Volume 35 Number 3 September 2011 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Higher Education, Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Regression Test Selection Techniques: A Survey

Swarnendu Biswas and Rajib Mall
Dept. of Computer Science and Engineering
IIT Kharagpur, India - 721302
E-mail: {swarnendu, rajib}@cse.iitkgp.ernet.in

Manoranjan Satpathy and Srihari Sukumaran
GM India Science Lab, Bangalore, India
E-mail: {manoranjan.satpathy, srihari.sukumaran}@gm.com

Overview paper

Keywords: software maintenance, regression testing, regression test selection, model-based testing, UML, software components, embedded programs

Received: April 12, 2010

Regression testing is an important and expensive activity that is undertaken every time a program is modified to ensure that the modifications do not introduce new bugs into previously validated code. An important research problem, in this context, is the selection of a relevant subset of test cases from the initial test suite that would minimize both the regression testing time and effort without sacrificing the thoroughness of regression testing. Researchers have proposed a number of regression test selection techniques for different programming paradigms such as procedural, object-oriented, component-based, database, aspect, and web applications. In this paper, we review the important regression test selection techniques proposed for various categories of programs and identify the emerging trends.

Povzetek: Podan je pregled tehnik izbora testov za regresijsko testiranje programov.

1 Introduction

Software maintenance activities, on an average, account for as much as two-thirds of the overall software life cycle costs [75]. Maintenance of a software product is frequently necessitated to fix defects, to add, enhance or adapt existing functionalities, or to port it to different environments. Whenever an application program is modified for carrying out any maintenance activity, *resolution* test cases are designed and executed to check that the modified parts of the code work properly. *Regression* testing (also referred to as *program revalidation*) is carried out to ensure that no new errors (called regression errors) have been introduced into previously validated code (i.e., the unmodified parts of the program) [55]. Although regression testing is usually associated with system testing after a code change, regression testing can be carried out at either unit, integration or system testing levels. The sequence of activities that take place during the maintenance phase after the release of a software is shown in Figure 1. The figure shows that after a software is released, the failure reports and the change requests for the software are compiled, and the software is modified to make necessary changes. Resolution tests are carried out to verify the directly modified parts of the code, while regression test cases are carried out to test the unchanged parts of the code that may be affected by the code change. After the testing is complete, the new version of the software is released, which then undergoes a similar cycle.

Regression testing is acknowledged to be an expensive activity. It consumes large amounts of time as well as effort, and often accounts for almost half of the software maintenance costs [55, 49]. The extents to which time and effort are being spent on regression testing are exemplified by a study [22] that reports that it took 1000 machine-hours to execute approximately 30,000 functional test cases for a software product. It is also important to note that hundreds of man-hours are spent by test engineers to oversee the regression testing process; that is to set up test runs, monitor test execution, analyze results, and maintain testing resources, etc [22]. Minimization of regression test effort is, therefore, an issue of considerable practical importance, and has the potential to substantially reduce software maintenance costs.

Regression test selection (RTS) techniques select a subset of valid test cases from an initial test suite (T) to test that the affected but unmodified parts of a program continue to work correctly. Use of an effective *regression test selection* technique can help to reduce the testing costs in environments in which a program undergoes frequent modifications. Regression test selection essentially consists of two major activities:

– Identification of the affected parts - This involves

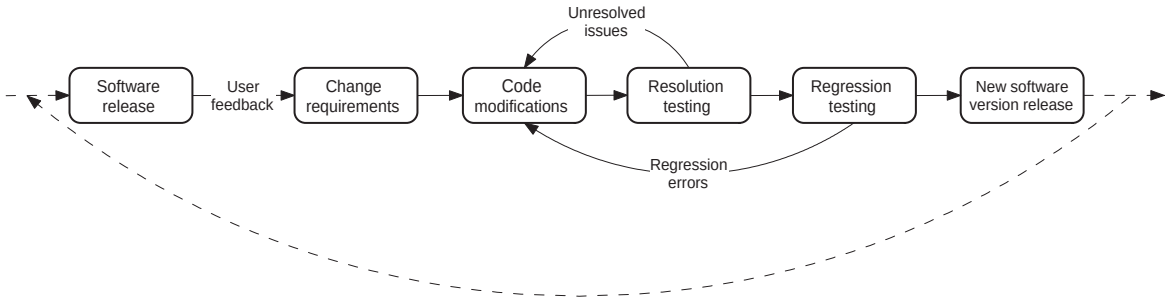


Figure 1: Activities that take place during software maintenance and regression testing.

identification of the unmodified parts of the program that are affected by the modifications.

– Test case selection - This involves identification of a subset of test cases from the initial test suite T which can *effectively* test the unmodified parts of the program. The aim is to be able to select the subset of test cases from the initial test suite that has the potential to detect errors induced on account of the changes.

Rothermel and Harrold [78] have formally defined the regression test selection problem as follows: *Let P be an application program and P' be a modified version of P. Let T be the test suite developed initially for testing P. An RTS technique aims to select a subset of test cases $T' \subseteq T$ to be executed on P', such that every error detected when P' is executed with T is also detected when P' is executed with T'.*

Leung and White [57] have observed that the use of an RTS technique can reduce the cost of regression testing compared to the *retest-all* approach, which involves running the entire test suite T to revalidate a modified program P', only if the cost of selecting a reduced subset of test cases to be run on P' is less than the cost of running the tests that the RTS technique omits. The *retest-all* approach is considered impractical on account of cost, resource and delivery schedule constraints that projects are frequently subjected to. Another approach is to *randomly* select test cases from T to carry out regression testing. However, random selection of test cases may fail to expose many regression errors. RTS techniques aim to overcome the drawbacks associated with the retest-all approach and in random selection of test cases by precisely selecting only those test cases that test the unmodified but affected parts of the program.

Though substantial research results on RTS have been reported in the literature, several studies [35, 36] show that very few software industries deploy systematic test selection strategies or automation support during regression testing. The approaches that are most often used in the industry for identification of relevant regression test cases are either based on expert judgment, or based on some form of manual program analysis. However, selection of test cases based on expert judgment tends to become ineffective and unreliable for large software products. Even for moderately complex systems, it is usually extremely difficult to manually identify test cases that are relevant to a change. This approach often leads to a large number of test cases being selected and rerun even for small changes to the original program, leading to unnecessarily high regression testing costs. What is probably more disconcerting is the fact that many test cases which could have potentially detected regression errors could be overlooked during manual selection. Another problem that surfaces during regression testing stems from the fact that testers (either from the same organization or from third-party companies) are usually supplied with only the functional description of the software, and therefore lack adequate knowledge about the code to precisely select only those test cases that are relevant to a modification [74].

A large number of RTS techniques have been reported for procedural [5, 7, 10, 37, 43, 44, 54, 56, 58, 80] and object-oriented programs [4, 14, 41, 73, 82], each aimed at leveraging certain optimization options. These techniques trade-off differently with regards to the cost of selection and execution of test cases and fault-detection effectiveness. In the recent past, the problem of RTS has actively been investigated and new approaches have emerged to keep pace with the newer programming paradigms. During the last decade, there has been a proliferation in the use of different programming paradigms such as component-based development, aspect-oriented programming, embedded and web applications, etc. It is, therefore, not surprising that a number of RTS techniques have been proposed for component-based [31, 66, 67, 72, 115, 116, 117], aspect programs [114, 109], web applications [86, 93, 61, 110, 85], etc.

RTS techniques have been reviewed by several authors [79, 6, 8, 34, 25, 24, 112]. In [79], Rothermel and Harrold have proposed a set of metrics to evaluate the effectiveness of different RTS techniques. Baradhi and Mansour [6], Bible et al. [8], and Graves et al. [34] have performed experimental studies on the performance and effectiveness of different RTS techniques proposed for procedural programs. Based on these studies, it is difficult to choose any technique as the best because these empirical studies have been performed on different categories of programs and also under different conditions [25]. This lead Engström et al. to perform a qualitative study [25, 24] of

the nature of the empirical data considered. The studies reported in [25, 24] are based on the similarities of the different RTS techniques and the quality of the empirical data used. Engström et al. [25] observe that it is very difficult to come up with an RTS technique which is generic enough (i.e., can be applied to different classes of applications) and is superior to all other techniques. The survey carried out by Engström et al. considers techniques which have been published before 2006. Therefore, their survey does not include many RTS techniques proposed after 2006 [114, 109, 18, 61, 86, 93, 85, 65, 31], and also does not include a few RTS techniques which were proposed before 2006 [10, 110, 107]. Moreover, their study does not include a detailed discussion about the merits and demerits of each technique.

In this paper, we present a detailed review of the RTS techniques proposed for different programming paradigms such as procedural, object-oriented, component-based, database, aspect and web software. Since a large number of RTS techniques have been proposed in the literature, we have limited our study to only the more prominent classes of RTS techniques. The techniques we have reviewed have been chosen based on their prominence determined by the number of citations and their frequency of referrals in other related studies. Our sources of information are existing reviews on RTS techniques [79, 6, 8, 25, 24, 34, 112], the citation index of the papers that we studied, and the online digital libraries, such as IEEE Xplore, ACM Digital Library, ScienceDirect, etc. The keywords that we used for our search on the online digital libraries include *regression testing*, *regression test selection*, *test selection*, etc. As an aid to understanding, and to keep the size of the review manageable, we have classified different RTS techniques together into relevant classes based on the motivation and similarity of the proposed approaches. We present a brief discussion on the working of each class of techniques, and discuss the merits and demerits of each. We also discuss issues that arise while designing RTS techniques for embedded programs, and identify the emerging trends in regression testing.

This paper is organized as follows: Section 2 presents basic concepts related to regression testing and which have been used in the rest of this paper. In Section 3, we discuss and compare various RTS approaches proposed for procedural programs. Subsequently, we discuss RTS techniques for object-oriented, component-based, database, web and AspectJ programs in Sections 4, 5, 6, 7, and 8 respectively. We discuss techniques for RTS of embedded software in Section 9. We discuss RTS techniques proposed for .Net and BPEL programs in Section 10. We discuss future research directions in regression testing and finally conclude the paper in Section 11.

2 Basic Concepts

In this section, we first discuss a few basic concepts that are extensively used in the context of regression testing. We then discuss some popular intermediate representations which are used for program model-based RTS.

For notational convenience, in the rest of the paper we denote the original and the modified programs by P and P' respectively. The initial regression test suite is denoted by T, and a test case in T is denoted by t.

2.1 Concepts Related to Regression Testing

In this section, we discuss a few important notations and concepts relevant to regression testing.

Obsolete, Retestable and Redundant Test Cases: According to Leung and White [55], test cases in the initial test suite can be classified as obsolete, retestable and redundant (or reusable) test cases. Obsolete test cases are no more valid for the modified program. Retestable test cases are those test cases that execute the modified and the affected parts of the program and need to be rerun during regression testing. Redundant test cases execute only the unaffected parts of the program. Hence, although these are valid test cases (i.e., not obsolete), they can be omitted from the regression test suite without compromising the quality of testing.

Execution Trace of a Test Case The execution trace of a test case t on a program P (denoted by $ET(P(t))$) is defined as the sequence of statements in P that are executed when P is executed with t [80]. The execution trace information for P can be generated by appropriately instrumenting the source code.

Fault-revealing Test Cases: A test case $t \in T$ is said to be *fault-revealing* for a program P, iff it can potentially cause P to fail by producing incorrect outputs for P [79].

Modification-revealing Test Cases: A test case $t \in T$ is considered to be *modification-revealing* for P and P', iff it produces different outputs for P and P' [79].

Modification-traversing Test Cases: A test case $t \in T$ is *modification-traversing* for P and P', iff the execution traces of t on P and P' are different [79]. In other words, a test case t is said to be modification-traversing if it executes the modified regions of code in P'. For a given original program and its modified version, the set of modification-traversing test cases is a super-set of the set of the modification-revealing test cases.

Inclusive, Precise and Safe Regression Test Cases: Inclusiveness measures the extent to which an RTS technique

selects modification-revealing tests from the initial regression test suite T [79]. Let us consider an initial test suite T containing n modification-revealing test cases. If an RTS technique M selects m of these test case, the inclusiveness of the RTS technique M with respect to P, P' and T is expressed as $(m/n) * 100$ [79].

A *safe* RTS technique selects all those test cases from the initial test suite that are modification-revealing [79]. Therefore, an RTS technique is said to be safe, iff it is 100% inclusive. Regression test cases that are relevant to a change but are not selected by an RTS technique are instances of *false negatives*. Therefore, an RTS technique is safe if the test suite selected by it has no false negatives [18].

Precision measures the extent to which an RTS algorithm ignores test cases that are non-modification-revealing [79]. Test cases that are selected by a technique but are not relevant are false positives. An RTS technique is, therefore, precise iff it there are no false positives among the selected test cases [18].

2.2 Regression Test Suite Minimization and Prioritization

Regression test suite minimization (TSM) techniques [40, 62, 64] aim to reduce the size of the regression test suite by eliminating redundant test cases such that the coverage achieved by the minimized test suite is same as the initial test suite. Different studies published in the literature [83, 106, 62] report conflicting results on the impact of TSM techniques on the fault-detection capabilities of the reduced test suites. Lin et al. have observed [62] that the TSM problem is *NP-complete*, since the minimum set-covering problem [20] can be reduced to the TSM problem in polynomial time.

Regression test case prioritization (TCP) techniques [23, 99, 84] order test cases such that test cases that have a higher fault-detection capability are assigned a higher priority and can gainfully be taken up for execution earlier. TCP approaches usually aim to improve the rate of fault detection by the ordered test suite [23, 84]. The main advantage of ordering test cases is that bugs are detected and can be reported to the development team early so that they can get started with fixing the bugs [84]. Also TCP techniques provide testers with the choice of executing only a certain number of higher priority test cases to meet the given time or cost considerations. This is advantageous especially in case of unpredicted interruptions to testing activities on account of delivery, resource or budget constraints.

Several TSM and TCP approaches have been proposed in recent years, and have emerged as active areas of research by themselves. However, our current work focuses only on RTS techniques. More detailed information about TSM and TCP approaches can be found in [22, 23, 112].

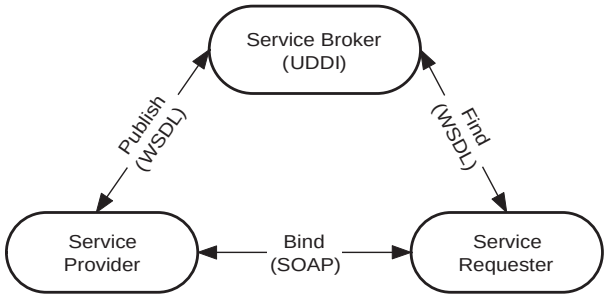


Figure 2: Web service architecture.

2.3 Few Other Relevant Concepts

In the following, we briefly discuss few other concepts that are relevant to this survey.

Program Slicing: Program slicing is a program analysis technique which was first introduced by Weiser [103] to aid in program debugging. A program slice is usually defined with respect to a slicing criterion. A slicing criterion SC is a pair $< p, V >$, where p is a program point of interest and V is a subset of the program's variables. A slice of a program P with respect to a slicing criterion SC is the set of all the statements of the program P that might affect the slicing criterion for every possible input to the program.

Since the publication of Weiser's seminal work, the concept of slicing has been extended and many slicing algorithms have been proposed in the literature for other areas of program analysis such as program understanding, compiler optimization, reverse engineering, etc. More detailed information regarding program slicing can be found in [108, 95].

Web Services: Web services are now being extensively used in application development across distributed and remote platforms, and are examples of service-oriented architecture (SOA) based development. SOA-based development has received a big boost with the advent of standardized web services. A *web service* can be defined as a software component which implements a logic and is designed to be inter-operable over a network providing platform-independence.

Figure 2 shows the typical architecture and the specifications of a web service [93, 13]. A service provider publishes services to a service broker. Service requesters find required services using a service broker and then bind to them. Platform independence is achieved through use of the following web specifications:

- Simple Object Access Protocol (SOAP) - SOAP is an XML-based protocol for information exchange over the network between web service and the users. The XML messages can be transferred using any application layer protocol such as Hypertext Transfer Protocol (HTTP). An advantage of SOAP messages is that

```
    int a, b, sum;
1.  read( a );
2.  read( b );
3.  sum = 0;
4.  while ( a < 8 ) {
5.      sum = sum + b;
6.      a = a + 1; }
7.  write( sum );
8.  sum = b;
9.  write( sum );
```

Figure 3: A sample program.

they can be exchanged between applications regardless of the development platform and the programming language being used.

- Web Service Description Language (WSDL) - WSDL is an XML-based language that is designed to provide an interface between a web service and its users.
- Universal Description Discovery and Integration (UDDI) - UDDI is an XML-based information registry where servers can publish their services. It allows users to locate any specific web services they might be interested in.

2.4 Graph Models for Procedural Programs

Graph models of programs have extensively been used in many applications such as program slicing [60, 89], impact analysis [52], reverse engineering [19], computation of program metrics [100], regression test selection [80, 73, 41, 7], etc. Analysis of graph models of programs is more efficient compared to textual analysis, and various types of relationships among program elements are also not explicit in the code. This has led to several representations such as Control Flow Graph (*CFG*) [3], Program Dependence Graph (*PDG*) [29] and System Dependence Graphs (*SDG*) [46] being proposed for procedural programs. In the following, we briefly discuss the important graph models proposed for procedural programs.

2.4.1 Flow Graph

A flow graph for a program P is a directed graph (N, E) where the program statements correspond to the set of nodes N in the flow graph, and the set of edges E represent the relationships among the program statements. However, the nodes in a flow graph can also correspond to basic blocks in a program. Typically it is assumed that there are two distinguished nodes called *start* with in-degree zero and *stop* with out-degree zero. There exists a path from $start$ to every other node in a flow graph, and similarly, there exists a path from every other node in the graph to $stop$.

2.4.2 Control Flow Graph

A control flow graph (*CFG*) [3] is a flow graph that represents the sequence in which the different statements in a program get executed. That is, it represents the flow of execution of control in the program. In fact, a *CFG* captures all the possible flows of execution of a program.

The *CFG* of the program P is the flow graph $G = (N, E)$ where an edge $(m, n) \in E$ indicates possible flow of control from node m to node n. Figure 4 represents the *CFG* of the program shown in Figure 3. Note that the existence of an edge (x, y) in a *CFG* does not necessarily mean that control *must* transfer from x to y during a program run.

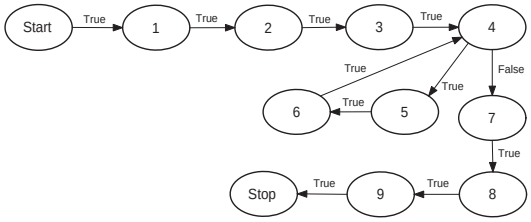


Figure 4: *CFG* for the example program shown in Figure 3.

2.4.3 Data Dependence Graph

Dependence graphs are used to represent potential dependencies between the elements of a program. In the following, we discuss data and control dependencies between program elements and their graph representations.

Data Dependence: Let G be the *CFG* of a program P. A node $n \in G$ is said to be data dependent on a node $m \in G$, if there exists a variable *var* of the program P such that the following hold:

1. The node m defines *var*,
2. The node n uses *var*,
3. There exists a directed path from m to n along which there is no intervening definition of *var*.

Consider the sample program shown in Figure 3 and its *CFG* shown in Figure 4. From the use of the variables sum and b in line 5, it is evident that node 5 is data dependent on nodes 2, 3 and 5. Similarly, node 8 is data dependent on only node 2. However, node 8 is not data dependent on either of the nodes 3 and 5.

Data Dependence Graph: The data dependence graph (*DDG*) of a program P is the graph $G_{DDG} = (N, E)$, where each node $n \in N$ represents a statement in the program P and if x and y are two nodes of G, then $(x, y) \in E$ iff y is data dependent on x.

2.4.4 Control Dependence Graph

The concept of control dependence [3] captures the dependency existing between two program elements when the execution of the second element is dependent on the outcome of the first.

Dominance: If x and y are two nodes in a flow graph, then x dominates y iff every path from $start$ to y passes through x. Similarly, y post-dominates x iff every path from x to $stop$ passes through y.

Let x and y be two nodes in a flow graph G. Node x is said to be the immediate post-dominator of node y iff x is a post-dominator of y, $x \neq y$ and every other post-dominator $z \neq x$ of y post-dominates x. The post-dominator tree of a flow graph G is the tree that consists of the nodes of G, has $stop$ as the root node, and has an edge (x, y) iff x is the immediate post-dominator of y.

Control Dependence: Let G be the *CFG* of a program P. Let x and y be two arbitrary nodes in G. A node y is said to be control dependent on another node x if the following hold:

1. There exists a directed path Q from x to y,
2. y post-dominates every z in Q (excluding x and y),
3. y does not post-dominate x.

The concept of control dependence implies that if y is control dependent on x, then x must have multiple successors in G. Conversely, if x has multiple successors, then at least one of its successors must be control dependent on it. Consider the program of Figure 3 and its *CFG* in Figure 4. Each of the nodes 5 and 6 is control dependent on node 4. Note that although node 4 has two successor nodes 5 and 7, only node 5 is control dependent on node 4.

Control Dependence Graph: The control dependence graph (*CDG*) of a program P is the graph $G_{CDG} = (N, E)$, where each node $n \in N$ represents a statement of the program P, and $(x, y) \in E$, iff y is control dependent on x.

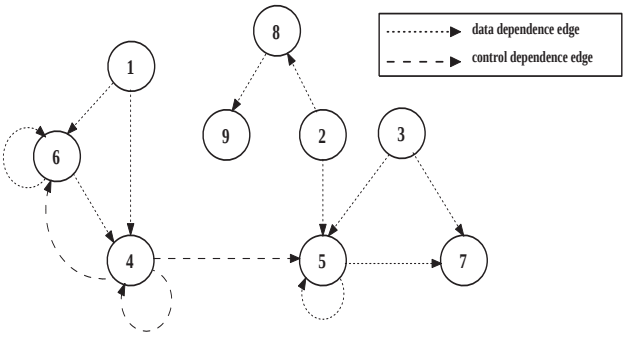


Figure 5: *PDG* of the program in Figure 3.

2.4.5 Program Dependence Graph

The program dependence graph (*PDG*) [29] for a program P explicitly represents both control and data dependencies in a single intermediate representation of P. The *PDG* of a program P is a directed graph $G_{PDG} = (N, E)$, where each node $n \in N$ represents a statement of the program P. A PDG contains both control dependence and data dependence edges. A control (or data) dependence edge (m, n) indicates that n is control (or data) dependent on m. Therefore, the *PDG* of a program P is the union of a pair of graphs: the data dependence graph of P and the control dependence graph of P. The *PDG* for the program in Figure 3 is shown in Figure 5.

2.4.6 System Dependence Graph

A major limitation of a *PDG* is that it can model only a single procedure and cannot handle inter-procedural calls. Horwitz et al. [46] enhanced the *PDG* representation to handle procedure calls and introduced the system dependence graph (*SDG*) representation which models the main program together with all the non-nested procedures.

```
int x = 0;

CE1  class A {
E2     void mA() {
S3       int a = 0;
S4       B *bptr = new B();
S5       cin>>a;
S6       try {
C7         bptr->mB(a); }
S8       catch(E1 &e1) {
S9         cout<<"Error E1"<<endl; }
S10      catch(...) {
S11        cout<<"Error"<<endl; }
S12      cout<<x<<endl; }
       };

CE13 class B {
E14    float mB( int y ) {
S15      try {
S16        if ( y < 0 )
S17          throw new E2();
S18        x = sqrt(y); }
S19      catch(E2 &e2) {
S20        cout<<"Error E2"<<endl;
S21        throw; }
S22      cout<<x<<endl; }
       };

E23  main(int argc, char *argv[]) {
S24    A *aptr = new A();
C25    aptr->mA();
       }
```

Figure 6: An example program.

An *SDG* is very similar to a *PDG*. In fact, the *PDG* of the main program is a subgraph of the *SDG*. In other words, for a program without procedure calls, the *PDG* and the *SDG* are identical. The technique for constructing an *SDG* consists of first constructing a *PDG* for every procedure, including the main procedure, and then adding auxiliary dependence edges which link together the various subgraphs while maintaining call-return discipline.

2.5 Graph Models for Object-Oriented Programs

The object-oriented paradigm is based on several important concepts such as encapsulation, inheritance, polymorphism, dynamic binding, etc. These concepts usually lead to complex relationships among program elements, and render the graph models proposed for procedural programs

inadequate for representing object-oriented programs [60]. Therefore, various models for object-oriented programs such as the Class Call Graph (*CCG*) [42], Inter-procedural Program Dependence Graph (*IPDG*) [77], Class Dependence Graph (*ClDG*) [77, 78], and Java Interclass Graph (*JIG*) [41] have been proposed. In the following, we briefly discuss the *ClDG* and *JIG* models.

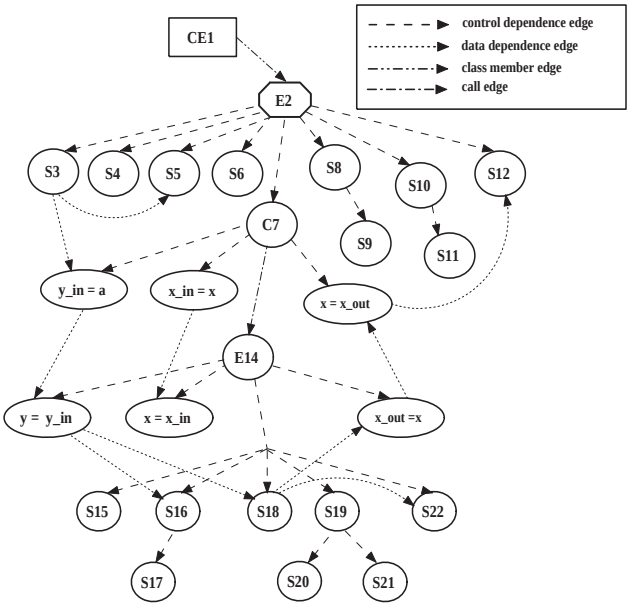


Figure 7: *ClDG* for class A of the program shown in Figure 6.

2.5.1 Class Dependence Graph

A *ClDG* [77, 78] is an extensively used model for intermediate representation of object-oriented programs. Each method in a *ClDG* is represented by its corresponding *PDG*. A class in a *ClDG* is denoted by a *class entry* node and the entry point for each method is represented by a *method entry* node. The *class entry* node is connected to each *method entry* node by a *class member* edge. A representative driver node (RDN) is added to the *ClDG* which summarizes the set of test driver routines used for class testing [77]. This RDN acts as the root of the *ClDG* for the whole program. Each entry node of a public method of the class is directly connected to the RDN by means of *driver edges*, thus implying that the driver routines can invoke the public methods of the class under test. For example, Figure 7 shows the *ClDG* constructed for class A of the program shown in Figure 6. The node labels in the *ClDG* correspond to the statement numbers in the program of Figure 6. The rectangular node labeled $CE1$ in Figure 7 represents the *class entry* vertex for class A. Node $E2$ represents a *method entry* node corresponding to the method `void A::mA()`. The edge $CE1 \rightarrow E2$ in Figure 7 is a *class member* edge. Figure 8 shows the *ClDG* for class B defined in Figure 6.

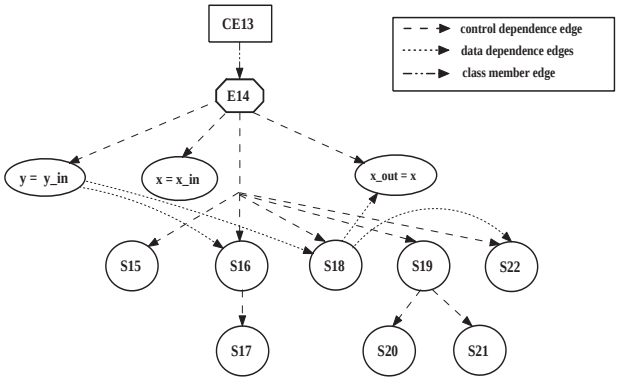


Figure 8: ClDG for class B of the program in Figure 6.

2.5.2 Java Interclass Graph

Intermediate representations such as *IPDG* and *ClDG* have been proposed in the context of C++ programs and do not satisfactorily model Java programs. Harrold et al. proposed an extended control flow model for Java programs called Java Interclass Graph (*JIG*) [41] that extends a *CFG* to capture the following features of a Java program:

- Variable and object type information - The variable or object type information is stored in a *JIG* node. The names of classes are represented using the full inheritance hierarchy which helps to easily detect any change to the inheritance tree for the class in the modified program.
- Internal and external methods of a class - Internal methods are represented in a *JIG* with an extended *CFG*. The extensions are: each call site is broken into a *call* and a *return* node. The *call* and *return* nodes are inter-connected with a *path* edge that represents the execution path through the called method.
  Since the source code is usually not available for externally-defined methods, these are represented in a *JIG* using collapsed *CFG*s.
- Calls to internal or external methods from internal methods - In a *JIG*, the *call* node is connected to the entry node of the called method with a *call* edge. There can only be one *call* edge if the method call is not polymorphic. For a polymorphic method call, the *call* node is connected to the entry node of each method that can be bound to the call. The class hierarchy analysis technique [21] can be used to identify all possible virtual call bindings.
  We illustrate the representation of method calls from internal methods in a *JIG* with the help of an example reported in [41]. Figure 9 shows a code snippet with calls to methods `foo()` and `m()`. In the program, class B extends class A (external to the program) and overrides the method `m()`. Class C extends class B and also overrides method `m()`. For polymorphic calls in a *JIG*, there exists an edge to all the methods for possible bindings. In the program, call to `A.foo()` from within function `bar()` represents a

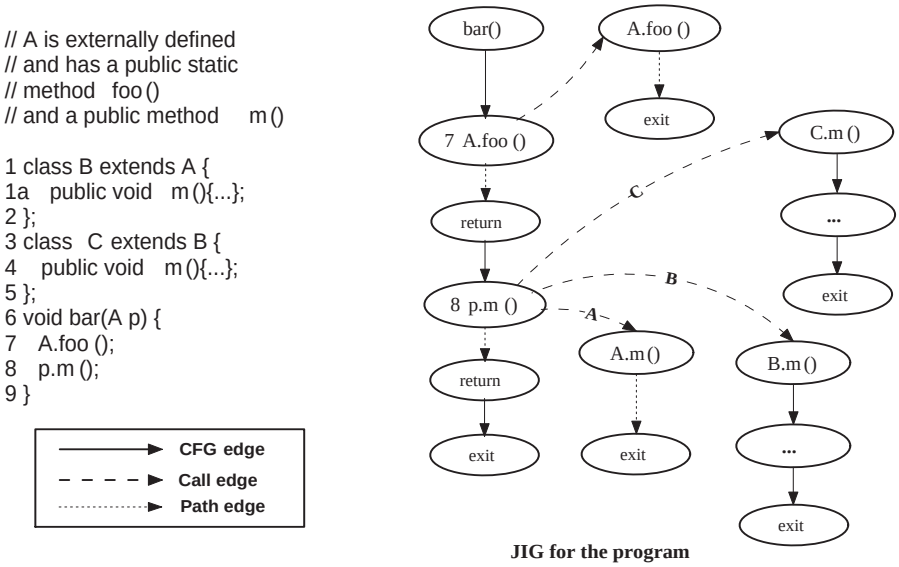


Figure 9: An example of method calls from internal methods in a *JIG*.

static binding. Therefore, in Figure 9, there exists only a single *call* edge between the nodes `7 A.foo()` and `A.foo()`. The method call `p.m()` is polymorphic, and there can be three possible bindings, one each for class A, B, and C. This is represented in the *JIG* by the three out edges from the node `8 p.m()`. Each outgoing edge connects to a possible method call to which it can bind during run-time.

– Calls to internal methods from external methods - There can be calls from externally-defined methods to internally-defined methods in Java due to inheritance and polymorphism. The external code is represented in a *JIG* as a node labeled *ECN* where *ECN* stands for *external code node*. For each internal class that is accessed from an external class, there is an outgoing edge from the *ECN* node to the *class entry* node of that internal class.
– Exception handling - A *JIG* represents a `try` statement with a *try* node. The code within the `try` block is represented as a *CFG*, and is connected with a control flow edge with the *try* node. Each `catch` statement is represented using a *catch* node, and the corresponding `catch` block is modeled using a *CFG*. The *catch* node is connected with the *CFG* using a control flow edge. The *try* node is connected to the *catch* node of the first `catch` block using a path edge labeled *exception*. A *finally* node and a *CFG* are used to represent the `finally` block of the `try` statement. Uncaught exceptions are modeled as *exceptional exit* nodes.

3 RTS Techniques for Procedural Programs

RTS techniques were first studied in the context of procedural programs [55, 56]. RTS for procedural programs is, therefore, an extensively researched topic, and many techniques have been proposed over the years [17, 56, 58, 37, 43, 92, 54, 76, 5, 80, 97, 98, 7, 10]. These techniques select relevant regression test cases using either control flow, data or control dependence analysis, or by textual analysis of the original and the modified programs. Depending on the type of the program analysis technique used and to aid in understanding, we have grouped the different RTS techniques into the following major classes:

1. Dataflow analysis-based techniques [43, 92, 44, 37]
2. Slicing-based techniques [7, 10, 2]
3. Firewall-based techniques [56, 58]
4. Differencing-based approaches [97, 98, 17]
5. Control flow analysis-based techniques [54, 80, 5]

In the following, we briefly discuss these different categories of RTS techniques and compare their effectiveness. We base our comparisons on the set of metrics introduced by Rothermel and Harrold [79]: safety, precision, efficiency, and generality. Rothermel and Harrold have presented a comprehensive survey of procedural RTS techniques in [79]. For the sake of completeness and continuity of the paper, we have included brief discussions on these techniques. We also discuss a few techniques [97, 98, 5] which were published after their work.

3.1 Dataflow Analysis-Based Techniques

In this subsection, we review RTS techniques [43, 92, 44, 37] based on dataflow analysis.

Dataflow analysis-based RTS techniques explicitly detect definition-use pairs for variables that are affected by program modifications, and select test cases that exercise the paths from the definition of modified variables to their uses. The use of a variable is further distinguished into computation uses (c-uses) and predicate uses (p-uses). A c-use occurs for a variable if it is used in computations, and a p-use occurs when it is used in a conditional statement. A c-use may have an indirect effect on the control flow of the program, while a p-use may either directly affect the flow of control or may also indirectly affect some other program statements.

Harrold and Soffa [43] have proposed a dataflow coverage-based RTS technique that can be applied to analyze changes across multiple procedures. Their approach involves processing the dataflow information incrementally, i.e., process a single change, select test cases for that change, and update the dataflow information and test coverage information. The same process is repeated for all the changes one by one. In their approach, P is represented by a *CFG*, in which the nodes represent basic blocks. This reduces the size of the flow graph and makes graph analysis more efficient as compared to representing the individual program statements as nodes. Additional nodes are introduced in the flow graph to model global variables, function parameters, and return values of functions. Modifications to P usually result in changes to the basic blocks or the control flow structure of the program. The information in each node of the flow graph is extended to include the associated dataflow information for variables present in the node. For each variable definition in a node n, the node numbers of all the c-uses of the variable in the flow graph are stored in node n. The block numbers for all the p-uses of the variable are also stored in n. The information about the paths traversed when P is executed on each $t \in T$ is used to select test cases which exercise the modified def-use pairs for any variable, and are selected for retesting P'.

Dataflow-based RTS techniques reported in [43, 92, 44] usually carry out analysis either by processing the changes one by one and then incrementally updating the dataflow information for P', or compute the full dataflow information for P and P' and compare the differences between def-use pairs. Both these approaches require saving the dataflow information across testing sessions, or recompute them at the beginning of each testing session. The program slicing-based RTS technique proposed by Gupta et al. [37] is based on inter-procedural slicing which does not require saving or recomputing the dataflow information across testing sessions. The technique uses the concepts of backward and forward slices to determine the affected def-use pairs that must be retested. The program to be regression tested is sliced to select test cases that execute the affected def-use pairs.

3.1.1 Critical Evaluation

The techniques reported in [37, 43, 44] are based on computing dataflows in a program and are not able to determine the effect of program modifications that do not cause changes to the dataflow information [112]. The techniques also do not consider control dependencies among program elements for selecting regression test cases. As a result, these techniques are unsafe. Dataflow techniques are also imprecise because the presence of an affected definition or use in a new block of code does not guarantee that all test cases which execute the block will execute the affected code [79]. Examples illustrating the unsafe and imprecise nature of dataflow-based techniques are available in [79].

3.1.2 Slicing-Based Techniques

Agrawal et al. [2] have proposed a set of program slicing-based RTS techniques. The aim of these techniques is to select those test cases which can produce different outputs when executed with the modified program version P'. The authors define a slice with respect to a test case t as the set of program statements which are executed when P is executed with t. The authors have proposed four slicing techniques [2]: *execution* slice, *dynamic* slice, *relevant* slice, and *approximate relevant* slice. The RTS techniques proposed in [2] select a test case t for regression testing only if the slice of t computed using any one of the four approaches contains a statement modified in P'.

A PDG-based slicing approach for procedural programs was proposed by Bates and Horwitz [7]. However, the *PDG*-based slicing technique did not support inter-procedural regression testing. In [10], Binkley proposed an inter-procedural RTS technique based on slicing *SDG* models of P and P'. Two components are said to have equivalent execution patterns, iff they are executed the same number of times on any given input [10]. The concept of *common* execution patterns [10] has been introduced as an inter-procedural extension of the *equivalent* execution patterns proposed in [7]. Code elements are said to have a common execution pattern if they have the same equivalent execution pattern during some call to procedures. Common execution patterns capture the semantic differences among code elements [10]. The semantic differences between P and P' are determined by comparing the expanded version (i.e., with every function call expanded in place) of the two programs. The expanded versions of the two programs are analyzed to find out affected program elements which need to be regression tested.

3.1.3 Critical Evaluation

The program slicing-based RTS techniques proposed by Agrawal et al. [2] are unsafe [112]. The techniques are however precise [6] because they omit test cases that do not produce a different output. This eliminates the possibility of selecting non-modification-revealing test cases.

According to the studies reported by Rothermel and Harrold [79], the *PDG* [7] and *SDG*-based [10] slicing techniques are not safe when the changes to the modified program involve deletion of statements. The techniques are also imprecise. However, the *SDG* slicing-based RTS technique can be applied to select test cases for both intra- and inter-procedural modifications.

3.2 Module Level Firewall-Based Techniques

The firewall-based approach, first proposed by Leung and White [56, 58], is based on analysis of data and control dependencies among modules in a procedural program. A *firewall* is defined as the set of all the modified modules in a program along with those modules which interact with the modified modules. A firewall is a conceptual boundary that helps in limiting the amount of retesting required by identifying and limiting testing to only those modules which are affected by a change. The firewall techniques use a *call graph* to represent the control flow structure of a program [56]. Module A is called an ancestor of module B, if there exists a path (a sequence of calls) in the call graph from module A to B, and module B is then called a descendant of module A. The direct ancestors and the direct descendants of the modified modules are also included during the construction of a firewall to account for all possible interactions with the modified modules. The test coverage information for P is used to select the subset of test cases from T which exercise the affected modules included in the firewall.

3.2.1 Critical Evaluation

The firewall technique is not safe as it does not select those test cases from outside the firewall that may execute the affected modules within the firewall [79]. The firewall techniques are imprecise because all test cases which execute the modules within the firewall do not necessarily execute the modified code within modules. However, the firewall techniques are efficient because the approaches consider only the modified modules and their relationships with other modules in the firewall, and hence limit the total amount of the source code that need to be analyzed. The firewall techniques handle RTS for inter-procedural program modifications but are not applicable for intra-procedural modifications [79].

3.3 Differencing-Based Techniques

In this subsection, we discuss RTS techniques [17, 97] that are based on analysis of the differences between the original and the modified programs.

3.3.1 Modified Code Entity-Based Technique

A modified code entity-based RTS technique was proposed by Chen et al. [17] for C programs. They have decomposed program elements into functional and non-functional code entities. A code entity is defined as either a directly executable unit of code such as a function or a statement, or a non-executable unit such as a global variable or a macro. The original program P is executed with each test case $t \in T$. The test coverage information is analyzed to determine the set of executable code entities that are exercised by each test case $t \in T$. For each function that is executed by a test case t, the transitive closure of the global variables, macros, etc. referenced by the function is computed. When the original program P is modified, all the code entities which were modified to create the revised program P' are identified. Test cases that exercise any of the modified entities are selected for regression testing P'.

3.3.2 Technique Based on Textual Differencing

Vokolos and Frankl [97, 98, 30] have proposed an RTS technique which is based on a textual differencing of the original and the modified programs (i.e., P and P'), rather than using any intermediate representation of the programs. A naive textual differencing of the programs will include trivial differences between the two versions, such as insertion of blank lines, comments etc. Therefore, their technique first converts a program to its *canonical* form [96, 97] before comparison. This conversion ensures that the original and the modified programs follow the same syntactic and formatting guidelines. The canonical version of P is instrumented and then executed to generate the test coverage information. The test coverage information identifies the basic blocks that are executed by each test case instead of the program statements. The canonical versions of P and P' are syntactically compared to find out modifications to the code. The test coverage information is then used to identify test cases which execute the affected parts of the code.

3.3.3 Critical Evaluation

The modified code entity technique is safe because it identifies all possible affected code entities, and selects regression test cases based on test coverage [8, 79]. The technique proposed in [97] is also safe because it identifies all the basic blocks that are affected due to modifications and selects regression test cases that execute those basic blocks. However, both the techniques are imprecise. For example, if a function f is modified, the modified code entity technique selects all those test cases which execute f. But there might be tests which execute f without executing the modified code in f. The textual differencing technique can be highly imprecise when code changes are arbitrary since differentiation is based on only syntax and the test cases are selected based on coverage of basic blocks. The code entity technique is considered to be the most efficient and safe RTS technique for procedural programs [79], and its time complexity is bounded by the size of T and P. The time complexity of the textual differencing technique

is $O(|P|*|P'|*log|P|)$ which may not be scalable for large programs.

3.4 Control Flow Analysis-Based Techniques

A few RTS techniques [54, 80, 5] have been proposed which analyze control flow models of the input programs for selecting regression test cases. We briefly discuss these RTS techniques in the following.

3.4.1 Cluster Identification Technique

The main concept used in the cluster identification technique proposed by Laski and Szermer [54] is localization of program modifications into one or more areas of the code referred to as *clusters*. Clusters are defined as single-entry, single-exit parts of code that have been modified from one version of a program to the next. The cluster identification technique models programs P and P' as *CFG*s (denoted by G and G'). The nodes in G and G' which correspond to the modifications in the code are identified, and the set of all such identified nodes in G and G' are marked as clusters. A cluster identification-based technique uses control dependence information of the original and the modified procedures to compute the clusters in the two graphs.

Once the clusters have been identified in the *CFG*s, each cluster is then represented by a single node to form a *reduced CFG*. Analysis of the reduced flow graphs is based on the assumption that any complex program modification can be achieved by one of the following three operations: inserting a cluster into the code, deleting a cluster, or changing the functionality of a cluster. Test cases are classified into two categories: local to the clusters and global in the entire program. The former includes test cases which execute modified clusters, and the latter includes test cases which execute other areas of the program affected due to the modified clusters based on control dependencies. The test coverage information is then used to select regression test cases.

3.4.2 Graph Walk-Based Technique

Rothermel and Harrold have proposed an RTS technique based on traversal of *CFG*s of the original and the modified programs [80]. This technique [80] is more efficient as compared to the graph walk-based RTS approaches based on dependence graph models [76, 78] proposed by the same authors. The approach proposed in [80] involves constructing *CFG*s G and G' for programs P and P' respectively. The execution trace information for each test case t, $ET(P(t))$, is recorded. This is achieved by instrumenting P. In [80], a simultaneous depth-first traversal of the two *CFG*s G and G' is performed corresponding to each modified procedure in P and P'. The traversal is performed according to the execution trace for each test case in T. For each pair of nodes n and n' belonging to G and G' respectively, the technique finds out whether the program statements associated with the successors of n and n' along identically-labeled edges of G and G' are equivalent or not. If a pair of nodes n_1 and n'_1 is found such that the statements associated with n_1 and n'_1 are not identical, then the edges that lead to the non-identical nodes are identified as *dangerous* edges. Test cases which execute the set of identified dangerous edges are assumed to be modification-revealing. Therefore, a test case $t \in T$ is selected for retesting P' if $ET(P(t))$ contains node n_1.

3.4.3 DFA Model-Based Approach

Ball [5] has proposed a more precise RTS technique compared to [80] by modeling *CFG* G for a program P as a deterministic finite state automaton (DFA). A DFA M for a *CFG* G can be constructed such that the following conditions hold:

1. Each node v in G corresponds to two states v_1 and v_2 of M. The two states are connected by a transition $v_1 \rightarrow_{BB(v)} v_2$, where $BB(v)$ is the basic block associated with node v in G.
2. The set of edges in G are modeled as state transitions. Therefore, an edge $m \rightarrow n$ in G represents a state transition $m_2 \rightarrow n_1$ in M.

These two conditions ensure that the DFA M accepts the set of all possible complete paths in G.

Ball introduced an intersection graph model for a pair of *CFG*s G and G' corresponding to the original and modified programs. The intersection graph also has an interpretation in terms of a DFA. Ball's RTS technique is based on reachability of edges in the intersection graphs. The technique uses edge coverage criterion as the basis for RTS analysis.

3.4.4 Critical Evaluation

The RTS techniques proposed in [80, 5, 54] are safe. Among the three techniques, the cluster identification technique is comparatively more imprecise because the test cases are selected based on whether they execute a cluster rather than the actually affected statements. The time complexity of the cluster identification technique [54] is bounded by the time required to compute the control scope of decision statements and is dependent on the input program size [79]. The techniques proposed in [80, 5] are the two most precise procedural RTS techniques. However, Ball's DFA-based approach is computationally more expensive than [80].

Ball has proposed another RTS technique [5] which uses path coverage criterion and is still more precise than the edge-coverage criterion proposed in [5]. The higher precision is attributable to the fact that path coverage is stronger than an edge coverage criterion. This increase in precision is however accompanied by an increase in the computation effort. Additionally, it cannot analyze control flows across

Class of RTS Techniques	References	Key Features	Merits	Demerits
Dataflow analysis-based techniques	[37, 43, 44, 92]	Based on dataflow and structural coverage criteria	Can analyze both intra- and inter-procedural modifications provided the modifications alter some def-use relations	Low on safety, imprecise
Slicing-based techniques	[7, 10, 2]	Based on slicing of programs or dependence graph models	Can analyze both intra- and inter-procedural modifications	Low on safety, imprecise, computationally more expensive than dataflow techniques
Module level firewall-based techniques	[56, 58]	Based on analyzing dependencies among modules	Comparatively more efficient as analysis of source code is limited to only modified modules	Low on safety, and highly imprecise
Modified code entity-based technique	[17]	Level of granularity can be adapted	Safe, and most efficient procedural RTS technique	Highly imprecise
Textual differencing-based technique	[97, 98, 30]	Based on textual differencing of C programs	Safe, and comparatively easy to implement a prototype	Imprecise, and difficult to adapt to other languages, maybe inefficient for large programs
Graph walk-based technique	[80]	Based on analysis of control flow models	Safe and most precise procedural RTS technique	Less efficient than [17, 56, 58]

Table 1: A comparison of RTS techniques for procedural programs.

procedures and hence cannot be applied for RTS of inter-procedural code modifications.

An important difference between graph walk and slicing-based techniques is that the latter uses dependence relationships to analyze the source code and identify the affected regions in the source code. Regression test selection is performed by monitoring the execution of the sliced region of code on T. On the other hand, the graph walk techniques use comparison of graph models of the program to identify the modifications [76, 80].

Table 1 summarizes the merits and demerits of the procedural RTS techniques discussed in Section 3. In column 3, we highlight the key features of each class of techniques, and summarize the merits and demerits in columns 4 and 5.

4 RTS Techniques for Object-Oriented Programs

The object-oriented paradigm is founded on several important concepts such as encapsulation, inheritance, polymorphism, dynamic binding, etc. These concepts lead to complex relationships among various program elements, and make dependency analysis more difficult [104]. Moreover, in object-oriented development, reuse of existing libraries, class definitions, program executables (blackbox components), etc. are emphasized to facilitate faster development of applications. These libraries and components frequently undergo independent modifications to fix bugs and enhance functionalities. This creates a new dimension in regression testing of object-oriented programs that use these third-party components or libraries, since the source code for such libraries are often not available. These features, therefore, raise challenging questions on how to effectively select regression test cases that are safe for such programs [9, 68].

The reported RTS techniques for object-oriented programs can broadly be classified into the following three major categories:

1. Firewall-based techniques [53, 47, 1, 48]
   (a) Class firewall technique [53]
   (b) Method level firewall technique [48]
2. Program model-based techniques [77, 82, 41, 73]
3. Design model-based techniques [4, 27, 69, 33, 14]

In the following, we briefly review the different classes of RTS techniques that have been proposed for object-oriented programs.

4.1 Firewall-Based Techniques

Firewall-based RTS techniques for object-oriented programs have been proposed by Kung et al. [53], Hsia et al. [47], Abdullah and White [1] and Jang et al. [48]. These techniques are based on the concept of a firewall defined originally by Leung and White [58] for procedural programs. The firewall techniques aim to identify the affected classes for the modified version of the software. A firewall can be defined as the set of all the affected classes that need to be retested. These techniques select all test cases which exercise at least one class from within the firewall.

4.1.1 Kung's Class Firewall Technique

Kung et al. [53] have proposed a firewall-based RTS technique for C++ programs. They have proposed the following three models to represent dependencies between various elements of a C++ program: Object Relation Diagram (*ORD*), Block Branch Diagram (*BBD*), and Object State Diagram (*OSD*). An *ORD* is a digraph that represents inheritance, aggregation and association relations, and captures the static dependencies between classes. An edge in

an *ORD* is annotated with the type of relationship (inheritance, association, aggregation) that exists between the end nodes associated with that edge. A *BBD* represents the interface and the control structure of a method of a class, and the relationship of a class with the other classes in the program. An *OSD* is designed to capture the dynamic behavior of a class.

Changes to data items, methods and class definitions of the original program (if any) are identified by analyzing the three models corresponding to P and P'. The technique of Kung et al. [53] instruments P to collect information about which classes are exercised by which test cases. A class is potentially affected by a change to another class if it is either directly or indirectly related to the changed class through inheritance, aggregation or association relationships. The firewall for a class C is computed as the set of classes that are directly or transitively dependent on C (by virtue of relations such as inheritance, aggregation or association) as described in an *ORD*. When a class C is modified, the technique selects all the test cases that exercise one or more classes within the firewall for C.

Figure 10 shows an example *ORD* (along with test cases) adapted from [90]. In the figure, a solid arrow from one class to another indicates that the two classes are related by inheritance, aggregation or association relationships. The boundary (denoted by the dashed line) around the classes A, B, C and D depicts the firewall computed for class D. Whenever the class D is modified, classes A, B and C also need to be regression tested for they belong to the set of classes which constitute the firewall for class D. In Figure 10, a solid line from a test case to a class indicates that the test case is used to test the class (e.g., test case $TC1$ is used to validate classes D and F). Thus according to the firewall technique, only test cases $TC1$ and $TC2$ should be executed again after class D is modified since they exercise those classes within the firewall for D.

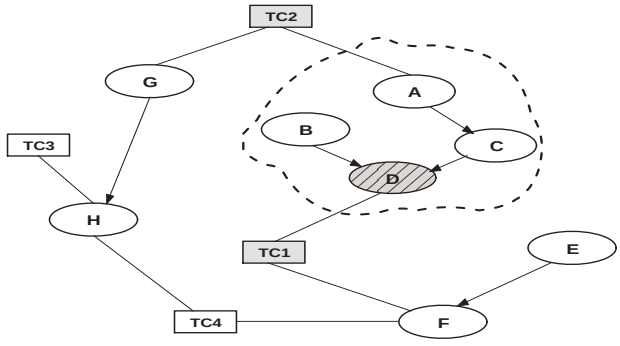


Figure 10: An example ORD and the firewall for class D.

4.1.2 Method-level Firewall Technique

Jang et al. [48] have proposed a change impact analysis approach to select regression test cases for C++ programs. While a class and a statement are considered as the units of testing in [53] and [77], the technique reported in [48] considers a method as the unit of retesting and aims to identify all affected methods. The authors have identified certain common types of modifications that are possible for a C++ program, and a method-level firewall is constructed for each modification to identify the impact of the changes.

4.1.3 Critical Evaluation

Firewall techniques are not safe because these techniques do not select test cases which may execute the affected modules from outside the firewall. These techniques are also imprecise since all test cases that execute a class in the firewall do not necessarily execute the affected parts of the code. For example, suppose that a class C is modified. Let another class D contain two methods `D::foo()` and `D::bar()`, of which the method `D::foo()` invokes the services provided by class C. Then, by the approach described in [53], class D is included in the firewall computed for C. Therefore, any test case which exercises D is included in the regression test suite. However, there might be test cases which exercise only the method `D::bar()` and hence could have been omitted from the regression test suite. The firewall-based approaches are however computationally more efficient and are preferred for RTS analysis of large programs. Moreover, the technique proposed in [48] is more efficient than [53] since this method aims to achieve a balance between the efficiency of class firewall-based technique [53], and the precision of more fine-grained approaches like [77].

4.2 Program Model-Based Techniques

In the following, we discuss different RTS techniques [77, 73, 82, 41, 65] that have been proposed for object-oriented programs and are based on an analysis of program models for selecting regression test cases.

4.2.1 Technique Based on Class Dependence Graphs

Rothermel and Harrold were one of the earliest to propose an RTS technique for object-oriented programs [77]. They have divided the problem of RTS for object-oriented programs into two parts: RTS of the application program, and RTS of the modified or derived classes. For RTS of the application program, the technique models the original program P and the modified program P' using *IPDG* models. However, it is difficult to use an *IPDG* for RTS of modified and derived classes because an *IPDG* models programs having a single entry point whereas a class can have multiple entry points. This problem can be overcome by treating the test routines as application programs and then applying the approach for RTS of application programs. However, this approach incurs a large overhead because it may be necessary to construct and traverse a *PDG* for each method of a class several times. Therefore, the original and the modified programs are modeled as *ClDG*s for RTS of modified and derived classes. The test coverage information is used to associate the predicate and statement nodes of the

ClDG models with each test case. Then, a technique similar to [80] is used to select regression test cases.

4.2.2 Technique Based on Extended Control Flow for C++

Rothermel et al. [82] have proposed an approach for RTS of C++ programs based on an analysis of the control flow representations of the original and the modified programs by extending the technique proposed in [80]. Since a *CFG* represents the control flow information of only a single method, the concepts of Inter-procedural Control Flow Graph (*ICFG*) and Class Control Flow Graph (*CCFG*) have been introduced to represent control flow of multi-function programs and object-oriented programs respectively. An $ICFG$ for program P is composed of *CFG*s for each method in P. Each call site in P is represented by a pair of nodes called *call* and *return* nodes [82]. Each *call* node is connected to the entry node of the called method by a call edge, and each exit node is connected to the *return* node of the calling method by a return edge. An *ICFG* is used to model programs having a single entry point, whereas a class can have multiple entry points [82]. A *CCFG* is used to model classes, and consists of individual *CFG*s for all methods of a class. Given the graph models for the original and the modified programs, the RTS algorithm [82] extends the graph walk-based approach [80] to traverse the models and select relevant regression test cases.

4.2.3 Technique Based on Extended Control Flow for Java

Harrold et al. were the first to develop a safe RTS technique [41] for Java programs based on control flow analysis. Their technique is an adaptation of the graph walk techniques proposed in [80, 82], and can handle various object-oriented features such as inheritance, polymorphism, dynamic binding and exception handling. Their method consists of three steps: constructing intermediate representations for the source programs, analyzing the graphs and determining the set of dangerous edges, and test case selection. Harrold et al. use a *JIG* representation for modeling Java programs.

The two *JIG*s constructed for the original and the modified programs are simultaneously traversed (depth-first) to identify dangerous edges. Finally, based on the test coverage information obtained through code instrumentation, the technique selects test cases that exercise the dangerous edges identified during graph traversal.

4.2.4 Partition-Based Techniques

Partition-based techniques are motivated by the need to combine the effectiveness of precise but expensive RTS techniques with techniques that work at a higher-level of abstraction and are relatively imprecise.

Partition-Based RTS Technique for Java Programs: Orso et al. have presented a novel two-phase partitioning approach for RTS of large Java programs [73]. Their technique works in two phases, called *partitioning* and *selection*. In the partitioning phase, the original and the modified programs are modeled as Interclass Relation Graphs (*IRG*) [73]. The two *IRG*s are analyzed to identify hierarchical, aggregation, and use relationships among classes and interfaces. Then, the set of classes and interfaces that have been changed are identified. The partition phase analyzes syntactical changes at the *statement* level and the *declaration* level. A change at the statement level consists of addition, deletion or modification of program statements. A declaration level change means modifications in the declaration of the type of a variable, addition or deletion of a method, change in the modifier list of an existing method, etc. The class dependency information along with the changes is used to identify at an abstract level the affected parts of the code. The set of affected classes and interfaces identified from the first phase constitutes a *partition* of the program.

In the selection phase, a more detailed analysis of the partitions are carried out. The selection phase builds *JIG* models representing the modified regions of code from the partitions. The *JIG* models are then analyzed using the RTS technique proposed by Harrold and Rothermel in [80]. An edge-level test selection criterion is used to select test cases which execute the affected parts of code.

Partition-Based RTS Technique for C# Programs: Mansour and Statieh [65] have proposed a two phase RTS technique targeted for the C# programs. Their RTS technique first constructs an Affected Class Diagram (*ACD*) based on the changes made to the modified program. An *ACD* represents modifications made at the level of a class, an interface, web or window services, and COM+ components. Their technique then uses a test coverage criterion based on the *ACD* to select a subset of test cases. An *ACD* models a program at a high level of abstraction. A more detailed analysis is then carried out by modeling the programs using C# Interclass Graphs (*CIG*). A C# Interclass Graph (*CIG*) is a control flow graph that captures all the affected methods in an *ACD*. The technique constructs *CIG* models for the original and the modified programs, and regression test cases are selected based on the graph walk techniques [41, 82].

4.2.5 Critical Evaluation

The program model-based object-oriented RTS techniques [77, 82] are safe and are more precise as compared to the firewall-based techniques [53, 47, 1, 48], but are less efficient. This is because of the high overhead incurred in inter-procedural dependence analysis for large software. The *ClDG* model-based technique is also less efficient than the technique proposed in [82] since it is based on an analysis of dependence graphs while the analysis in [82] is based on control flow relationships. Both the techniques reported

in [82, 77] do not consider several other common features of object-oriented programs such as exception handling. These techniques [82, 77] can also be imprecise when testing affected polymorphic calls [41].

The techniques proposed in [41, 73] are safe RTS techniques for Java software, and are able to handle important object-oriented features of Java such as polymorphism, dynamic binding and exceptions. These techniques use a different method for capturing polymorphism than [82] which leads to a more precise selection of regression test cases. The two phase partition-based RTS technique proposed for Java programs [73] is comparatively more efficient than [41] because the analysis of the affected parts of the program is divided across two phases - a coarse-grained first phase and a more fine-grained second phase. This two-phased approach helps to limit the extent of code for which a minute low-level analysis is required.

Although the class firewall based technique [53] does not consider certain object-oriented features such as exceptions, the technique can still be extended for selecting regression test cases for a subset of the Java program features. The advantages of the firewall techniques are that they are comparatively more efficient than the program model-based techniques [41, 73], and can be applied for RTS of large programs. However as already discussed, the firewall techniques are unsafe and are relatively less precise than the techniques reported in [41, 73].

Mansour and Statieh's RTS technique [65] is a safe RTS technique tailored for C# programs. But the technique can be computationally expensive for large programs since the complexity of the algorithm is quadratic in the number of the *CIG* nodes.

4.3 Design Model-Based Techniques

Model-based testing of software has become very popular with the advent of the model-driven development paradigm. In the model-driven development (MDD) paradigm, a design model is usually refined to obtain the code. The widespread use of CASE tools for object-oriented system development ensures a close correspondence between a design model and its code. Hence, design models can effectively be used for RTS analysis of object-oriented programs [15].

Unified Modeling Language (UML) [12] is an ISO standard for representing analysis and design models of object-oriented programs. The following are some important advantages of UML-based regression testing [26]:

- *Traceability* - It is easier to maintain traceability between the design artifacts and the test cases than maintaining traceability between code and the test cases. It is also easier to identify changes between across different versions of design artifacts as compared to analyzing changes across code versions [15, 14].
- *Scalability* - Code-based regression testing becomes very expensive when applied to large programs. A model being a simplified representation of a code, model-based testing is comparatively more efficient.
- *Language independence* - Different parts of a software may be developed using different programming languages. It, therefore, is difficult to design and implement an RTS technique which can take into account parts developed using different programming languages during test case selection. A UML model-based RTS technique helps to overcome this shortcoming since it is independent of the implementation [14].

We now briefly discuss few UML model-based RTS techniques [4, 27, 33, 14, 69] that have been proposed in the literature.

4.3.1 RTS Based on Class and Sequence Models

Ali et al. have proposed an RTS technique based on analysis of UML class and sequence diagrams [4]. Their technique analyzes class and sequence diagrams at the level of class attributes and operations. Concurrency in sequence diagrams is captured by the use of asynchronous messages and parallel instructions which cannot be adequately represented by traditional *CFG* models [32]. Therefore, an extended control flow model called Concurrent Control Flow Graph (*CCFG*) has been introduced in [32]. Ali et al. extends the model-based control flow analysis proposed by Garousi et al. in [32] for regression testing based on UML design models by also including information available from class diagrams. A *CCFG* model is constructed for each sequence diagram. To model a sequence diagram invoking other sequence diagrams, the corresponding *CCFG*s are connected using control flow edges. The sequence and the corresponding class diagrams are analyzed and an extended concurrent control flow graph (*ECCFG*) is constructed to model the program. The information about which attributes of a class receive messages in a sequence diagram are derived from the corresponding class diagrams, and are represented in the *ECCFG*. The pre- and post-conditions of a method are also represented in an *ECCFG* by introducing new nodes. The *ECCFG* models for the original and the modified version of the application are then analyzed to find out the changes between program versions. This information is used to select regression test cases.

4.3.2 RTS Based on Class and State Machine Diagrams

Farooq et al. [27, 28] have presented a model-based RTS technique that uses information from UML 2.1 behavioral state machine and the structural class diagrams for test selection analysis. During software development, UML documents such as state machine and class diagrams describing the design and working of the software often undergo several modifications. The modifications made to one document may also affect other parts of the software.

Their proposed approach uses information from the modified class and state diagrams to find out the directly and indirectly affected elements of the model. For example, a state transition is considered to be affected if it uses any changed attribute or method of the corresponding class in its events, guard conditions, or actions. Those test cases that cover the modified transitions during execution are classified as retestable test cases.

4.3.3 RTS Based on Control Flow Analysis of Sequence Diagrams

Naslavsky and Richardson have proposed an RTS approach [69] based on control flow analysis of UML sequence diagrams for a MDD environment. The technique involves a model-based transformation from a sequence diagram to a *CFG*. The traceability between test cases and the sequence diagrams is used to determine which *CFG* elements are executed by each test case. The two *CFG*s corresponding to P and P' are then analyzed to find out the affected model elements, and the traceability information is used to select relevant regression test cases.

4.3.4 RTS Based on Use Case Diagrams

Gorthi et al. have proposed an RTS technique [33] based on UML use case diagrams. Their approach uses the concept of *behavioral* slicing which decomposes use cases into user actions followed by some computations and the output. Behavioral slicing helps in identifying changes made to the activity diagrams. Each node in an activity diagram is also assigned a criticality value to help increase the effectiveness of the selected test cases. Whenever the requirements are modified, the activity diagrams are also modified to reflect the changes to the system. The models for the original and the modified specifications are then analyzed to find out the *affected* paths in the diagram. The paths in the diagram that have one or more modified nodes are considered to be affected and the test cases which execute the affected paths are selected for regression testing.

4.3.5 RTS Based on UML Architectural and Design Models

Briand et al. have proposed an RTS approach based on analysis of UML design models [14]. Their approach assumes full traceability between the design model(s), the code and the test cases. The traceability between the design and test cases helps in associating the changes in the design models to the test cases which need to be executed to exercise the affected parts in design. Their approach involves analysis of use case, class and sequence diagrams. Their technique also assumes that there is a unique sequence diagram specifying possible object interactions along with each use case. The approach assumes that any pre- or post-conditions among classes are specified using the Object Constraint Language (OCL). Their analysis classifies test cases as obsolete, retestable and reusable test cases.

4.3.6 Critical Evaluation

In the following, we present a comparative evaluation of the UML-based RTS techniques that we discussed in subsection 4.3. The RTS evaluation framework proposed by Rothermel and Harrold [79] were originally for code-based techniques, and hence cannot be used in a straightforward manner to evaluate UML-based RTS techniques. For example, in the context of a UML-based RTS technique, a test case is modification-traversing iff it triggers a changed UML model element (e.g., messages for UML sequence diagrams).

The techniques proposed in [27, 28, 14] are safe with respect to the changes possible to the UML artifacts. An advantage of RTS based on analysis at a higher level of abstraction is improved efficiency as compared to code-based techniques. However, RTS based on UML design models are not as precise when compared to detailed code analysis-based techniques [26].

UML model-based RTS techniques require a close correspondence between the requirement artifacts, design models, code and the test cases, which may not always be possible in practice. Therefore, the applicability of these techniques is limited to a MDD environment.

4.4 Specification-Based RTS Techniques

In the industry, a practical difficulty in RTS is that the testers may not have access to the design models or the actual source code. In such scenarios, model-based or code-based analysis is not possible. This is especially true for COTS applications. Also, it is difficult to apply program analysis techniques and tools for many legacy software which have been developed using older programming languages (e.g., COBOL) for which there is a dearth of effective program analysis techniques [18]. Code-based techniques may also suffer from problems of scalability [16]. These limitations have motivated researchers to develop RTS techniques [18, 16] based on specifications which are usually available to the testers. In this context, it should be noted that although we have classified these techniques as a subtype of object-oriented RTS techniques, these techniques can be extended to a wider variety of programming paradigms such as component-based software.

4.4.1 Activity Diagram-Based Selection

Chen et al. have proposed a specification-based RTS technique [16] which uses UML activity diagrams for modeling the potentially affected requirements and system behavior. They have also classified the regression test cases that are to be selected into *target* and *safety* test cases. Target test cases are those that exercise the affected requirements, while safety test cases help achieve a pre-defined coverage target. The steps involved in selecting target test cases are as follows: A traceability matrix is created to capture the association between requirements and the test cases, i.e., which test cases exercise a particular requirement. The

Class of RTS Techniques	References	Key Features	Merits	Demerits
Firewall-based techniques	[53, 1, 48, 47]	Analyzes dependencies among modules	Computationally efficient	Unsafe and imprecise, need to be extended to handle certain object-oriented features such as exceptions
Program model-based techniques	[77]	Analysis is based on dependencies among class elements	Safe, and more precise than firewall-based techniques, is applicable for RTS of both modified classes, and classes derived from the modified classes, and application programs	Computationally more expensive than the firewall techniques
	[82]	Based on analysis of control flow models	Safe RTS technique for C++ programs, more efficient than [77]	Does not consider some common object-oriented constructs like exception handling, can be imprecise
	[41, 73]	Based on analysis of control flow models, two-phased technique [73]	Safe and precise RTS techniques for Java programs, two-phased technique is more computationally more efficient than [41]	Expensive for large programs with small changes because of fine-grained analysis
Design model-based techniques	[69, 4, 14, 27, 33]	Based on analysis of different UML design models (e.g, sequence, activity, use case diagrams), assumes that a traceability exists between the design models, the source code, and the test cases, suited to model-driven development environments	More efficient than program model-based approaches, suited for RTS of large programs, analysis is at a higher level of abstraction, and is independent of the implementation	Not safe, comparatively less precise than program model-based RTS techniques
Specification-based techniques	[18, 16]	Based on analysis of requirement models, assumes complete traceability from the specifications to test cases	More efficient than program model-based approaches, can be applied to systems with large test suites, techniques are platform-independent can be easily extended to a wide class of programs	Not safe, comparatively less precise than program model-based RTS techniques

Table 2: A comparison of RTS techniques for object-oriented programs.

modifications that are made to the original program P can result in a change of specification, or can be changes which are limited only to the code. In case when the changes are limited only to the code, the elements (nodes and edges) which are affected in the relevant activity diagrams are identified. Chen et al. have extended the RTS technique proposed in [82] to handle those modifications which lead to changes in the specifications also. Safety test cases are selected with an aim to mitigate risks. The idea is to more thoroughly test those parts of the code for which the probability of a fault being present and its cost (i.e., consequence of impact) is high [16].

4.4.2 Requirement Coverage Matrix-based Approach

Chittimalli and Harrold [18] have proposed a specification-based RTS approach. Their technique is essentially based on tracking which specifications are being tested by which test case from T. This information is represented as a *requirement coverage* matrix between the set of requirements and the test cases. The technique proposed in [73] has been used to identify the affected parts of the code, and subsequently the set of requirements that are affected due to changes are also identified. These are termed as *affected* requirements. The information from the requirement coverage matrix is used to select the test cases which exercise the affected requirements.

4.4.3 Critical Evaluation

The specification-based approaches are efficient as they do not depend on any static analysis of the source code. For the technique proposed in [18], the safety and precision of the approach is largely dependent on the quality and accuracy of the requirement coverage matrix. However, the safety of the approach is compromised by fact that dependence relationships existing among program elements cannot be completely and accurately captured by the requirement coverage matrix. Moreover, often in practical situations, code changes may be too trivial to affect the requirements, and the requirement coverage matrix may also be out of date.

We summarize the merits and demerits of the different RTS techniques applicable for object-oriented programs in Table 2. In column 3, we highlight the key features of each class of techniques, and summarize the merits and demerits in columns 4 and 5.

5 RTS Techniques for Component-Based Software

In the component-based software development model, a software product is developed by integrating different components developed either in-house or by third-party vendors. The reliability of a component-based software ap-

plication, to a large extent, depends on the reliability of the individual components. These blackbox components are often modified by the concerned vendor to fix bugs and incorporate enhancements. Hence, regression testing of component-based software needs to address how the changes made to a component might affect the execution of application programs which use those modified components. Techniques which perform RTS of traditional programs cannot meaningfully be used for RTS of software using COTS (Commercial Off-The-Self) components because the code for the components are usually not available. RTS for component-based software is a challenging research problem due to the following reasons [31, 72]:

- In a component-based development environment, often there is a lack of adequate information about the changes made to each release of a component. Relevant information such as control and data flow relationships among the modules are usually not supplied to the application programmer. Moreover, there is also a lack of adequate documentation for third-party components.
- A change made to a component may be reflected both at the component level and at the system level functioning of the software. Even trivial changes made to a component in a system may at times affect the proper working of the software as a whole.
- There is a lack of test tools which can be used to identify changes in a component and its impact on the software.

Depending on the type of program analysis, we classify the RTS techniques [72, 66, 67, 87, 31, 74, 115, 117, 116, 107] proposed for component-based software into the following classes:

1. Metacontent-based RTS approaches
   (a) Code coverage-based approach [72]
   (b) Enhanced change information-based approaches [66, 67]
2. Model-based techniques
   (a) UML model-based techniques [87, 107]
   (b) Component model-based technique [31]
   (c) Dynamic behavior and impact analysis using models [74]
3. Analysis of executable code [115, 116, 117]

In the following, we review a few prominent RTS techniques reported for component-based software.

5.1 Metacontent-Based RTS Approaches

The difficulty of inadequate information exchange between the component user (c-user) and the component developer (c-developer) during component-based software development can be overcome by sharing relevant component information required for RTS analysis. Orso et al. [71] have proposed the concept of content change information, called *component metacontent*, as a means of sharing information about the changes that a component undergoes across different versions. Different RTS techniques may define their own sets of required metacontents that need to be shared by the c-developers. Some examples of the type of information that are shared as metacontents range from the component version to more detailed like the coverage information of a particular test suite on the concerned component.

In the following, we discuss the different metacontent-based RTS techniques [72, 66, 67] reported in the literature.

5.1.1 Code Coverage-Based Approach

The code coverage-based RTS technique for component-based software was proposed by Orso et al. [72] and is based on existing procedural RTS techniques [17, 80, 81].

In case the c-users are unaware of the components that have undergone a change, then during RTS for the application code, any test case in which a method of the modified components is called is selected for regression testing. This can lead to selection of test cases unrelated to the specific change. A more precise selection of test cases can be made with information about the modifications made to the components. To enable a more precise selection of regression test cases, the technique [72] assumes the availability of the following metacontent information:

- Coverage information of the initial test suite on the component.
- Component version.
- Set of control flow edges affected due to the modifications to the component.

The c-developer supplies this information in the form of metadata and metamethods during the release of the modified component. The coverage information is based on the *CFG* edges traversed during execution of a test case. Based on the coverage information, test cases which execute the affected edges of the *CFG* are selected for regression testing according to the graph walk technique proposed by Rothermel and Harrold [80] .

5.1.2 Enhanced Change Information-Based Approaches

Mao et al. have observed [66] that the applicability of the technique suggested by Orso et al. [71, 72] is restricted due the fact that it requires a very detailed metacontent information to be provided by the c-developer. They have proposed RTS approaches [66, 67] which emphasize the availability of specific data from the c-developers to the c-users.

Change Information-Based Approach: A component provides services when invoked through its published APIs. Information is exchanged between components and the application program by means of the parameters of the published APIs, and the component variables which can directly be accessed from the application program (called

published variables or PVs). Keeping this in view, the approach suggested by Mao and Lu [66] aims to identify changes at the method level for the modified components (to be performed by the c-developers). Invocation of methods within each component is modeled by constructing a Labeled Method Call Graph (*LMCG*) for each component. An *LMCG* for a component C is defined as $LMCG(C) = (V, E)$, where V represents the set of methods in the component (both published APIs and internal methods), and E represents the call relations among different methods of the component along with the pre-conditions required for a successful invocation. The enhanced change information (ECI) consists of the set of published APIs and the pre-conditions for invoking each published API. The ECI for the modified program statements in a component can be computed as follows: Suppose a certain method A invokes a method B in some component C. Then, the pre-condition for the method A invoking method B can be found out by analyzing $LMCG(C)$.

The ECI for the modified components are supplied to the c-users as files in a standard format (such as XML) along with the executable of the updated component. The c-users need to instrument the application source code to find out the values of the PVs and the parameters that are passed to each published API that is present in the ECI. For each test case, if the recorded values of the input parameters and the PVs satisfy the pre-condition for that published API, then the test case is selected for retesting the application integrated with the modified component.

Built-in Test Script-Based Approach: An RTS technique for component-based software using another level of information interchange between c-users and c-developers has been reported in [67]. This approach is motivated by the fact that it is only the c-developers who have detailed knowledge of the working of a component and the modifications effected to each of its versions. This technique proposes that the c-developers place test scripts in the component source code during modifications. The purpose of these test scripts is to gather information about the execution pattern of the component during execution of the test cases. This information helps to identify the test cases which cover the modified statements of the component.

A Method Call Graph (*MCG*) for a component C is defined as $MCG(C) = (V, E)$, where V represents the set of methods in the component (both published APIs and internal methods), and E represents call relations among the different methods in the component. The component APIs affected due to modifications to a component C are identified by the c-developers by analyzing the relationships between component methods using *MCG(C)*. Test functions for the affected methods are designed by the c-developers and are also published so as to facilitate selection of test cases by the c-users. The execution information gathered on invoking the test methods are used by the c-users to select test cases which execute the affected component methods.

5.1.3 Critical Evaluation

The metacontents-based approach [72] selects regression test cases by performing control flow analysis at the statement-level, and hence can be expensive for large programs. This problem can be overcome by using a coarser granularity during RTS analysis (method or class level) [66, 67]. The metacontent information in this case should be provided at the method or class levels.

5.2 Model-Based RTS Techniques

Model-based RTS techniques proposed for component-based software products are essentially refinements to model-based RTS techniques proposed for procedural and object-oriented programs. In the following, we briefly discuss a few model-based RTS techniques [87, 107, 31, 74] proposed for component-based software.

5.2.1 UML Model-Based RTS Techniques

Sajeev and Wibowo have proposed an RTS technique [87] for component-based software using UML and OCL models. Their technique assumes that the functionalities provided by the modified component is a superset of the functionalities provided by the original component, i.e., the new component version may include bug fixes and optimizations of the existing functionalities along with new functionalities that have been introduced. While UML is used to model the function call relations across components, OCL is used to represent the change information across component versions. The sequence of methods that are invoked by each test case is also tracked. All those test cases which either execute a directly modified method or a method which in turn invokes a directly or indirectly modified method are selected for regression testing.

Wu and Offutt have proposed another UML model-based RTS technique [107] for component-based software. In this technique, collaboration and sequence diagrams are used to analyze the control flow behavior of a component and how objects interact with each other through message description. The changes made to a modified component version will be reflected in a collaboration diagram as a change to a class method, or a change in the interaction sequences. The statechart diagrams are used to analyze the internal behavior of objects of a component. The class diagrams are used to identify the affected classes when the definition of one class is modified. For each modification in the collaboration diagram, the affected parts of the component are identified using control and data dependency analysis on the collaboration and the corresponding statechart diagrams. The test cases executing the affected parts are selected for regression testing.

5.2.2 Component Model-Based Technique

Gao et al. [31] have introduced several new models such as the Component Function Access Graph (*CFAG*), the Dy-

Class of RTS Techniques	References	Key Features	Merits	Demerits
Metacontent-based approaches	[72, 66, 67]	Assumes availability of metacontent information for RTS analysis	Metacontent information can be easily prepared, exchange of change information is simpler in [66, 67] than [72]	Emphasizes mutual collaboration between c-users and c-developers, control flow-based analysis in [72] may be expensive for large programs
Model-based	[87, 31, 74, 107]	Based on analysis of component models, models are passed as metadata	Computationally more efficient than the metacontent-based approaches	Proposed techniques are not safe and are less precise than metacontent-based approaches
Executable analysis-based	[115, 117, 116]	Novel approach based on reverse engineering the component binaries	Minimum dependence on the c-developers	Can be imprecise as selection analysis is at the function level, difficult to precisely identify changes among binaries

Table 3: A comparison of RTS techniques for component-based software.

namic CFAG (*DCFAG*), the Function Dependency Graph (*FDG*) and the Data-and-Function Dependency Graph (*DFDG*) to represent component API-based information at the system level. A *CFAG* models the static function call relationship of the component APIs, i.e., function calls from the application code to the component APIs. Each node in a *CFAG* represents a component API. An edge $e_i = (f_i, f_j)$ between two nodes (i.e., methods) f_i and f_j denotes that the second method is invoked after the first method. A *DCFAG* model provides a dynamic view of the function call sequences during the execution of a particular test case. Therefore, there can be many *DCFAG*s possible for a component C and a component API A_i. An *FDG* model is used to represent invocation dependencies between two functions in a component. A *DFDG* model is used to represent the define and use relationships among functions and variables. The technique [31] assumes that these models are supplied as metadata with new component releases.

For RTS analysis, the c-users require information about the modifications made to the component APIs. Changes to a component API are possible due to many reasons, such as modification to a function prototype, addition/deletion of parameters to a function, etc. These changes can be identified by comparing the revised component API specifications with the older version. However, there may be other dependency relations (control and data) which may indirectly affect APIs which are not themselves modified. These indirectly affected APIs are identified by analyzing the *FDG*s (for functions) and the *DFDG*s (for data variables).

Gao et al. have extended the firewall approach [53, 58] to identify the impact of component modifications on the other elements of the component (functions and variables). New types of firewalls are introduced to compute the set of affected functions due to changes in other APIs, functions or data variables of the component. For each modified element of a component, the firewall approach helps to identify the set of directly or indirectly affected component APIs. Regression test cases are then selected based on whether a test case executes the affected component APIs or not.

5.2.3 Dynamic Behavior and Impact Analysis Using Models

In [74], Pasala et al. have proposed an RTS technique for component-based software that analyzes the dynamic behavior (e.g., interaction of methods at runtime) of components to select test cases. This technique [74] is able to select regression test cases for components developed in .NET and Java. The information about the dynamic behavior is captured by executing the initial test suite and tracking the sequence of method invocations. These interactions are modeled as Functional Interaction Graphs (*FIG*). This step needs to be run once for every software application that is to be regression tested. To identify the affected methods in the newer component versions, the component binaries are reverse engineered to generate an intermediate code. The syntactical changes between different components are identified to track the directly affected methods, and the changes are then semantically analyzed to determine the set of indirectly affected methods. Once the complete set of affected methods are determined, the *FIG*s are then analyzed to select test cases relevant for regression testing.

5.2.4 Critical Evaluation

The RTS techniques proposed by Sajeev and Wibowo [87] and Wu and Offutt [107] are imprecise because these techniques perform RTS analysis at a high level of abstraction such as classes and methods. These techniques are also less safe than [31, 74] because they do not involve detailed dependency analysis at the statement level. However, the technique proposed by Gao et al. [31] is also not safe as it does not consider the effect of component modifications on the software as a whole and limits impact analysis to only the component level.

5.3 Analysis of Executable Code

Zheng et al. have proposed a family of RTS techniques [115, 116, 117] based on analysis of the executable code (binaries such as .dll, .lib) of the modified components. Their techniques are known as Integrated - Black-box Approach for Component Change Identification (*I-BACCI*),

along with a version number to specify the exact approach. The *I-BACCI* technique uses the firewall approach for analysis of the glue code (application code which integrates the COTS components [116]). This technique utilizes the following information for RTS analysis:

- Binary files for the original and the modified versions of all the changed components.
- Glue code.
- Initial test suite developed for the glue code.

This technique involves reverse engineering the executables to identify the function definitions, code sections, etc. The extracted source code of the two versions of a component are then analyzed to find out the functions which have been modified between the two versions. Then, function call graphs (*FCG*) are constructed based on the identified function call relationships for the modified components. The *FCG*s are analyzed to find out the functions in the glue code which call published component functions. The glue code functions which directly or indirectly invoke the modified component functions are considered to be affected. The test cases that execute the affected functions in the glue code are selected for regression testing.

5.3.1 Critical Evaluation

The *I-BACCI* technique has the minimum dependency on information required from c-developers. However, an important limitation of the approaches proposed in [115, 117, 116] is precise identification of the changes between the component versions by reverse engineering the executables. For example, during RTS analysis, the technique may fail to identify and ignore all trivial differences that are introduced in the component executables due to build configurations, build and target platforms, etc. These techniques are also not precise since they do not perform a statement-level analysis, and affected code elements are identified at the level of functions. Therefore, there might be glue code functions which invoke modified published functions of the component but do not actually execute the modified program statements in the published function. Test cases which exercise such glue code functions can in fact be safely ignored during regression testing.

We summarize the important characteristics of RTS techniques proposed for component-based software in Table 3. The key features, merits and demerits of each RTS technique as compared to similar techniques proposed for component-based software have been presented in columns 3, 4 and 5 respectively.

6 RTS Techniques for Database Applications

A large number of database applications are currently in use. These applications are usually composed of several components contributing to an increase in their sophistication [38]. Database applications also need to be frequently modified due to different requirements, e.g., change in components, growing number of users and data, etc. In this context, regression testing of database applications is an important activity.

The requirements and challenges in regression test selection of database applications are different from the classes of programs that we have discussed so far. Regression test selection of database applications need to take into account the following features:

- RTS techniques for other classes of programs implicitly assume that the test cases are independent of each other and can be executed in any order. This assumption is not valid for database applications as the output of a test case may change the *database state*, in the process affecting the execution of other test cases. Therefore, in addition to the global program state, the states of the database need to considered during RTS for database applications.
- The state of the database may have to be *reset*, i.e., restore the initial database configuration, many times during regression testing. Resetting of a database is acknowledged to be an expensive activity both in terms of cost and time [38].
- Database languages support features such as structured queries, integrity constraints, exception handling and table triggers, which complicate impact analysis of the modified parts of the program. For example, firing of triggers can create implicit intermodular control dependencies [39].

The traditional notions of safety and dependencies cannot be applied in regression testing of database applications because those techniques were developed for *stateless* applications. In this context, a few RTS techniques have been proposed for database applications [105, 39]. We briefly review these techniques in this section.

6.1 Two Phase RTS Technique for SQL-Based Systems

Haraty et al. [39] have proposed a two-phase technique for RTS of structured query language (SQL) based database applications. Apart from traditional control and data dependencies among elements in a database application, Haraty et al. have identified the following aspects that need to be considered:

- Dataflow dependencies - Dataflow dependencies can arise among database modules due to usage of tables across modules.
- Component dependencies - These arise among different database modules due to firing of table triggers, modifications to tables or views, or due to modifications to SQL statements. Component dependencies are transitive in nature.

– Exception handling - Raising of exceptions can affect control flow relationships, which need to be taken into account during RTS analysis.

Haraty et al. have proposed a control flow model of SQL statements where a node in the *CFG* represents an SQL statement. Their modeling technique also represents possible changes in control flow that arise due to exceptions. They have identified two types of changes that are possible in a database application:

1. Code changes - These are possible additions, deletions, and modifications to SQL statements within a database module.
2. Database component changes - These include changes to the database component definition itself, e.g., changes in the interface.

Their technique determines modifications between the two versions of the program and identifies potential areas of the code where the changes can impact. To identify the set of affected components due to modifications, Haraty et al. have used the concept of a *component firewall*. A database module is considered to be affected and is, therefore, included in the component firewall if any one of the following conditions holds:

– The definition of the module is modified.
– The module is deleted.
– The module is data or control dependent on another modified or deleted module.
– The module becomes dependent on some other module due to modifications.

The first step in constructing the component firewall is to identify the directly changed modules. Then, the transitive closure of the directly changed modules is computed to find the set of all potentially affected database modules. In the second phase, relevant test cases are selected based on any one of the two algorithms: one is based on traversal of *CFG*s and the other is based on firewalls. The firewall-based algorithm is based on analyzing the module-level dependencies among database components.

6.1.1 Critical Evaluation

Experimental studies show that the firewall-based RTS may ignore omitting potential modification-revealing test cases, and is therefore unsafe [105]. The firewall-based technique is also imprecise for reasons similar to the firewall approaches proposed for traditional programs.

6.2 CFG-Based Safe RTS Technique

Willmor and Embury [105] have extended the safe control flow analysis-based RTS algorithm proposed by Rothermel and Harrold [80] for procedural programs to database applications. RTS based on only definition-use relationships is not safe for database applications. This is because it is possible for an instruction to write some data to the database that will later be read by a program statement that precedes the earlier instruction in some execution path [105]. Therefore, the authors have introduced the concept of *database dependencies* to capture the additional dependencies that arise among elements in a database program. A statement is called *database dependent* if the statement can update the database, and has been modified in P' such it can affect the database state. Statements which are dependent on database dependent statements are considered affected, and the test cases that execute these statements are called database-dependent test cases. Based on these additional dependencies, Willmor and Embury's technique selects modification-revealing test cases with respect to the program state, and database-dependent test cases with respect to the database state.

6.2.1 Critical Evaluation

The technique proposed in [105] is the first safe RTS technique for database applications. However, the approach can be imprecise. Consider the case in which a statement that adds new tuples to a table is modified so that it is capable of adding only a subset of the tuples that could be added by the original statement. In such circumstances, no new faults can be introduced in the modified code due to the change which cannot already be detected in the original program. Therefore, those test cases which test code that can potentially be affected by this change need not be selected.

7 RTS Techniques for Web Applications and Services

Web applications and services are dynamic in nature and constantly evolve. They are frequently updated and, therefore, need to be regression tested to verify the correctness of the unmodified functionalities. In the following, we discuss the main features of web applications and services that need to be taken into account during RTS:

– Web applications are composed of server side services, user applications, and middleware. A safe RTS technique for web applications should consider all types of dependencies that can arise in the different layers of the web application under test.
– Web applications and services are inherently distributed in nature and are loosely-coupled.
– Web services are usually composed of and make use of other services. Therefore, the dependencies arising due to a modification to another service also need to be considered during RTS.

A difference in the nature of composition of component-based and web applications is that a component-based software physically integrates a component. Therefore, it is up to the component user to upgrade to newer releases

of the component. However, a web service can be updated as deemed fit by the concerned developer and is not owned and neither controlled by the application developers, thereby further complicating RTS of web applications.

Recently, many RTS techniques have been proposed for web applications [86, 93, 61, 110, 85]. We briefly review these techniques in this section.

7.1 RTS for Web Applications Based on Slicing

Xu et al. have proposed an RTS technique for web applications based on slicing [110]. They assume that web applications consist of multiple static HTML pages and programs running on the server side. The types of changes that an HTML page can undergo can be divided into the following basic classes: insertion of a page element (e.g., anchor, hyperlink, etc.), deletion of a page element, insertion of a page and deletion of a page. More complex changes are decomposed into a combination of these basic modifications. In this context, it needs to be noted that certain kinds of changes, such as formatting related changes, cannot affect other web pages. The different HTML pages in an web application can be data or *hyperlink* dependent on each other. The dependencies that can arise are further divided into direct and indirect dependencies. For example, if a hyperlink is inserted, then it needs to be checked that the link is working if the target page is part of the website. Indirect dependencies arising due to definition-usage relationships of variables are analyzed using traditional techniques. Then, the slice is computed on the indirect data dependencies on an extended *SDG* model of the web application which is to be regression tested. Test cases that execute the potentially affected web elements are selected for regression testing.

7.1.1 Critical Evaluation

The details of the slicing technique used for identifying potentially affected web elements have not been provided in [110]. However, it can be inferred that the technique will suffer from drawbacks similar to slicing *SDG*s for procedural programs. The technique is, however, precise in selecting relevant test cases for the types of changes that have been considered.

7.2 RTS Based on System Models

Tarhini et al. have proposed a safe RTS technique for web services-based applications [93]. The technique defines web services as self-contained component-based applications residing at separate locations and communicating using XML-encoded messages using SOAP interfaces. The communication using message exchange may also be time-constrained. The services provided by a web service are shared using WSDL specifications.

The authors have modeled a web application in two hierarchical levels to avoid state explosion. In the first level, the interaction of the components with the main application is modeled using a Timed Labeled Transition System (*TLTS*). Each node in a *TLTS* represents a component, and an edge joining two nodes represents a transition between the two components. The internal behavior of each component is modeled in the second level. Each node in the second-level *TLTS* represents a state of the component that is being modeled. The authors have proposed an RTS technique which selects all relevant test cases that test the side-effects of adding, removing or fixing an operation or a timing constraint in an existing component based on an analysis of the constructed two-level *TLTS* models. The approach for selecting relevant regression test cases is as follows: Construct the *TLTS* for the modified web service; generate test cases for testing the *TLTS* model corresponding to the modified web service; find the difference between the initial test suite and the generated test suite. The differential set of test cases are selected for regression testing.

7.2.1 Critical Evaluation

The technique in [93] is safe because it selects every test case that produces a different behavior in the modified system. However, this technique cannot strictly be considered as a pure RTS technique because the analysis involves generation of test cases as an intermediate step.

7.3 Control Flow-Based RTS Techniques

Ruth et al. [86, 85] and Lin et al. [61] have proposed safe RTS techniques for web services based on analysis of control flow models. We discuss these techniques in the following.

The RTS technique proposed by Ruth et al. [86, 85] is a gray-box technique since it is difficult to carry out white-box regression testing for web services because often the source code for the components may not be available with the web service developer. It is a gray-box technique because it does not require the source code of the web services. Instead, their approach assumes that the component web service providers would provide the following information as metadata along with a service release: WSDL specification, a set of test cases, *CFG*s for the web services, and test coverage information.

Their technique requires that each procedure in a web service is modeled as a *CFG* at the service developer side. Their technique also assumes that the method calls to other services are decided statically. The *CFG*s for all the individual procedures are then combined to form a global *CFG*. When a web service is modified, then a global *CFG* is also constructed for the modified web service. Each node in a *CFG* stores a hash code of the corresponding statement. The authors have extended the graph traversal algorithm proposed in [80] to simultaneously traverse the global *CFG*s for the original and the modified programs and identify the nodes of the graph which are changed. All control flow edges which can be reached from the modified

nodes are marked as *dangerous*. The changes made to the modified web service can be identified from a difference in the hash values without requiring analysis of the source code. The test cases which execute the dangerous edges are selected for regression testing.

Lin et al. have proposed a safe RTS technique [61] for Java web services based on code transformation. Their technique models Java code at the client side and the service side as a single *combined* program. The services and the interfaces provided by the web service are available from the WSDL specifications. The technique simulates message passing between the client application and the web service through local proxy objects. The merged program is then modeled as a *JIG* which has the same structure as the original application. Once modeling of the original and the modified web service is complete, the algorithm proposed in [41] is used to select relevant regression test cases.

7.3.1 Critical Evaluation

The control flow-based techniques proposed in [61, 85, 86] have advantages and disadvantages that are comparable to the control flow-based RTS techniques proposed for procedural programs. These techniques are safe, and comparatively more precise and at the same time less efficient than the technique proposed in [93]. The technique proposed by Ruth et al. [86, 85] is also similar to component-based RTS techniques [72, 66, 67] because it relies on metacontent information supplied by the web services developers.

8 RTS Techniques for Aspect-Oriented Programs

Aspect-oriented software development paradigm is an emerging methodology that aims to modularize software development by isolating low priority and auxiliary functionalities from the application's main business logic. In the traditional programming model, it is up to the programmer to manage and interleave other auxiliary issues (called as *concerns*) into the main application code. Concerns which are spread across multiple modules are called *crosscutting* concerns. For example, for a programmer who is developing a module for a banking software, related issues such as logging, performance, security, authentication, exception handling, etc. are examples of crosscutting concerns. Aspect-oriented programming (AOP) allows programmers to relegate these secondary crosscutting concerns to stand-alone modules called *aspects*. AOP has also introduced new terminologies such as *advice*, *pointcut*, *introduction*, *join points*, *shadow*.

AOP has been adopted for many object-oriented programming languages and AOP languages such as AspectJ has gained considerable popularity among the Java developer community. Introduction of aspects usually change the behavior of the original Java program. Therefore, AspectJ programs also need to be thoroughly regression tested after some modifications. In this section we discuss the proposed RTS techniques for AspectJ programs [114, 109] since AspectJ is the most widely used aspect-oriented language [50].

8.1 RTS of AspectJ Programs using Control Flow Models

Zhao et al. [114] proposed an RTS technique for AspectJ programs by extending the work of Harrold et al. [41]. They have proposed a System Control Flow Graph (*SCFG*) and an Aspect Control Flow Graph (*ACFG*) to model an AspectJ program. The authors have introduced additional nodes and edges, such as *join point* vertex, in an *SCFG* to model AspectJ constructs. Each individual aspect in a program is represented using an *ACFG*. An *ACFG* is composed of individual *CFG*s which represent static control flow relationships that exist among advice, inter-type members, and methods of an aspect. An *aspect entry* vertex is used to represent entry to the aspect. An *aspect membership* edge is used to connect the *aspect entry* vertex to different possible aspect members such as advice, inter-type members, pointcuts, or methods. Their model is also able to represent interactions between aspects and classes.

Once the *SCFG* graphs have been constructed for the original and modified pair of AspectJ programs, the depth-first search technique proposed in [41] is used to identify the dangerous edges in the graph. The test cases that execute the dangerous edges are selected for regression testing.

8.2 RTS for AspectJ Programs Based on Extended JIG

Xu and Rountev [109] have proposed a safe intermediate graph representation-based RTS technique for AspectJ programs. They have first proposed a control flow-based graph model for AspectJ programs named AspectJ Intermodule Graph (*AJIG*) which is an extension of a *JIG*. An *AJIG* consists of *CFG*s that model control flow relationships within Java classes similar to *JIG*s, within aspects, and across boundaries between aspects and classes through non-advice method calls. An *AJIG* also consists of interaction graphs that model interactions between methods and advices at certain join points. The following types of interactions are modeled by an *AJIG*:

- A method call from a Java class method to another class method or an aspect method.
- A method call from an advice to a class method or an aspect method.
- A method call from an aspect method to a class or an aspect method.

Relevant regression test cases are selected by comparing the *AJIG* graphs for P and P'. The authors have extended the graph traversal algorithm proposed in [41]. Their two-phase algorithm also handles situations where the destina-

tion nodes for a pair of edges that are compared are statement shadows. In the first phase, the invocation order of the advices are compared using the two interaction graphs corresponding to P and P'. The output of the first phase is a set of *dangerous* edges in P that are changed in P' and a set of advices whose invocation order remains the same and whose bodies need to be further inspected in the second phase. In the second phase, the *CFG*s for each advice identified in the first phase are traversed to identify dangerous edges. The test cases executing the set of dangerous edges are selected for regression testing.

8.3 Critical Evaluation

The technique proposed by Zhao et al. [114] ignores situations where multiple advices apply at a shadow, or where there can be dynamic advices. Their proposed graph model cannot represent these suitably, and hence, may miss out on selecting potentially fault-revealing test cases.

The RTS technique proposed by Xu and Rountev is a safe RTS technique for AspectJ programs and overcomes the drawbacks inherent in the RTS technique proposed in [114].

9 RTS Techniques for Embedded Programs

During the last decade, there has been a rapid surge in the usage and reach of embedded applications. A variety of embedded applications have infiltrated almost every facet of our daily lives. Over the years, embedded applications are becoming more and more sophisticated and are being extensively used in real-time and safety critical applications. The domains where embedded applications are being heavily used at present include entertainment, automobiles, life-saving medical instruments, nuclear power stations, and defense warfare. As a result, extremely reliable operation of these applications has become an essential necessity.

Regression testing is a challenging task in the life cycle of an embedded software [88, 70, 91]. Embedded applications are often composed of concurrent, co-operating tasks, many of which may be real-time in nature. Issues such as concurrent execution and deadlines of tasks add new dimensions to the complexity of testing embedded programs. For example, concurrent tasks in an embedded program may get scheduled differently even when the same set of events occur with minor alterations to their timing of occurrence. This can cause unrepeatable test results. Furthermore, an error which is not manifested in one test case may be exposed by another test case having the same inputs, same start state and executing the same functions but having a different timing behavior. Embedded systems usually accept inputs (in the form of events) from the environment concurrently and asynchronously. Since it is not always possible to predict the exact input pattern, therefore, the behavior of an embedded system needs to be tested for all possible input combinations. This may necessitate testing of embedded software using a large number of test cases. Moreover, the high cost of execution of test cases for embedded programs makes minimization of the costs incurred in regression testing highly desirable [36]. Selection of a set of safe test cases for embedded applications has, therefore, been acknowledged as an important research problem [118].

The RTS techniques for traditional programs cannot satisfactorily be used to select regression test cases for embedded programs, since embedded programs have many features that are radically different from traditional programs. A few examples of these features are the following:

- A real-time task is usually associated with a deadline by which it needs to produce the required results. Thus, test cases validating the timing aspects of a modified feature need to included. Therefore, an RTS approach based solely on analysis of data and control dependency aspects alone would be unsafe. In this context, analysis of control flow information for checking the timing properties has been advocated by many researchers [102, 45].
- Embedded programs are concurrent and event-driven. The dependencies arising due to these features can result in subtle bugs in the programs, and need to be specifically regression tested.
- Embedded programs often use explicit exception handling mechanisms. This is especially true for safety-critical applications where error situations need to be properly handled. Throwing an exception alters the normal flow of control in a program. Hence, all affected control flow paths in the program need to be regression tested.

In the literature, we could find only one study by Biswas et al. [11] related to RTS of embedded applications. In [11], the authors have proposed an *EClDG* model for representing embedded programs. An *EClDG* model is an extension of a *ClDG* and represents both control and data dependencies that exist among program elements. An *EClDG* also contains control flow edges to represent tasks in an embedded program which are essentially a sequential execution of program statements. An *entry* node in an *EClDG* model associated with each task in the corresponding embedded program also stores the priority and the criticality information related to the task. Some of the additional features that are represented in an *EClDG* are exception handling, and information available from UML design models, such as, object states and state transitions. Regression test cases are selected by slicing the *EClDG* model. Each point of change between the original (P) and the modified (P') program acts as a slicing criterion. Identification of which model element is executed by each test case is determined by instrumenting the source code. The test cases that execute the potentially affected model elements are selected for regression testing.

Class of RTS Techniques	References	Key Features	Merits	Demerits
Database	[105, 39]	RTS techniques need to consider database states	Willmor and Embury's technique [105] is safe	Proposed techniques are imprecise
Web applications	[86, 93, 61, 110, 85]	Analysis cannot rely on the availability of the source code of web services	Techniques proposed in [93, 86, 85, 61] are safe, system model-based approach [93] is more efficient than [86, 85, 61]	Techniques can be imprecise, and depend on metacontent information
AspectJ	[114, 109]	Needs to take into account the dependencies that arise due to *pointcut*, *join* points, etc.	Technique reported in [109] is safe	Control flow-based techniques may be computationally expensive, cannot be directly adapted for higher-level analysis

Table 4: A comparison of RTS techniques proposed for database, web, and aspectj programs.

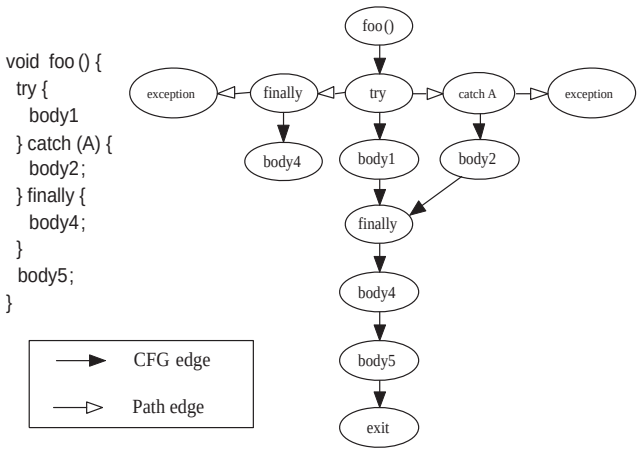


Figure 11: Modeling exceptions in the RTS technique proposed in [51].

10 Other RTS Techniques

In this section, we discuss a few RTS techniques proposed for BPEL programs [59, 63, 101] and programs developed in .Net framework [51]

10.1 RTS Technique for .Net Programs

In recent times, many virtual machine environments have been proposed such as Java and Microsoft .Net framework. In a virtual machine (VM) environment, the program is compiled into a platform-independent intermediate code. The advantage of such virtual machine environment is that it introduces a layer of abstraction and hides the low-level intricacies of the target architecture. The VM environment can also introduce check points to enhance performance, security, etc. of the application code. In the following, we discuss a safe RTS technique proposed for programs developed in Microsoft .Net framework.

Koju [51] have presented a safe RTS technique for programs developed in .Net framework. Since Microsoft .Net framework supports many programming languages such as Visual Basic, C++, C#, an RTS technique based on source-code analysis would require to take into account the features of all the .Net framework supported programming languages. The authors have avoided this problem by selecting regression test cases based on an analysis of the intermediate code, which is in Microsoft Intermediate Language (MSIL). Their technique is based on the graph walk-based RTS technique proposed by Harrold [41] for Java programs. However, the *JIG* model proposed in [41] cannot model .Net specific features such as *delegates*. The authors have also proposed a more efficient and precise way of analyzing the dependencies introduced due to class hierarchies and exceptions compared to [41]. The improved analysis of class hierarchies is applied to model method calls from code internal and external to the application under test. The modeling of exceptions is improved by representing the `catch` and `finally` block on the opposite sides of a `try` block. An example of the exception modeling technique proposed by Koju et al. [51] is shown in Figure 11. The figure shows a partial *JIG* modeling the exception handling code shown in Figure 11.

The important steps in the RTS technique proposed in [51] are as follows:

- Construct the extended *JIG* models corresponding to the MSIL code for the original and the modified programs.
- Instrument the original source program and execute the instrumented program with the initial test suite to generate the test coverage information.
- Traverse the two extended *JIG* models to identify dangerous edges. The test cases executing the dangerous edges are selected for regression testing.

This technique is safe for RTS based on MSIL languages, and is more precise than [41] because of the improved representation for exception handling and the dependencies arising due to class hierarchies. In spite of our best efforts, this is the only technique that we could find in the literature for RTS in a virtual machine framework.

10.2 RTS Techniques for BPEL Programs

We have pointed out in subsection 2.3 that SOA-based development is increasingly being adopted in different services industries. Business process execution language

(BPEL) is a part of the SOA standards, and is popularly being used to develop business process and composite services. Composite services in BPEL are composed of a process, an interface described in WSDL, and component services that interact with the process. The component services can be elementary services or composed of other elementary services. Modifications to a BPEL composite service can take place due to different reasons such as modifications to the process or the interface, replacement of a service with another service, etc. Whenever a BPEL composite service is modified, it becomes necessary to select regression test cases to test the unmodified parts of the program. In the following, we briefly discuss the control flow analysis-based RTS techniques proposed by Li et al. [59, 63, 101].

A BPEL flow graph [113] can only capture the control flow relations in a BPEL process. Therefore, it cannot be used to model BPEL composite services. In view of this, Li et al. [59] have proposed an Xtended BPEL Flow Graph (*XBFG*) to model BPEL composite services. Along with the business process, an *XBFG* is also able to model composite services and the message interactions between the process and the composite services. The technique constructs *XBFG* models for both the original and the modified BPEL composite services. The types of changes possible between two BPEL composite services are assumed to be: process change, binding change, change in the path conditions, and interface change. The technique then compares the test paths between the two *XBFG*s to find out the model elements influenced by the process and the binding changes. The paths in the *XBFG* models which are affected due to the changes are identified, and relevant regression test cases are selected to test the affected paths.

11 Conclusion and Future Research Directions

It is acknowledged that RTS techniques which analyze modifications at a finer level of granularity (e.g., program statements) are more precise than techniques which perform analysis at a comparatively higher level of abstraction (e.g., design models). Rothermel and Harrold have shown that the problem of designing precise RTS techniques is PSPACE-hard [80]. Moreover, the extensive computations for a fine-grained analysis (e.g., graph walk-based techniques for procedural/object-oriented programs) make these techniques more expensive, less efficient, and less scalable compared to the coarse-grained approaches. This is an important trade-off that needs to be considered while selecting a suitable RTS technique. After all, selection of an RTS technique makes sense only if the cost of test selection is less than the difference in cost between running the entire test suite and the selected test suite [57].

Modern commercial software products are becoming increasingly large and complex, and are usually tested using thousands of test cases. Therefore, to obtain further savings in regression testing effort, researchers need to consider the following issues:

- With the trend of increasing application size, an RTS technique should scale to very large programs having code sizes of the order of millions of KLOC. For modern large software systems, scalability is an important issue. Therefore, an interesting direction of research would be to investigate compositional and summary-based approaches to RTS.
- The RTS technique should take into account all possible relationships depending on the targeted class of programs while selecting test cases, i.e., it should be a safe technique for that class of programs.

Model-based regression testing: In view of the fact that static analysis of large software systems is computationally expensive, model-based RTS techniques appear to be a promising approach that not only scales well, but is more efficient [112]. Furthermore, of late MDD has been receiving a lot of attention. In MDD, there exists a close relationship between the design model(s) and code in the sense that any change to the model gets reflected in the code and vice versa. Therefore, instead of performing RTS on code, test selection could be automatically performed based on design models. Model-based RTS can also help to take into consideration several aspects of program behavior (like state transitions, message paths, task criticality, etc.) that are not easily identified from static code analysis.

Improved RTS tool support: In future, the reported work on RTS should gradually shift from theoretical research to tool implementations. It has been pointed out in several studies [35, 36, 112] that the current tool support for automated RTS is rather poor. Therefore, concerted effort should be directed towards developing integrated RTS tools using capture-and-replay mechanisms.

Synthesized regression testing techniques: Most of the RTS techniques reported in the literature are either code-based or model-based. Since both these approaches have their own unique advantages, these approaches can possibly be meaningfully synthesized and this issue deserves further investigation. For example, the analysis performed in a code-based RTS technique can be made more effective by using the information available from the UML design models, SRS documents, etc.

Yoo and Harman [112] have pointed out that real-world regression testing needs to select test cases keeping in mind multiple objectives such as, the number of test cases executed, cost involved in testing, code coverage achieved, time available for testing, etc. However, most of the reported work on multi-objective regression testing is in the fields of test suite minimization and prioritization [111, 99]. An interesting avenue of research could be to merge regression test selection techniques with either minimization or prioritization approaches. In such a synthesized approach,

the regression test suite first selected by a structural RTS technique can then be further minimized/prioritized. Regression testing using such a synthesized approach can help take into account multiple objectives during testing, and can potentially help achieve further savings in regression test effort without compromising the thoroughness of testing.

RTS techniques for other domains: Increased usage of real-time embedded products in safety-critical applications has resulted in greater emphasis being placed on the quality of the code. The high costs and complexities involved in carrying out regression testing of these products act as an added incentive for developing improved RTS techniques for these programs. However, not much research work has so far been reported on investigations into effective RTS for embedded, real-time and safety-critical software, though it appears to be a promising avenue for research.

For discrete control applications, industry practitioner's usually use UML models whereas for hybrid control applications, MATLAB Simulink/Stateflow models [94] are popular. In this context, suitable RTS techniques are needed for hybrid control applications and reactive software.

Moreover, as pointed out by Yoo and Harman [112], more detailed investigation is required to study the effectiveness of RTS techniques for testing non-functional requirements.

References

[1] K. Abdullah and L.White. A firewall approach for the regression testing of object-oriented software. In *Proceedings of 10th Annual Software Quality Week*, page 27, May 1997.

[2] H. Agrawal, J. Horgan, E. Krauser, and S. London. Incremental regression testing. In *IEEE International Conference on Software Maintenance*, pages 348–357, 1993.

[3] A. Aho, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques and Tools*. Dorling Kindersley (India) Pvt Ltd, 2nd edition, 2008.

[4] A. Ali, A. Nadeem, Z. Iqbal, and M. Usman. Regression testing based on UML design models. In *Proceedings of the 13th Pacific Rim International Symposium on Dependable Computing*, pages 85–88, 2007.

[5] T. Ball. On the limit of control flow analysis for regression test selection. In *ISSTA '98: Proceedings of the 1998 ACM SIGSOFT international symposium on Software testing and analysis*, pages 134–142, 1998.

[6] G. Baradhi and N. Mansour. A comparative study of five regression testing algorithms. In *Proceedings of Australian Software Engineering Conference, Sydney*, pages 174–182, 1997.

[7] S. Bates and S. Horwitz. Incremental program testing using program dependence graphs. In *Conference Record of 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 384–396, January 1993.

[8] J. Bible, G. Rothermel, and D. Rosenblum. A comparative study of coarse- and fine-grained safe regression test-selection techniques. *ACM Transactions on Software Engineering and Methodology*, 10(2):149–183, April 2001.

[9] R. Binder. *Testing Object-Oriented Systems:Models, Patterns, and Tools*. Addison-Wesley, 1999.

[10] D. Binkley. Semantics guided regression test cost reduction. *IEEE Transactions on Software Engineering*, 23(8):498–516, August 1997.

[11] S. Biswas, R. Mall, M. Satpathy, and S. Sukumaran. A model-based regression test selection approach for embedded applications. *ACM SIGSOFT Software Engineering Notes*, 34(4):1–9, July 2009.

[12] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley, 2nd edition, 2005.

[13] Mustafa Bozkurt, Mark Harman, and Youssef Hassoun. Testing web services: A survey. Technical Report TR-10-01, Kings College London, 2010.

[14] L. Briand, Y. Labiche, and S. He. Automating regression test selection based on UML designs. *Information and Software Technology*, 51(1):16–30, January 2009.

[15] L. Briand, Y. Labiche, and G. Soccar. Automating impact analysis and regression test selection based on UML designs. In *Proceedings of the International Conference on Software Maintenance (ICSM'02)*, pages 252–261, 2002.

[16] Y. Chen, R. Probert, and D. Sims. Specification-based regression test selection with risk analysis. In *CASCON '02: Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research*, page 1, 2002.

[17] Y. Chen, D. Rosenblum, and K. Vo. TestTube: A system for selective regression testing. In *Proceedings of the 16th International Conference on Software Engineering*, pages 211–222, May 1994.

[18] P. Chittimalli and M. Harrold. Regression test selection on system requirements. In *ISEC '08: Proceedings of the 1st conference on India software engineering conference*, pages 87–96, 2008.

[19] A. Cleve, J. Henrard, and J. Hainaut. Data reverse engineering using system dependency graphs. In *Proceedings of the 13th Working Conference on Reverse Engineering*, pages 157–166, 2006.

[20] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[21] J. Dean, D. Grove, and C. Chambers. Optimization of object-oriented programs using static class hierarchy analysis. In *Lecture Notes in Computer Science*, volume 952, pages 77–101. Springer-Verlag, 1995.

[22] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel. The effects of time constraints on test case prioritization: A series of controlled experiments. *IEEE Transactions on Software Engineering*, 36(5):593–617, September 2010.

[23] S. Elbaum, A.Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. *IEEE Transactions of Software Engineering*, 28(2):159–182, February 2002.

[24] E. Engström, P. Runeson, and M. Skoglund. A systematic review on regression test selection techniques. *Information and Software Technology*, 52(1):14–30, January 2010.

[25] E. Engström, M. Skoglund, and P. Runeson. Empirical evaluations of regression test selection techniques: a systematic review. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 22–31, 2008.

[26] M. Fahad and A. Nadeem. A survey of uml based regression testing. In Zhongzhi Shi, E. Mercier-Laurent, and D. Leake, editors, *Intelligent Information Processing IV*, volume 288 of *IFIP Advances in Information and Communication Technology*, pages 200–210. Springer Boston, 2008.

[27] Q. Farooq, M. Iqbal, Z. Malik, and A. Nadeem. An approach for selective state machine based regression testing. In *Proceedings of the 3rd international workshop on Advances in model-based testing*, A-MOST '07, pages 44–52. ACM, 2007.

[28] Q. Farooq, M. Iqbal, Z. Malik, and M. Riebisch. A model-based regression testing approach for evolving software systems with flexible tool support. In *17th IEEE International Conference on Engineering of Computer-Based Systems (ECBS)*, pages 41–49. IEEE Computer Society, March 2010.

[29] J. Ferrante, K. Ottenstein, and J. Warren. The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems*, 9(3):319–349, July 1987.

[30] P. Frankl, G. Rothermel, K. Sayre, and F. Vokolos. An empirical comparison of two safe regression test selection techniques. In *ISESE '03 Proceedings of the 2003 International Symposium on Empirical Software Engineering*, pages 195–204. IEEE Computer Society, 2003.

[31] J. Gao, D. Gopinathan, Q. Mai, and J. He. A systematic regression testing method and tool for software components. In *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, pages 455–466, 2006.

[32] V. Garousi, L. Briand, and Y. Labiche. *Model Driven Architecture - Foundations and Applications*, volume 3748 of *Lecture Notes in Computer Science*, chapter Control Flow Analysis of UML 2.0 Sequence Diagrams, pages 160–174. Springer Berlin / Heidelberg, October 2005.

[33] R. Gorthi, A. Pasala, K. Chanduka, and B. Leong. Specification-based approach to select regression test suite to validate changed software. In *Proceedings of the 2008 15th Asia-Pacific Software Engineering Conference*, pages 153–160, 2008.

[34] T. Graves, M. Harrold, J. Kim, A. Porter, and G. Rothermel. An empirical study of regression test selection techniques. *ACM Transactions on Software Engineering and Methodology*, 10(2):184–208, April 2001.

[35] M. Grindal, J. Offutt, and J. Mellin. On the testing maturity of software producing organizations. In *TAIC-PART '06: Proceedings of the Testing: Academic & Industrial Conference on Practice And Research Techniques*, pages 171–180, 2006.

[36] J. Guan, J. Offutt, and P. Ammann. An industrial case study of structural testing applied to safety-critical embedded software. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 272–277, 2006.

[37] R. Gupta, M. Harrold, and M. Soffa. Program slicing-based regression testing techniques. *Journal of Software Testing, Verification, and Reliability*, 6(2):83–112, June 1996.

[38] F. Haftman, D. Kossmann, and E. Lo. A framework for efficient regression tests on database applications. *The VLDB Journal*, 16(1):145–164, January 2007.

[39] R. Haraty, N. Mansour, and B. Daou. *Advanced Topics in Database Research*, volume 3, chapter Regression test selection for database applications, pages 141–165. Idea Group, 2004.

[40] M. Harrold, R. Gupta, and M. Soffa. A methodology for controlling the size of a test suite. *ACM Transactions on Software Engineering and Methodology*, 2(3):270–285, July 1993.

[41] M. Harrold, J. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi. Regression test selection for Java software. In *Proceedings of the 16th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications*, pages 312–326, January 2001.

[42] M. Harrold and G. Rothermel. Performing data flow testing on classes. In *Proceedings of the 2nd ACM SIGSOFT symposium on Foundations of software engineering*, pages 154–163, 1994.

[43] M. Harrold and M. Soffa. An incremental approach to unit testing during maintenance. In *Proceedings of the International Conference on Software Maintenance*, pages 362–367, October 1988.

[44] M. Harrold and M. Soffa. Interprocedural data flow testing. In *Proceedings of the ACM SIGSOFT '89 third symposium on Software testing, analysis, and verification*, pages 158–167, December 1989.

[45] D. Hatley and I. Pirbhai. *Strategies for Real-Time System Specification*. Dorset House Publishing Company, 1987.

[46] S. Horwitz, T. Reps, and D. Binkley. Interprocedural slicing using dependence graphs. *ACM Transactions on Programming Languages and Systems*, 12(1):26–61, January 1990.

[47] P. Hsia, X. Li, D. Kung, C. Hsu, L. Li, Y. Toyoshima, and C. Chen. A technique for the selective revalidation of object-oriented software. *Journal of Software Maintenance: Research and Practice*, 9(4):217–233, 1997.

[48] Y. Jang, M. Munro, and Y. Kwon. An improved method of selecting regression tests for C++ programs. *Journal of Software Maintenance: Research and Practice*, 13(5):331–350, September 2001.

[49] G. Kapfhammer. *The Computer Science Handbook*, chapter on Software testing. CRC Press, Boca Raton, FL, 2nd edition, 2004.

[50] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. Griswold. An overview of AspectJ. In *Proceedings of the 15th European Conference on Object-Oriented Programming*, ECOOP '01, pages 327–353. Springer-Verlag, 2001.

[51] T. Koju, S. Takada, and N. Doi. Regression test selection based on intermediate code for virtual machines. In *Proceedings of the International Conference on Software Maintenance*, ICSM '03, page 420. IEEE Computer Society, September 2003.

[52] J. Korpi and J. Koskinen. *Advances and Innovations in Systems, Computing Sciences and Software Engineering*, chapter Supporting Impact Analysis by Program Dependence Graph Based Forward Slicing, pages 197–202. Springer Netherlands, 2007.

[53] D. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, and C. Chen. On regression testing of object-oriented programs. *Journal of Systems and Software*, 32(1):21–40, January 1996.

[54] J. Laski and W. Szermer. Identification of program modifications and its applications in software maintenance. In *Proceedings of the Conference on Software Maintenance*, pages 282–290, November 1992.

[55] H. Leung and L. White. Insights into regression testing. In *Proceedings of the Conference on Software Maintenance*, pages 60–69, 1989.

[56] H. Leung and L. White. A study of integration testing and software regression at the integration level. In *Proceedings of the Conference on Software Maintenance*, pages 290–300, November 1990.

[57] H. Leung and L. White. A cost model to compare regression test strategies. In *Proceedings of the Conference on Software Maintenance*, pages 201–208, 1991.

[58] H. Leung and L. White. A firewall concept for both control-flow and data-flow in regression integration testing. In *Proceedings of the Conference on Software Maintenance*, pages 262–270, 1992.

[59] B. Li, D. Qiu, S. Ji, and D. Wang. Automatic test case selection and generation for regression testing of composite service based on extensible BPEL flow graph. In *26th IEEE International Conference on Software Maintenance, ICSM 2010*, pages 1–10. IEEE Computer Society, 2010.

[60] D. Liang and M. Harrold. Slicing objects using system dependence graphs. In *Proceedings of the International Conference on Software Maintenance*, pages 358–367, November 1998.

[61] Feng Lin, Michael Ruth, and Shengru Tu. Applying safe regression test selection techniques to Java web services. In *International Conference on Next Generation Web Services Practices, 2006. NWeSP 2006.*, pages 133–142, Los Alamitos, CA, USA, September 2006. IEEE Computer Society.

[62] J. Lin, C. Huang, and C. Lin. Test suite reduction analysis with enhanced tie-breaking techniques. In *4th IEEE International Conference on Management of Innovation and Technology, 2008. ICMIT 2008.*, pages 1228–1233, September 2008.

[63] H. Liu, Z. Li, J. Zhu, and H. Tan. Business process regression testing. In *Proceedings of the 5th international conference on Service-Oriented Computing*, ICSOC '07, pages 157–168. Springer-Verlag, 2007.

[64] N. Mansour and K. El-Fakih. Simulated annealing and genetic algorithms for optimal regression testing. *Journal of Software Maintenance: Research and Practice*, 11(1):19–34, 1999.

[65] N. Mansour and W. Statieh. Regression test selection for C# programs. *Advances in Software Engineering*, 2009:1:1–1:16, January 2009.

[66] C. Mao and Y. Lu. Regression testing for component-based software systems by enhancing change information. In *APSEC '05: Proceedings of the 12th Asia-Pacific Software Engineering Conference*, pages 611–618. IEEE Computer Society, December 2005.

[67] C. Mao, Y. Lu, and J. Zhang. Regression testing for component-based software via built-in test design. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 1416–1421, 2007.

[68] J. McGregor and D. Sykes. *A Practical Guide to Testing Object-Oriented Software*. Addison-Wesley, March 2001.

[69] L. Naslavsky and D. Richardson. Using traceability to support model-based regression testing. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, ASE '07, pages 567–570. ACM, November 2007.

[70] M. Netkow and D. Brylow. Xest: an automated framework for regression testing of embedded software. In *Proceedings of the 2010 Workshop on Embedded Systems Education*, WESE '10, pages 7:1–7:8. ACM, October 2010.

[71] A. Orso, M. Harrold, and D. Rosenblum. Component metadata for software engineering tasks. In *Revised Papers from the Second International Workshop on Engineering Distributed Objects*, EDO '00, pages 129–144. Springer-Verlag, November 2000.

[72] A. Orso, M. Harrold, D. Rosenblum, G. Rothermel, M. Soffa, and H. Do. Using component metacontent to support the regression testing of component-based software. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01)*, pages 716–725, 2001.

[73] A. Orso, N. Shi, and M. Harrold. Scaling regression testing to large software systems. In *Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering*, pages 241–251, November 2004.

[74] A. Pasala, Y Fung, F. Akladios, A. Raju, and R. Gorthi. Selection of regression test suite to validate software applications upon deployment of upgrades. In *19th Australian Conference on Software Engineering*, pages 130–138, March 2008.

[75] R. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, New York, 2002.

[76] G. Rothermel and M. Harrold. A safe, efficient algorithm for regression test selection. In *Proceedings of the Conference on Software Maintenance*, pages 358–367, 1993.

[77] G. Rothermel and M. Harrold. Selecting regression tests for object-oriented software. In *International Conference on Software Maintenance*, pages 14–25, March 1994.

[78] G. Rothermel and M. Harrold. Selecting tests and identifying test coverage requirements for modified software. In *Proceedings of the International Symposium on Software Testing and Analysis*, pages 169–184, August 1994.

[79] G. Rothermel and M. Harrold. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, 22(8):529–551, August 1996.

[80] G. Rothermel and M. Harrold. A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology*, 6(2):173–210, April 1997.

[81] G. Rothermel and M. Harrold. Empirical studies of a safe regression test selection technique. *IEEE Transactions on Software Engineering*, 24(6):401–419, June 1998.

[82] G. Rothermel, M. Harrold, and J. Dedhia. Regression test selection for C++ software. *Software Testing, Verification and Reliability*, 10(2):77–109, June 2000.

[83] G. Rothermel, M. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In *Proceedings of the International Conference on Software Maintenance*, pages 34–43, November 1998.

[84] G. Rothermel, R. Untch, C. Chu, and M. Harrold. Prioritizing test cases for regression testing. *IEEE Transactions on Software Engineering*, 27(10):929–948, October 2001.

[85] M. Ruth, S. Oh, A. Loup, B. Horton, O. Gallet, M. Mata, and S. Tu. Towards automatic regression test selection for web services. In *Proceedings of the 31st Annual International Computer Software and Applications Conference - Volume 02*, COMPSAC '07, pages 729–736. IEEE Computer Society, 2007.

[86] M. Ruth and S. Tu. A safe regression test selection technique for web services. In *Proceedings of the Second International Conference on Internet and Web Applications and Services*, pages 47–. IEEE Computer Society, 2007.

[87] A. Sajeev and B. Wibowo. Regression test selection based on version changes of components. In *APSEC '03: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference*, APSEC '03, pages 78–. IEEE Computer Society, 2003.

[88] A. Sangiovanni-Vincentelli and M. Di Natale. Embedded system design for automotive applications. *Computer*, 40(10):42–51, October 2007.

[89] S. Sinha, M. Harrold, and G. Rothermel. System-dependence-graph-based slicing of programs with arbitrary interprocedural control flow. In *Proceedings of the 21st International Conference on Software Engineering*, pages 432–441, 1999.

[90] M. Skoglund and P. Runeson. A case study of the class firewall regression test selection technique on a large scale distributed software system. In *International Symposium on Empirical Software Engineering*, pages 74–83, November 2005.

[91] D. Sundmark, A. Pettersson, and H. Thane. Regression testing of multi-tasking real-time systems: A problem statement. *ACM SIGBED Review*, 2(2):31–34, April 2005.

[92] A. Taha, S. Thebaut, and S. Liu. An approach to software fault localization and revalidation based on incremental data flow analysis. In *Proceedings of the 13th Annual International Computer Software and Applications Conference*, pages 527–534, September 1989.

[93] A. Tarhini, H. Fouchal, and N. Mansour. Regression testing web services-based applications. In *AICCSA '06 Proceedings of the IEEE International Conference on Computer Systems and Applications*, pages 163–170. IEEE Computer Society, 2006.

[94] The Mathworks, Inc. MATLAB. Website, April 2011. `http://www.mathworks.com`.

[95] F. Tip. A survey of program slicing techniques. *Journal of Programming Languages*, 3(3):121–189, September 1995.

[96] F. Vokolos. *A regression test selection technique based on textual differencing*. PhD thesis, Polytechnic University, 1998. UMI Order No. GAX98-10583.

[97] F. Vokolos and P. Frankl. Pythia: A regression test selection tool based on textual differencing. In *Proceedings of the 3rd International Conference on Reliability, Quality & Safety of Software-Intensive Systems (ENCRESS' 97)*, pages 3–21, May 1997.

[98] F. Vokolos and P. Frankl. Empirical evaluation of the textual differencing regression testing technique. In *ICSM '98: Proceedings of the International Conference on Software Maintenance*, pages 44–53, 1998.

[99] K. Walcott, M. Soffa, G. Kapfhammer, and R. Roos. Time aware test suite prioritization. In *Proceedings of the 2006 International Symposium on Software Testing and Analysis*, pages 1–12, 2006.

[100] N. Walkinshaw, M. Roper, and M. Wood. The Java system dependence graph. In *Third IEEE International Workshop on Source Code Analysis and Manipulation*, pages 55–64, September 2003.

[101] D. Wang, B. Li, and J. Cai. Regression testing of composite service: An XBFG-based approach. In *Proceedings of the 2008 IEEE Congress on Services Part II*, pages 112–119. IEEE Computer Society, 2008.

[102] P. Ward and S. Mellor. *Structured Development for Real-Time Systems*. Prentice Hall Professional Technical Reference, 1991.

[103] M. Weiser. Program slicing. In *ICSE '81: Proceedings of the 5th international conference on Software engineering*, pages 439–449, 1981.

[104] N. Wilde and R. Huitt. Maintenance support for object-oriented programs. *IEEE Transactions on Software Engineering*, 18(12):1038–1044, December 1992.

[105] D. Willmor and S. Embury. A safe regression test selection technique for database-driven applications. In *Proceedings of the 21st IEEE International Conference on Software Maintenance*, pages 421–430. IEEE Computer Society, 2005.

[106] W. Wong, J. Horgan, S. London, and A. Mathur. A study of effective regression testing in practice. In *Proceedings of the Eighth International Symposium on Software Reliability Engineering*, pages 230–238, November 1997.

[107] Y. Wu and J. Offutt. Maintaining evolving component-based software with UML. In *Proceedings of 7th European Conference on Software Maintenance and Reengineering (CSMR '03)*, pages 133–142, March 2003.

[108] B. Xu, J. Qian, X. Zhang, Z. Wu, and L. Chen. A brief survey of program slicing. *ACM SIGSOFT Software Engineering Notes*, 30(2):1–36, March 2005.

[109] G. Xu and A. Rountev. Regression test selection for AspectJ software. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, pages 65–74, 2007.

[110] Lei Xu, Baowen Xu, Zhenqiang Chen, Jixiang Jiang, and Huowang Chen. Regression testing for web applications based on slicing. In *Proceedings of the 27th Annual International Computer Software and Applications Conference, 2003. COMPSAC 2003.*, pages 652–656, Los Alamitos, CA, USA, November 2003. IEEE Computer Society.

[111] S. Yoo and M. Harman. Pareto efficient multi-objective test case selection. In *Proceedings of the 2007 International Symposium on Software Testing and Analysis*, pages 140–150, 2007.

[112] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, 1(1):121–141, March 2010.

[113] Y. Yuan, Z. Li, and W. Sun. A graph-search based approach to BPEL4WS test generation. In *Proceedings of the International Conference on Software Engineering Advances (ICSEA' 06)*, pages 14–. IEEE Computer Society, October 2006.

[114] J. Zhao, T. Xie, and N. Li. Towards regression test selection for AspectJ programs. In *Proceedings of the 2nd workshop on Testing aspect-oriented programs*, WTAOP '06, pages 21–26. ACM, 2006.

[115] J. Zheng, B. Robinson, L. Williams, and K. Smiley. An initial study of a lightweight process for change identification and regression test selection when source code is not available. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, pages 225–234, November 2005.

[116] J. Zheng, B. Robinson, L. Williams, and K. Smiley. Applying regression test selection for COTS-based applications. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 512–522, May 2006.

[117] J. Zheng, B. Robinson, L. Williams, and K. Smiley. A lightweight process for change identification and regression test selection in using COTS components. In *ICCBSS '06: Proceedings of the Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems*, pages 137–143, February 2006.

[118] F. Zhu, S. Rayadurgam, and W. Tsai. Automating regression testing for real-time software in a distributed environment. In *ISORC '98: Proceedings of the The 1st IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, page 373, 1998.

Distributed Multi-ant Algorithm for Capacity Vehicle Route Problem

Jie Li, Yi Chai and Cao Yuan
Chongqing University
E-mail: leighby16@gmail.com, cqchaiyi@gmail.com, 122269587@qq.com

Keywords: capacity vehicle route problem, distributed multi-ant algorithm, cellular ants, performance potential

Received: November 2, 2010

This paper proposes a Distributed Multi-ant Algorithm for capacity vehicle route problem (CVRP) where cooperation is helpful for accelerating prior solution by executing a decomposition-based separation methodology for the unsteady capacity constraints. It decreases the complex coupling network with others to solve small instances with less correlation in parallel processing. The main goal of this work is to play well on large scale CVRP with interaction between subsystems and certain state vectors. The results show that Distributed Multi-ant Algorithm plays better performance on average solution and the importance of potential action is analyzed.

Povzetek: Za problem CVRP je razvit izboljšan postopek, temelječ na algoritmih z mravljami.

1 Introduction

With decomposition, every subsystem could be described as a Markov Decision Process (MDP) model: Let $M = \{S, A, T, R, fv\}$ be a five tuple model, where $S = \{s\}$ is a set of states, $A = \{a\}$ is a set of actions, $T = \{p(\cdot|s,a), s \in S, a \in A\}$ is the next-state transition probability distribution, with $p(\cdot \mid s,a)$ describing the probability of action a in state s to s'. $R(s,a,s')$ is the reward function. fv is the additional reward function. $\pi(s)$ is a policy function in the state space. The discussion about applicability and feasibility is based on discount value MDP and Q-learning.

Algorithms works on CVRP are researched for years. Edge assembly (EAX) crossover with well-known local searches is employed to CVRP (Yuichi Nagata et al, 2009). Deoxyribonucleic acid (DNA) computing model and a modified Adleman-Lipton model accelerate the search on large nodes CVRP (Yeh Chung Wei, 2009). Ellipse rule approach reduces the average distance to the lower bound by about 44% (Santos Luis et al, 2009). A multi-objective evolutionary algorithm is used for CVRP (Borgulya Istvan, 2008). Particle swarm optimization (PSO) can also apply for CVRP, in two models (Ai The Jin et al, 2009). Cellular GAs has solved vehicle routing problem, minimized transportation cost and recombined a new problem (Carlos Bermudez et al, 2010) and genetic algorithm has worked at it fewer than 100 nodes (Wang Chungho et al, 2010).

Normally, it is solved based on decentralized model. However, coupling among subsystems (called SCVRP in the paper) is not considered well and being trap into a local solution or no solution easily, deviating from what we expected. Multi-ant algorithm is extension of ant algorithm who plays a better performance in the best solution and used to VRP already (Yuvraj Gajpal et al, 2009). The ant isn't punished if the strategy misleads it to suboptimal policies. And if there isn't new knowledge during state s, the reward function is still working accumulation. In this paper, decomposition of CVRP based on a distributed model is presented with iteration and cooperation between subsystems searching for their own optimization by distributed multi-ant algorithm. As cooperation, a cellular ant contacts with other ants both in its own subsystem and others through reward strategies obtained whenever the related strategy is optimal by traditional relative reinforcement learning. Reward shaping undergoes through structuring additional reward function for distributed multi-ant algorithm, making it more efficient.

As the large scale of CVRP is divided into smaller SCVRPs in a distributed system where bunches of cellular ants set to seek an optimal solution of corresponding subgraph for SCVRP. On the one hand, cellular ants in the same subsystem collaborate with each other and refresh their own knowledge. On the other hand, cellular ants in different SCVRP regenerate strategies as they meet in the crossed arcs over several disparate SCVRPs.

The paper is organized as follows: Section 2 will introduce some basic knowledge of tree cutest and decomposition of CVRP for distributed system using tree cut-set. Section 3 describes how those SCVRPs figuring out the optimal solution respectively based on distributed multi-ant algorithm with reward shaping. The procedure of cooperation between subsystems is gotten. Then, the experimental results in several algorithms are contrasted using ArcView, matlab R2009a and GAMS softwares in Section 4. Finally, the discussion and conclusions is in Section 5, as well as directions for future work.

2 Decomposition algorithms for CVRP

For the limited capacity constraint, a large scale of CVRP is decomposed into SCVRPs with unsteady capacity constraints in the distributed system. The mathematical model of CVRP is transformed through $Tree\ Description$ (Chen Yulin et al, 2002). An algorithm named Tree Cut-Set (TCS) is proposed for decomposition. The number of subsystems is determined by the carrying capability of vehicles, demands of customers and connection between customers.

A large scale of undirected graph can be transformed into complete trees and split into subgraphs where leaves of subgraphs are the compound boundaries. Then, the semantic representation of CVRP could be replaced by SCVRPs. Based on these definitions in (Y Xiang, 1996), calculation must comply for some principle as follows:

(1) $a * b = ab$
(2) $(a * b)' = ab' + a'b = a + b$
(3) $a\Lambda b = a'b + ab' = a + b$
(4) $(a\Lambda b)' = (ab)' = 1$

In ripping, two subgraphs are brought out and the summation of their semantic representation is equat to the original graph. We can call the original graph the father graph of the two subgraphs. It requires the knowledge of the interface between SCVRPs only, not the knowledge of the internal structure of them. TCS is applied to closed the structural details to separate the model. In the result, every SCVRP could obtain its own solution through an independent set of ants and the link between them is loosely coupled by employing TCS. Whatever the structure CVRP is, TCS is suitable for it, whether customers may join and leave or not.

We will explain the theory of $t-sepset$ couple based on $d-sepset$ (Li Yan et al, 2004).

Definition 1 In a tree, for tree-node i, there is only two parents and one child node j. For tree-node j, there is only two different children. Then, (i, j) is called as a t-sepset couple nodes.

Searching the tree created from Definition 1 from top to button by Greedy policy, t-sepset couple nodes for customers with uncertain demands are ripped. As what Bayesian Theory (Andrew Y Ng et al, 1999) says, every separate demand is weighted by w_i a random variable that must satisfy some limitation: $w > 0$ and $\sum_{i=1}^{2} w_i = 1$.

For the limited carrying capability of vehicles b, the total demands t_a in each subsystem must be lower than b. The purpose is that the parameter $e = t_a - b$ is gradually close to zero. If every SCVRP meets this condition, the problem would be solved perfectly. The procedure would stop till e is lower than a by constant p. Otherwise, TCS will continue.

3 Distributed Multi-ant Algorithm

3.1 Reward shaping

(Andrew Y Ng et al, 1999) presents a method that if a potential function $\Phi(s)$ exists so that $R'(s, a, s') = R(s, a, s') + \Phi(s') - \Phi(s)$ for any policy $\pi(s)$, $V^*(s) = \pi(s) - \Phi(s)$. Reward shaping causes the optimal policies in M would be the optimal policies in M', to exchange the original reward function R of $MDP\ M$ with new reward function R' of new $MDP\ M'$. In this paper, we definite the R' in M' :

$$R'(s, a, s') = R(s, a, s') + fv(s, a, s') \quad (1)$$

Where $fv(s, a, s')$ is a function, carrying ant colony information. The proof of this equation is in next chapter.

3.2 Cellular ant

Cellular automata (CA) is a discrete grid dynamics model both in time, space and state vector. It is utilized to imitate complex and abundant macroscopic phenomenon in single regulations of parallel evolutionary. The distributed cellular in grid net of $SCVRP$ has finite discrete state, following the same action regulations to update its state by local rules synchronously.

As equations in reference (Moere Andrew Vande et al, 2005), the dynamic evolutionary of CAs is: $F(S_t^i + 1) = f(s_t^{i-r}, \ldots, s_t^i, \ldots, s_t^{i+r})$. S_t^i describes the state of cellular ant in position i at time t and the local updating rule is $f : S_t^{2r+1} \longrightarrow S_t$.

The structure of CAs contain four basic parts: cellular ants space, grid dynamics net, local rules and transfer function, discrete time set. For the speciality of $SCVRP$, $cellular\ space$ has two dimensions with uncertainty states of cellular ant. The renewed knowledge function is composed of its information at time t and its neighbors' using the extended Moore neighbor model:

$$f : s_{t+1}^i = f(s_t^i, s_t^N) \quad (2)$$

The key of CAs is to gain the strategies of certain neighbors' by the extended Moore neighbor model, then it could compare decision strategies referred to the last state in MDP of cellular ant with its neighbors. Reward function of distributed multi-ant algorithm is gotten as follows by $f = f(s_t^i(a), \sum_{j=0}^{N} s_t^j(a))$ and $fv(s, a, s') = F(s_{t+1}^i)$:

$$R'(s, a, s') = R(s, a, s') + \sum_{r=0}^{2r} f(s_t^{i+r}(a), \sum_{j=0}^{N} s_t^j(a)) \quad (3)$$

Proof of Equation 3: From Equation 1, we can see the additional reward function should be potential function which is proven (Moere Andrew Vande et al, 1999). Equation 2 gives the description of f function that includes local rules and transfer principles. The action of f function is renewing information among neighbors for best states of each

ant at the next moment. For this system, it is a discrete simple space model. Every state is a dot in the time vector, function f could be considered as the max value in each state dot. We could say f is the gradient function in the space/time vector space. Then, F is a vector field composed by a set of gradient field, in other words, F is deemed as potential field which is also called potential function.

3.3 Distributed Multi-ant Algorithm

Dynamic learning promotes the efficiency in Back Propagation (Yu Xiaohu et al, 1995). It is expected to accelerate the learning rate so that running time could cut down and final solution could be gained sooner. The value $\varphi_t(i)$ describes the learning rate of ant i at time t being gradually decreased to zero in the limit of search procedure. Let $\varphi_t(i) > 0$ be a series of constants for every ant at time t and satisfy the equation: $\lim_{T\longrightarrow\infty}\sum_{t+1}^{T}\varphi_t(i) = \infty$.

There are some destabilization in the circumstance in $SCVRP$, for example, the weather will delay the arriving time of transportation, the difference performance of the vehicles may also influent the efficiency, and so on. The influence of disturbance in the system can be measured by Performance Potential (Cao Xien et al, 1997). As the definition, it could set performance potential of a random state being benchmark for any other states. We choose the last-step state as the basis of performance potential for current-step state where it helps to make decision more accurately. Let $X = \{X_t, t = 1, 2\dots\}$ picture the decision progress of MDP. Considering the principles of reward function, the description of performance potential in state s' (s is the last-step state) is as following:

$$g_{s'} = \lim_{T\longrightarrow\infty}\{E[\rho^N\sum_{s=0}^{N-1}R'(s,a,s')|X_0 = s] - (N-1)g_s\}$$

Standard ant colony algorithm is integrated with reward shaping function. New value function and strategy function are gained based on Q-learning (Bagnell J Andrew et al, 2006; Dietterich Thomas G, 2000):

$$\pi_k^*(i,j) = argminmax\{\sum_{a\in A}\pi_a(i,j) * Q^\star\} \quad (4)$$

$$Q^\star = max_{a\neq k}Q(i,j,k) - Q(i,j,s)$$

$$Q(i,j,k) = (1-\frac{r_k}{R_k})\cdot Q - \delta\cdot g_{s'} + \frac{r_k}{R_k}\cdot V^\star$$

$$V^\star = R'(i,k,j) + \gamma V^*(s')$$

Where α is the discounted factor. As the amount of cellular ants which arrived in city i is R_k, and the amount of cellular ants which chose city j for the next city is r_k where $\varphi_t(i) = 1 - \frac{r_k}{R_k}$.

Proof of Equation 4 Convergence : An equation is proposed and proven in (Jiang Lingyan et al, 2007) as $Q_{t+1}(s,a) \longleftarrow (1-\alpha)Q_t(s,a) + \alpha[r_k + \gamma * maxQ_t(s',a')]$, according to this equation, it is easy to say $\gamma * maxQ_t(s',a')$ is bounded. Because R_k is greater than r_k, $\triangle\tau_{ij}$ is bounded in each updating, in other words, the amount of trail every iteration is bounded. By reason of η_{ij} is finite constant, then $P(i,j)$ is bounded. To sum up, $f = f(s_t^i(a), \sum_{j=0}^{N} s_t^j(a))$ should be bounded and that results in bounded $\pi(i,j)$. $|S|$ and $|A|$ are both finite sets, then $f = f(s_t^i(a), \sum_{j=0}^{N} s_t^j(a))$ is the state transition in MDP that must be finite. Therefore, $\pi(i,j)$ is bounded and finite, for every decision, there is a prior strategy $\pi(i,j)$ to leading cellular ants towards global optimal solution.

3.4 Algorithm process

For the simulation, one formula should be presented firstly. Formula 1: Transition probability P_{ij}^k of each cellular ant facing the near city is defined as following:

$$P_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha(t)\eta_{ij}^\beta(t)}{\sum_{s\in j^\star}\tau_{is}^\alpha(t)\eta_{is}^\beta(t)} & j\in j^\star \\ 0 & \text{otherwise} \end{cases}$$

Where τ_{ij} expresses the trail of arc (i,j), η_{ij} describes heuristic degree and j^* is the set of allowed cities for cellular ant j. The pseudo-codes are shown as following:

1: Put m cellular ants on n cities of a decomposed SCVRP stochastically.
2: Choose the next city j for each ant by probability P_{ij}^k when it stands in city i;
3: Calculate values of target function (F function) for each cellular ant and list the best one;
4: Search for the best performance by evolving in neighbors' as the definition from equation 2;
5: Update R' function. If latest strategy is helpful for the program, reward value is positive and it becomes advanced step, vice versa;
6: Replace g_s for $g_{\dot{s}}$ by rewards from the latest iteration to weaken influence from disturbance variables;
7: Renew $Q(i,j,k)$ function for ant k through their knowledge and others';
8: Make the prior strategy π following equation 4 and choose the next city according to strategy π;
9: End till no optimal solution figuring out.

4 Computational experiments

Computational experiments have been conducted to analyze the performance of the proposed algorithm and present the results along with comparative analyse. All these algorithms have been compared with result quality. In the experiments, corresponding parameters could be: The amount of cellular ants is 31; Parameter α is trail evaporation coefficient, if it is over certain limit, the probability of revisiting the same city could be increased. If it is lower than certain limitation, it could influence the convergence; Parameter β is the heuristic information. From (Jiang Lingyan et al, 2007), the best parameters regions are $0.1 \leq \alpha \leq 0.3$ $3 \leq \beta \leq 6$; Combining the Q-learning, the parameters are set as following: $\gamma = 0.8$, $\alpha = 0.2$, $\beta = 4$, $Q = 2$,$\rho = 0.7$.

4.1 Case 1: computational analysis on benchmark problems

For decomposition, we establish the extended codes based on GrThoery toolbox ($http : //www.mathworks.com/matlabcentral/file-exchange/4266$) in software Matlab R2009a. Some existing functions are utilized for simulations, such as grMinCutSet function, grMaxFlows function, grDecOrd function and grValidation function.

With further verification of our algorithm, standard CVRP (http://www.branchandcut.org/) is also solved by those three algorithms. For each algorithm variant, ten independent simulations are taken per benchmark. With different iteration value, the average distances are illustrated in Table 1. The convergence of distributed multi-ant algorithm (DMA) is around 100 to 150 shown in Figure 1 as the benchmark problem E-n101-k14 presented by (Christofides Nicos et al, 1979). Therefore, it is quick to find out best solutions with our algorithm. To test the determinacy, we make experiments under iteration 1000. In most benchmark problems, best solutions are almost the same as the one that under around iteration 150. To limit disturbance, iteration value of our algorithm is set as 200.

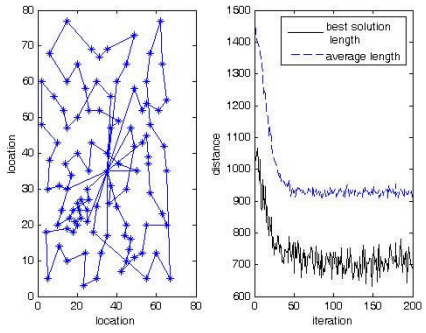


Figure 1: Computational results of E-n101-k14

In Table 2, the figures stand for best obtained fitness values (column Best solution) and average objective values of the best found feasible solutions (column Average). Compared to adaptive ant colony (AAC) and distributed ant cooperation without decomposition (DAC), DMA obtains better solutions to those 10 problems. Moreover, DMA runs less time (The unit of CPU time is second.). With incremental scale of problem, executed time increases slowly.

Table 3 illustrates the comparative results of best performances on the benchmark problem proposed by (Christofides Nicos et al,1979) according to the literature (Richard Eglese, 2009). For the experimental results of branch-and-cut algorithm and the algorithm presented by Richard Eglese, performing larger problems by exponential growing costs too much time. But the performance on DMA is stable. The fluctuation of time consume is steady growth and the quality of solution is outperformed at the large scale CVRP. By reason of decomposition, DMA plays well even on large scale problems, inducing the disadvantage of DMA that behaviors on smaller problems also take high time consume.

4.2 Case 2: computational analysis with an ArcView graph data

Based on the customers address and request, we try to set forth the location of cities by utilizing an GIS software, "ArcView", creating geographic data and transformed to network data containing latitude vector and longitude vector. The data set of original map is loading automatically in ArcView software shown as Figure 2.

We store the digital map data of autologous city in ArcView. Its information is shown through digital road map utilized in several areas and edited layer structured data of spatial objects with latitude and longitude, buildings and so on. It could be found in Figure 3 amplifying Figure 2 for details. The building with red square is the distributed center and the buildings with red triangle are part of the distributed destinations. For the readability, we scale down the latitude and longitude in the digital road map by one hundred times.

Figure 2: Digital road map of city

Table 1: Results comparison under different iterations

Benchmark problem	Iteration: 200		Iteration: 50	
	Best solution	Time	Best solution	Time
E-n22-k4	252.610	56.3131	305.696	12.8750
E-n30-k3	393.427	79.1482	463.636	18.9688
E-n33-k4	511.208	84.1334	640.079	19.8906
E-n51-k5	416.059	114.7873	525.942	34.3906
E-n76-k7	528.711	187.1651	677.091	53.2969
E-n76-k8	536.684	189.2290	668.377	57.0469
E-n76-k10	559.024	242.3379	672.265	59.3671
E-n76-k14	610.793	250.3661	784.509	61.2823
E-n101-k8	635.468	307.6808	854.114	89.3750
E-n101-k14	677.007	386.9579	757.438	87.7813

Table 2: Solutions comparison with the Christofides et al. instances

Benchmark problem	AAC		DAC		DMA	
	Best solution	Average	Best solution	Average	Best solution	Average
E-n22-k4	310.524	317.191	305.696	309.958	252.610	257.295
E-n30-k3	466.714	472.816	458.745	465.064	393.427	400.841
E-n33-k4	651.878	659.097	640.079	651.852	511.208	527.384
E-n51-k5	520.126	523.726	514.174	518.546	416.059	419.572
E-n76-k7	677.091	680.483	672.844	673.267	528.711	535.547
E-n76-k8	686.901	688.339	668.377	676.189	536.684	539.960
E-n76-k10	679.881	683.925	672.265	675.061	559.024	560.275
E-n76-k14	689.764	690.598	672.265	675.696	610.793	613.670
E-n101-k8	816.362	819.362	732.735	751.482	635.468	637.117
E-n101-k14	927.577	934.523	883.894	898.789	677.007	678.563

Figure 3: Details of digital road map

4.2.1 Results comparison

The number of places in the city is 500. In other words, the scale of CVRP is 500. Performance potential parameter δ is 1. The simulations have undergone under three algorithms: AAC, DAC and DMA. Time consume time ordered by AAC,DAC and DMA is 1.7514e+003 s | 202.8906 s | 179.5156 s. Shortest distance by the same order is 1.4263e+004 | 1.4164e+003 | 1.3541e+003. From these data, DMA gets prior performance on best solution and costs less running time. With the amount of places increasing, distributed multi-ant cooperation with decomposition will play more excellent. These graphs also display that DAC is easy to trap into local solution, even though DMA takes more iteration.

4.2.2 Performance potential analysis

With the scale of problem increasing, decentralized algorithm becomes weaker than distributed one according to its interrelate variables restrained with each others. Interaction and cooperation are critical characters of distributed model where each ant communicates through reinforcement learning and renovates its next strategy even under additional places. The experiments run from two points.

In Table 4, performance potential parameter δ is 1. The scales of CVRP are 50,100,200,300,500 and 1000. Three algorithms are executed: adaptive ant colony (AAC), distributed ant cooperation without decomposition (DAC) and DMA. The results are in table 4. Best solutions in DAC and AAC are trapping into local ones. While the scale is small, DMA costs more time than DAC. Because complex structure process in DMA needs additional operation. However, DMA shows disadvantages in a large scale one. It takes less running time and gains better solutions than DAC and AAC.

As the scale of problem is 1000, simulation runs with different δ value. Noise misleads to trivial solutions. The efficiency of $performance\,potential$ who is utilized to reduce noisy impact on the system is determined by parameter δ value. If δ is too small to zero, the influence from $performance\,potential$ can be ignored. From Table 5, because of noisy disturbance, smaller δ costs much iteration and more time for best solutions. The difference between best solution and average solution is in inverse proportion to δ value. Consequently, $performance\,potential$

Table 3: Best solution and time consume comparison with the Christofides et al. instances

Benchmark problem	Branch-and-cut algorithm		Richard Eglese's algorithm		DMA	
	Best solution	Time	Best solution	Time	Best solution	Time
E-n22-k4	375.28	0.2	252.614	0.08	252.610	56.3131
E-n30-k3	535.797	2	393.512	1	393.427	79.1482
E-n33-k4	837.672	2	511.263	0.6	511.208	84.1334
E-n51-k5	524.611	11	416.063	16	416.059	114.7873
E-n76-k7	682.563	3600	530.022	1103	528.711	187.1651
E-n76-k8	733.686	3600	537.239	636	536.684	189.2290
E-n76-k10	828.655	3600	559.233	3600	559.024	242.3379
E-n76-k14	989.257	3600	614.442	3600	610.793	250.3661
E-n101-k8	820.552	3600	639.744	2379	635.468	307.6808
E-n101-k14	1049.534	3600	699.985	3600	677.007	386.9579

Table 4: Experimental results comparison of DAC and DMA(δ=1)

amount of places	Adaptive ant colony		Decentralized algorithm		Distributed multi-ant algorithm	
	Best solution	Cost time	Best solution	Cost time	Best solution	Cost time
Amount=50	58.76	3.4875	58.69	3.8653	56.86	4.1250
Amount=100	157.4581	25.1541	111.86	20.6354	109.22	24.7343
Amount=200	451.5714	74.1572	296.26	57.1351	235.72	50.5688
Amount=300	2874.1674	558.1547	882.29	198.7326	719.18	103.2497
Amount=500	1.4263e+004	1.7514e+004	2967.53	289.5187	1354.11	179.5156
Amount=1000	4.8417e+005	7.1541e+005	9233.76	3.5763e+003	4617.43	815.9327

plays a crucial role in distributed multi-ant algorithm.

5 Conclusion

Decomposition is usually used in decentralized model to scale down the problem into subsystems we can handle with. However, the relationship between subsystems is ignored easily, leading to local solution or non-solution. In this analysis, decomposition is undergoing through hybrid algorithms for large scale of $CVRP$ in a distributed model. Cooperation and interaction are considered and solved by distributed multi-ant algorithm. Disturbance from circumstance is conquered by Potential function whose efficiency is verified from simulations. From the experiments, the algorithm has solved the large scale $CVRP$ better and efficiently. Furthermore, the next work is further control of fluctuation on solutions.

References

[1] Yuichi Nagata, Olli Braysy (2009) Edge Assembly based Memetic Algorithm for the Capacitated Vehicle Routing Problem, *Networks*, 54(4) pp. 205-215.

[2] Yeh Chung Wei (2009) Solving Capacitated Vehicle Routing Problem using DNA-based Computation, *Proceedings International Conference On Information Management and Engineering*, pp. 170-174.

[3] Santos Luis, Coutinho Rodrigues Joao, Current John R (2009) An Improved Heuristic for the Capacitated Arc Routing Problem, *Computers and Operations Research*, 36(9) pp. 2632-2637.

[4] Borgulya Istvan (2008) An Algorithm for the Capacitated Vehicle Routing Problem with Route Balancing, *Central European Journal of Operations Research*, 16(4) pp. 331-343.

[5] Ai The Jin, Kachitvichyanukul Voratas (2009) Particle Swarm Optimization and Two Solution Representations for Solving the Capacitated Vehicle Routing Problem, *Computers and Industrial Engineering*, 56(1) pp. 380-387.

[6] Carlos Bermudez, Patricia Graglia, Natalia Stark, Carolina Salto, Hugo Alfonso (2010) Comparison of Recombination Operators in Panmictic and Cellular GAs to Solve a Vehicle Routing Problem, *Inteligencia Artificial*, 14(46) pp. 34-44.

[7] Wang Chungho, Lu Jiuzhang (2010) An Effective Evolutionary Algorithm for the Practical Capacitated Vehicle Routing Problems, *J Intell Manuf*, 21(4) pp. 363-375.

[8] Gajpal Yuvraj, Abad PL (2009) Multi-ant Colony System(MACS) for a Vehicle Routing Problem with Backhauls, *European Journal of Operational Research*, 196(1) pp. 102-117.

[9] Chen Yulin, Liu Jiancheng (2002) An Tree Generation Algorithm of Undirected Graphs, *The Application of Computer Engineer*, 38(20) pp. 115-117.

[10] Y Xiang (1996) Distributed Structure Verification in Multiply Sectioned Bayesian Networks, *Florida Artificial Intelligence Research Symposium*, pp. 295-299.

Table 5: Experimental results under different δ value (places=1000)

Parameter value	Distributed ant algorithm			
	Best solution	Average solution	Cost time	Iteration
δ=0	4822.31	6728.34	3.2644e+004	545
δ=0.1	4792.93	6577.19	2.9883e+004	357
δ=0.2	4881.27	6221.69	2.3458e+004	288
δ=0.5	4632.36	5994.62	1.2336e+004	214
δ=0.8	4723.56	5938.43	899.3654	194
δ=1	4617.43	5769.27	815.9327	106

[11] Li Yan, Yin Zongmou (2004) Techniques by Compound Branch and Network Ripping to Find out All Spanning Trees of an Undirected Graph, *Journal of Naval University of Engineering*, 16(5) pp. 74-77.

[12] Andrew Y Ng, Daishi Harada, Stuart Russell (1999) Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping, *ICML 1999*.

[13] Moere Andrew Vande, Clayden Justin James (2005) Cellular Ants: Combining Ant-based Clustering with Cellular Automata, *International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 177-184.

[14] Yu Xiaohu, Chen Guoan, Cheng Shixin (1995) Dynamic Learning Rate Optimization of the Backpropagation Algorithm, *IEEE Transactions on Neural Networks*, 6(3) pp. 669-677.

[15] Cao Xien, Chen Hanfu (1997) Perturbation Realization, Potentials and Sensitivity Analysis of Markov Processes, *IEEE Transactions of Automatic Control*, 42(10) pp. 1382-1393.

[16] Bagnell J Andrew, Ng Andrew (2006) On Local Rewards and Scaling Distributed Reinforcement Learning, *Neural Information Processing Systems*.

[17] Dietterich Thomas G (2000) Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, *JAIR*.

[18] Jiang Lingyan, Zhang Jun, Zhong Shuhong (2007) Analysis of Parameters in Ant Colony System, *Computer Engineering and Applications*, Beijing, China, 40(20) pp. 31-36.

[19] Richard Eglese (2009) The Open Vehicle Routing Problem and the Disrupted Vehicle Routing Problem: a Tale of two Problems. *http:// www-eio.upc.es/seminar/09/r_eglese.pdf*.

[20] Christofides Nicos, Mingozzi Aristide, Toth Paolo (1979) The Vehicle Routing Problem-Combinatorial Optimization, *Wiley*, Chichester, pp. 315-338.

Mutant Hierarchies Support Selective Mutation

Kalpesh Kapoor
Department of Mathematics,
Indian Institute of Technology Guwahati 781 039, India.
E-mail: kalpesh@iitg.ernet.in

Keywords: fault-based testing, mutation testing, first-order mutants, selective mutation

Received: October 25, 2009

Mutation testing attempts to assess the quality of a test set by its ability to distinguish the program under test from its mutants. One of the main difficulties faced in practice is due to the large number of mutants that can be generated for a program under test. Earlier research to solve this problem has suggested variants of mutation testing, and finding an effective set of mutation operators referred to as selective mutation. This paper presents an alternative approach for reducing the cost of testing by identifying hierarchies among first-order mutants. The key idea is to evaluate the strength of a mutant with respect to other mutants and ignore "weaker" mutants during testing. Unlike previous approaches, our method is formal and it is guaranteed that the effectiveness of a test suite will be identical with that can be achieved using all mutants. The theory described here is also applicable to the quantitative assessment of testing effort and can be used to guide successive testing steps in fault-based testing. We present an empirical evaluation to find reduction in the test effort using mutant classification and show that it supports selective mutation.

Povzetek: Metodo testiranja z mutanti so izboljšali s hierarhijo mutacij, ki izloča slabše mutante.

1 Introduction

Fault-based [23] testing approach relies on generating test set that can guarantee to detect all hypothesized faults in a Program Under Test (PUT). A fault is a manifestation of an error, for example, misunderstanding about the semantics of an operator. A failure is the inability of the system or a system component to perform a function as dictated by the specification [15, 37]. In other words, a fault is locally incorrect (computational or control) operation that, when propagated, results in a failure [29, 35].

Mutation testing [4, 6] is a fault-based approach to test programs written in an imperative programming language. In mutation testing, a set of programs is generated by making a single (well-defined) syntactic change in a given PUT. This set of programs, referred to as first-order mutants, are used for evaluating a test set. A test set is a collection of test cases where each test case is a set of inputs with expected output values for a PUT.

A mutant is said to be *killed* by a test set if it can distinguish the mutant to be different from a PUT. Given a test set, its effectiveness, defined as *mutation score*, is measured by computing the percentage of first-order mutants that it killed with respect to the total number of non-equivalent mutants. Since the semantics of a PUT is not considered while generating mutants, some of them could be semantically equivalent to the PUT. Such mutants have to be identified manually or by other methods [26] and their number is reduced from the total number of mutants before computing the mutation score. In rest of the paper we refer to non-equivalent first-order mutants as simply mutants.

The two test hypotheses [11, 13] that form the basis for mutation testing are competent programmer; and coupling effect [6]. The former assumes that programmers are competent, however, they make small mistakes while writing programs. These small mistakes can be modelled as syntactic changes such as replacing $>$ by $\leq$ etc. and are referred to as mutation operators. In practice, it is expected that the set of mutation operators either represent the commonly occurring faults or they enable generation of test cases that can expose complex faults. Thus the benefits of mutation analysis depends on the mutation operators that are used to generate the mutants from a PUT.

During mutation testing only those mutants are considered that can be obtained by making single syntactic change. Those mutants that can be obtained by making multiple changes, called as higher-order mutants, in a PUT are ignored. The basis for this is the coupling effect hypothesis which states that if a test set can guarantee killing first-order mutants then it is also likely to guarantee the same for higher-order mutants. Coupling hypothesis has been investigated both theoretically [19, 36] and empirically [25] and is found to hold for several fault classes. However, there is a recent research on higher order mutants by Jia and Harman [17], which suggests that some strongly subsuming higher order mutants are in fact harder to kill than some

A preliminary version of the theoretical foundation of this paper appeared in: K. Kapoor and J. P. Bowen, Ordering mutants to minimise test effort in mutation testing, proceedings of 4th International Workshop on Formal Approaches to Testing of Software (FATES), Springer-Verlag, LNCS, Volume 3395, pages 195–209, September 2004.

first order mutants.

If a PUT gives correct results for all the test cases in a test set with 100% mutation score then it is concluded that the PUT is correct with respect to the faults represented by the mutation operators. Thus such a test set is good at distinguishing a program from its mutants and, if the program is faulty, the test set is also likely to be good at distinguishing the program from a correct program [13].

Thus, mutation testing provides a means of evaluating a test criteria and test sets. However, with the increase in size of implementation, it is computationally expensive or infeasible to consider all possible mutants that can be hypothesized as the number of possible mutants is usually very large which makes the testing process expensive [14, 28, 27, 39].

The number of possible mutants is proportional to the product of the number of variables with the count of number of times they are referred either in a definition or in use[27, 38]. A consequence of the generation of a large number of possible mutant programs is that they need to be executed in each step of the testing phase till an adequate test set is obtained. To overcome this problem, a number of approaches have been suggested, such as finding an effective set of mutation operators, referred to as selective mutation [27], and variants of mutation testing [14, 40]. The original idea, as described above, is referred to as *strong mutation testing*.

This paper presents an alternative approach for reducing the cost of mutation testing by the identification of hierarchies among mutants. Let P_k and P_j be two mutants of a given PUT. It is possible to reduce test effort by considering only P_k if it can be deduced that a test set which can kill P_k is also be guaranteed to kill P_j.

The approach described here is also applicable to the quantitative assessment of testing effort and can be used to guide successive testing steps in fault-based testing. In particular, the objectives of this paper are:

a. To give theoretical foundation for identifying the relationship among mutants and show that mutation operator cannot be ordered without reference to a PUT;

b. To empirically evaluate the reduction in test effort that can be achieved by identifying the relationship among mutants;

c. To find if selective mutation study [27] is supported by our study.

The rest of the paper is organized as follows. The next section presents formal definitions in the context of mutant hierarchies. Section 3 describes the properties and conditions to identify the relationship among mutants. Section 4 gives an overview of the related work. An empirical study conducted to evaluate our approach and compare it with selective mutation is described in Section 5. Finally, conclusions are presented in Section 7.

2 Mutant Hierarchies

The theoretical and empirical study presented in this paper is done for the programs written in a subset of C programming language which include constructs such as loop, array and function calls. However, this does not impose any restriction on the use of those language constructs in programs that are not included in the subset.

For the purpose of theoretical analysis below, we assume that the statements, and predicates in conditional statements (such as `if` and `while`), of a program are uniquely labelled. Boolean conditions are used solely for deciding the branch to be followed in the next step of the execution and therefore are assumed not to modify the state of a program during execution. The assumption is justified as program transformation techniques can be used to achieve this and make programs more testable [12].

A label when given to a Boolean condition is said to be a *p-location*, otherwise it is said to be a *c-location*. Note that a condition in an **if** or **while** statement is given a unique label (i.e., different from the labels that are given to statements that appear inside the **then-else** or body of a **while** statement, respectively). Let l_k and l_j be two locations in P that are mutated to obtain mutants P_k and P_j respectively. These mutants will be known as *intra-location* mutants of P if $l_k = l_j$, otherwise they will be referred to as *inter-location* mutants.

We denote the output obtained on execution of a program M with an input x by $M(x)$. The notation $M = M'$ will be used to signify that a program M is semantically equivalent to another program M'. The state of a program under execution at an instant is set of pairs of variable and their corresponding values. A test case is a pair of input and the expected output. For simplicity, we will use input and test case interchangeably.

Definition Let P be a PUT and P_k be a first-order mutant that differs from P at a location. A test case, t, is said to kill a mutant P_k if one of the following conditions hold:

a. $P(t) \neq P_k(t)$, where both $P(t)$ and $P_k(t)$ are non-erroneous states.

b. either $P(t)$ or $P_k(t)$, but not both, results in an erroneous state.

c. both $P(t)$ or $P_k(t)$ results in different erroneous states.

We say a test set kills P_k if it includes a test case that kills P_k. Thus, a test case that kills a mutant identifies that a PUT and its mutant represent two distinct functions. For comparison, we observe the final internal state of programs.

During an execution of a program, it may fail, for example due to division by zero or insufficient memory, we call such a state an erroneous state. It may happen that one of the program fails while the other does not, in which case they are obviously distinguishable as stated in the definition 2(b) above. We classify non-termination of a program among the erroneous state. The definition 2(c) includes the

case where both programs result in distinguishable erroneous states such as a floating point exception and a memory fault.

Our analysis remains applicable even if we change the definition with respect to other variants: weak [14] and firm [40] mutation testing which allow to distinguish a PUT and its mutant by observing their internal states that are not final. A partial order between mutant programs can be defined using the following relation.

Definition [Relation between Mutants] Let P_k and P_j be the two mutants of P and t be a test case. Then P_k is said to be stronger than P_j denoted by $P, t \vdash P_k \geq_m P_j$ if

$$\exists\, t \mid t \text{ kills } P_k \Rightarrow t \text{ kills } P_j$$

The notation $P, t \vdash P_k \not\geq_m P_j$ will be used to indicate that P_k is not stronger than P_j. For a pair of mutants, P_k and P_j, of P if both $P, t \vdash P_k \not\geq_m P_j$ and $P, t \vdash P_j \not\geq_m P_k$ hold for all test cases t in a test set then both P_k and P_j must be considered during mutation testing.

Definition [Mutant Class] A set of mutants, S, of a PUT is said form a mutant class if there exists a test case that kills all the mutants in S.

Note that the relation among the mutants and the definition of mutant class are within the context of one test case. In other words, the mutant classes are induced by a test set. Therefore, for a given set of mutants, the mutant classes may be different with respect to different test sets. An alternative way to define the mutant relation, $\geq_m$, could be on the basis of comparing the constraints of all possible test inputs that kills a mutant rather than just by one single test case. This would make mutant classes unique for a given program and independent of test cases. However such a strong requirement will be hard and expensive to analyse in comparison to our weaker definition.

Fact 1. *[26, 29, 35] A test case, t, can kill P_k provided the following necessary and sufficient conditions hold on executing P and P_k with input t:*

a. the execution must reach location l (reachability);

b. the evaluation of expressions at location l in P and P_k must result in different values at least once (infection);

c. the final states on termination of execution of P and P_k must be different (propagation).

Condition (b) (i.e., infection) has been referred to as *necessity* in [26], and the *original state failure condition* in [29] consisting of an *origination condition* and *computational transfer conditions*.

Given a mutant P_k which is obtained by applying a mutation operator at a location l in a PUT with input domain D, let subdomain $D_k^r \subseteq D$ be the set of inputs which reaches location l; similarly, $D_k^i \subseteq D$ be the set of inputs that can cause the original and mutated expression at the location l to result in different values and $D_k^p \subseteq D$ be the set that causes P and P_k to result in different final outcomes.

```
    int fun(int x, int i) {
L1:   while (i <= 2) {      fun1          fun2
L2:     if (x <= 4)       if (x < 4)    if (x > 4)
L3:       x = x + 1;
L4:     else
L5:       x = x + 2;
L6:     i = i + 1;
L7:   }
L8:   return x;
    }
```

Figure 1: An example to illustrate the insufficiency of infection conditions.

Fact 2. *[26] Given P, a test case, t, will kill P_k iff $t \in D_k^p$ which implies $t \in D_k^r \cap D_k^i$ and $D_k^p \subseteq D_k^r \cap D_k^i$.*

Note that there may be test cases in D_k^i that does not satisfy the reachability condition. The computation of a test case that can kill a mutant is undecidable as the sets D_k^r, D_k^i and D_k^p cannot be computed, in general. However, in practice it is often possible to find such test cases using approximation techniques. On one hand, to compute (whenever feasible) the set D_k^i requires only analysis of the expression at location l. On the other hand, computation of D_k^r is more expensive and complex as it requires analysis of the paths that can reach location l.

Proposition 1. *$\forall t(P, t \vdash P_k \geq_m P_j) \Leftrightarrow D_k^p \subseteq D_j^p$, where P_k and P_j are mutants of P and t represents a test case.*

Proof. The proof follows from the definitions. □

3 Identifying Mutant Hierarchies

A brute-force method to identify $\geq_m$ relation is by checking if $D_k^p \subseteq D_j^p$. It is also possible to restrict the test cases to be selected from the set $D_k^p \cap D_j^p$, provided that this set is not empty, in which case killing P_k will also guarantee the same for P_j.

The objective of our analysis is to identify a subset of mutants with the same effectiveness as the whole set without generating all mutants. Therefore, if possible, the $\geq_m$ relation between mutants should be established during their generation itself, thereby only producing the strongest mutants. This approach is an improvement over the method where mutants are first generated explicitly and then an attempt to establish a partial order among them is made.

3.1 Intra-location Mutants

Let P be an implemented program and P_k and P_j be two mutants of P that are obtained by applying mutation operator at location l in P. Let C_k and C_j be the predicates that correspond to the sets D_k^i and D_j^i, respectively.

Now consider the example program shown in Figure 1. The mutants fun_1 and fun_2 are shown in boxes and are obtained by applying mutation operator at location $L2$, where

Input		Output		
x	i	fun	fun_1	fun_2
3	1	5	6	6
4	1	7	8	7

Table 1: A counter example for Theorem 1 based on the program shown in Figure 1.

```
   int Comp(int x) {  Comp1    Comp2
L1:   x = x + 1;     x = x − 1;   x = x + 2;
L2:   if (x == 5 || x == 7)
L3:      x = 9;
L4:   else
L5:      x = 6;
L6:   return x;
   }
```

Figure 2: An example for Theorem 1.

$<=, <$ and $>$ are relational operators. The conditions (obtained by taking exclusive-or of two expressions, for instance, $C_{fun_1} \equiv x <= 4 \oplus x < 4$, where $\oplus$ is exclusive-or operator) C_{fun_1} and C_{fun_2}, in this case are x = 4 and **true**, respectively. Although C_{fun_1} implies C_{fun_2}, it is not sufficient to claim a hierarchy between fun_1 and fun_2, in general. This is illustrated by the following theorem.

Theorem 1. *Let P be a PUT and P_k and P_j be its two mutants obtained by mutating a statement at location l then, $D^i_k \subseteq D^i_j$ does not guarantee $\forall t(P, t \vdash P_k \geq_m P_j)$.*

Proof. The statement holds for both a p-location and a c-location. We give counter-examples as a proof. Table 1 gives the output of two test cases for the program and its two mutants that are shown in the Figure 1. The first row shows that the mutants are not equivalent to the original program, whereas the second row shows that the conjecture is true for p-locations as fun_1 is killed and fun_2 remains live. Note that, as mentioned above, $C_{fun_1} \Rightarrow C_{fun_2}$ i.e. $D^i_{fun_1} \subseteq D^i_{fun_2}$.

Figure 2 shows a concrete example that illustrates the above theorem for a c-location. Consider the two mutants, $Comp_1$ and $Comp_2$, obtained by applying mutation operator at location $L1$ in Comp (see Figure 2). The D^i sets for both mutants is the whole input domain D since the mutated statements would result in different values for any integer input. In other words, the conditions C_1 and C_2 are *true*. However, input $x = 8$ will kill $Comp_1$; whereas input $x = 3$ will kill $Comp_2$ (see Table 2). Thus, mutants

Input (x)	Output		
	Comp	$Comp_1$	$Comp_2$
8	6	9	6
3	6	6	9

	$Comp_1$	$Comp_2$
D^i	D	D
D^p	$\{4, 8\}$	$\{3, 4, 5, 6\}$

Table 2: Input, output and subdomains for the programs shown in Figure 2.

$Comp_1$ and $Comp_2$ are not related under $\geq_m$ with respect to all possible test cases. □

The following formal observation gives insight into Theorem 1.

$$
\begin{array}{ll}
D^p_k \subseteq D^r_k \cap D^i_k & \text{(Fact 2)} \\
D^p_j \subseteq D^r_j \cap D^i_j & \text{(Fact 2)} \\
D^r_k = D^r_j & \text{(Same location mutants)} \\
D^i_k \subseteq D^i_j & \text{(given)}
\end{array}
$$

The most favourable conclusion that can be drawn from the above statements is that both D^p_k and D^p_j are subsets of $D^r_k \cap D^i_k$. But this does not guarantee $D^p_k \subseteq D^p_j$ (as required by Proposition 1).

Theorem 2. *Let P, P_k, P_j and l be the entities as stated in Theorem 1. If $D^i_k \cap D^i_j = \emptyset$ then P_k and P_j are not related under $\geq_m$ P or P is equivalent to P_k or P_j.*

Proof. The first three conditions are identical with the above formal observation.

$$
\begin{array}{lll}
 & D^i_k \cap D^i_j = \emptyset & \text{(given)} \\
\Rightarrow & D^p_k \cap D^p_j = \emptyset & \text{(set theory)} \\
\Rightarrow & D^p_k = \emptyset \vee D^p_j = \emptyset \vee & \\
 & (D^p_k \not\subseteq D^p_j \wedge D^p_j \not\subseteq D^p_k) & \text{(set theory)} \\
\Leftrightarrow & P = P_k \vee P = P_j \vee & \\
 & \quad \forall t\ (P, t \vdash P_k \not\geq_m P_j \wedge & \\
 & \qquad P, t \vdash P_j \not\geq_m P_k) & \text{(Fact 2 \& Prop. 1)}
\end{array}
$$

□

This is particularly helpful in *isolating* those mutants that definitely need to be considered during testing. However, a hierarchy can be established between mutants under certain conditions. These conditions are discussed below.

Theorem 3. *Let P be a given PUT and l be a p-location that corresponds to a condition, c. Further, let c be mutated to c' and c'' giving mutants P_k and P_j, respectively. If $(c' \Leftrightarrow c'')$ then one of the mutant P_k or P_j need not be considered during testing.*

Proof. As per Fact 1 (c), P and P_r ($r \in \{k, j\}$) must follow different paths after reaching location l (sometime during an execution) in order to be killed.

The condition $(c' \Leftrightarrow c'')$ ensures that condition c' and c'' always evaluate to the same Boolean value. Thus, for a given test case, the path followed by P_k and P_j will always be the same, ensuring that the infection and propagation conditions for both P_k and P_j will be identical i.e. $\forall t(P, t \vdash P_1 \geq_m P_2 \wedge P, t \vdash P_2 \geq_m P_1)$. □

The above condition in Theorem 3 is a very strong requirement. However, the property was found to be helpful in reducing the test effort during empirical study described in Section 5.

Remark 1. *Why do we need $c' \Leftrightarrow c''$ condition to hold in general? To answer this question, let us consider the two mutants P_k and P_j, obtained by mutating a condition for a* **while** *loop of P. For a given test case, let i, i_k and i_j be*

the number of times the **while** *loop is executed in P, P_k and P_j, respectively. Thus, the necessary conditions for killing P_k and P_j are $i \neq i_k$ and $i \neq i_j$ respectively. To establish $\forall t(P, t \vdash P_k \geq_m P_j)$, one of the following properties must hold:*

a. $i_k = i_j$, or

b. the resulting states must be the same after executing the body of **while** *loop i_k and i_j times.*

The condition in Theorem 3 is equivalent to (a) above. However, the condition (b) is equally acceptable, but requires analysis of the program segment to guarantee that it holds for any test case that kills P_k; this may be difficult to establish. It is also possible to weaken the requirement in Theorem 3 under certain conditions as described by the following theorem.

Theorem 4. *Let P be a PUT, t be a test case and l be a p-location that corresponds to a condition, c, in P. Further, let c be mutated by two operators to c' and c'' giving mutants P_k and P_j, respectively. If $(c \oplus c') \Rightarrow (c \oplus c'')$, where $\oplus$ is exclusive-or operator, and the label l is reached during the execution exactly once then $P, t \vdash P_k \geq_m P_j$.*

Proof. The programs P and P_k must follow different paths after reaching location l, sometime during an execution, in order to be distinguished. The condition $(c \oplus c') \Rightarrow (c \oplus c'')$ ensures that P_k and P_j will follow the same path, if P and P_k take different paths.

The necessity for the criterion of checking the internal state can be explained as follows. Assume that the Boolean condition is evaluated twice and c' differs from c in the second execution. However, it is possible that only c'' may differ from c in the first execution but not in the second execution, in which case it is not guaranteed that killing P_k will also ensure the same for P_j. □

Thus, Theorem 3, Remark 1 and Theorem 4 give three different possibilities to identify the hierarchies among mutants and also present the reasoning for the conditional requirements associated with them.

Note that in Theorem 4, c and c'' may differ, but c and c' may evaluate to the same values, in which case P_j may be killed but P_k will not. Thus P_j could be killed by more test cases than P_k. This can also be observed by noting that $c \oplus c'$ defines the subdomain D_k^i and considering the implication as a subset relation.

3.2 Inter-location Mutants

The above analysis is restricted to mutations in p-locations. For mutants that can be generated by mutating c-locations, we need to symbolically propagate the effect to nearest variable use in a predicate and then use theorems mentioned above to identify the relationship among mutants. We illustrate this by an example:

Consider the fragment of a PUT, R, shown in Figure 3. Let R_1, R_2 and R_3 be three mutants of R as shown in the

```
                          R1
La: b = -y;          [ b = y; ]
      :                          R2                R3
L:  if (x + y == a + b) [ x - y == a + b ] [ x + y == a - b ]
      St
    else
      Se
    end
```

Figure 3: Fragment of a program R.

figure. If the definition of variable b at location L_a is guaranteed to reach location L then we can compare R_1, R_2 and R_3 under $\geq_m$. Let `orig`, `r1`, `r2` and `r3` be predicates as defined below:

```
orig: x + y == a + b
  r1: x + y == a + y
  r2: x - y == a + b
  r3: x + y == a - b
```

By observing that the predicate

$$b == -y \Rightarrow ((orig \oplus r2) \Rightarrow (orig \oplus r3))$$

is valid, we conclude that $R_2 \geq_m R_3$. To check the relation of R_1 with R_2, we propagate the assignment at L_a to L which give us the following predicates:

```
orig': x + y == a - y
  r2': x - y == a - y
```

Since the predicate $(orig' \oplus r2') \Rightarrow (orig' \oplus r1)$ is also valid, $R_2 \geq_m R_1$. Thus, if the mutant R_2 can be killed, the other two mutants need not be considered during the testing. Note that, here we have also used known properties of the PUT in establishing the hierarchies. To consider the mutant R_1, the impact of mutation is propagated to location L. This can also be seen as making the control conditions more explicit and has been studied in the context of test data generation in [12].

Let P_j and P_k be two mutants of a program P that are obtained by mutating location l_j and l_k respectively. Consider the set $R = D_j^r \cap D_k^r$. If $R = \emptyset$, there is no relationship between P_j and P_k and both of them must be considered during mutation testing. The checking of condition $R = \emptyset$ does not necessarily require explicit computation of the reachability sets. An example where such a condition holds is when l_j and l_k appear in **then** and **else** branches of an **if** statement that is reached exactly once in an execution with a test case.

Consider the other case when $R \neq \emptyset$; i.e., there may be an execution that passes through both l_j and l_k. In this case in order to find if the two mutants are related, it is necessary to evaluate the impact of mutation at l_j at location l_k (or vice versa). If this symbolic evaluation can be done then

the hierarchies among the mutants can be identified with the aid of theorems mentioned above.

Such an analysis, in which the internal state is observed and the effect of a fault is propagated, is studied in the literature in terms of propagation conditions. For example, the local propagation of origination conditions is referred to as transfer conditions in [29].

The deductions required can be done automatically using existing tools such as the CVC3 [5], which can check the validity of quantifier-free first-order formulas over several interpreted theories including real linear arithmetic, arrays, uninterpreted functions, constants and abstract data types.

In the next section we give an overview of the related work.

4 Related Work

A number of research studies have been conducted to test the feasibility of mutation testing in an industrial context. For a recent survey on mutation testing see [18]. These studies concluded that mutation testing is difficult as the number of mutants generated and the time taken to kill each mutant when executed against the original program is myriad [28]. These research studies have proposed several variations of mutation testing. The two variants suggested by them are weak mutation [14] and firm mutation [40].

In weak mutation, the outputs of the original and mutated programs are compared immediately after the execution of the mutated statement. Firm mutation is a kind of "in-between" approach where the output of the original and mutated program can be compared at any location after the mutated statement and the end of the program. A short survey is presented in [28]. The variants of firm mutation testing can save some time during the execution and comparison phase of a PUT with its mutants, since the whole program need not be executed when comparing the outputs of the original program with the mutants.

An approach of computing the detection conditions for hypothesized faults has been extensively studied, for example, in *constraint-based testing* [7] and computation of failure conditions in [23, 29]. This require the sets D_k^p and D_j^p to be computed by using symbolic execution techniques, which give a constraint on inputs that must be satisfied by a distinguishing test case. Let C_k and C_j be two constraints that correspond to the subdomains D_k^p and D_j^p respectively. Then, $C_k \Rightarrow C_j$ will also guarantee that $\forall t(P, t \vdash P_k \geq_m P_j)$.

Another alternative to reduce the test effort is *selective mutation* [27] which attempts to identify a subset of mutation operators without significantly affecting the effectiveness. The analysis of Boolean expressions has been extensively studied in the literature; see for example [33, 37]. In [21, 34], a hierarchy between different types of faults that can arise in Boolean specifications is analyzed. These results are applicable in the context of Theorem 4.

It was found that the mutation operators, ABS, AOR, LCR, ROR and UOI, were sufficient for generating mutants. It was also observed that among those generated by these operators 57% mutants were equivalent mutants.

A similar approach, but complementary to that presented in this paper, has been suggested in [9, 30], involving the determination of an optimal ordering for the relational operators. The key idea can be stated as follows. Let P be a given program and P_k and P_j be two of its mutants that are obtained by replacing a relational operator, say RO, in P by other relational operators, RO_k and RO_j respectively, where RO_k is higher in the optimal ordering relation than RO_j [30]. Then, for a given input, if P_k remains live then P_j will also remain live. Thus, when attempting to kill mutants, P_k should be tried before P_j. However, Woodward in [39] has shown a fallacy in the above argument by providing a counter example and suggested the following in the conclusion:

> *One final point is that the fallacious argument ... is, in a sense, the opposite of that which a mutation tester really wants. The argument that "since this test data kills this mutant, it must be good and will kill these other mutants", would offer even greater potential benefits.*

This paper contributes towards the above argument.

In the next section, we describe the empirical study performed to evaluate the cost savings obtained by identifying the relationship among mutants.

5 Experimental Evaluation

The programs considered in this study are written in C programming language. As mentioned earlier, a mutation operator can be defined to be a rule for generating mutants by making a single syntactic change in PUT for example, changing operator '$+$' to '$-$'.

5.1 Experimental Setup

We have used 23 mutation operators that are adapted from operators defined in [20] for FORTRAN programming language. The summary of these operators is given in Table 3. Since mutation testing is applied at a unit or component level, in our study we have mutated all the functions in a PUT except the `main` function.

As mentioned before, in common with a number of program analysis problems such as reachability of a location, identifying every possible $\geq_m$ relation is undecidable in general. Nevertheless, in the restricted setup of mutation testing, where a program differs from its mutants in a well-defined way, it is possible to find the relationship between some, if not all, mutant programs. The consequence of any technique being inherently incomplete is that it may not always be able to deduce the $\geq_m$ relation between two given mutants. However, this is not harmful except that the number of mutants to be considered during testing will not

Operator	Description
AAR	array reference for array reference replacement
ABS	absolute value insertion
ACR	array reference for constant replacement
AOR	arithmetic operator replacement
ASR	array reference for scalar variable replacement
CAR	constant for array reference replacement
CNR	comparable array name replacement
CRP	constant replacement
CSR	constant for scalar variable replacement
DER	do/while statement end replacement
LCR	logical connector replacement
ROR	relational operator replacement
RSR	return statement replacement
SAN	statement analysis (replacement by TRAP)
SAR	scalar variable for array reference replacement
SCR	scalar for constant replacement
SDL	statement deletion
SRC	source constant replacement
SVR	scalar variable replacement
UOI	unary operator insertion

Table 3: Mutation operators [20]

be minimal. The symbolic evaluation is not done for the complete unit under test but for a segment consisting of the mutated line upto nearest p-location as mentioned in the section 2.

However, generating the set of strongest mutants may lead to problems due to the presence of equivalent mutants already in the set. Note that the property $P, t \vdash P_k \geq_m P_j$ holds trivially for all j if P is equivalent to P_k (i.e., when $D_k^p = \emptyset$). The statistics, as reported in [26], indicate that the number of equivalent mutants is typically in the range of 7 to 12% of the total number of mutants and the automatic detection rate using symbolic execution varies from 12 to 84% of the total number of equivalent mutants. Another recent study [31] further explores the identification of equivalent mutants. Therefore in our empirical study we rely on the fact that a randomly selected mutant from the set of all mutants is likely to be non-equivalent.

Thus, although explicit generation of all mutants may not be required, the information regarding them must still be maintained. This information about $P, t \vdash P_k \geq_m P_j$ will be required whenever it is deduced or suspected that $P = P_k$, for example when a significant amount of effort is spent in killing P_k without success (such as, large size of the test set and large number of times reachability and infection conditions are met). The instantiation order for mutants is guided by the hierarchies among them.

We considered terminating sequential programs. Since application of a mutation operator can generate a program which may not terminate on execution, we kept a threshold of 5 seconds to decide if the execution is in an infinite loop. This approach is similar to that implemented in MuJava tool[22]. The comparison is made on the output as in the case of strong mutation testing.

The analysis for empirical study was done manually with the aid of CVC [5] tool. To analyze a given program, we used symbolic execution techniques which have been used in a wide variety of problems, such as, test data generation [7] and detecting equivalent mutants [26]. In symbolic execution, a program is executed with the symbolic values representing arbitrary values, instead of actual input values. Such an execution results in a tree in which every node consists of symbolic values of the variables and the path constraint that must be true to reach that node.

The steps to carry out the experiments are given in Table 4. Let P be a PUT and $\mathcal{M}$ be the set of mutants of P. We start with the execution of an instrumented mutant using a test case that kills the selected mutant. The instrumentation is done to enable the observation of internal state which are used to identify other mutants in $\mathcal{M}$ that can also be killed using the same test case. Such mutants form a class. We restart the process with the remaining mutants and continue until no mutants are left to be classified or they are identified as equivalent.

We illustrate the approach using an example shown in Figure 4. Consider the three possible mutants marked as min_1 to min_3. Assume that we explicitly generate min_1 and would like to know if a test set that kills min_1 will also kill any of the other two mutants. Since min_1 and min_2 are obtained by mutating a c-location, we will propagate the effect to the p-location at `L3`. The necessary (but not sufficient) condition at location `L3` to kill these mutants are given below.

Cmin_1 : `a != b` $\vee$ `(a > b` $\oplus$ `b > b)`

Cmin_2 : `(a != -abs(a))` $\vee$ `(a > b` $\oplus$ `-abs(a) > b)`

1. *Create a list of all mutants for the given PUT in increasing order of line number of the PUT. Let N be the number of possible mutants.*
2. *Set i = 0*
3. *Let P_i be the i^{th} mutant.*
4. *If P_i is already classified to be equivalent or in a mutant class then goto step 9.*
5. *Identify the condition that are needed to establish the hierarchies for other intra-location and inter-location mutants with P_i.*
6. *Check the conditions with theorem prover if they are valid. If yes, mark the corresponding mutants and P_i to belong to a single class.*
7. *Generate P_i with statements to observe satisfiability of conditions identified in the step 5.*
8. *Identify a test case to kill P_i. Mark the mutants to belong to a single class for which conditions identified in step 5 are satisfied while executing P_i with the test case that kills P_i.*
9. *Set i = i + 1*
10. *If i ≠ N goto step 3.*

Table 4: Steps for Experimental Study.

```
int min(int x, int y) {
L1:   float m;
                    min1          min2
L2:   m = a;      | m = b; |  | m = −abs(a); |
                          min3
L3:   if (m > b)  | if (m != b) |
L4:     m = b;
L5:
L6:   return m;
 }
```

Figure 4: An example to illustrate the experimental approach.

$\mathbf{C}min_3$: `(a > b ⊕ a != b)`

If the test case $\{x = -6, y = 2\}$ is used to kill mutant min_1 then it is guaranteed that the same test case will also kill min_2 and min_3. However, if the test case $\{x = 0, y = 1\}$ is used to kill mutant min_1 then only min_3 will be killed. This can be observed by checking the satisfiability of conditions given above for these test cases. Therefore, if we use latter test case, min_1 and min_2 will be the same class, whereas with former all three mutants will belong to the same class.

In step 6, the identification of hierarchies require checking validity of first-order quantifier-free logical formulas. Let C be such a formula that we want to validate to determine if it holds at a location l in a program P. There are three possibilities for the property to hold: (a) C is valid;(b) C is satisfiable for some test cases; (c) C is false. For case (c), we did not the put the required checks in the generated mutant.

The following programs were considered for the empirical study.

compare_str The program is taken from [1] and is intended to compares two strings. The specification requires to return true if the given input strings are identical, otherwise false. However, the implementation returns correct output for unequal length strings and strings of equal length with the identical last characters. Thus, the strings "cat" and "mat" are reported to be identical. The subtle issue, as mentioned in [1], is that the probability of random selection of input which will reveal the bug is very low. Probability that n strings will expose the fault is $1 - (25/26)^n$ [1]. In our study, a strong mutant forced to select such an input. The implemented program is a first-order mutant of the intended program.

find This is an implementation of Hoare's find algorithm. This program and its buggy version are studied earlier in [3, 10]. We have used the correct version of find algorithm. One of the strong mutant forced selection of input that detected it to be different from the incorrect version. As noted in [3], for the buggy version, it is extremely difficult to identify test case using random selection. The buggy version has two faults and therefore is a second order mutant. Unlike previous studies, we decided to consider the correct version and check if the test set with 100% mutation score can also kill the second order buggy mutant.

gcd This program computes the greatest common divisor of given two positive integers.

iroot The program is taken from [16] and computes integer square root of a given positive integer. This was given as an exercise to the students of first course in structured programming. We observed that it was difficult to get a correct implementation.

min Computes minimum of two numbers and is also studied in [26, 27].

prime Tests if a given positive integer is prime. The implementation has a subtle bug which causes it to give incorrect result for only one input. The implemented (faulty) program is directly representable as a first order mutant of the intended program. It was required to generate the test case that detects the fault.

selection This is a faulty implementation of selection sort algorithm. The program is taken from the book [2] which contains program with a single, hard-to-detect

but realistic bug. In this case also, a test case detected the fault.

triangle This is the famous program, first mentioned in Myers [24], for testing whether the three given integers form a triangle and its classification.

tcas TCAS (Traffic Alert and Collision Avoidance System) is an aircraft conflict detection and resolution system. The SIR Siemens suite [8, 32] includes an ANSI C version of the resolution advisory component of TCAS system along with 41 faulty versions. The "Siemens" programs were assembled by Tom Ostrand and colleagues at Siemens Corporate Research. Our experiments include the 41 faulty versions and some other mutants.

5.2 Results Discussion

For the above programs, we studied number of equivalent mutants, number of mutant classes, operators with respect to those mutants in each mutant class that cannot be killed by the test cases used for other mutant classes. The statistics for these programs is given in Table 5 and 6. The savings are calculated using the formula given below.

$$\frac{\text{number of mutant classes}}{\text{number of mutants} - \text{equivalent mutants}} \times 100$$

As can be observed in Table 6 the overall savings is above 90% in all the cases.

The maximum number of equivalent mutants and mutant classes were found for the *triangle* program. The reason for the higher number of equivalent mutants is due to the redundant test $a \leq 0 \vee b \leq 0 \vee c \leq 0$ in the presence of another check that sum of two sides is more than the third side. Also, the application of *abs* operator after initial testing of values to be positive since all values are guaranteed to be positive afterwards. Finally, the application of SVR operator on test for equilateral triangle $a = b \wedge b = c$ also generates equivalent mutants.

For each of the above example, once the identification of mutant classes was complete, we identified those mutants in each class that cannot be killed by the test cases of all other classes. The table reports the mutation operators associated with such mutants.

In selective mutation study [27], it was found that the mutation operators, ABS, AOR, LCR, ROR and UOI, were sufficient for generating mutants. In our study, we also find that the strong mutant were generated by these operators. However, we also observed that some strong mutants were generated by SVR operator. In one case we found that DER operator too generated a strong mutant.

6 Threat to Validity

The threat to validity of our empirical results could be due the set of programs used in the study. Although the programs are selected from the variety of sources, they cannot be claimed to be representatives of set of all programs. Our strategy requires pre-analysis to identify the relationship among mutants.

In comparison, the selective mutation [27] does not require any pre-analysis to generate mutants for a program although it may generate more mutants. The mutants are generated using a subset of set of operators defined in the previous study [20]. In contrast, our approach does not classify operators but attempts to identify relationship among mutants to avoid ignoring any mutant. Thus it is guaranteed to ensure the quality of a test set. Although this enables inclusion of new operators without affecting the effectiveness but increases the overall cost of testing.

We expect the cost of our technique to be more than selective mutation but less compared to full mutation testing. From the effectiveness point of view, if selective mutation chooses appropriate set of mutants than it may be as effective as full mutation testing. Also, since the program analysis in our study was done manually there is a possibility that the results may differ with a completely automated analysis. Nevertheless, the results of empirical study provide some confidence in the approach.

7 Conclusions

Mutation testing is a powerful testing approach that can not only ensure the checking of hypothesized faults but also the generation of test data satisfying common structural coverage criteria. The main difficulty faced in mutation testing is due to the large number of mutant programs that can be generated for a given implemented program. We have given a strategy that suggests the ordering of the mutants such that if a mutant is stronger than another, then killing the stronger will automatically kill the weaker. This approach can significantly reduce the cost of mutation testing. Although our approach ensures the same effectiveness as full mutation testing, it is expensive than selective mutation testing. To obtain the exact cost comparison we will require extensive research on automation of our approach, which we plan to pursue in future.

Identification of such hierarchies is also useful in the quantitative assessment of the quality of fault detection effectiveness, since, with the knowledge about mutant hierarchies, it is possible to reason if the mutation-adequacy score includes *strong* mutants. This is particularly helpful in directing the test effort with every step of the testing process.

The issue in identifying all possible orderings is mainly due to the undecidability of the problem and it also depends on the complexity of the program. However as our empirical study shows, a significant cost saving can be achieved even if one can identify some of the possible orderings among the mutants.

We have presented various conditions to identify the relationship between mutants that can be analyzed locally

Program	Lines of code	Mutants	Equivalent mutants
compare_str	26	111	9
find	65	464	36
gcd	23	136	13
iroot	26	134	5
min	21	53	3
prime	26	99	10
selection	55	263	38
triangle	37	524	59
tcas	174	85	2

Table 5: Statistics for programs considered in empirical study

Program	Mutant classes	Operators for strong mutants	Savings %
compare_str	7	ABS, ROR, RSR, SVR, UOI	93
find	5	ABS, ROR, UOI, SVR	99
gcd	4	ABS, AOR, ROR, UOI, SVR	97
iroot	4	AOR, CSR, UOI, DER	96
min	4	ABS, SVR, UOI	92
prime	4	CSR, ROR, UOI	95
selection	4	ABS, AOR, ROR, UOI	96
triangle	24	ABS, ROR, SVR	93
tcas	17	CRP, LCR, ROR, SVR	79

Table 6: Results for programs considered in empirical study

and thus can be evaluated in an effective way. Our work gives theoretical proof and the empirical evaluation with the examples taken from previous studies shows the feasibility of idea and significant cost savings that can be achieved. We found that the operators associated with strong mutants were the same as those identified in selective mutation. However in addition SVR and DER operators also generated strong mutants in some cases.

As the kind of analysis required to establish hierarchies is already part of various program analysis such as constant propagation, and transformation tools, and also tools like CVC3 [5] are already available, the given approach should be practical.

References

[1] R. C. Backhouse. *Program Construction and Verification*. Prentice-Hall International, 1986.

[2] A. Barr. *Find the Bug: A Book of Incorrect Programs*. Addison-Wesley, 2004.

[3] R. S. Boyer, B. Elspas, and K. N. Levitt. SELECT — a formal system for testing and debugging programs by symbolic execution. In *International conference on Reliable software*, pages 234–245, 1975.

[4] T. A. Budd, R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Theoretical and Empirical Studies on using Program Mutation to Test the Functional Correctness of Programs. In *7th Symposium on Principles of Programming Languages*, pages 220–233. ACM, 1980.

[5] CVC3: An Automatic Theorem Prover, 2007. http://www.cs.nyu.edu/acsys/cvc3/ (last accessed: Dec. 2007).

[6] R. A. DeMillo, R. J. Lipton, and F. G. Sayward. Hints on Test Data Selection: Help for the Practicing Programmer. *IEEE Computer*, 11(4):34–41, April 1978.

[7] R. A. DeMillo and A. J. Offutt. Constraint-Based Automatic Test Data Generation. *IEEE Transactions on Software Engineering*, 17(9):900–910, September 1991.

[8] H. Do, S. G. Elbaum, and G. Rothermel. Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact. *Empirical Software Engineering: An International Journal*, 10(4):405–435, 2005.

[9] I. M. M. Duncan and D. J. Robson. Ordered Mutation Testing. In *ACM SIGSOFT Software Engineering Notes*, volume 15, pages 29–30, April 1990.

[10] P. G. Frankl, S. N. Weiss, and C. Hu. All-uses vs mutation testing: An experimental comparison of effectiveness. *Journal of Systems and Software*, 38(3):235–253, September 1997.

[11] M.-C. Gaudel. Testing can be Formal too. In *Theory and Practice of Software Development (TAPSOFT)*, volume 915, pages 82–96, March 1995.

[12] M. Harman, L. Hu, R. Hierons, J. Wegener, H. Sthamer, A. Baresel, and M. Roper. Testability Transformation. *IEEE Transactions on Software Engineering*, 30(1):3–16, 2004.

[13] R. M. Hierons. Comparing test sets and criteria in the presence of test hypotheses and fault domains. *ACM Transactions on Software Engineering and Methodology*, 11(4):427–448, October 2002.

[14] W. E. Howden. Weak Mutation Testing and Completeness of Test Sets. *IEEE Transactions on Software Engineering*, 8:371–379, July 1982.

[15] IEEE. *IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std.* 610.12. New York: Institute of Electrical and Electronics Engineers, 1990.

[16] J. Jacky. *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press, 1997.

[17] Y. Jia and M. Harman. Higher Order Mutation Testing. *Information and Software Technology*, 51(10):1379–1393, October 2009.

[18] Y. Jia and M. Harman. An Analysis and Survey of the Development of Mutation Testing. *IEEE Transactions on Software Engineering*, To appear, 2010.

[19] K. Kapoor. Formal Analysis of Coupling Hypothesis for Logical Faults. *Innovations in Systems and Software Engineering (A NASA Journal)*, 2(2):80–87, July 2006.

[20] K. N. King and A. J. Offutt. A Fortran Language System for Mutation-Based Software Testing. *Software Practice and Experience*, 21(7):685–718, 1991.

[21] D. R. Kuhn. Fault Classes and Error Detection Capability of Specification-based Testing. *ACM Transactions on Software Engineering and Methodology*, 8(4):411–424, October 1999.

[22] Y.-S. Ma, A. J. Offutt, and Y. R. Kwon. MuJava: An Automated Class Mutation System. *Software Testing, Verification and Reliability*, 15(2):97–133, June 2005.

[23] L. J. Morell. A Theory of Fault-based Testing. *IEEE Transactions on Software Engineering*, 16(8):844–857, August 1990.

[24] G. Myers. *The Art of Software Testing*. Wiley-Interscience, 1979.

[25] A. J. Offutt. Investigations of the Software Testing Coupling Effect. *ACM Transactions on Software Engineering and Methodology*, 1(1):5–20, January 1992.

[26] A. J. Offutt and J. Pan. Automatically Detecting Equivalent Mutants and Infeasible Paths. *Software Testing, Verification and Reliability*, 7(3):165–192, September 1997.

[27] A. J. Offutt, G. Rothermel, R. H. Untch, and C. Zapf. An Experimental Determination of Sufficient Mutant Operator. *ACM Transactions on Software Engineering and Methodology*, 5(2):99–118, April 1996.

[28] A. J. Offutt and R. H. Untch. Mutation 2000: Uniting the Orthogonal. In W. E. Wong, editor, *Mutation Testing in the Twentieth and the Twenty First Centuries*, pages 45–55. Kluwer, October 2000.

[29] D. J. Richardson and M. C. Thompson. An Analysis of Test Data Selection Criteria using the RELAY Model of Fault Detection. *IEEE Transactions on Software Engineering*, 19(6):533–553, June 1993.

[30] I. J. Riddell, M. A. Hennell, M. R. Woodward, and D. Hedley. Practical Aspects of Program Mutation. Technical report, University of Liverpool, UK, 1982.

[31] D. Schuler and A. Zeller. (Un-)Covering Equivalent Mutants. In *3rd International Conference on Software Testing Verification and Validation (ICST'10)*, Paris, France, April 2010.

[32] SIR: Software-artifact Infrastructure Repository (SIR), 2010. http://sir.unl.edu/ (last accessed: Oct. 2010).

[33] K.-C. Tai. Theory of Fault-based Predicate Testing for Computer Programs. *IEEE Transactions on Software Engineering*, 22(8):552–562, August 1996.

[34] T. Tsuchiya and T. Kikuno. On Fault Classes and Error Detection Capability of Specification-based Testing. *ACM Transactions on Software Engineering and Methodology*, 11(1):58–62, January 2002.

[35] J. M. Voas. PIE: A Dynamic Failure-Based Technique. *IEEE Transactions on Software Engineering*, 18(2):717–727, August 1992.

[36] K. S. H. T. Wah. An Analysis of the Coupling Effect I: Single Test Data. *Science of Computer Programming*, 48:119–161, 2003.

[37] E. J. Weyuker, T. Goradia, and A. Singh. Automatically Generating Test Data from a Boolean Specification. *IEEE Transactions on Software Engineering*, 20(5):353–363, May 1994.

[38] W. E. Wong and A. P. Mathur. Reducing the Cost of Mutation Testing: An Empirical Study. *Journal of Systems and Software*, 31(3):185–196, December 1995.

[39] M. R. Woodward. Concerning Ordered Mutation Testing of Relational Operators. *Software Testing, Verification and Reliability*, 1(3):35–40, October 1991.

[40] M. R. Woodward and K. Halewood. From Weak to Strong, Dead or Alive? An Analysis of some Mutation Testing Issues. In *2nd Workshop on Software Testing, Verification, and Analysis*, pages 152–158, July 1988.

Improved ID-based Ring Signature Scheme with Constant-size Signatures

Hongwei Li, Xiao Li and Mingxing He
School of Mathematics and Computer Engineering
Xihua University
E-mail: lhwlihongwei@gmail.com

Shengke Zeng
School of Computer Science and Engineering
University of Electronic Science and Technology of China
E-mail: doris82414@sina.com

Keywords: ring signature, accumulator, constant-size, random oracle

Received: September 7, 2010

Ring signature enable a user to sign a message on behalf of the ring, without revealing the actual signer. Constant-size ring signature is the ring scheme that the size of the signature does not grow with the size of the ring(or group), so it is practical for large rings. In this paper we use the Collision Resistant Accumulator from bilinear pairing to construct an identity-based ring signature scheme with constant-size signature. Our scheme actually is an improvement on the modified version of the scheme proposed by Nguyen, but we greatly improved the efficiency in terms of computational complexity and signature size. To the best of our knowledge, our scheme is the most efficient secure ID-based ring signature with constant-size based on accumulator proposed to date. Our scheme is proven secure in the random oracle model based on a simplified and general Forking Lemma under the k-strong Diffie-Hellman assumption.

Povzetek: Predstavljena je izboljšana metoda podpisa za obroč.

1 Introduction

Ring signature schemes, introduced by Rivest, Shamir and Tauman [1], allow a signer to form a group without a central authority and sign messages on behalf of the group. A user might not even know that he has been included in a group and even a party with unlimited computing resources can not find out the actual signer. In order to remove the need of certification of the public keys, Shamir [2] proposed the concept of ID-based cryptology to simplify public key management. Zhang and Kim [3] extended the ring signature to the ID-based ring signature schemes, where the user's public keys is their identities. The accumulator was introduced by Benaloh and de Mare [4] in order to design distributed protocols without the presence of a trusted central authority. Such a cryptographic primitive is an algorithm allowing the aggregation of a large set of elements into a single value of constant size. So the accumulator could be applied to construct constant size ring signature. Barić and Pfitzmann [5] generalized the definition of accumulators and constructed a collision-free subtype. As an application, they construct a fail-stop signature scheme based on their collision-free accumulator. Camenisch and Lysyanskaya [6] extended the concept of accumulators to dynamic accumulators which allow the addition and deletion of values from the original set of elements. Dodis, Kiayias, Nicolosi and Shoup [7] introduced ad hoc anonymous identification schemes based on the notion of accumulator with one-way domain, an extension of cryptographic accumulators. In 2005, Nguyen [8] proposed a dynamic accumulator based on bilinear pairings to design ID-based ad-hoc anonymous identification schemes and identity escrow protocols with membership revocation. However, Tartary, Zhou, Lin, Wang and Pieprzyk [9] demonstrated that the security model proposed by Nguyen did lead to a cryptographic accumulator which is not collision resistant. later, Nguyen had modified the security model [10] so that collision resistance can be provided. In 2009, Camenisch, Kohlweiss and Soriente proposed a new dynamic accumulator [24] based on bilinear maps for revocation of the authentication credentials. In their construction, however, in the case of accumulating an arbitrary set of size n, the issuer of the accumulator would need to publish a mapping from the set of identities to the elements of destined group. It looks like very difficult to construct ring signature schemes by hiring their accumulator [24].

Since Zhang and Kim [3] proposed the first ID-based ring signature scheme, there are lots of excellent Id-based ring signature schemes have been proposed [11, 12, 13], but all of them the size of ring signatures linearly depend on the group size, thus not practical for large groups. Actually, all the previous proposals had signature size proportional to the size of the ring before the scheme [7] proposed by Dodis, Kiayias, Nicolosi and Shoup. They

provided an ad-hoc anonymous identification scheme and used the Fiat-Shamir heuristics [15] to convert it into the public key which was prime, so an extension supporting ID-based keys seemed to be non-trivial [14]. The first ID-based ring signature scheme with constant-size signatures [8] was proposed by Nguyen. Similar to scheme [7], Nguyen also obtained the constant-size ring signature using the Fiat-Shamir transform [15] from anonymous identification scheme. However, the scheme [8] was found flawed by Zhang and Chen [16]. After that, Nguyen proposed the modified version [10] of the original scheme [8] and shown that the ring signature in their scheme [10] is much more efficient than previous one [7]. So, is there still some room for the computational efficiency of the constant-size ring signature to improve? Is there a way to reduce the size of the constant-size ring signature scheme?

We provide the affirmative answer to these questions and deem that the constant-size ring signature scheme [10] is still not efficient enough. We propose the improved constant-size ring signature scheme which is much more efficient either on computational complexity or signature size than the scheme [10] proposed by Nguyen. Moreover, Nguyen doesn't directly give the security reduction of their ring signature scheme, but we provide the security proof in the random oracle model under the k-strong Diffie-Hellman assumption. To the best of our knowledge, our scheme is the most efficient ID-based constant-size ring signature based on accumulator in the literature.

The rest of this paper is organized as follows. In Section 2, we briefly review some notations and complexity assumptions that will be used throughout this paper. We explain the general characteristics of a ID-based constant-size ring signature scheme, and the security properties that such a scheme must satisfy in Section 3. In Section 4, We present our new ring scheme, and provide security results for it in the section 5. In section 6, we compare its efficiency with previous schemes. Finally, we sum up the work in Section 7.

2 Preliminaries

In this section, we briefly introduce some preliminaries that will be used throughout this paper. A string means a binary one. If $x_1, x_2, \ldots$ are objects, then $x_1||x_2||\ldots$ denotes an encoding of them as strings from which the constituent objects are easily recoverable. If S is a set, $s \in_R S$ denotes the operation of assigning to s an element of S chosen at random. $s \leftarrow s^{'}$ means we let $s = s^{'}$. If A is a randomized algorithm, then $A(x_1, \ldots; \rho)$ denotes its output on inputs $x_1, \ldots$ and ρ, while $y \leftarrow_R A(x_1, \ldots; \rho)$ means that we choose ρ at random and let $y = A(x_1, \ldots; \rho)$.

2.1 Bilinear map groups and related computational problems [25]

Let l be a security parameter and p be a l-bit prime. Let us consider groups G_1, G_2 and G_T of the same prime order p and let P, Q be the generators of G_1 and G_2 respectively. We say that (G_1, G_2, G_T) are bilinear map groups if there exists a bilinear map $e : G_1 \times G_2 \rightarrow G_T$ satisfying the following properties:

- Bilinearity: $\forall (S,T) \in G_1 \times G_2, \forall a, b \in \mathbb{Z}_p^*, e(aS, bT) = e(S,T)^{ab}$.
- Non-degeneracy: $\forall S \in G_1, e(S,T) = 1$ for all $T \in G_2$ iff $S = \mathcal{O}$.
- Computability: $\forall (S,T) \in G_1 \times G_2, e(S,T)$ is efficiently computable.
- There exits an efficient, publicly computable (but not necessarily invertible) isomorphism $\psi : G_2 \rightarrow G_1$ such that $\psi(Q) = P$

Such bilinear map groups are known to be instantiate with ordinary elliptic curves such as that suggested in [21]. In this case, the trace map can be used as an efficient isomorphic ψ as long as G_2 is properly chosen [22]. With supersingular curves, symmetric pairings(i.e.$G_1 = G_1$) can be obtained and ψ is the identity. The computational assumption for the security of our scheme was proposed by Boneh and Boyen [27] and is recalled in the following definition.

Definition 1. *Let us consider the bilinear map groups (G_1, G_2, G_T) and the generators $P \in G_1$ and $Q \in G_2$. The k-strong Diffie-Hellman problem in the groups G_1, G_2 is defined as follows: given a (k+2)-tuple $(P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^k Q)$ as input, where $P = \psi(Q)$, output a pair $(c, \frac{1}{c+\alpha} P)$ with $c \in Z_p^*$.*

2.2 Collision Resistant Accumulator

Here we present the definition of accumulators and the collision resistance property as set by Nguyen [10].

Definition 2. ***(Accumulator[10])*** *Accumulator is a tuple $(\{X_l\}_{l \in N}, \{F_l\}_{l \in N})$, where $\{X_l\}_{l \in N}$ is called the value domain of the accumulator and $\{F_l\}_{l \in N}$ is a sequence of families of pairs of functions such that each $(f, g) \in F_l$ is defined as $f : U_f \times X_f^{ext} \rightarrow U_f$ for some $X_1 \subseteq X_f^{ext}$, and $g : U_f \rightarrow U_g$ is a bijective function. In addition, the following properties are satisfied:*

- *(efficient generation) There exists an efficient algorithm $\mathcal{G}$ that takes as input a security parameter 1^l and outputs a random element $(f, g) \in_R F_l$, possibly together with some auxiliary information a_f.*
- *(quasi commutativity) For every $l \in N$, $(f, g) \in F_l$, $u \in U_f$, $x_1, x_2 \in X_l$, $f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$. For any $l \in N$, $(f, g) \in F_l$, and $X = \{x_l, ..., x_k\} \subset X_l$, we call $g(f(...f(u, x_1)..., x_k))$ the*

accumulated value of the set X over u. Due to quasi commutativity, the value $g(f(...f(u,x_1)...,x_k))$ is independent of the order of the x_i's and is denoted by $f(u,X)$.

- *(efficient evaluation) For every $(f,g) \in F_l$, $u \in U_f$ and $X \subset X_l$ with size bound by a polynomial of l : $g(f(u,X))$ is computable in time polynomial in l, even without the knowledge of a_f .*

Definition 3. *(Collision Resistant Accumulator [9] [10]). An accumulator is defined as collision resistant if for every PPT algorithm A, the following function $Adv_A^{col.acc}(l) = Pr[(f,g) \in_R F_l; u \in_R U_f; (x,w,X) \leftarrow A(g \circ f, U_f, u) | (X \subset X_l) \wedge (w \in U_g) \wedge (x \in X_f^{ext} \setminus X) \wedge (f(g^{-1}(w),x) = f(u,X))]$ is negligible as a function of l. We say that w is a witness for the fact that $x \in X_l$ has been accumulated in $v \in U_g$ whenever $g(f(g^{-1}(w),x) = v$.*

To generate an instance of the accumulator [10] from the security parameter l, run the algorithm $\mathcal{G}$ with 1^l to obtain a tuple $t = (p, G_1, G_2, G_T, e(\cdot,\cdot), P, Q)$ and a uniformly chosen element s from Z_p^*. We construct a tuple $t^{'} = (P, Q, sQ, \ldots, s^qQ)$ where q is the an upper bound on the number of elements to be accumulated. The corresponding functions (f,g) for this instance $(t, t^{'})$ are defined as:

$$f : Z_p \times Z_p \to Z_p \qquad g : Z_p \to G_2$$
$$(v,x) \mapsto (x+s)v \qquad v \mapsto vQ$$

This construction involves that we have:

$$U_f = X_f^{ext} = Z_p \quad U_g = G_2 \quad X_l = Z_p \setminus \{-s\}$$

It is clear that f is quasi-commutative. In addition, for $u \in Z_p$ and a set $X = \{x_1, ..., x_k\} \subseteq Z_p \setminus \{-s\}$ where $k \leq p$, the accumulated value $g(f(u,X)) = \prod_{i=1}^{k}(x_i + s)uQ$ is computable in time polynomial in l from the tuple $t^{'}$ and without the knowledge of the auxiliary information s. The accumulator proposed by Nguyen [10] has been proven secure by [9] under the k-strong Diffie-Hellman assumption.

2.3 New General Forking Lemma

The security proof of our ID-based constant-size Ring Signature scheme relies on a generalization of the Forking Lemma [18] proposed by Bellare and Neven instead of the Forking Lemma in ring scenario [20] proposed by Herranz and Sáez.

Lemma 1. *(General Forking Lemma [18] [23]). Fix an integer $Q \geq 1$ and a set H of size $\mid H \mid \geq 2$. Let B be a randomized algorithm that on input $x, h_1, \ldots, h_Q$ returns a pair (J,σ) where $J \in \{0, \ldots, Q\}$ and σ is referred as side output. Let IG be a randomized algorithm called the input generator. Let $acc_B = Pr[J \geq 1 : x \leftarrow_R IG, h_1, \ldots, h_Q \leftarrow_R H; (J,\sigma) \leftarrow_R B(x, h_1, \ldots, h_Q)]$ be the accepting probability of B. The forking algorithm F_B associated to B is the randomized algorithm that takes in input x and proceeds as follows:*

Algorithm $F_B(x)$
Pick random coins ρ for B
$h_1, \ldots, h_Q \leftarrow_R H$
$(J,\sigma) \leftarrow B(x, h_1, \ldots, h_Q; \rho)$
If $J = 0$ then return $(0, \perp, \perp)$
$h_1^{'}, \ldots, h_Q^{'} \leftarrow_R H$
$(J^{'}, \sigma^{'}) \leftarrow B(x, h_1, \ldots, h_{J-1}, h_J^{'}, \ldots, h_Q^{'}; \rho)$
If $(J = J^{'}$ and $h_J \neq h_J^{'})$ then return $(1, \sigma, \sigma^{'})$
Else return $(0, \perp, \perp)$.

Let $frk = Pr[b = 1 : x \leftarrow_R IG; (b, \sigma, \sigma^{'} \leftarrow_R F_B(x)]$. Then $frk \geq acc_B(\frac{acc_B}{Q} - \frac{1}{|H|})$.

The exactly proof of this lemma could be found in [18]. Roughly says that if an algorithm B accepts with some non-negligible probability, then a "rewind" of B is likely to accept roughly with the probability squared[23]. The intuitions are that: $(1) h_1, \ldots, h_Q$ can be seen as the set of replies to random oracle queries made by the original adversary and (2) the forking algorithm implements the rewinding. Moreover, it is important that in F_B the two executions of B are run with the same random coins.

3 The Model of ID-based Constant-size Ring Signature

3.1 ID-based Constant-size Ring Signature Schemes

Here we give the definition of ID-based constant-size ring signature schemes, which is quite the same as the definition in [10].

Definition 4. *An ID-based constant-size ring signature scheme is as a tuple IR =(Setup, KeyGen, MakeGPK, MakeGSK, Sign, Verify) of PT algorithms, which are described as follows.*

- ***Setup:** takes as input a security parameter 1^l and returns the public parameters $params$ and a master key mk. The master key is only known to the Private Key Generator (PKG).*
- ***KeyGen:** run by the PKG, takes as input $params$, mk and an arbitrary identity id_i and outputs a private key s_{id_i}. The identity is used as the corresponding public key.*
- ***MakeGPK:** takes as input $params$ and a set of identities and deterministically outputs a single group public key gpk which is used in the ID-based ring signature scheme described below. Its cost linearly depends on the number of identities being aggregated. The algorithm is order invariant that means the order of aggregating the identities does not matter.*

- ***MakeGSK:*** *takes as input $params$, a set of identities R, a pair of an identity id_i and the corresponding private key s_{id_i} and deterministically outputs a single group secret key gsk_{id_i} which is used in the ID-based ring signature scheme described below. It should be noted that each user has its own group secret key gsk_{id_i} which is different from the others. Its cost linearly depends on the number of identities being aggregated. It can be observed that a group secret key $gsk_{id_i} \leftarrow MakeGSK(params, S^{'}, (s_{id_i}, id_i))$ corresponds to a group public key $gpk \leftarrow MakeGPK(params, S)$ if and only if $S = S^{'} \cup id_i$. More than one group secret key might correspond to the same group public key.*

- ***Sign:*** *takes as input the public parameter $params$, a user private key s_{id_i}, the user's group secret key gsk_{id_i}, group public key gpk which includes the identity corresponding to id_i, and a message m, outputs a signature σ for m.*

- ***Verify:*** *The deterministic polynomial time(DPT) algorithm takes as input a set of identities R, group public key gpk, a message m and a ring signature σ, and outputs either $accept$ or $reject$.*

3.2 Security Requirements

There are two preliminary security requirements for ID-based ring signature schemes: Anonymity and Unforgeability.

- **Anonymity:** the anonymity requires, informally, that an adversary should not be able to tell which member of a ring generated the particular signature.

- **Unforgeability:** the intuitive notion of unforgeability is that an adversary should be unable to output (m, σ) such that Verify$(m, \sigma) = 1$. However, there are lots of security definitions about unforgeability of ring signature [20]. We use the unforgeability definition [26] proposed by Herranz. It should be noted that [26]'s unforgeability definition is very similar to the strongest unforgeability definition(unforgeability w.r.t. insider corruption) proposed in [20].

  Definition 5. *(Unforgeability against chosen messages/identies attacks). A ring signature scheme (Setup, KeyGen, MakeGPK, MakeGSK, Sign, Verify) is unforgeable with chosen-subring attacks if for any PPT adversary A and for any polynomial $n(\cdot)$, the probability that A succeeds in the following game is negligible:*

  - *the challenger takes a security parameter k and runs the Setup algorithm of the scheme. He gives the resulting parameter to adversary.*
  - *A is given access to a signing oracle $OSign(\cdot, \cdot, \cdot)$, $OSign(id_s, m, R)$ returns $Sign_{s_{id_s}}(m, R)$, where we require $id_s \in R$, where R is a set of identities.*
  - *A is also given access to extraction oracle $Extraction(\cdot)$, where $Extraction(ID_i)$ outputs corresponding secret key sk_i.*
  - *at the end of the above execution, A outputs (R^*, m^*, σ^*) and succeeds if $Verify_{R^*}(m^*, \sigma^*) = 1$, A never queried $(R^*, m^*, \cdot)$ to its signing oracle, and for all $ID_i \in R$, the adversary has not requested an extraction query for ID_i.*

4 The Proposed Constant-size Ring Signature Scheme

In this section, we present our ID-based constant-size ring signature scheme, Our scheme is the modified version of the scheme [10] proposed by Nguyen, we describe our scheme as the following algorithms: Setup, KeyGen, MakeGPK, MakeGSK, Sign, Verify.

- **Setup:** on a security parameter l, chooses $s \in_R Z_p^*$, $u \in_R Z_p^*$ and generates an collision resistant accumulator as in section 2.2, including functions (f, g) and tuples $t \leftarrow (p, G_1, G_2, G_T, e(\cdot, \cdot), \psi)$ and $t^{'} \leftarrow (P, Q, sQ, \ldots, s^qQ)$, where q is the upper bound on the number of identities to be aggregated. It sets $Q_{pub} = sQ, P_{pub} = \psi(Q_{pub})$. Let H_0, H_1 be collision-free hash function $H_0 : \{0,1\}^* \rightarrow Z_p^*$, $H_1 : \{0,1\}^* \rightarrow Z_p^*$. Then, public parameter $params \leftarrow (l, t, t^{'}, u, H_0, H_1, f \circ g)$ and the master key is $mk \leftarrow s$.

- **KeyGen:** extracts a private key $s_{id_i} \leftarrow \frac{1}{H_0(id_i)+s}P$ for an identity id_i. The identity is used as the corresponding public key. The user can verify the private key by checking $e(H_0(id_i)Q + Q_{pub}, s_{id_i}) \stackrel{?}{=} e(Q, P)$.

- **MakeGPK:** given a set of identities $R = \{id_i\}_{i=1}^k$, computes the set X = $\{H_0(id_i)\}_{i=1}^k$ and generates the group public key for the set $gpk = V \leftarrow g(f(u, X))$.

- **MakeGSK:** generates the group secret key gsk for a user $id_s \in R$, $R = \{id_i\}_{i=1}^k$ by just computing the set $X^{'} \leftarrow \{H_0(id_i)\}_{i=1, i \neq s}^k, h_{id_s} \leftarrow H_0(id_s)$ and the witness $W \leftarrow g(f(u, X^{'}))$. Note that $X = X^{'} \cup h_{id_s}$. The group secret key is $gsk = (h_{id_s}, s_{id_s}, W)$.

- **Sign:** given a message m, a set of identities $R = \{id_i\}_{i=1}^k$ which includes the signer's identity id_s, the signer's private key s_{id_s}.

  - Given a message $m \in_R \{0,1\}^*$, choose $r_1, r_2, k_1, k_2, k_3, k_4, k_5 \in_R Z_p^*$.

- Compute $U_1 \leftarrow s_{id_s} + r_1 P$; $U_2 \leftarrow W + r_2 Q$; then compute $\prod_1 \leftarrow e(Q, U_1)^{-k_5} \cdot e(Q, P)^{k_2} \cdot e(Q_{pub}, P)^{k_1}$; $\prod_2 \leftarrow e(P, U_2)^{-k_5} \cdot e(P, Q)^{k_4} \cdot e(P_{pub}, Q)^{k_3}$.
- Get $c \leftarrow H_1(m||U_1||U_2||\prod_1||\prod_2||R)$.
- Then compute $s_1 \leftarrow k_1 + cr_1$; $s_2 \leftarrow k_2 + cr_1 h_{id_s}$; $s_3 \leftarrow k_3 + cr_2$; $s_4 \leftarrow k_4 + cr_2 h_{id_s}$; $s_5 \leftarrow k_5 + ch_{id_s}$.
- The signature is $\sigma = (U_1, U_2, \prod_1, \prod_2, s_1, s_2, s_3, s_4, s_5)$.

– **Verify:** Given signature σ, message m, a set of identities $R = \{id_i\}_{i=1}^{k}$.

- Get $c' \leftarrow H_1(m||U_1||U_2||\prod_1||\prod_2||R)$.
- Check $\prod_1 \stackrel{?}{=} e(Q, U_1)^{-s_5} \cdot e(Q, P)^{s_2} \cdot e(Q_{pub}, P)^{s_1} \cdot e(Q_{pub}, U_1)^{-c'} \cdot e(P, Q)^{c'}$; $\prod_2 \stackrel{?}{=} e(P, U_2)^{-s_5} \cdot e(P, Q)^{s_4} \cdot e(P_{pub}, Q)^{s_3} \cdot e(P_{pub}, U_2)^{-c'} \cdot e(P, V)^{c'}$.

If above condition holds, the verifier accept the ring signature, else reject. It's easy to see that our ring signature scheme can be converted into an ad-hoc anonymous identification scheme [7], where a user can form ad-hoc groups and anonymously prove membership in such group. Although we use the collision-free accumulator in our scheme, the dynamic accumulators are also available such that the addition and deletion of members from the original group(or ring) are allowed.

It should be noted that Nguyen' scheme and our ring signature scheme actually are both non-interactive proof of the knowledge of (s_{id}, h_{id}, W) satisfying $e(h_{id}Q + Q_{pub}, S_{id}) = e(Q, P)$ and $e(h_{id}P + P_{pub}, W) = e(P, V)$, although Nguyen' scheme is much more complex than ours. The exact reason why our scheme could cut down the computation and size of the signature will be given in section 6.

5 Security Analysis

5.1 Unforgeability

theorem 1. *Assume that an adversary F has an advantage ε against our scheme when running in time t, asking q_{H_i} queries to random oracles $H_i (i = 0, 1)$, q_s signature queries to signature oracle. Then there is an adversary B to solve the k-strong Diffie-Hellman problem with probability $\varepsilon' \geq \frac{\varepsilon^2}{q_{H_1}^2} + \frac{(q_{H_0}+q_{H_1}+q_s)^2 - 4\varepsilon(q_{H_0}+q_{H_1}+q_s)}{q_{H_1}^2 \cdot 2^{l+1}} - \frac{1}{2^l}$ within a time $t' \leq 2t + q_s[10t_{exp} + 9t_p + (n+1)q_{mult}] + \mathcal{O}[(q_s(n+1)+q_H)(1+q_s+q_H)]$, where t_p denotes the require time for a paring evaluation, t_{exp} denotes the costs of an exponentiation in G_T, t_{mult} denotes the costs of a multiplication in G_2 and q_H denotes the maximum total number of queries to all random oracles.*

Proof: Here, we are ready to present the actual proof. On a security parameter l, Algorithm B takes as input $(p, G_1, G_2, G_T, \psi, P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^k Q)$ and aims to find a pair $(w^*, \frac{1}{w^*+\alpha} P)$ where $w^* \in Z_p^*$. We first show how to provide the adversary with a consistent view. In setup phase, it builds a generator $G \in G_1$ such that it knows $k-q (k > q)$ pairs $(w_i, \frac{1}{w_i+\alpha} G)$ for $w_1, \ldots, w_{k-q} \in_R Z_p^*$. It should be noted that q is the an upper bound on the number of elements to be accumulated and $k-q$ is the an upper bound on the number of extraction queries. To do so,

– It picks $w_1, \ldots, w_{k-q} \in_R Z_p^*$ and expands $f(z) \leftarrow \prod_{i=1}^{k-q}(w_i + z)$ to obtain $e_0, \ldots, e_{k-q} \in_R Z_p^*$ so that $f(z) \leftarrow \sum_{i=0}^{k-q} e_i z^i$.

– It sets generators $H \leftarrow \sum_{i=0}^{k-q} e_i(\alpha^i Q) = f(\alpha)Q \in G_2$ and $G \leftarrow \psi(H) = f(\alpha)P \in G_1$.

– For $1 \leq i \leq k-q$, B expands $f_i(z) \leftarrow \frac{f(z)}{z+w_i}$ to obtain $d_{i_0}, \ldots, d_{i_{k-q-1}} \in_R Z_p^*$ so that $f_i(z) = \sum_{j=0}^{k-q-1} d_{i_j} z^j$. Then compute $\sum_{j=0}^{k-q-1} d_{i_j} \psi(\alpha^j Q) = f_i(\alpha)P = \frac{f(\alpha)}{\alpha+w_i} P = \frac{1}{\alpha+w_i} G$. Then all pairs $(w_i, \frac{G}{\alpha+w_i})$ for $1 \leq i \leq k-q$ could be available.

– It sets $\alpha H \leftarrow \sum_{i=0}^{k-q} e_i(\alpha^{i+1}Q), \alpha^2 H \leftarrow \sum_{i=0}^{k-q} e_i(\alpha^{i+2}Q), \ldots, \alpha^q H \leftarrow \sum_{i=0}^{k-q} e_i(\alpha^{i+q}Q)$. let $t = \{p, G_1, G_2, G_T, e(\cdot,\cdot), \psi, G, H, \alpha H, \ldots, \alpha^q H\}$.

Now, B first chooses a random $u \in_R Z_p^*$ and generates an collision resistant accumulator $f \circ g$ as in section 2.2, then send $(l, t, u, f \circ g)$ to adversary F. To handle the oracle queries, B maintains two lists L_{H_0} and L_{H_1}. For simplicity, we assume that adversary F asks q_{H_0} distinct queries for q_{H_0} distinct identities. Simulates adversary's environment as follows:

– H_0 queries on an identity $ID \in \{0,1\}^*$: B selects a random index γ, where $1 \leq \gamma \leq q_{H_0}$ and fixes ID_γ as target identity. B first initializes a counter $index$ to 1 and answers $w \leftarrow w_{index} \in Z_p^*$ and increments $index$ if $ID \neq ID_\gamma$, else B returns a random $w_\gamma \in_R Z_p^*$. Add the tuple (ID, w_{index}) to L_{H_0}.

– H_1 queries on a tuple $\mu = (m||U_1||U_2||\prod_1||\prod_2||R)$: If μ has been defined in L_{H_1}, retrieves c from L_{H_1} and returns c to F, else chooses a new random $c \in_R Z_p^*$ and adds (μ, c) into L_{H_1}.

– Key extraction queries on ID: B recovers the matching pair (ID, w) from L_0 and returns the previously computed $\frac{1}{\alpha+w} G$ if $ID \neq ID_\gamma$, else B sets $bad_1 = true$, then aborts.

– Signature queries on a pair (m, ID, R): If $ID \neq ID_\gamma$, B proceeds according to the sign algorithm. This is possible for B knows the private key of ID. If $ID = ID_\gamma$, then:

- Chooses $U_1 \in_R G_1$, $U_2 \in_R G_2$, chooses pairwise different $s_1, \ldots, s_5 \in_R Z_p^*$.
- Selects a random $c \in_R Z_p^*$, then computes $\prod_1 \leftarrow e(H,U_1)^{-s_5} \cdot e(H,G)^{s_2} \cdot e(H_{pub},G)^{s_1} \cdot e(H_{pub},U_1)^{-c} \cdot e(G,H)^c$; $\prod_2 \leftarrow e(G,U_2)^{-s_5} \cdot e(G,H)^{s_4} \cdot e(G_{pub},H)^{s_3} \cdot e(G_{pub},U_2)^{-c} \cdot e(G,V)^c$. Then add the new tuple $(m||U_1||U_2||\prod_1||\prod_2||R,c_i)$ in L_{H_1}(if (μ,c_i) had already been defined in L_{H_1}, set $bad_2 \leftarrow true$, aborts).
- Returns $\sigma = (U_1, U_2, \prod_1, \prod_2, s_1, s_2, s_3, s_4, s_5)$ as the signature.

We have explained how to simulate F's environment in chosen-messages and chosen-identities attack. So, B runs the algorithm F_B(t) as described in section 2.3. In this way we get two forgeries σ_0 and σ_1 together with a set of identities R and message m. Let c_0 be the answer from the random oracle H_1 given to F in the first execution, i.e., h_J in F_B(t), and let c_1 be the second answer $h_J^{'}$. The forged signature $\sigma_0 = (U_1, U_2, \prod_1, \prod_2, s_1, s_2, s_3, s_4, s_5)$ and another signature is $\sigma_1 = (U_1^{'}, U_2^{'}, \prod_1^{'}, \prod_2^{'}, s_1^{'}, s_2^{'}, s_3^{'}, s_4^{'}, s_5^{'})$. Let $f_i \leftarrow \frac{s_i - s_i^{'}}{c_0 - c_1}$ for $i \in \{1, \ldots, 5\}$, then we get a tuple $(f_5, U_1 - f_1 G)$ satisfying $e(f_5 H + H_{pub}, U_1 - f_1 G) = e(H,G)$. It implies that $w^* = f_5$ and $\frac{1}{\alpha + w^*} G = (U_1 - f_1 G)$. We note that $w^* \neq w_1, \ldots, w_{k-q}$ with probability at least $1 - \frac{k-q}{2^l}$. If both forgeries satisfy the verification equation, B can proceed as in [27] to extract $\frac{1}{\alpha + w^*} P$ from $\frac{1}{\alpha + w^*} G$:

- Writes $f(z) = \prod_{i=1}^{k-q}(w_i + z) = \gamma(z)(z + w^*) + \gamma_{-1}$, where $\gamma_{-1} \in Z_p$ and $\gamma(z) = \sum_{i=0}^{k-q-1} \gamma_i z^i$.
- Then $\frac{f(z)}{w^* + z} = \frac{\gamma_{-1}}{w^* + z} + \sum_{i=0}^{k-q-1} \gamma_i z^i$. Since $G = \psi(H) = f(\alpha) P \in G_1$, as thus $\frac{1}{\alpha + w^*} G = \frac{f(\alpha)}{\alpha + w^*} P = \frac{\gamma_{-1}}{\alpha + w^*} P + \sum_{i=0}^{k-q-1} \gamma_i (\alpha^i P)$.
- It's easy to get $\frac{1}{\alpha + w^*} P = \frac{1}{\gamma_{-1}} [\frac{1}{w^* + \alpha} G - \sum_{i=0}^{k-q-1} \gamma_i (\alpha^i P)]$, then the tuple $(w^*, \frac{1}{\alpha + w^*} P)$ will be the answer of the k-strong Diffie-Hellman problem.

Let Pr[bad_i] denote the probability of the event that flag bad_i set to be true(fails in providing a consistent simulation). We bound the accepting probability acc as follows:

$$\begin{aligned} acc &\geq \varepsilon - Pr[bad_1] - Pr[bad_2] \\ &\geq \varepsilon - \frac{q_{H_0}}{2^l} - \frac{q_{H_1} + q_s}{2^l} \end{aligned}$$

The probability that algorithm B succeeds in getting the answer of the k-strong Diffie-Hellman problem is given by

$$\begin{aligned} \varepsilon^{'} &\geq \frac{frk}{q_{H_1}} \\ &\geq \frac{acc^2}{q_{H_1}^2} - \frac{1}{2^l} \end{aligned}$$

The running time $t^{'}$ is twice that of once execution in $F_B(t)$ plus the time needed to compute the solution of the k-strong Diffie-Hellman problem. The running time of once execution in $F_B(t)$ is the running time t of F plus the time needed to answer q_H random oracle queries and q_s signature queries, where q_H denotes the maximum total number of queries to all random oracles. We assume that t_p denotes the require time for a paring evaluation, t_{exp} and t_{mult} respectively denotes the costs of an exponentiation in G_T and a multiplication in G_2, and all other operations take unit time. Each random oracle query at most cause B to perform $\mathcal{O}(1 + q_H + q_s)$ unit-time operations. Each signature query involves at most $10t_{exp} + 9t_p + (n+1)q_{mult} + (n+1)\mathcal{O}(1 + q_H + q_s)$ operations, where n is the maximum number of identities of each signature query. Therefore, we have $t^{'} \leq 2t + q_s[10t_{exp} + 9t_p + (n+1)q_{mult}] + \mathcal{O}[(q_s(n+1) + q_H)(1 + q_s + q_H)]$.

5.2 Anonymity

In order to give the proof for anonymity, we present the proofs of our scheme's perfect zero-knowledge is enough. The simulator randomly chooses $U_1, U_2 \in_R \mathbb{G}_1$, $c, s_1, s_2, s_3, s_4, s_5 \in_R \mathbb{Z}_p$, then computes $\prod_1 = e(Q,U_1)^{-s_5} \cdot e(Q,P)^{s_2} \cdot e(Q_{pub},P)^{s_1} \cdot e(Q_{pub},U_1)^{-c} \cdot e(P,Q)^c$; $\prod_2 = e(P,U_2)^{-s_5} \cdot e(P,Q)^{s_4} \cdot e(P_{pub},Q)^{s_3} \cdot e(P_{pub},U_2)^{-c} \cdot e(P,V)^c$. We can see that the distribution of the simulation is the same as the real transcript. This completes the proof.

6 Some Remarks and Efficiency Comparison

In many scenarios, as pointed in [7], the group doesn't change for a long time or has a short description. So an appropriate measurement of ring signature-size does not need to include the group description. In this situation, both the signer and verifier need to perform a one-time computation proportional to the size of the ring, and get the gpk and gsk which allow them to produce/verify many subsequent signatures in constant time. It's obvious that the constant-size ring signature scheme will be much more efficient than the previous schemes which signature size proportional to the size of the ring in such scenarios. We note that even in large ad-hoc groups, the size of our signature scheme is much smaller than that of schemes which the size of signature linearly depends on the group size. To the best of our knowledge, Chow et al.'s scheme [14] is the most efficient one among all the ID-based ring signature schemes which the size of signature linearly depends on the group size. For the sake of comparison and concreteness, we fix $|G_1| = |Z_p| = 256$ bits for a security level equivalent to a 128-bit symmetric key for AES(cf.[28]). We conclude that our scheme has smaller size than Chow et al.'s scheme when the number of identities of the ring over 8.

The first constant-size ring signature scheme (DKNS04)

had been proposed by Dodis, Kiayias, Nicolosi and Shoup [7], after that, no more efficient constant-size ring signature scheme was found until the first secure ID-based ring signature scheme with constant-size signatures proposed by Nguyen [10]. Nguyen had compared his constant-size ID-based ring signature scheme with DKNS04 at the same level of security. The conclusion is that the signature size is very much smaller than that of constant-size ring signature scheme DKNS04 [7]. He also pointed out that in the future, when higher levels of security are required, this difference even grows much larger.

We now make a specific comparison between our scheme and that of Nguyen's. Due to our scheme actually is an improvement on the modified version of the scheme proposed by Nguyen [10], it seems that our scheme and Nguyen's scheme are implemented by the same elliptic curve or hyperelliptic curve over a finite field is reasonable. As shown in [10], we assume p is a 160-bit Jacobian of a hyperelliptic curve over a finite field with order p and compression techniques are used. G_T is a subgroup of a finite field of size approximately 2^{1024}. A possible choice of these parameters is that $G_1(G_1 = G_2)$ is derived from the curve $E/GF(3^l)$ defined by $y^2 = x^3 - x + 1$.

We summarize the result in Table 1. It's obvious that we greatly reduce the size of the signature, although the computational efficiency is improved slightly. It should be noted that our scheme has the same keys(GSK, GPK) with Nguyen's, so we don't list them(i.e. computation of keys and size of keys) in the table 1. Here we give our analysis of why our scheme could cut down the computation and size of the signature. As we mentioned is section 4, our ring signature scheme and Nguyen's scheme actually are both non-interactive proof of the knowledge of (s_{id}, h_{id}, W) satisfying $e(h_{id}Q + Q_{pub}, S_{id}) = e(Q, P)$ and $e(h_{id}P + P_{pub}, W) = e(P, V)$. In our signature $\sigma = (U_1, U_2, \prod_1, \prod_2, s_1, s_2, s_3, s_4, s_5)$, $\sigma_1 \stackrel{def}{=} (U_1, \prod_1, s_1, s_2, s_5)$ are used to prove the knowledge (s_{id}, h_{id}) satisfying $e(h_{id}Q + Q_{pub}, S_{id}) = e(Q, P)$, and $\sigma_2 \stackrel{def}{=} (U_2, \prod_2, s_3, s_4, s_5)$ are used to prove the knowledge (s_{id}, h_{id}) satisfying $e(h_{id}P + P_{pub}, W) = e(P, V)$. It looks like that there should be some extra "useful" data to set up a tough relation between σ_1 and σ_2 to build up resistance to attack whereby the adversary "tricked" generate a new valid signature use several valid signatures. Actually, this is what Nguyen's scheme did. However, it's easy to see that σ_1 and σ_2 share the same element $s_5 = k_5 + ch_{id}$ which is used to prove the relationships of (s_{id}, h_{id}, W). In our scheme, s_5, $\prod_1$ and $\prod_2$ share the same random number k_5. It implies that each signature has a unique random number, and it doesn't leave open the possibility of an attack whereby the adversary "tricked" generate a new valid signature use several valid signatures. So, the extra "useful" data is really redundancy. Our constant-size ring signature actually is the essence of the Nguyen's scheme, i.e. the remaining part of the Nguyen's scheme after cut extra "useful" data down. Then, there is a question: is there a possibility that cut something down from our signature scheme? Due to the way we construct the private key of user's, it looks like paring operation is necessary. If paring operation is necessary, it's really very hard to cut something down from our signature scheme.

Table 1: Efficiency comparison

scheme	signature size	mul	padd	pmul
Nguyen's	2,240 bits	7	15	20
Ours	1,440 bits	5	2	2

*mul, $padd$ and $pmul$ respectively indicate the number of multiplications, point additions and point scalar multiplications.

7 Conclusions

We have proposed an improved ID-based constant-size ring signature scheme based on Nguyen's scheme, which will be useful for implementation in large ring scenario. Our scheme outperforms in size of signature the previously proposed constant-size ring signatures and admits proofs of secure in the random oracle model based on a simplified and general Forking Lemma under the k-strong Diffie-Hellman assumption.

Acknowledgments

This work is supported by National Natural Science Foundation of China (60773035), The fund of Key Disciplinary of Computer Software and Theory(SZD0802-09-1), The research fund of key disciplinary of application mathematics (XZD0910-09-1).

References

[1] R.Rivest,A.Shamir, and Y.Tauman. How to Leak a Secret: Theory and Applications of Ring Signatures. in: Theoretical Computer Science. LNCS, vol.3895, Springer Berlin, 2006, pp.164-186.

[2] A.Shamir. Identity-Based Cryptosystems and Signature Schemes. in: Advances in Cryptology. LNCS, vol.196, Springer Berlin, 1985, pp.47-53.

[3] F.Zhang, and K.Kim. ID-Based Blind Signature and ring from pairings. in: Advances in Cryptology - ASIACRYPT 2002. LNCS, vol.2501, Springer Berlin, 2002, pp.629-637.

[4] J.Benaloh and M.de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Signatures. in: Advances in Cryptology - EUROCRYPT'93. LNCS, vol.765, Springer Berlin, 1994, pp.274-285.

[5] N.Barić and B.Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. in: Advances in Cryptology - EUROCRYPT'97. LNCS, vol.1233, Springer Berlin, 1997, pp.480-494.

[6] J.Camenisch, and A.Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. in: Advances in Cryptology - CRYPTO 2002. LNCS, vol.2442, Springer Berlin, 2002, pp.101-120.

[7] Y.Dodis, A.Kiayias, A.Nicolosi, and V.Shoup. Anonymous Identification in Ad Hoc Groups. in: Advances in Cryptology - EUROCRYPT 2004. LNCS, vol.3027, Springer Berlin, 2004, pp.609-626.

[8] Lan Nguyen. Accumulators from Bilinear Pairings and Applications. in: Topics in Cryptology - CT-RSA 2005. LNCS, vol.3376, Springer Berlin, 2005, pp.275-292.

[9] C Tartary, S Zhou, D Lin, H Wang and J Pieprzyk. Analysis of bilinear pairing-based accumulator for identity escrowing. Information Security, IET 2(4)(2008), pp.99-107.

[10] Lan Nguyen. Accumulators from Bilinear Pairings and Applications to ID-based Ring Signatures and Group Membership Revocation. http://eprint.iacr.org/2005/123.

[11] Chih-Yin Lin and Tzong-Chen Wu. An Identity-based Ring Signature Scheme from Bilinear Pairings. in: AINA'04, Also appear in http://eprint.iacr.org/2003/117.

[12] Javier Herranz and Germán Sáez. New Identity-Based Ring Signature Schemes. in: Information and Communications Security. LNCS, vol.3269, Springer Berlin, 2004, pp. 269-274.

[13] Sherman S.M.Chow, S.M.Yiu, and Lucas C.K.Hui. Efficient Identity Based Ring Signature. in: Applied Cryptography and Network Security. LNCS, vol.3531, Springer Berlin, 2005, pp.499-512.

[14] S.S.M. Chow,R.W.C. Lui, L.C.K.Hui, and S.M.Yiu. Identity Based Ring Signature: Why, How and What Next. in: Public Key Infrastructure. LNCS, vol.3545, Springer Berlin, 2005, pp.144-161.

[15] A.Fiat, and A.Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. in: Advances in Cryptology - CRYPTO'86. LNCS, vol.263, Springer Berlin, 1987, pp. 186-194.

[16] F.Zhang and Xiaofeng Chen. Cryptanalysis and improvement of an ID-based ad-hoc anonymous identification scheme at CT-RSA 05. Information Processing Letters 105(15)(2009) pp.846-849.

[17] D.Boneh, and X.Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups . Journal of Cryptology 21(2)(2008), pp.149-177.

[18] M.Bellare and G.Neven. Multi-signatures in the plain public-key model and a generalized forking lemma. in: Conference on Computer and Communications Security: Proceedings of the 13th ACM conference on Computer and communications security, ACM, New York, 2006, PP.390-399.

[19] J.Herranz and G.Sáez. Forking lemmas for ring signature schemes. in: Progress in Cryptology - INDOCRYPT 2003. LNCS, vol.2904, Springer Berlin, 2003, pp.266-279.

[20] Adam Bender, Jonathan Katz, Ruggero Morselli. Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. Journal of Cryptology 22(1)(2009), pp.114-138.

[21] A.Miyaji, M.Nakabayashi, and S.Takano. New explict conditions of elliptic curve traces for FR-reduction. IEICE Transactions on Fundamentals E84-A(5)(2001), pp.1234-1243.

[22] N.P.Smart and F.Vercauteren. On computable isomorphisms in efficient pairing based systems. http://eprint.iacr.org/2005/116.

[23] Fiore,D., Gennaro,R. Making the Diffie-Hellman Protocol Identity-Based. in: Topics in Cryptology - CT-RSA 2010. LNCS, vol.5985, Springer Berlin, 2010, pp.165-178.

[24] J.Camenisch, M.Kohlweiss and C.Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. in: Public Key Cryptography - PKC 2009. LNCS, vol.5443, Springer Berlin, 2009, pp.481-500.

[25] Paulo S.L. Barreto, B.Libert, N.McCullagh, J.J.Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. in: Advances in Cryptology - ASIACRYPT 2005. LNCS, vol.3788, Springer Berlin, 2005, pp.515-532.

[26] J.Herranz. Identity-based ring signatures from RSA. Theoretical Computer Science 389(1-2)(2007) pp.100-117.

[27] D.Boneh and X.Boyen. Short Signatures Without Random Oracles. in: Advances in Cryptology - EUROCRYPT 2004. LNCS, vol.3027, Springer Berlin, 2004, pp.56-73.

[28] ECRYPT II Yearly Report on Algorithms and Key Lengths (2010), http://www.ecrypt.eu.org/documents/D.SPA.13.pdf (revision 1.0, 30 March 2010).

Content-sensitive Approach for Video Browsing and Retrieval in the Context of Video Delivery: *VBaR* Framework

Phooi Yee Lau and Sungkwon Park
Media Communications Laboratory, Hanyang University, Seoul 133-791, Republic of Korea
E-mail: {laupy,sp2996}@hanyang.ac.kr

Keywords: browsing behavior, corner detection, image processing, video analysis, shot detection, low bit-rate channel, video delivery

Received: August 27, 2009

Information is doubling at a rate of once every few months and the rate of increase is growing. Video and media transport plays an important part of that growth. It has spurred the development of broadband access and slowly gained increasing prominence especially for multimedia-rich Internet contents. Operators are now under pressure to efficiently use available resources for delivering targeted contents in a reliable and consistent manner. The present work proposes a new strategy to select intra-coded frames that best represent the entire video. The proposed approach, tri-step approach, uses low-level image features to select representative key frames which enable random access to any part of the video sequence while avoiding the overhead incurred to transmit periodic intra-coded frames. The first step discriminates non-transition frames by checking adjacent frame characteristics using low level features. The second step refines our selection of key frame by dropping non-commonly browsed frames, being the non-informative frames. The final step verifies the remaining frames if they are far apart in time to maintain the efficiency of video delivery over low bit rate channel. Thirteen video sequences, obtained from MPEG-4 industry forum, are used in the experiments. A framework, named the Video Browsing and Retrieval (VBaR), is developed and it allows user to analyze input videos at their full control. Experimental results, using the proposed framework, show the ability to effectively select intra-coded frames in video delivery.

Povzetek: Predstavljeno je kontekstno odvisno indeksiranje in iskanje video vsebin.

1 Introduction

Today, streaming videos became a popular medium of entertainment and advertisement, with many websites providing direct links to videos from all over the internet. Viewers can now stream videos through a simple and searchable interface. What actually attracts the viewer's interest? Research shows that it is important to track the browsing behavior of viewers, both within and across videos, to obtain important clues to the effectiveness of a videos e.g. to an advertisement or to a trailer video preview [1-3]. Browsing behavior can be revealed by the choice of control used, e.g. fast-forwarding or pause, or by the enhanced user area which allows viewers to rate their video links and states their favorites.

Viewers, nowadays, have the privilege to download videos according to their preferences which sometimes, may require constant interactivity. Among the existing pressing issues resulted from the interactivities are: 1) the ability to playback or pause a video, and 2) the ability to guarantee transmittable and playable video using available resources. Because a video is essentially a collection of still images, presumably long (e.g. 120 minutes), one cannot tell what it is about without watching the whole video. Nonetheless, there has already been a wide spread of research interest in the delivery of selected content to any users, e.g. content-based video analysis or segmentation which has been intensively studied since the past decade [1-6]. In 1999, He et al. proposed a simple tracking of video usage by using server logs, which keeps information about the segments watched by viewers, to generate the summary of viewing behavior [5]. In 2001, Syeda-Mahmood presents a framework to continuously track viewers through HMMs, by observing their interaction with video based on deducing interest of viewers, a rather unusual approach [1]. Both methods studied the viewer's interest, irrespective of the video content or scene. At the content level, Zhu et al. present a system developed for content-based video browsing and retrieval, integrating audio-visual as well as text information and natural language understanding techniques analysis to extract scenes and content information of video documents, and to organize and classify video scenes [2]. In 2002, Chen and Yang presented an MPEG4 simple profile compatible approach for video browsing and retrieval over low bit rate channel, whereby, a new stream is generated from the video server to enable random access [6]. This method is oriented towards the viewers' browsing requirements, i.e. according to video content and the channel

characteristics, so that transmission overhead could be concealed. These two later methods, though studied video contents, ignore the viewer's interest with respect to the contents. Most importantly, these works show that there is a need to study viewer's interest and video content to enable video to be streamed across in the shortest possible time.

But, some questions still arise, such as, how to best represent the underlying content? Due to the need to have an efficient playback system, extracting representative key frames to describe the contents of a video will play a fundamental role for many video applications. Key frames are often used to define the starting and ending point of smooth transition; i.e. frames that best represent the underlying content. Key frames can be sampled randomly or uniformly at some definite time intervals. The main drawback of uniform sampling, while easy to implement, is that it may cause some important short clips without representative key frames while longer clips might have multiple frames with similar content, thus failing to represent the actual video [7]. One of the most popular ways to extract key frames is by adapting to the dynamic video content. Shot-based key frame extraction segments a video within a continuous period and uses the first frame of each shot as key frames. It heavily depends on the temporal dynamics in the scene. In this case, though the selected key frames can represent the entire video, it may miss the important part of video as it is not possible to select key frames that can represent the video content well [8-10]. On the other hand, we know that content-based video analysis has been studied intensively for the past decade to support a variety of applications, including video indexing and browsing [11-13]. For example, shot-based video segmentation is used to provide abstraction and delineation for video indexing, browsing and retrieval [14].

But, in video delivery, not only do we need to consider the above stated requirements, we also need to consider how to deliver videos in the shortest time in order to maximize available resources, especially videos that are simultaneously viewed by many subscribers or high quality multimedia contents such as HDTV and 3DTV. Therefore, video delivery services need to be natural, intuitive, and guided by user's viewing interest. It is, thus, important to optimally select intra-coded frames, e.g. incorporating viewer's browsing patterns and viewer's interest into the selection process, since the largest part of traffic growth over the next decade will be associated with video delivery [11, 15]. Failure to do so will prove to be very capital-intensive as service operator may be force to purchase new infrastructure components. One of the solutions is to represent the content of video using key frames and these key frames should be able to 1) index videos to help search for a particular scene 2) automatically identify user preference through preference modeling, 3) facilitate automatic movie content summarization.

In this paper, a new strategy, tri-step approach, is used to select intra-coded frames that best represent and describe the entire video well. The first step uses low-level image processing techniques used for shot detection (scene change); the second step, uses low-level image processing techniques to classify key frames into *informative* and *non-informative* (common browsing behavior); and the third step, uses temporal constraints to enable distinct distribution of key frames spanning the entire video (decoder limitations). The method is tested under two experimental set-ups: 1) different video sequence: sports video (with motion) and news (less motion), and 2) different scenario: multiple-scene video (abrupt scene change) and single-scene video (gradual scene change). The purpose is to allow user randomly access any part of the video sequence while avoiding the overhead incurred in transmitting periodic intra-coded frames, thus, maximizing the resources available.

The remainder of this paper includes: Section 2 that describes and discusses the proposed approach and algorithms for selecting intra-coded frames; Section 3 that outlines the Video Browsing and Retrieval (*VBaR*) framework; Section 4 that evaluates several experimental results; Section 5 that discusses and concludes the paper with future work.

2 Analysis of Content-sensitive Frames for Video Streams – Tri-Step Approach

A video can be considered as being made up of numerous snapshots, called frames or picture. The volume generated by digitizing all frames is too large for the video delivery channel. Among the much used video compression standards, aimed to reduce the amount of data required to store or transmit video while maintaining an acceptable level of video quality on low-bit-rate channels, are the ISO MPEG4 Part 10 of MPEG4 and ITU-T H.264. Low bitrate channels are constraint by a few important characteristics: 1) prevent transporting video frames that takes up resource, and 2) eliminate redundant data to be delivered to prevent channel congestion. Let us look at the role of image coding in video delivery. Intra-coding, often used to enable random access, refers to the individually compressed image without any reference to the other frames. On the other hand, the compression performance could be further improved when the temporal redundancy in video sequences is exploited. Known as the inter-coded frame, this coding refers to a frame that is coded based on some relationship with adjacent frames, i.e. proposed to exploit the interdependencies between adjacent frames. In reality, transmitting intra-coded frames (I-frames), compared to inter-coded frames (P-frames or B-frames), will increase the bit-rates greatly. Therefore, for video streaming in low bitrate channel, transmitting inter-coded frames are more favorable. In general, these three pictures types (I-, P-, and B-frames) are encoded with a group of pictures (GOP) length in the general reference encoder [16].

In a video itself, there are two potential issues which will affect the quality of video received. One is the delay in packet delivery which may prevent the video being

played smoothly, often referred as jitter or frame reversal. If jitter exceeds the buffer size in a device, video quality will degrade noticeably. The second is dropped frames. During congestions, significant numbers of packets are dropped; inter-coded frames are dropped first, followed by intra-coded frames, and this may also cause a noticeable degradation in video quality. In reality, network bandwidth is usually time-varying. If a user constantly searches a video for interesting video clips, then adapting a video to the start of all interesting clips could minimize video traffic as "watch" only frames are delivered. For example, Lee's work [15] adaptively assigned intra-coded frames by considering the scene changes and rapid motion in video sequences. Such type of strategy, i.e. placing intra-coded frames strategically to improve coding efficiently, is receiving increasing attentions in the video research community [17-18]. These works, for example, discuss how to efficiently place inter-coded frames for variety of video clips, especially those that do not contain frequent scene change. It shows that there is a need to reduce bandwidth consumption, especially crucial during peak-hours. It could be accomplished by avoiding the delivery of "non-watched" video data units to the set-top-box (STB), especially if users often quit video sessions prior to its completion. As we know, contents that are requested on demand, i.e. stored video contents, has recently emerged as a new business model. This business model redefines the subscriber-provider relationship, i.e. delegating content selection to the customer while the service provider manages the content distribution. As roughly 40% of sessions contain some interactivity, certainly there is a pressing need to deliver video data units efficiently in order to reduce performance bottlenecks, long delays and poor user experience for subscribers..

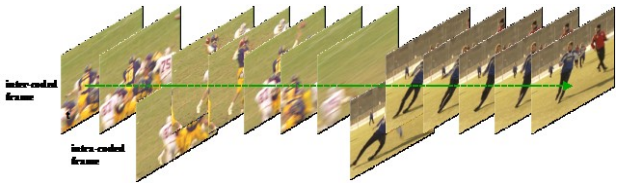


Figure 1: Input sequence: illustration of key frames for a video sequence.

The primary objective of this paper is to present a tool for selecting intra-coded frames that best represent and describe the entire video well. The proposed approach, using a tri-step approach, are able to select intra-coded frames by analyzing the video content for interesting scene that would attract viewers – see Figure 1. Our motivation is to enable user to retrieves requested video clips efficiently, without delivering too many redundant frames. We satisfy the following criteria in our approach: 1) reduce redundant frames that viewers will receive due the start of the clip is far from the periodically intra-coded frames; and 2) allow viewers to make browsing selection because a full video clip can themselves be long and needs to be segmented; and 3) adjust the video sequence to fully grasp the scene change, depending on the content, thus saving bandwidth and storage.

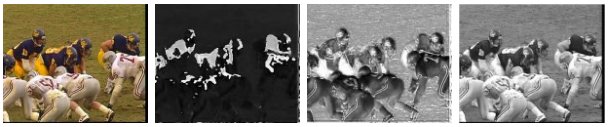
The proposed technique is divided into 3-step. The first step provides an efficient discrimination of input videos to select scene change frames.. The second step proceeds with a further evaluation of the selected frames in *Step 1*, verifying if these frames correspond to a set of common browsing behavior, i.e. *informative* frames. The final step studies the decoder limitation and verifies if the remaining frames are far apart in time to maintain the efficiency of video delivery over a low bitrate channel, i.e. determining the start frame of interesting video segments to enable distinct distribution of these frames spanning the entire video.

2.1 Step 1: Discriminate non-shot transition frames as candidate key frames

There are two type of scene change: 1) abrupt transition, which corresponds to a sudden change between two consecutive frames, and 2) gradual transition, which corresponds to a small change throughout a number of frames, detecting a transition frame could mean detecting the precise frame when the changes happen. The simplest feature that indicates a scene change is with low level features. They can be reliably used to indicate the starting position of a change in video sequences for shot boundary studies. The low level features applied in this paper are hue, saturation, value, and corners.

2.1.1 Preliminary Analysis

As discussed above, most images present high relativity with regards to some of its basic features except when scene change occurs, due to the presence of a new scene. This paper extracts the hue (H_F), saturation (S_F), value (V_F), and corner (C_F), as the set features to determine a scene change. Color saturation, hue and value can be easily obtained by converting the input videos to the HSV color space. The RGB color space is fundamentally different from the HSV color space as it separates the luminance from the color information (chromaticity) – see experimental results in Figure 2. Therefore, RGB color space image has to be converted to HSV color



(a) (b) (c) (d)

Figure 2: An input image from 'football' sequence: (a) Original CIF (352 x 288) video frame; (b) HSV - Hue feature; (c) HSV – Saturation feature; and (d) HSV – Value feature.

space by normalizing the RGB values – see equation (1). The H component, S component and V component can be obtained using equation (2), equation (3), equation (4) and equation (5), respectively.

$$r = \frac{R}{R+G+B},\ g = \frac{G}{R+G+B},\ b = \frac{B}{R+G+B} \quad (1)$$

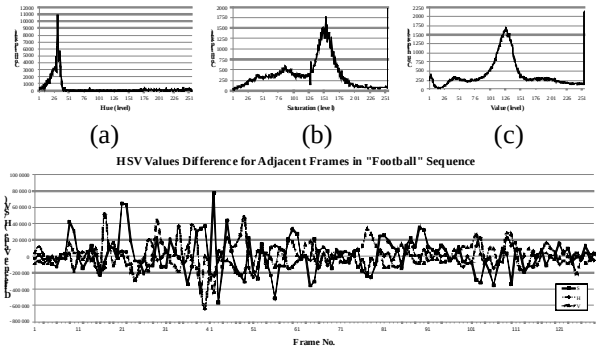
$$H = \left(\cos^{-1}\left\{ \frac{0.5\times[(r-g)+(r-b)]}{\left[(r-g)^2+(r-b)(g-b)\right]^{\frac{1}{2}}} \right\} \right) \times \frac{180}{\pi}, if\ b \le g \quad (2)$$

$$H = \left(2\pi - \cos^{-1}\left\{ \frac{0.5\times[(r-g)+(r-b)]}{\left[(r-g)^2+(r-b)(g-b)\right]^{\frac{1}{2}}} \right\} \right) \times \frac{180}{\pi}, \text{if } b > g \quad (3)$$

$$S = (1-3\times\min(r,g,b))\times 255 \quad (4)$$

$$V = (\max(r,g,b))\times 255 \quad (5)$$

The hue (H) varies from 0° to 360°, and is here quantized into 12 color intervals, each spanning 30°, [red, orange, yellow, yellow-green, green, green-cyan, cyan, cyan-blue, blue, blue-magenta, purple and magenta-red]. Saturation, S, is the intensity of specific hue, whereby, highly 'attractive' areas typically have vivid colors; therefore the color saturation is high. The V, also known as value, is also used as it allows selecting the highest pixel values and visually corresponding to brighter image areas. The HSV value seems to be a good cue to discriminate the non-short transition frames as they almost do not change with respect to small scene change. The HSV value that is computed for each video frame and the difference of these value based on adjacent frames are plotted onto chart – see Figure 3 (a/b/c) and Figure 3 (d), respectively. Later, a threshold is applied for each frame to determine if there are the scene changes.

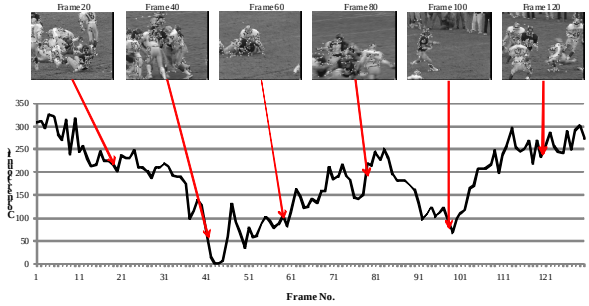


(a) (b) (c)

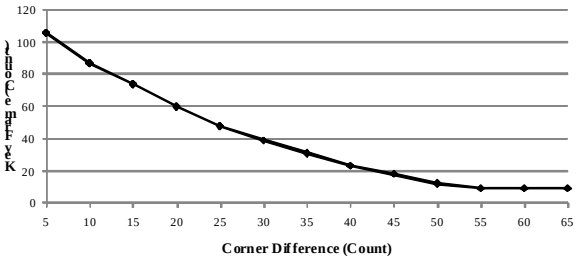
(d)

Figure 3: An input image (Figure 2) pixel count for: (a) Hue-, (b) Saturation-, (c) Value-level, and (d) Difference between adjacent frames for HSV value.

Corners have been traditionally used to detect or track motion. Here, we used it to assist us in tracking transition frames. This paper adopts a curvature-based corner detector which detects both fine and coarse features accurately at low computational cost [5]. It utilizes global and local curvature properties and balances their influence when extracting corners, allowing different parameters to be automatically determined for different images, different curves, and different kinds of corner candidates. The corner detector's step-by-step details can be found in [5] and experimental results are shown in Figure 4. Figure 4 (a) shows the corners count for "Football" sequence with sample frames. The number of corners detected for each consecutive frame is later threshold to determine if there is a scene change. Figure 4 (b) shows the number of key frames selected using different threshold values for the "Football" sequence.



(a)



(b)

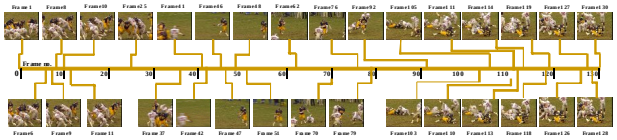
Figure 4: (a) Corners count for "Football" sequence with sample frames (b) Total key frames selected with different corner difference value.

2.1.2 Shot detection and elimination of frames

From exhaustive tests, it was observed that a scene change incurs more abrupt changes in the HSV values and the corner count, and is illustrated in Figure 3 (d) and Figure 4, respectively. In practice, one feature alone could not identify clearly the position of a scene change. Due to this, a more reliable shot detection can be obtained by combining the results coming from a set of features, discussed earlier in subsection 2.1.1. Votes are taken from each feature which favours the scene change detection and decisions are made based on a majority vote, according to equation (6). Experimental examples are shown in Figure 5.

$$F_{Step1} = \begin{cases} Cut: & > \text{two vote from } H_F S_F V_F C_F \\ Non-cut: & otherwise \end{cases} \quad (6)$$

Video frames that are classified as transition frames will be kept for further processing. This initial discrimination is conducted to allow fast analysis of a complete video and to discard non-scene transition key frames. Figure 5 shows key frames selected in *Step 1* for "Football" sequence. Notice, however, some frames shown in Figure 5 (a) are either appearing too close together in time or do not provide sufficient information about the scene, even though they represent a distinctive scene change. These frames will be further verified in the subsequent step and its initial classification revised.



(a)

(b)

(c)

Figure 5: (a) Candidate key frames selected in *Step 1* (b) Frame 42 (c) Frame 47.

2.2 Step 2: Discriminate *non-informative* candidate key frames

Tracking the browsing behavior of viewers are valuable, not only for forecasting future visit patterns for content-sensitive e-commerce, but are also useful in the generation of fast previews of videos for easy pre-download [20-22]. If we could learn the center of interest to which the viewers are attracted, it could be used to help viewers find and select appropriate content from the vast amount of data available. Viewers often look for ways to quickly grasp the content by visual fast-forwarding and the ability to determine each interesting clip would enable distinct browsing states, distributed over the entire video. Whole video, which is sometimes too long, needs to be segmented into shorter and more interesting segments. These segments are often determined by the statistical inference over extensive historical samples, i.e. preferred content or preferred video clips. The basic idea is to evaluate the content using low-level features to enable the user to grasp the potential knowledge about the content to be browsed. But, what content would represent an interesting video clips?

Syeda-Mahmood's work [1] grouped three browsing behaviors by summarizing various potential states that viewers could be in: 1) curious, 2) aimless browsing, and 3) explicit queries/search [1]. The first two states are viewers with no specific agenda and they generally do not capture the actual browsing behavior i.e. passive viewers. The third state requires urgency and viewers tend to look for something intriguing, which often state the browsing patterns – active viewers (age under 40s). To assist these active viewers, the authors provide video abstracts, i.e. representative key frames, to represent the video's content (aka summary shots), to denote each shot. They then reclassify these shots into two categories – interesting and mundane. So, the question is how we can classify interesting shot, beyond the browsing behavior? Rich content shots, e.g. shots which include many details and garner most viewers' attention are considered interesting – see Figure 6.

The selection of interesting shots may be associated with the search for content which could elicit viewer's behavioral patterns in browsing. We relate rich content shots, i.e. interesting video frames, with image details which could be obtained through low-level image processing techniques. For example, we obtained the rough contour of objects, which strongly relates to the image content itself, by studying the edges and the corners of an input image. But analyzing the whole image could not specifically represent the content details. As such, here, we adopted the block based approach. The following describes how interesting frames (aka *informative*) can be extracted.

(a)

(b)

Figure 6: Sample images – (a) rich content shot, and (b) and mundane shot.

At first, a block-based analysis of the image's edge and corner is performed. Each image is divided into 16 x 16 pixel blocks, and for every block, B_i, the edge, $I_E(B_i)$ and corner, $I_C(B_i)$ are calculated. The images are discriminated into *informative* and *non-informative* using a 5-level classification system. Each level is represented using a gray level code, as shown in Figure 7. The decision on the *informative* level can be expressed as follows:

- Level 0 (Non-*informative* block):
  If $I_E(B_i) | I_C(B_i) = 0$
- Level 1 (Low *informative* block):
  If $0 < I_E(B_i) <= n$ or $0 < I_C(B_i) <= m$
- Level 2 (Average *informative* block):
  If $n < I_E(B_i) <= 2n$ *or* $m < I_C(B_i) <= 2m$
- Level 3 (High *informative* block):
  If $2n < I_E(B_i) <= 3n$ or $2m < I_C(B_i) <= 3m$
- Level 4 (Extreme *informative* block):
  If $I_E(B_i) > 3n$ or $I_C(B_i) > 3m$

Informative level	0	1	2	3	4
Gray level					

Figure 7: Representation of *informative* level using 5 different gray levels.

m and *n* is a margin of safety, here set to 1 and 10, based on extensive experimental testing. *k* is the total block in a video frame. The image is then discriminated into the *informative* (*non-informative*) if more than (less than) 10 blocks are *Level 3* - see equation (7). Candidate key frames from *Step 1* are further classified into *informative* and *non-informative* frames. *Informative* frames are selected as *Step 2* candidate key frames and will be further classified and analysed – see Figure 8.

$$F_{Step\,2} = \begin{cases} Informative: & if\,(sum(B_i > Level\,2) > 10) \\ Non-informative: & otherwise \end{cases} \quad i = 1,2,3...k \qquad (7)$$

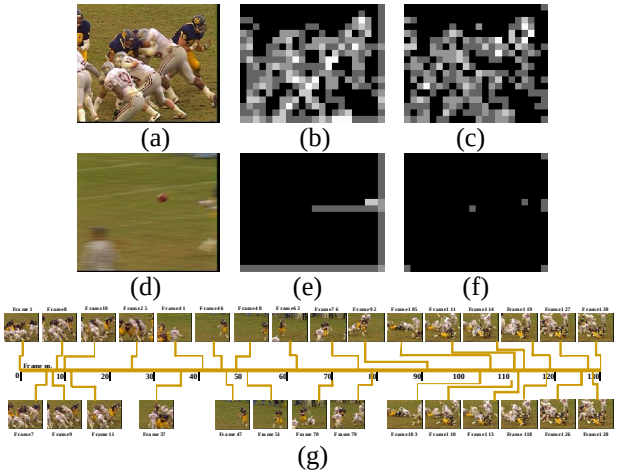


Figure 8: Input images (a and d) with its respective *informative* level - edge blocks (b and e) and - corner blocks (c and f), and (g) *Step 2* candidate key frames after discriminating non-*informative* frame.

2.3 Step 3: Applying temporal constraints to verify selected key frames

Only frames labelled as *informative* (in *Step 2*) are further analyzed in *Step 3*. This final step verifies if the remaining frames occur far apart in time to maintain the efficiency of video delivery over low bitrate channel. This is because there is a significant overhead associated with the transmission of periodic intra-coded frames, as intra-coded frames typically require 5-10 times as many bits as inter-coded frames [23-24]. For videos, during normal speed playback, intra-coded frames do not provide additional functionality, and this overhead should be avoided. Conversational applications, on the other hand, do not require frequent transmission of intra-coded frames, and often placed infrequently to allow bandwidth saving.

It is typical to have at least two intra-coded frames per each second of video in order to allow decoder to begin decoding with sufficient frequency. For example, we have an intra-coded frame every 15th frame on 29-30Hz systems, or every 12th frame on 24-25Hz systems, insinuating that a transition frame could take place in a window of 10-30 frames. If an intra-coded frame (after *Step 2*) surpasses this window, which may represent a gradual transition between the two shot, and if channel error propagation occurs, the inability to correct the error due to the unavailability of intra-coded frame will degrade the video quality at the receiver. On the other hand, more intra-coded frames are needed for a more frequent and pronounced scene change activity in a video sequence.

Figure 9: Examples of *Step 3* final key frames selection: (a) select (b) dropped (c) insert additional key frames.

Here, we proposed using temporal constraint to limit the GOP size (or window), a minimum of five to a maximum of thirty, using one-per-window (intra-coded frame) policy. This window allow us to select representative key frames that describe the content of the video, whether in abrupt scene transition or in gradual scene transition. Figure 9 (a) shows that frame number 14 (*Step 2*) or video frame number 62 has a GOP of 11 while Figure 9 (b) shows that frame number 26 (*Step 2*) or video frame number 119 has a GOP of 1 – being too small, and Figure 9 (c) shows that frame number 2 (*Step 2*) or video frame number 98 has a GOP of 97 – being too big. In *Step 3*, when the GOP size is too small - as shown in Figure 9 (b) - the intra-coded frame will be dropped. On the other hand, when the GOP size is too big – as shown in Figure 9 (c) – additional intra-coded frame will be inserted based on one-per-window policy. This is to ensure the videos have an intra-coded frame that are placed a second apart in order to control random accesses, at least, to every second – see experimental results in Figure 10. The remaining frames are named key frames.

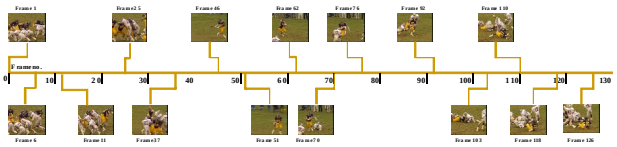


Figure 10: Selected key frames after temporal constraints in *Step 3*.

3 Content-sensitive Video Delivery System

3.1 Materials

The analysis was run on a PC platform, using thirteen (13) YUV format test video sequence from MPEG-4 industry forum (“http://www.m4if.org/resources.php”). The dataset is composed of thirteen sequences with different scenarios ranging from news, foreman, tennis, soccer, football, hall, coastguard, harbour, mobile, city skyline, crew, bus, and multiple scenarios - as shown in Table 2. Each selected video sequence has different number of frames and with 352 x 288 (CIF) resolutions.

3.2 *VBAR* Framework

The analysis framework, Video Browsing and Retrieval (*VBaR*) framework, is a research-oriented framework

developed in the Media Communication Laboratory at Hanyang University to analyze video delivery performances. The framework was developed using MATLAB®GUIDE tools to achieve a user-friendly interface, as shown in Figure 11. At the moment, the system can analyze targeted videos, i.e. to select a set of representative key frames for a video

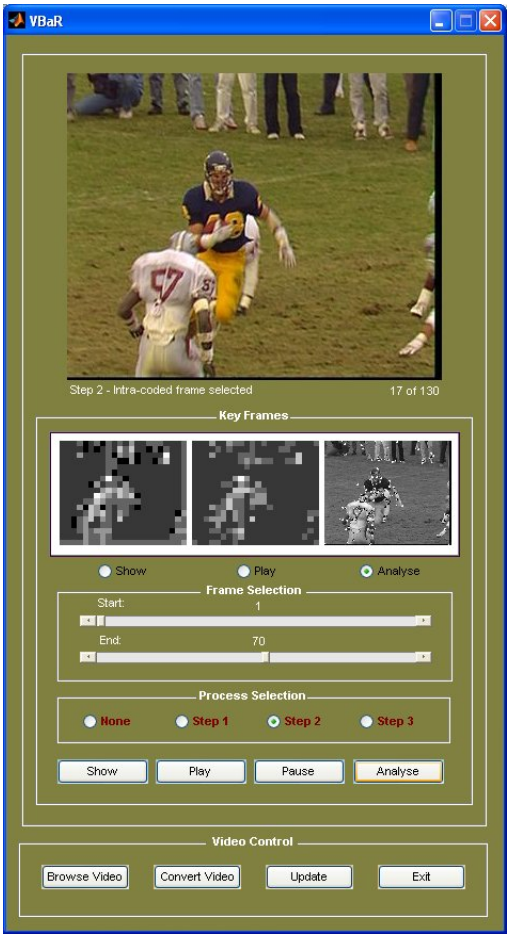


Figure 11: Graphical user interface for *VBaR* framework.

3.3 Using the GUI

Table 1: Standard operating procedure for *VBaR*

Video Control: Select input video
1) ***Browse video*** for the *.yuv video
2) ***Convert video*** to extract the video

Frame Selection: Select the start and end frame for analysis
1) Select ***Start*** frame by clicking at the bar
2) Select ***End*** frame by clicking at the bar

Control Key Selection: Select the control options
Options: *Show/Play/Pause/Analyze*
Show : Show an image
Play : Play a video sequence from *Start* frame to *End* frame
Pause : Pause the execution
Analyze : Analyze the selected process – *Step 1/Step 2/Step 3*

Analyze Selection: Select the process option
Option: *None/Step 1/Step 2/Step 3*
None : No process
Step 1 : To select shot detection
Step 2 : To eliminate non-*informative* frames
Step 3 : To dropped or insert frames

**Results are shown in the main window and stored to file

The user interface supports the configuration of the desired task in a user friendly way. The input images or video, and the analysis results are shown in Figure 11. The software is able to analyze grayscale and color video sequences. The user interaction with the analysis software using the GUI follows the main steps listed in Table 1.

4 Experimental Results

The proposed algorithm is evaluated using thirteen selected video sequences, each with different number of video frames. The selected video sequences evidence a range of scenarios. Experimental results obtained using *VBaR* framework is presented. In these videos, experimental results are evaluated: 1) key frame analysis, 2) GOP analysis, 3) performance – speed, 4) comparison among different sequences, and 5) performance comparison.

Table 2: Video sequence details

Video 1: News – camera static, slow motion and few scene changes
Video 2: Foreman – camera moving, slow motion and few scene changes
Video 3: Stefan – camera moving, fast motion and many scene changes
Video 4: Soccer – camera moving, fast motion and many scene changes
Video 5: Football – camera moving, fast motion and many scene changes
Video 6: Hall – camera static, slow motion and few scene changes
Video 7: Coastguard – camera moving, slow motion and few scene changes
Video 8: Harbour – camera static, fast motion and few scene changes
Video 9: Mobile – camera moving, fast motion and few scene changes
Video 10: City – camera moving, fast motion and many scene changes
Video 11: Crew – camera moving, slow motion and few scene changes
Video 12: Bus – camera moving, fast motion and many scene changes
Video 13: Multiple scenes – combination of Stefan/Football/Soccer/News

4.1 Key Frame Analysis

This key frame analysis counts the number of frames selected/dropped in each process, e.g. *Step 1*/*Step 2*/*Step 3* – as shown in Table 3, VideoFile 1, VideoFile 2, and VideoFile 3.[1] For example, in *Video 5*, a total of 15 frames were selected as key frames, i.e. that is average GOP size of about 9, compared to *Video 1*, with 10 key frames and average GOP size of about 30. The reason is because *Video 5* has frequent scene change (usual for sports video). Notice that when there are few scene changes – e.g. *Video 1* and *Video 6*, only few key frames are selected in *Step 1* and *Step 2*. *Step 3* compensates this problem by inserting appropriate key frames based on one-per-window policy. In *Video 1*, a total of 6 key frames are inserted in *Step 3* - as shown in Table 3. Scene changes for sports video are more frequent. Due to this, quite a number of frames were selected as candidate key frames in *Step 1* and *Step 2*; with many having a GOP size as small as 1. In this case, the framework automatically dropped candidate key frames when the GOP is smaller than 5. For example, for *Video 3*, out of the 20 candidate key frames, 12 frames are selected as

[1] Please contact first author for VideoFile*.

key frames in *Step 3*; i.e. 8 candidate key frames are dropped.

We notice that *Video 3* and *Video 8* have the smallest average GOP size, even though both videos observe different camera settings and scene change details. In *Video 13*, for example, where multiple scenes change occurs ("Stefan"→ "Football"→ "Soccer"→ "News"), our framework is robust in selecting key frames – see VideoFile4.

Table 3: Step-by-step key frame(s) selection

Video No.	Total Frame	*Step 1*	*Step 2*	*Step 3*	Ave GOP size
Video 1	300	4	4	10	30
Video 2	150	9	9	10	15
Video 3	90	20	20	12	7
Video 4	150	12	12	9	16
Video 5	130	31	30	15	8
Video 6	300	4	4	10	30
Video 7	300	22	22	18	16
Video 8	150	31	31	19	7
Video 9	150	17	17	12	12
Video 10	150	22	22	13	11
Video 11	150	31	31	15	10
Video 12	75	18	18	9	8
Video 13	300	34	34	21	14

To summarize, in order to meet bandwidth and buffer size limit, especially if service providers want to guarantee transmittable and playable video, there exist two important contribution of our work: 1) for video sequences with few scene change, the system compensates by inserting key frames, and 2) for video sequences with frequent scene change, key frames are dropped.

4.2 GOP Analysis

Table 4: GOP size for each video sequence

Video / GOP No.	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	30*	30*	6	15	5	30*	8	5	17	21	14	6	14
3	30*	30*	8	13	5	30*	30*	5	8	8	16	15	12
4	30*	30*	7	6	14	30*	9	14	12	14	6	8	10
5	30*	9	5	25	12	30*	12	5	30*	14	6	5	5
6	30*	9	7	24	9	30*	7	15	9	15	30*	12	6
7	30*	7	7	12	5	30*	7	14	8	9	23	6	12
8	30*	9	8	20	11	30*	17	5	14	13	6	14	15
9	30*	13	8	30*	8	30*	30*	8	9	9	10	8	5
10	30*	8	5		6	30*	27	5	21	28	6		10
11			13		16		30*	7	7	5	9		13
12			15		11		17	9	14	8	5		6
13					7		10	5		5	7		9
14					8		7	7			5		8
15					8		10	14			6		5
16							30*	5					16
17							30*	5					30*
18							18	14					30*
19								6					30*
20													13
21													30*
FIX GOP (15):	20	10	6	10	9	20	20	10	10	10	10	5	20
Difference	-10	0	+6	-1	+6	-10	-2	+9	+2	+3	+5	+4	+1

*key frames inserted

We analyze the GOP size for all video sequences using our proposed *VBaR* framework. We compared the GOP size and its frequency for all video sequences, and experimental results are shown in Table 5. To begin with, there are more key frames selected for *Video 3*, *Video 5*, *Video 8* and *Video 11* (GOP size small) compared to *Video 1* and *Video 6* – due to the number of scene change detected. More significantly, we manage to show the robustness of our proposed framework for all types of video: sequences that have few scene change (*Video 1*) or sequences that have frequent scene change (e.g. *Video 13*) – as shown in Table 4. Our proposed method, i.e. variable GOP, compared to fixed GOP leads to significant reduction in the overhead involved in the transmission of intra-coded frames – shown in Table 4 (*Video 1* and *Video 6*). The GOP size, for all thirteen video sequences, range from 5 to 18, with few intra-coded frames of higher than 18 – see Figure 12. Those GOPs that is higher than 20 are mainly for *Video 1*, *Video 2* and *Video 6* - having fewer scenes change. Experimental results show that our proposed approach (variable GOP) is very useful to save bit rate for video with fewer scene change (e.g. romance or documentary film) and to compensate frequent scene change video (e.g. action film) with lower GOP.

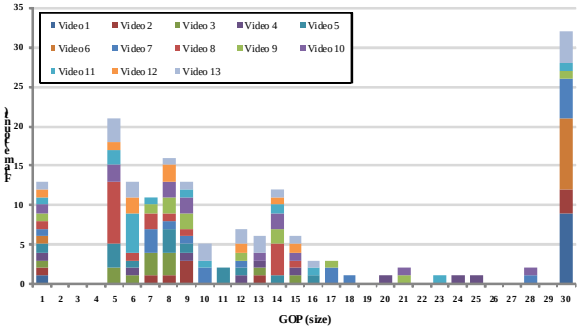


Figure 12: GOP statistics for thirteen different video sequences.

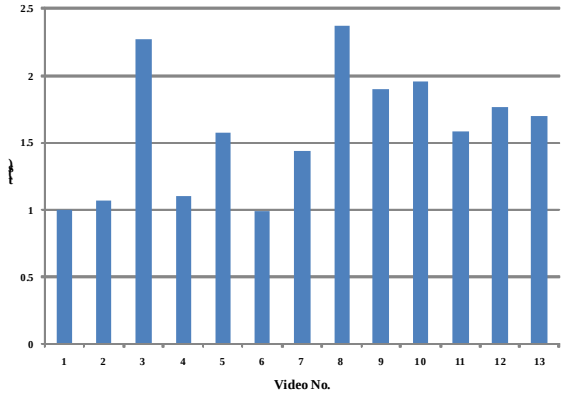
4.3 System Performances

The performance of the proposed framework is evaluated for all thirteen selected video sequences. Comparison of computational speed on the PC platform helps to identify the potential implementation of the method proposed. The algorithms are tested on an Intel E7300 Core 2 Duo 2.66GHz with 2GB of RAM – PC platform, using MATLAB®. The computation time is calculated for every *Step* – shown in Table 5. The time durations are represented in second (s).

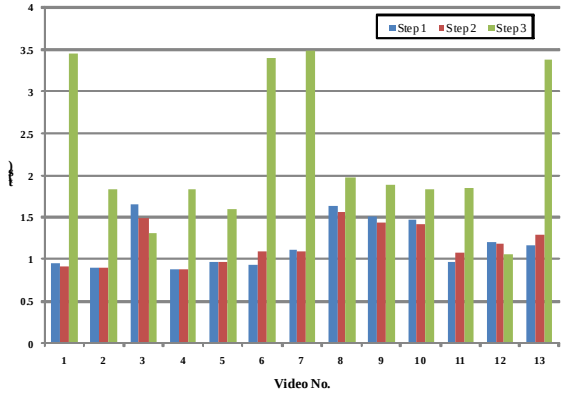
Table 5: Comparison of Test Speed (in seconds)

Process	*Step 1*	*Step 2*	*Step 3*	Total time	Average/frame
Video1	285.8	3.7	13.8	303.3	1.01
Video2	135.5	8.1	16.6	160.2	1.06
Video3	149.3	29.8	26.1	205.2	2.28
Video4	133.0	10.6	22.1	165.7	1.10
Video5	126.5	30.3	48.5	205.3	1.57
Video6	279.4	4.4	13.6	297.4	0.99
Video 7	333.6	24.1	76.9	434.6	1.44
Video 8	245.9	48.8	61.4	356.1	2.37
Video 9	228.9	24.6	32.2	285.7	1.90
Video 10	221.0	31.6	40.4	29	1.95
Video 11	146.7	33.4	57.6	237.7	1.58
Video 12	91.5	21.5	19.2	132.2	1.76
Video 13	351.2	44.1	115.4	510.7	1.70

Video 6, which requires an average of 0.93 seconds/frame in *Step 1* and 1.1 seconds/frame in *Step 2*, while 3.4 seconds/frame in *Step 3*, has one the fastest processing speed – shown in Figure 13 (b). Figure 13 (a) also shows that complex video such as *Video 8* requires higher computation time. Nevertheless, experimental result shows that the time required for selecting key frame depends on the content in each video sequence, highest being 2.37 seconds/frame and lowest being 0.99 seconds/frame, relatively short.



(a)



(b)

Figure 13: Average processing time (seconds) per frame using *VBAR* for each (a) Video (b) *Step*

4.4 Comparison with different sequences

Table 6: Comparison among different types of video sequences

Features		# frame(s)	# key frame(s)	Average GOP
1.	Camera setting			
a.	Camera static	750	39	19
b.	Camera moving	1645	133	12
2.	Activities			
a.	Slow motion	1350	63	16
b.	Fast motion	1045	109	9
3.	Scene change			
a.	Few scene changes	1500	94	16
b.	Many scene changes	895	78	11

We compared different type of video sequences characteristics to ascertain the potential of our proposed approach. We classified all videos into different settings: (1) camera settings, (2) activities, and (3) scene changes. We analyzed the number of key frame selected for each category. Experimental results are presented in Table 6. Experimental results show that fast motion video sequences have lower average GOP size, about 9, or more key frames, about 109. Also, the number of key frames selected for videos with frequent scene change and moving camera are much lower. Overall, in terms of video representation efficiency, the experiment results demonstrate the performance of the proposed *VBAR* framework.

4.5 Comparison with other work

We compared four other key frames selection methods to ascertain the potential of our proposed approach – shown in Table 7.

Table 7: Comparison with other key frames extraction methods

Features	Andreas [25]	Liu [11]	Chau [26]	Ouyang[27]	*VBAR*
1. Key frame selection	Temporal(T)	Color(C)	Visual (V)	Motion	T / C/ V
2. Application	Skimming	Segmentation	Efficiency	Skimming	Skimming
3. Test Videos	ND**	9 video	3 videos	18 videos	13 videos
4. Total Frames	ND**	1397	ND**	3912	2395
5. Analysis***	FrB	FrB	BlB	FrB	FrB/BlB
6. Temporal Constraint	Yes	No	No	No	Yes
a. Min (Reported)	ND**	None	1	None	5
b. Max (Reported)	ND**	None	13	None	30

*Color/Visual/Temporal **ND -Not described ***FrB–Frame-based/BlB–Block-based

Our proposed approach not only outperforms these methods, but also has three other advantages. Firstly, it applies temporal constraints for key frame selection (*Step 3*). Our proposed temporal constraints prevent key frames from occurring too far apart in time or too close together in time. Although Andreas's work [25] uses temporal constraints to filter out unsuitable clusters, its method only uses temporal constraints to cluster the frames in a video and to select a representative frame for each cluster in order to prevent selecting key frames which appear too close together in time. Chau's work [26], on the other hand, gives a set of key frame based on an objective model of visual content flow, but it failed to consider the temporal constraints in its method. As such, there exist many video shots having a GOP size as small as 1 to a maximum size of 13. Secondly, Liu's work [11] which extracts key frames based on parametric Gaussian Mixture Model (GMM) that are associated with video objects but did not report the computation time required. In comparison, the *VBAR* has lower computational complexity, as reported in Table 5, and a simpler representation. It is easy to implement and comes with a user-friendly interface. The *VBAR* system is able to generate summary for quick browsing of video content, i.e. dynamically generate flexible and effective summary.

Though Ouyang's work [27] proposed a similar interactive model of key frame selection, the graphical user interface does not describe and display how the parameters are selected and managed for the key frame selection.

5 Conclusions

In this paper, we assume that the intra-coded frame assignment and the intra-quantization parameter is taken care by the decoder. We focus on selecting intra-coded frames that best represent and describe the entire video well. We studied viewer's browsing patterns and incorporate viewer's interest into the key frame selection. Our major contributions are threefold. First, our proposed method manage to reduce the number of intra-coded frames by optimally selecting key frames using the proposed tri-step approach, being a simpler and faster alternative. Second, we proposed to limit the GOP size to allow bandwidth saving and to avoid video degradation due to unavailability of key frames. Third, we implement our framework on *VBaR* on a software platform to enable users to analyze selected video using a user-friendly graphical user interface.

The proposed *VBaR* framework is evaluated based on four important aspects: 1) key frame analysis, 2) GOP analysis, 3) system performance, and 4) performance comparison. The framework has been tested under various scenarios: sports video sequences, news video sequence, and other motion-filled sequences. The results were nevertheless promising: a consistent ability to divide a video into different clips using representative key frames. We anticipate that by delivering interesting video clips to subscriber, i.e. select representative key frames for all targeted content, we could minimize interactivity and eventually, reduce performance bottlenecks, long delays and poor user experience for subscribers, especially when roughly 40% of sessions contain some interactivity.

Future plans include testing the framework by using videos taken from different formats and scale. For the specific case of the intra-frame selection, where quantization parameter (QP) selection and transcoding technique selection can further reduce channel bandwidth utilization within a guaranteed picture quality, there is still considerable amount of work ahead. Nonetheless, the usage of *VBAR* in the present conditions is possible, providing a means to divide an entire video into video data units using representative key frames, and deliver only "watch" video data units to the STB, thus, enabling tremendous savings in bandwidth.

Acknowledgement

This work was supported by research fund of Hanyang University (HYU-2006-I).

References

[1] T. Syeda-Mahmood and D. Ponceleon, "Learning video browsing behavior and its application in the generation of video previews", in *Proceedings of the Ninth ACM International Conference on Multimedia*, Ottawa, Canada, 30 September – 05 October, 2001, pp. 119-128.

[2] Y. Zhu and D. Zhou, "Video Browsing and Retrieval Based on Multimodal Integration", in *Proceedings of the 2003 IEEE/WIC*, 13 – 17 October, 2003, pp.650.

[3] H. J. Zhang, J. Wu, D. Zhong and S. W. Smoliar, "An integrated system for content-based video retrieval and browsing", *Pattern Recognition*, vol. 30, no. 4, pp. 643-658, 1997.

[4] T. Syeda-Mahmood, S. Srinivasan, A. Amir, D. Ponceleon, B. Blanchard, D. Petkovic , "CueVideo: a system for cross-modal search and browse of video databases," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, vol.2, no., pp.786-787 vol. 2, 2000

[5] L. He, E. Sanocki, A. Gupta, and J. Grudin, "Auto-summarization of audio-video presentations", in *Proceedings of the Seventh ACM international Conference on Multimedia (Part 1)*, Orlando, Florida, United States, 30 October – 05 November 1999, pp. 489-498.

[6] C. Chen and Z. Yang, "MPEG4 Compatible Video Browsing and Retrieval over Low Bitrate Channel", in *Proceedings of the Third IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia information Processing*, 16 – 18 December, 2002, pp. 1221-1226.

[7] M. Mills, "A magnifier tool for video data", in *Proceedings of ACM Human Computer Interface*, pp. 93-98, May 1992.

[8] W. Wolf, "Key frame selection by motion analysis", in *Proceedings of the 21st International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 1228-1231, 1996.

[9] H. Zhang, Z. Liu, H. Zhao, G., Cheng "Recognizing Human Activities by Key Frame in Video Sequences", *Journal of Software*, vol. 5, no. 8, pp. 818-825, Aug 2010

[10] A. Divakaran, K. A. Peker, R. Radhakrishnan, Z. Xiong and R. Cabasson, "Video Summarization Using MPEG-7 Motion Activity and Audio Descriptors", *Video Mining*, Rosenfeld, A.; Doermann, D.; DeMenthon, D., October 2003 (Kluwer Academic Publishers)

[11] L. Liu and G. Fan; , "Combined key-frame extraction and object-based video segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.15, no.7, pp. 869- 884, July 2005.

[12] X. Song, G. Fan, , "Joint Key-Frame Extraction and Object-Based Video Segmentation", in *Proceedings of IEEE Workshop on Motion and Video Computing*, vol.2, pp.126-131, Jan. 2005.

[13] X. Song, G. Fan, "A New Video Analysis Approach for Coherent Key-frame Extraction and Object Segmentation", in *Proceedings of IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1-4, Oct. 30 2005-Nov. 2 2005

[14] P. Aigrain, H. J. Zhang, D. Petkovic, "Content-

Based Representation and Retrieval of Visual Media: A State-of-the-Art Review", *MultToolApp*, no. 3, pp. 179-202, November 1996.

[15] J. Lee, I. Shin H. W. Park, "Adaptive intra-frame assignment and bit-rate estimation for variable GOP length in H.264", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 10, pp. 1271-1279, Oct 2006.

[16] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard", *IEEE on Transactions Circuits and Systems for Video Technology* , vol. 13, no. 7, 2003, pp. 560-576.

[17] R. Hammoud and R. Mohr, "A probabilistic framework of selecting effective key frames for video browsing and indexing," in *Proceedings of International Workshop on Real-Time Image Sequence Analysis*, pp. 79–88, 2000.

[18] A. Girgensohn and J. Boreczky, "Time-constrained keyframe selection technique," *Multimedia Tools Application*, vol. 11, pp. 347–358, 2000.

[19] X. C. He and N. H. C. Yung, "Corner detector based on global and local curvature properties", *Opt. Eng.*, vol. 47, no. 5, 2008, pp. 057 008–1–057 008–12,.

[20] C. Y. Wei, M. B. Evans, M. Eliot, J. Barrick, B. Maust, and J. H. Spyridakis, "Influencing web-browsing behavior with intriguing and informative hyperlink wording", *J. Inf. Sci.*, vol. 31, no. 5, 2005, pp. 433-445.

[21] T. Zhu, R. Greiner, G. Häubl, K. Jewell, and R. Price, "Using Learned Browsing Behavior Models to Recommend Relevant Web Pages", in *Proceeding of the 2005 International Joint Conference on Artificial Intelligence*, Edinburg, Scotland, 30 July–5 August, 2005, pp. 1589-1590.

[22] B. A. Huberman, P. L. T. Pirolli, J. E. Pitkow, and R. M. Lukose, "Strong Regularities in World Wide Web Surfing", *Science*, Apr 3, vol. 280, no. 5360, pp. 95- 97, 1998

[23] D. Tao, J. Cai, H. Yi, D. Rajan, L. T. Chia, and K. N. Ngan, "Dynamic Programming-Based Reverse Frame Selection for VBR Video Delivery Under Constrained Resources," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no.11, pp. 1362-1375, 2006.

[24] E. R. Iain, "Video Codec Design", *John Wiley & Sons*, April 2002.

[25] A. Girgensohn and J. Boreczky, "Time-Constrained Keyframe Selection Technique", *Multimedia Tools Application*, vol. 11, no. 3, 347-358, August 2000.

[26] W.-S. Chau, O. C. Au, T.-W. Chan, T.-S. Chong, "Optimal key frame selection using visual content metric," in *Proceedings of International Conference on Communications, Circuits and Systems*, vol. 1, pp. 551- 555, 27-30 May 2005.

[27] J. Ouyang, J. Li, and H. Tang, "Interactive key frame selection model", *Journal Visual, Communication and Image Representation*, vol. 17, no. 6, pp. 1145-1163, December 2006.

Multivariable Generalized Predictive Control Using an Improved Particle Swarm Optimization Algorithm

Moussa Sedraoui and Samir Abdelmalek
Laboratoire (PI : MIS) Problèmes Inverses: Modélisation, Information et Systèmes,
Université 08Mai 1945, Guelma, Algérie
E-mail: msedraoui@gmail.com

Sofiane Gherbi
Laboratoire d'Automatique de Skikda (LAS),
Université 20 Aout1955, Skikda, Algérie
E-mail: sgherbi@gmail.com

Keywords: improved particle swarm optimization, multivariable generalized predictive control, feasible region

Received: November 16, 2010

In this paper, an improvement of the particle swarm optimization (PSO) algorithm is proposed. The aim of this algorithm is to iteratively resolve the cost problem of the Multivariable Generalized Predictive Control (MGPC) method under multiple constraints previously reduced. An ill-conditioned chemical process modelled by an uncertain Multi-Input & Multi-Output (MIMO) plant is controlled in order to verify the validity and the effectiveness of the proposed algorithm. The performances obtained are compared with those given by the MGPC method using the standard PSO algorithm. The simulation results shows that the proposed algorithm outperforms standard PSO algorithm in terms of performance and robustness.

Povzetek: Predstavljena je metoda optimiranja z roji za nadzor s splošnim napovedovanjem in več spremenljivkami.

1 Introduction

Multivariable generalized predictive control (Morari & Lee, 1999) is a very powerful method. It has been the subject of many researches during the last few years and it was applied successfully in industry, particularly in chemical processes. It is based on MIMO predictive model [1], [2] where the expected behaviour of the system can be predicted in the extended time horizon. The MGPC law is obtained by minimizing linear or non-linear criterion (Magni, 1999, Duwaish, 2000). This criterion is composed by the sum of the square prediction errors between the predicted and desired outputs, the weighted sum of the square change-controls (control-increments) and others [3]. The constraints inclusion (as mathematical inequalities type) distinguishes most clearly MGPC from other process control paradigms as suggested in (Richalet, 1993, Qin 1997, Rawlings, 1999). These constraints are imposed in order to ensure a better stability and performance robustness (Al Hamouz and Duwaish, 2000, Imsland, 2005). The MGPC method formulates the constraint optimization problem at every step time for solving the optimal control move vector [4]. At the next sampling time, a new process measurement is received, the process is updated, and a new constraint optimization problem is solved for the next control move vector. An efficient randomized constraint optimization algorithm is suggested to the MGPC method named by PSO algorithm (Rizvi & al, 2010, Yousuf & al, 2009, Al Duwaish, 2010). This algorithm explores the search space using a population of particles, each one with a particle or an agent, starting from a random velocity vector and a random position vector. Each particle in the swarm represents a candidate solution (treated as a point) in an n-dimensional space for the constraint optimization problem, which adjusts its own "flying" according to the other particles [5]. The PSO algorithm can resolve successively various constraint optimization problems, such as linear or non-linear, convex or non-convex problems. Unfortunately, it cannot provide satisfactory results when the MGPC method is applied to poorly modelled processes [6] operating in ill-defined environments. This is, as often, the case when the plant has different gains for the operational range designed by user's trial-and- error. In addition, the PSO algorithm's convergence cannot satisfy multiple time domain specifications if the process (to be controlled) is constrained by a high number of hard constraints (Leandro dos Santos Coelho & al, 2009). Several heuristic algorithms have been developed in recent years to improve the performance and set up the parameters of the PSO algorithm [7].This paper investigates the analysis of the above mentioned problems. Two main contributions are proposed in this paper in order to

improve the performances of the MGPC method. The first one consists to reduce (if possible) the imposed inequality constraints which are reformulated as boundary constraints. The second one is to resolve the bounds constraints optimization problem by the improved PSO algorithm.

2 Unconstrained MGPC Method

All the considered matrices are in discrete time domain.

A *CARIMA* (Controller Auto Regression Integrated Moving Average) model for an *m* inputs and *m* outputs multivariable process can be expressed by [8]:

$$A(q^{-1})\Delta y(t) := B(q^{-1})\Delta u(t-1) + C(q^{-1})\zeta(t) \quad (1)$$

Where

$y(t) \in \Re^{m\times 1} := \left[y_1(t) \quad y_2(t) \quad ... \quad y_m(t)\right]^T$

$u(t) \in \Re^{m\times 1} := \left[u_1(t) \quad u_2(t) \quad ... \quad u_m(t)\right]^T$

$A(q^{-1})$, $B(q^{-1})$ and $C(q^{-1})$ are $m \times m$ monic polynomial matrices. Set $C(q^{-1})$ equal to the unity diagonal matrix. $\zeta(t)$ is an uncorrelated random process and $\Delta(q^{-1}) = 1 - q^{-1}$, this form enables to introduce an integrator in the control law. Without lost of generality one can suppose *A* as diagonal polynomial matrix.

$y_i(t) \in \Re$, $u_i(t) \in \Re$ denotes respectively, the process output and the control input of the channel number '*i*'. q^{-1} denotes the backward shift operator. The role of $\Delta(q^{-1})$ is to ensure an integral action of controller in order to cancel the effect of the step varying output in the channel '*i*'.

As in all receding horizon predictive control strategies, the control law provides that, for each channel '*i*', the control-increment $\Delta u_i(t)$ which minimizes the following unconstraint cost problem of the MGPC method [8]:

$$J := \sum_{i=1}^{m}\left(\sum_{j=1}^{N_2^i}\left[\hat{y}_i(t+j/t) - w_i(t+j)\right]^2 + \lambda_i \sum_{j=1}^{N_u^i}\left[\Delta u_i(t+j-1)\right]^2\right) \quad (2)$$

Where

$\hat{y}_i(t+j) \in \Re$ is an optimum *j*-step-ahead prediction of the system output vector on data up to time *t*, therefore, the expected value of the output vector at time *t* if the past input vector, the output vector, and the future control sequence are known. Noting that $\hat{y}_i(t+j)$ is depending to the control-increment Δu_i from resolving two Diophantine equations (more details are available in the reference [9]).

$w_i(t) \in \Re$ is the future set-point or the reference sequence for the output $y_i(t)$.

N_2^i, N_u^i (with respect: $N_u^i \le N_2^i$) denotes respectively, the maximum output prediction horizon (assumed equal to $N_2 \in \Re^+$) and the maximum control prediction (assumed equal to $N_u \in \Re^+$) for each channel '*i*'. $\lambda_i \in \Re^+$ denotes the positive parameter weighting the control input for each channel '*i*'.

3 Classification of Constraints and Problem Formulation

In constrained control, a set of inequality constraints may be set as addition of the control objective and the variation limits of certain variables to the given ranges:

$\underline{v}_i \le v_i(t+j) \le \overline{v}_i$, with $i := 1,2,..m$ and $j := N_{s1},...,N_{s2}$.

Where

$v_i(t+j) \in \Re$ is a variable under restriction,

$\underline{v}_i \in \Re$ and $\overline{v}_i \in \Re$ are the lower and the upper boundaries,

N_{s1} and N_{s2} are the lower and the upper constraint horizons respectively.

The two main objectives of constrained predictive control are set-point tracking and prevention / reduction of constraint transgressions. These constraints can be imposed (with respect to the time index) on the control-increment vector, or/and on the control vector as follows:

- Constrained on the control-increment:

$$\underline{\Delta u}_i \le \Delta u_i(t+j) \le \overline{\Delta u}_i \quad (3)$$

Where $i = 1,2,..m$ and $j = 0,...,N_u - 1$.

- Constrained on the control:

$$\underline{u}_i \le u_i(t+j) \le \overline{u}_i \quad (4)$$

Where $i = 1,2,..m$ and $j = 0,...,N_u - 1$.

By using:

$$u_i(t+j) := u_i(t-1) + \sum_{k=0}^{j}\Delta u_i(t+k) \quad (5)$$

The control constraints (4) becomes as follow:

$$\underline{u}_i - u_i(t-1) \le \sum_{k=0}^{j}\Delta u_i(t+k) \le \overline{u}_i - u_i(t-1) \quad (6)$$

The constraints on the control vector and the rate of control changes, with respect to the batch index, can be easily combined together:

$$A_{inq} \cdot \Delta U \le B_{inq} \quad (7)$$

Where

$\Delta U_{(m\cdot N_u)\times 1} := \begin{pmatrix} \Delta U(t) \\ \Delta U(t+1) \\ \vdots \\ \Delta U(t+N_u-1) \end{pmatrix}$ denotes the design parameter vector which will be determined later by the PSO algorithm, it contains the future control- increment vector $(\Delta U(t+j))_{m\times 1}$ of each channel as:

$$\Delta U(t+j) := \begin{pmatrix} \Delta u_1(t+j) \\ \Delta u_2(t+j) \\ \vdots \\ \Delta u_m(t+j) \end{pmatrix}_{j=0,1,\cdots,N_u-1}$$

$\left(A_{inq}\right)_{(4.m.N_u)\times(m.N_u)}$, $\left(B_{inq}\right)_{(4.m.N_u)\times 1}$ are defined by:

$$A_{inq} := \begin{pmatrix} diag(I_{m\times m}) \\ -diag(I_{m\times m}) \\ tril(I_{m\times m}) \\ -tril(I_{m\times m}) \end{pmatrix}$$

Where

$diag(I_{m\times m}) \in \Re^{(m.N_u)\times(m.N_u)}$ denotes the unity diagonal matrix, and $tril(I_{m\times m}) \in \Re^{(m.N_u)\times(m.N_u)}$ denotes the lower triangular matrix of the unity diagonal matrix $(I_{m\times m})$.

$$B_{inq} := \begin{pmatrix} \left[\Delta u_i\right]_{(m\cdot N_u)\times 1} \\ -\left[\Delta u_i\right]_{(m\cdot N_u)\times 1} \\ \left[\overline{u}_i - u_i(t-1)\right]_{(m\cdot N_u)\times 1} \\ -\left[\underline{u}_i - u_i(t-1)\right]_{(m\cdot N_u)\times 1} \end{pmatrix}$$

The cost index (2) can be expressed in matrix form as:

$$J(\Delta U,t) := \Delta U^T \cdot Q_2 \cdot \Delta U + Q_1^T \cdot \Delta U + Q_0 \quad (8)$$

Where $Q_2 := G^T \cdot G + \Lambda$, $Q_1^T := 2(\Gamma - W)^T G$ and $Q_0 := (\Gamma - W)^T \cdot (\Gamma - W)$

$\Lambda := \lambda \cdot I_{(m\cdot N_u)\times(m\cdot N_u)}$ is diagonal matrix weighting the control-increment vector, and $W_{(m\cdot N_2)\times 1}$ is the projected set-point vector.

$G_{(m\cdot N_2)\times(m\cdot N_u)}$, $\Gamma_{(m\cdot N_2)\times 1}$ are the polynomial matrices which are determined by the recursively resolution of the two Diophantine equations [9].

The cost index (8) and the inequality constraints (7) formulate the following constraint optimization problem as:

$$\begin{cases} \min\limits_{\Delta U} J(\Delta U,t) := \Delta U^T \cdot Q_2 \cdot \Delta U + Q_1^T \cdot \Delta U + Q_0 \\ \text{s.t}: \quad A_{inq} \cdot \Delta U \le B_{inq} \end{cases} \quad (9)$$

Now, an optimal control vector is given by the PSO algorithm. This algorithm should minimize the objective function (8) under $4\times m\times N_u$ inequality constraints. The computational requirements of the PSO algorithm depend heavily on the number and the type of the constraints to be satisfied. An efficient off-line constraint PSO algorithm, suggested by Ichirio & al, 2009, can resolve this problem [10]. Unfortunately, this algorithm is difficult to extend to the MGPC method for two reasons: the first one is due to a large dimension of the inequality constraints which needs excessive computation time. The second one is due to a real-time output feedback implementation of the MGPC method which requires a minimum consuming time. To resolve these above problems, the inequality constraints should be reduced, for each step time, and reformulated as bounds constraints type. Only those constraints (which limit the feasible region) must be taken into account. The efficiency of the PSO algorithm can be increased if the superfluous constraints (which do not limit the feasible region) should be eliminated [11]. In this paper we propose a systematic method that determines the minimum set vector of limiting constraints. The lower and the upper bounds of the feasible region are given as below:

For each channel $i := 1,\cdots,m$, the control-increment $\Delta u_i(t)$ is simultaneously constrained by:

1- For the control prediction horizon $j = 0$:

$$\begin{cases} \underline{\Delta u}_i \le \Delta u_i(t) \le \overline{\Delta u}_i \\ \underline{u}_i - u(t-1) \le \Delta u_i(t) \le \overline{u}_i - u(t-1) \end{cases} \quad (10)$$

It is easy to see that the new lower and upper bounds are determined by:

$$\underline{v}_i(t) \le \Delta u_i(t) \le \overline{v}_i(t) \quad (11)$$

Where

$$\underline{v}_i(t) := \max\{\underline{\Delta u}_i \quad \underline{u}_i - u(t-1)\} \quad (12)$$

$$\overline{v}_i(t) := \min\{\overline{\Delta u}_i \quad \overline{u}_i - u(t-1)\} \quad (13)$$

2- For the control prediction horizon $j = 1$:

$$\begin{cases} \underline{\Delta u}_i \le \Delta u_i(t+1) \le \overline{\Delta u}_i \\ \underline{u}_i - u(t-1) \le \Delta u_i(t+1) + \Delta u_i(t) \le \overline{u}_i - u(t-1) \end{cases} \quad (14)$$

The new lower and upper bounds are determined for $j = 1$ by:

$$\underline{v}_i(t+1) \le \Delta u_i(t+1) \le \overline{v}_i(t+1) \quad (15)$$

Where

$$\underline{v}_i(t+1) := \max\{\underline{\Delta u}_i \quad \{\underline{u}_i - u(t-1)\} - \underline{v}_i(t)\} \quad (16)$$

$$\overline{v}_i(t+1) := \min\{\overline{\Delta u}_i \quad \{\overline{u}_i - u(t-1)\} - \overline{v}_i(t)\} \quad (17)$$

This procedure is repeated until the control prediction horizon $j = N_u - 1$. Therefore the control-increment $\Delta u_i(t+N_u-1)$ is constrained by the new bounds:

$$\underline{v}_i(t+N_u-1) := \max\{\underline{\Delta u}_i \quad \{\underline{u}_i - u(t-1)\} - \sum_{k=0}^{N_u-2} \underline{v}_i(t+k)\} \quad (18)$$

$$\overline{v}_i(t+N_u-1) := \min\{\overline{\Delta u}_i \quad \{\overline{u}_i - u(t-1)\} - \sum_{k=0}^{N_u-2} \overline{v}_i(t+k)\} \quad (19)$$

Then, for each step time value $t_0, t_1, \cdots, t_{\max}$, the feasible region $D(i,j,t) := (\underline{v}_i(t+j) \quad \overline{v}_i(t+j))_{\substack{j=0,\cdots,N_u-1 \\ i=1,\cdots,m}}$ can be determined by the following proposed algorithm:

3.1 Reduced constraints algorithm

For each point time $t_0, t_1, \cdots, t_{\max}$, the feasible region is determined by the followings steps:

[Step 1]: *Set the first counter* $i \leftarrow 1$ *which denotes the number of channels, and go to the next step.*

[Step 2]: *Set the second counter* $j \leftarrow 0$ *which denotes the control horizon prediction, and go to the next step.*

[Step 3]: *Set the parameters* $h_{\max} \leftarrow 0$, $h_{\min} \leftarrow 0$ *and go to the next step.*

[Step 4]: *Build the followings ranges:*

$bound\_\max_i := \{\overline{\Delta u}_i \quad \{\overline{u}_i - u_i(t-1)\} - h_{\max}\}$.

$bound\_\min_i := \{\underline{\Delta u}_i \quad \{\underline{u}_i - u_i(t-1)\} - h_{\min}\}$.

[Step 5]: *Calculate the new upper and the new lower bounds which limit the control- increment* $\Delta u_i(t+j)$ *by:*

$\overline{v_i}(t+j) := \min\{bound\_\max_i\}$

$\underline{v_i}(t+j) := \max\{bound\_\min_i\}$

From these above bounds, the feasible region is determined as follow:

$D(i,j,t) := (\underline{v_i}(t+j) \quad \overline{v_i}(t+j))$

[Step 6]: *Update the parameters:* $h_{\min}$, $h_{\max}$ *as follows*

$h_{\max} \leftarrow h_{\max} + \overline{v_i}(t+j)$,

$h_{\min} \leftarrow h_{\min} + \underline{v_i}(t+j)$, and g*o to the next step*

[Step 7]: *Update the second counter* $j \leftarrow j+1$ *and go back to the step 4 if* $j \prec N_u - 1$. *Otherwise, go to the next step.*

[Step 8]: *Update the first counter* $i \leftarrow i+1$ *and stop algorithm if* $i := m$ **.** *Otherwise go back to the step 2.*

From this above algorithm, the constraint optimization problem (9) under inequality constraints is reformulated as the bounds optimization problem:

$$\begin{cases} \min\limits_{\Delta U} J(\Delta U, t) := \Delta U^T \cdot Q_2 \cdot \Delta U + Q_1^T \cdot \Delta U + Q_0 \\ \text{s.t}: \quad \underline{\Delta U} \le \Delta U \le \overline{\Delta U} \end{cases} \tag{20}$$

Where $\underline{\Delta U}$, $\overline{\Delta U}$ denotes respectively, the new lower and the new upper bounds vector which limit the feasible region $D_{(m.N_u)\times 2} := (\underline{\Delta U}, \overline{\Delta U})$, with:

$$\left(\underline{\Delta U}\right)_{(m.N_u)\times 1} := \left(\underline{v}_i(t) \quad \underline{v}_i(t+1) \quad \cdots \quad \underline{v}_i(t+N_u-1)\right)^T_{i=1,2\cdots m}$$

$$\left(\overline{\Delta U}\right)_{(m.N_u)\times 1} := \left(\overline{v_i}(t) \quad \overline{v_i}(t+1) \quad \cdots \quad \overline{v_i}(t+N_u-1)\right)^T_{i=1,2\cdots m}$$

From (20), it is easy to see that the inequality constraints number is reduced to $m \times N_u$ constraints at each step time. This dramatic reduction has a capital importance for the success of the PSO algorithm.

Now, we are able to find the optimal control of the MGPC law. The new constraint optimization problem (20) should be resolved for each step time $t := t_0, t_1, \cdots, t_{\max}$, its solution vector ΔU^* denotes the optimal design parameter vector. Only the first m rows of ΔU^* is used to obtain the optimal desired control-increment vector of each channel (' i '). The optimal control vector is obtained by adding the previous control vector to the optimal control-increment vector as follow:

$$u_i(t) := u_i(t-1) + \Delta u_i^*(t) \tag{21}$$

4 Improved PSO Algorithm [6]

Particle swarm optimization algorithm, introduced first by Kennedy and Eberhart in (1995), is one of the modern heuristic algorithms which belong to the category of *Swarm Intelligence* method (Kennedy, 2001). The PSO algorithm uses a swarm consisting of $N_p \in \aleph$ particles for each control-increment vector $\left(\Delta u_i(t+j)\right)_{\substack{j=0,1,\cdots N_u-1 \\ i=1,2,\cdots m}}$ to get an optimal solution $\Delta u_i^*(t+j)$ which minimizes the optimization problem (20). The position of (i_{th}) particle and its velocity are respectively denoted as [12]:

$$\Delta u_i(t+j) := \left(\Delta u_{i,1}(t+j) \quad \Delta u_{i,2}(t+j) \quad \ldots \quad \Delta u_{i,N_p}(t+j)\right)^T$$

$$\psi_i(t+j) := \left(\psi_{i,1}(t+j) \quad \psi_{i,2}(t+j) \quad \ldots \quad \psi_{i,N_p}(t+j)\right)^T$$

.

Then, the position of the (i_{th}) particle, $\Delta u_i(t+j)$, is based on the following update law:

for $\ell = 1,2,\ldots \ell_{\max}$, which indicates the iteration number [12],

$$\psi_i^{\ell+1} := c_0\psi_i^{\ell} + c_1 r_{1,i}^{\ell}\left(H_i^{best,\ell} - \Delta u_i^{\ell}\right) + c_2 r_{2,i}^{\ell}\left(h_{swarm}^{best,\ell} - \Delta u_i^{\ell}\right) \tag{22}$$

$$\Delta u_i^{\ell+1} := \Delta u_i^{\ell} + \psi_i^{\ell+1} \tag{23}$$

Where c_1 and c_2 are respectively, the cognitive (individual) and the social (group) learning rates and are both positive constants. The value of cognitive parameter c_1 signifies a particle's attraction to a local best position based on its past experience. The value of social parameter c_2 determines the swarm's attraction towards a global best position.

$c_0 \in \Re^+$ is the inertia weight factor whose value decreases linearly with the iteration number (Shi & Eberhart, 1999) as [13]:

$$c_0 := \theta_{\max} - \left(\frac{\theta_{\max} - \theta_{\min}}{\ell_{\max}}\right) \cdot \ell \tag{24}$$

Where $\theta_{\max}$ and $\theta_{\min}$ are the initial and the final values of the inertia weight, respectively. The values of $\theta_{\max} = 0.9$ and $\theta_{\min} = 0.4$ are commonly used [13].

The random numbers $r_{1,i}^{\ell}$ and $r_{2,i}^{\ell}$ are uniformly distributed in $[0,1]$.

$H_i^{best,\ell}$ and $h_{swarm}^{best,\ell}$ denotes respectively, the best previously obtained position of the (i_{th}) particle (the position giving the lower value of the objective criterion) and the best position in the entire swarm at the current iteration ℓ [10]:

$$H_i^{best,\ell} := \arg\min_{\Delta u_i^r}\{J(\Delta u_i^r), 0 \le r \le \ell\} \tag{25}$$

$$h_{swarm}^{best,\ell} := \arg\min_{\Delta u_i^{\ell}}\{J(\Delta u_i^{\ell}), \forall i\} \tag{26}$$

From equation (23), some current position of (i_{th}) particle (in each dimension) can exceed the corresponding lower bound or upper bound of the feasible region. Consequently, the given optimal control vector of the MGPC method cannot satisfy some specifications and also some constraints are non-satisfactoriness' in some range time. To avoid, we should improve the convergence of PSO algorithm by adjusting only the corrupted position of (i_{th}) particle with the region around the current established solution, if it is too smaller than the corresponding lower bound, its value $\underline{v}_i$ should be replaced. If it is too higher than the

corresponding upper bound, then its value is replaced by $\overline{v_i}$. The proposed modification can be formulated as follows:

Let consider:

$\Delta u_i^q(t+j)$: The corrupted position of (i_{th}) particle given at current iteration $\ell := q$.

$\underline{v}_i(t+j)$, $\overline{v_i}(t+j)$: The lower bound and upper bound which are determined by the reduced constraints algorithm. So that, the above corrupted position can be adjusted by using the following inequalities:

$$\Delta u_i^q(t+j) := \begin{cases} \underline{v}_i(t+j) : if \quad \Delta u_i^q(t+j) \prec \underline{v}_i(t+j) \\ \overline{v_i}(t+j) : if \quad \Delta u_i^q(t+j) \succ \overline{v_i}(t+j) \end{cases} \tag{27}$$

Consequently, from the equation (23), the current velocity should be limited by the following bounds:

$$\underline{v}_i - \Delta u_i^{q-1} \prec \psi_i^q \prec \overline{v_i} - \Delta u_i^{q-1} \tag{28}$$

Now, the modified current positions with their modified velocity are used to improve the next best position and their velocity vector for the next iteration as follow:

$$\psi_i^{q+1} := c_0 \psi_i^q + c_1 r_{1,i}^q \left(H_i^{best,q} - \Delta u_i^q \right) + c_2 r_{2,i}^q \left(h_{swarm}^{best,q} - \Delta u_i^q \right) \tag{29}$$

$$\Delta u_i^{q+1} := \Delta u_i^q + \psi_i^{q+1} \tag{30}$$

The improved PSO algorithm consists of the following steps:

4.1 Proposed algorithm

For each step time $t := t_0, t_1, \cdots, t_{\max}$ the optimal control-increment is determined by the following steps:

[Step 1]: *Determine the lower bound and the upper bound* $\underline{v}_i(t+j)$, $\overline{v_i}(t+j)$ *which are corresponding the design parameter* $\left[\Delta u_i(t+j)\right]_{\substack{i:=1,\cdots,m \\ j:=0,\cdots,N_u-1}}$.

[Step 2]: *Initialize random swarm positions and velocities:*

initialize a population (array) of particles with random positions and velocities (array) from the search domain $D := (\underline{\Delta U}, \overline{\Delta U})$.

Set the counter $\ell \leftarrow 1$ *and go to the next step.*

[Step 3]: *Evaluate the objective criterion (20) and obtain* $H_i^{best,\ell}$, $h_{swarm}^{best,\ell}$ *according to (25) and (26).*

[Step 4]: *Update of a particle's velocity and its position according to (22) and (23).*

[Step 5]: *Check each parameter of the particle's position by the following corresponding lower bound and upper bound* $\underline{v}_i(t+j)$, $\overline{v_i}(t+j)$. *Replace only those exceeding these above bounds.*

[Step 6]: *Update the counter* $\ell \leftarrow \ell + 1$ *and go back to the step 3 if* $\ell \prec \ell_{\max}$. *Otherwise, stop algorithm and take the best position vector as an optimal solution which minimize the constrained optimization problem (20).*

5 Simulation Results and Discussion

In this section, a multivariable generalized predictive control method using a modified particle swarm optimization algorithm is applied to a distillation column which is MIMO plant with two input and output vectors (benchmark problem, see [14]). The two inputs are the reflux and the vapour boil up rate and the outputs are the distillate and the bottom product. The results are compared with those given by the MGPC method using the standard PSO algorithm. The mathematical model is given by [14]:

$$G(s) := \frac{1}{75s+1} \begin{bmatrix} 0.878 & -0.864 \\ 1.082 & -1.096 \end{bmatrix} \begin{bmatrix} K_1 e^{-\tau_1 s} & 0 \\ 0 & K_2 e^{-\tau_2 s} \end{bmatrix} \tag{31}$$

$K_{i(i=1,2)} \in [0.8 \quad 1.2]$, $\tau_{i(i=1,2)} \in [0.0 \quad 1.0]$.

Where

τ_i, K_i denotes respectively, the uncertainty temperatures and uncertainty gains of the process.

The time domain specifications are formulated, for the time range $t \in [0,400]$ minutes, as below:

a- For the first set-point reference vector: $w = (1 \quad 0)^T$, the first and the second output channels y_1 and y_2 must satisfy[14]:

(S_1): $y_1(t) \geq 0.9$ in more than 30 minutes.

(S_2): $y_1(t) \leq 1.1$: the maximum over-shoot corresponding the first output channel cannot exceed 11% for all range time $t \in [0,400]$.

(S_3): $0.99 \leq y_1(\infty) \leq 1.01$: the static error value cannot exceed 1% ($|y_1(\infty) - w_1(\infty)| \leq 1\%$).

(S_4): $y_2(t) \leq 0.5$: the maximum over-shoot of the second output channel cannot exceed to 50% for all range time $t \in [0,400]$.

(S_5): For $t \to \infty$: $-0.01 \leq y_2(t) \leq 0.01$: the static error value cannot exceed 1%. From another word:

$|y_2(\infty) - w_2(\infty)| \leq 1\%$.

(S_6): Closed loop stability.

(S_7): Control signals should be limited by $[-200 \quad +200]$.

(S_8): Control-increment signals should be limited by $[-12 \quad +12]$.

For the set-point reference vector: $w = (1 \quad 0)^T$, the sampling time $T_e = 1$ minute is used to determine a *CARIMA* predictive model of the chemical process for two followings parameters cases [14]:

$K_{1,2} = \tau_{1,2} = 1$ and $K_1 = 1.2, K_2 = 0.8, \tau_{1,2} = 1$.

b-The same previous time domain specifications should be satisfied for the second set-point reference vector $w = (0 \quad 1)^T$ corresponding to the low gains direction $K_1 = K_2 = 0.8$ and the same time delay constants $\tau_1 = \tau_2 = 1$.

The MGPC method is tuned by choosing: $(N_2^i, N_u^i, \lambda_i)_{i=1,2} = (8,6,0.01)$ at time range $t := [0,400]$ minutes.

For each step time: $t := t_0, t_1, \cdots, 400$, the feasible region is determined from the following constraints:

$$-200 \le u_i(t+j)_{\substack{j=0,\cdots,5 \\ i=1,2}} \le +200 .$$

$$-12 \le \Delta u_i(t+j)_{\substack{j=0,\cdots,5 \\ i=1,2}} \le +12 .$$

From the reduced constraints algorithm (see section 3.1), these above inequality constraints are reduced in order to determine the search space D at each step time. The constrained optimization problem is resolved by standard and improved PSO algorithms according to the following parameters:

- Swarm size: $N_p := 24$.
- Maximum iteration: $\ell_{\max} := 100$.
- Cognitive and social learning rates: $c_1 = c_2 := 1$.

For the set-point reference vector: $w = (1 \quad 0)^T$ and the parameter system's: $K_{1,2} = \tau_{1,2} = 1$, the figures 1.1 to 1.3 shows the results given by the MGPC method using the standard PSO algorithm (dashed curves), and the MGPC method using the improved PSO algorithm (line curves). The table1 summaries the results obtained by the two algorithms.

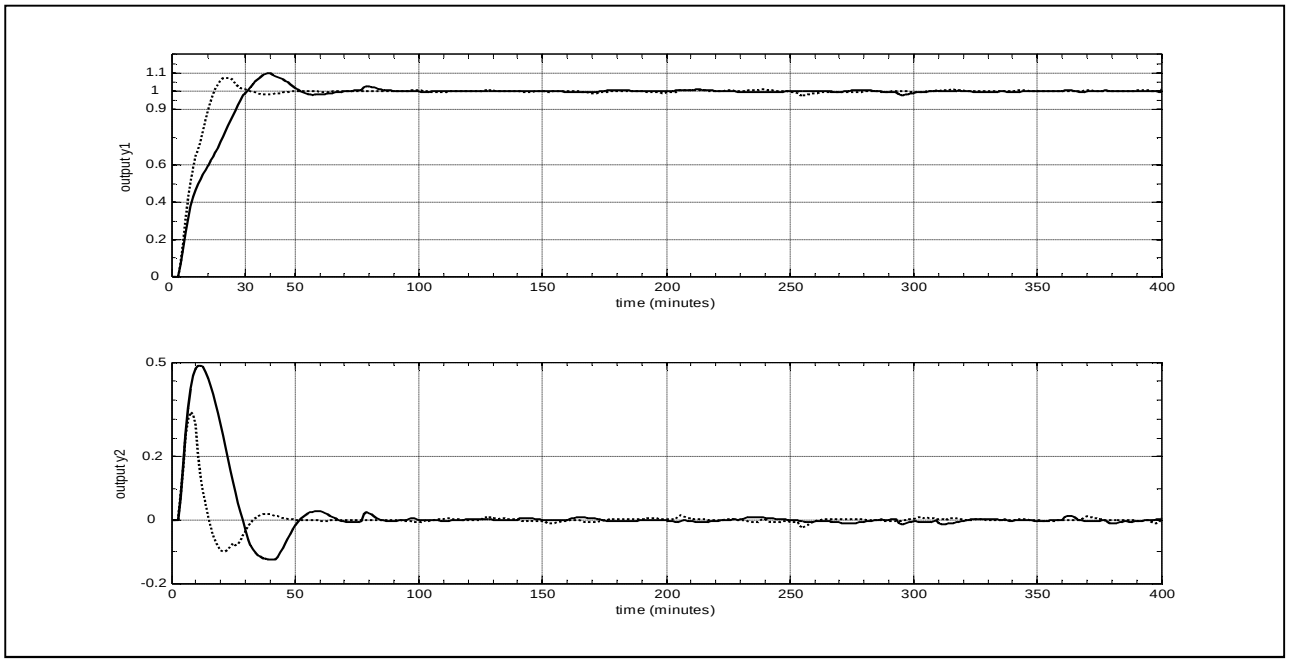


Figure 1.1: Set-point tracking results with standard and improved PSO algorithms for $w = (1 \quad 0)^T$ and $K_{1,2} = \tau_{1,2} = 1$.

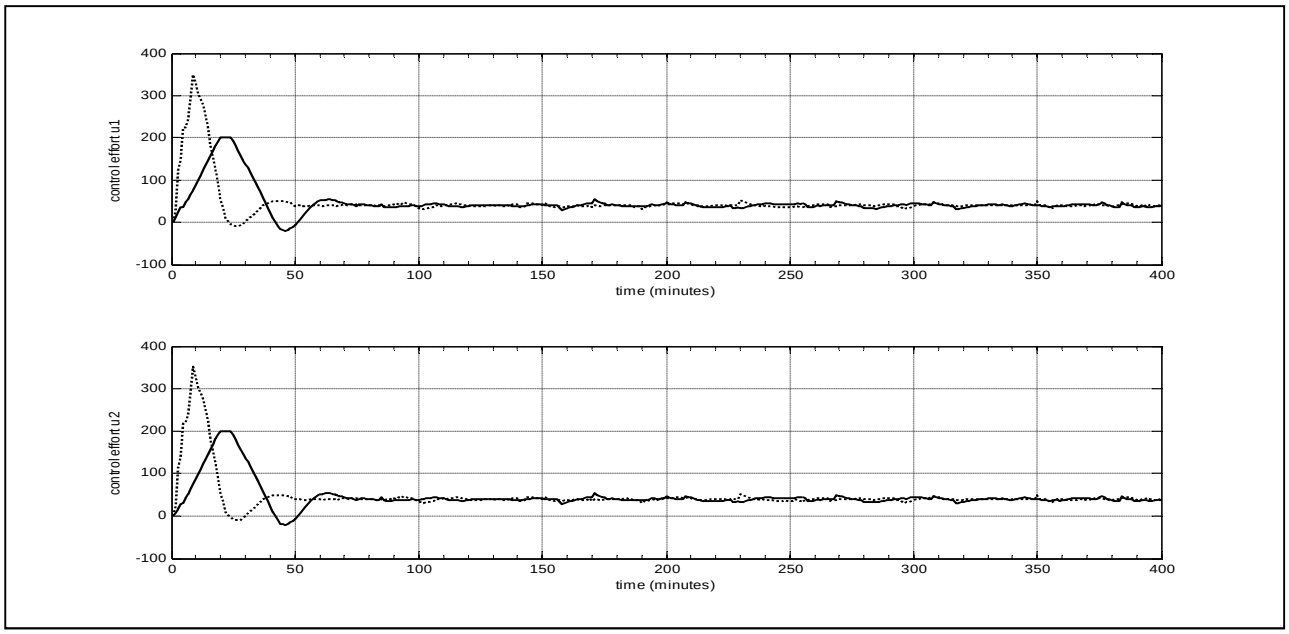


Figure 1.2: Control effort results with standard and improved PSO algorithms for $w = (1 \quad 0)^T$ and $K_{1,2} = \tau_{1,2} = 1$.

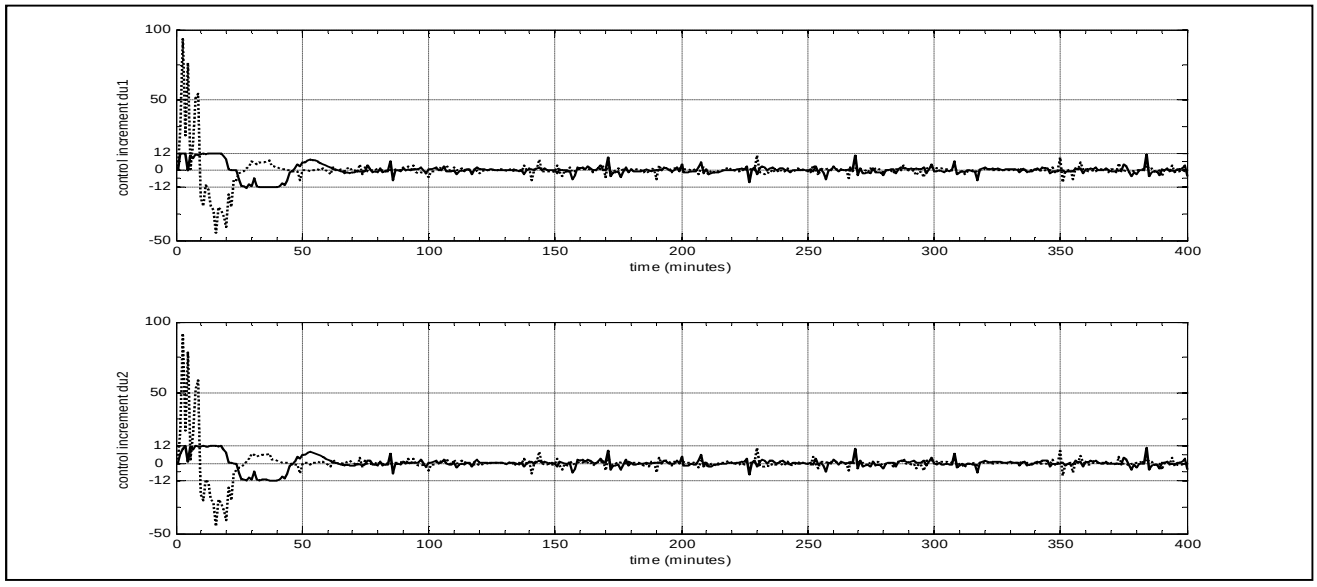


Figure 1.3: Control-increment results with standard and improved PSO algorithms for $w=\begin{pmatrix}1 & 0\end{pmatrix}^T$ and $K_{1,2}=\tau_{1,2}=1$.

Specifications (S_k):	***(S1) $y_1(30)$***	***(S2)***		***(S3) $y_1(400)$***	***(S4)***		***(S5) $y_2(400)$***	***(S6) stable / unstable***	***(S7) Unsatisfactory/ satisfactory constraints***	***(S8) Unsatisfactory/ satisfactory constraints***	***Decision & reasons***
Algorithms:		max(y_1)	time		max(y_2)	time					
Standard PSO	*1.010*	*1.074*	*23*	*1.010*	*0.343*	*8*	*0.006864*	*stable*	***unsatisfactory*** ***-9.1≤u_1≤349.3*** ***-9.9≤u_2≤351.4***	***unsatisfactory*** ***-44≤Δu_1≤93*** ***-44≤Δu_2≤91***	***Rejected*** *algorithm (S7),(S8)*
Improved PSO	*0.990*	*1.096*	*40*	*0.9975*	*0.4846*	*13*	*0.002512*	*stable*	*Satisfactory* *-20.1≤u_1≤200* *-20.3≤u_2≤199*	*Satisfactory* *-12≤Δu_1≤12* *-12≤Δu_2≤12*	*Accepted algorithm*

Table 1: Summary of the results (unsatisfactory performances are in bold) for the nominal model and the set-point reference $w=\begin{pmatrix}1 & 0\end{pmatrix}^T$.

According to the figure 1.1, we can see that, the tracking dynamic of set-point reference vector found by MGPC method based on a standard PSO algorithm is better than the other algorithm but unfortunately, the time domain specifications: (S_7) and (S_8) are not satisfied.

In the figure 1.2, the obtained control signals of the MGPC method based on standard PSO algorithm exceed the constraint ranges at $t:=\{5,6,\cdots 15\}$ minutes such as: $u_{1\max}(9)=349.3$ and $u_{2\max}(9)=351.4$. In addition, the control-increment signals presented in the figure 1.3 also violate the constraint ranges at times:

$t:=\{(2\cdots 5),(7\cdots 11),(13\cdots 22)\}$ minutes.

Consequently, the performance robustness of this method is very poor in comparison with the MGPC method using the improved PSO algorithm which is capable to satisfy all time domain specifications. These results confirm the usefulness and the robustness of the proposed algorithm.

Figures 2.1, 2.2, 2.3 and table 2 give the results of the MGPC method with the following parametric changes in the process: $(K_1=1.2, K_2=0.8, \tau_{1,2}=1)$ for the set-point reference vector $w=\begin{pmatrix}1 & 0\end{pmatrix}^T$.

According to the figures 2.1 to 2.3, the better results are obtained by the improved PSO algorithm which satisfies all time specifications (S_1 to S_8). These results can be explained by the best stability robustness against the process parametric disturbances. Furthermore, the control and the control-increment signals from the standard PSO algorithm show a dramatic oscillation at transient time region and exceed the constraint ranges. In fact, this algorithm cannot fulfill the three followings time domain specifications: (S_2) with $\max(y_1)=11.124\%$, (S_7) with $u_{1\max}=251$, $u_{2\max}=374$ and(S_8)with $-26.4\le\Delta u_1=70.76$, $-39.7\le\Delta u_2=99.05$, which can be explained by a high sensitivity to the parametric process variations. Thus, from these figures and table 2, we confirm the superiority of the proposed algorithm.

Figures 3.1, 3.2, 3.3 and table 3 give the results of the MGPC method using both algorithms when low gains directions of the process and set-point reference vector change simultaneously as follows: $(K_1=0.8, K_2=0.8, \tau_{1,2}=1), w=\begin{pmatrix}0 & 1\end{pmatrix}^T$.

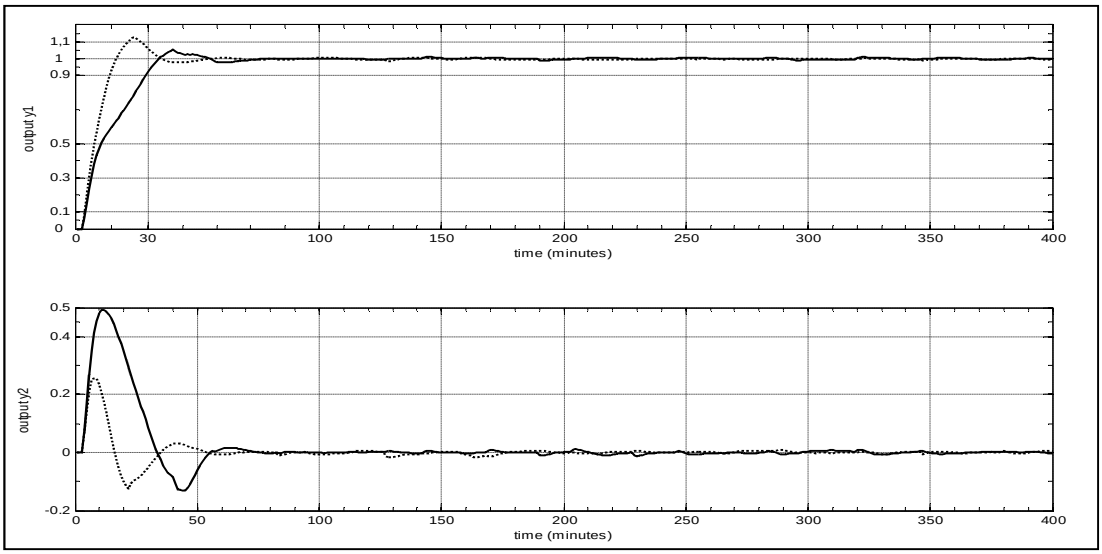


Figure 2.1: Set-point tracking results with standard and improved PSO algorithms for $w = \begin{pmatrix} 1 & 0 \end{pmatrix}^T$ and $\left(K_1 = 1.2, K_2 = 0.8, \tau_{1,2} = 1\right)$.

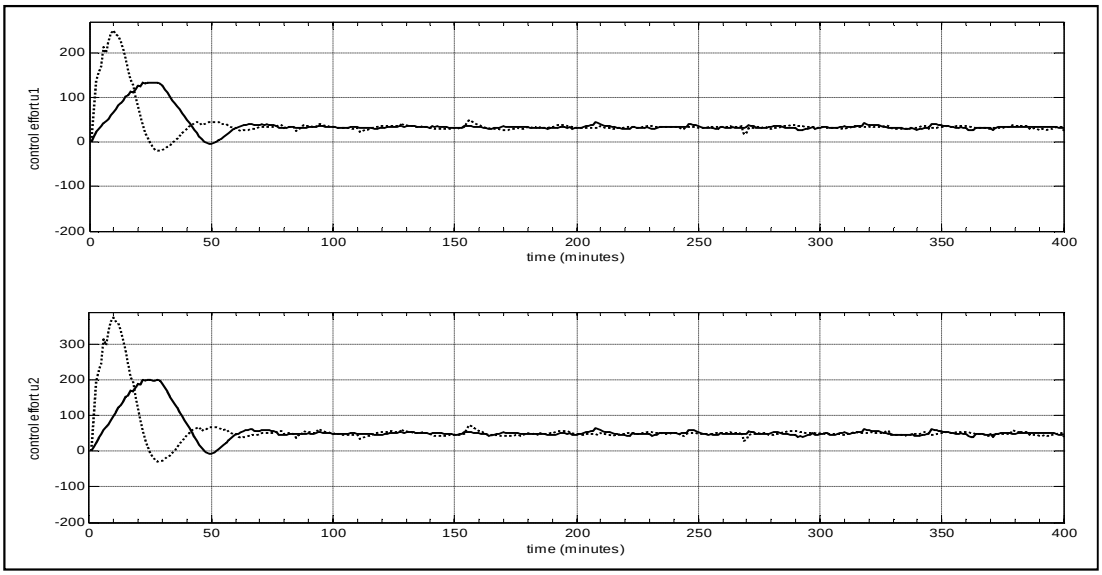


Figure 2.2: Control effort results with standard and improved PSO algorithms for $w = \begin{pmatrix} 1 & 0 \end{pmatrix}^T$ and $\left(K_1 = 1.2, K_2 = 0.8, \tau_{1,2} = 1\right)$.

These above figures clearly show the performance superiority of the proposed PSO algorithm over standard PSO. For this case, the time domain specifications: S_2, S_7 and S_8 are satisfied with the proposed PSO algorithm, while the same specifications are not satisfactoriness with standard PSO. In addition, the obtained outputs by the standard PSO algorithm converge to the set-point references but unfortunately, two other specifications cannot be satisfied at time $t = 208$ minutes which are:

(S_3): $\left|y_2(208) - w_2(208)\right| = 5\%$.

(S_5): $\left|y_1(208) - w_1(208)\right| = 3\%$.

6 Conclusion

In this study, we proposed an improvement of the PSO algorithm, it has been introduced and applied to solve the constrained MGPC problem. In order to find a feasible region, the constraints on the controls and their increments have been previously reduced at each step time, the obtained convergences by improved PSO algorithm are well improved in comparison with the standard PSO algorithm. The efficient of the proposed algorithm is clearly shown and the performances robustness and the stability robustness are guaranteed with little still sensitivity to a set-point references changes and parametric model uncertainties. The results of the proposed algorithm justifies its efficiency and presents quite promising results and can be a subject of an interesting investigations.

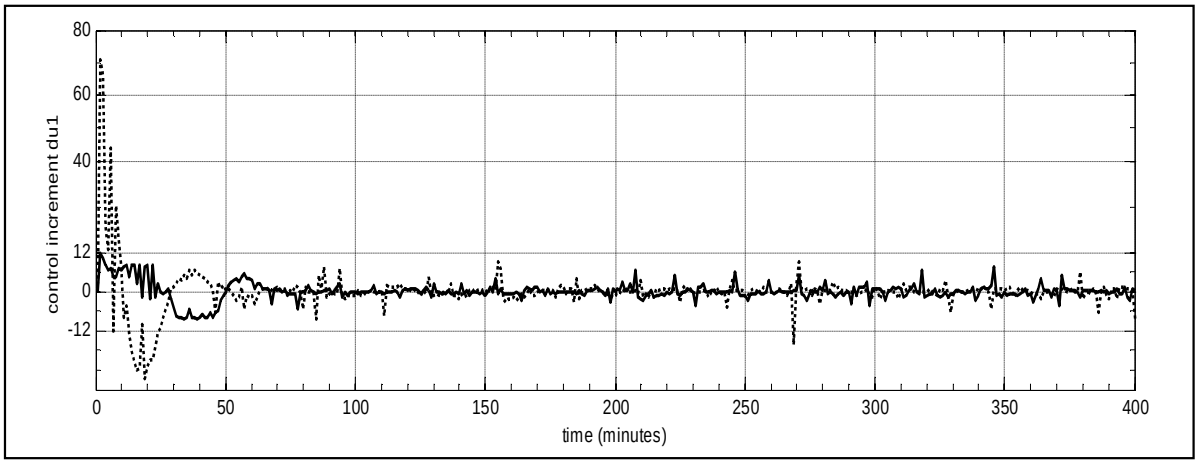


Figure 2.3: Control-increment results with standard and improved PSO algorithms for $w=\begin{pmatrix}1 & 0\end{pmatrix}^T$ and $\left(K_1=1.2, K_2=0.8, \tau_{1,2}=1\right)$.

Specifications (S_k):	***(S1) $y_1(30)$***	***(S2)***		***(S3) $y_1(400)$***	***(S4)***		***(S5) $y_2(400)$***	***(S6) stable / unstable***	***(S7) Satisfactory/ unsatisfactory constraints***	***(S8) Satisfactory/ unsatisfactory constraints***	***Decision for reasons (S_k)***
		max(y_1)	time		max(y_2)	time					
Algorithms:											
Standard PSO	*1.061*	***1.1124***	*24*	*0.9953*	*0.2562*	*8*	*0.006142*	stable	***unsatisfactory*** ***-19.2≤u_1≤251*** ***-29.4≤u_2≤374***	***unsatisfactory*** ***-26.4≤Δu_1≤70.76*** ***-39.7≤Δu_2≤99.05***	***Rejected algorithm*** *(S_2),(S_7),(S_8)*
Improved PSO	*0.920*	*1.054*	*40*	*0.9953*	*0.493*	*12*	*0.006142*	*stable*	*Satisfactory* *-3.5≤u_1≤133.5* *-6.4≤u_2≤199*	*Satisfactory* *-8.38≤Δu_1≤12* *-12≤Δu_2≤12*	*Accepted algorithm*

Table 2: Summary of the results (unsatisfactory performances are in bold) for the uncertainty model $\left(K_1=1.2, K_2=0.8, \tau_{1,2}=1\right)$ and the set-point reference $w=\begin{pmatrix}1 & 0\end{pmatrix}^T$.

References

[1] D.W Clarke, C. Mohtadi & P.S Tuffs, Generalized Predictive Control –Part I : The basic algorithm – Part II: extensions and interpretation. *Automatica*, 23(2), 1987, 137-160.

[2] D.W Clarke & C. Mohtadi, Properties of generalized predictive control, *Automatica*, 25(6), 1989, 859-875.

[3] P. Codron & P. Boucher, Improvement for multivariable generalized predictive control with multiple reference models. *Proc. 12th IASTED International Conf. Modeling. Identification. And Control, Innsbrûck,* Austria, Feb.1993.

[4] T.S. Chang & D.E. Seborg, A linear approach programming for multivariable feedback control with inequality constraints, *International journal of control* 37,1983, 583-597.

[5] M. Zribi, M. Al-Rashed & M. Alrifai. Adaptive decentralized load frequency control of multi-area power systems. *Electrical Power and Energy Systems,* 27:575583, 2005.

[6] J.P. Coelho, P.B.M. Oliveira & J.B. Cunha. Greenhouse air temperature control using the particle swarm optimization algorithm. *15th Triennial world congress.* Barcelona, Spain. IFAC, 2002.

[7] L.D.S. Coelho & al. Model-free adaptive control optimization using a chaotic particle swarm approach. *Chaos, solutions and fractals.* Elsevier. 41, 2009, pp2001-2009.

[8] P. Codron & P. Boucher. Multivariable generalized predictive control with multiple reference models: A new choice for the reference model, *ECC'93. European Control Conf.*, June 28- July1993.

[9] P. Zelinka, B.R. Ilkiv & A.G. Kuznetsov. Experimental verification of stabilizing predictive control. *Control engineering practice7.* 1999, pp 601-610.

[10] I. Maruta, T. Hyoung & T. Sugie: Fixed-Structure H_∞ Controller: A Meta-Heuristic Approach Using Simple Constrained Particle Swarm Optimization. pp:553-559.Automatica 45(2009).

[11] E.F. Camacho. Constrained Generalized Predictive Control, *IEEE Transaction on Automatic Control*, AC-38 (2), 1993, 327-332.

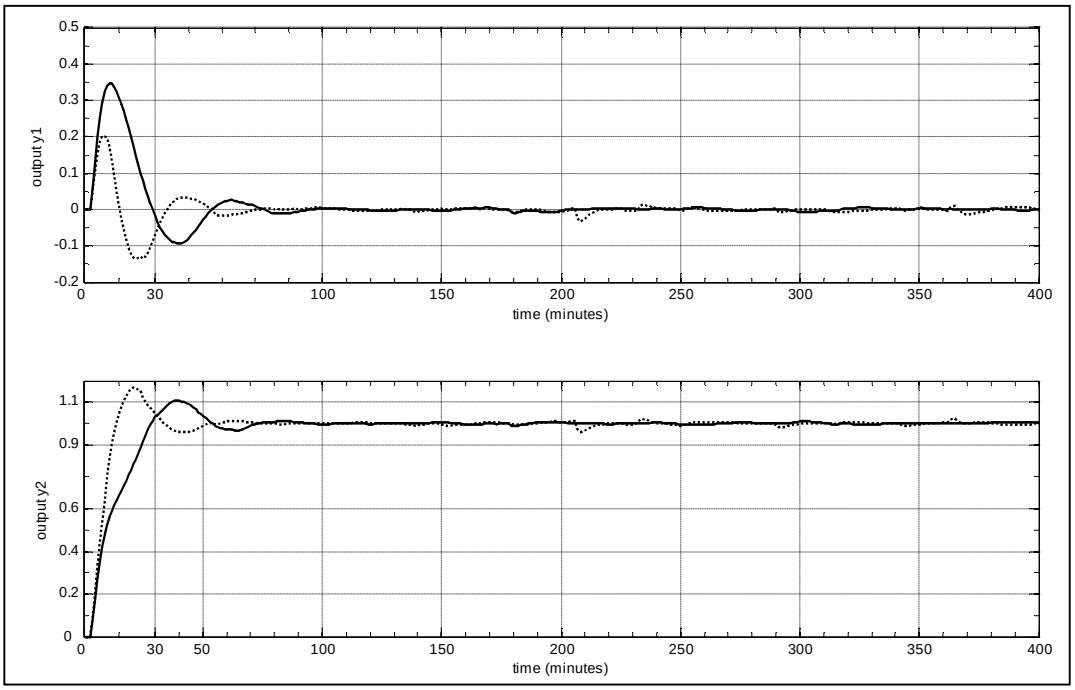


Figure 3.1: Set-point tracking results with standard and improved PSO algorithms for $w = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$ and $\left(K_1 = 0.8, K_2 = 0.8, \tau_{1,2} = 1\right)$.

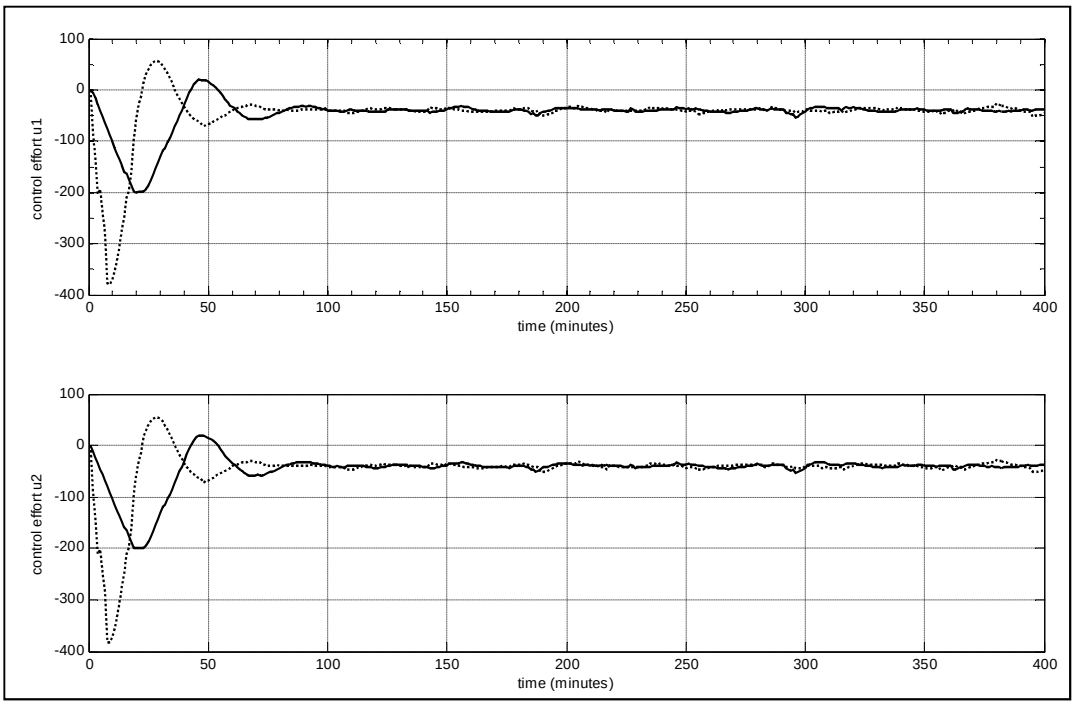


Figure 3.2: Control effort results with standard and improved PSO algorithms for $w = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$ and $\left(K_1 = 0.8, K_2 = 0.8, \tau_{1,2} = 1\right)$.

[12] H.N. Al-Duwaish, S.Z. Rizvi & al. PSO based Hammerstein modeling and predictive control of non linear multivariable boiler. *Control Engineering practice,* pp.1-12, 2010.

[13] S. R. Singiresu: Engineering Optimization: Theory and Practice, Fourth Edition, by John Wiley & Sons, Inc. Hoboken, New Jersey. pp:708-714, & pp:761-768, 2009.

[14] D.J.N Limebeer. The specification and purpose of a controller design case study, *IEEE Conf. Decision Control, Brighton*, U.K, 1991, 1579-1580.

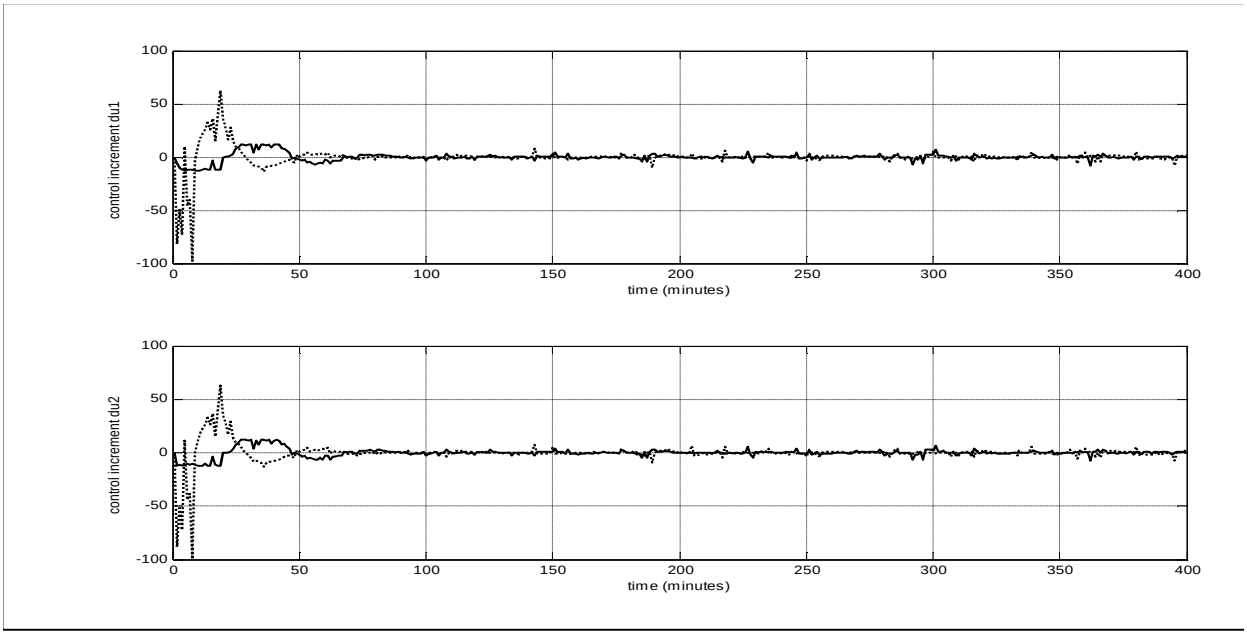


Figure 3.3: Control-increment results with standard and improved PSO algorithms for $w = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$ and $\left(K_1 = 0.8, K_2 = 0.8, \tau_{1,2} = 1\right)$.

Specifications (S_k)***:***	***(S1)*** $y_2(30)$	***(S2)***		***(S3)*** $y_1(400)$	***(S4)***		***(S5)*** $y_2(400)$	***(S6) stable / unstable***	***(S7) Satisfactory/ unsatisfactory constraints***	***(S8) Satisfactory/ unsatisfactory constraints***	***Decision for reasons*** (S_k)
Algorithms:		max(y_2)	time		max(y_1)	time					
Standard PSO	*1.046*	***1.167***	*21*	*0.9985*	*0.202*	*8*	*-0.003*	stable	***unsatisfactory*** $-379.7 \le u_1 \le 56$ $-383 \le u_2 \le 55.3$	***unsatisfactory*** $-97.7 \le \Delta u_1 \le 62.4$ $-99.2 \le \Delta u_2 \le 62.5$	***Rejected algorithm*** $(S_2),(S_7),(S_8)$
Improved PSO	*1.032*	*1.10*	*40*	*1.005*	*0.345*	*12*	*0.0009*	*stable*	*Satisfactory* $-199 \le u_1 \le 20.3$ $-200 \le u_2 \le 19.9$	*Satisfactory* $-12 \le \Delta u_1 \le 12$ $-12 \le \Delta u_2 \le 12$	*Accepted algorithm*

Table 3: Summary of the results (unsatisfactory performances are in bold) for the uncertainty model $\left(K_1 = 0.8, K_2 = 0.8, \tau_{1,2} = 1\right)$and the change set-point reference $w = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$.

Mutual Information and Cross Entropy Framework to Determine Relevant Gene Subset for Cancer Classification

Rajni Bala
Deen Dayal Upadhyaya College, University of Delhi, Delhi, India
E-mail: rbala@ddu.du.ac.in

R.K. Agrawal
School of Computer and Systems Sciences, Jawaharlal Nehru University, Delhi, India
E-mail: rka@mail.jnu.ac.in

Keywords: cancer classification, microarray datasets, mutual information, cross entropy, gene selection

Received: August 5, 2010

Classification of microarray datasets has drawn attention of research community in last few years. Microarray datasets are characterized by high dimension and small sample size. To avoid curse of dimensionality good gene selection methods are needed. Here, we propose a two stage algorithm MICE for finding a small subset of relevant genes responsible for classification of high dimensional microarray datasets. The proposed method is based on the principle of Mutual Information and Cross Entropy. In first stage of algorithm, mutual information is employed to select a set of relevant genes and cross entropy is used to determine independent genes. In second stage, a wrapper based forward feature selection method is used to obtain a set of optimal genes for a given classifier. The efficacy of proposed algorithm is tested on seven well known publicly available microarray datasets. Comparison with other state-of-art methods shows that our proposed algorithm is able to achieve better classification accuracy with less number of genes.

Povzetek: Opisana je metoda za določanje relevantnih skupin genov za rakave bolezni.

1 Introduction

In last few years, classification of microarray datasets has drawn attention of research community. Various machine learning and data mining methods have been applied for classification of microarray datasets. But classification of microarray datasets faces many challenges. One of the main challenges is that such datasets are characterized by large number of genes and small number of samples. This small number of samples compared to the large number of genes wakes up the curse of dimensionality [2]. Also, many of these genes are not relevant to discriminate samples. These irrelevant genes not only have negative effect on the classification accuracy of the classifier but also increase data acquisition cost and learning time. For better classification there is a need to reduce dimension of such datasets.

Dimension Reduction can be done in two ways: feature selection and feature extraction [8]. Feature Selection refers to reducing the dimensionality of measurement space by discarding redundant, noisy and irrelevant features. It leads to saving in measurement cost and the selected features retain their original physical interpretation. In addition, the retained features may be important for understanding the physical process that generates patterns. Feature extraction methods like Principle Component Analysis, Independent Component Analysis utilize all the information contained in the measurement space to obtain a new transformed space and then important features are selected from the new transformed space. Transformed features generated by feature extraction methods may provide a better discriminative ability than the best subset of given features, but these new features may not provide any physical meaning. The choice between feature selection and feature extraction depend on the application domain and specific training data available. In microarray datasets one is not only interested in classifying the sample based on gene expression but also in identifying important genes/features. Hence dimension reduction is normally carried out with feature selection rather than feature extraction. Therefore, efficient feature/gene selection methods are necessary for selecting a small set of informative features/genes. Gene selection not only allows for faster and efficient model building by removing irrelevant, redundant and noisy features but also provides better understanding of genes which lead to a particular disease.

There are numbers of feature selection methods proposed in last few years. These methods broadly fall into two categories: filter and wrapper methods [8]. Most filter methods employ statistical characteristics of data for feature selection which needs less computation. They independently measure the importance of features without involving any learning algorithm. The filter approach does not take into account the learning bias introduced by the final learning algorithm, so it may not be able to select the most relevant set of features for the learning algorithm. Wrapper methods use learning algorithm for selecting feature set. It tends to find features better suited to the predetermined learning algorithm resulting in better performance. But, it is

computationally more expensive since the classifier must be trained for each candidate subset. Hence, when number of features is large wrapper approaches become unfeasible.

Many filter based feature/gene selection methods have been proposed in literature [1, 6, 14, 17, 22, 23]. Broadly they are categorized in two categories: univariate and multivariate evaluation methods. Univariate evaluation methods evaluate the relevance of each feature individually. They are simple and fast therefore appealing and popular [3, 7, 13, 26]. However, they assume that the features are independent of each other. Multivariate approaches, on the contrary, evaluate the relevance of features considering how they function as a group, taking into consideration their dependency [4, 5, 9].

One of the univariate methods determines the relevance of genes by computing the mutual information between each gene and the class label. The genes are ranked based on their relevance with class i.e. in decreasing order of their mutual information and then top *m* genes are selected. In literature, it has been observed that the combination of individual good genes does not necessarily lead to good classification performance. Also, since genes are selected based on the correlation between individual gene and target class, it doesn't capture the correlation among genes. Hence gene subset so obtained may contain redundant genes. A good gene selection method is the one that not only selects the relevant genes but also reduces redundancy among the selected gene subset. In literature, some multivariate methods have been suggested to reduce the redundancy among the selected set of genes [1, 15, 22]. However, these methods consider weighted average of only pairwise correlation instead of considering joint correlation among a set of features. Hence these approaches may not select the optimal feature subset in the presence of large number of redundant features.

In this paper, we have proposed a two stage algorithm MICE for determining an optimal feature subset. In the first stage, a pool of relevant and independent genes is created using Mutual Information (MI) and cross-entropy (CE). In second stage, forward feature selection is used to find a compact feature subset that minimizes classification error (maximizes classification accuracy), from the candidate feature set.

This paper is organized as follows – Section 2 presents some of feature selection methods based on mutual information. Section 3 includes proposed algorithm MICE for gene selection based on mutual information and cross entropy. Experimental results on some well-known publicly available datasets are presented in Section 4. Conclusions are drawn in Section 5.

2 Mutual Information

Mutual information (MI) measures the dependency between a feature set and target class. Mutual information $I(\mathbf{X}_m;\mathbf{C})$ between set of *m* features ($\mathbf{X}_m$) and class label ($\mathbf{C}$) is given by

$$I(\mathbf{X}_m;\mathbf{C}) = \int\int p(\mathbf{X}_m,\mathbf{C}) \log \frac{p(\mathbf{X}_m,\mathbf{C})}{p(\mathbf{X}_m) f(\mathbf{C})} d\mathbf{X}_m d\mathbf{C} \quad (1)$$

Higher value of MI means given feature set represents the class well. Battiti [1] defined feature selection problem as the process of selecting the most relevant *m* features from the initial set of *d* features. Ideally, problem can be solved by maximizing $I(\mathbf{X}_m;\mathbf{C})$, the joint entropy between set of features $\mathbf{X}_m$ and the target class $\mathbf{C}$. But it is often difficult to estimate the joint probability density $p(\mathbf{X}_m)$. For this a greedy feature selection method based on mutual information (MIFS) was proposed by Battiti. Battiti [1] adopted a heuristic criterion for approximating the ideal solution. Battiti's MIFS selects the subset of features which maximizes the information about the class corrected by subtracting a quantity proportional to average MI with the previously selected features. Kwak and Choi [15] proposed a greedy feature selection method called MIFS-U which provides a better estimate of the MI between input features and target class in comparison to MIFS. Peng et al. [22] suggested another variant of Battiti's MIFS as min-redundancy and max-relevance (mRMR) criterion. In this work a heuristic framework was suggested to minimize redundancy and maximize relevance to select important features incrementally [22]. In this incremental approach, mth feature x_j can be selected from the remaining set of features which maximizes the following criterion:

$$\max_{x_j \in X - S_{m-1}} \left[I(x_j, C) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_j, x_i) \right] \quad (2)$$

where S_{m-1} is the set of already selected (m-1) features.

However, MIFS, MIFS-U and mRMR algorithms use incremental search approach which considers weighted average of only pair-wise correlation instead of considering joint correlation among a set of features. Hence, these approaches might not select the optimal feature subset in presence of large number of redundant features. In this paper, we have calculated the joint MI between a set of features and target class C under the assumption that data follows multivariate normal distribution. We have employed cross entropy to determine dependency among selected genes. By combining MI and CE measures we are able to obtain a reduced set of non-redundant relevant genes. This allows us to use a wrapper based forward feature selection at the second stage to search a compact feature subset, from the above gene set, that maximizes classification accuracy.

3 Proposed Method

To measure the relevance of a gene subset, mutual information between gene subset and target class is calculated for which we should have the knowledge of the joint probability density. In reality, the probability

density is not known. So, we can assume parametric form of p($\mathbf{X}_m$) and p($\mathbf{X}_m$|**C**) and the parameters involved in parametric form of probability density can be estimated from the observed data. Here, we assume that the probability density p(X_m|C*)* follows multivariate normal density which is given by

$$p(X_m \mid C) = \frac{1}{(2\pi)^{\frac{m}{2}} \mid \Sigma_c \mid^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(X_m - \mu_c)^t (\Sigma_c)^{-1} (X_m - \mu_c) \right] \quad (3)$$

where μ_c , $\sum_c$ are respectively mean and covariance of class C data. p($\mathbf{X}_m$) is also approximated by a multivariate normal density with mean μ and covariance Σ. Under this approximation, the closed form expression for MI is known. According to Padmanabhan and Dharanipragada [21], for multivariate normal density, the upper bound on mutual information is given by

$$I_{upp}\ (X_m ; C) = \frac{1}{2} \log\ \mid \Sigma \mid - \frac{1}{2} \sum_c\ p_c\ \log\ \mid \Sigma_c \mid \quad (4)$$

Equ. (4) gives an upper bound on joint mutual information between m-dimensional gene vector $\mathbf{X}_m$ and the target class **C**. However, the selected gene subset $\mathbf{X}_m$ may contain redundant features which can degrade the performance of the classifier. We can reduce the redundancy by using cross-entropy. The cross entropy D [12], measures the difference between the two probability distributions $f(X)\ and\ q(X)$ and is given by

$$D\big(f(X), q(X)\big) = \int f(X) \log \frac{f(X)}{q(X)} dX \quad (5)$$

If we consider

$$f(X) = p(x_1, x_2, ..., x_m) \quad (6)$$

And $q(X) = p(x_1)p(x_2)...p(x_m)$ (7)

Then D takes the following form

$$D_m = \int ..... \int p(x_1, x_2, x_3, ..., x_m) \log \frac{p(x_1, x_2, x_3, ..., x_m)}{p_1(x_1) p_2(x_2) ... p_m(x_m)} dx_1 dx_2 .... dx_m \quad (8)$$

If $x_1, x_2, x_3, ..., x_m$ are statistically independent, then $p(x_1, x_2, ..., x_m) = p_1(x_1) p_2(x_2) ..... p_m(x_m)$. In this case D_m becomes zero. When features are not statistically independent, D_m is given by

$$D_m = \int ..... \int p(x_1, x_2, ..., x_m) \log[p(x_1, x_2, ..., x_m) dx_1 dx_2 ... dx_m - \sum_{i=1}^{m} \int p_i(x_i) \log p_i(x_i) dx_i \quad (9)$$

$$D_m = -S + \sum_{i=1}^{m} S_i \quad (10)$$

where

$$S = -\int ..... \int p(x_1, x_2, ..., x_m) \log\ p(x_1, x_2, ..., x_m)\, dx_1 dx_2 ... dx_m$$

and

$$S_i = -\int p(x_i) \log p(x_i) dx_i \quad i = 1, 2, ..., \quad m \quad (11)$$

D_m can be used to measure dependency among genes. D_m is nonnegative i.e. $D_m >= 0$. If genes $x_1, x_2, x_3, ..., x_m$ are independent then $D_m = 0$, whereas if there is a dependency among the set of genes then $D_m > 0$. Higher value of D_m signifies greater dependency among the genes. For m dimensional multivariate normal density the values of joint entropy S and marginal entropy S_i [12] are given as follows:

$$S = \frac{m}{2} \log 2\pi + \frac{1}{2} \log \mid \Sigma \mid + \frac{1}{2} m \quad (12)$$

$$S_i = \frac{1}{2} \log 2\pi + \frac{1}{2} \log \sigma_i^{\ 2} + \frac{1}{2} \quad i = 1, 2, ..., \ m \quad (13)$$

Using equ. (12) and equ. (13), we can rewrite equ. (10) as (14)

$$D_m = \frac{1}{2} \sum_{i=1}^{m} \sigma_i^{\ 2} - \frac{1}{2} \log \mid \Sigma \mid \quad (14)$$

Since the value of D_m increases with number of features so there is need of defining normalized value of D_m. It is known that marginal entropy S_i is always less than equal to joint entropy S i.e. $S_i \le S$ $i = 1, 2, ..., m$. This allows us to write

$$S_1 + S_2 + ... + S_m \le mS \quad (15)$$

Using equ. (10) and equ. (15), we have

$$D_m = S_1 + S_2 + S_3 + ... + S_m - S \le (m-1)S \quad (16)$$

The normalized value of $\overline{D}_m$ is given by

$$\overline{D}_m = \frac{S_1 + S_2 + ... + S_m - S}{(m-1)S} \quad (17)$$

Here, $0 \le \overline{D}_m \le 1$. Zero value of $\overline{D}_m$ corresponds to gene set consisting of independent genes. Higher value of $\overline{D}_m$ signifies more dependence among genes. To consider a set of independent genes, we can choose a threshold T. If the value of $\overline{D}_m$ is less than equal to threshold value T then gene subset is considered as a set of independent genes.

We have employed MI to measure relevance between a subset of genes and class label. Cross entropy is used to measure redundancy among genes. Our proposed algorithm MICE is incremental in nature and consists of two phase. In the first phase a set S of relevant and independent genes is created. We initially start with S as empty set and F a set which contain all the genes. Mutual information of each gene with respect to target class is

estimated using equ 4. The gene which maximizes mutual information is selected. Let it be x_k, then S={x_k }and F=F- {x_k}. Now a set of genes independent of selected gene subset S is created. This is done by calculating the cross entropy of SU{x_j} for all x_j in F. i.e. D(S,x_j). All the features whose D(S,x_j) >T are considered dependent with respect to the geneset S and hence these genes are removed from F. Again, a gene x_i in F which maximizes the mutual information of set S=SU{x_i} with respect to target class, is selected and included in set S. This gene x_i is removed from F. We determine consequently gene subset F which contains genes independent of S using cross entropy. This process is repeated till F becomes empty. In this way we create a set of independent and relevant genes.

In the second stage an optimal set of genes is determined from the gene subset selected in the first stage. To obtain the optimal set we have used a wrapper based forward feature selection. We have used classification accuracy as a criterion in the forward feature selection. The gene subset that maximizes the classification accuracy is selected. The outline of the proposed algorithm MICE is as follows.

MICE Algorithm

Input–Initial Set of genes,Class Labels C,Classifier M

PHASE 1 // to determine a subset of relevant and independent genes S

1. Intialization: Set F="initial set of genes" ; S = Φ //Set of Selected Attributes
2. Choose Threshold value T.
3. For each gene x_i in F calculate $I(x_i;C)$ using (4)
4. Select the gene x_k which maximizes Mutual Information $I(x_i;C)$ i.e. $x_k = \max_i I(x_i, C)$
5. $S = S \cup \{x_k\}$; $F = F - \{x_k\}$
6. Calculate $\overline{D}_m(S, x_j)$ for all $x_j \in F$;

if $\overline{D}_m(S, x_j)$ >T $F = F - \{x_j\}$ //Identifying set of independent genes F with respect to S

7. Choose a gene from $x_k \in F$ which maximizes $I(S, x_k; C)$
8. $S = S \cup \{x_k\}$, $F = F - \{x_k\}$
9. Repeat steps 6-8 till F becomes empty
10. Return S

PHASE 2 // to determine subset of genes which provides maximum classification accuracy

1. Initialization R = Φ
2. For each $x_i \in S$ calculate classification accuracy for classifier M.
3. $[x_k, \max\_acc] = \max_i Classif\_Acc(x_i)$
4. $R = R \cup \{x_k\}$; $S = S - \{x_k\}$; $R\_\min = R$ // R_min is the gene subset corresponding to maximum accuracy
5. For each $x_j \in S$ calculate classification_accuracy of S U{x_j} for classifier M
6. $[x_k, new\_\max\_acc] = \max_i Classif\_Acc(S \cup x_i)$
7. $R = R \cup \{x_k\}, S = S - \{x_k\}$
8. If new_max_acc > max_acc then R_min=R; max_acc=new_max_acc;
9. Repeat steps 5-8 until max_acc=100 or S = Φ
10. Retum R_min, max_acc

4 Experimental Setup and Results

To test the efficacy of our proposed algorithm MICE, we have carried out experiments on seven well known datasets. Colon, Leukemia, Prostate, Lung cancer and Ovary datasets are downloaded from Kent Ridge Bio-medical Dataset data repository [31]. For SRBCT we have used the dataset used by Khan [13]. NCI60 data is downloaded from the NCI Genomics and Bioinformatics Group Datasets resource [32]. The details of these datasets are given in Table 1. Before carrying experiments datasets are normalized using Z-score. In NCI60 dataset one class contained only two samples, so this class is removed from the dataset. Also number of samples belonging to each class is very small, therefore 2000 genes with highest variance are selected and then algorithm is applied on the reduced set of 2000 genes.

Table 1: Datasets Used.

Dataset	No. of Samples	No. of Features	Classes
Colon	62	2000	2
SRBCT	83	2308	4
Leukemia	72	7129	3
Prostate	102	5966	2
Ovary	253	15154	2
Lungcancer	181	12533	2
NCI60	60	2000	9

After normalizing the datasets, the first phase of our proposed algorithm MICE is applied to obtain the subset of relevant and independent genes. We performed experiments with different values of threshold. The value of threshold is varied from 0.1 to 0.9 with an increment of 0.1. It is observed that subset of genes selected is same for threshold values between 0.4 and 0.6. So, the value of threshold T is set as 0.5 in our experiments i.e. all the genes with $\overline{D}_m$ greater than 0.5 are rejected as dependent genes. The number of the reduced genes obtained after phase I of MICE algorithm for each dataset is given in Table 2.

Table 2: Size of Reduced Dataset after Phase I.

Dataset	Original No. Of genes	No. of genes selected
Colon	2000	56
SRBCT	2308	46
Leukemia	7129	54
Prostate	5966	66
Ovary	15154	98
Lungcancer	12533	159
NCI60	2000	60

It can be observed from Table 2 that the number of relevant genes obtained is significantly smaller in comparison to the original gene set. Now once a reduced set of relevant and independent genes is obtained, phase II of MICE algorithm is applied. Phase II of our proposed algorithm uses a forward feature selection strategy with a known classifier to obtain a set of genes which maximizes classification accuracy. We have employed four classifiers: linear discriminant classifier (LDC), quadratic discriminant classifier (QDC), k-nearest neighbor (KNN) and support vector machine (SVM). Classification accuracy is calculated using leave-one-out cross validation (LOOCV). The algorithm is implemented in matlab. For KNN the optimal value of k is chosen. In SVM linear kernel is used. Results of our algorithm MICE are presented in Table 3. It contains maximum classification accuracy achieved along with the number of genes obtained by our algorithm MICE. It can be observed from Table 3 that our algorithm MICE is able to achieve maximum classification accuracy with small number of genes. For all the classifiers used we are able to achieve good classification accuracy with few numbers of genes. We compared the performance of our algorithm MICE with well known algorithm mRMR given by Peng et al (2005). The code of mRMR_d and mRMR_q is taken from [30]. As a preprocessing step, datasets are discretized into 3 values using $\mu \pm \sigma$. The values which are less than $\mu - \sigma$ are assigned -1 , values between $\mu - \sigma$ and $\mu + \sigma$ are assigned value 0 and rest 1. Discretized data are passed to mRMR_d and mRMR_q and a ranked list of features is obtained from both the methods. Using the ranked list of genes obtained from mRMR, classification accuracy (LOOCV) is calculated as genes are added one by one. The maximum classification accuracy along with the minimum number of genes obtained for each classifier is shown in Table 3. We can observe the following from Table 3:

1. For Colon dataset a maximum accuracy of 96.77% is achieved with genes selected by our proposed algorithm MICE. It is achieved with 14 genes with QDC. For KNN results of MICE are better than mRMR. For SVM and LDC classification accuracy using MICE is same as mRMR.
2. For SRBCT dataset maximum classification accuracy of 100% is achieved with 15 genes with LDC and SVM classifier using MICE. For QDC results of MICE are better than mRMR. Only for KNN performance of mRMR is better.

Table 3: Comparison of maximum classification accuracy along with number of genes for different classifiers using various genes selection methods.

Dataset	Classifier	MICE (LOOCV)	mRMR_d (LOOCV)	mRMR_q (LOOCV)
Colon	LDC	**91.94(9)**	**91.94(3)**	90.32(6)
	QDC	**96.77(14)**	88.71(6)	87.10(27)
	KNN	**96.77(31)**	93.55(5)	91.94(32)
	SVM	**93.55(23)**	**93.55(18)**	93.55(50)
SRBCT	LDC	**100(15)**	97.59(21)	98.80(30)
	QDC	**98.80(10)**	49.40(66)	71.08(73)
	KNN	98.80(15)	100(83)	**100(29)**
	SVM	**100(15)**	100(19)	**100(15)**
Leukemia	LDC	**98.61(12)**	97.22(35)	98.61(14)
	QDC	**100(6)**	95.83(5)	88.89(9)
	KNN	**100(15)**	97.22(75)	97.22(80)
	SVM	98.61(7)	98.61(60)	**100(18)**
Prostate	LDC	**97.06(6)**	96.10(6)	96.08(8)
	QDC	**96.08(4)**	92.16(22)	89.22(9)
	KNN	**98.04(9)**	97.16(14)	98.04(27)
	SVM	**99.02(45)**	98.04(87)	98.04(26)
Ovary	LDC	**100(5)**	100(4)	100(8)
	QDC	**100(4)**	100(4)	100(8)
	KNN	**100(4)**	100(4)	100(10)
	SVM	**100(3)**	**100(5)**	100(8)
LungCancer	LDC	**100(44)**	100(36)	99.45(14)
	QDC	**100(5)**	100(40)	100(41)
	KNN	**100(3)**	100(20)	100(5)
	SVM	**100(4)**	100(23)	100(6)
NCI60	LDC	**84.48(26)**	75.86(94)	82.76(67)
	QDC	**56.90(5)**	56.90(5)	43.10(5)
	KNN	86.21(18)	**89.66(95)**	87.93(98)
	SVM	87.93(36)	81.03(34)	**89.66(97)**

3. For Leukemia dataset maximum classification accuracy of 100% is achieved with 6 genes in QDC classifier and 15 genes in KNN classifier using MICE. Same classification accuracy is achieved using MICE with LDC as with mRMR but with less number of genes.
4. For prostate dataset maximum classification accuracy of 99.02% is achieved with 45 genes in SVM classifier using MICE. Also for other classifiers, the accuracy achieved by our algorithm MICE is better in comparison to mRMR.
5. For Ovary dataset maximum classification accuracy of 100% is achieved for all classifiers using different gene selection methods but the number of genes selected by our proposed method MICE is same or comparitively less.
6. For Lungcancer dataset maximum classification accuracy of 100% is achieved for all classifiers using genes selected by our method MICE . The number of genes selected by MICE are significantly less in comparison to mRMR with QDC, KNN and SVM. The best result is obtained for KNN using only 3 genes.

   For NCI60 dataset maximum classification accuracy of 87.93% is achieved with 36 genes in SVM classifier using MICE. For LDC and QDC classifier accuracy achieved using genes selected by MICE is better in comparison to mRMR. For KNN and SVM performance of mRMR is better than MICE.
7. The performance of our proposed algorithm MICE with different classifiers is better in comparison to

mRMR algorithm in terms of classification accuracy in most cases. Our proposed algorithm MICE also provides a smaller subset of relevant genes for most of the cases.

The comparative results of classification accuracy obtained by different methods as the genes are added one by one for Leukemia dataset are shown in Figure 1. It can be observed from Figure 1 that classification accuracy obtained by our algorithm is more in comparison to mRMR_d and mRMR_q with the same number of genes for all the classifier. Similar results are observed for other datasets also.

To check the relevance of the selected genes subset we carried out 10 fold cross validation using the selected genes for all the datasets. Experiment is repeated 10 times. The average accuracy of 10 runs along with standard deviation is given in Table 4. It can be observed from the table that the 10 fold cross validation accuracy does not deviate much from LOOCV accuracy except for NCI60 dataset with QDC classifier. This shows that the gene set selected is not over fitted.

Table 4: 10 fold cross-validation accuracy achieved by the genes selected by MICE for different classifier. Quantity in bracket represents standard deviation.

Dataset	Classifier			
	LDC	QDC	KNN	SVM
Colon	91.13(1.14)	86.32(3.30)	**96.32(1.41)**	91.13(1.90)
SRBCT	99.40(0.85)	96.39(2.13)	98.07(1.3)	**99.52(1.02)**
Leukemia	96.94(1.71)	98.89(1.28)	**99.44(0.97)**	98.33(0.59)
Prostate	96.67(0.69)	96.20(1.08)	96.78(0.93)	**96.96(1.26)**
Ovary	**100(0)**	**100(0)**	99.88(0.27)	**100(0)**
LungC	99.17(0.39)	99.83(0.27)	99.89(0.23)	**100(0)**
NCI60	75.69(3.68)	41.72(2.12)	**81.72(2.72)**	79.66(4.36)

In literature a number of gene selection methods have been proposed and applied on these datasets. In Table 5, we have compared performance of our proposed method in terms of classification accuracy achieved and number of genes selected with some already existing gene selection methods in literature [6, 9, 10, 11, 13, 16, 18, 19, 20, 24, 25, 26, 27, 28, 29]. From Table 5, it can be observed that the performance of our proposed algorithm MICE is significantly better in terms of both classification accuracy and number of genes selected.

5 Conclusion

In this paper, we proposed a two stage algorithm MICE for finding a small subset of relevant genes responsible for better classification of high dimensional microarray datasets. The proposed method is based on the principle of Mutual Information and Cross Entropy. In first stage of algorithm, Mutual information is employed to select a set of relevant genes and Cross Entropy is used to determine independent genes. This provides a set of independent and relevant genes and reduces the size of gene set significantly. This allows us to use wrapper approach at the second stage. The use of wrapper method at the second stage gives a better subset of genes.

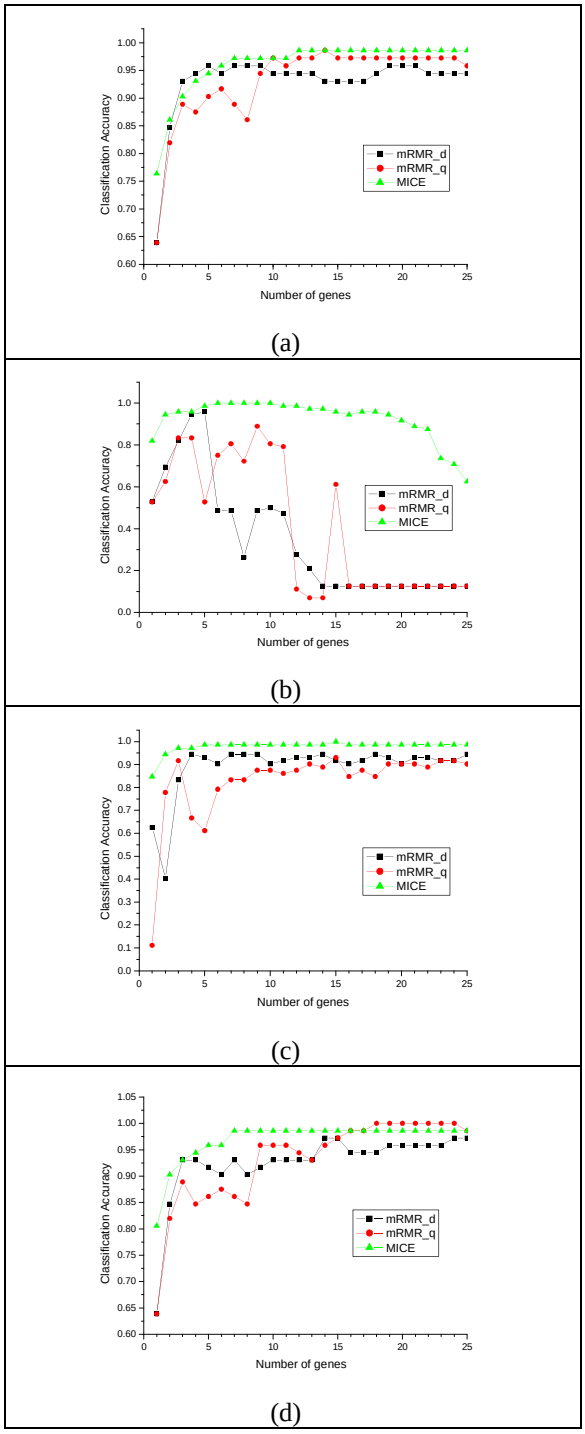


Figure 1: Classification accuracy Vs number of genes for Leukemia dataset using (a) LDC (b) QDC (c) KNN (d) SVM.

Experimental results show that our proposed method MICE is able to achieve a better classification accuracy with small number of genes. In case of Lungcancer and Ovary 100% accuracy is achieved with 3 genes. For other datasets, the method provides competitive accuracy. Comparisons with other state-of-art methods show that our proposed algorithm is able to achieve better or comparable accuracy with less number of features in all the datasets.

Table 5: Comparison of Maximum Classification accuracy and number of genes selected with other state of art methods.

COLON	
Proposed method	**96.77(14)**
PSO+ANN [27]	88.7
Chen and Zhao [29]	95.2
BIRSW [24]	85.48(3.50)
BIRSF [24]	85.48(7.40)
OVARY	
Proposed Method	**100(3)**
PSO+ANN [27]	97.0
NB [10]	96.2
BKS [10]	97.0
DT[10]	97.8
Chen and Zhao [29]	99.6
PROSTATE	
Proposed Method	**99.02(45)**
GAKNN [16]	84.6(205)
BIRS [24]	91.2(3)
Hong and Cho [[10]	96.3(79)
NCI60	
Proposed Method	**87.93(36)**
Jirapech-Umpai [11]	76.23
Liu [19]	88.52
Ooi [20]	85.37
Lin [18]	87.80
ReliefF/SVM [28]	58.33(30)
mRMR/RelieF [28]	68.33(30)
LEUKEMIA	
Proposed Method	**100(6)**
GS2+KNN [27]	98.6(10)
GS1+SVM [27]	98.6(4)
Cho's+SVM [27]	98.6(80)
Ftest + SVM [27]	98.6(33)
Fu and Liu [6]	97.0(4)
Guyon [9]	100(8)
Tibsrani [26]	100(21)
Chen and Zhao [29]	98.6
SRBCT	
Proposed Method	**100(15)**
GS2+SVM [27]	100(96)
GS1+SVM [27]	98.8(34)
Cho's+SVM [27]	98.8(80)
Ftest + SVM [27]	100(78)
Fu and Liu [6]	100(19)
Tibsrani [26]	100(43)
Khan [13]	100(96)
LUNGCANCER	
Proposed Method	**100(3)**
GS2+KNN [27]	93.1(44)
GS1+SVM [27]	98.6(4)
Cho's+SVM [27]	98.6(80)
Ftest + SVM [27]	98.6(94)
Shah and Kaushik [25]	100(8)
PSO+ANN [27]	98.3
Chen and Zhao [29]	98.3
GAKNN [16]	95.6(325)
Hong and Cho [10]	99.4(135)

References

[1] Battiti R (1994), Using mutual information for selecting features in supervised neural net learning, IEEE Trans. Neural Network 5(4) pp 537-550.

[2] Bellman R (1961), Adaptive Control Processes: A Guided Tour, Princeton University Press.

[3] Ben-Dor A, Bruhn L, Friedman N, Nachman I, Schummer M and Yakhini (200), Tissue classification with gene gene expression profiles, In Proceedings of the fourth annual international conference on Computational molecular biology, pp 54-64, ACM Press.

[4] Bhattacharya C, Grate LR, Rizki A, Radisky D, Molina FJ, Jordan MI, Bissell MJ and Mian IS 92003), Simultaneously classification and relevant feature identification in high dimensional spaces: application to molecular profiling data, Signal Processing 83(4).

[5] Bo T and Jonassen I (2002), New feature subset selection procedures for classification of expression profiles, Genome biology 3.

[6] Fu LM and Liu CSF (2005), Evaluation of gene importance in microarray data based upon probability of selection, BMC Bioinformatics 6(67).

[7] Golub TR, Slonim DK, Tamayo, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri, Bloomfield CD and Lander ES (1999), Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science 286 pp 531-537.

[8] Guyon I and Elisseeff A (2003), An Introduction to Variable and feature Selection, Journal of Machine Learning Research (3):1157-1182.

[9] Guyon I, Weston J, Barnhill S, Vapnik V (2003), Gene Selection for cancer classification using support vector machine, Machine Learning (46) pp 263-268.

[10] Hong JH and Cho SB (2006), The classification of cancer based on DNA microarray data that uses diverse ensemble genetic programming, Artif. Intell. Med. 36 pp 43–58.

[11] Jirapech-Umpai T and Aitken S (2005), Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes, BMC Bioinformatics 6:148.

[12] Kapur JN, Kesavan HK (1992) Entropy Optimization Principles with Applications, Academic Press.

[13] Khan J, Wei S, Ringner M, Saal LH, Ladanyi M, Westermann F (2001), Classification and diagnosis prediction of cancers using gene expression profiling and artificial neural networks, Nat. Med 7 pp 673-679.

[14] Kohavi R and John G (1997), Wrapper for feature subset selection, Artificial Intelligence (1-2) pp 273-324.

[15] Kwak N and Choi CH (2002), Input Feature Selection for classification problems, IEEE Trans. Neural Netw 3(1) pp 143-159.

[16] Li L,Weinberg CR, Darden TA, Pedersen LG (2001), Gene Selection for sample classification based on gene expression data: Study of sensitivity to choice of parameters of the GA/KNN method, Bioinformatics 17(12) pp 1131-1142.

[17] Li T ,Zhang C, Ogihara M (2004), Comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression, Bioinformatics (20) pp 2429-2437.

[18] Lin TC, Liu RS, Chen CY, Chao YT and Chen SY (2006), Pattern classification in DNA microarray data of multiple tumor types, Pattern Recognition, 39 pp 2426-2438.

[19] Liu JJ, Cutler G, Li WX, Pan Z, Peng SH, Hoey T, Chen LB and Ling XFB (2005), Multiclass cancer classification and biomarker discovery using GA-based algorithms. Bioinformatics 21 pp 2691-2697.

[20] Ooi CH, Tan P (2003), Genetic algorithms applied to multi-class prediction for the analysis of gene expression data, Bioinformatics, 19 pp 37-44.

[21] Padmanabhan M and Dharanipragada S (2005), Maximizing Information Content in Feature Extraction, IEEE Transaction on speech and audio processing 13(4).

[22] Peng H, Long F, Ding C (2005), Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance and Min-Redundancy, IEEE Trans. On Pattern Analysis and Machine Intelligence 27 pp 1226-1238.

[23] Ramaswamy S, Tamayo P (2001), Multiclass cancer diagnosis using tumour gene expression signature, Proc Natl Acad Sci, USA, 98(26) pp 15149-15154.

[24] Ruiz R, Riqueline J C, Aguilar-Ruiz JS (2006), Incremental wrapper based gene selection from microarray data for cancer classification, Pattern Recognition 39(12) pp 2383-2392.

[25] Shah, S and Kusiak A (2007), Cancer gene search with Data Mining and Genetic Algorithms, Computer in Biology medicine, Elsevier 37(2) pp 251-261.

[26] Tibsrani R , Hastie T, Narasimhan B and Chu G (2002), Diagnosis of multiple cancer types by shrunken centriods of gene expression, Proc. Natl Acad. Sci., USA (99) pp 6567-6572.

[27] Yang K, Cai Z, Li J, Lin G (2006), A stable gene selection in microarray data analysis, BMC Bioinformatics 7:228 .

[28] Yi Zhang, Chris HQ, Ding, Tao Li (2007), A Two-Stage Gene Selection Algorithm by Combining ReliefF and mRMR. BIBE 2007 pp 164-171.

[29] Chen Y and Zhao Y (2008), A novel ensemble of classifiers for microarray data classification, Applied Soft computing (8) pp 664-1669.

[30] http://www.mathworks.com/matlabcentral/fileexchange/14608.

[31] http://datam.i2r.a-star.edu.sg/datasets/krbd/

[32] http://discover.nci.nih.gov/datasetsNature2000.jsp

An Intelligent Indoor Surveillance System

Rok Piltaver, Erik Dovgan and Matjaž Gams
Jožef Stefan Institute, Department of Intelligent Systems,
Jamova cesta 39, 1000 Ljubljana, Slovenia.
E-mail: rok.piltaver@ijs.si, erik.dovgan@ijs.si, matjaz.gams@ijs.si

Keywords: intelligent system, fuzzy logic, expert system, real-time locating system, surveillance

Received: February 2, 2011

The development of commercial real-time location system (RTLS) enables new ICT solutions. This paper presents an intelligent surveillance system for indoor high-security environments based on RTLS and artificial intelligence methods. The system consists of several software modules each specialized for detection of specific security risks. The validation shows that the system is capable of detecting a broad range of security risks with high accuracy.

Povzetek: Predstavljen je varnostni sistem za nekaj sob z uporabo RTLS.

1 Introduction

Security of people, property, and data is becoming increasingly important in today's world. Security is ensured by physical protection and technology, such as movement detection, biometric sensors, surveillance cameras, and smart cards. However, the crucial factor of most security systems is still a human [7] , providing the intelligence to the system. The security personnel has to be trustworthy, trained and motivated, and in good psychically and physical shape. Nevertheless, they are still human and as such tend to make mistakes, are subjective and biased, get tired, and can be bribed. For example, it is well known that a person watching live surveillance video often becomes tired and may therefore overlook a security risk. Another problem is finding trustworthy security personnel in foreign countries where locals are the only candidates for the job.

With that in mind there is an opportunity of using the modern information-communication technology in conjunction with methods of artificial intelligence to mitigate or even eliminate the human shortcomings and increase the level of security while lowering the overall security costs. Our first intelligent security system that is focused on the entry control is described in [5] . In this paper we present a prototype of an intelligent indoor-surveillance system (i.e. it works in the whole indoor area and not only at the entry control) that automatically detects security risks.

The prototype of an intelligent security system, called "Poveljnikova desna roka" (PDR, eng. commander's right hand), is specialized for surveillance of personnel, data containers, and important equipment in indoor high-security areas (e.g., an archive of classified data with several rooms). The system is focused on the internal threats; nevertheless it also detects external security threats. It detects any unusual behaviour based on user-defined rules and automatically extracted models of the usual behaviour. The artificial intelligence methods enable the PDR system to model usual and to recognize unusual behaviour. The system is capable of autonomous learning, reasoning and adaptation. The PDR system alarms the supervisor about unusual and forbidden activities, enables an overview of the monitored environment, and offers simple and effective analysis of the past events. Tagging all personnel, data containers, and important equipment is required as it enables real-time localization and integration with automatic video surveillance. The PDR system notifies the supervisor with an alarm of appropriate level and an easily comprehensible explanation in the form of natural language sentences, tagged video recordings and graphical animations. The PDR system detects intrusions of unidentified persons, forbidden actions of known and unknown persons and unusual activities of tagged entities. The concrete scenarios detected by the system include thefts, sabotages, staff negligence and insubordination, unauthorised entry, unusual employee behaviour and similar incidents.

The rest of the paper is structured as follows. Section 2 summarizes the related work. An overview of software modules and a brief description of used sensors are given in Section 3. Section 4 describes the five PDR modules, including the Expert System Module and Fuzzy Logic Module in more detail. Section 5 presents system verification while Section 6 provides conclusions.

2 Related Work

There has been a lot of research in the field of automatic surveillance based on video recordings. The research ranges from extracting low level features and modelling of the usual optical flow to methods for optimal camera positioning and evaluating of automatic video surveillance systems [8] . There are many operational implementations of such system increasing the security in public places (subway stations, airports, parking lots).

On the other hand, there has not been much research in the field of automatic surveillance systems based on real-time locating systems (RTLS), due to the novelty of sensory equipment. Nevertheless, there are already some simple commercial systems with so called room accuracy RTLS [20] that enable tracking of objects and basic

alarms based on if-then rules [18] . Some of them work outdoors using GPS (e.g., for tracking vehicles [21]) while others use radio systems for indoor tracking (e.g., in hospitals and warehouses). Some systems allow video monitoring in combination with RTLS tracking [19] .

Our work is novel as it uses several complex artificial intelligence methods to extract models of the usual behaviour and detect the unusual behaviour based on an indoor RTLS. In addition, our work also presents the benefits of combining video and RTLS surveillance.

3 Overview of the PDR System

This section presents a short overview of the PDR system. The first subsection presents the sensors and hardware used by the system. The second subsection introduces software modules. Subsection 3.3 describes RTLS data pre-processing and primitive routines.

3.1 Sensors and other hardware

The PDR system hardware includes a real-time locating system (RTLS), several IP video cameras (Figure 1), a processing server, network infrastructure, and optionally one or more workstations, such as personal computers, handheld devices, and mobile phones with internet access, which are used for alerting the security personnel.

RTLS provides the PDR system with information about locations of all personnel and important objects (e.g. container with classified documents) in the monitored area. RTLS consists of sensors, tags, and a processing unit (Figure 1). The sensors detect the distance and the angle at which the tags are positioned. The processing unit uses these measurements to calculate the 3D coordinates of the tags. Commercially available RTLS use various technologies: infrared, optical, ultra-sound, inertial sensors, Wi-Fi, or ultra-wideband radio. The technology determines RTLS accuracy (1 mm – 10 m), update frequency (0.1 Hz – 120 Hz), covered area (6 – 2500 m^2), size and weight of tags and sensors, various limitations (e.g., required line of sight between sensors and tags), reliability, and price (2.000 – 150.000 €) [13] . PDR uses Ubisense RTLS [15] that is based on the ultra-wide band technology and is among the more affordable RTLSs. It uses relatively small and energy efficient active tags, has an update rate of up to 9 Hz and accuracy of ±20 cm in 3D space given good conditions. It covers areas of up to 900 m^2 and does not require line of sight.

The advantages of a RTLS are that people feel more comfortable being tracked by it than being filmed by video cameras and that localization with a RTLS is simpler, more accurate, and more robust than localization from video streams. On the other hand, RTLS is not able to locate objects that are not marked with tags. Therefore, the most vital areas need to be monitored by video cameras also in order to detect intruders that do not wear RTLS tags. However, only one PDR module requires video cameras, while the other four depend on RTLS alone. Moreover, the cameras enable on-camera processing, therefore only extracted features are sent over the network.

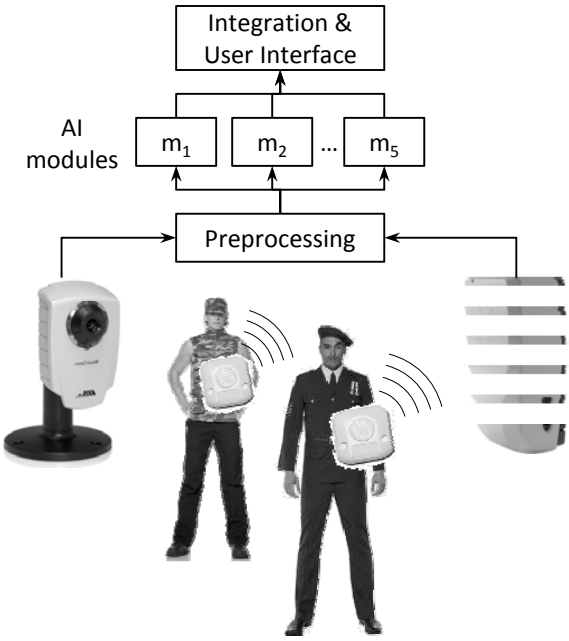


Figure 1: Overview of the PDR system.

3.2 Software structure

The PDR software is divided into five modules. Each of them is specialized for detecting a certain kind of abnormal behaviour (i.e., a possible security risk) and uses an appropriate artificial intelligence method for detecting it. The modules reason in real time independently of each other and asynchronically trigger alarms about detected anomalies. Three of the PDR modules are able to learn automatically while the other two use predefined knowledge and knowledge entered by the supervisor. The Video Module detects persons without tags and is the only module that needs video cameras. The Expert System Module is customisable by the supervisor, who enters information about forbidden events and actions in the form of simple rules, thus enabling automatic rule checking. The three learning modules that automatically extract models of the usual behaviour for each monitored entity and compare current behaviour with it in order to detect abnormalities are Statistic, Macro and Fuzzy Logic Modules. The Statistic Module collects statistic information about entity movement such as time spent walking, sitting, lying etc. The Macro model is based on macroscopic properties such as the usual time of entry in certain room, day of the week etc. Both modules analyse relatively long time intervals while the Fuzzy Logic Module analyses short intervals. It uses fuzzy discretization to represent short actions and fuzzy logic to infer whether they are usual or not.

3.3 RTLS data pre-processing and primitive routines

Since the used RTLS has relatively low accuracy and relatively high update rate, a two-stage data filtering is used to increase the reliability and to mitigate the negative effect of the noisy location measurements. In the first stage, median filter [1] with window size 20 is

used to filter sequences of x, y, and z coordinates of tags. Equation (1) gives the median filter equation for direction x. The median filter is used to correct the RTLS measurements that differ from the true locations by more than ~1.5 m and occur in up to 2.5 % of measurements. Such false measurements are relatively rear and occur only in short sequences (e.g., probability of more than 5 consecutive measurements having a high error is very low) therefore the median filter corrects these errors well.

$$\tilde{x}_n = med\{x_{n-10}, x_{n-9}, \ldots, x_{n+8}, x_{n+9}\} \quad (1)$$

The second stage uses a Kalman filter [6] that performs the following three tasks: smoothing of the RTLS measurements, estimating the velocities of tags, and predicting the missing measurements. Kalman filter state is a six dimensional vector that includes positions and velocities in each of the three dimensions. The new state is calculated as a sum of the previous position (e.g. x_n) and a product between the previous velocity (e.g. $v_{x,n}$) and the time between the consecutive measurements Δ_t for each direction separately. The velocities remain constant. Equation (2) gives the exact vector formula used to calculate the next state of the Kalman filter. The measurement noise covariance matrix was set based on RTLS system specification, while the process noise covariance matrix was fine-tuned experimentally.

Once the measurements are filtered, primitive routines can be applied. They are a set of basic pre-processing methods used by all the PDR modules and are robust to noise in 3D location measurements. They take short intervals of RTLS data as input and output a symbolic representation of the processed RTLS data.

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \\ \hdashline v_{x,n+1} \\ v_{y,n+1} \\ v_{z,n+1} \end{bmatrix} = \left[\begin{array}{ccc:ccc} 1 & 0 & 0 & \Delta_t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta_t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta_t \\ \hdashline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x_n \\ y_n \\ z_n \\ \hdashline v_{x,n} \\ v_{y,n} \\ v_{z,n} \end{bmatrix} \quad (2)$$

The first primitive routine detects in which area (e.g., a room or a user-defined area) a given tag is located, when it has entered, and when it has exited from the area. The routine takes into account the positions of walls and doors. A special method is used to handle the situations when a tag moves along the boundary between two areas that are not separated by a wall.

The second primitive routine classifies the posture of a person wearing a tag into: standing, sitting, or lying. A parameterized classifier, trained on pre-recorded and hand-labelled training data, is used to classify the sequences of tag heights into the three postures. The algorithm has three parameters: the first two are thresholds t_{lo} and t_{hi} dividing the height of a tag into the three states, while the third parameter is tolerance d. The algorithm stores the previous posture and adjusts the boundaries between the postures according to it (Figure 2). If the current state is below the threshold t_i, it is increased by d, otherwise it is decreased by d. The new posture is set to the posture that occurs most often in the window of consecutive tag heights according to the dynamically set thresholds. The thresholds t_{lo} and t_{hi} were obtained from the classification tree that classifies the posture of a person based on the height of a tag. It was trained on half an hour long manually labelled recording of lying, sitting and standing.

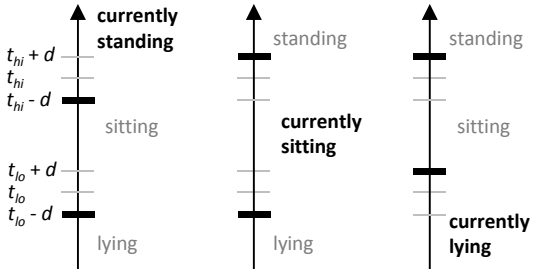


Figure 2: Dynamic thresholds.

The third group of primitive routines is a set of routines that detect whether a tag is moving or not. This is not a trivial task due to the considerable amount of noise in the 3D location data. There are separate routines for detecting movement of persons, movable objects (e.g., a laptop) and objects that are considered stationary. The routines include hardcoded, handcrafted, common sense algorithms and a classifier trained on extensive, pre-recorded, hand labelled training set. The classifier uses the following attributes calculated in a sliding window with size 20: the average speed, the approximate distance travelled, sum of consecutive position distances, and the standard deviation of moving direction. The classifier was trained on more than two hours long hand-labelled recording of consecutive moving and standing still. Despite the noise in the RTLS measurements the classification accuracy of 95 % per single classification was achieved. [12] describes the classifier in more detail.

The final group of routines detects if two tags (or a tag and a given 3D position) are close together by comparing the short sequences of tags' positions. There are separate methods used for detecting distances between two persons (e.g., used to detect if a visitor is too far away from its host), between a person and an object, and between a person and a given 3D location (e.g., used to assign tags of moving persons to locations of moving objects detected by video processing).

All of the described primitive routines are robust to the noise in RTLS measurements and are specialized for the PDR's RTLS. Primitive routines' parameters were tuned according to the noise of the RTLS and using data mining tools Orange [4] and Weka [16] . In case of more accurate RTLS, the primitive routines could be simpler and more accurate. Nevertheless, the presented primitive routines perform well despite the considerable amount of noise. This is possible because of the relatively high update rate. If it was significantly lower, the primitive routines would not work as well. Therefore, the accuracy, reliability and update rate of RTLS are crucial for the performance of the entire PDR system.

4 PDR Modules

4.1 Expert System Module

The Expert System Module enables the supervisor to customize the PDR system according to his/her needs by setting simple rules that must not be violated. It is the simplest and the most reliable module of the PDR system [11] . It is capable of detecting a vast majority of the predictable security risks, enables simple customization, is reliable, robust to noise, raises almost no false alarms, and offers comprehensible explanation for the raised alarms. In addition, it does not suffer from the typical problems common to the learning modules/algorithms, such as long learning curve, difficulty to learn from unlabeled data, relatively high probability of false alarms, and the elusive balance between false negative and false positive classifications. The expert system consists of three parts described in the following subsections.

4.1.1 Knowledge base

Knowledge base of an expert system contains the currently available knowledge about the state of the world. The knowledge base of PDR expert system consists of RTLS data, predefined rules, and user-defined rules. The first type of knowledge is in form of data stream, while the latter two are in form of if-then rules.

The expert system gets the knowledge about objects' positions from the RTLS data stream. Each unit of the data stream is a filtered RTLS measurement that contains a 3D location with a time stamp and a RTLS tag ID.

User-defined rules enable simple customization of the expert system according to specific supervisor's needs by specifying prohibited and obligatory behaviour. Supervisor can add, edit, view, and delete the rules at any time using an intuitive graphic user interface. There are several rule templates available. The supervisor has to specify only the missing parameters of the rules, such as for which entities (tags), in which room(s) or user-defined areas(s), and at which time the rules apply.

For instance, a supervisor can choose to add a rule based on the following template: "Person P must be in the room R from time T_{min} to time T_{max}." and set P to John Smith, R to the hallway H, T_{min} to 7 am, and T_{max} to 11 am. Now the expert system knows that John must be in the hallway from 7 am to 11 am. If he leaves the hallway during that period or if he does not enter it before 7 am, the PDR supervisor will be notified.

Some of the most often used rule templates are listed below:

- Object O_i is not allowed to enter area A_i.
- Object O_i can only be moved by object O_j.
- Object O_i must always be close to object O_j.

The predefined rules are a set of rules that are valid in any application where PDR might be used. Nevertheless, the supervisor has an option to turn them on or off. Predefined rules define when alarms about hardware failures should be triggered.

4.1.2 Inference engine

The inference engine is the part of the PDR expert system that deduces conclusions about security risks from the knowledge stored in the knowledge base. The inference process is done in real-time. First, the RTLS data stream is processed using the primitive routines. Second, all the rules related to a given object (e.g., a person) are checked. If a rule fires, an alarm is raised and an explanation for the raised alarm is generated. An example is presented in the next paragraph.

Suppose that the most recent 3D location of John Smith's tag (from the previous example) has just been received at 8:32 am. The inference engine checks all the rules concerning John Smith. Among them is the rule R_i that says: "John Smith must be in the hallway H from 7 am to 11 am." The inference engine calls the primitive routine that checks whether John is in the hallway H. There are two possible outcomes. In the first outcome, he is in the hallway H, therefore, the rule R_i is not violated. If John was not in the hallway H in the previous instant, there is an ongoing alarm that is now ended by the inference engine. In the second outcome, John is not in the hallway H; hence the rule R_i is violated at this moment. In this case the inference engine checks if there is an ongoing alarm about John not being in the hallway H. If there is no such ongoing alarm the inference engine triggers a new alarm. On the other hand, if there is such an alarm, the inference engine knows that the PDR supervisor was already notified about it.

If an alarm was raised every time a rule was violated, the supervisors would get flooded with alarm messages. Therefore, the inference engine automatically decreases the number of alarm messages and groups alarm messages about the same incident together so that they are easier to handle by the PDR supervisor. The method will be illustrated with an example. Because of the noise in 3D location measurements the inference engine does not trigger or end an alarm immediately after the status of rule R_i (violated/not violated) changes. Instead it waits for more RTLS measurements and checks the trend in the given time window: if there are only few instances when the rule was violated they are considered as noise. On the other hand, if there are many (over the global threshold set by the supervisor) such instances, then the instances when rule was not violated are treated as noise. Two consecutive alarms that are interrupted by a short period of time will therefore result in a single alarm message. A short period in which a rule seems to be violated because of the noise in RTLS data, however, will not trigger an alarm. The grouping of alarms works in the following way: the inference engine groups the alarm messages based on the two rules R_i and R_j together if at the time when rule R_i is violated another rule R_j concerning John Smith or hallway H is violated too. As a result, the supervisor has to deal with fewer alarm messages.

4.1.3 Generating alarm explanations

The Expert System Module also provides the supervisor with an explanation of the alarm. It consists of three

parts: explanation in natural language, graphical explanation, and video recording of the event.

Each alarm is a result of a particular rule violation. Since each rule is an instance of a certain rule template, explanations are partially prepared in advance. Each rule template has an assigned pattern in the form of a sentence in natural language with some objects and subjects missing. In order to generate the full explanation, the inference engine fills in the missing parts of the sentence with details about the objects (e.g., person names, areas, times, etc.) related to the alarm.

Figure 3: Video explanation of an alarm.

Graphical explanation is given in form of a ground plan animation and can be played upon supervisors' request. The inference engine determines the start and the end times of an alarm and sets the animation to begin slightly before the alarm was caused and to end slightly after the causes for the alarm are no longer present. The animation is generated from the recorded RTLS data and the ground plan of the building under surveillance. The animated objects (e.g., persons, objects, areas) that are relevant to the alarm are highlighted with red colour.

If a video recording of the incident that caused an alarm is available it is added to the alarm explanation. Based on the location of the person that caused the alarm, the person in the video recording is marked with a bounding rectangle (Figure 3). The video explanation is especially important if an alarm is caused by a person or object without a tag.

The natural language explanation, ground plan animation, and video recordings with embedded bounding rectangles produced by the PDR expert system efficiently indicate when and to which events the security personnel should pay attention.

4.2 Video Module

The video Module periodically checks if the movement detected by the video cameras is caused by people marked with tags. If it detects movement in an area where no authorised humans are located, it triggers an alarm. It combines the data about tag locations and visible movement to reason about unauthorised entry.

Data about visible moving objects (with or without tags) is available as the output of video pre-processing. Moving objects are described with their 3D locations in the same coordinate system as RTLS data, sizes of their bounding boxes, similarity of the moving object with a human, and a time stamp. The detailed description of the algorithm that processes the video data (developed at the Faculty of Electrical Engineering, University of Ljubljana, Slovenia) can be found in [9] and [10] .

The Video Module determines the pairing between the locations of tagged personnel and the detected movement locations. If it determines that there is movement in a location that is far enough from all the tagged personnel, it raises an alarm. In this case the module reports moving of an unauthorised person or an unknown object (e.g., a robot) based on the similarity between the moving object and a person. The probability of false alarms can be reduced if several cameras are used to monitor the area from various angles. It also enables more accurate localization of moving objects.

Whenever the Video Module triggers an alarm it also offers an explanation for it in form of video recordings with embedded bounding boxes highlighting the critical areas (Figure 3). The supervisor of the PDR system can quickly determine whether the alarm is true or false by checking the supplied video recording.

The video pre-processing algorithm is also capable of detecting if a certain camera is blocked (e.g. covered with a piece of fabric). Such information is forwarded to the Video Module that triggers an alarm.

4.3 Fuzzy Logic Module

The Fuzzy Logic Module is based on the following presumption: frequent behaviour is usual and therefore uninteresting while rare behaviour is interesting as it is highly possible that it is unwanted or at least unusual. Therefore the module counts the number of actions done by the object under surveillance and reasons about oddity of the observed behaviour based on the counters. If it detects a high number of odd events (i.e., events that rarely took place in the past) in a short period of time, it triggers an alarm.

The knowledge of the module is stored in two four-dimensional arrays of counters for each object under surveillance (implemented as red-black trees [2]). Events are split into two categories, hence the two arrays: events caused by movement and stationary events. A moving event is characterised by its location, direction, and the speed of movement. A stationary event, on the other hand, is characterised by location, duration and posture (lying, sitting, or standing). When an event is characterised, fuzzy discretization [17] is used, hence the name of the module. The location of an event in the floor plane is determined using the RTLS system and discretized in classes with size 50 cm, therefore the module considers the area under surveillance as a grid of 50 by 50 cm squares. The speed of movement is estimated by the Kalman filter. It is used to calculate the direction which is discretized in the 8 classes (N, NE, E, SE, S, SW, W, and NW). The scalar velocity is discretized in the following four classes: very slow, slow, normal, and fast. The posture is determined by a

primitive routine (see Section 3.3). The duration of an event is discretized in the following classes: 1, 2, 4, 8, 15, 30, seconds, minutes or hours.

The fuzzy discretization has four major advantages. The first is a smaller amount of memory needed to store the counters, as there is only one counter for a whole group of similar events. Note that the accuracy of the stored knowledge is not significantly decreased because the discrete classes are relatively small. The second advantage is the time complexity of counting the events that are similar to a given event, which is constant instead of being dependent on the number of events seen in the past. The third advantage is the linear interpolation implicitly introduced by fuzzy discretization, which enables a more accurate estimation of the rare events' frequencies. The fourth advantage is the low time complexity of updating the counters' values compared to the time complexity of adding a new counter with value 1 for each new event.

The oddity of the observed behaviour is calculated using a sliding window over which the average oddity of events is calculated. Averaging the oddity over time intervals prevents the false alarms that would be triggered if the oddity of single events was used whenever RTLS data noise or short sequences of uncommon events would occur. The oddity of a single event is calculated by comparing the frequency of events similar to the given event with the frequencies of the other events. For this purpose the supervisor sets the two relative frequencies f_{low} and f_{hi}. The threshold f_{low} determines the share of the rarest events that are treated as completely unusual and therefore they get assigned the maximum level of oddity. On the other hand, f_{hi} determines the share of the most frequent events that are treated as completely usual and therefore they get assigned 0 as the level of oddity. The oddity of an event whose frequency is between the thresholds f_{low} and f_{hi} is linearly decreasing with the increasing share of the events that are rarer than the given event (Figure 4).

The drawback of the described method is a relatively long learning period which is needed before the module starts to perform well. On the other hand, the module discards the outdated knowledge and emphasizes the new data, which enables adapting to the gradual changes in observed person's behaviour. The module is also highly responsive: it takes only about 3 seconds to detect the unusual behaviour. The module autonomously learns the model of usual behaviour which enables the detection of the unusual behaviour. It can detect events such as an unconscious person lying on the floor, running in a room where people usually do not run, a person sitting at the table at which he usually does not sit etc. The module also triggers an alarm when a long sequence of events happens for the first time. If such false alarm is triggered, the supervisor can mark it as false. Consequently, the module will increase the appropriate counters and will not raise an alarm for that kind of behaviour in the future.

When the Fuzzy Logic Module triggers an alarm, it also provides a graphical explanation for it. It draws a target-like graph in each square of the mesh dividing the observed area. The colour of a sector of the target represents the frequency of a given group of similar events. The concentric circles represent the speed of movement, e.g., a small radius represents a low speed. The triangles, on the other hand, represent the direction of movement. The location of a target on the mesh represents the location in the physical area. White colour depicts the lowest frequency, black colour depicts the highest frequency while the shades of grey depict the frequencies in between. The events that caused an alarm are highlighted with a scale ranging from green to red. For stationary events, tables are used instead of the targets. The row of the table represents the posture while the column represents the duration. A supervisor can read the graphical explanations quickly and effectively. The visualization is also used for the general analysis of the behaviour in the observed area.

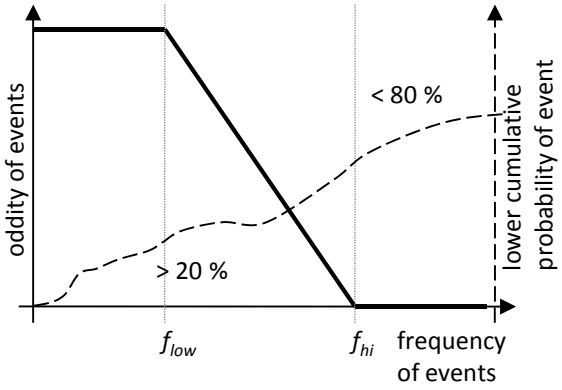


Figure 4: Calculating the oddity of events.

4.4 Macro and Statistic Modules

Macro and Statistic modules analyse persons' behaviour and trigger alarms if it significantly deviates from the usual behaviour. In order to do that, several statistics about the movement of each tagged person are collected, calculated, and averaged over various time periods. Afterwards, these statistics are compared to the previously stored statistics of the same person and the deviation factor is calculated. If it exceeds the predefined bound, the modules trigger an alarm.

The Statistic Module collects data over time periods from one minute to several hours regardless of person's location or context. On the other hand, the Macro Module collects data regarding behaviour in certain areas (e.g. room), i.e. the behaviour collection starts when a person enters the area and ends when he/she leaves it.

Both modules use behaviour attributes such as: the percentage of the time the person spent lying, sitting, standing, or walking during the observed time period, the average walking speed. Additionally, Macro module uses the following attributes: area id, day of the week, length of stay, entrance time, and exit time.

The behaviours are classified with the LOF algorithm [3] , a density-based kNN algorithm, which calculates the local outlier factor of the tested instance with respect to the learning instances. The LOF algorithm was chosen based on the study [14] . Bias towards false positives or false negatives can be adjusted by setting the alarm threshold.

The modules show a graphical explanation for each alarm in form of parallel coordinates plot. Each attribute is represented with one of the parallel vertical axes, while statistics about given time periods are represented by a zigzag line connecting values of each attribute from the leftmost to the rightmost one. Past behaviour is represented with green zigzag lines, while the zigzag line portending to the behaviour that triggered the alarm is collared red. The visualisation offers a quick and simple way of establishing the cause of alarm and often indicates more specific reason for it.

5 Verification

Due to the complexity of the PDR system and the diverse tasks that it performs it is difficult to verify its quality with a single test or to summarize it in a single number such as true positive rate. Therefore, validation was done on more subjective and qualitative level with several scenarios for each of the individual modules. Four demonstration videos of the PDR tests are available at http://www.youtube.com/user/ijsdis. A single test case or a scenario is a sequence of actions and events including a security risk that should be detected by the system. "A person enters a room without the permission" is an example of scenario. Each scenario has a complement pair: a similar sequence of actions which, on the contrary, must not trigger an alarm. "A person with permission enters the room" is the complement scenario for the above example. The scenarios and their complements were carefully defined in cooperation and under supervision of security experts from the Slovenian Ministry of Defence .

The Expert System Module was tested with two to three scenarios per expert rule template. Each scenario was performed ten times with various persons and objects. The module has perfect accuracy (no false positives and no false negatives) in cases when the RTLS noise was within the normal limits. When the noise was extremely large, the system occasionally triggered false alarms or overlooked security risks. However, in those cases even human experts were not able to tell if the observed behaviour should trigger an alarm or not based on the noisy RTLS measurements alone. Furthermore, the extreme RTLS noise occurred in less than 2 % of the scenario repetitions and the system made an error in less than 50 % of those cases.

The Video Module was tested using the following three scenarios: "a person enters the area under surveillance without the RTLS tag", "a robot is moving without authorised person's presence", and "a security camera is intentionally obscured". Scenarios were repeated ten times with different people as actors. The module detected the security risks in all of the scenario repetitions with movement and distinguished between a human and a robot perfectly. It failed to detect the obscured camera in one out of 10 repetitions. The module also did not trigger any false alarms.

The Fuzzy Logic Module was tested with several scenarios while the fuzzy knowledge was gathered over two weeks. The module successfully detected a person lying on the floor, sitting on colleagues chair for a while, running in a room, walking on a table, crawling under a table, squeezing behind a wardrobe, standing on the same spot for extended period of time, and similar unusual events. However, the experts' opinion was that some of the alarms should not have been triggered. Indeed we expect that in more extensive tests the modules supervised learning capabilities would prevent further repetitions of unnecessary alarms.

The test of the Macro and Statistic Modules included the simulation of a usual day at work condensed into one hour. The statistic time periods were 2 minutes long. Since the modules require a collection of persons' past behaviour, two usual days of work were recorded by a person constituting of two hours of past behaviour data. Afterwards, the following activities were performed 10 times by the same person and classified: performing a normal day of work, stealing a container with classified data, acting agitated as under the effect of drugs and running. The classification accuracy was 90 %. This was due to the low amount of past behaviour data. Therefore, the modules did not learn the usual behaviour of the test person but only a condensed (simulated) behaviour in a limited learning time. We expect that the classification accuracy would be even higher, if the learning time was extended and if the person would act as usual instead of simulating the condensed day of work.

Module	TP	TN	FP	FN	N
Expert Sys.	197	199	2	2	400
Video	30	30	0	0	60
Fuzzy Logic	47	42	8	3	100
Macro	9	10	1	0	20
Statistic	9	10	1	0	20
Total	292	291	12	5	600
Percentage (%)	**48.7**	**48.5**	**2**	**0.8**	

Table 1: Evaluation of PDR system.

The overall system performance was tested on a single scenario: "stealing a container with classified documents". In the test five persons tried to steal the container from a cabinet in a small room under surveillance. Each person tried to steal the container five times with and without a tag. All the attempts were successfully detected by the system that reported the alarm and provided an explanation for it.

The validation test data is summarized in Table 1. It gives the number of true positive (TP), true negative (TN), false positive (FP), and false negative alarms (FN), and total number (N) of scenario repetitions. Each row gives the results for one of the five modules. The bottom two rows give the total sum for each column and the relative percentage.

The system received the award for the best innovation among research groups in Slovenia for 2009 at the Fourth Slovenian Forum of Innovations.

6 Conclusion

This paper presents an intelligent surveillance system utilizing a real-time location system (RTLS), video cameras, and artificial intelligence methods. It is designed for surveillance of high security indoor environments and is focused on internal security threats. The data about movement of personnel and important equipment is gathered by RTLS and video cameras. After basic pre-processing with filters and primitive routines the data is sent to the five independent software modules. Each of them is specialized for detecting specific security risk. The Expert System Module detects suspicious situations that can be described by location of a person or other tagged objects in space and time. It detects many different scenarios with high accuracy. The Video Module automatically detects movement of persons and objects without tags, which is not allowed inside the surveillance area. Fuzzy Logic, Macro, and Statistics Modules automatically extract the usual movement patterns of personnel and equipment and detect deviations from the usual behaviour. Fuzzy Logic is focused on short-term anomalous behaviour such as entering an area for the first time, lying on the ground or walking on the table. Macro and Statistic Modules, on the other hand, are focused on mid- and long-term behaviour such as deviations in daily work routine.

The validation of the system shows that it is able to detect all the security scenarios it was designed for and that it does not raise too many false alarms even in more challenging situations. In addition, the system is customizable and can be used in a range of security applications such as confidential data archives and banks.

Acknowledgement

Research presented in this paper was financed by the Republic of Slovenia, Ministry of Defence. We would like to thank the colleges from the Machine Vision Laboratory, Faculty of Electrical Engineering, University of Ljubljana, Slovenia and Špica International, d.o.o. for fruitful cooperation on the project. Thanks also to Boštjan Kaluža,. Mitja Luštrek, and Bogdan Pogorelc for help regarding the RTLS and discussions, and Anže Rode for discussions about security systems, expert system rules templates and specification of scenarios.

References

[1] G. R. Arce. "Nonlinear Signal Processing: A Statistical Approach", Wiley: New Jersey, USA, 2005.

[2] R. Bayer. "Symmetric Binary B-Trees: Data Structures and Maintenance Algorithms", *Acta Informatica,* 1, pp. 290–306, 1972.

[3] M. M. Breunig, H. P. Kriegel, R. T. Ng, J. Sander. "LOF: Identifying densitybased local outliers," *Proceedings of the International Conference on Management of Data* –SIGMOD ’00, pp. 93–104, Dallas, Texas, 2000.

[4] J. Demšar, B. Zupan, G. Leban. *"Orange: From Experimental Machine Learning to Interactive Data Mining,"* White Paper (www.ailab.si/orange), Faculty of computer and information science, University of Ljubljana, Slovenia, 2004.

[5] M. Gams, T. Tušar. (2007), "Intelligent High-Security Access Control", Informatica, vol 31(4), pp. 469-477.

[6] R.E. Kalman. "A new approach to linear filtering and prediction problems". *Journal of Basic Engineering*, 82 (1), pp. 35–45, 1960.

[7] M. Kolbe, M. Gams. "Towards an intelligent biometric system for access control," *Proceedings of the 9th International Multiconference Information Society* - IS 2006, Ljubljana, Slovenia, 2006, pp. 118-122.

[8] B. Krausz, R. Herpers. 'Event detection for video surveillance using an expert system', *Proceedings of the 1st ACM Workshop on Analysis and Retrieval of Events/Actions and Workflows in Video Streams* - AREA 2008, Vancouver, Canada, pp. 49-56.

[9] M. Kristan, J. Perš, M. Perše, S. Kovačič. "Closed-world tracking of multiple interacting targets for indoor-sports applications", *Computer Vision and Image Understanding*, vol 113, 5, pp. 598-611, 2009.

[10] M. Perše, M. Kristan, S. Kovačič, G. Vučković, J. Perš. "A trajectory-based analysis of coordinated team activity in a basketball game", *Computer Vision and Image Understanding*, vol 113, 5, pp. 612-621, 2009.

[11] R. Piltaver, G. Matjaž. "Expert system as a part of intelligent surveillance system", *Proceedings of the 18th International Electrotechnical and Computer Science Conference* - ERK 2009, vol. B, pp. 191–194, 2009.

[12] R. Piltaver. "Strojno učenje pri načrtovanju algoritmov za razpoznavanje tipov gibanja", *Proceedings of the 11th International Multiconference Information Society* - IS 2008, str. 13–17, 2008.

[13] V. Schwarz, A. Huber, M. Tüchler. "Accuracy of a Commercial UWB 3D Location Tracking System and its Impact on LT Application Scenarios," *Proceedings of the IEEE International Conference on Ultra-Wideband*, Zürich, Switzerland, 2005.

[14] T. Tušar, M. Gams. "Odkrivanje izjem na primeru inteligentnega sistema za kontrolo pristopa," *Proceedings of the 9th International Multiconference Information Society* - IS 2006, Ljubljana, Slovenia, 2006, pp. 136-139.

[15] Ubisense: awailable at: http://www.ubisense.net/

[16] H. Witten, E. Frank. Data Mining. "Practical Machine Learning Tools and Techniques" (2nd edition), Morgan Kaufmann, 2005.

[17] L. A. Zadeh. "Fuzzy sets", *Information and Control* 8 (3), pp. 338–353, 1965.

[18] http://www.pervcomconsulting.com/secure.html

[19] http://www.visonictech.com/Active-RFID-RTLS-Tracking-and-Mangement-Software-Eiris.html

[20] http://www.aeroscout.com/content/healthcare

[21] http://www.telargo.com/solutions/track_trace.asp

Query Preserving Relational Database Watermarking

S.A. Shah, Sun Xingming and Hamadou Ali
Network and Information Security Lab Hunan University, Changsha, Hunan, China
E-mail: saeed.arif@gmail.com; sunnudt@126.com: alihamadou@yahoo.fr

Majid Abdul
Pervasive Computing Lab Zhejiang University Hangzhou, Zhejiang P.R. China
E-mail: majedabbasi@yahoo.com

Keywords: right protection, relational data, watermarking

Received: March 25, 2010

In order to preserve the query results after watermarking relational data, it is necessary to keep the semantic value of data intact during watermark embedding process. A query preserving relational database-watermarking scheme is proposed in this paper. For watermark embedding, we use alphanumeric data as new embedding channel. The scheme retains the semantic meaning of data after embedding, which makes this a distortion free and query preserving technique. Watermark Embedding is done by adjusting the case of securely selected text data. Our proposed method provides better relational database watermarking solution for the database, which either has no numeric attribute, or has data with zero resilience to data alteration. To make this scheme more secure tuples and attributes selection is done using a secret key known only to the owner. Later the same key is used in detection process. Moreover, there is no need of original data for watermark detection so it is a fully blind scheme. The method proved (through experiments) to be robust against various kinds of attacks. An SQL Server database implementation has shown that our algorithms can be used successfully in real world applications.

Povzetek: Predstavljeno je obvarovanje relacij pri povpraševanju v relacijskih bazah.

1 Introduction

Easy modification and reproduction of digital data (software, images, video, audio, and text) without leaving any trace of manipulation makes it very easy victim of piracy. Number of watermarking based solutions proposed so far for copyright protection of relational data. Watermark is a secret code embedded in digital contents. This watermark can be extracted/detected from the watermarked contents and can be used to establish the ownership of data. Watermarking fails to prevent illegal copying but it can be an effective tool for establishing original ownership of pirated data. This discourages piracy and enables owners to prosecute copyright violators.

Growing use of outsourced relational data, especially availability of relational data over the internet, demanded an effective mechanism for copyright protection so that owner of the data can identify pirated copies of their data. Watermarking has proved to be an effective tool for multimedia data so researchers explored this technology for relational data also. Agrawal et al [1] pioneered research on relational database watermarking in 2002. Different schemes [1, 4, 3, 9, and 13] have been proposed after that. Most of the previous work in this area use numeric data as embedding domain for watermark insertion. All these schemes are based on the assumption that there are some data, which can tolerate small changes, without affecting its usability. Some of them use direct LSB domain [1] while other manipulates statistics of the data for watermark embedding [4, 13].

There are few schemes proposed for categorical data watermark but these schemes also introduce significant change to data [2, 15]. A large number of techniques available for multimedia watermarking [3, 5, 7, 10] which proved to be effective but these cannot be applied directly to database. For multimedia, there is a lot of room for embedding any extra information, as there is a large amount of redundant bits. One can play with these bits as long as these manipulations are imperceptible. For multimedia, the most important requirement is to avoid visual distortion whereas for relational data, preservation of semantic value of data is essential. Sometimes even a change of a single bit will change the meaning of data and thus affect query results. For example change of single bit in date like name, address, age, account number etc, will change the value and in turn the query result. Another important challenge that still needs attention is; what if there is no numeric data or, there is data which is not resilient enough for watermark embedding? All these factors lead to the need of scheme, which not only retain the semantic value of data but also preserve query results after watermark embedding. In this paper we are going to propose such a scheme which

works for non-numeric data and also preserve query results by introducing almost zero distortion to semantic value of data while watermark embedding.

The paper is organized as follows. In section 2 related work is discussed. Section 3 discusses our scheme in detail. Section 4 analyzes main features of our scheme. In section 5 different attacks are discussed. Experimental setup and results are also outlined in section 5.A multi-bit watermarking scheme; an extended version of the proposed solution is described in Section 6. Section 7 concludes.

2 Related Work

Work on database watermarking started in 2002 when Agrawal and Kiernan presented a robust watermarking scheme for databases [1]. The scheme focused on watermarking relational data with numeric attributes. It is assumed that these numeric attributes can tolerates small amount of modification. Using a secret key and secure hash algorithm, first tuple and then attribute within that tuple are randomly selected for watermark embedding. Finally, selected bits of that attribute are modified in order to embed watermark bits. Robustness of the scheme is shown through experiments and theoretical analysis against different kind of attacks including such as rounding attack, subset attack, and additive attack etc.

Another significant contribution in this area is by Radu Sion et al. [2]. Sion presented different schemes for numeric data and categorical data [4]. For the first time Sion proposed a subset based watermarking method for numeric data. According to this method first all the numeric data is partitioned into subsets using some secret key and then a single bit is embedded within each subset by playing with its statistics. The scheme claimed to be robust against variety of attacks including subset attack, data resorting and transformations attacks. However, it does not seem effective for database which need frequent update, as it requires re watermarking of all the data.

In [14] Yingjiu Li et al proposed a fragile watermarking scheme for relational data authentication. This is a group based technique. Watermark calculated from message digest of the group, which is then embedded in the same group. Since the message digest is fragile even for single bit change, it can be used for authentication of relational data.

In [13] M. Shehab et al presented optimization based watermarking for numerical data. The relational data watermarking is first formulated as constrained optimization problem then solution to this problem is sought either using Genetic algorithms or Pattern search. Here again data is partitioned into subsets and distribution of each partitions is modified to embed a single bit, but embedding is done by solving the optimization problem either for maximization or minimization. This technique is state of the art for numerical data as it introduces minimum distortion to data and is more robust against different attack than earlier schemes.

The above mentioned schemes work only for numeric data. In [2] a robust watermarking proposed by Sion for categorical data copyright protection. In the scheme first tuples are securely selected using secret key then values of categorical attributes of selected tuples are changed to some other values from available pool valid values based on the watermark to be embedded (e.g. change city from New York to Washington). The values are changed by satisfying some constraints so that this change does not affect data usability.

David Gross proposed a scheme for query preserving relational data watermarking in [15]. Scheme claimed to be robust against local queries. For watermark embedding first local queries are identified and then selected data values are modified while preserving these queries. However, the changes made to the data values are significant to most of the applications, which limits its scope.

Notation	Description
r	Row or record of a relation
K	Secret key known only to owner
n	Number of tuples in the relation
v	Number of attributes in the tuple
t	No of tuples to be marked
$1/m$	Fraction of tuples to be watermarked
s	Size of watermark
K_w	Selected watermark key
E_b	Existing bit pattern
W	Embedded Watermark
G	Pseudo random sequence generator

Table 1: Notations and parameters.

3 Our Scheme

Most of the above mentioned schemes [1, 2, 4, 9] are based on the manipulation of numeric data (which must have some margin of error), thus have limited domain. In addition, the schemes discussed above including those for categorical data, introduce distortion in the contents by changing meaning of attribute values which is often not desirable.

In our scheme we introduce a new embedding channel by embedding watermark in non numeric data or more precisely the alphabetic data attributes. Since the database, queries are case insensitive so it will not affect the semantic meaning of data if the case is changed from small to capital or vice versa. We are going to exploit this inherent property present in such kind of data attributes. The proposed method is applicable to all the languages with upper/lower character cases. This work is actually an extension of our previous work for copyright protection of relational data [16]. We now present our technique for watermarking relational database. This technique marks only alphabetic attributes without introducing any change in their semantic meaning. Not all attributes need to be watermarked. Data owner will decide which attributes are more suitable for watermarking. Let R be the database relation with schema R (P, A_0…A *v-1*) where P is the primary key attribute. Table 1 shows the important parameters used in

our algorithms. For simplicity assume that all the attributes are candidate for watermarking. *m* is used to determine the number of tuple to be watermarked. If *t* denotes the number of tuples to be marked then

$$t \approx n / m$$

r.Ai is used to denote the value of attribute Ai. In this technique, we are using one way hash function H for hash value calculation. There are number of hash function like MD4, MD5 SHA1 SHA256 etc.

A Message Authentication Code (MAC) is computed using a one way hash function that depends on a key[11]. If K denotes the key then H will randomize the input primary key ***"r.P"*** of relation R when H is seeded with **K** known only to owner. The Following MAC is used in our Scheme

$$MAC=H(K, r.P)$$

Where we are using SHA-1 as one way hash function H.

3.1 Watermark embedding

Watermark embedding algorithm is given in table 2. Given the relation *R* (P, A1, A2……A_{v-1}) with primary key P. Lines 1 through 6 determine tuples and attribute to be marked respectively, using primary key Hash value. Selection of both depends on secret key **K** known to the owner, so only the owner can identify which tuple and which attribute of that tuple is to be marked. An attacker has to guess the tuple as well as attribute within the tuple to destroy the watermark. In line 8 existing bit sequence Eb is extracted by inspecting the text case of selected data. Bit 1 or 0 is extracted following same rules outlined in detection algorithm. Lines 9-11 generate L number of candidate watermark sequences Cj, each of which is then compared to existing bit pattern Eb. Keys used to generate these candidate watermarks may be obtained by seeding G with secret key K and index j of the watermark.

// Secret keys **K, Kw** and parameters ***m, v*** are private to the owner.
1. foreach tuple $r \in$ do step 2 to 5
2. Calculate PrimaryKeyHash pHash=H (K, r.P)
3. Select tuple with (pHash mod *m* ==0)
4. Select attribute with index: i= pHash mod *v*
5. selected attribute array: SelectedValue[i]=r.Ai
6. Sort Selected tuples using pHash
7. watermark size *s*= length of SelectedValue[ ]
8. Extract Existing Bit (E_b) pattern From selected data values
9. Generate random candidate watermarks Cj each of length *s* using G and keys K_j
Cj= G(K_j) where 1<=j<=L
10. a) Select watermark W= Cj with minimum hamming distance from E_b
b) K_w = Key of selected wm sequence
11. Call embedwm(SelectedValue[ ], W)//Embed Watermark in Selected Data

Table 2: Watermark embedding algorithm.

Finally the bit sequence having minimum hamming to Eb is selected for embedding as watermark W and its key isrecorded as Kw which will be used later for watermark extraction. *embedwm* actually embeds the watermark depending on the corresponding watermark bit. It adjusts the case of selected attribute value according to the conditions laid down in Algorithm 1. Case is adjusted so as to follow most common practice e.g. if watermark bit is 1 then case is converted to title case and for 0 to sentence case.

Sometimes database contain null values in that case mark is not applied. In addition, when there is text such as abbreviation, where a standard is there, no change is applied.

//Given *R*, K, Kw and parameters m***, v***
1. foreach tuple r $r \in$ do steps 2 to 4
2. PrimaryKeyHash pHash=H(K, r.P)
3. if (pHash mod *m* ==0) then Select This tuple
4. attribute_index i= pHash mod *v* //Select Attribute Ai
5. Sort Selected tuples using pHash
//Extract Watermark W_e
6. Repeat 7 to 10 for selected tuples
7. Case-I: When r.Ai is single word
8. If (r.Ai has title case) then
W_e [i]=1
else If (r.Ai has all caps) then
W_e [i]=0
9. Case-II: When r.Ai is multi word
10. If (whole text of r.Ai has Title case) then
W_e [i]=1
else W_e [i] =0
// Verify Watermark
11. Generate W using key K_w
12. result_vector=W XOR W_e

Table 3: Watermark detection algorithm.

3.2 Watermark detection algorithm

Let Alice be the owner of the database and Mallory another person with pirated copy of Alice's Data. We assume that the primary Key is intact because dropping it may cause loss of important data. The algorithm for watermark detection is given in the table 3. In line 3 the tuple is selected where watermark is supposed to be embedded. Line 4 determines attribute marked. Both of these are selected using same secret key K used during embedding. In lines 6 to 10, watermark bits are extracted using the predefined conditions.

When the attribute value consists of a single word then extracted watermark bit is 1, if it has Title case and 0 otherwise. For attribute having multiple words, the watermark bit is 1 if whole text has title case and 0 for sentence case. Watermark verification is done in lines 11 & 12, where, first the original watermark is generated using same secret key and then compared with the extracted watermark.

4 Discussion

In this section, we discuss some important features of our scheme related to security, detection, query preservation and case alterations.

4.1 Security

Security of our schemes can be defined in terms of difficulty for a malicious attacker to recover, locate or even guess originally embedded watermark. For watermark generation a secure pseudorandom sequence generator G is used. It is computationally infeasible to predict the next number in the sequence [6]. Statistically, the numbers generated by G appear to be a realized sequence of independent and identically distributed random variables, in the sense that the numbers pass standard statistical tests for these properties [8]. A seed value is used to generate the random number.

Same sequence will be generated every time with repeated execution of G with given seed value [6]. We used the secret key K as seed and only the owner knows it. Selection of tuples and attributes is purely key based. Moreover, use of a secure one-way hash function makes this scheme more secure.

4.2 Blind detection

There are two types of watermarking methods, one which require original data for the detection of embedded watermark called non-blind and other which don't called blind [10]. Since there is no need of original database to recover/decode embedded watermark, so we can claim that our scheme is Blind in nature.

The watermark W_e is extracted from watermarked relation, which is then verified. It is difficult to keep original version of distributed copy of database because it requires frequent updates, so a blind technique is very helpful.

4.3 Query preserving watermark

Watermark embedding is done by adjusting the case of selected data according to predefined rules, which does not change the meaning of data so queries result will not be affected even after embedding.

4.4 Reduced number of case alterations

For watermark, a number of random sequences are generated and one of them is selected for embedding. This selected sequence has minimum hamming distance to existing bit sequence, which is extracted from current text case of selected attribute values before embedding. Applying hamming distance for final watermark selection reduces number of text case alteration leading to low text case distortion. This is one of the important contributions.

5 Experiments and Attacks Analysis

In this section, we will discuss the survival of embedded watermark against common database attacks. Watermark that survives when its host data is exposed to attacks is called robust watermark. The watermarked data can be attacked in various ways through malicious attacks and benign updates. The most common attacks are:

i. Tuple deletion
ii. Modifying attributes values
iii. Case alteration

The first two attacks may affect the usability of data, so, for an attacker these kinds of attack are often less desirable. The third one is actually a legal attack, which does not affect the usability of data so is most important to study it. We analysed our scheme against these attacks through experiments, and results show that it is robust against the above attacks; means there is very high probability of correctly decoding the embedded watermark even after these attacks.

For experimental purpose, we used SQL server database of more than half million records on Windows Xp platform. The value of "m" is kept 10 and number of watermarkable attributes "ν" is 3 in our sample database. In the following, we present experiment involving different attacks (Data loss, Data Alterations, Change in Case). Experiments were performed repeatedly and their results are averaged over multiple runs.

5.1 Tuple deletion attack

In this experiment, randomly selected tuples are deleted and after deletion of every few tuples, the watermark is extracted and compared with originally embedded watermark. The experiment performed many times and average behaviour is plotted in figure 1. It shows that even after deleting 35-40% of tuples, distortion in decoded watermark is up to 12%. In our experiments, we used binary watermark so in this case 88% bits of decoded watermark matched with the actual embedded watermark so distortion (damage/loss in watermark) is only 12%.

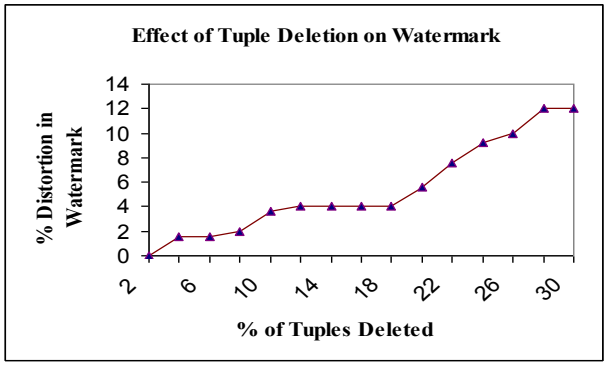


Figure 1: Tuple deletion attack.

5.2 Data modification attack

Mallory's (The attacker) priority would be to destroy watermark, while preserving the data. Given no

knowledge of secret key or the original data, the attacker may try to make random modifications to watermarked data values, thus hoping to destroy watermark at some point. In this experiment, we analyse the sensitivity of our scheme to random updates of watermarked data. The demonstrated behaviour is shown in the figure 2. The results show that only 6% distortion in watermark is observed if 35-40% data values are randomly modified. Hence, it is more robust against such kind of attacks as compared to other attacks.

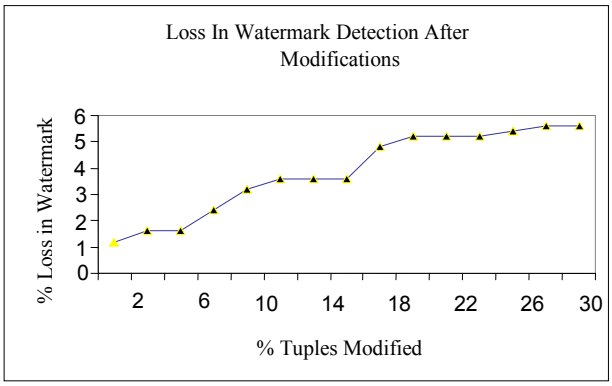


Figure 2: Data modification attack

5.3 Tuple sorting attack

During embedding, we sort the selected tuples before embedding watermark, using primary key hash value, and the same process is repeated during detection, so any kind of sorting attack will not harm the detection of watermark.

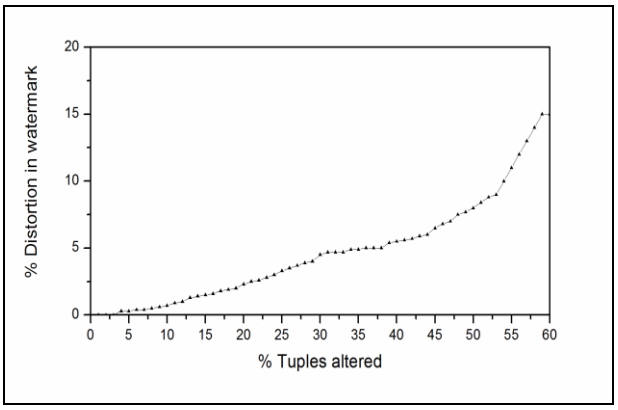


Figure 3: Case alteration attack.

5.4 Case alteration attacks

Since we are playing with the case of attributes values so this kind of attack is only specific to our scheme and it is a legal attack so robustness against it must be checked. In this experiment, we randomly (and repeatedly) change the case of attributes values, then extract watermark, and compare it with embedded one. The results are averaged our various runs. As shown in the figure 3. It is observed that our scheme is robust against this kind of attack. After changing the case of up to 60% tuples, the watermark distortion is less than 20%.

It is evident from the results of above experiments, that we can correctly decode the embedded watermark with very high ratio (up to 80%) even after different attacks. Hence we can claim that our scheme is robust against common database attacks.

6 Multi-bit watermarking

In this section, an extension to multi-bit watermarking of the proposed scheme is presented. For this purpose, the watermark embedding and detection algorithms are modified. For embedding first all the tuples are securely divided into partitions. A single bit embedded into each partition, which requires that the number of partitions should be much greater than the watermark size so that a single bit can be embedded multiple times.

The simplified versions of embedding and detection algorithms are given in the table 4 and 5 respectively. For sake of simplicity only overview of embedding and detection process is given.

E1. Secret Grouping:

All tuples are securely divided into "g" number of groups. Grouping is done as proposed by Shehab in [13]

E2. Tuple Sorting:

All the tuples are sorted based on secure hash value of each tuple's primary key

E3. Secure Tuple and Attribute Selection:

Within each group, first a tuple then an attribute is randomly selected for marking (same as in algorithm 1)

E4. Watermark Embedding:

The case of selected attribute data in a specific group is adjusted to represent the embedded watermark bit

Table 4: Multi-bit embedding algorithm.

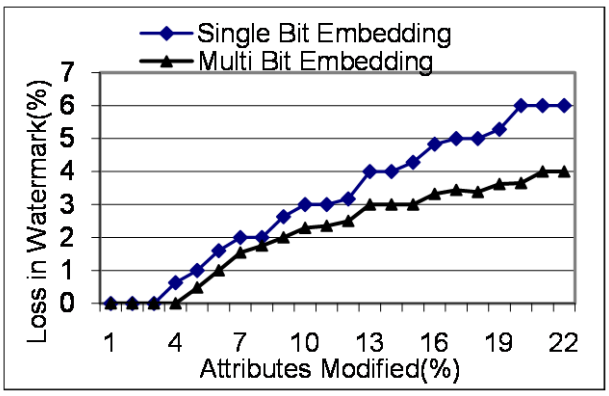


Figure 4: Robustness against value modification attack.

D1. Grouping:

All the tuples are securely divided into g groups using the same secret key K and number of group's g

D2. Extract Watermark W_e:

Sort tuples within each group

(a). Determine marked tuples and attributes from groups (same as in encoding)

(b). Extract watermark bit from the Case of selected data

(c). Apply majority voting method for final watermark extraction

D3. Watermark verification:
(a) Bit matching of W_e with actual watermark W (Generated using same key K)
(b) If match_count/total_count > τ then watermark detected

Table 5: Multi-bit detection algorithm

Experiments show that robustness of the scheme can be improved significantly by using multi-bit embedding. Figure 4 shows the robustness of the multi-bit scheme against data modification attack.
In our experiments we used value of threshold $\tau = 0.8$. We suggest its value within $0.6 \leq \tau \leq 0.8$. If an image is embedded as watermark then τ can be set to even lower value.

7 Conclusions and Future Work

In this paper, we introduced a new scheme for watermarking relational database for owner verification and copyright protection. A solution is proposed by:

i. Discovering a new embedding channel for watermarking.
ii. Building an algorithm for watermarking such that it preserves data integrity by introducing zero distortion to its semantic meaning.
iii. Improving robustness by multi-bit embedding

We thus provided an efficient watermarking technique for copyrights protection of relational data. Through experiments, we proved that our scheme is robust against common database attack as well attacks specific to our scheme. In future, we intend to analyze and improve this scheme against other attacks such as subset and partitioning attacks. Another research direction may be to investigate a method for data authentication using fragile watermarks.

Acknowledgment

This work is supported by the National Basic Research Program of China (973 Program) under grant No. 2006CB303000, the National High Technology Research and Development Program of China (863 Program) under grant 2007AA010404, NSFC Key Project grant No. 60736016, NSFC grants No. 60702065, 60873198, 60973113 and 60973128, Science and Technology Program of Hunan Province grants No.2008FJ4221, Special for National Basic Research Program of China (973 Program) grants No. 2009CB326202 and Higher Education Commission Pakistan through University of AJ&K FDP 2008.

References

[1] R. Agrawal and J. Kiernan. Watermark relational databases. In *Proc. of the 28th International . Conference. On Very Large Data Bases*, 2002.
[2] R.Sion. Proving ownership over categorical data. In *Proceedings of ICDE* 2004.
[3] I. J. Cox, M. Miller, and J. Bloom. Watermarking applications and properties. In *Proc. International Conference on Information Technology: Coding and Computing*, 2000.
[4] R.Sion, M. Atallah, and S. Prabhakar. Rights protection for relational data. In *Proceedings of ACM SIGMOD 2003*, 2003.
[5] C. Rey and J. Dugelay, A Survey of Watermarking Algorithms for Image Authentication In JASP, No.6, pp. 613-621, 2002.
[6] Schneider B. Applied cryptography", 2nd ed. (1996) Wiley, New York
[7] I. J. Cox, J. Kilian, T. Leighton and T. Shamoon, Secure Spread Spectrum Watermarking for Multimedia, *IEEE Trans. on Image Processing, 6, 12, 1673-1687,* (1997).
[8] Knuth D. Semi numerical algorithms. In: The art of computer programming, vol 2.Addison-Wesley, Reading, MA, 1981
[9] Y. Li, V. Swarup, and S. Jajodia, A Robust Watermarking scheme for relational data. In *Proc. The 13th workshop on information technology and engineering*, pages 195–200, December 2003.
[10] Ingemar Cox, Matthew Miller, Jeffrey Bloom. Digital Watermarking Morgan Kaufmann .2002
[11] N. Ferguson and B. Schneider, Practical Cryptography. Wiley & Sons, 2003.
[12] J. Brassil, L. O'Gorman Watermarking Document Images with Bounding Box Expansion, in Proc. of 1st Int'l Workshop on Information Hiding, Newton Institute, Cambridge UK, May 1996, pp. 227-235.
[13] Mohammad Shehab, Elisa Bertino, Arif Ghafoor Watermarking Relational Data using Optimization Based Techniques *IEEE Transactions on Knowledge and Data Engineering Volume 20 , Issue 1 (January 2008) Pages 116-129*
[14] Y. Li, H. Guo, S. Jajodia, Tamper detection and localization for categorical data using fragile Watermarks in: *4th ACM Workshop on Digital Rights Management, CCS04, October 2004.*
[15] D. Gross-Amblard, Query-preserving watermarking of relational databases and xml documents, in: *Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database, June 9–12, 2003, pp. 191–201.*
[16] S. A. Shah, S. A. M. Gilani, I. A. Awan: Owner Verification and Copyright Protection of Relational Data. in: *Proc. Of IMEC Jun 2006 252-257 Hong Kong.*

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 900 staff, has 700 researchers, about 250 of whom are postgraduates, around 500 of whom have doctorates (Ph.D.), and around 200 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

From the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

Part of the Institute was reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of the Economy, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit a manuscript at: http://www.informatica.si/Editors/PaperUpload.asp. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than seventeen years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

http://www.informatica.si/

Referees from 2008 on:

Ajith Abraham, Siby Abraham, Renato Accornero, Raheel Ahmad, Cutting Alfredo, Hameed Al-Qaheri, Gonzalo Alvarez, Wolfram Amme, Nicolas Anciaux, Rajan Arora, Costin Badica, Zoltán Balogh, Andrea Baruzzo, Borut Batagelj, Norman Beaulieu, Paolo Bellavista, Steven Bishop, Marko Bohanec, Zbigniew Bonikowski, Borko Bosković, Marco Botta, Pavel Brazdil, Johan Brichau, Andrej Brodnik, Ivan Bruha, Maurice Bruynooghe, Wray Buntine, Dumitru Dan Burdescu, Yunlong Cai, Juan Carlos Cano, Tianyu Cao, Norman Carver, Marc Cavazza, Jianwen Chen, L.M. Cheng, Chou Cheng-Fu, Girija Chetty, G. Chiola, Yu-Chiun Chiou, Ivan Chorbev, Shauvik Roy Choudhary, Sherman S.M. Chow, Lawrence Chung, Mojca Ciglarič, Jean-Noël Colin, Vittorio Cortellessa, Jinsong Cui, Alfredo Cuzzocrea, Darko Čerepnalkoski, Gunetti Daniele, Grégoire Danoy, Manoranjan Dash, Paul Debevec, Fathi Debili, Carl James Debono, Joze Dedic, Abdelkader Dekdouk, Bart Demoen, Sareewan Dendamrongvit, Tingquan Deng, Anna Derezinska, Gaël Dias, Ivica Dimitrovski, Jana Dittmann, Simon Dobrišek, Quansheng Dou, Jeroen Doumen, Erik Dovgan, Branko Dragovich, Dejan Drajic, Jozo Dujmovic, Umut Riza ErtÃijrk, CHEN Fei, Ling Feng, YiXiong Feng, Bogdan Filipič, Iztok Fister, Andres Flores, Vladimir Fomichov, Stefano Forli, Massimo Franceschet, Alberto Freitas, Jessica Fridrich, Scott Friedman, Chong Fu, Gabriel Fung, David Galindo, Andrea Gambarara, Matjaž Gams, Maria Ganzha, Juan Garbajosa, Rosella Gennari, David S. Goodsell, Jaydeep Gore, Miha Grčar, Daniel Grosse, Zhi-Hong Guan, Donatella Gubiani, Bidyut Gupta, Marjan Gusev, Zhu Haiping, Kathryn Hempstalk, Gareth Howells, Juha Hyvärinen, Dino Ienco, Natarajan Jaisankar, Domagoj Jakobovic, Imad Jawhar, Yue Jia, Ivan Jureta, Dani Juričić, Zdravko Kačič, Slobodan Kalajdziski, Yannis Kalantidis, Boštjan Kaluža, Dimitris Kanellopoulos, Rishi Kapoor, Andreas Kassler, Daniel S. Katz, Samee U. Khan, Mustafa Khattak, Elham Sahebkar Khorasani, Ivan Kitanovski, Tomaž Klobučar, Ján Kollár, Peter Korošec, Valery Korzhik, Agnes Koschmider, Jure Kovač, Andrej Krajnc, Miroslav Kubat, Matjaz Kukar, Anthony Kulis, Chi-Sung Laih, Niels Landwehr, Andreas Lang, Mohamed Layouni, Gregor Leban, Alex Lee, Yung-Chuan Lee, John Leggett, Aleš Leonardis, Guohui Li, Guo-Zheng Li, Jen Li, Xiang Li, Xue Li, Yinsheng Li, Yuanping Li, Shiguo Lian, Lejian Liao, Ja-Chen Lin, Huan Liu, Jun Liu, Xin Liu, Suzana Loskovska, Zhiguo Lu, Hongen Lu, Mitja Luštrek, Inga V. Lyustig, Luiza de Macedo, Matt Mahoney, Domen Marinčič, Dirk Marwede, Maja Matijasevic, Andrew C. McPherson, Andrew McPherson, Zuqiang Meng, France Mihelič, Nasro Min-Allah, Vojislav Misic, Vojislav Mišić, Mihai L. Mocanu, Angelo Montanari, Jesper Mosegaard, Martin Možina, Marta Mrak, Yi Mu, Josef Mula, Phivos Mylonas, Marco Di Natale, Pavol Navrat, Nadia Nedjah, R. Nejabati, Wilfred Ng, Zhicheng Ni, Fred Niederman, Omar Nouali, Franc Novak, Petteri Nurmi, Denis Obrul, Barbara Oliboni, Matjaž Pančur, Wei Pang, Gregor Papa, Marcin Paprzycki, Marek Paralič, Byung-Kwon Park, Torben Bach Pedersen, Gert Schmeltz Pedersen, Zhiyong Peng, Ruggero G. Pensa, Dana Petcu, Marko Petkovšek, Rok Piltaver, Vid Podpečan, Macario Polo, Victor Pomponiu, Elvira Popescu, Božidar Potočnik, S. R. M. Prasanna, Kresimir Pripuzic, Gabriele Puppis, HaiFeng Qian, Lin Qiao, Jean-Jacques Quisquater, Vladislav Rajkovič, Dejan Rakovic, Jean Ramaekers, Jan Ramon, Robert Ravnik, Wilfried Reimche, Blagoj Ristevski, Juan Antonio Rodriguez-Aguilar, Pankaj Rohatgi, Wilhelm Rossak, Eng. Sattar Sadkhan, Sattar B. Sadkhan, Khalid Saeed, Motoshi Saeki, Evangelos Sakkopoulos, M. H. Samadzadeh, MariaLuisa Sapino, Piervito Scaglioso, Walter Schempp, Barabara Koroušić Seljak, Mehrdad Senobari, Subramaniam Shamala, Zhongzhi Shi, LIAN Shiguo, Heung-Yeung Shum, Tian Song, Andrea Soppera, Alessandro Sorniotti, Liana Stanescu, Martin Steinebach, Damjan Strnad, Xinghua Sun, Marko Robnik Šikonja, Jurij Šilc, Igor Škrjanc, Hotaka Takizawa, Carolyn Talcott, Camillo J. Taylor, Drago Torkar, Christos Tranoris, Denis Trček, Katarina Trojacanec, Mike Tschierschke, Filip De Turck, Aleš Ude, Wim Vanhoof, Alessia Visconti, Vuk Vojisavljevic, Petar Vračar, Valentino Vranić, Chih-Hung Wang, Huaqing Wang, Hao Wang, Hui Wang, YunHong Wang, Anita Wasilewska, Sigrid Wenzel, Woldemar Wolynski, Jennifer Wong, Allan Wong, Stefan Wrobel, Konrad Wrona, Bin Wu, Xindong Wu, Li Xiang, Yan Xiang, Di Xiao, Fei Xie, Yuandong Yang, Chen Yong-Sheng, Jane Jia You, Ge Yu, Borut Zalik, Aleš Zamuda, Mansour Zand, Zheng Zhao, Dong Zheng, Jinhua Zheng, Albrecht Zimmermann, Blaž Zupan, Meng Zuqiang

Informatica

An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 2011 (Volume 35) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printing: Dikplast Kregar Ivan s.p., Kotna ulica 5, 3000 Celje.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Igor Grabec)
ACM Slovenia (Dunja Mladenič)

Informatica is surveyed by: ACM Digital Library, Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

The issuing of the Informatica journal is financially supported by the Ministry of Higher Education, Science and Technology, Trg OF 13, 1000 Ljubljana, Slovenia.

Volume 35 Number 3 September 2011 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Regression Test Selection Techniques: A Survey	S. Biswas, R. Mall, M. Satpathy, S. Sukumaran	**289**
Distributed Multi-ant Algorithm for Capacity Vehicle Route Problem	J. Li, Y. Chai, C. Yuan	**323**
Mutant Hierarchies Support Selective Mutation	K. Kapoor	**331**
Improved ID-based Ring Signature Scheme with Constant-size Signatures	H. Li, X. Li, M. He, S. Zeng	**343**
Content-sensitive Approach for Video Browsing and Retrieval in the Context of Video Delivery: VBaR Framework	P.Y. Lau, S. Park	**351**
Multivariable Generalized Predictive Control Using An Improved Particle Swarm Optimization Algorithm	M. Sedraoui, S. Abdelmalek, S. Gherbi	**363**
Mutual Information and Cross Entropy Framework to Determine Relevant Gene Subset for Cancer Classification	R. Bala, R.K. Agrawal	**375**
An Intelligent Indoor Surveillance System	R. Piltaver, E. Dovgan, M. Gams	**383**
Query Preserving Relational Database Watermarking	S.A. Shah, S. Xingming, H. Ali, M. Abdul	**391**

Informatica **35** (2011) Number 3, pp. 289–397