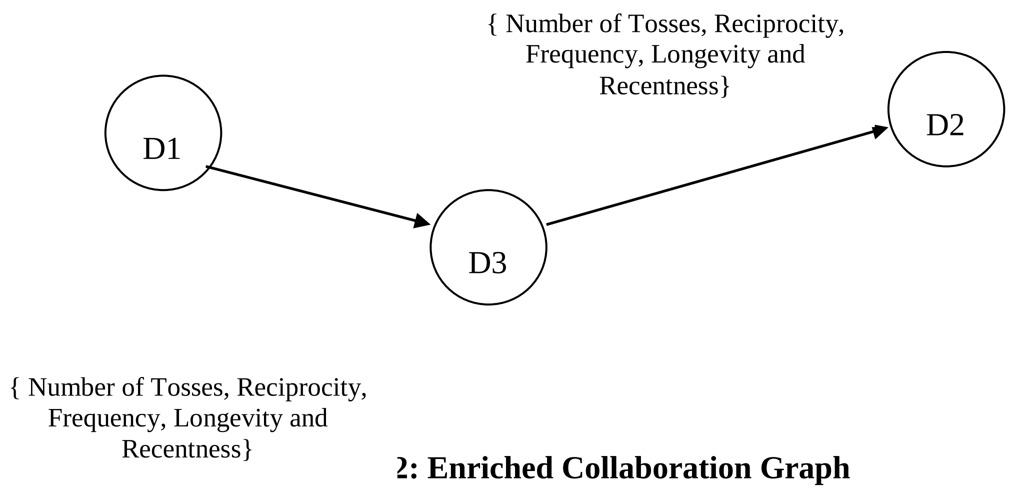
6.4.1 Collaboration in Open Source System

OSS system is a special case of geographically distributed software development. Collaboration is the norm of OSS software development. Bug resolution in OSS also falls under the same analogous category. The resolution of the bug hinges on human factors. The human factors comprise of physical or cognitive property of social behavior [91]. More specifically, the human behavior comprises of collaboration among developers, capacity of developers and developers' past performance. Human factors affect the quality and productivity of the task at hand.

The Bug Toss Graph serves as the data structure that is appropriate for Bug Triage. In the existing system, the Bug Toss Graph uses only "the number of tosses" to capture the collaboration among the developers involved in bug resolution. The Enriched Collaboration Graph is an improvisation over the existing Bug Toss Graph. The Enriched Collaboration Graph characterizes the developers' social behavior. This Enriched Collaboration Graph can be used for Bug Triage.

6.5 BUG TRIAGE USING ENRICHED COLLABORATION GRAPH

The bug log activity is utilized to form a social network [92]. Bug resolution is a team activity. The team dynamics needs to be realized in the underlying graph structure. Each developer has a role to play towards the resolution of the bug. The different roles may be that of resolver, assignee, triager, broker, etc. The roles of the developers are also transitory. The relationship that exists among the developers can be identified from the social network formed from the bug log. The strength of the relationship is measured and modeled as an Enriched Collaboration Graph.


{ Number of Tosses, Reciprocity, Frequency, Longevity and Recentness}
D1
D2
D3
{ Number of Tosses, Reciprocity, Frequency, Longevity and Recentness}

2: Enriched Collaboration Graph

The affiliation among the developers is based on the tosses that emanate among them. The strength of the relationship among the developers is manifested based on the Number of Tosses, Reciprocity, Frequency, Longevity and Recentness [93]. The Enriched Collaboration Graph is constituted based on these factors and is shown in Figure 6.22.

The relationship that exists among the developers is calibrated as per the proximity of the developer with his peer. The proximity is basically derived from the tossing relation among the developers. In the existing system, the tossing relation among the developers is analyzed only in one dimension that is the number of tosses. Whereas, in this research work, the tossing relation is fine grained using additional dimension to extract the proximity among the developers in social context.

The Multi Objective Ant algorithm is deployed on the Enriched Collaboration Graph so as to retrieve the set of shortest paths. The conceptual diagram of the Bug Triage based on the enriched collaboration graph is portrayed in the Figure 6.23.

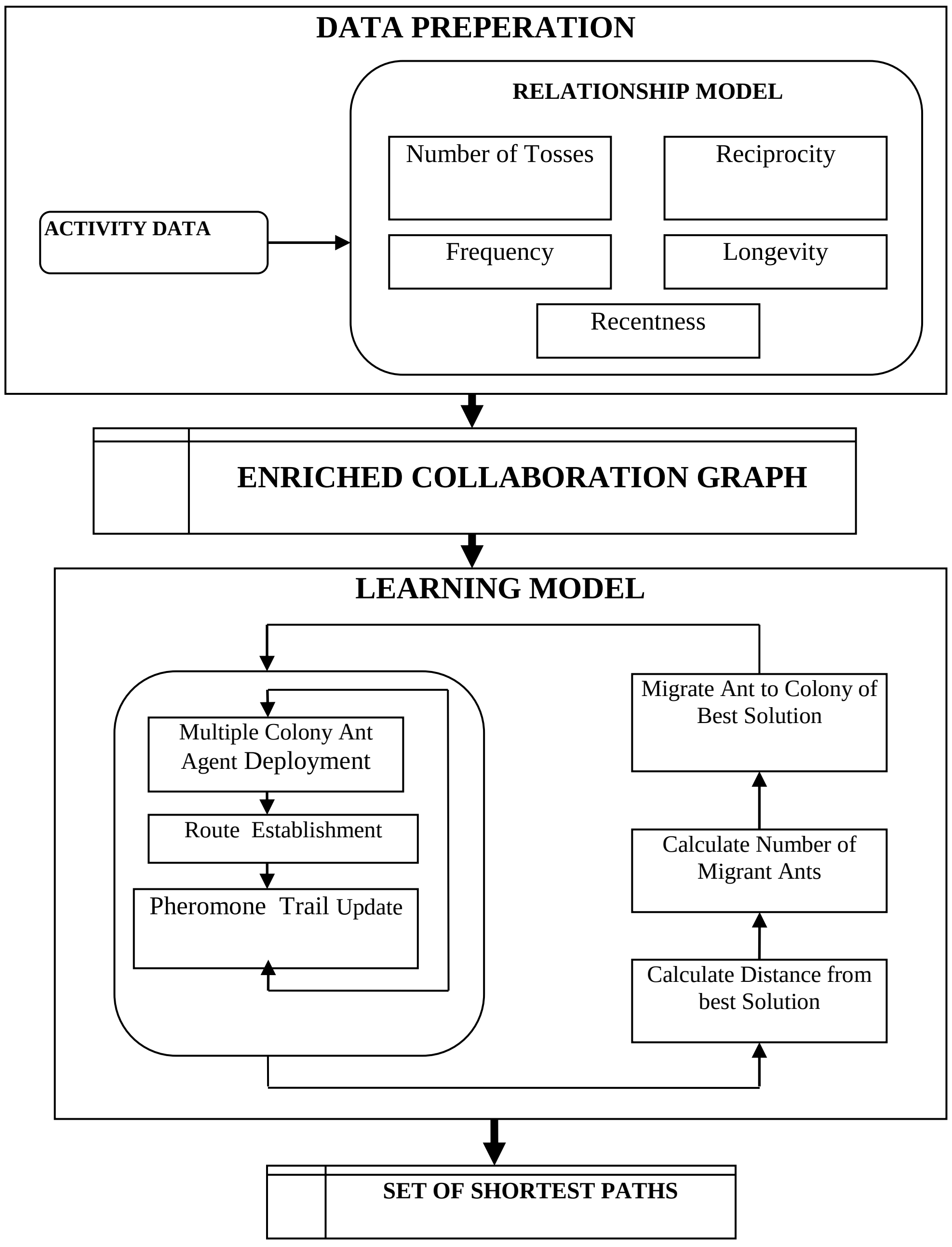


Figure 6.23: Bug Triage using Enriched Collaboration Graph

6.5.1 Formal Definition of the Enriched Collaboration Graph

The Enriched Collaboration Graph is multi featured and models the proximity of one developer with another. The labels in the graph are Number of tosses, Longevity, Recentness, Frequency and Reciprocity [93]. The Number of tosses quantifies the total number of tossing relation between any two developers.

The Longevity factor codifies the duration of relation between any two developers. This parameter brings out any long standing relation that prevails among the developers. It encodes the age of a relation as shown in Figure 6.24. The Recentness factor reveals whether the relation is alive on date. The Recentness parameter is the ratio between amount of time passed, since the beginning of the relation and the current toss compared to the total age of the relation [94]. It is shown in the Figure 6.25.The Frequency parameter quantifies the density of the tossing relation. Finally, the Reciprocity parameter divulges whether the relation among any two developers is balanced. The Reciprocity encodes the mutual trust between the developers.

Number of Tosses (T)	:	Total number of tosses from one developer to another.
Longevity (L)	:	The duration of relation between any two developers.
Recentness(R)	:	The freshness of the relation between any two developers.
Frequency (F)	:	The number of tosses for the duration of relation between any two developers.
Reciprocity (Rc)	:	The strength of the two way relation between any two developers.

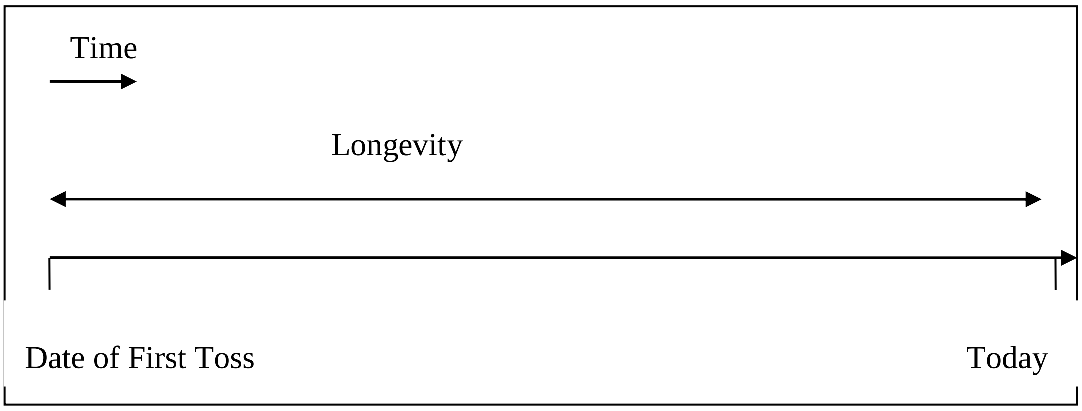


Figure 6.24: Longevity

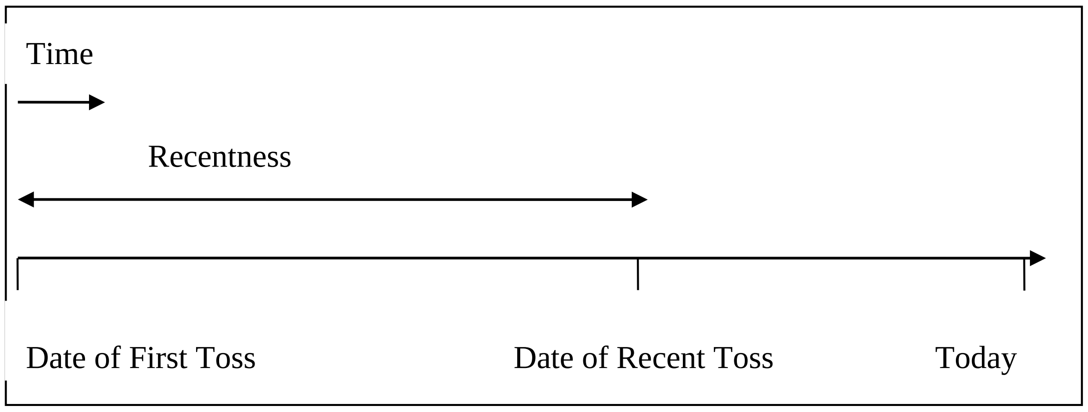


Figure 6.25: Recentness

The Enriched Collaboration Graph ‘E’ contains a set of developers represented by ‘D’ and the relationship among the developers is represented by edge ‘E’. Each edge in the graph is represented by a Vector ‘V’.

$$V = \{'T', 'L', 'F', 'R', 'Rc'\}$$

where, $T = \text{Number of Tosses}$.

$L = \text{Date of Recent Toss} - \text{Date of First Toss}$

F= T/L

$R = 1 - (L/(\text{Current Date} - \text{Date of first Toss}))$

$Rc = 1 - \text{Mod}((\text{Indegree} - \text{Outdegree})/\text{Nex})$

Indegree - Number of Tosses from Developer

Outdegree- Number of Tosses to Developer

Nex - Number of Tosses exchanged between the two Developers

6.5.2 Cooperative Ant Algorithm

The Cooperative Ant algorithm (Co-Ant) is used over the enriched collaboration graph to fetch the referral chain of developers who can optimally collaborate on a bug. The Co-Ant algorithm is a Multi-Objective Ant Colony Algorithm.It employs multiple colonies of ants.

There are five colonies of ants utilized. One ant colony is used for each factor ie., Number of Tosses, Reciprocity, Frequency, Longevity and Recentness. The bug reports are retrieved from the bug repository. The activity data is used to extract the tossing relation. The bug reports are portioned to ‘n’ folds in order to bring the learning of the ants in an incremental framework. The first fold is considered as the current fold. The number of top developers for the fold is evaluated. The top developers are evaluated according to the number of bugs they have resolved. The number of ants in each colony is determined by the number of top developers. Within each colony, an algorithm similar to the BT-ANT is deployed. The visibility factor on each of the graph is determined by the corresponding proximity factor. The ant chooses the next node, based on the heuristic function.

The heuristic function comprises the visibility factor and the pheromone factor. When the ant traverses an edge, it augments the pheromone value. When an ant reaches one of the top developers more than four times, it is considered that the ants have converged to a shortest path and the iteration for one fold is completed. After this iteration, the lengths of paths determined by the multiple ant colonies are compared.

The distance of the solution from each colony from the best solution is computed. Based on the distance of each solution from the best solution, the number of ants to be migrated (Nmig) to the colony that produced the best solution is estimated. ‘Nmig’ ants migrate to the colony that computed the best solution. The pheromone value on each edge is adjusted by the pheromone evaporation function. The Enriched Collaboration Graph is updated with edges and nodes from the next fold. The proximity factors are adjusted accordingly. The multiple colony ants are again deployed over the Enriched Collaboration Graph. The process is repeated until the Enriched Collaboration Graph is updated with edges and nodes from the last fold. The algorithm for Co-Ant is specified in the Figure 6.26.

```
Procedure Co-Ant
  T : Number of Tosses
  L : Longevity
  F : Frequency
  R : Recentness
  RP : Reciprocity
  Best: Cost of Best Solution of all the Ant Colonies
  B: Cost of Best Solution for an Ant Colony
  Tot : Total Number of Ants Deployed
  Nmig: Number of Ants to Migrate
 TotNmig: Total number of Ants to Migrate
Begin
1:partition the Bug Reports to N folds
2:initialize Ant colonies for T,L,F,R,RP
3:for each Ant Colony
4:          set number of ants= Total number of top developers
5:for   fold =1 to N
6:          repeat until ants reach Top developer more than 4 times
7:          repeat until ants reach Top developer node
8:          deploy Ants from first assignee node
9:          choose Next node by Probability Equation
10:lay Pheromone
11:deploy ants
12:endfor
/* Adapt the Number of Ants */
13:for  each Ant Colony
14:Nmig= ((Best -  B)/100)*Tot
15:Number of ants= Number of ants – Nmig
16:TotNmig= TotNmig+ Nmig
17:endfor
18:order according to increasing order Best Solution
19:for  each Ant Colony
20:Number of ants = Number of ants + TotNmig/2
21:TotNmig= TotNmig- TotNmig/2
22:endfor
23:N++
24:endfor/* fold =1..N*/
End
```

Figure 6.26: Co-Ant Algorithm

6.5.3 Performance Evaluation

The experiments were conducted in a Pentium4 Processor machine with 320 GB hard disk. The front end used was Netbeans 7.2 and backend was Oracle 10g. The JDK tool was used to conduct the experiments. The bug reports from bugzilla.org were downloaded from the year 2009 to 2013. Only bugs that with status resolved and finished and with a vote count of two were used to conduct the experiments. The experiments were conducted as three studies. The study 1 comprises bug reports from 2009 to 2011, the study 2 comprises bug reports from 2009 to 2012 and study 3 comprises bug reports from 2009 to 2013. The three studies were used to evaluate the consistency to the experiments. The Co-Ant algorithm deployed over the Enriched Collaboration Graph was compared against the existing GP-WBFS and the proposed AP-WBFS, Bi-Objective and BT-ANT with parameters Path Length, Precision, Recall and Path Similarity. The results were subject to one way ANOVA to study the statistical significance of the results.

6.5.3.1 Path Length

The performance of the Co-Ant with respect to Path Length parameter, compared with the existing GP-WBFS, proposed AP-WBFS, Bi-Objective and BT-ANT is rendered in the Figure 6.27, Figure 6.28 and Figure 6.29 for all the three data sets. From the experimental results, it is evident that among all the proposed system Co-Ant performs superiorly. The statistical analysis of the experimental performance is reported below.

Hypothesis with respect to the result of the parameter P(Path Length)

Null hypothesis H0 : P1 = P2 = P3 = P4=P5, where P1= Path Length obtained in GP-WBFS, P2 = Path Length obtained in AP-WBFS,P3 = Path Length obtained in Bi-Objective, P4=Path Length obtained in BT-ANT and P5=Path Length obtained in Co-Ant

(There is no significant difference among the five systems in terms of Path Length obtained)

Alternate hypothesis H13: Path Length mean values are not equal for at least one pair of the result mean values of the parameter P.

(There is a significant difference among the five systems in terms of Path Length obtained)

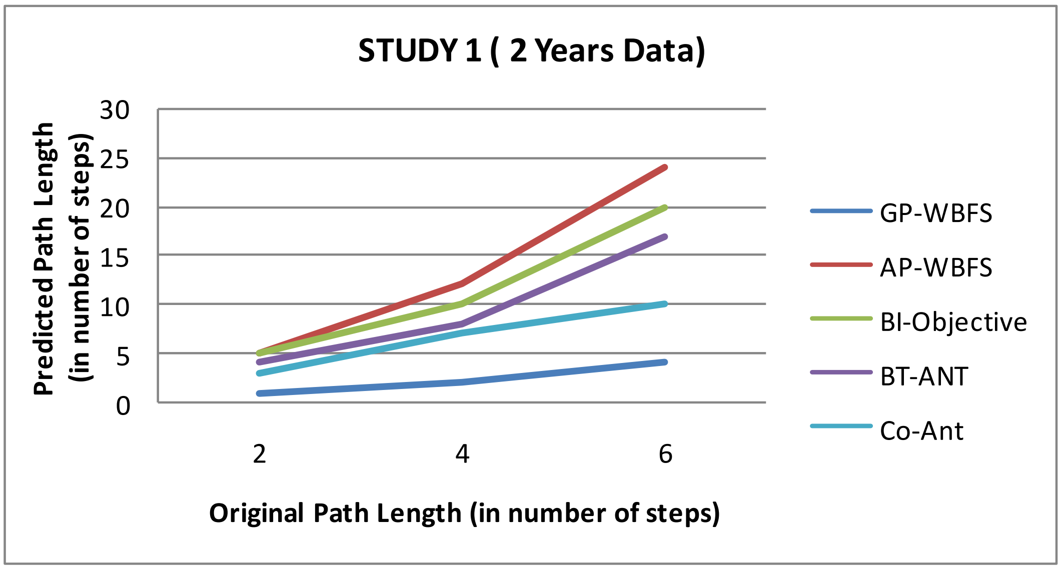


Figure 6.27: Co-Ant: Path Length for Study 1

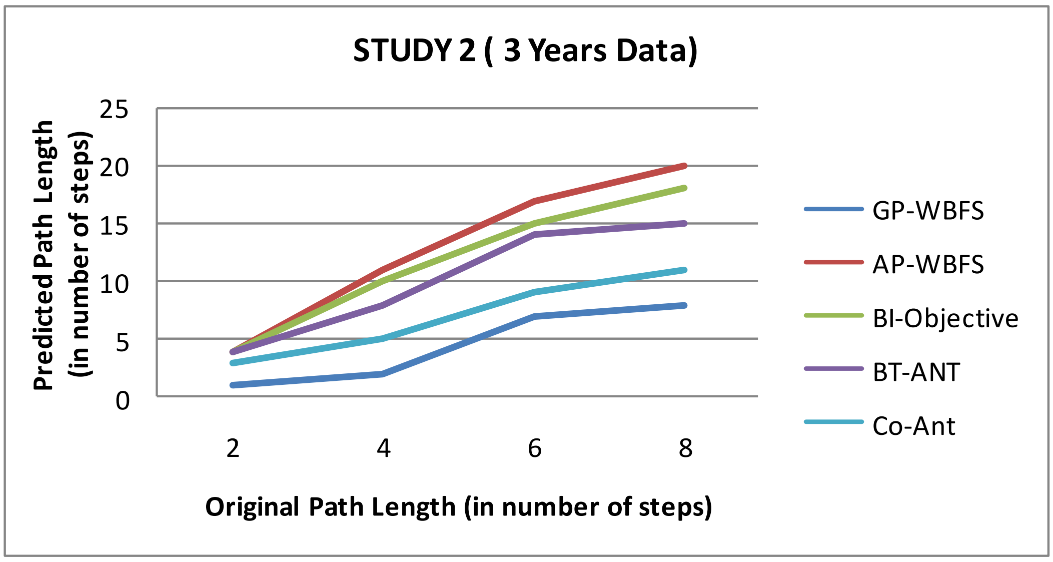


Figure 6.28: Co-Ant : Path Length for Study 2

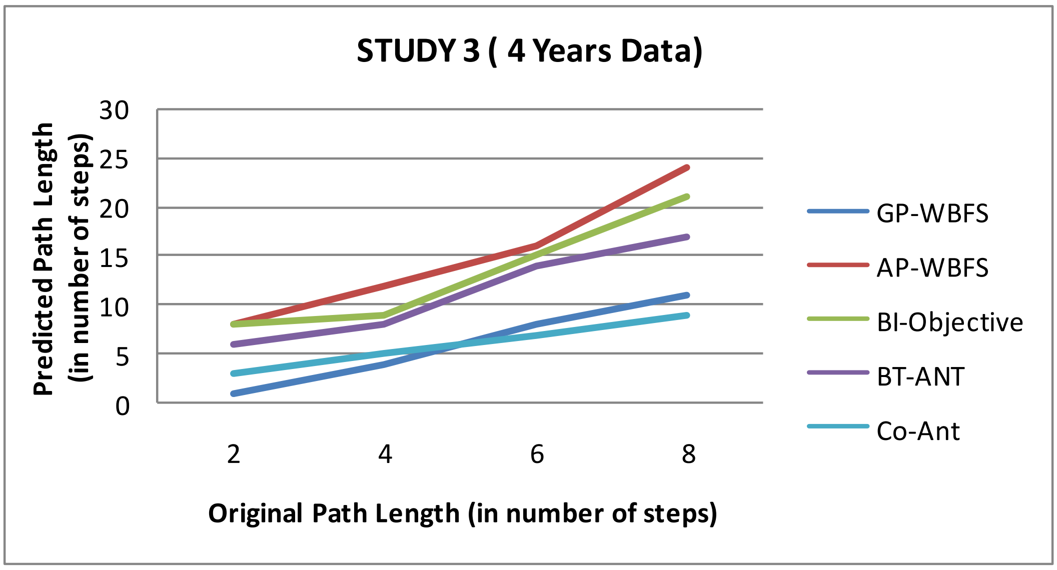


Figure 6.29: Co-Ant : Path Length for Study 3

Hypothesis Evaluation with respect to P (Path Length)

Table 6.5: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant based on Path Length

Hypothesis	**Technique**					**Study**	**Path Length**
	I	**II**	**III**	**IV**	**V**		**p value**
H13	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	Co-Ant	**Study 1**	<0.0001
						Study 2	<0.0001
						Study 3	<0.0001

From the Table 6.5, it is concluded that the calculated significant level of the parameter Path Length by comparing the five systems namely, GP-WBFS,AP-WBFS, Bi-Objective ,BT-ANT and Co-Ant always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Path Length values of the above five systems namely, GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant. Hence, the null hypothesis for H13 may be rejected.

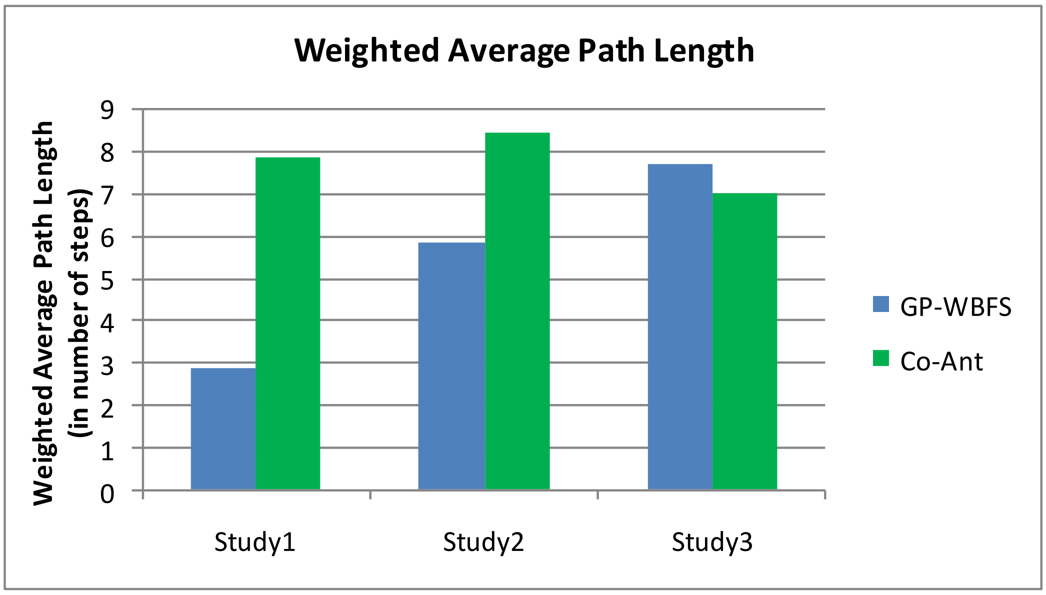


Figure 6.30: Co-Ant : Weighted Average Path Length

The results obtained from the three studies are consolidated using the Weighted Average method. From the Figure 6.30, it is evident that Co-Ant outperforms GP-WBFS, AP-WBFS and Bi-Objective with respect to Path Length. The rate of increase of Path Length of Co-Ant with respect to GP-WBFS is 6.74.

6.5.3.2 Precision

The performance of the Co-Ant with respect to Precision parameter,compared with the existing GP-WBFS, proposed AP-WBFS, Bi-Objective and BT-ANT is rendered in Figure 6.31, Figure 6.32 and Figure 6.33 for all the three data sets. From the experimental results, it is evident that among all the proposed system Co-Ant performs superiorly. The ANOVA analysis of the experimental performance is given below.

Hypothesis with respect to the result of the parameter PR(Precision)

Null hypothesis H0: PR1 = PR2 = PR3 = PR4 = PR5, where PR1= Precision obtained in GP-WBFS, PR2 = Precision obtained in AP-WBFS, PR3 = Precision obtained in Bi-Objective, PR4=Precision obtained in BT-ANT and PR5=Precision obtained in Co-Ant.

(There is no significant difference among the five systems in terms of Precision obtained)

Alternate hypothesis H14: Precision means values are not equal for at least one pair of the result mean values of the parameter PR.

(There is a significant difference among the five systems in terms of Precision obtained)

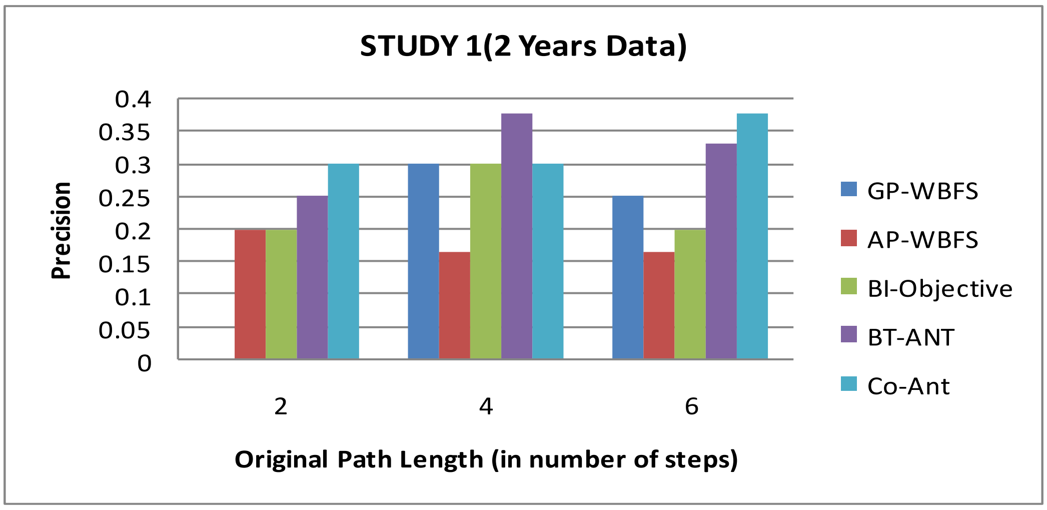


Figure 6.31: Co-Ant: Precision for Study 1

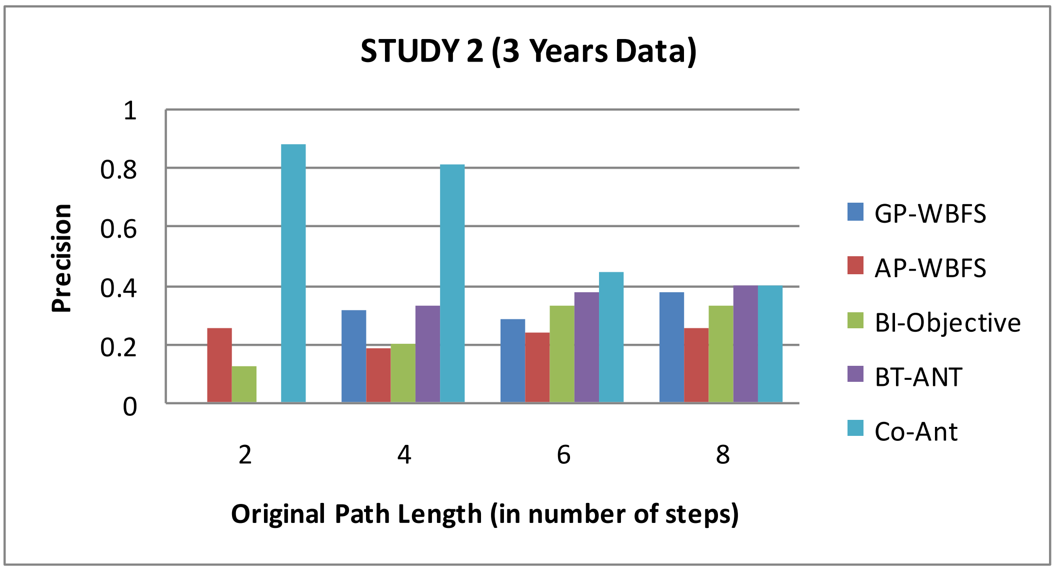


Figure 6.32: Co-Ant: Precision for Study 2

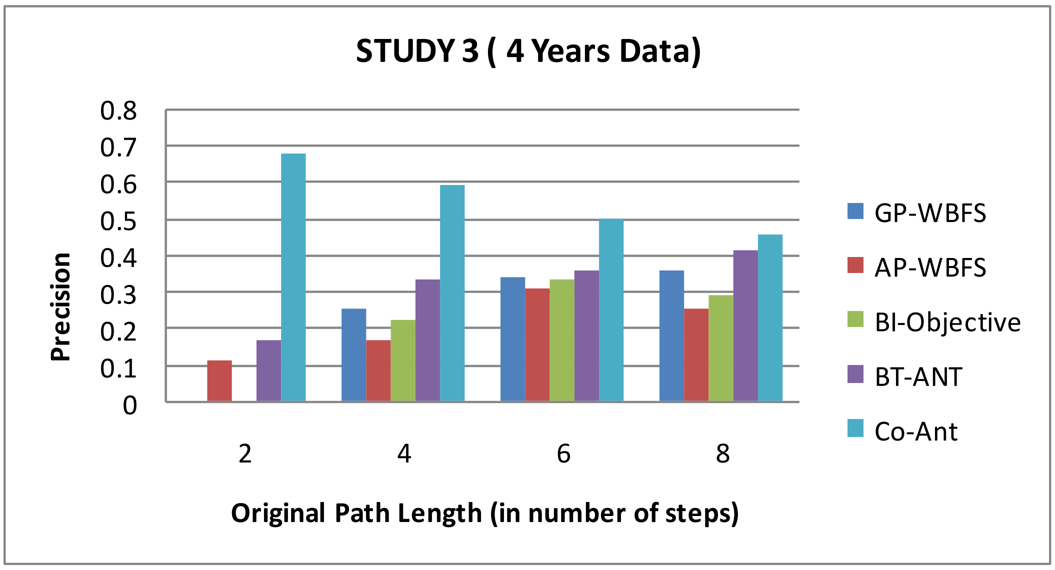


Figure 6.33: Co-Ant: Precision for Study 3

Hypothesis Evaluation with respect to PR (Precision)

Table 6.6: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant based on Path Length

Hypothesis	**Technique**					**Study**	**Precision**
	I	**II**	**III**	**IV**	**V**		**p value**
H14	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	Co-Ant	**Study 1**	<0.0001
						Study 2	<0.0001
						Study 3	<0.0001

From the Table 6.6, it is concluded that the calculated significant level of the parameter - Precision by comparing the five systems namely, GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Precision values of the above five systems namely, GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant. Hence, the null hypothesis for H14 may be rejected.

The results obtained from the three studies are consolidated using the weighted average Precision. Based on which, the results are consolidated in the Figure 6.34.

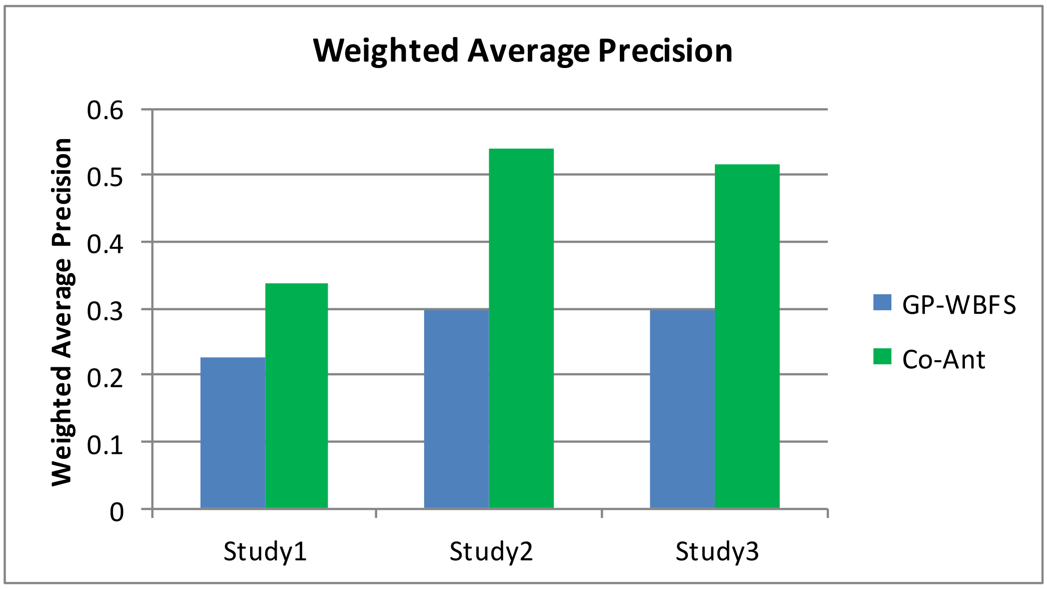


Figure 6.34: Co-Ant: Weighted Average Precision

The rate on increase of Precision of Co-Ant with respect to GP-WBFS is 0.708.

6.5.3.3 Recall

The performance of the Co-Ant with respect to Recall parameter, compared with the existing GP-WBFS, proposed AP-WBFS, Bi-Objective and BT-ANT is rendered in Figure 6.35, Figure 6.36 and Figure 6.37for all the three data sets. From the experimental results, it is evident that among all the proposed systems, Co-Ant performs superiorly. The statistical analysis of the experimental performance is given below.

Hypothesis with respect to the result of the parameter R: (Recall)

Null hypothesis H0 : R1 = R2 = R3 = R4 = R5, where R1= Recall obtained in GP-WBFS, R2 = Recall obtained in AP-WBFS,R3 = Recall obtained in Bi-Objective, R4=Recall obtained in BT-ANT and R5=Recall obtained in Co-Ant.

(There is no significant difference among the five systems in terms of Recall obtained)

Alternate hypothesis H15: Recall mean values are not equal for at least one pair of the result mean values of the parameter R.

(There is a significant difference among the five systems in terms of Recall obtained)

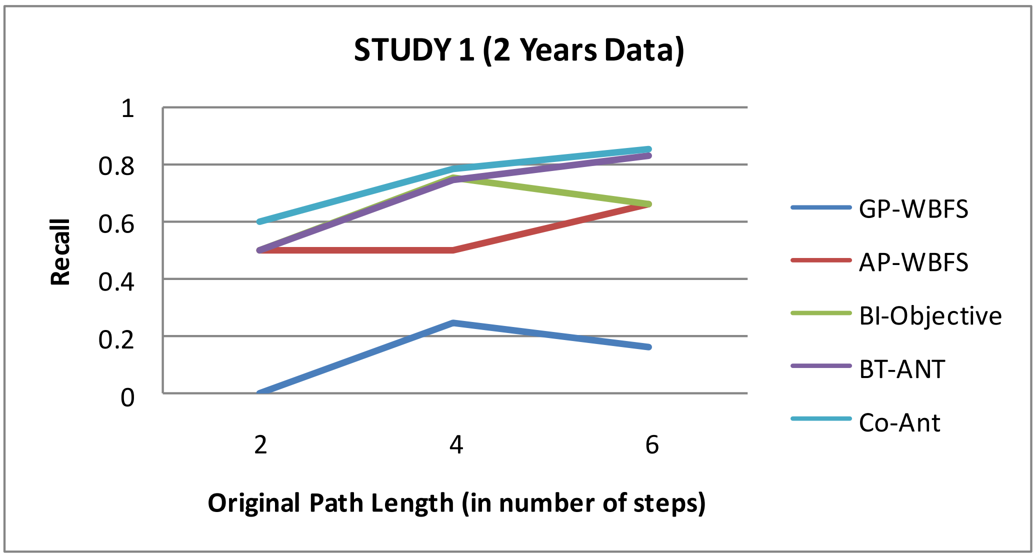


Figure 6.35: Co-Ant: Recall for Study 1

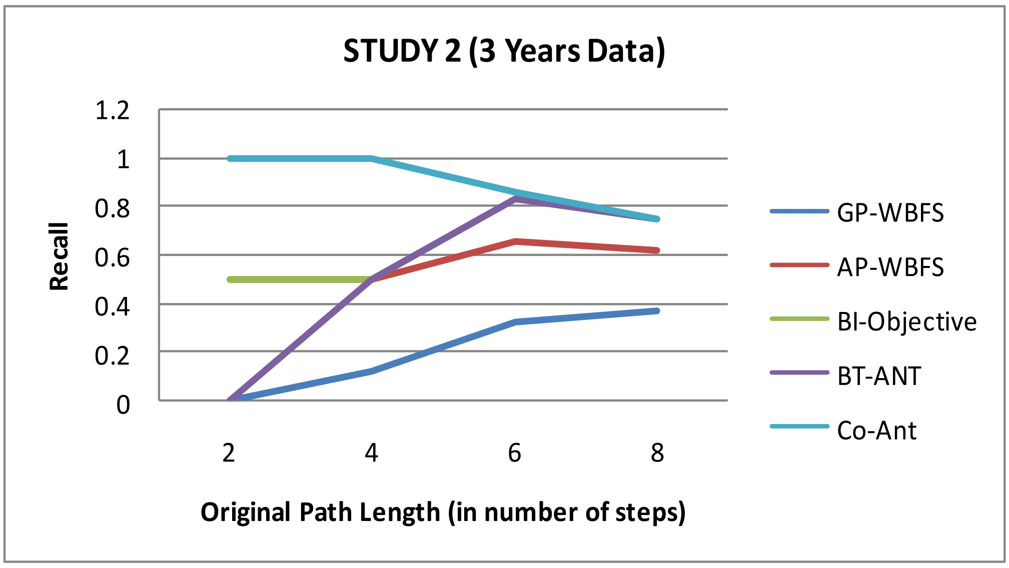


Figure 6.36: Co-Ant: Recall for Study 2

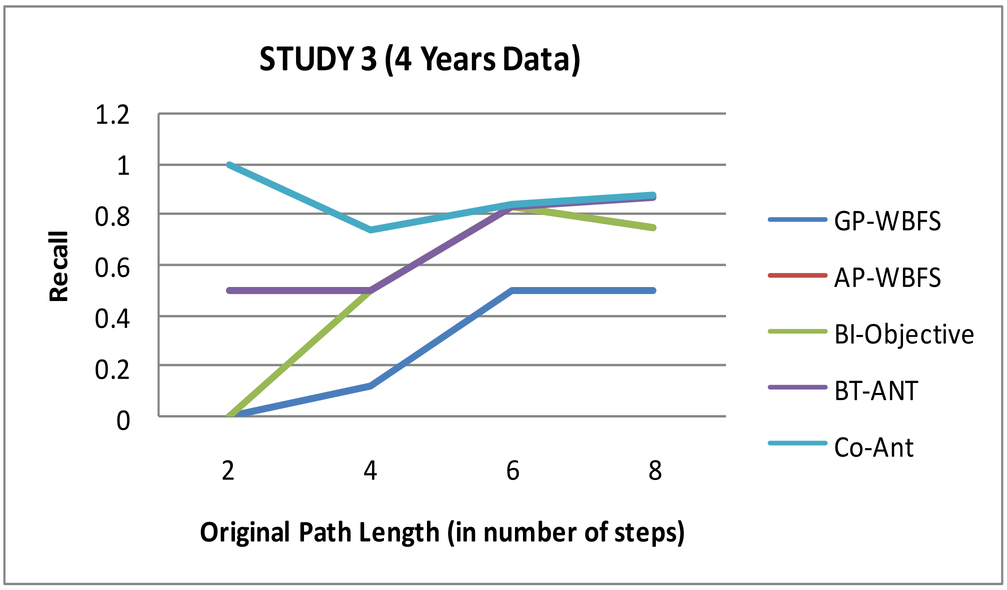


Figure 6.37: Co-Ant: Recall for Study 3

Hypothesis Evaluation with respect to R (Recall)

Table 6.7: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant based on Recall

Hypothesis	**Technique**					**Study**	**Recall**
	I	**II**	**III**	**IV**	**V**		**p value**
H15	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	Co-Ant	**Study 1**	<0.0001
						Study 2	<0.0001
						Study 3	<0.0001

From the Table 6.7, it is concluded that the calculated significant level of the parameter Recall by comparing the five systems namely, GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Recall values of the above five systems namely, GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant. Hence, the null hypothesis for H15 may be rejected.

The results obtained from the three studies are consolidated using the Weighted Average Recall in the Figure 6.38.

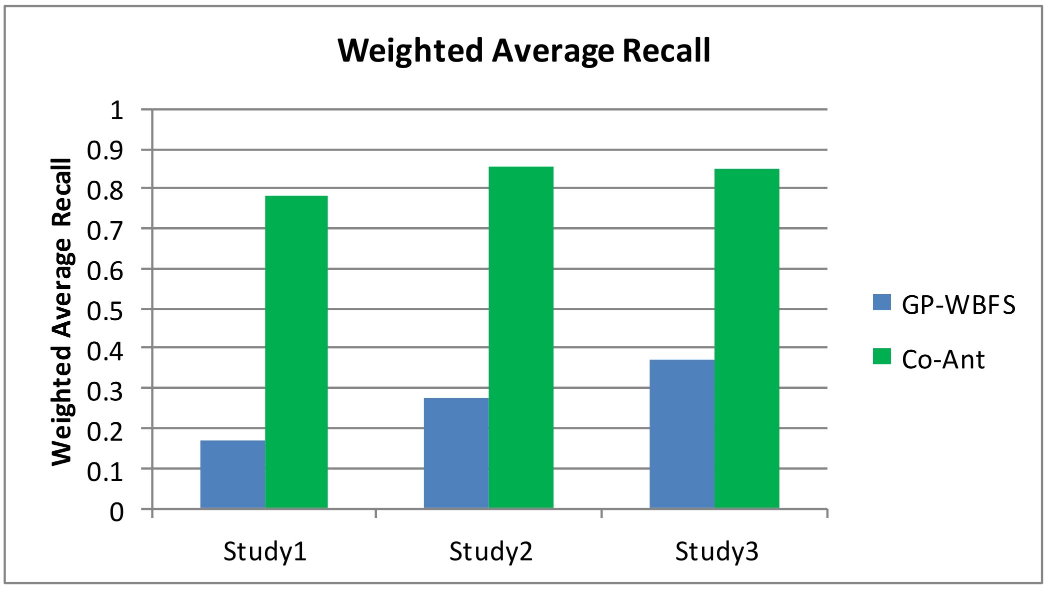


Figure 6.38: Co-Ant: Weighted Average Recall

The rate on increase of Precision of Co-Ant with respect to GP-WBFS is 2.05.

6.5.3.4 Path Similarity

The performance of the Co-Ant with respect to Path Similarity parameter, compared with the existing GP-WBFS, proposed AP-WBFS, Bi-Objective and BT-ANT is manifested in Figure 6.39, Figure 6.40 and Figure 6.41for all the three data sets. From the experimental results, it is evident that among all the proposed systems, Co-Ant performs superiorly. The statistical analysis of the experimental performance is given below.

Hypothesis with respect to the result of the parameter PS (Path Similarity)

Null hypothesis H0 : PS1 = PS2 = PS3 = PS4 = PS5, where PS1= Path Similarity obtained in GP-WBFS, PS2 = Path Similarity obtained in AP-WBFS,PS3 = Path Similarity obtained in Bi-Objective, PS4=Path Similarity obtained in BT-ANT and PS5=Path Similarity obtained in Co-Ant.

(There is no significant difference among the five systems in terms of Path Similarity obtained)

Alternate hypothesis H16: Path Similarity mean values are not equal for at least one pair of the result mean values of the parameter PS.

(There is a significant difference among the five systems in terms of Path Similarity obtained)

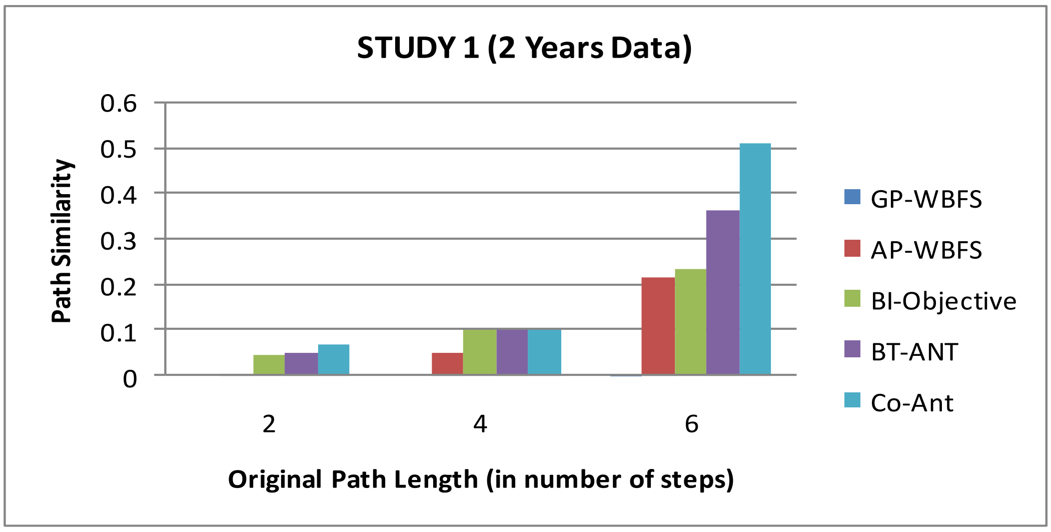


Figure 6.39: Co-Ant: Path Similarity for Study 1

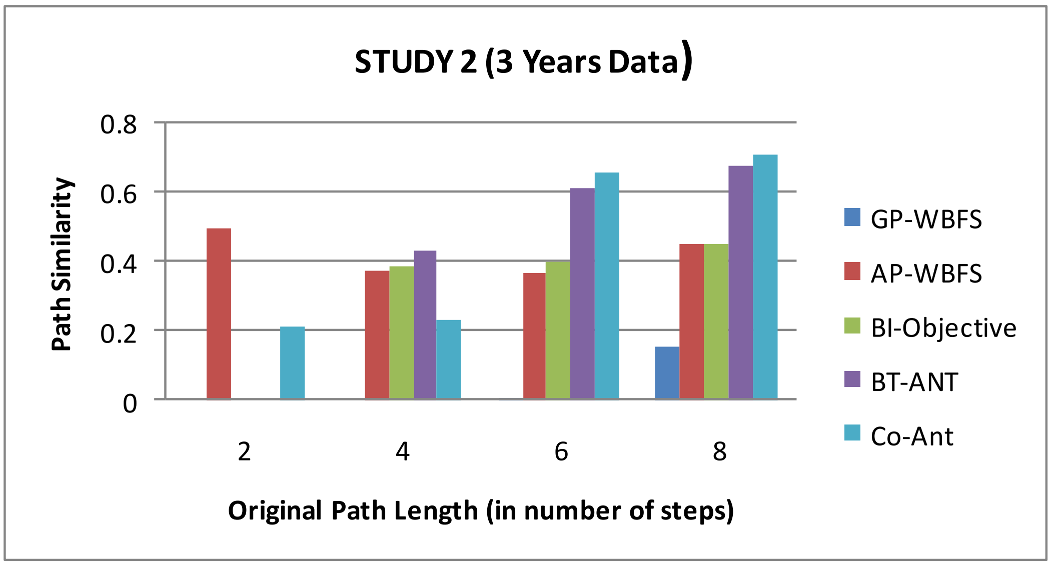


Figure 6.40: Co-Ant: Path Similarity for Study 2

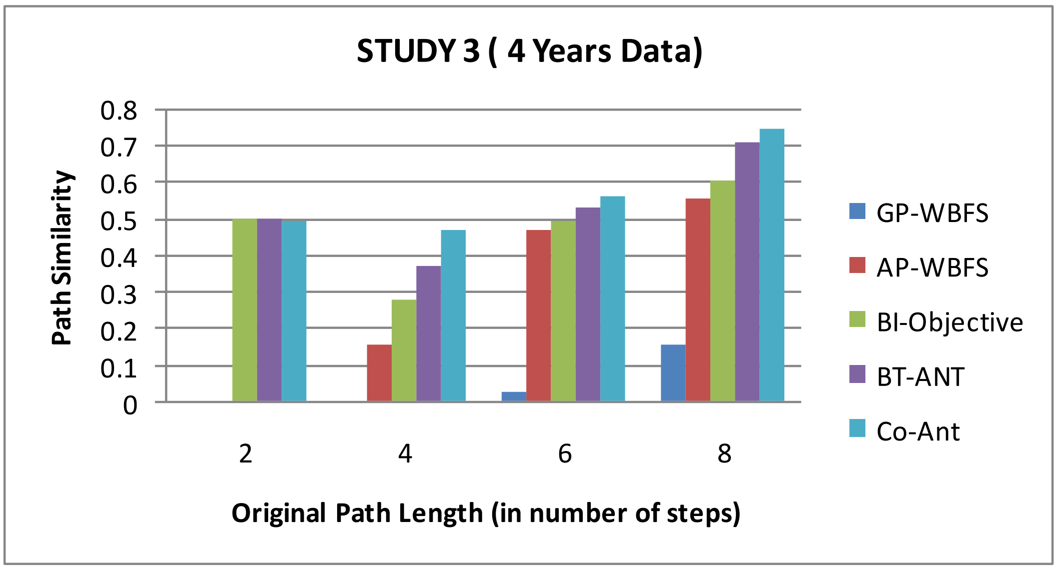


Figure 6.41: Co-Ant: Path Similarity for Study 3

Hypothesis Evaluation with respect tops (Path Similarity)

Table 6.8: ANOVA for GP-WBFS, AP-WBFS, Bi-Objective, BT-ANT and Co-Ant based on Path Similarity

Hypothesis	**Technique**					**Study**	**Path Similarity**
	I	**II**	**III**	**IV**	**V**		**p value**
H16	GP-WBFS	AP-WBFS	Bi-Objective	BT-ANT	Co-Ant	**Study 1**	<0.0001
						Study 2	<0.0001
						Study 3	<0.0001

From the Table 6.8, it is assured that the calculated significance level of the parameter Path Similarity by comparing the five systems namely, GP-WBFS,AP-WBFS, Bi-Objective, BT-ANT and Co-Ant always satisfy the condition (p value<0.05) for all the three studies. There is significant difference among the results for different Path Similarity values of the above five systems namely, GP-WBFS,AP-WBFS, Bi-Objective, BT-ANT and Co-Ant. Hence, the null hypothesis for H16 may be rejected.

The results obtained from the three studies are consolidated using the Weighted Average Path Similarity and are illustrated in the Figure 6.42.

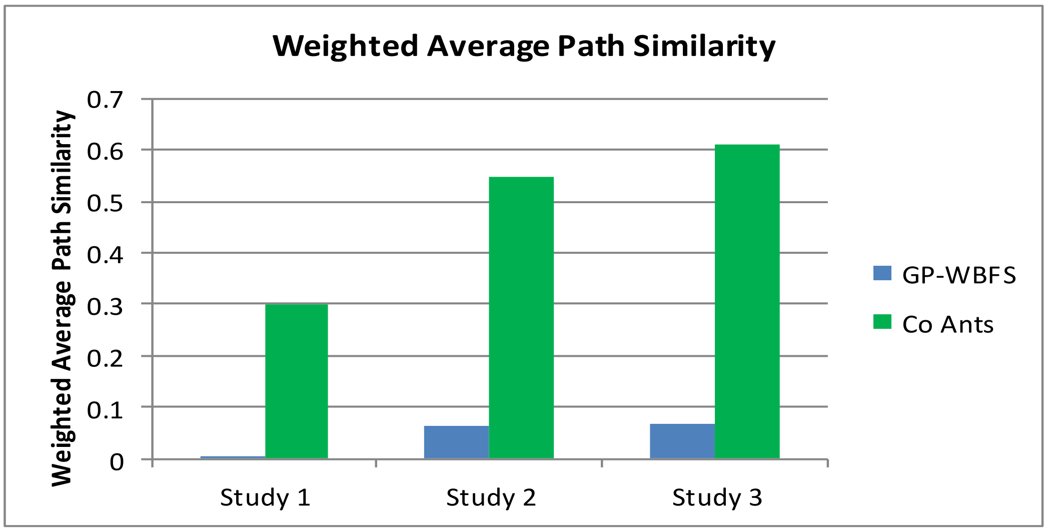


Figure 6.42: Co-Ant: Weighted Average Path Similarity

The rate on increase of Path Similarity of Co-Ant with respect to GP-WBFS is 9.84.

6.6 SUMMARY

This chapter presents a Bug Triage System based on ant system. The OSS development model is an evolving, dynamic model of development. The relationship among the developers is modeled based on their tossing activities.The Bug Triage System based on ant system is deployed over the Bug Toss Graph. The ant system implements adaptivity by using the positive and negative feedback mechanism. The implementation of adaptive techniques for Bug Triage over the Bug Toss Graph delivers promising results.

This chapter also casts the Bug Toss Graph as Enriched Collaboration Graphs. The Enriched Collaboration Graph is rich in structure, as it captures the social structure by adding dimensions to the relationship that exist among the developers. The multi objective cooperating ant algorithm is used on the Enriched Collaboration Graph to infer the set of developers who can collaborate on a bug. The cooperating ant algorithm is adaptive in nature. It converges more rapidly towards a solution because of the ant migration feature embedded in it. This ant migration feature helps the solution to be achieved at a faster rate. Further, the entire path learning concept is contained in an Incremental Learning framework. The Incremental Learning framework helps in adjusting the underlying graph according to the changes added to the underlying structure. The fixed window Incremental Learning framework is applied in the Ameliorated Bug Triage System.

The experimental results indicate that the Bug Triage System based on Co-Ant algorithm outperforms the existing Bug Triage System based on GP-WBFS with respect to the parameters - Precision, Recall and Path Similarity. The Bug Triage System based on Co-Ant algorithm outperforms AP-WBFS, Bi-Objective and BT-ANT with respect to the parameters - Path Length, Precision, Recall and Path Similarity.

CHAPTER - 7

HOLISTIC EVALUATION FRAMEWORK: INTEGRATION OF DEVELOPER PERFORMANCE

7.1 PREAMBLE

This chapter furnishes an Holistic Evaluation Framework for Bug Triage System by integrating the performance of the developers. Key Performance Indicators(KPI)are proposed for appraising a developer's effectiveness in contribution towards the resolution of the bug. The Bug Triage System retrieves a set of developers for resolving a bug. By applying the KPIs on the retrieved set of developers, the Bug Triage System is evaluated quantitatively. This is because the metrics that are used to evaluate the Bug Triage System in the literature is only recommendation centric [95]. That is the existing metrics cover only the correctness, coverage, etc of the Bug Triage System. There is a need for user centric evaluation of the Bug Triage System.

7.2 DEVELOPER PERFORMANCE ASSESMENT

Developer Performance Assessment is a necessity in identifying the strength and weakness of a developer, for career advancement, and fine tuning a business organization [96]. Contribution of a developer towards the software maintenance is quite different from a developer's contribution in developing a software product. Measuring a developer's contribution towards the maintenance of an OSS System is even more complicated. This is because there are no explicit assigned roles for the developers. But, yet, there are different roles a developer may assume in the course of bug resolution.

The different roles that the developer may play are reporter of a bug, triager, commenter and assignee [97]. In OSS, usually there are metrics for evaluating the bug characteristics. These metrics focus on the program slicing characteristics of the bug like number of lines of code affected by the bug, Cyclomatic Complexity of the bug, etc [98]. But, these metrics are underutilised in evaluating the Bug Triage System. Further, the developer's performance may be assessed based on Buggy commits, code contributions and priority bugs. Buggy commits are used to identify developers who performed less buggy commits. Code contribution is measured in terms of code addition, code removal, method addition and method modification. The developer may also be assessed in terms of the number of high priority bug that he has resolved [99]. In most of the existing works, developer's performance assessment is treated as an independent module. In the following section,the developer's performance assessment is integrated in the evaluation of the Bug Triage System. There are several KPIs proposed to assess the developer. These indicators are then utilized in quantifying the performance of the Bug Triage System.

7.2.1 Key observations from the dataset

This section gives a brief preview of the various factors that affect the bug resolution which is observed in the dataset. The bug reports of Eclipse project from www.bugzilla.org from 2009 to 2013 were analyzed. The developers contribution for the various fields in the bug report like CC, status, Keywords, Summary priority, Assignee, resolution etc are given in the Figure 7.1.It is evident that 62% of the developers change the status of the bug to 'resolved'.

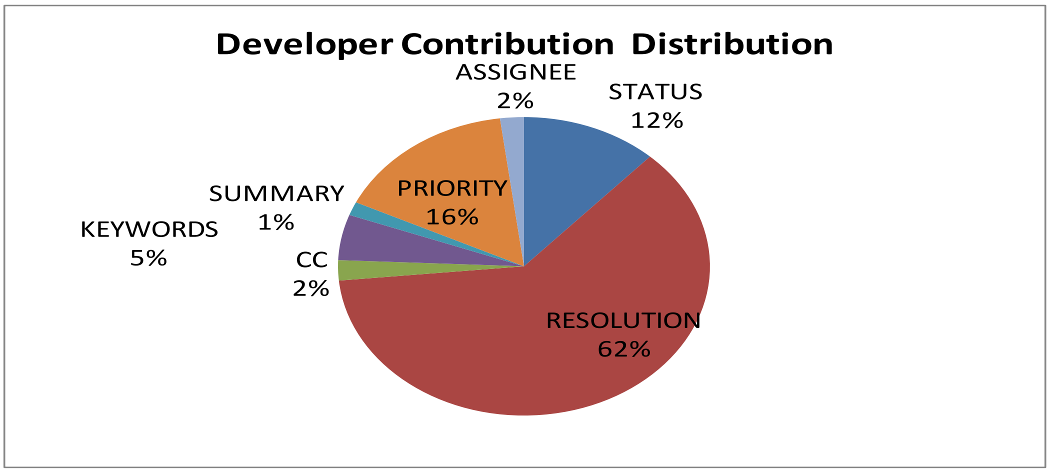


Figure 7.1: Developer Contribution Distribution

The average time spent by a developer on a particular field of the bug report is given in the Figure 7.2. As it can be observed, the time spent to set the assignee field, status field and resolution field contribute mostly in the bug resolution time.

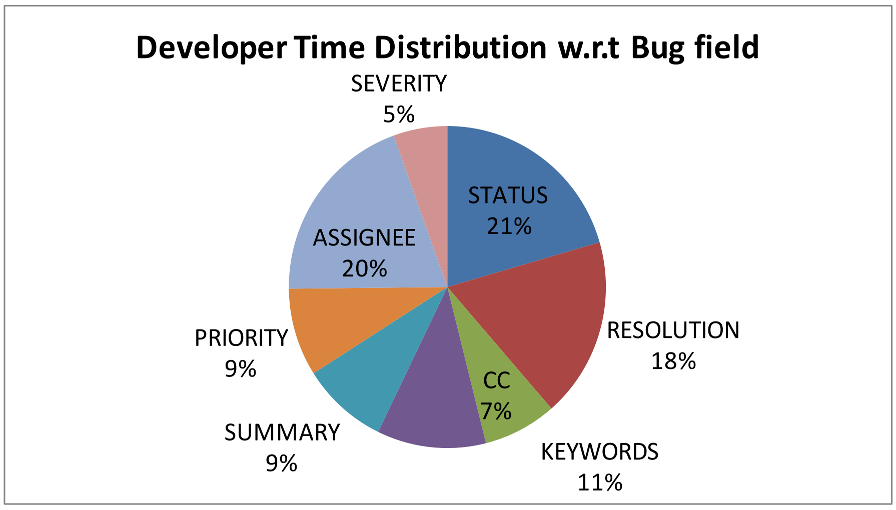


Figure 7.2: Developer Time Distribution w.r.t Bug field

The developer distribution with respect to the time spent by a developer on a particular bug is given in the Figure 7.3. It can be observed that 39% of the developers spend 121 to 700 days on a particular bug. Only 17% of the developers spend less than 6 months on a bug. Any Bug Triage System that extracts its set of developers mostly from this pool of 17% is a successful triage system.

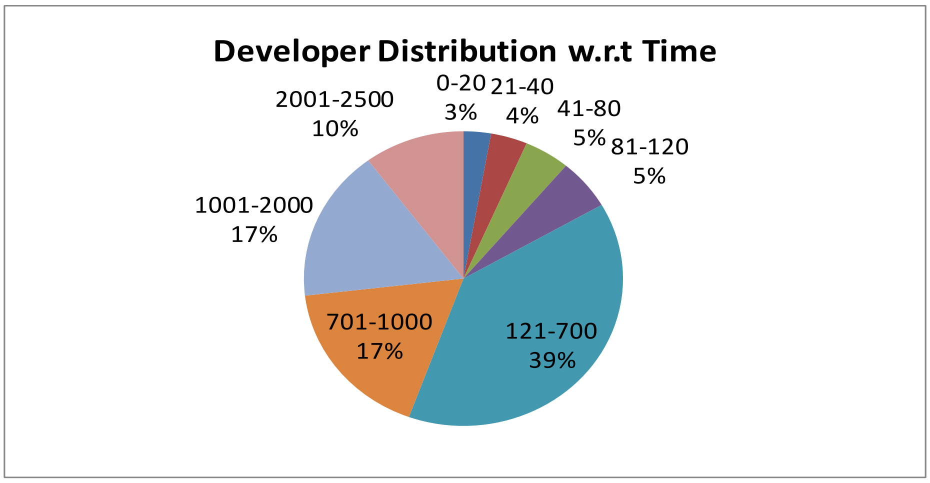


Figure 7.3: Developer Distribution w.r.t Time

The Figure 7.4 shows the Developer Distribution against the range of Bug Resolution Time, measured in days. It can be observed that the most ineffective bug resolution is when the bug resolution time is more than 2 years. There are 44% of developers who spend time on bug whose resolution time is > 2 years. It can be observed from the chart that only 25% of developer has spent time in bugs that were resolved before six months. The motivation behind any Bug Triage System is to retrieve the developers from this pool of 25% of developers.

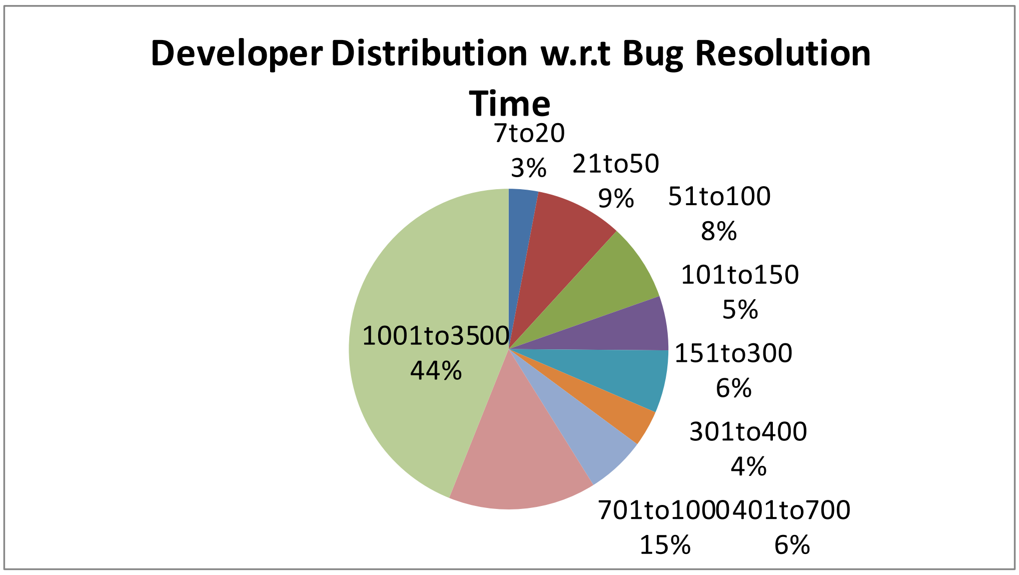


Figure 7.4: Developer Distribution w.r.t Bug Resolution Time

Based on these observations, the Key Performance Indicators for assessing the Developer is introduced in the next section.

7.3 KEY PERFORMANCE INDICATORS FOR ASSESSING DEVLOPER PERFORMANCE

The KPIs introduced to assess the developer are Developer Time Index,Developer Effective index and Developer Productivity. The Developer Time Index, Developer Effective index and Developer Productivity are derieved from Developer Contribution Count and Developer Contribution Time. The dependencies among the KPIs are depicted in Figure 7.5.

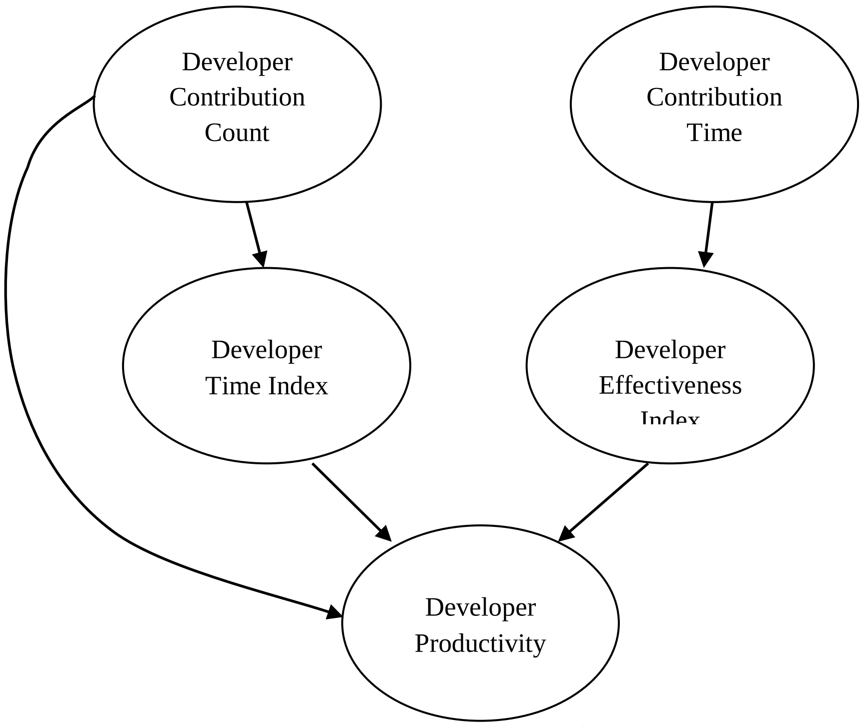

Developer Contribution Count
Developer Contribution Time
Developer Time Index
Developer Effectiveness Index
Developer Productivity

Figure 7.5: Dependency in the Key Performance Indicators

The various kinds of contribution which are contributed by the developers are reassign the bug, change the Status field or finally resolve the bug. For convenience, all the contributions are equally treated.

7.3.1 Developer Contribution Count

Developer Contribution Count (DCC) is defined as the number of contributions made by each developer in the process of resolving them.

$$\mathrm{DCC} = \sum_{i}^{n} \mathrm{C_i}$$

where, C_i - Contribution by a developer to a single bug.

n - Total number of bugs assigned to a developer

7.3.2 Developer Contribution Time

Developer Contribution Time (DCT) is defined as the time taken by each developer to make a contribution on a single bug.

$$\mathrm{DCT} = \forall\ \mathrm{Bug(Developer)} \sum(\mathrm{DBR} - \mathrm{DBA})$$

where, DBR- Date of Bug Reassignment

DBA – Date a Bug Assignment

7.3.3 Developer Time Index

Developer Time Index (DTI) is defined as the ratio of DCT to DCC. This indicator captures the amount of time taken by a developer to make a single contribution.

$$\mathrm{DTI} = \frac{\mathrm{DCT}}{\mathrm{DCC}}$$

7.3.4 Developer Effectiveness Index

The bug resolution time is considered to calculate the Developer Effectiveness Index (DEI). The intuition behind DEI is that, if a developer has contributed towards a bug that has been resolved with less time, then the developer's effectiveness is increased. Contrarily, if a developer has contributed towards a bug that has taken a long time to resolve, then the weight assigned to the developer is reduced.

Table 7.1: Weight Assignment Table

Bug Resolution Time (in days)	**Weights**
7– 20	7
21-50	6
51-100	5
101-150	4
151-300	3
301-400	2
401-700	1
701-1000	-0.25
1001-3500	-0.50

The bugs for 10 years were studied and the Resolution Time (RT) was extracted. RT varies from lower to higher values. RT was divided into nine ranges and their weights were assigned as given in the Table 7.1. The highest weight is assigned to the range of Resolution Time that falls between 7 to 20 days. Negative weights are assigned to a range which took more than 700 days to resolve a bug.

The weights given here are inversely proportional to RT.

$$\text{Weight } (W_i) \alpha \frac{1}{RT}$$

DEI is defined as the ratio of the summation of Weights W_i of the bugs to the DCC.

$$\text{Developer Effectiveness Index DEI} = \frac{1}{DCC} \sum_i^{DCC} W_i$$

7.3.5 Developer Productivity

Developer Productivity (DP) is defined as the product of Developer Effectiveness Index, Developer Contribution Count and the Developer Time Index.

$$DP = DEI * DTI * DCC$$

7.3.6 A Holistic Evaluation Framework with Developer Performance

The framework for evaluating the Bug Triage System is given in the Figure 7.6. The Bug Triage System extracts the optimal set of developers. KPIs of the retrieved developers are calculated and thereby, the Bug Triage System is assessed.

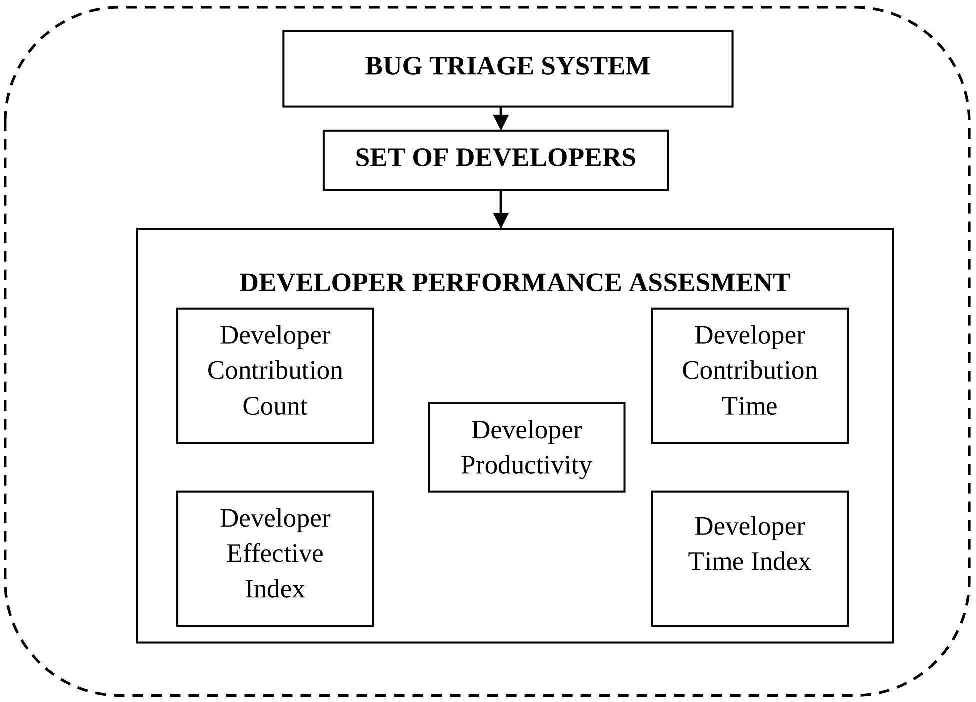

BUG TRIAGE SYSTEM
SET OF DEVELOPERS
DEVELOPER PERFORMANCE ASSESMENT
Developer Contribution Count
Developer Contribution Time
Developer Productivity
Developer Effective Index
Developer Time Index

Figure 7.6: A Holistic Evaluation Framework with Developer Performance

7.3.6.1 Performance Evaluation of Bug Triage System with KPIs

The performance of the existing system GP-WBFS, BT-ANT and the Co-Ant were analyzed using the KPIs of Developer Productivity, Developer Effectiveness and Developer Time Index. The existing GP-WBFS was compared only with BT-ANT and the Co-Ant because only in these systems adaptive learning was adopted. The graph for Developer Time Index is given in the Figure 7.7. It is evident from the Figure 7.7 that the Developer Time Index for the proposed Co-Ant as well as the BT-ANT is skewed towards Developer Time Index of<300. Almost 85% of the retrieved developers by Co-Ant has a Developer Time Index of <300 and 65% of the developers retrieved by BT-ANT has a Developer Time Index of <300,whereas in the existing GP-WBFS, 77% of the developers have a Developer Time Index >300.

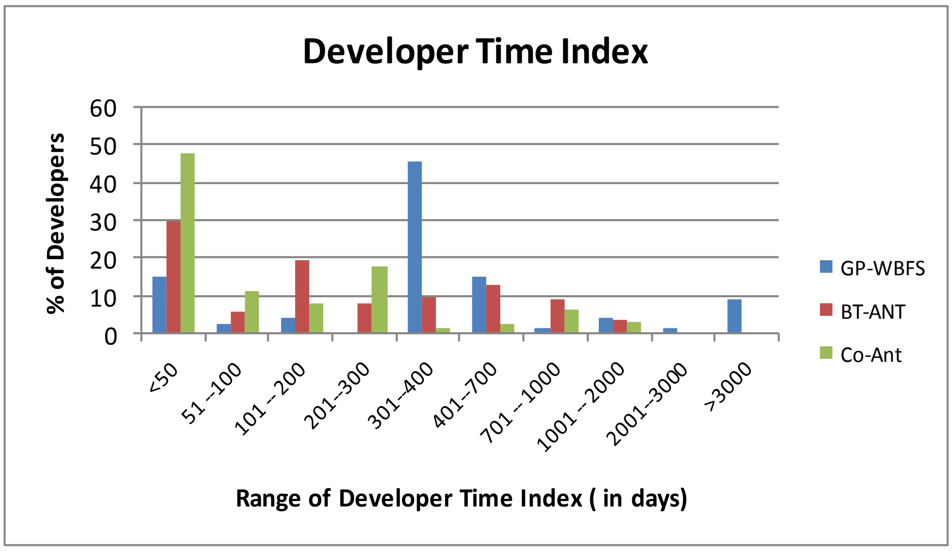


Figure 7.7: Developer Time Index

The performance of the systems for Developer Effectiveness Index is given in the Figure 7.8. Developer Effectiveness Index encodes the contribution of the developers for bugs that were resolved in a shorter period of time.

From the graph, it is evident that 88% of the developers retrieved by Co-Ant possess a Developer Effectiveness Index of >60 and 78% of the developers retrieved by the BT-ANT possess a Developer Effectiveness Index of >60, whereas in the developers retrieved by GP-WBFS, 78% of the developers have a Developer Effectiveness Index of <60.

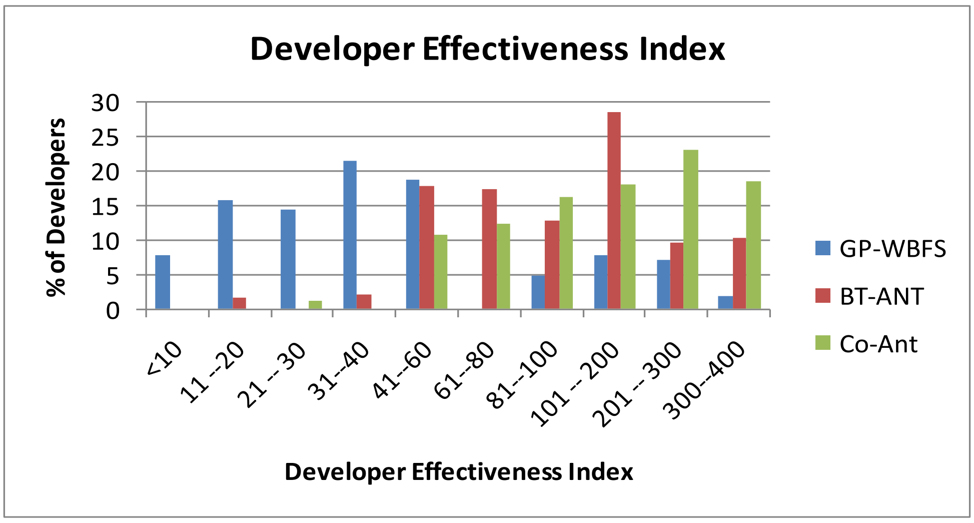


Figure 7.8: Developer Effectiveness Index

The performance of the systems for Developer Productivity is given in the Figure 7.9. Developer Productivity is a cumulative index that encodes the Developer Effectiveness, Developer Time Index and Developer Contribution Count.

From the Figure 7.9, it is evident that 91% of the developers retrieved by Co-Ant possess a Developer Productivity of >50 and 75% of the developers retrieved by the BT-ANT possess a Developer Productivity of >50, whereas in the developers retrieved by GP-WBFS, 73% of the developers have a Developer Productivity of <50.

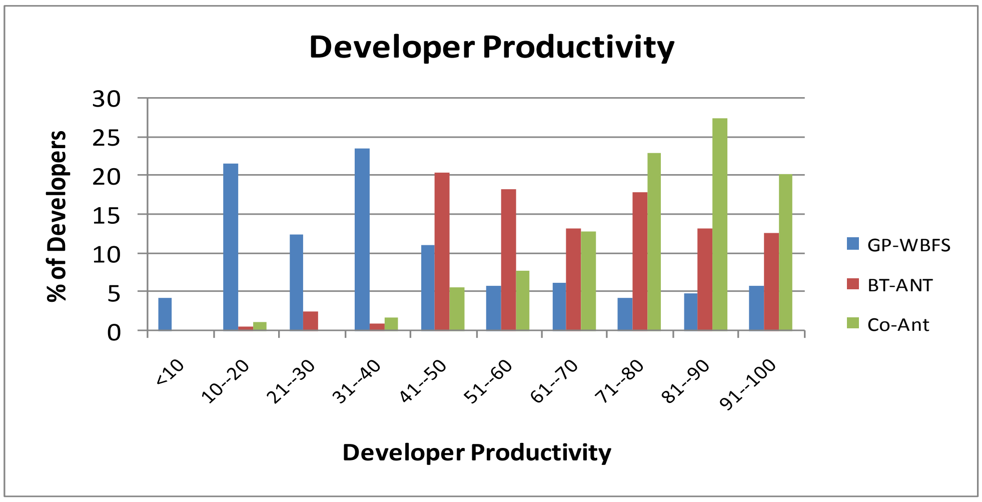


Figure 7.9: Developer Productivity

7.4 SUMMARY

This chapter presents an Holistic Evaluation Framework for Bug Triage using the performance of the developers. The existing metrics that were used to evaluate the Bug Triage System were recommendation centric. The recommendation centric metrics evaluated the correctness and completeness of the recommendation mostly based on Precision and Recall measures. This chapter adds a new dimension to the evaluation of the Bug Triage System. A new evaluation metrics based on the usefulness of the Bug Triage System are proposed. This is done by computing Key Performance Indicator values for the performance of the developers involved in the bug resolution. These calculated indices are then utilized to evaluate the Bug Triage System.

CHAPTER - 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

Over the years, the OSS model has evolved to a potent phase. While the straightforward benefit of OSS is licence free software, the ancillary benefit is the availability of software repositories in the public domain. The software repositories contain the DNA of a software project. The information in these repositories can be utilized in making decisions so as to ease the OSS maintenance. Bug management forms a significant aspect of OSS maintenance. Bug management encompasses Bug Triage, reassignment of bug and finally, resolving of the bug. Bug Triage has proved to be sluggish and erroneous, if done manually. Under these circumstances, automated support to Bug Triage is unavoidable.

The main goal of this thesis is to ameliorate the Bug Triage process by making the following contributions:

i) An Enriched Collaboration Graph was developed. The Enriched Collaboration Graph augments the Number of tosses with additional attributes like Recentness, Frequency, Longevity and Reciprocity that exist in the relationship among the developers. First, the performance of the GP model was compared with the AP model. The data for the experiments were the bug reports of Eclipse project for the period from 2009 to 2013 from the website - www.bugzilla.org. AP model was designated as the model for Bug Toss Graph based on its performance in terms of the parameters viz., Path Length, Precision, Recall and Path Similarity. The Bi-Objective Optimaization algorithm for Bug Triage was used over the AP model. Further, the Bug Toss Graph was enhanced to an Enriched Collaboration Graph.

ii) Adaptive techniques based on ant systems have been designed and implemented for retrieving the referral chain of developers.The Bug Triage System based on Ants was developed as the learning model for the Bug Toss Graph based on AP model. The same dataset was utilized for the experiments. The Bug Triage System based on Ants outperformed the baseline system based on GP-WBFS as well as the proposed AP-WBFS comprehensively. Moreover, the Multi-Objective Co-Ant algorithm was deployed on the Enriched Collaboration Graph. The Co-Ant algorithm excelled the Bug Triage System based on Ants in terms of parameters viz., Path Length, Precision, Recall and Path Similarity.

iii) An Holistic Evaluation Framework has been developed to evaluate the proposed Ameliorated Bug Triage System. The evaluation of the Ameliorated Bug Triage System was performed using the recommendation metrics viz., Precision and Recall as well as user metrics.Further, the Path Similarity metric based on Levenshtein Similarity has been developed to reinforce the recommendation metrics. The user metrics have their base in the developer performance encoded as KPIs of Developer Productivity, Developer Effectiveness and Developer Time Index.

The experimental results were subject to statistical evaluation by ANOVA. The statistical analysis substantiated that there was a significant difference in performance between the Ameliorated Bug Triage System based on the base line GP-WBFS and the proposed AP-WBFS, Bi-Objective, BT-ANT and Co-Ant algorithms. Thus, the Ameliorated Bug Triage System has been envisioned and implemented successfully.

8.2 FUTURE ENHANCEMENT

The Bug Triage process can be enhanced further by integrating the social context of the developers in the collaboration graph.

The collaboration among the developers can be modeled in the underlying graph structure. The current graph model uses a probabilistic graph model to capture the developer structure. This structure encodes the intensity of the bug transfers among the developers in the bug resolution. However, the objective of the Bug Triage System is to resolve a bug and retrieve the efficient developers. The temporal aspect of ‘How much time a developer takes to toss a bug’ is mostly ignored. This information can be built into the graph model by applying Probabilistic Timed Automata to capture the underlying graph structure.

In this thesis, attempt has been made to develop an Holistic Evaluation Framework for the Bug Triage System in a coarse grained manner. It is coarse grained in the sense that the KPIs are based on the number of contributions, time taken for developer contribution and time taken for bugs to be resolved. This can be fine grained by adding additional layers like type, product, component, the operating system etc to these KPIs. Further, at present, there is no differentiation among the developers in terms of their roles. In future, the roles played by the developers can also be incorporated in the evaluation framework.

The social context of the developer is at present modeled based on the tosses, comments or both. Other additional dimensions like the geographical location, time zone, ethnicity, etc., can also be integrated to scrutinize the social context of the developer.

REFERENCES

[1] Prajod S Vettiyattil, "Open Source Software and the Enterprise," *Wipro Limited*, 2014.

[2] "Is Outsourcing the Answer in an Economic Downturn?," *ActiveState Software Inc*, 2009.

[3] Sulayman Sowe, Ioannis Stamelos, and Lefteris Angelis, "Identifying Knowledge Brokers that yield Software Engineering Knowledge in OSS projects," *Information and Software Technology,* vol. 48, no. 11, pp. 1025–1033, 2006.

[4] Huzefa Kagdi, Michael L. Collard, and Jonathan I. Maletic, "A Survey and Taxonomy of approaches for Mining Software Repositories in the context of Software Evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 19, no. 2, pp. 77-131, 2007.

[5] Ahmed E. Hassan, "The road ahead for Mining Software Repositories," in *Frontiers of Software Maintenance*, Beijing, 2008, pp. 48 - 57.

[6] Hadi Hemmati et al., "The MSR Cookbook: Mining a Decade of Research," in *10th Working Conference on Mining Software Repositories,* San Francisco, CA, 2013, pp. 343-352.

[7] Tudor Girba, Adrian Kuhn, Mauricio Seeberger, and Stphane Ducasse, "How Developers Drive Software Evolution," in *Eighth International Workshop on Principles of Software Evolution*, 2005, pp. 113-122.

[8] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie van Deursen, "Communication in Open Source Software Development Mailing Lists," in *10th Working Conference on Mining Software Repositories*, San Francisco, CA, 2013, pp. 277-286.

[9] Megan Squire, "How the FLOSS Research Community Uses Email Archives," *International Journal of Open Source Software and Processes (IJOSSP),* vol. 4, no. 1, pp. 37-59, 2012.

[10] John Anvik, Lyndon Hiew, and Gail C. Murphy, "Coping with an Open Bug Repository," in *OOPSLA Workshop on Eclipse Technology eXchange*, San Diego, California, 2005, pp. 35-39.

[11] Pamela Bhattacharya and Iulian Neamtiu, "Fine-grained Incremental Learning and Multi-feature Tossing Graphs to Improve Bug Triaging," in *IEEE International Conference on Software Maintenance*, 2010, pp. 1-10.

[12] Pamela Bhattacharya, Iulian Neamtiu, and Christian R. Shelton, "Automated, Highly-accurate, Bug Assignment using Machine Learning and Tossing Graphs," *Journal of Systems and Software*, vol. 85, no. 10, pp. 2275-2292, 2012.

[13] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann, "Improving Bug Triage with Bug Tossing Graphs," in *7th Joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE '09)*, New York, NY, USA, 2009, pp. 111-120.

[14] John Anvic and Gail C. Murphy, "Reducing the Effort of Bug Report Triage: Recommenders for Development-Oriented Decisions," *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 3, pp. 10:1-10:35, August 2011.

[15] John Anvik, "Automating Bug Report Assignment," in *28thInternational Conference on Software Engineering (ICSE '06)*, Shanghai, China, 2006, pp. 937-940.

[16] Shadi Banitaan and Mamdouh Alenezi, "TRAM: An Approach for Assigning Bug Reports using their Metadata," in *International Conference on Communications and Information Technology (ICCIT 2013)*, Beirut, Lebanon, 2013, pp. 215-219.

[17] Song Wang, Wen Zhang, and Qing Wang, "FixerCache: Unsupervised Caching Active Developers for Diverse Bug Triage," in *8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Torino, Italy, 2014, pp. 25:1-25:10.

[18] Weiqin Zou, Yan Hu, Jifeng Xuan, and He Jiang, "Towards Training Set Reduction for Bug Triage," in *IEEE 35th Annual Computer Software and Applications Conference (COMPSAC)*, Munich, 2011, pp. 576 - 581.

[19] Davor Cubranic and Gail C. Murphy, "Automatic Bug Triage using Text Categorization," in *Sixteenth International Conference on Software Engineering & Knowledge Engineering*, 2004, pp. 92-97.

[20] Olga Baysal, Michael W. Godfrey, and Robin Cohen, "A Bug You Like: A Framework for Automated Assignment of Bugs," in *IEEE 17th International Conference on Program Comprehension*, Vancouver, BC, 2009, pp. 297 - 298.

[21] Wenjin Wu, Wen Zhang, Ye Yang, and Qing Wang, "DREX: Developer Recommendation with K-Nearest-Neighbor Search and Expertise Ranking," in *18th Asia Pacific Software Engineering Conference (APSEC)* , Ho Chi Minh, 2011 , pp. 389 - 396.

[22] John Anvik, Lyndon Hiew, and Gail C. Murphy, "Who should fix this bug?," in *28th International Conference on Software Engineering (ICSE '06),* Shanghai, China, 2006, pp. 361-370.

[23] Xin Xia, David Loy, Xinyu Wang, and Bo Zhou, "Accurate Developer Recommendation for Bug Resolution," in *20thWorking Conference on Reverse Engineering (WCRE),* Koblenz, 2013, pp. 72 - 81.

[24] Shenglong Tan, Shenghong Hu, and Lihong Chen, "A Framework of Bug Reporting System Based on Keywords Extraction and Auction Algorithm," in *Fifth Annual China Grid Conference (ChinaGrid),* Guangzhou, 2010, pp. 281 - 284.

[25] Amir Moin and Günter Neumann, "Assisting bug Triage in Large Open Source Projects Using Approximate String Matching," in *Seventh International Conference on Software Engineering Advances*, Wilmington, 2012, pp. 22-27.

[26] Ibrahim Aljarah, Shadi Banitaan, Sameer Abufardeh, Wei Jin, and Saeed Salem, "Selecting Discriminating Terms for Bug Assignment: A Formal Analysis," in *7th International Conference on Predictive Models in Software Engineering,* Banff, Alberta, Canada, 2011, pp. 12:1-12:7.

[27] Syed Nadeem Ahsan, Javed Ferzund, and Franz Wotawa, "Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine," in *Fourth International Conference on Software Engineering Advances (ICSEA '09),* 2009, pp. 216-221.

[28] Ahmed Tamrawi, Tung Thanh Nguyen, Jafar M. Al-Kofahi, and Tien N. Nguyen, "Fuzzy Set and Cache-based Approach for Bug Triaging," in *19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE '11),* Szeged, Hungary, 2011, pp. 365-375.

[29] Jin-woo Park, Mu-Woong Lee, Jinhan Kim, Seung-won Hwang, and Sunghun Kim, "COSTRIAGE: A Cost-Aware Triage Algorithm for Bug Reporting Systems," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, San Francisco, California, 2011, pp. 139-144.

[30] Hoda Naguib, Nitesh Narayan, Bernd Brugge, and Dina Helal, "Bug Report Assignee Recommendation using Activity Profiles," in *10th IEEE Working Conference on Mining Software Repositories (MSR)*, San Francisco, CA, 2013 , pp. 22 - 30.

[31] Katja Kevic, Sebastian C. Muller, Thomas Fritz, and Harald C. Gall, "Collaborative Bug Triaging using Textual Similarities and Change Set Analysis," in *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, San Francisco, CA, 2013, pp. 17 - 24.

[32] Tung Thanh Nguyen, Anh Tuan Nguyen, and Tien N. Nguyen, "Topic-based, Time-aware Bug Assignment," *SIGSOFT Software Engineering Notes*, vol. 39, no. 1, pp. 1-4, 2014.

[33] Gerald Bortis and André van der Hoek, "PorchLight: A Tag-based Approach to Bug Triaging," in *International Conference on Software Engineering*, San Francisco, CA, USA, 2013 , pp. 342-351.

[34] Xihao Xie, Wen Zhang, Ye Yang, and Qing Wang, "DRETOM: Developer Recommendation Based on Topic Models for Bug Resolution," in *8th International Conference on Predictive Models in Software Engineering*, Lund, Sweden, 2012, pp. 19-28.

[35] Ramin Shokripour, John Anvik, Zarinah M. Kasiruna, and Sima Zamani, "A Time-based Approach to Automatic Bug Report Assignment," *Journal of Systems and Software*, vol. 102, pp. 109–122, 2015.

[36] Kalyanasundaram Somasundaram and Gail C. Murphy, "Automatic Categorization of Bug Reports Using Latent Dirichlet Allocation," in *5th India Software Engineering Conference*, Kanpur, India, 2012, pp. 125-130.

[37] Ramin Shokripour, John Anvik, Zarinah M. Kasirun, and Sima Zamani, "Why so complicated? Simple Term Filtering and Weighting for Location-based Bug Report Assignment Recommendation," in *10th IEEE Working Conference on Mining Software Repositories (MSR)*, San Francisco, CA, 2013, pp. 2-11.

[38] Huzefa Kagdi, Malcom Gethers, Denys Poshyvanyk, and Maen Hammad, "Assigning Change Requests to Software Developers," *Journal of Software: Evolution and Process*, vol. 24, no. 1, pp. 3-33, 2012.

[39] Dominique Matter, Adrian Kuhn, and Oscar Nierstrasz, "Assigning Bug Reports using a Vocabulary-Based Expertise Model of Developers," in *6th IEEE International Working Conference on Mining Software Repositories*, Vancouver, BC, 2009, pp. 131 - 140.

[40] Hao Hu, Hongyu Zhang, Jifeng Xuan, and Weigang Sun, "Effective Bug Triage Based on Historical Bug-Fix Information," in *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*, Naples, 2014 , pp. 122 - 132.

[41] Senthil Mani et al., "Bug Resolution Catalysts: Identifying Essential Non-committers from Bug Repositories," in *10th Working Conference on Mining Software Repositories*, San Francisco, CA, 2013, pp. 193-202.

[42] Jifeng Xuan, He Jiang, Zhilei Ren, and Weiqin Zou, "Developer Prioritization in Bug Repositories," in *34th International Conference on Software Engineering (ICSE)*, Zurich, 2012, pp. 25 - 35.

[43] Liguo Chen, Xiaobo Wang, and Chao Liu, "An Approach to Improving Bug Assignment with Bug Tossing Graphs and Bug Similarities," *Journal of Software*, vol. 6, no. 3, pp. 421-427, 2011.

[44] Wen Zhang, Song Wang, Ye Yang, and Qing Wang, "Heterogeneous Network Analysis of Developer Contribution in Bug Repositories," in *International Conference on Cloud and Service Computing (CSC)*, Beijing, 2013, pp. 98 - 105.

[45] Song Wang, Wen Zhang, Ye Yang, and Qing Wang, "DevNet: Exploring Developer Collaboration in Heterogeneous Networks of Bug Repositories," in *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, Baltimore, MD, 2013, pp. 193 - 202.

[46] Tao Zhang and Byungjeong Lee, "A Hybrid Bug Triage Algorithm for Developer Recommendation," in *28th Annual ACM Symposium on Applied Computing*, Coimbra, Portugal, 2013, pp. 1088-1094.

[47] Geunseok Yang, Tao Zhang, and Byungjeong Lee, "Utilizing a Multi-Developer Network-based Developer Recommendation Algorithm to Fix Bugs Effectively," in *29th Annual ACM Symposium on Applied Computing (SAC '14).*, Gyeongju, Republic of Korea, 2014, pp. 1134-1139.

[48] Tao Zhang and Byungjeong Lee, "An Automated Bug Triage Approach: A Concept Profile and Social Network Based Developer Recommendation," *Intelligent Computing Technology,Lecture Notes in Computer Science*, vol. 7389 , pp. 505-512, 2012.

[49] Shadi Banitaan and Mamdouh Alenezi, "DECOBA: Utilizing Developers Communities in Bug Assignment," in *12th International Conference on Machine Learning and Applications (ICMLA)*, Miami, 2013, pp. 66 - 71.

[50] Liguo Chen, Xiaobo Wang, and Chao Liu, "Improving Bug Assignment with Bug Tossing Graphs and Bug Similarities," in *International Conference on Biomedical Engineering and Computer Science (ICBECS), 2010*, Wuhan, 2010, pp. 1 - 5.

[51] A Medem, M-I Akodjenou, and R Teixeira, "TroubleMiner: Mining Network Trouble Tickets," in *IFIP/IEEE International Symposium on Integrated Network Management-Workshops*, New York, NY, 2009, pp. 113 - 119.

[52] Qihong Shao, Yi Chen, Shu Tao, Xifeng Yan, and Nikos Anerousis, "Efficient Ticket Routing by Resolution Sequence Mining," in *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* Las Vegas, Nevada, USA, 2008, pp. 605-613.

[53] Yi Chen, Shu Tao, Xifeng Yan, Nikos Anerousis, and Qihong Shao, "Assessing Expertise Awareness in Resolution Networks," in *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Odense, 2010, pp. 128 - 135.

[54] Gengxin Miao et al., "Understanding Task-Driven Information Flow in Collaborative Networks," in *21st International Conference on World Wide Web (WWW '12),* Lyon, France, 2012, pp. 849-858.

[55] Gengxin Miao et al., "Generative Models for Ticket Resolution in Expert Networks," in *16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* Washington, 2010, pp. 733-742.

[56] Gengxin Miao et al., "Reliable Ticket Routing in Expert Networks," in *Reliable Knowledge Discovery*. US: Springer, 2012, pp. 127-147.

[57] Peng Sun, Shu Tao, Xifeng Yan, Nikos Anerousis, and Yi Chen, "Content-aware Resolution Sequence Mining for Ticket Routing," in *8th International Conference on Business Process Management*, Hoboken, NJ, 2010, pp. 243-259.

[58] Christian Bird, Brendan Murphy, Nachi Nagappan, and Thomas Zimmermann, "Empirical Software Engineering at Microsoft Research," in *ACM Conference on Computer Supported Cooperative Work (CSCW)*, Hangzhou, China, 2011, pp. 143-150.

[59] Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, and Brendan Murphy, ""Not My Bug!" and Other Reasons for Software Bug Report Reassignments," in *ACM 2011 Conference on Computer Supported Cooperative Work*, Hangzhou, China, 2011, pp. 395 - 404.

[60] Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan, and Brendan Murphy, "Characterizing and Predicting Which Bugs Get Fixed: An Empirical Study of Microsoft Windows," in *32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, Cape Town, South Africa}, 2010, pp. 495-504.

[61] Jayalath Ekanayake, Jonas Tappolet, Harald C. Gall, and Abraham Bernstein, "Tracking Concept Drift of Software Projects Using Defect Prediction Quality," in *6th IEEE International Working Conference on Mining Software Repositories*, 2009, pp. 51-60.

[62] Grzegorz Chrupala, "Learning from Evolving Data Streams: Online Triage of Bug Reports," in *13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, 2012, pp. 613-622.

[63] Pamela Bhattacharya, Iulian Neamtiu, and Michalis Faloutsos, "Determining Developers' Expertise and Role: A Graph Hierarchy-Based Approach," in *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Victoria, BC, 2014, pp. 11-20.

[64] Poornalatha.G and Raghavendra S Prakash, "Web Page Prediction by Clustering and Integrated Distance Measure," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2012, pp. 1349-1354.

[65] Panagiotis Papapetrou, Vassilis Athitsos, Michalis Potamias, George Kollios, and Dimitrios Gunopulos, "Embedding-Based Subsequence Matching in Time-Series Databases," *ACM Transactions On Database Systems*, vol. 36, no. 3, pp. 17:1-17:39, 2011.

[66] Steven Burrows, S. M. M. Tahaghoghi, and Justin Zobel, "Efficient Plagiarism Detection for large Code Repositories," *Software: Practice and Experience*, vol. 37, no. 2, pp. 151–175, February 2007.

[67] Christian Kreibich and Jon Crowcroft, "Efficient Sequence Alignment of Network Traffic," in *6th ACM SIGCOMM Conference on Internet Measurement*, Rio de Janeriro, Brazil, 2006, pp. 307-312.

[68] R.W. Irving, "Plagiarism and Collusion Detection using the Smith-Waterman algorithm," Dept of Computing Science, University of Glasgow, University of Glasgow, Department of Computing Science, Glasgow, Technical Report 2004.

[69] Waqar Haque, Alex Aravind, and Bharath Reddy, "Pairwise Sequence Alignment Algorithms: A Survey," in *Conference on Information Science, Technology and Applications (ISTA '09)*, New York, NY, USA, 2009, pp. 96-103.

[70] Maxime Crochemore and Thierry Lecroq, "Alignments and Approximate String Matching," in *New Developments in Formal Languages and Applications, Studies in Computational Intelligence*.: Springer, 2008, vol. 113, pp. 59-93.

[71] G. Sudha Sadasivam and G. Baktavatchalam, "A Novel Approach to Multiple Sequence Alignment using Hadoop Data Grids," in *Workshop on Massive Data Analytics on the Cloud (MDAC '10)*, Raleigh, North Carolina, USA, 2010, pp. 2:1--2:7.

[72] Nimisha Singla and Deepak Garg, "String Matching Algorithms and their Applicability in various Applications," *International Journal of Soft Computing and Engineering*, vol. 1, no. 6, pp. 218-222, 2012.

[73] Taoxin Peng, Lin Li, and Jessie Kennedy, "A Comparison of Techniques for Name Matching," *GSTF Journal on Computing (JoC)* , vol. 2, no. 1, pp. 55-61, April 2012.

[74] Bin Cao, Ying Li, and Jianwei Yin, "Measuring Similarity between Graphs Based on the Levenshtein Distance," *Applied Mathematics & Information Sciences*, vol. 7, no. 1, pp. 169-175, 2013.

[75] E, Hassan Ahmed and Tao Xie, "Software Intelligence: The Future of Mining Software Engineering Data," in *FSE/SDP Workshop on the Future of Software Engineering Research*, 2010, pp. 161-166.

[76] Dominique Feillet, Pierre Dejax, and Michel Gendreau, "Traveling Salesman Problems with Profits," *Transportation Science*, vol. 39, no. 2, pp. 188-205, 2005.

[77] Andrea Raith and Matthias Ehrgott, "A Comparison of Solution Strategies for Biobjective Shortest Path Problems," *Journal Computers and Operations Research*, vol. 36, no. 4, pp. 1299-1331, 2009.

[78] Enrico Angelelli, Cristina Bazgan, M. Grazia Speranza, and Zsolt Tuza, "Complexity and Approximation for Traveling Salesman Problems with Profits," *Theoretical Computer Science*, vol. 531, pp. 54-65, 2014.

[79] Ashish Sureka, Atul Goyal, and Ayushi Rastogi, "Using Social Network Analysis for Mining Collaboration Data in a Defect Tracking System for Risk and Vulnerability Analysis," in *4th India Software Engineering Conference*, Thiruvananthapuram, Kerala, India, 2011, pp. 195-204.

[80] Mikkel Bundgaard, Troels C. Damgaard, and Federico Decara og Jacob W. Winther, "Ant Routing System a Routing algorithm based on Ant Algorithms Applied to a Simulated Network," Copenhagen, 2002.

[81] Elke Michlmayr, Ant Algorithms for Self-Organization in Social Networks, May 14, 2007.

[82] John E. Bella and Patrick R. McMullen, "Ant Colony Optimization Techniques for the Vehicle Routing Problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.

[83] Ehsan Sherkat, Maseud Rahgozar, and Masoud Asadpour, "Structural Link Prediction based on Ant Colony Approach in Social Networks," *Physica A: Statistical Mechanics and its Applications*, vol. 419, no. C, pp. 80-94, 2015.

[84] Yan Liu, QingXian Wang, Qiang Wang, Qing Yao, and Yao Liu, "Email Community Detection using Artificial Ant Colony Clustering," *Lecture Notes in Computer Science*, vol. 4537, pp. 287–298, 2007.

[85] Pablo Loyola, Pablo E. Roma, and Juan D. Vela´squez, "Predicting Web User Behavior using Learning-based Ant Colony Optimization," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 5, pp. 889-897, 2012.

[86] Kazuyuki Fujita, Akira Saito, Toshihiro Matsui, and Hiroshi Matsuo, "An Adaptive Ant-Based Routing Algorithm used Routing History in Dynamic Networks and Learning," in *4th Asia-Pacific Conference on Simulated Evolution*, 2002.

[87] Milos Kudelka et al., "Social and Swarm aspects of Co-authorship Network," *Logic Journal of the IGPL*, vol. 20, no. 3, pp. 634-643, 2012.

[88] Muhammad Aurangzeb Ahmad and Jaideep Srivastava, "An Ant Colony Optimization Approach to Expert Identification in Social Networks," in *Social Computing, Behavioral Modeling, and Prediction*.: Springer US, 2008, pp. 120-128.

[89] Filippo Lanubile, "Collaboration in Distributed Software Development," *International Summer Schools, ISSSE ,Lecture Notes in Computer Science*, vol. 5413, 2008.

[90] Ferenc Molnar. (2011) Link Prediction Analysis in the Wikipedia Collaboration Graph. [Online]. http://www.cs.rpi.edu/~magdon/courses/casp/projects/Molnar.pdf

[91] Nguyen Duc Anh, Daniela S. Cruzes, Reidar Conradi, and Claudia Ayala, "Empirical validation of Human Factors in predicting issue lead time in Open Source Projects," in *7th International Conference on Predictive Models in Software Engineering*, Banff, Alberta, Canada, 2011, pp. 13:1--13:10.

[92] Subhajit Datta, Vikrant S. Kaulgud, Vibhu Saujanya Sharma, and Nishant Kumar, "A Social Network based study of Software Team Dynamics," in *3rd India Software Engineering Conference*, Mysore, India, 2010, pp. 33-42.

[93] Verónica Rivera-Pelayo, Simone Braun, Uwe V. Riss, Hans Friedrich Witschel, and Bo Hu, "Building Expert Recommenders from Email-Based Personal Social Networks," *The Influence of Technology on Social Network Analysis and Mining, Lecture Notes in Social Networks*, vol. 6, pp. 129-156, 2013.

[94] Francisco Servant and James A. Jones, "WHOSEFAULT: Automatic Developer-to-Fault Assignment through Fault Localization," in *34th International Conference on Software Engineering (ICSE)*, Zurich, Switzerland, 2012, pp. 36-46.

[95] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy, "Dimensions and Metrics for Evaluating Recommendation Systems," in *Recommendation Systems in Software Engineering*.: Springer Berlin Heidelberg, 2014, pp. 245-273.

[96] Ayushi Rastogi, Arpit Gupta, and Ashish Sureka, "Samiksha: Mining Issue Tracking System for Contribution and Performance Assessment," in *6th India Software Engineering Conference*, New Delhi,India, 2013, pp. 13-22.

[97] Tao Zhang, Geunseok Yang, Byungjeong Lee, and Ilhoon Shin, "Role Analysis-based Automatic Bug Triage Algorithm," Technical Report 2012.

[98] Raula Gaikovina Kula, Kyohei Fushida, Shinji Kawaguchi, and ajimu Iida, "Analysis of Bug Fixing Processes Using Program Slicing Metrics," in *Product-Focused Software Process Improvement, Lecture Notes in Computer Science*.: Springer Berlin Heidelberg, 2010, vol. 6156, pp. 32-46.

[99] Daniel Alencar da Costa, Uirá Kulesza, Eduardo Aranha, and Roberta Coelho, "Unveiling Developers Contributions Behind Code Commits: An Exploratory Study," in *29th Annual ACM Symposium on Applied Computing*, Gyeongju, Republic of Korea, 2014, pp. 1152-1157.

LIST OF PUBLICATIONS

INTERNATIONAL JOURNALS

[1] “Bug Triaging Based on Ant Systems”, *International Journal of Bio-inspired Computing,Inderscience Publishers*. (In Press). **Thomson Reuters-Science Citation Index Expanded (SCIE) - Impact Factor : 1.681**

[2] “Bi Objective Optimization for Bug Triage”, *Future Generation Computer Systems,Elsevier Science Publishers*. (Accepted for Publication).**Thomson Reuters-Science Citation Index Expanded (SCIE) - Impact Factor : 2.639**

[3] “Novel Metrics for Bug Triage”, *Journal of Software,* vol. 9, no.12, pp.3035-3040, Dec 2014. DOI:10.4304/jsw.9.12.3035-3040

[4] “Bug Triage in Open Source System- A Review”, *International Journal of Collaborative Enterprise*, *Inderscience Publishers*, vol.4, no.4, pp.299–319,2014.DOI: 10.1504/IJCENT.2014.067002

INTERNATIONAL CONFERENCES

[1] “A Brief Survey on Concept Drift”, *International Conference on Intelligent Computing, Communication and Devices, Bhubaneshwar, Springer book on Advances in Intelligent Systems and Computing*, vol. 308, pp. 293-302, 2015. DOI: 10.1007/978-81-322-2012-1_31

[2] “Effective Bug Triage – A Framework”, *International Conference on Intelligent Computing, Communication and Convergence, Bhubaneshwar,Elsevier Procedia Computer Science,* vol.48 , pp.114-120, 2015. doi:10.1016/j.procs.2015.04.159

PAPERS COMMUNICATED

[1] “A Quantitative Analysis of Actual Path Model for Toss Graphs”, Malaysian Journal of Computer Science.

VITAE

V.AKILA, the author of this thesis is currently working as Assistant Professor in the Department of Computer Science and Engineering at Pondicherry Engineering College, Puducherry, India. She was born in May 1975 at Neyveli. She received her B.E degree in Computer Science and Engineering from Bharathidasan University, Tiruchirappalli, India in the year 1996 and M.E degree in Computer Science and Engineering from St. Joseph College of Engineering, Chennai, affiliated to Anna University in 2006. Akila is the College Topper and Distinction Holder in the P.G course. She joined as Lecturer at Pondicherry Engineering College in the year 2007 and has fourteen years of teaching experience. Her areas of interest are Mining Software Repositoires, Social Network Analysis and Swarm Optimization.