Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

Approach for the development of mobile applications based on migrant objects

Aissa ElMahdi Bourahla, Mustapha Bourahla

Abstract

Software engineering principles are very required for the development of mobile applications, which are necessary for many life applications. In this paper, we present a development process based on transformation of UML models, which describe mobile applications based on migrant objects to Mobile Maude language that extends the Maude language and implements the Rewriting Logic (RL). The generated rewriting theories will be executed for simulation using located configurations, which are produced from UML object diagrams. State transition models may be built during simulation to describe behaviours of mobile applications based on migrant objects, which will be verified against LTL properties with the technique of model-checking. The verified model of mobile application based on migrant objects can be used to generate Android application to be deployed on mobile devices.

Keywords: Mobile Application, Migrant Objects, Rewriting Logic, Mobile Maude, Maude LTL Model Checker.

1 Introduction

The mobile application development has exponential growth since the iPhone AppStore opened in 2008. There are many documents for sets of best practices to guide developers of mobile applications, like those proposed to Android and Apple iPhone developers, but rarely use formal development techniques. Despite the development of large number of mobile applications, there is still not much formal research

around their engineering processes. There are powerful development tools and frameworks (for example, Android Studio, Eclipse and Windows Phone) offering programming environments for the major mobile platforms to simplify the task of implementing mobile applications. These tools are focusing on the individual developer who is trying to create a mobile application as quickly as possible.

A formal technique will help to develop mobile apps independently of the mobile platform as Android, iOS, etc. when the devices with different software versions or screen sizes might have issues that aren't found elsewhere, which can also help to handle testing. The formal specification and verification process is still a crucial part of development and it gives an added insight into the inner workings of mobile app, where alternative ways can be potentially found to achieve our goals. These alternatives can save work time and cost. Thus, it is not sufficient to test the app once on mobile phone and assume that it's working correctly. A quality assurance process can find problems with our app before it goes to market. It's much better to identify these using formal development; otherwise, users will find them in real life.

Techniques for developing mobile applications are similar to software engineering for other embedded applications with additional requirements. The mobile applications have potential interaction with other applications, sensor handling, data of physical location, proximity to other mobile devices, the activation of various device features, complexity of testing, power consumption, and issues associated with transmission through gateways and the telephone network [1], [2].

Software engineering techniques should assure data integrity and synchronization using for example, client-server computing. They should consider the risk of program and/or data integrity when events of potential loss of connectivity or battery power occur during a transaction or system update [3]. They should also design applications differently depending on the speed of the network on which they are being used.

The development of prototypes of the user interfaces is very required, particularly when multiple devices will be supported. An important area for mobile software engineering research is the develop-

ment of techniques for testing. Development of the mobile application is test driven [4] and it will typically be done within the context of the overall software development effort. It is insufficient to merely test mobile application on an emulator; it must be tested across many different mobile devices running different versions of the operating system on various telecom networks. Integrated test tools would help the development process [1]. There is need for customized tools to support cross-platform development and testing.

In this paper, we propose a framework for developing mobile applications based on migrant objects. A mobile application based on migrant objects is defined as a mobile application composed of a set of modules (or procedures) with small code. These modules can migrate from a mobile device to another to communicate with a located module. The module migration is the move of the procedure code and its state (context) represented by a set of attributes.

A module is represented by an object of a class with attributes and methods. This module object will be associated with a mobile object, which can move from device to another using TCP/IP sockets. The located mobile module contains a root object to play the role of server (server root object) or the role of client (client root object). These root objects are responsible for communication to move module objects between the different mobile devices.

There are many life applications for this kind of mobile applications based on migrant objects: health care, electronic commerce, electronic learning, etc. In the health care domain, for example, a doctor can use a mobile server application to migrate from his/her own mobile device to another mobile client application owned by its patients. Hence, the mobile client application now running on the patient's device can interact with its mobile server application. All required health information can be asked by this mobile module representing the doctor behaviour within the patient mobile device. When the doctor module (mobile object) returns back to the doctor mobile device, it can inform him with the patient's health information.

This development framework uses a UML like language for modelling server/client processes composing distributed mobile applications

based on migrant objects, where a mobile object resident in a process can migrate (move) to another process for communication with its resident mobile objects. For each mobile application based on migrant objects, we develop the class, object, and statechart diagrams. These diagrams are used to generate specifications as rewriting theories implemented by the Mobile Maude language [5], which extends the system Maude [6] implementing the Rewriting Logic. These specifications are executable, which help to do simulation and testing. In addition to simulation, it is possible to verify formally specified properties using Linear Temporal Logic (LTL) and the technique of model-checking with Maude LTL Model-Checker [7]. When the UML model of mobile application based on migrant objects is tested by simulation of its specification and its LTL properties are verified, an implementation code can be generated.

1.1 Related works

There are studies on software engineering issues for mobile application development, which help to be aware of some challenges during the application development life cycle and try to resolve problems to improve the performance of mobile applications. The authors in [8] provide an overview of important software engineering research issues related to the development of applications that run on mobile devices. They highlighted some software engineering issues for development processes, tools, user interface design, application portability, quality, and security. New set of research issues is asked for the different characteristics of mobile applications and their operating environments.

The authors of [9] have proposed a MDD approach for mobile application development, which includes modeling and code generation strategies for Android and Windows Phone, where UML class and sequence diagrams are employed for modeling mobile applications and code is generated from these models. The paper [10] has presented processes and procedures for developing mobile cloud applications by effectively applying UML, where the Android mobile platform and Amazon Web Service are used for cloud computing in order to demonstrate

the applicability of the proposed approach to systematically apply the UML profiles and diagrams for cloud-based mobile applications.

The paper [11] has presented model-based analysis to study energy consumption issues at early stages of mobile development. It uses Real-Time Maude for analyzing energy consumption of a whole system behavior consisting of hardware components and application programs as well as the framework. Example scenarios on detecting energy bugs demonstrate that the time-bounded analysis method using Real-Time Maude is effective. The authors in [12] have written a survey as a result of interviews with mobile application developers. With this survey, we can understand the main challenges that face mobile application developers, which are developing applications across multiple platforms: lack of robust analysis, testing tools, and the problems of emulators that are slow or miss many features of mobile devices. The most useful tool for mobile application development is Android Studio [13], which has become the primary IDE for native Android application development. It provides code editing, debugging, and testing tools within the development environment using emulators.

This paper is organized as follows. In Section 2, we present how to model a mobile application based on migrant objects with UML. The model transformation to Mobile Maude specifications is presented in Section 3. Section 4 explains in detail the simulation of the specifications and their formal verification against LTL properties. An implementation of automatic generation of Mobile Maude specifications from UML diagrams based on Transformation Graph Grammars (TGGs) [14] is presented in Section 5. At the end, conclusions and perspectives are given.

2 Modelling mobile applications based on migrant objects

The key elements for modelling Mobile Applications based on Migrant Objects (we call them MAMOs) are processes and mobile objects. A process is a computational environment, which is located in a mobile

device, where mobile objects can reside. Each mobile object, which is characterized by its own code and state, has the ability to move from a process to another in different locations and it uses messages to communicate asynchronously with the other mobile objects. The mobile objects can execute their code in the located computational environment as response to incoming events.

An overall configuration of MAMO application is a set of processes, where a mobile object resident in a process can migrate to another process for communication with its resident mobile objects [6]. The code of each mobile object is represented by a small object-oriented program (module), and its state is represented by data of a set of objects and messages, which can be changed by executing its own code at the process level.

To model a MAMO, we need to describe a model composed of three diagrams (class, statechart and object diagrams). A class diagram is composed of three classes (Figure 1). The first two classes are called "ServerRootObject" and "ClientRootObject", which are specializations (subclasses) of the class "RootObject" and they have different behaviours. The third class is called "MobileObject", which is used to create mobile objects.

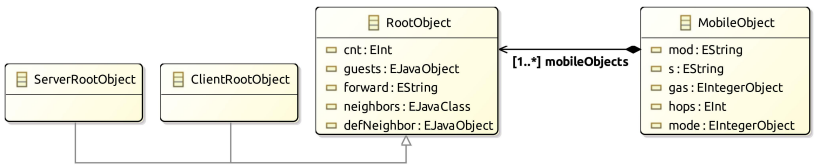


Figure 1. Class diagram of the main classes: RootObject, ServerRootObject, ClientRootObject and MobileObject

To facilitate the transformation process, the definitions of these classes are based on Mobile Maude implementation [5]. The first two classes are defined to create sockets for communication between mobile objects composing the MAMO. There will be only one object (instance) of the class "ServerRootObject" to create TCP/IP sockets for

listening on for connections from other root objects, which are of the class "ClientRootObject". The mobile objects are instances of the class "MobileObject" and they can reside with a server root object or a client root object to be able to communicate.

The root object ("RootObject") class has a set of attributes. The attribute "cnt" is a counter to produce names for new mobile objects that are created at the process level. The root object of each process keeps information, which is a set of names (identifiers) of the mobile objects currently in it in the "guests" attribute. Each root object has another attribute called "forward" to save the forwarding information concerning the whereabouts of its residents. The communication actions of root objects are represented by its attribute "state", which is only "idle" during their creation to indicate their activation as a client socket (if it is a client root object) or as a server socket (if it is a server root object). Then, the root objects change their state to the state "waiting-connection" and they stay on it until getting acceptation of connection from the server or client socket, which will pass them to the state of "active" mode to begin the normal activity of communication. Each root object has a routing table maintained in its attribute "neighbors". It is used to associate socket object identifiers with location identifiers. If the neighbours table doesn't contain a communication socket associated with a particular location, then an additional default socket will be specified in the attribute "defNeighbor" to be associated with this particular location.

The "MobileObject" class has the attribute "mod" to store the meta-representation of the object-based module using the operator "upModule". The mobile object's state (context) must be stored in the attribute "s", which is meta-representation (using the operator "upTerm") of a pair of configurations meaningful for the module in "mod". The attribute "hops" of the class "MobileObject" is defined to be incremented every time a mobile object has moved from one process to another. This number of hops performed by a mobile object will be used by the forwarding process. The attribute "mode" indicates if the mobile object is idle or active. Its initial value is set to be active to make the mobile object on activity from the beginning and it cannot

be in active mode if it is in motion. The other attribute in this set, is the "gas" attribute, which is used to limit the number of resources to be consumed by the mobile objects. It is possible to add attributes to this set during the modelling if necessary.

The class MobileObjects' attributes named "mod" for procedure code specification and "s" for procedure state context will be defined by a class association, which is associated with the class MobileObject. This class association extends the class diagram model by classes for mobile applications (modules) to define the behaviour of the mobile objects (instances of the class MobileObject). This mobile objects modelling is viewed as a distribution of located configurations; each one is executed in a different process. The created mobile objects can reside in a process to execute their code specified by the attribute "mod". Its execution can change its state identified by the attribute "s" and it can also communicate with other mobile objects by sending and receiving messages to and from the processes locating them.

The server and client root objects are identified by names of the form $l(IP, n)$, where l stands for location, IP is an Internet Protocol address of the machine in which the process is being executed and n is a natural number. The uniqueness of names for root objects composing located configurations of distributed processes should be guaranteed for proper functionality. Only one root object of the class *RootObject*, will be in a located configuration to be responsible to maintain information concerning the process location, its mobile objects, and the situation of the mobile objects that are created in it and moved to other processes.

Mobile objects with their context state and their code written with an object-oriented language, can migrate between processes and communicate by sending and receiving asynchronous messages. The mobile objects are identified by names of the form $o(l(IP, n), k)$, where o stands for object, $l(IP, n)$ is the identifier of the root object of the process in which it was created, and k – a natural number. A mobile application based on migrant objects, which runs on a mobile device, is a configuration of an object of the class "ServerObjectManager" or "ClientObjectManager" and a set of objects of the class "MobileObject". A mobile object can move (moving its code and its current state)

from a device configuration to another using the TCP/IP sockets created by the server and/or client objects. Two mobile objects in the same configuration can communicate using messages defined during modelling to change their states. So, if a mobile object wants to communicate with another mobile object in a different configuration, the former should first move to this configuration containing the mobile object to be able to communicate with it.

3 Generation of mobile Maude specifications

To simulate mobile application based on migrant objects (MAMO), a transformation must be applied on its models of class diagram and statechart diagram to generate Mobile Maude specifications with respect to the rewriting logic syntax. These specifications are rewriting theories, which can be executed via Mobile Maude configurations that will be generated from the object diagram modelling objects of the MAMO. Mobile Maude [5] is a mobile object language, which extends Maude [6] for specifying mobile computation. The Mobile Maude specifications are executable, which helps to use them as prototypes of the language and then applications implementing mobile object can be simulated.

The mobile object's state identified by the attribute s must be the meta-representation of a pair of configurations meaningful for the module in the attribute *mod*. These two attributes, which are specified in a class association will be added to the attributes set of the class "MobileObject". The attribute s has the form of a conjunction $Conf$ & $Msgs$, where the first part of the conjunction $Conf$, is a configuration of objects and incoming messages to be processed by the mobile Maude system. The second part of the conjunction $Msgs$, represents a multi-set of messages to be sent as a result of handling the configuration $Conf$.

The root and mobile objects are modelled by a small set of mobile Maude rules, which code their mobility and message sending. These rules are independent from the application. It is possible also to write the MAMO code as Maude object-oriented modules. The mobility of an object can be realized by sending two different messages to the

root object. The first message is $go(L)$, which means the sender mobile object request moving from its current location to the specified location L. This means the mobile object only specifies the location to go to. The second message is $go - find(O, L)$, which is sent when requesting the move to a location, where the mobile object O resides (it can be L). In this case, the mobile object only knows the identifier of the object (O) to catch up with, not the location it is at. But, it can give a tentative location (L), which can be the home location of O.

A mobile object can send messages to other mobile objects. These messages between mobile objects have the form $to\ O\ :\ msg$, where O is the receiver of the message and msg is the message content, which can be of any kind and it can indicate the name of the sender, so the receiver will know the identifier of the sender. The MAMO applications are object-oriented, where the operator "&" is used as constructor by the mobile objects to send messages and to move to other processes using the socket connections. The communication can be between objects inside the same mobile object, where the messages can be of any format and this communication may be synchronous or asynchronous. Communication can also be between objects in different mobile objects. In this case, these mobile objects can be in the same process or in different processes. This communication should be asynchronous and it is transparent to the mobile objects and the messages must be of the form $to\ O\ :\ msg$, which is explained before.

When a rule of a mobile Maude module has a configuration identifying a mobile object A is executed, and the second component of its result state has the message go or $go - find$, then the mobility of the identified mobile object A in the configuration is initiated. The $go(L)$ message will make the root object (manager) moving the mobile object A to the location L. However, the $go - find(B, L)$ message will try to move the mobile object A to reach the objet B that can be itself on move. It begins with the specified location L as the first tentative, if A doesn't find the object B, it will go on looking for B in a different location.

Each mobile object has a tray of outgoing messages; if it decides to migrate to a different process in a target location L, it will add the

message $go(L)$ to this tray of messages. This will invoke the operation of sending the *go* message to inter mobile objects, where the sender mobile object is indicated as an argument of this message. At the end of this operation, the *go* message will be then removed from the outgoing messages tray. If the mobile object is on move, it will be made inactive by the *go* message. If the location of the message's sender is different from the location of its receiver, then the root object of the sender's location will send this message to the desired location through the appropriate socket. The root object of the destination location will update its forwarding information if it receives this message. When the mobile object reaches its home location, it will be informed by the corresponding root object of this message.

3.1 Example of healthcare domain

We explain on a simple example how a mobile application based on migrant objects (MAMO) can be developed with this formalism. In this example we have patients and a doctor in a MAMO for healthcare domain; a doctor visits several patients, who provide him information on their health. The doctor looks for the patients need care, and once he has visited all the patients, he goes back to his home location where human doctor who is the owner of the mobile device can consult his patients' health information. This description allows us to identify the actors to be represented as mobile objects, which may migrate (move) between the different processes composing the MAMO application. In this approach the specification of the MAMO applications consists of objects embedded inside mobile objects, which communicate with each other via messages. According to the semantics of the language Mobile Maude, we represent this as a dependency between the dependent class "MobileObject" and its dependency class "Patient" or "Doctor", where their codes should be executed (see the code of the processes in Section 4).

To model the doctor and patient classes, we represent patients and doctor as objects of respective classes *Patient* and *Doctor*. Such objects in the MAMO application code will then be embedded inside their corresponding mobile objects. In the class diagram, which extends the

class diagram of Figure 1 of this particular mobile application based on migrant objects, the class "Patient" has attributes description with the patient name, temperature, blood pressure and glucose levels.

$$\begin{aligned} &class\ Patient \mid name\ :\ String, temperature\ :\ Float, \\ &\qquad\qquad blood{-}pressure\ :\ Float, glucose\ :\ Float \end{aligned}$$

A doctor class has an attribute called *patients* with a list of identifiers for the known patients mobile objects. It has also an attribute called *state* with its current state, which can be $State1$ (initial state), $State2$, $State3$ or $lastState$ (last state). These states (generated from the statechart diagram) are used to synchronise the rules execution, which represent its behaviour. Finally, the doctor class keeps information about the patients' health information.

$$\begin{aligned} &class\ Doctor \mid patients\ :\ patients, state\ :\ DoctorState, \\ &\qquad\qquad health{-}informations\ :\ patient(temperature, \\ &\qquad\qquad blood{-}pressure, glucose) \end{aligned}$$

Each mobile object will carry the representation term of its state (context) and the code managing the behaviour of the objects and messages of the configuration representing this state. In the sample mobile application MAMO, we have two different classes of mobile objects: patients and doctor. Although the objects representing the patients don't move, they should be modelled as mobile objects to be able to send and receive messages from other mobile objects through the mobile Maude system. The following statechart diagram (Figure 2) summarizes their behaviours. We remark that the mobile object doctor continues its execution from the state "doctorState3" after it moves to the client process of the mobile object patient when it was at the state "doctorState2".

The transformation to Mobile Maude code is based on formal semantics given to the statechart diagram. Every mobile object has an initial (located configuration) state, which will be defined in the object diagram. The doctor mobile object has a last state ($lastState$), however the mobile object patient has no last state, which means its execution doesn't terminate. A transition in the statechart diagram is

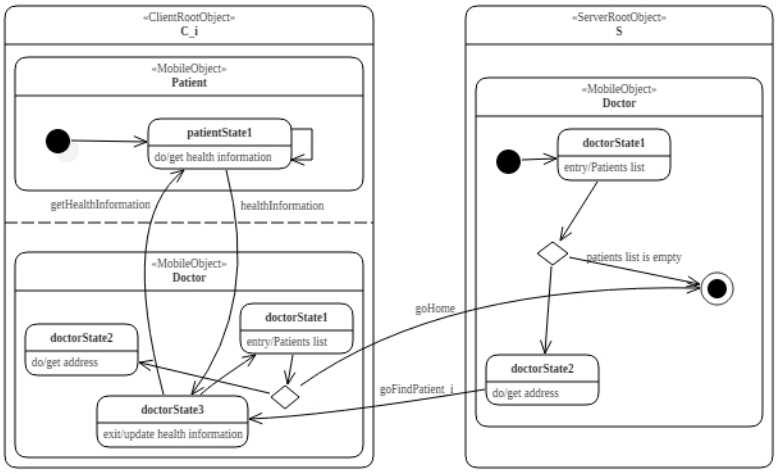


Figure 2. Statechart diagram of the mobile objects doctor and patient

represented by a rewriting rule of the form.

$$rl[State] : (State \wedge Condition)\ \&\ incoming-message => State'\ \&\ outgoing-message$$

where $State$ is a state value of the attribute $state$, $Condition$ is any value of any attribute from the attributes set. The incoming message $incoming-message$ is a received message that labels the incoming edge of the state $State$ (or $none$, if there is no label), which constitutes a transition guard with the condition $Condition$. The execution of this rule will change the mobile object configuration by changing the attribute value of $state$ to be $State'$ and sending the outgoing message $outgoing-message$ that labels the outgoing edge of the state $State'$ (or $none$, if there is no label). A doctor visits several patients to ask each one he visits for the description of his health, represented here by its temperature, blood pressure and glucose levels. In the statechart diagram, when the doctor is in the server root object at the state $State1$ (the initial state) looks for the address of the first patient in the patients list. If there is no patient location to visit, it goes to its last state (named $lastState$), else it goes to the state $State2$, from which it

can go to find the patient location.

$$\begin{aligned} rl[State1] :&(State1 \wedge list\ of\ patients\ is\ empty)\ \&\ none => \\ &lastState\ \&\ goHome \\ rl[State1]\ :&(State1 \wedge list\ of\ patients\ is\ not\ empty)\ \&\ none => \\ &State2\ \&\ go\ find\ first\ patient\ in\ the\ list \end{aligned}$$

A message $go - find$ is sent to the root object. As response to this message, the doctor mobile object moves to the process location of the first patient mobile object in the patients list. At this process locating the patient mobile object, the doctor mobile object continues its execution from the state $State3$ by sending the message $get–health–information$ to the patient mobile object.

$$rl[State2]\ :\ State2\ \&\ none => State3\ \&\ get\ health\ information$$

The patient sends back his health information stored in its attributes, which will be received when the mobile object doctor is in its state $State3$ and then saved in its attribute $health-information$.

$$rl[State3]\ :\ (State3 \wedge health\ information)\ \&\ none => State1\ \&\ none$$

From the state $State3$, the doctor returns back to the state $State1$ if it has received the health information for looking the next patient address to move to its process location by the same way. Once the mobile object doctor has visited all the patients it knows (the patients list is empty), it goes back to its home location by sending the message $go(home-location)$. Then, all the patients' health information is in its attribute named $health-information$, which can be consulted by the mobile device owner. Patients receive from the doctor messages of the form $get-health-information(D)$ with D as the identifier of the doctor mobile object sending the message. Patients can send messages of the form $health–information(Name, T, BP, G)$ with $Name$ – string representing the patient's name, T, BP, and G – real numbers representing the patient's temperature, blood pressure, and glucose levels, respectively.

$$\begin{aligned} rl[State1]\ :&(State1 \wedge get\ health\ information)\ \&\ none => \\ &State1\ \&\ health\ information \end{aligned}$$

From the extended class diagram and the statechart diagram, we generate Mobile Maude modules for patient's behaviour and doctor's behaviour. Patient's behaviour is represented by the rewrite rule labelled with $State1$ (indicated in the statechart diagram), which corresponds to its unique state $State1$. When a patient receives a heath information request, it sends back the information to the doctor. The whole module defining the patient's behaviour in Maude is below.

```
mod PATIENT is
  ...
  rl [State1] : Conf (to P : get-health-information(D))
     < P : V@Patient | name : Name, temperature : T,
        blood-pressure : BP, glucose : G, AtS > & none =>
     Conf < P : V@Patient | name : Name, temperature : T,
        blood-pressure : BP, glucose : G, AtS > &
        (to D : health-information(Name, T, BP, G)) .
endm
```

However, the doctor module, which is presented below, is more complex. Its behaviour is composed of four states. In the state encoded by the rule labelled by $State1$, it moves to the process containing the mobile object, where the object patient identified by $o(L, N)$ is in and it changes its state to $State2$ to ask the patient object its health information by sending the message $to\ P\ :\ get{-}health{-}information(D)$, where P is the object identifier of the patient (i.e., $o(L, N)$) and D is the object identifier of the doctor requesting health information.

The patient object, which is now located with the doctor mobile object in the same process, responds with the message $to\ D\ :\ health{-}information(Name, T, BP, G)$ (the rule encoding the state $State1$ of the patient mobile object). When it is in the state $State3$ and the patients list is not empty, it will return back to the state $State1$ to move to the located configuration of the next patient mobile object, else it will return back to its original home location (the server root object) carrying the health information of all the patients.

```
mod DOCTOR is
  ...
  rl [State1] : < D : V@Doctor | patients : o(L,N) . OP,
     state : State1, AtS > & none => < D : V@Doctor |
     patients : o(L,N) . OP, state : State2, AtS > &
     go-find(o(L,N), L) .
  rl [State1] : < D : V@Doctor | patients : no-id,
```

```
      state : State1 , home-location : o(L,N), AtS > &
      none => < D : V@Doctor | patients : no-id ,
      state : last , home-location : o(L,N),AtS> & go(L).
  rl [State2] : < D : V@Doctor | patients : P . OP,
      state : State2 , AtS > & none => < D : V@Doctor |
      patients:P . OP, state:State3 , AtS > &
      (to P : get-health-information(D)) .
  rl [State3] : (to D : health-information(Name, T, BP,
      G)) < D : V@Doctor | patients : P . OP, state :
      State3 , AtS > => < D : V@Doctor | patients : OP,
      health-information : Name(temperature: T,
      blood-pressure: BP, glucose: G), state : State1 , AtS > .
endm
```

4 Simulation and verification

We will show how we can simulate and verify this mobile application based on migrant objects by using the Maude system. Our sample doctor/patients configuration is constituted of four located configurations; each one will be executed in a Maude process. Each located configuration contains one root object (there are four configurations).

The mobile object doctor is the resident of the server root object; however, the patients are residents of the client root objects. From the object diagram, we generate four located configurations. The first located configuration (shown below) contains a *ServerRootObject*, with identifier $l(IP, 0)$, and a mobile object identified by $o(l(IP, 0), 0)$ with a doctor module in its belly. This configuration represents the central process of the star network and it must be executed first, because it has the object *ServerRootObject*, which is responsible for creating the server socket to listen and then accept connections from the other objects.

```
mod PROCESS-DOCTOR is
  ex DISTRIBUTED-MOBILE-MAUDE . ex DOCTOR . op IP : -> String .
  op initial : -> Configuration . eq IP = "localhost" .
  eq port = 8000 . eq initial =<><l(IP, 0) : ServerRootObject |
    cnt : 1,state:idle,guests : o(l(IP, 0),0),defNeighbor:null,
    neighbors : empty, forward : 0 |-> (l(IP, 0), 0) >
  < o(l(IP,0),0):MobileObject | mod:upModule('DOCTOR,false),
      s : upTerm(< o(l(IP, 0), 0) : Doctor | status : done,
      home-location : o(l(IP, 0), 0), patients : o(l(IP, 1),0).
      o(l(IP, 2),0) . o(l(IP, 3),0)> & none), gas : 200,
```

```
      mode : active , hops : 0 > .
endm
```

Note how the Maude meta-level function *upModule* is used to obtain the meta-representation of the module *DOCTOR*, and how the function *upTerm* is used to meta-represent the initial state of the inner object, where the list of patients is declared as value of the attribute *patients*. The other configurations below are for the three mobile objects patients, which contain *ClientRootObject* with a Patient object in the belly of a mobile object. Each patient has a different name "PatI" ("I" stands for 1, 2 or 3) and different health information: the temperature T (39.0, 39.5, and 39.0), the blood-pressure B (12.0, 13.2, and 14.0) and the glucose G (1.3, 1.56, and 1.2). The mobile object of the patient, which has the name "PatI" is called "o(l(IP,I),0)" and it is located in a process with the client root object "l(IP,I)", where "I" stands for 1, 2 or 3.

```
mod PROCESS-PATI is
  ex DISTRIBUTED-MOBILE-MAUDE . ex PATIENT . op IP : -> String .
  eq IP = "localhost" . eq port = 8000 .
  eq server-address = "localhost" . op initial :->Configuration .
  eq initial = <> < l(IP, I) : ClientRootObject | state : idle,
     cnt : 1, guests : o(l(IP, I), 0), neighbors : empty,
     forward : 0 |-> (l(IP, I), 0), defNeighbor : null >
     < o(l(IP, 1), 0):MobileObject |mod:upModule('PATIENT, false),
       s : upTerm(< o(l(IP, I), 0) : Patient | name : "PatI",
       temperature : TI, blood-pressure : BI, glucose : GI >
       & none), gas:200, mode:active, hops:0 > .
endm
```

These four configurations represent an overall located configuration (processes) of the mobile application MAMO. In this case the four processes run on the same machine, with IP address *localhost*. The execution results of these four different Maude processes are described as follows. First, the doctor travels to the location $l(IP, 1)$ of the first patient $o(l(IP, 1), 0)$ and asks him about his health information. The patient resident in this location responds with his information. Then, the doctor travels to the next location $l(IP, 2)$ to ask its resident, which is the patient $o(l(IP, 2), 0)$.

The last move before returning to its home location $l(IP, 0)$, the doctor identified by $o(l(IP, 0), 0)$ travels to the third patient loca-

tion $l(IP, 3)$ to ask its resident, which is the patient identified by $o(l(IP, 3), 0)$. The doctor has finished his travel at his home location and has the names and health information of all the patients. Its attribute "hops" has the value 4, which means that it has 4 jumps, it has moved to 3 locations and it has returned back home. The value of the attribute "gas" is $200 - 193$, which means it has consumed 7 resources. The simulation results for the mobile objects "Pat1", "Pat2", and "Pat3" show that these three mobile objects didn't move (the values of their attribute "hops" are zero) and they have consumed two resources ("gas" equals $200 - 198$).

4.1 Formal verification

Formal verification is useful to check properties that cannot be verified by simulation as, for example, the property a doctor mobile object should visit only once every patient. The Maude LTL model checker implemented within the system Maude [7] can be used to verify the satisfaction of LTL properties by Maude specification if its set of states reachable from an initial state is finite. The Maude LTL model checker can be used to check whether a given initial overall configuration fulfills a mobile application MAMO property described by Linear Temporal Logic (LTL), such as safety property (something bad never happens) and liveness property (something good eventually happens) [15]. Verifying LTL properties on mobile applications MAMOs is not easy. The MAMOs applications are distributed among several hosts; therefore, the Maude LTL model checker cannot be applied directly to prove global properties. In the following, we show how this issue is addressed.

The problem has been solved by specifying additional mobile object responsible for model-checking models that can be built during the activity of the other mobile objects. A model checker mobile object is another associated class with attributes, the first attribute *model* is for capturing the states sent by the specified mobile objects, the second attribute $formula$ is to specify the LTL formula to verify the model built, and the last attribute $model-check$ will show the model checking result, which is true if the property is satisfied or a counterexample (a

trace) showing why the property is falsified. This mobile object is located in the process of the server root object, and it will receive states from the other mobile objects (in the patients/doctor example, it receives location of the doctor mobile object each time it travels to it). The following Mobile Maude code represents the modified doctor configuration in the object diagram.

```
mod PROCESS-DOCTOR is
  ...
  < o(l(IP, 0), 0) : MobileObject | mod :upModule('DOCTOR, false),
      s : upTerm(< o(l(IP, 0), 0) : Doctor | status : done,
      model-check:o(l(IP,0), 1), home-location : o(l(IP, 0), 0),
      patients : o(l(IP, 1), 0) . o(l(IP, 2), 0) . o(l(IP, 3), 0)>
      & none), gas : 200, mode : active, hops : 0 >
  < o(l(IP, 0), 1) : MobileObject | model : nil,
      model-check : waiting-model,
      formula : ((<> (location(l(IP, 2)))) /\
         (location(l(IP,2)) -> [](~ location(l(IP,2))))),
      mod : upModule('MC, false),gas : 200,mode :active,hops :0> .
endm
```

The mobile object identified by $l(IP, 0), 1)$ is the model checker mobile object and its belly module MC is the Maude module containing rewriting theory for building the required model to be the value of the attribute *model*. The LTL property in the attribute $formula$ is used to verify the constructed model against the property that the mobile object doctor eventually reaches the location of the mobile object of the patient *pat*2 and it will never return to it, which formally is expressed by

$$\Diamond(loc(l(IP, 2))) \wedge (loc(l(IP, 2)) \implies \Box(\neg loc(l(IP, 2))))$$

The result of model checking is shown in Figure 3 (the value of the attribute model-check is true, which means the property is satisfied).

5 Implementation

The objective of the implementation is to develop a tool by which we can generate automatically the Maude specifications from the UML diagrams to do simulation and model checking of LTL properties. When

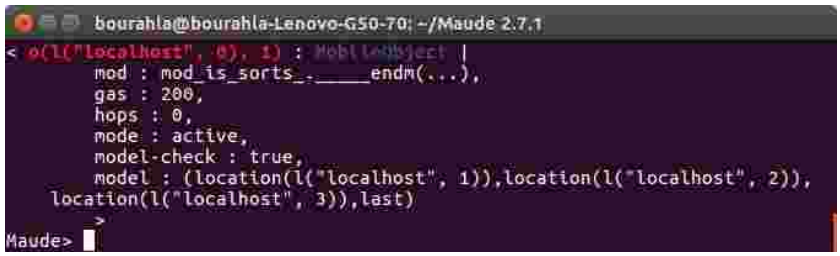


Figure 3. Model construction and model checking results

the simulation results and LTL properties are satisfied, we can use this implemented tool to generate concrete mobile application (its code), which can be deployed on mobile devices. For this automatic generation, we use generation and transformation of models based on Model Driven Architecture (MDA) [16]. Triple Graph Grammars (TGGs) are used to specify bidirectional model transformation [14]. Consistency relations can be specified with TGGs as rules [17] to control the correspondence between the source and target models. With TGGs, to guarantee that a source model is consistent with a target model, we specify a correspondence model between them, which will be used by a set of TGG rules for checking the consistency. This correspondence model should be conforming to a defined correspondence meta-model. Also, the source and target models conform to their corresponding meta-models that are connected by the correspondence meta-model. The triple of source, target and correspondence meta-models is referred to as a TGG schema [18].

The class (declarative parts) and statechart (behaviour parts) diagrams are used to generate the Maude specifications of modules representing the migrant objects. The EMF (Eclipse Modelling Framework) ECore of the Eclipse platform is used for the implementation of this automatic transformation. Thus, the meta-modelling layer of the EMF tool is used to graphically model the meta-models of the class and statechart diagrams, which represent the source meta-models and the meta-model of the mobile Maude as the target meta-model. The

correspondence meta-model is added to connect the source and the target meta-models. The Graphical Modelling Framework (GMF) [19] tool will be then used for editing the different models of the MAMO application based on the specified formalism defined by the created meta-models.

The TGG Interpreter [20] is used to generate mobile Maude models by executing graph rewriting rules based on the source, target, and correspondence meta-models. These rules can be specified to do model-to-model (M2M) or model-to-text (M2T) transformations using the TGG technique. The text generated in the model-to-text transformation is a mobile Maude code, which is generated with respect to defined templates. The template is a target text containing holes as variable parts. These holes contain meta-code, this means code creating code. The variable parts are computed during the template instantiation time. In this case, we use Xpand language to create templates for code generation of mobile Maude modules from EMF models of mobile Maude.

For the verification (simulation and formal verification) of these migrant objects in the located configurations of processes, we generate from object diagrams representing the located configurations the mobile Maude processes. By the same way, using TGG transformations as explained before, we realize this task. We use Maude LTL Model-Checker to verify the state-transition model constructed during simulation against specified LTL properties, a counter example is generated in case a property fails.

By the same process, Figure 4 shows the steps of the code generation, where the mobile Maude modules and located configurations are used as inputs to generate Android Java Code [21] to be deployed on Android devices. However, the programmer will be asked to add additional Java code for user interface, as to make secure logins, to input user data and to handle information as key data or SQL data base, etc.

6 Conclusions and perspectives

A technique to use software engineering principles for developing mobile applications is proposed. The UML class and statechart diagrams

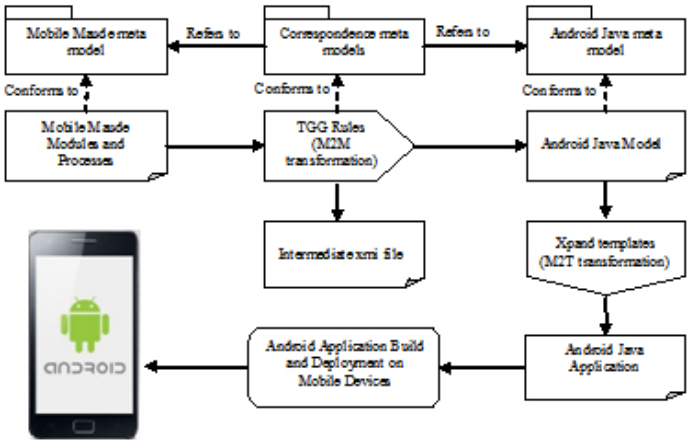


Figure 4. Generation of mobile application code

modelling the mobile app, are first transformed to mobile rewriting theories to be the belly modules of mobile objects that will reside in processes containing root objects (server root object and client root objects) to manage communication between the different mobile objects and hence creating located configurations by transformation of object diagrams. Thus, it is possible to execute each located configuration in a process to simulate these UML models and the execution results will be checked and compared with the desired behaviours. We can also formally verify these located configurations using the model-checking technique. At the end, an equivalent Android mobile application can be generated using a transformation process of mobile rewriting theories to Android Java code to be built and deployed on mobile devices. This software engineering technique is implemented as a prototype and extensively tested with small examples. The implementation results encouraged us to continue with this work. As perspectives, we would like to integrate it with the existing tools for development of mobile applications.

References

[1] A. I. Wasserman, "Software engineering issues for mobile application development," in *Proceedings of the Workshop on Future of Software Engineering Research, FoSER 2010, at the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, November 7-11, 2010*, G. Roman and K. J. Sullivan, Eds. ACM, 2010, pp. 397–400.

[2] L. Corral, A. Sillitti, and G. Succi, "Software assurance practices for mobile applications - A survey of the state of the art," *Computing*, vol. 97, no. 10, pp. 1001–1022, 2015.

[3] L. Corral, A. B. Georgiev, A. Sillitti, and G. Succi, "Can execution time describe accurately the energy consumption of mobile apps? an experiment in android," in *Proceedings of the 3rd International Workshop on Green and Sustainable Software, GREENS 2014, Hyderabad, India, June 1, 2014*, H. A. Müller, P. Lago, M. Morisio, N. Meyer, and G. Scanniello, Eds. ACM, 2014, pp. 31–37.

[4] I. do Nascimento Mendes and A. C. Dias-Neto, "A process-based approach to test usability of multi-platform mobile applications," in *Design, User Experience, and Usability: Design Thinking and Methods - 5th International Conference, DUXU 2016, Held as Part of HCI International 2016, Toronto, Canada, July 17-22, 2016, Proceedings, Part I*, ser. Lecture Notes in Computer Science, A. Marcus, Ed., vol. 9746. Springer, 2016, pp. 456–468.

[5] F. Durán, A. Riesco, and A. Verdejo, "A distributed implementation of mobile maude," *Electr. Notes Theor. Comput. Sci.*, vol. 176, no. 4, pp. 113–131, 2007.

[6] M. Clavel, F. Durán, S. Eker, P. Lincoln, J. Meseguer, N. Martí-Oliet, and C. L. Talcott, Eds., *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, ser. Lecture Notes in Computer Science, vol. 4350. Springer, 2007.

[7] S. Eker, J. Meseguer, and A. Sridharanarayanan, "The maude LTL model checker," *Electr. Notes Theor. Comput. Sci.*, vol. 71, pp. 162–187, 2002.

[8] A. I. Wasserman, "Developing mobile software with FLOSS," in *Open Source Systems: Long-Term Sustainability - 8th IFIP WG 2.13 International Conference, OSS 2012, Hammamet, Tunisia, September 10-13, 2012. Proceedings*, ser. IFIP Advances in Information and Communication Technology, I. Hammouda, B. Lundell, T. Mikkonen, and W. Scacchi, Eds., vol. 378. Springer, 2012, pp. 401–402.

[9] L. Brisolara, A. Parada, and M. Marques, "Automating mobile application development: Uml-based code generation for android and windows phone," *Revista de Informática Teórica e Aplicada: RITA*, vol. 22, pp. 31–50, 11 2015.

[10] D. Kim, "Development of mobile cloud applications using uml," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 596–604, 02 2018.

[11] S. Nakajima, "Formal analysis of android application behavior with real-time maude," in *2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications*, 2015, pp. 7–12.

[12] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, Maryland, USA, October 10-11, 2013*. IEEE Computer Society, 2013, pp. 15–24.

[13] "https://developer.android.com/studio/index.html, Android Studio," accessed: 2019-06-01.

[14] A. Anjorin, E. Leblebici, and A. Schürr, "20 years of triple graph grammars: A roadmap for future research," *ECEASST*, vol. 73, 2015.

[15] M. Clavel, F. Durán, S. Eker, P. Lincoln, J. Meseguer, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, "LTL model checking," in *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, 2007, pp. 385–418.

[16] A. Kleppe, J. Warmer, and W. Bast, *MDA explained - the Model Driven Architecture: practice and promise*, ser. Addison Wesley

object technology series. Addison-Wesley, 2003.

[17] M. Bendiaf, M. Bourahla, M. Boudia, and S. Rehab, "A model transformation approach for specifying real-time systems and its verification using rt-maude," *IJITWE*, vol. 12, no. 4, pp. 22–41, 2017.

[18] E. Leblebici, A. Anjorin, A. Schürr, S. Hildebrandt, J. Rieke, and J. Greenyer, "A comparison of incremental triple graph grammar tools," *ECEASST*, vol. 67, 2014.

[19] "URL: https://www.eclipse.org/modeling/gmp/, Graphical Modeling Framework," accessed: 2019-04-10.

[20] "http://jgreen.de/tools/tgg-interpreter/, TGG Interpreter," accessed: 2019-04-10.

[21] M. Murphy, *The Busy Coder's Guide to Advanced Android Development*. CommonsWare, LLC, 2009. [Online]. Available: https://books.google.dz/books?id=fQmvPwAACAAJ

Aissa ElMahdi Bourahla, Mustapha Bourahla

Received July 5, 2020
Accepted October 6, 2021

Aissa ElMahdi Bourahla
Computer Science Department, University of M'Sila
BP. 166 Ichebilia, M'Sila, 28000, Algeria
Phone: +213 0540834764
E–mail: aissa.bourahla@univ-msila.dz

Mustapha Bourahla
Laboratory of Informatics and its Applications, Computer Science Department, University of M'Sila
BP. 166 Ichebilia, M'Sila, 28000, Algeria
Phone: +213 0778574572
E–mail: mustapha.bourahla@univ-msila.dz

Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

Interactive System for Algorithm and Data Structure Visualization

Patrik Perháč, Slavomír Šimoňák

Abstract

This work is dedicated to the design, implementation and evaluation of a new algorithm visualization system. The currently available systems and libraries are briefly compared with each other based on the visualizations and functionalities they provide. Since the analyzed tools didn't meet all of the given requirements, we decided that the development of a new system for algorithm and data structure visualizations would be beneficial for use in teaching the subject Data Structures and Algorithms. The new system was designed to be easily usable, extensible, available and to cover the basic functionalities available in similar systems and other useful features. The proposed system provides three types of visualizations: predefined visualizations, to explain how each data structure and algorithm works; interactive visualizations, to let the user interact with the visualization directly; and interactive exercises, to let the users test their knowledge. These three types of visualizations cover the whole learning process, provide theoretical and practical knowledge, and also a way to test their knowledge. The system is implemented in the form of a web application and, for the visualizations, the JSAV library is used. The system was also evaluated by the users via a survey and several improvements were implemented in the system based on the feedback provided by the users.

Keywords: algorithms, data structures, visualization, web application, JavaScript, JSAV.

MSC 2010: 68P05, 68P10, 68Q65, 97U50.

1 Introduction

In this work, we focus on making the learning process of data structures and algorithms easier for the students. These topics are thaught in the subject Data Structures and Algorithms. This subject is taught in the second year of the bachelor's degree program Informatics of the Department of Computers and Informatics. In this subject, different abstract data types and algorithms are introduced to the students. Understanding how some of the more complex data structures and algorithms work can be harder to comprehend for some students at first. However, this process can be made easier by the use of supporting tools that help visualize the data structures, so students can understand them better. Different kinds of supporting tools were in use while teaching the subject Data Structures and Algorithms, each of which had its benefits and limitations. These tools are evaluated later in this paper.

The use of visualizations alongside with pseudocode, in which the currently executed line is highlighted can help significantly in understanding more complex data structures and algorithms [1]. This way the student has an overview of how each step of the algorithm affects the data structures used [2]. Another way to ensure the students get the most out of learning these topics is to provide them with engaging activities besides the visualizations of the data structures and algorithms [3]. Just looking at the visualizations may not be sufficient enough to fully understand the topic, but interacting with the data structures directly can help students gain a better, more in depth understanding. This is why, besides predefined visualizations, it is advantageous if the user has the ability to interact with the visualization in some way, either by defining the order of the operations executed, or defining the values used within the visualized data structures. This can help the user gain some practical knowledge too [4].

1.1 Requirements for the supporting tool

Before we select the supporting tool to be used in the teaching of the subject Data Structures and Algorithms, we need to define the require-

ments it needs to satisfy. These requirements need to be adjusted to the learning process and the material thought in this subject. Initially, there were two main requirements for the system:

- *Usability:* the system must be easy to understand and simple to use.
- *Accessibility:* the system must be widely accessible, and the initial configuration or setup of the system should not take more than a few minutes.

After defining these requirements, a more detailed specification was designed with the following parts [5]:

- *Online solution:* designing the system in the form of a web application and deploying it to a web server solves the accessibility requirement, since the application would be accessible to anyone.
- *Clear visualizations:* the visualizations included in the system should be easy to understand and interactions with the visualizations should be intuitive and straightforward.
- *Brief explanations:* the system should include a brief overview of the visualized algorithm or data structure.
- *Pseudocode:* for more complex visualizations, pseudocode should be provided to help better understand the visualized algorithm. The currently executed line should be highlighted in the pseudocode.
- *Controls:* a way to play the visualizations step by step in any direction should be provided [2].
- *Defining elements:* the user should be able to define the value of the data structure elements.
- *Extensibility:* the system must be easily extensible with new visualizations.
- *Predefined visualizations:* the system must provide predefined visualizations for the basic data structures.
- *Exercises:* the system should allow students to test their understanding of the visualized algorithms [6].

2 Related work

This section provides a brief overview of some of the existing systems that were analyzed in this work. The analysis included desktop applications previously used in teaching the subject Data Structures and Algorithms as well as some of the available web applications.

2.1 VizAlgo

VizAlgo is a desktop application for algorithm and data structures visualizaton, used in teaching of the subject Data Structures and Algorithms, written in the Java language [7]. Since it is written in Java, it doesn't require installation, but it has limited compatibility in Linux systems due to the libraries used, some of which need to be additionally installed on Linux systems. It provides visualizations for the basic data structures such as the Stack, Queue, Binary Search Tree. Sorting algorithms, tree traversal algorithms, MinMax and Chainmatrix algorithms are also visualized in the application [8]. Because of the need of obtaining a *.jar* file to run the program and the limited compatibility with Linux systems, VizAlgo didn't meet the requirements specified above.

2.2 Algomaster

Another desktop application analyzed was Algomaster which is also used in teaching the subject Data Structures and Algorithms. It's built on the .NET platform and provides functionalities like stepping the visualization, the ability for the user to define the data structures, pseudocode, and also call stack visualization, which is useful for visualizing recursive algorithms [9] [10]. The visualizations are implemented in the form of plugins [11]. Algomaster includes more visualizations of data structures and algorithms than VizAlgo, but since it is built on the .NET platform, it has limited compatibility.

2.3 Online learning tools

Besides the previously mentioned desktop applications, some of the online solutions available were also analyzed and evaluated based on the requirements.

2.3.1 Algorithm Visualizer

This online platform[1] provides visualizations directly from source code. The source code of the included visualizations can be edited and run after successful build. Since the visualization is generated from the source code directly, it is suitable for visualizing complex algorithms. However, this feature makes the code more complicated since the code doesn't only contain the code for the algorithm, but also the code responsible for visualizing the algorithm, which could lead to confusion while learning basic data structures and algorithms.

2.3.2 VisuAlgo

VisuAlgo[2] provides pseudocode for the visualizations as well as explanations for each crucial step of the visualization. The values in the data structures can be defined by the user, and stepping the visualization forward and backward is also supported. VisuAlgo is overall a great learning tool, but it's license doesn't allow it to be modified, which makes it not suitable for the purposes of this work.

2.3.3 Data structure visualizations

This system[3] uses its own visualization library, which uses the HTML canvas [12]. Visualizations for most basic data structures and algorithms are provided in this system, however it lacks features like the display of pseudocode or explanations for the individual steps of the algorithm. The more complex visualizations are also harder to follow

[1] Algorithm Visualizer: `https://algorithm-visualizer.org/`

[2] VisuAlgo: `https://visualgo.net/`

[3] Data structure visualizations: `https://www.cs.usfca.edu/~galles/visualization/Algorithms.html`

because of this, which makes it not suitable for the purposes of this work.

2.4 Summary

The analyzed desktop applications VizAlgo and Algomaster offer visualizations for a wide variety of data structures and algorithms, but they do not meet the requirements in terms of availability, since both applications require downloading an executable file, and have limited compatibility with operating systems other than Windows.

The analyzed online learning tools were either too complex for the teaching of basic data structures or algorithms (Algorithm Visualizer), or their license doesn't allow them to be customized for use in teaching the subject Data Structures and Algorithms (VisuAlgo). For these reasons, the development of a new system seems to be a preferable option.

3 Data structure visualization library

Since none of the analyzed learning tools met all the defined requirements, the creation of a new application, designed specifically for supporting the teaching process within the Data Structures and Algorithms subject was needed.

A framework or library was needed for implementing the visualizations in the proposed new application. For this purpose, three possibilities were compared:

- *Data structure visualizations* – the library used in the online learning tool with the same name. This library provides basic functionalities, but is not flexible enough to be used for the purposes of this work.
- *Custom library for data structure visualizations* – the creation of a custom visualization library built on the CreateJS library[4] was also considered.

[4]CreateJS: `https://createjs.com/`

- *JSAV* – provides visualizations for basic data structures, supports the creation of interactive exercises, and covers all functionalities needed for the application proposed in this work with little to no modifications.

The JSAV[5] library was selected for creating the visualizations in the new application, since it provides visualizations for the basic data structures like *array*, *list*, *graph*, *tree* or *matrix*. JSAV also supports stepping the visualization both forwards and backwards, displaying explanations for each step, displaying pseudocode with the possibility to highlight certain lines in each step. Another useful feature is the Exercise API, which helps creating interactive exercises for the students to test their knowledge of the visualized algorithm. These exercises are evaluated in real time based on a model solution which needs to be defined in advance.

4 Application design

The application that is the result of this work was designed with all the requirements in mind. It's implemented in the form of a web application, which fulfills the accessibility requirement. The usability requirement is covered by the design of the user interface of the application [13]. The application is implemented using *NuxtJS* [6], which is a framework for creating Vue applications. The layout of the application consists of three main parts, as seen on the screenshot of Figure 1.

4.0.1 Left sidebar

The left side contains the main navigation menu, where the included algorithms and data structures are listed and can be selected. The menu is implemented with the help of the *vue-sidebar-menu*[7] component.

[5] JSAV: `http://jsav.io/`

[6] NuxtJS: `https://nuxtjs.org/`

[7] vue-sidebar-menu component: `https://www.npmjs.com/package/vue-sidebar-menu`

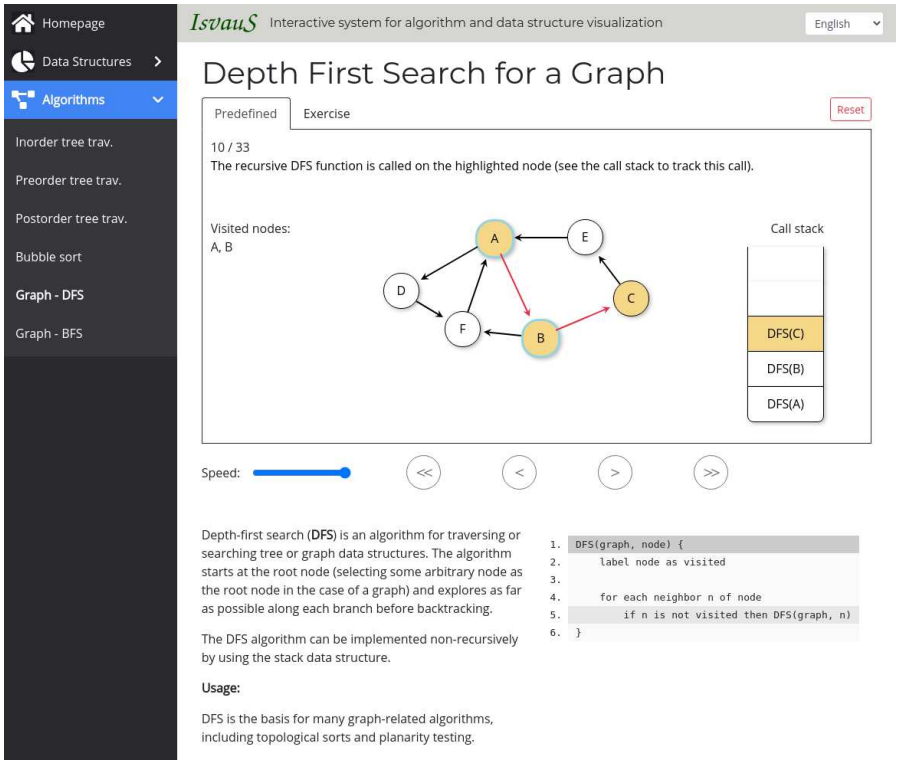


Figure 1. Screenshot of the application created as part of this work

The menu items are generated dynamically from the configuration files of the visualizations, which means that after a new visualization is added to the application the menu does not need to be changed, the new visualization is added automatically.

4.0.2 Visualizations

This is the main part of the webpage in which the visualization is displayed. The user can select which type of visualization should be displayed using the tabs beneath the title (see Figure 1). Each visualization of a data structure or algorithm can support any or all of the

three types of visualizations:

- *Predefined visualizations* consist of predefined steps containing detailed explanations for each step, a description of the selected data structure or algorithm, and pseudocode, in which the line that is currently being executed, is highlighted. Controls for the visualization are also provided, so the user can step through or jump to the beginning or the end of the visualization.

- *Interactive visualizations* allow the user to interact with the visualization directly using controls that are specific to each visualization. These interactions can include executing an operation over a data structure with user defined parameters or defining the values of a data structure for which the steps should be generated. The steps associated with each operation are recorded, which allows the user to step through the visualization if needed. Letting the user engage with the visualization increases their understanding of the visualized algorithm or data structure [4].

- *Exercises* allow users to test their knowledge about the visualized data structure or algorithm. Exercises consist of a description containing instructions and data structures that the user needs to modify in the correct way. The actions of the user are continuously compared to the model solution. The user can redo incorrect actions; however, points are not assigned for such actions.

4.0.3 Language selection

The application supports multiple languages, currently Slovak and English. The module i18n[8] is used for translating texts and automatic route generation for the different languages. Translations are loaded from *JSON* files, in which text in a given language is assigned to the translation key. Slovak language is configured as the default and fallback language of the application. The user can change language at any

[8]Module i18n: `https://i18n.nuxtjs.org/`

time using the language select component in the title of the application (as seen on the top right of Figure 1).

4.1 Configuration files

As mentioned previously, all information about the visualizations is included in the configuration files. There are two files, one for data structures and one for algorithms. These files contain *JSON* objects with all the information needed to display a visualization. The information is stored in the form of an associative array. The visualizations in the application are loaded based on the path defined under the *jsFile* key in this array. Information about each visualization includes:

- *title:* translation key for the main title of the visualization.
- *menuEntry:* translation key for the entry in the side menu.
- *jsFile:* path to the JavaScript file containing the definition of the visualization.
- *options:* a list of visualization types supported by the visualization (allowed values: static, interactive, exercise).
- *description:* object containing translations of the main description of the visualization that is displayed for all types. Translations for both languages are saved in the attributes *"sk"* and *"en"* of the object. These texts are not included in the files containing other texts because of their potential size[9].

5 Implementing visualizations

The visualizations in the application are created using the JSAV library, and each visualization is represented as a class extending a common abstract class in their own JavaScript files. This separation makes the

[9] Translations are loaded for both languages at all times, and since descriptions can be extensive, it is unnecessary for them to be loaded for all visualizations and both languages.

application easily extensible, since the structure of the application does not change when a new visualization is added. The file containing the visualization is specified by the attribute *jsFile* in the configuration. The class which defines the visualization is instantiated after the file containing it is loaded[10].

The different types of visualizations are defined in the same file by overriding one or multiple of the following methods: *initStatic*, *initInteractive*, *initExercise*. These methods are then called from the NuxtJS template when the user changes the visualization type. The data structures used in the visualization are defined and initialized in these methods. For the predefined visualizations, each step is also recorded. If interactive visualization is supported, the *setupCustomControls* method also needs to be overridden where the appropriate controls are defined for the visualization as needed. When adding controls to an interactive visualization, a callback function needs to be defined in which the structures in the JSAV visualization can be modified or new steps can also be created. Some controls require the user to input custom values. These values are encoded properly before being passed to the callback function to prevent Cross-Site Scripting (XSS) attacks [14]. To create an exercise, an initialization function and a model solution function must be created and passed to the JSAV exercise instance. More on exercise creation can be found in the documentation of the Exercise API[11].

6 Included visualizations

One of the requirements for the system was that it should include visualizations for the basic data structures and algorithms. The application currently provides visualizations for basic data structures such as the *linked list*, *stack*, and *queue*. Visualizations are also provided for *inorder*, *preorder*, and *postorder* tree traversal algorithms; *bubble sort*, *depth first search*, and *breadth first search* graph algorithms.

[10] JS files are loaded after information about the visualization is loaded form the configuration file using path parameters.

[11] Exercise API: `http://jsav.io/exercises/exercise/`

6.1 Linked list visualization

For this visualization both predefined and interactive types are supported. The predefined visualization shows how the *insert*, *remove*, *append* and *prepend* operations work on a linked list [15]. The *append* and *prepend* operations are not standard operations on a linked list, but they are suitable for visualizing how pointers behave when adding new elements to the beginning or end of the list. The interactive visualization is initialized with a linked list containing five elements. The user can interact with this list by inserting a new element to any index, removing the element on any index, appending or prepending a new element.

6.2 Stack and queue visualizations

The predefined visualization of the stack data structure uses an indexed array, a stack and a pointer structure from the JSAV library [16]. The two main operations *push* and *pop* are explained in the predefined visualization [17] [18]. The interactive visualization uses the same structures and the user can push a new value to the stack, pop the value on the top of the stack or retrieve the value on the top of the stack with the *top* operation.

The visualization of the queue data structure is similar to the visualization of the stack data structure. It also uses an indexed array, a stack, and two pointer structures from the JSAV library [16]. One pointer is for the top of the stack, the other is for the tail. The *enqueue* and *dequeue* operations are explained in detail in the predefined visualization [17] [18]. The interactive visualization lets the user enqueue custom values in the queue, dequeue the value from the front of the queue or retrieve the value on the front of the queue with the additional *front* operation.

6.3 Tree traversal algorithms

There are visualizations in the application for the *preorder*, *postorder* and *inorder* tree traversal algorithms [18]. Each visualization provides

a predefined type and an exercise. The algorithms are explained step by step in the predefined visualizations. The trees on which the algorithms are illustrated share the same structure, so the relation between the three algorithms can be understood more easily. Since these algorithms are recursive, a visualization of the call stack is also provided [19]. A screenshot of the Preorder tree traversal visualization can be seen in Figure 2.

In the interactive exercises, the user's task is to highlight the nodes of the displayed tree in the correct order according to the chosen tree traversal algorithm. The trees in the exercises are generated pseudo randomly with a maximum height of five.

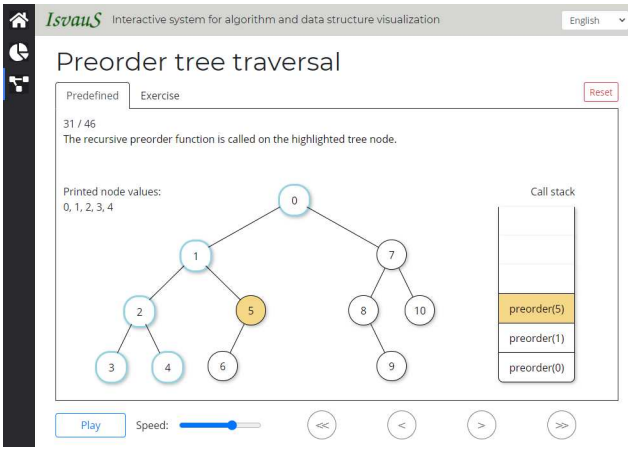


Figure 2. Preorder tree traversal visualization

6.4 Graph algorithms

The *depth first search* (DFS) and *breadth first search* (BFS) algorithms also consist of a predefined part and an exercise. The recursive DFS algorithm visualization makes use of the call stack visualization to help better explain the algorithm. In the BFS algorithm, a queue is used to help keep track of the order of the nodes, which is also included in the visualization. Similarly to the tree traversal algorithms, the

predefined visualizations of both graph algorithms also use a graph with the same structure to help understand the difference between the two algorithms [20].

While solving the exercise, the user must highlight the nodes of the displayed graph in the correct order according to the selected algorithm. The graphs generated in the exercises have eight nodes and the edges between the nodes are directed and generated randomly, while ensuring that there are no isolated nodes in the graph. However, the graph can contain cycles.

6.5 Bubble sort algorithm

For the bubble sort algorithm all three types of visualizations are supported. The predefined visualization shows the algorithm in detail on a randomly generated array of seven elements [21], as seen in Figure 3. The steps of the visualization are generated dynamically for the randomly generated array. The interactive visualization allows the user to enter custom values for the array separated by a space character. The steps of the visualization are then generated for the array defined by the user. In the exercise provided for this algorithm, the user is required to highlight the pairs of elements that are to be swapped according to the bubble sort algorithm in the correct order.

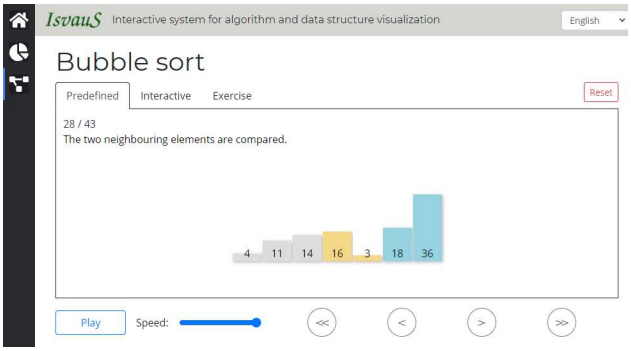


Figure 3. Bubble sort algorithm visualization

7 Application evaluation

In comparison with the other applications analyzed in this work, the designed system covers most of the functionalities which are provided by the other applications. These functionalities include stepping the visualization forward and backward, pausing and playing the visualization, changing the speed at which the visualization is played, and interacting with the data structures being visualized. The new application also provides better availability than the systems previously used in teaching the subject Data Structures and Algorithms. The shortcoming of the newly developed application is the number of provided visualizations, which is less than the number of visualizations included in the previously used systems. Although the new application provides less visualizations, new visualizations can be implemented and added to the application quickly, due to the way the application is implemented.

Evaluation of the application was conducted also by users via a survey. This survey was filled out by 23 students aged between 18 and 25, most of whom took part in the subject Data Structures and Algorithms. The respondents gave the application an overall rating of 4.6 out of 5. The participants were evaluating the application based on the initial requirements. Visualizations of the linked list, stack, and queue data structures and the three tree traversal algorithms were included in the application at the time when the user tests were conducted. The survey was focused on the overall design and the three types of visualizations included in the application. All three types of visualizations were evaluated as easy to understand and helpful in the learning process. The main part of the survey consisted of ten statements, where the respondents needed to mark the extent they agreed with the statement (a score of 1 means they strongly disagreed, a score of 5 means they strongly agreed). The average results of this part of the survey are displayed in Table 1 alongside with the statements.

After filling out the main part of the survey the users were asked to give feedback on which of the provided visualizations were the most helpful for them. The results of this question are displayed in Figure 4. The most helpful were the tree traversal algorithm visualizations

according to the respondents. In the last part of the survey, the respondents had an opportunity to give feedback on the application and suggest new visualizations or functionalities that would be beneficial to implement. The respondents requested the addition of new visualizations, such as the heap data structure, AVL tree, graph algorithms, sorting algorithms, and more. Visualizations of the graph algorithms DFS and BSF and also the bubble sort algorithm were added to the application based on these requests. The most requested functionality was the autoplay feature, which was also implemented in the application after the review of the survey, and responsive design for mobile devices.

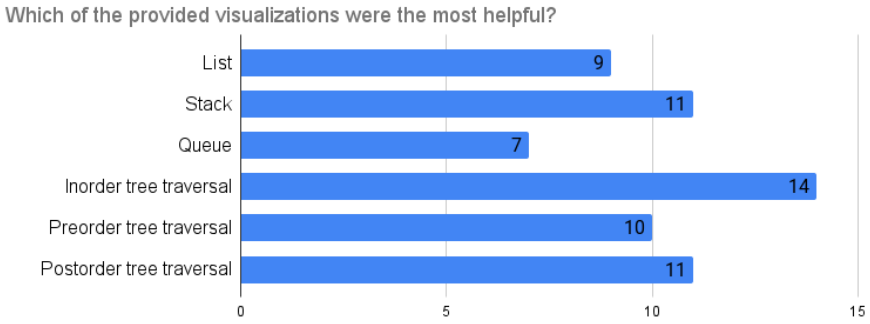


Figure 4. The most useful visualizations according to the survey

8 Conclusion

The result of this work is a web application implemented using the NuxtJS framework and the JSAV library for visualizing data structures and algorithms. The application is designed to be easily extensible with new visualizations without the need to modify its structure. The three supported visualization types allow users to learn from predefined visualizations, interact with them and then test their knowledge via the provided interactive exercises.

The resulting application meets the initial requirements defined in

Table 1. Survey results

Statement	Avg. response
The application is intuitive and easy to use.	4.54
The application is easy to navigate.	4.58
The layout of the application is appropriate for displaying visualizations alongside with pseudocode and descriptions.	4.25
The application helped you to understand how the visualized algorithms and data structures work.	4.58
The predefined visualizations are easy to follow.	4.5
The displayed notes for each step of the predefined visualizations are helpful.	4.38
Having pseudocode alongside the visualizations makes it easier to understand the algorithm.	4.54
Having interactive visualizations in the application is helpful.	4.5
The controls for the interactive visualizations are intuitive and easy to use.	4.42
The prepared exercises are easy to understand and complete.	4.5

this work. By implementing the system in the form of a web application and deploying it to a web server the availability requirement is satisfied. This also ensures that when a new version of the application is released, all users have access to it without the need to download a new executable file or install the new version of the application on

their device.

The oveverall design and layout of the application ensures that the usability requirement is also met as it was confirmed by users during tests.

Descriptions for the included data structures and algorithms are provided in the application. The predefined visualizations contain explanations for each step of the visualized algorithm or data structure, to help better understand them. The visualization of pseudocode in which the currently executed line is always highlighted also helps in the learning process.

Interactive visualizations allow the user to modify the visualized data structures directly, or define custom values for the data structures in the visualization. Each visualization has its own set of controls by which the user can interact with it.

The prepared exercises ensure that the users have a way to test their knowledge. The data structures in the exercises are generated pseudorandomly to ensure the students can't memorize the solutions of the exercises.

One of the main features of the resulting application is that it's easily extensible. Satisfying this requirement affected the design of the application greatly. Adding a new visualization to the application can be done in three steps: extending the configuration file with information about the new visualization; extending the common abstract class and implementing some or all types of visualizations using the JSAV library; finally, providing translations for the newly added translation keys. Everything else (like menu entry and link to the new visualization) will be generated automatically. An example of adding a new visualization to the application is provided in the System manual included in the related work [5].

The application provides visualizations of the basic data structures such as the linked list, stack, queue. These visualizations are made up of predefined and interactive parts. Visualizations for tree traversal algorithms, graph algorithms, and a sorting algorithm are also provided. Exercises are also provided for all the mentioned algorithms.

As of now, the application includes visualizations for only the basic

data structures and algorithms. In the future, we would like to extend the application with new visualizations selected from the topics covered by the Data Structures and Algorithms subject. Implementation of the visualizations suggested by the users should also be considered. New features requested by the users can also be considered for implementation in the future. From the teachers point of view, implementation of group testing of the students via interactive exercises, with an overview of the student's scores might also be beneficial.

References

[1] V. Karavirta and C. A. Shaffer, "Creating engaging online learning material with the jsav javascript algorithm visualization library," *IEEE Transactions on Learning Technologies*, vol. 9, no. 2, pp. 171–183, 2016.

[2] G. Rößling, "A first set of design patterns for algorithm animation," *Electronic Notes in Theoretical Computer Science*, vol. 224, pp. 67–76, 2009, proceedings of the Fifth Program Visualization Workshop (PVW 2008). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1571066108005136

[3] D. Dicheva and A. Hodge, "Active learning through game play in a data structures course," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 834–839. [Online]. Available: https://doi.org/10.1145/3159450.3159605

[4] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. A. Velázquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," *SIGCSE Bull.*, vol. 35, no. 2, p. 131–152, Jun. 2002. [Online]. Available: https://doi.org/10.1145/782941.782998

[5] P. Perháč, "Interactive system for algorithm and data structure visualization (in Slovak)," Master's thesis, Department of Com-

puters and Informatics Faculty of Electrical Engineering and Informatics Technical University of Košice, 4 2021.

[6] G. Rößling, M. Mihaylov, and J. Saltmarsh, "Animalsense: Combining automated exercise evaluations with algorithm animations," in *ITiCSE'11 - Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science*, 01 2011, pp. 298–302.

[7] S. Šimoňák, "Algorithm visualizations as a way of increasing the quality in computer science education," in *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2016, pp. 153–157.

[8] S. Šimoňák, "Using algorithm visualizations in computer science education," *Open Computer Science*, vol. 4, 10 2014.

[9] S. Šimoňák, "Increasing the engagement level in algorithms and data structures course by driving algorithm visualizations," *Informatica*, vol. 44, 09 2020.

[10] M. Benej and S. Šimoňák, "Algomaster platform extension for improved usability," *Journal of Electrical and Electronics Engineering*, vol. 10, no. 1, pp. 27–30, 2017.

[11] S. Šimoňák and M. Benej, "Visualizing algorithms and data structures using the algomaster platform," *Journal of Information, Control and Management Systems*, vol. 12, no. 2, pp. 189–201, 2014.

[12] D. Galles, "Data structure visualizations," Library homepage, 2011. [Online]. Available: https://www.cs.usfca.edu/~galles/visualization/about.html

[13] S. Khuri, "A user-centered approach for designing algorithm visualizations," *Informatik/Informatique, Special Issue on Visualization of Software*, 2001.

[14] V. Papaspirou, L. Maglaras, and M. A. Ferrag, "A tutorial on cross site scripting attack - defense," 10.20944/preprints202012.0063.v1, 12 2020.

[15] M. Azmoodeh, *Abstract Data Types and Algorithms*, ser. Macmillan Computer Science Series. Macmillan Education UK, 1990.

[16] K. Mehlhorn and P. Sanders, *Algorithms and Data Structures: The*

Basic Toolbox. Springer, 2008.

[17] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data structures and algorithms*. Addison-Wesley, 1985.

[18] D. P. Mehta and e. S. Sahni, *Handbook of data structures and applications*, 1st ed., ser. Chapman & Hall/CRC computer and information science series. Chapman & Hall/CRC, 2004.

[19] A. Cisneros, "Visualizing recursion," 6 2020. [Online]. Available: https://medium.com/swlh/visualizing-recursion-6a81d50d6c41

[20] K. Mocinecova and W. Steingartner, "Software support for visualizing of the graph algorithms in a novel approach in educating of young it experts," *IPSI BGD TRANSACTIONS ON INTERNET RESEARCH*, vol. 16, pp. 14–23, 7 2020.

[21] P. Kavdikar, "Comparative study of sorting algorithms," 04 2021. [Online]. Available: https://www.researchgate.net/publication/350959496_Comparative_study_of_sorting_algorithms

Patrik Perháč, Slavomír Šimoňák,

Received September 17, 2021
Accepted October 15, 2021

Patrik Perháč
Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic
E–mail: perhac.patrik97@gmail.com

Slavomír Šimoňák
Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic
E–mail: slavomir.simonak@tuke.sk

Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

Revisiting the Role of Classical Readability Formulae Parameters in Complex Word Identification (Part 2)*

Gayatri Venugopal, Dhanya Pramod †,
Jatinderkumar R. Saini ‡

Abstract

Accessibility of text is an attribute that deserves the attention of researchers and content creators. This study is an attempt to determine the lexical features that play a key role in identifying complex words in Hindi text. As the first step, we studied the parameters used in readability metrics in different languages and tested their importance on classifiers built on datasets created with the help of a user study. In part of the study, we reported the results of two different approaches used to label a word as complex. In this part, we compare the previous results with the results obtained from a third labeling approach. We found satisfactory evidence for certain parameters and also observed a new parameter that could be used while devising readability metrics for Hindi.

Keywords: complex word identification, readability, hindi, binary classification, natural language processing.

MSC 2010: 68R10, 68Q25, 05C35, 05C05.

* This work was supported by Symbiosis Centre for Research and Innovation, Symbiosis International (Deemed University), grant number 1591

† Equal contribution

‡ Equal contribution

1 Introduction

Text simplification refers to the process of modifying a text in such a manner that it becomes more comprehensible to reader with no loss of information. The words that the content creators use may not always be understood by the consumers of the content. They may use words that they are comfortable with or that would distinguish them from the others [1]. Readable texts would not only help readers who are new to the language, but could also help readers with reading disabilities such as dyslexia [2], aphasia [3], and also readers with a poor level of literacy, and children [4]. Various readability formulae have been devised by researchers that would help identify the complexity level of a given text. We can find the usage of these formulae even today in various studies [5]–[7]. This study is focused on identifying the similarities between the lexical parameters used in the readability formulae, and the features deemed to be significant in determining a word in Hindi to be complex or not. Hindi, the official language of India, is ranked high in the list of languages spoken by first language speakers in the world [8]. Although the readability formulae have been devised for a multitude of languages, most of the formulae are centered around English. Besides readability formulae, we have also taken into account the lexical characteristics used in the creation of word lists that contain words that are considered to be simple.

We began our study by identifying the quantifiable characteristics of words that were used as parameters in commonly used readability formulae. We focused our study on the most common characteristics and used them as features of words in different classification models. The study focuses on the lexical complexity of a word. We used two approaches to identify the important features, the outcomes of which prominently highlighted the importance of the frequency of a word and the irrelevance of the word's length. This result contradicts the importance given to the length of a word by various readability formulae. We also observed that the number of hyponyms of a word, which has not been covered by any formula, is a key predictor of the complexity of a word.

This study is the second part of a two-part study that was conducted to determine the significance of parameters used in readability formulae, in the identification of complex words in Hindi text. In the earlier study, we used two approaches to create the dataset. The first approach consisted of deeming a word as complex if at least two users rated the word as complex, whereas a word was deemed complex using the second approach if the majority of the raters rated the word as complex. We discarded the first approach based on the results discussed in Part 1 of this study.

In this paper, we discuss another approach used to label a word as complex and report a comparison of the results obtained using this approach and the previous approach discussed in Part 1 of the study.

2 Lexical Parameters

This section gives an overview of the lexical parameters used in popular readability formulae and word-lists.

One of the earliest studies on readability in English was done by L.A. Sherman, an English professor, who claimed that the length of a sentence and the concreteness of a word impact the comprehension of the reader [9]. Although we could not find any word list or formula proposed by Sherman, there exist numerous studies that have proposed various parameters to test the readability of a given text, most of them written in English.

Rubakin was a prominent Russian writer who published a list of 1500 words in Russian in 1889, that he claimed to be easy to understand. According to Rubakin, words that were not known to the users and sentences that have many words act as hindrances to comprehension [10]. One of the popular word lists is the Teacher's Word Book, that was created in 1921 [11], wherein the author focussed on including words with a short length and high frequency. More words were added to this list in 1944, making it a list of 30,000 words [10].

Readability of text encompasses various aspects such as content, style, format and structure [12]. Our study is focused on the lexical parameters used in readability formulae. The Flesch Reading Ease

formula takes into account the number of syllables, number of words and number of sentences [13]. Researchers have adapted this formula to languages other than English by modifying the values of coefficients [14]–[16]. The number of syllables has been considered to be a major factor in various other readability measures such as the Gunning Fog index [17], the readability system created by Edward Fry [18], SMOG [19], and the Flesch-Kincaid Reading Ease formula [20]. Other common parameters used in readability formulae are word length [21], [22] and frequency [14], [23], [24].

In [25] the authors proposed a readability formula that focused on the number of words in a sentence and the number of sentences in 300 words in a given text. In [26] the authors devised a formula for determining the readability of Vietnamese text which took into account the average sentence length in characters, the average of word length in terms of the number of characters, and the percentage of difficult words that was calculated using a list of easy words.

The parameters used in readability formulae that targeted the Hindi language were length, number of consonants, and number of consonant conjuncts [27], [28]. Therefore our study aimed to analyse these lexical parameters and their importance in complex word classification models.

3 Methodology

We conducted a user study consisting of 50 native and 50 non-native speakers of Hindi in the age group of 18 to 30 years. The user study involved two steps – annotating complex words in a set of 100 sentences, and ranking the annotated word in comparison with its synonyms, using a Liker scale, where 1 indicated very complex and 5 indicated very simple.

3.1 Labeling Method

In order to build a classifier, our next step was to label a word as complex or simple. We used three approaches, two of which have been discussed in Part 1 of this study, which also contains the details of the

sources of feature values used to build the model. As part of our third approach, we labeled a word as complex if the average rating assigned to the word was less than or equal to 3.

In this approach, which we believe is more bias-free as compared to the first approach, only words that were rated by at least two participants were considered. The dataset thus generated consisted of 7326 records out of which 2958 records were labeled as complex and 4368 records were labeled as simple. The dataset size is less as compared to the dataset generated in approach 1 because we did not consider words that were ranked by only one person (in order to avoid bias). The training-test proportion was 70:30. The training set consisted of 5129 records out of which 40.38% were labeled 1 and 59.62% were labeled 0. Since this was not an undesired proportion, resampling techniques were not used.

3.2 Feature Evaluation Methods

As discussed in Part 1 of the study, we used the traditional as well as ensemble classification algorithms with k-fold cross-validation with five splits. The algorithms used were decision tree, support vector classifier, nearest centroid classifier, random forest, extra trees, Ada boost, gradient boosting and XG boost. We chose sense-normalised values of length, number of syllables, frequency of the lemma of the word, number of consonants, number of vowels, number of consonant conjuncts, number of synsets, number of synonyms, number of hypernyms and the number of hyponyms as the features used to build the classifiers.

In order to evaluate the features, we used two methods as shown in Figure 1 and Figure 2, respectively.

In the first method, eight models were built and each feature's importance value was calculated using permutation feature importance and exhaustive feature selection. The importance value generated from all the models are aggregated to obtain one score for each feature. Accuracy and Macro-F1 scores were used as the metrics to evaluate the performance.

In the second method, we tuned the models using random search hyperparameter tuning based on Receiver Operating Characteristic

(ROC) scores and built a soft voting classifier as our final model. The feature importance values were calculated for this model.

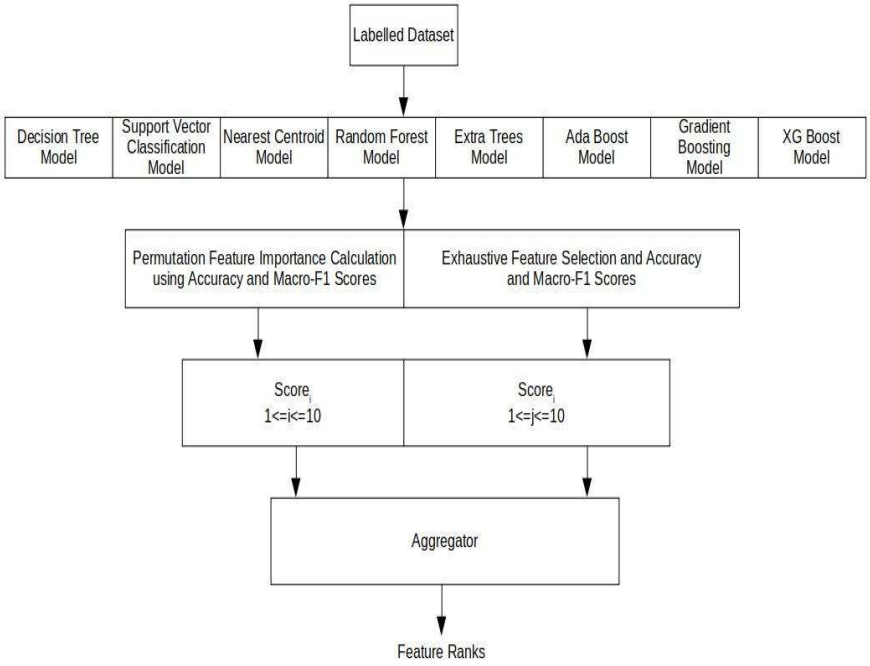


Figure 1. Method 1

4 Results and Discussion

Besides the features mentioned in the previous section, we included four other lexical features, which were synonyms, number of synsets, number of hyponyms and number of hypernyms to build the models. We calculated the feature importance scores using accuracy and macro-F1 scores, and assigned the highest rank, e.g. 1, to the feature with the highest average. With regard to exhaustive feature selection, the value 1 was assigned to a feature if it was present in a feature subset for a model, and the value 0 was assigned to it if it was not present in

the feature subset. The results can be seen in Table 1. The ROC curve for the models can be seen in Figure 3.

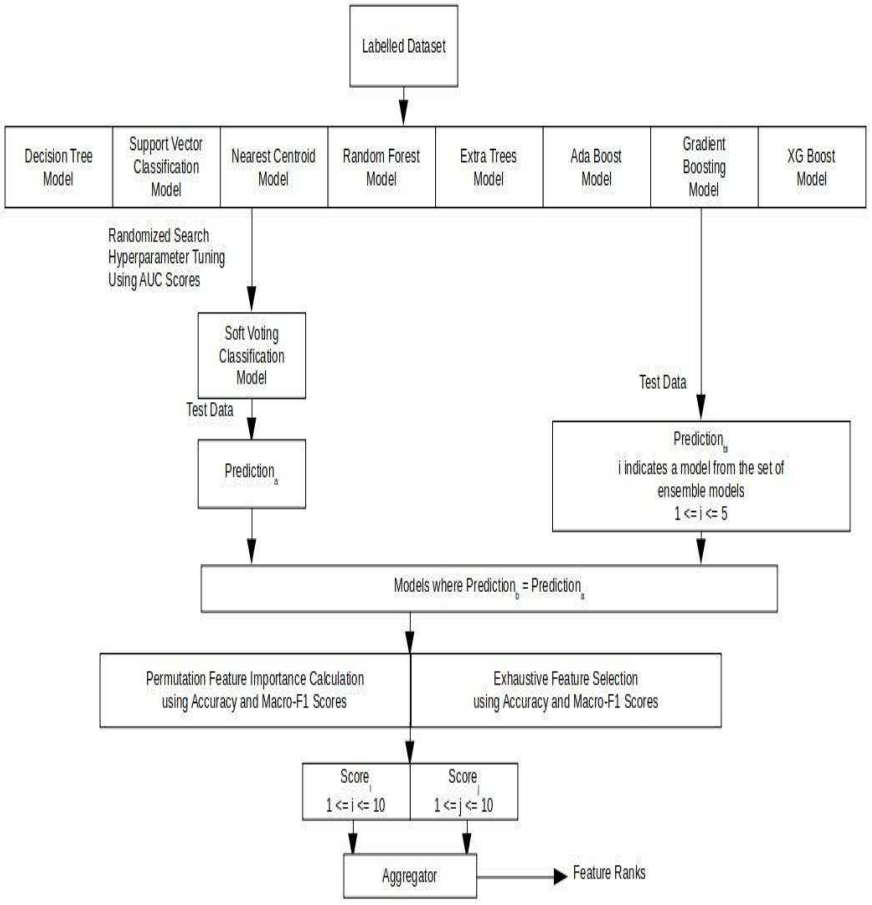


Figure 2. Method 2

Table 1. Feature importance values for each feature based on accuracy and macro-F1 scores for all the models

Feature	**Permutation Feature Importance**		**Exhaustive Feature Selection**	
	Accuracy	**Macro-F1**	**Accuracy**	**Macro-F1**
n_synonyms	5.125	5.75	0	0
n_synsets	4.625	4.625	0	0
frequency	10	10	0.625	0.625
n_hyponyms	8.25	8	0	0
n_syllables	3	2.75	0	0
n_hypernyms	4.75	5	0	0
length	4.125	4.125	0	0
n_consonants	6.5	6.5	0	0
n_vowels	3.75	3.5	0	0
n_consonantconjuncts	4.875	4.75	0	0
Mean	5.5	5.5	0.0625	0.0625
Median	4.8125	4.875	0	0
Standard Deviation	2.157	2.1755	0.1976	0.1976

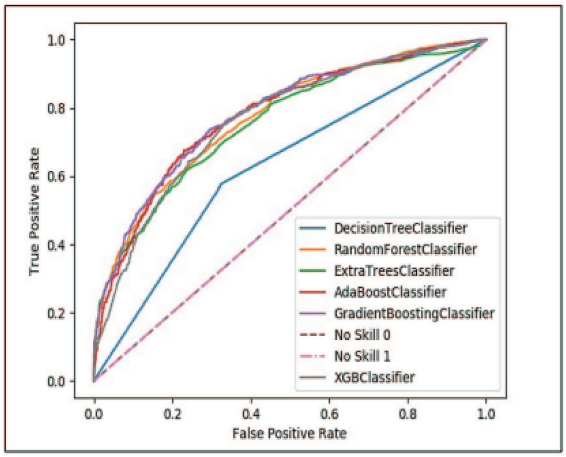


Figure 3. ROC curve for all the tree based models

We then took an aggregate of the values for each method of feature importance calculation. The values can be seen in Table 2.

Table 2. The aggregate of the feature importance values for each feature for all the models

Feature	**Importance Value Based On**		**Feature Importance Value**
	Accuracy	**Macro-F1**	
n_synonyms	5.125	5.75	5.4375
n_synsets	4.625	4.625	4.625
frequency	10.625	10.625	10.625
n_hyponyms	8.25	8	8.125
n_syllables	3	2.75	2.875
n_hypernyms	4.75	5	4.875
length	4.125	4.125	4.125
n_consonants	6.5	6.5	6.5
n_vowels	3.75	3.5	3.625
n_consonantconjuncts	4.875	4.75	4.8125

The second method encompasses analysing the importance values obtained for all the tree-based models. These values were compared against the baselines (ALL 0 and ALL 1). The results can be seen in Table 3.

As was the case with Approach 2 mentioned in Part 1, ensemble classifiers showcase a better performance than the traditional decision tree classifier. We then built a soft voting classifier and selected the label with the maximum vote, among the labels generated using the ensemble classifiers and the tuned ensemble classifiers. The Area under the ROC Curve (AUC) scores can be seen in Table 4.

In order to identify the significant features, we considered each prediction and used the classification models which made the correct prediction. The feature importance values of these models were calculated by adopting the strategy implemented in Method 1. The ranks of the features calculated based on their importance values, can be seen in Table 5.

Table 3. Analysis of Importance Values

Classifier	**Macro-F1**	**Accuracy**
Baseline (ALL 0)	0.187	0.596
Baseline (ALL 1)	0.144	0.404
Support Vector	0.289	0.618
Nearest Centroid	0.307	0.611
Extra Trees	0.338	0.695
Random Forest	0.348	0.714
XGB	0.343	0.703
Gradient Boosting	0.353	0.719
Ada Boost	0.353	0.719
Decision Tree	0.319	0.652

Table 4. AUC Scores

Model	**AUC Score**
Ada	0.776
Tuned Ada	0.781
Extra Trees	0.760
Tuned Extra Trees	0.762
Gradient Boosting	0.783
Tuned Gradient Boosting	0.755
Random Forest	0.770
Tuned Random Forest	0.785
XGBoost	0.785
Tuned XGBoost	0.782
Soft Voting	0.790

The ranks obtained using this approach are identical to the ranks obtained using Approach 2 in Part 1 of the study. Therefore we have strong evidence that frequency is a major predictor of the complexity of a word. This aligns with the results reported by studies conducted on other non-Indian languages.

Table 5. Feature Importance Values and Ranks

Feature	**Feature Importance Value**	**Rank**
frequency	1.156	1
n_hyponyms	1.04	2
n_syllables	1.036	3
n_vowels	1.033	4
n_consonants	1.027	5
n_synsets	1.023	6
n_hypernyms	1.022	7
length	1.017	8
n_synonyms	1.013	9
n_consonantconjuncts	1.002	10

5 Conclusion and Future Scope

The goal of the study was to ascertain the significance of lexical parameters used in readability measures, in complex word identification in Hindi text. We used two methods and three approaches to approach the problem. The feature importance values were calculated using accuracy and macro-F1 scores. A soft voting classifier was used as it performed better than the individual models involved in the study.

Through this study, we reinstated the importance of the role that frequency plays in determining the complexity of a word. Many readability measures used word length as one of the parameters. However, we found from both our approaches, Approach 2 and 3, that length of a word is not a significant factor. The readability measures also focused on the number of syllables, which was proven to be an important predictor of word complexity. We suggest the use of the number of hyponyms of a word as a parameter in a readability measure for Hindi text as this has proven to be an important feature in a complex word classifier.

This work could be refined by using a different approach for complex

word labeling and by tuning the models using grid search hyperparameter tuning. Researchers may use this word to create a readability metric for Hindi text focusing on lexical attributes of the text.

6 Acknowledgements

This study was sponsored by Symbiosis Centre for Research and Innovation, Symbiosis International (Deemed University) and has approval from the Independent Ethics Committee, Symbiosis International (Deemed University) in July 2019. We are grateful to the Linguistic Data Consortium for Indian Languages and IIT Bombay for providing the corpora that were used in the study, and also the Stanford NLP group for releasing the stanfordnlp Python package that was used to retrieve the lemmas of words in Hindi [29]. We thank the reviewers for their valuable comments that helped improve the paper.

References

[1] E. L. Thorndike, "The psychology of semantics," *The American journal of psychology*, vol. 59, no. 4, pp. 613–632, 1946.

[2] L. Rello, R. Baeza-Yates, S. Bott, and H. Saggion, "Simplify or help?" in *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility - W4A '13*. ACM Press, 2013. [Online]. Available: https://doi.org/10.1145/2461121.2461126

[3] J. Carroll, G. Minnen, Y. Canning, S. Devlin, and J. Tait, "Practical simplification of english newspaper text to assist aphasic readers," in *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, 1998, pp. 7–10.

[4] J. De Belder and M.-F. Moens, "Text simplification for children," in *Prroceedings of the SIGIR workshop on accessible search systems*. ACM; New York, 2010, pp. 19–26.

[5] M. A. Kugar, A. C. Cohen, W. Wooden, S. S. Tholpady, and M. W. Chu, "The readability of psychosocial wellness patient resources: improving surgical outcomes," *Journal of Surgical Research*, vol. 218, pp. 43–48, 2017.

[6] J. C. Brewer, "Measuring text readability using reading level," in *Encyclopedia of Information Science and Technology, Fourth Edition*. IGI Global, 2018, pp. 1499–1507. [Online]. Available: https://doi.org/10.4018/978-1-5225-2255-3.ch129

[7] J. M. Marsh, T. D. Dobbs, and H. A. Hutchings, "The readability of online health resources for phenylketonuria," *Journal of Community Genetics*, vol. 11, no. 4, pp. 451–459, mar 2020. [Online]. Available: https://doi.org/10.1007/s12687-020-00461-9

[8] "Languages of the world." [Online]. Available: www.ethnologue.com

[9] L. A. Sherman, *Analytics of literature: A manual for the objective study of English prose and poetry*. Ginn, 1893.

[10] E. L. Thorndike and I. Lorge, "The teacher's word book of 30,000 words." 1944.

[11] E. L. Thorndike, "The teacher's word book," 1921.

[12] W. S. Gray and B. E. Leary, "What makes a book readable." 1935.

[13] R. Flesch, "A new readability yardstick." *Journal of Applied Psychology*, vol. 32, no. 3, pp. 221–233, 1948. [Online]. Available: https://doi.org/10.1037/h0057532

[14] S. Štajner and H. Saggion, "Readability indices for automatic evaluation of text simplification systems: A feasibility study for spanish," in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, 2013, pp. 374–382.

[15] W. Douma *et al.*, "Readability of dutch farm papers: a discussion and application of readability-formulas." *Readability of Dutch farm papers: a discussion and application of readability-formulas.*, no. 17, 1960.

[16] R. Brouwer, "Onderzoek naar de leesmoeilijkheden van nederlands proza," *Pedagogische studiën*, vol. 40, pp. 454–464, 1963.

[17] R. Gunning *et al.*, "Technique of clear writing," 1952.

[18] E. Fry, "A readability formula that saves time," *Journal of reading*, vol. 11, no. 7, pp. 513–578, 1968.

[19] G. H. Mc Laughlin, "Smog grading-a new readability formula,"

Journal of reading, vol. 12, no. 8, pp. 639–646, 1969.

[20] J. P. Kincaid, J. Fishburne, R. R. P., C. R. L., and B. S., "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," Tech. Rep., feb 1975. [Online]. Available: https://doi.org/10.21236/ada006655

[21] R. Senter and E. A. Smith, "Automated readability index," CINCINNATI UNIV OH, Tech. Rep., 1967.

[22] M. Coleman and T. L. Liau, "A computer readability formula designed for machine scoring." *Journal of Applied Psychology*, vol. 60, no. 2, pp. 283–284, 1975. [Online]. Available: https://doi.org/10.1037/h0076540

[23] G. R. Weir and C. Ritchie, "Estimating readability with the strathclyde readability measure," in *ICT in the Analysis, Teaching and Learning of Languages, Preprints of the ICTATLL Workshop 2006*, 2006, pp. 25–32.

[24] A. Anula, "Tipos de textos, complejidad lingüıstica y facilicitación lectora," in *Actas del Sexto Congreso de Hispanistas de Asia*, 2007, pp. 45–61.

[25] N. Hazawawi, M. Zakaria, and S. Hisham, "Formulating an algorithm to detect readability level of malay texts," *Proceedings of Mechanical Engineering Research Day 2017*, vol. 2017, pp. 77–78, 2017.

[26] A.-V. Luong, D. Nguyen, and D. Dinh, "A new formula for vietnamese text readability assessment," in *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, nov 2018. [Online]. Available: https://doi.org/10.1109/kse.2018.8573379

[27] M. Sinha, S. Sharma, T. Dasgupta, and A. Basu, "New readability measures for bangla and hindi texts," in *Proceedings of COLING 2012: Posters*, 2012, pp. 1141–1150.

[28] M. Sinha, T. Dasgupta, and A. Basu, "Text readability in hindi: A comparative study of feature performances using support vectors,"

in *Proceedings of the 11th International Conference on Natural Language Processing*, 2014, pp. 223–231.

[29] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, "Universal dependency parsing from scratch," *arXiv preprint arXiv:1901.10457*, 2019.

Gayatri Venugopal, Dhanya Pramod,
Jatinderkumar Saini

Received May 27, 2021
Accepted September 9, 2021

Gayatri Venugopal
Symbiosis Institute of Computer Studies and Research (SICSR)
Symbiosis International (Deemed University) (SIU),
Model Colony, Pune, Maharashtra, India
Phone: +91-9665856569
E-mail: gayatri.venugopal@sicsr.ac.in

Dhanya Pramod
Symbiosis Centre for Information Technology (SCIT)
Symbiosis International (Deemed University) (SIU),
Hinjewadi, Pune, Maharashtra, India
E–mail: dhanya@scit.edu

Jatinderkumar R. Saini
Symbiosis Institute of Computer Studies and Research (SICSR)
Symbiosis International (Deemed University) (SIU),
Model Colony, Pune, Maharashtra, India
E-mail: saini_expert@yahoo.com

Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

The response time analysis of queuing model in cloud computing environment*

Ali Madankan

Abstract

In this paper, the queuing theory is used to model cloud computing environment. The aim of this job is to analyze the response time as a measure of the Quality of Service (QoS) of computer services. In the cloud computing, multi resources need to be allocated simultaneously to multiple customers. When a customer requests for a service, if servers are busy, the requested job enters into the waiting line until a server completes its service. So, this may lead to a bottleneck in the network. By modeling cloud platforms by a queuing network, it can be easy to determine and to measure the QoS. The arrival rate and the service rate of processing servers are two main parameters that can affect the performance of the model; so, they are used to analyze the performance of the model. This paper proposes a queuing model which is applied at multiple servers in order to analyze the response time and also to improve the network performance and QoS effectively in a cloud computing environment.

Keywords: Cloud Architecture, Queuing System, Response Time, Quality of Service.

1 Introduction

Cloud computing has been an emerging technology for provisioning computing resource and providing infrastructure of web applications in recent years. Cloud computing greatly lowers the threshold for deploying and maintaining web applications since it provides infrastructure

* This work was supported by University of Zabol, Grant Ref. No. UOZ-GR-0065538536

as a service (IaaS) and platform as a service (PaaS) for web applications [1]. Consequently, a number of web applications, particularly the web applications of medium and small enterprises, have been built into a cloud environment. Meanwhile, leading IT companies have established public commercial clouds as a new kind of investment. For example, Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make webscale computing easier for developers [2]. Google App Engine enables enterprises to build and host web applications on the same systems that power Google applications. App Engine offers fast development and deployment; simple administration, with no need to worry about hardware, patches or backups; and effortless scalability [3]. The cloud providers usually offer high performance, scalability, security, and high availability services. In summary, both of the numbers of cloud applications and providers have kept gradually increasing for a couple of years. As a result, performance managing and guaranteeing the Quality of Service (QoS) have been ones of the most important aspects of clouding computing [8].

The response time is an important characteristics of the system performance, and response time metrics may be a part of service level agreement. This paper considers the response time for services as a measure to evaluate the QoS. The response time is the total time that takes for a job to enter into the system and depart from it. It includes the waiting time before getting the service and the service time. In the architecture of cloud computing systems, it is considered that there are infinite computing resources available on-demand, which allows the resources to be expanded as needed. Response time is an important characteristics of system performance, and response time metrics may be a part of service level agreement.

[4] studied a popular model of cloud computing that consists of services that are entered and delivered from a service center that is accessible from anywhere in the world. The two main components in architecture are the front-end (the gateway) and the back-end (servers). The front-end provides the software components and interfaces that a customer connects to the servers through it. The back-end, which is

the actual cloud architecture, includes a function to manage the jobs queue, the servers and their virtual machines (VMs), and the database system. To avoid database inconsistencies, it is usually considered only one storage (i.e., database) server.

Our focus is on the back-end (which we call cloud). This component of the architecture contains the processing servers and a data service. We considered an entry point for the cloud to contact customers to receive their requested jobs and to send the result through the (Internet) network. Customers' requests are transmitted to the webserver running a service application [5], associated with an SLA (Service Layer Agreement)

Queuing theory is used to model the architecture and to manage the response time for the services. The queued model lets the cloud to be optimally scaled to guarantee the quality of service for the response time. Also this model considers the proper deployment and removal of virtual machines according to the system load. Authors in [6] used queuing model to model the process of entering into the cloud and to schedule and to serve incoming jobs. In that paper, the main problem is to allocate resources in the queuing systems as a general optimization problem for controlled Markov process with finite state space. Authors in [7] studied performance management on cloud using multi-priority queuing systems. In that work, the web applications are modeled as queues, and the virtual machines are modeled as service centers; they could be distinguished into two groups of priority classes with each class having its own arrival and service rates. In addition, the author in [8] showed how to compute performance bounds for the stochastic control policy of Markov decision processes with average cost criteria. In other words, he found bounds on performance with respect to an optimal policy.

An open Jackson network [9], [10] of M/M/m and M/M/1 queues is considered to model the cloud. We have two types of server, processing and database servers. We are also interested in the modeling QoS performance by scaling cloud platforms, leaving aside other issues such as cloud availability [11], energy consumption [12], variability [13] or reliability [14].

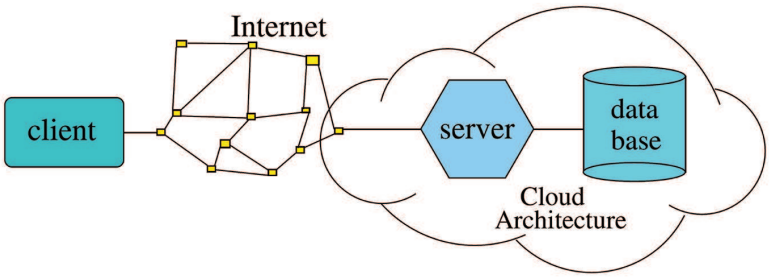


Figure 1. Cloud Computing Environment

2 Model

Let us consider a sequence of M/M/1 and M/M/N, a queuing system model which represents a cloud platform like the model in [6]. The first queue is the entry point that manages incoming jobs and sends them into a server of the second queueing system. Let us call it Input Server (IS). The IS is actually a load balancer. This server is a M/M/1 queue with an exponential arrival and service rates λ and L, respectively. To have steady state, we consider $\lambda < L$. Its aim is to find a server to deliver the job. In order to keep the system in the optimal situation, it uses a distribution algorithm depending on the averaged workload in each server.

The second queueing system, the M/M/N queueing system, has N individual servers, where these processing servers (PS_i), for $i = 1, ..., N$, can serve any incoming jobs. Let consider that each server can serve all accepted job by IS. Also let consider that each server, PS_i, has the same service rate μ, this is $\mu = \mu_i$, $i = 1, ..., m$.

Let us consider there is a database server (DS) in the model, where all PS_i are connected to it and can send their requests to it during the service in the cloud. This kind of access to the databases and directories between virtual machines (VMs) is very important. To avoid data inconsonance, let us consider to have only one database server in this model. Processing servers access to the database server with a

probability σ. DS has all files, dictionaries, databases, and secondary memories and provides these facilities for other servers in the cloud architecture. DS, itself, puts requests into an M/M/1 queue, where it assumes exponential arrival and service rates of $\sigma\gamma$ and D, respectively.

There is another server in this model that is called Outputer Server (OS). In the cloud architecture, jobs come from Client Server (CS); so, the cloud manager has to deliver the completed job to the CS over the network (say internet). The OS receives the responses from the PS_i and sends the responses to CS with an exponential parameter λ.

In the model, the response time (T), which is a very important measure in the cloud architecture, is calculated as:

$$T = T_{IS} + T_{PS} + T_{DS} + T_{OS} + T_{CS}, \tag{1}$$

where each of these terms are the total time that each servers spends to complete its service in this model. In the following, the terms of Eq.1 are calculated.

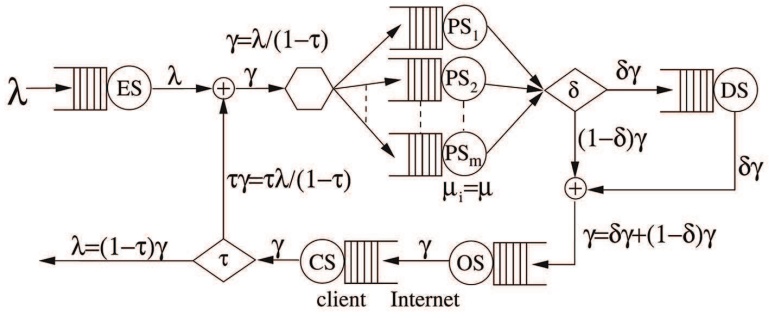


Figure 2. Queuing Model of Cloud Computing

2.1 Inputting Service Time T_{IS}

The Input Server is a load balancer; so, the parameter T_{IS} shows the spent time by Input Server to distribute incoming jobs. Since we modeled the input server (IS) as a M/M/1 queueing system, thus, the

response time for a M/M/1 queue is [16]:

$$T_{IS} = \frac{1}{L - \lambda}, \tag{2}$$

where λ and L are the arrival rate and the service rate of the IS, respectively.

2.2 Processing Server Time T_{PS}

The parameter T_{PS} shows the time spending by Process Servicing to process the job. Since there are N processing servers (PS) PS in the cloud architecture, and these PSs are modeled as an M/M/N queue, then the spent time by this kind of queue is as follows [15]:

$$T_{PS} = \frac{1}{\mu} + \frac{C(N, \rho)}{N\mu - \gamma}, \tag{3}$$

where γ is arrival rate and $\mu = \mu_i$, $i = 1, ..., N$, is service rates of each processing servers. In the (3), the term $C(m, \rho)$ gives the probability of a new coming job to the M/M/N queue. The formula of calculating of $C(m, \rho)$ is defined as [16]:

$$C(m, \rho) = \frac{\left(\frac{(mp)^m}{m!}\right)\left(\frac{1}{\rho}\right)}{\sum_{k=0}^{m-1}\left(\frac{(mp)^k}{k!}\right) + \left(\frac{(mp)^m}{m!}\right)\left(\frac{1}{\rho}\right)}, \tag{4}$$

where $\rho = \gamma/\mu$.

2.3 Database Server Time T_{DS}

In this model, there is a database server that with the probability of σ jobs needs some files from the database server. So, $\sigma\mu$ is the arrival rate to the DS. Also, if the parameter T_{DS} is the total time that Database Server spends to respond, and since the DS is modeled as an M/M/1 queue, so:

$$T_{DS} = \frac{1}{D - \sigma\mu}. \tag{5}$$

2.4 Output Server Time T_{OS}

When process of a job is done by processing servers, Output Server sends the job to the customer. This function takes time that we use the parameter T_{OS} to show the spent time by Output Server (OS). The OS is considered as an M/M/1 queue too. Since this server is connected to the customer over the internet, if F is the average size of jobs and O is the average bandwidth speed of the OS, then the service rate of this part of architecture is O/F. So, the response time of this server is:

$$T_{OS} = \frac{F}{O - \gamma F}. \tag{6}$$

2.5 Client Server Time T_{CS}

The parameter T_{CS} shows the spent time by Client Server to respond. The OS sends the data to CS through Internet and CS receives it. The CS is modeled as an M/M/1 queue, where if C is the average bandwidth speed of the CS, then the service rate is defined as C/F. As a consequence, the response time of CS is:

$$T_{CS} = \frac{F}{C - \gamma F}. \tag{7}$$

3 Result

Regarding the presented model, it is obvious that several parameters are involved in the response time. So, in this section, an analysis of how the response time varies by changing in some of these parameters in the model, is presented. The aim of this paper is to study the expected behaviour of the model when system configurations are modified. In the following subsection, we simulated the model by using the Sage mathematical software to obtain the results.

3.1 Performance Test

To evaluate the presented methodology, Sage mathematical software is used to run this model. We purpose to show the effects of the parameters on total response time (T) by modifying the parameters. We compare the results with real data from a real cloud system with regard to the following parameters:

λ is the arrival rate. Customers send their job requests to the system; so, λ is the average number of incoming jobs to the system. In other meaning, $1/\lambda$ is the average time between consequently incoming jobs.

μ is the service rate, which means $1/\mu$ is the mean service time. For ease of computation, we considered the same service rate for all PS_i servers. In order to analyze the system, we consider the system is not the steady-state, which means the total service rate of the system must always be greater than the incoming rate λ; and, as a result, the system is stable. Also we consider the same service rate for all servers (ES, PS_i, $i = 1, ..., N$, DS, OS, and CS. So, we have $\mu = L = D = O/F = C/F$.

δ is the database access probability. As the model is a web-based system, some incoming jobs need to access the DS and some do not. So, we consider this probability that an incoming job needs to access the DS.

N is the number of PSs that serve requests. Varying in this parameter shows how adding or removing servers affects the system.

F is the average file size. The system sends the result to the CS by files. Files have different sizes which depend on the block size of the web application. This parameter is considered to be the mean size of the files which is not greater than 1MB in most cases.

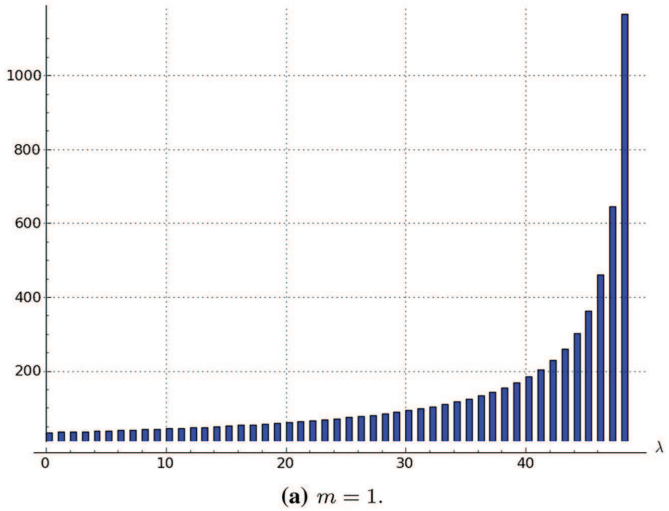
O is the server bandwidth.

C is the client bandwidth. Each client usually has different speed of receiving data from the system and it is usually out of our control.

3.1.1 Response time

First, we tested the response time variation regarding the number of processing servers. Here, in Fig.3, you can see the difference between

the existing one or two processing servers. In Fig.3, it is obvious that increasing the number of processing server has a great effect on decreasing the total response time. By our considerations, the response time becomes fix when the number of servers increases and there is no significant differences when we add more than 5 processing servers. To explain why it happened, we use utilization of the PS_i server, defined as $\rho = \gamma/\mu$, where $\gamma = \lambda/(1-\tau)$. When we add a server to the processing servers, then the utilization rate of the M/M/m queue decreases; and, as a result, the response time becomes equal to the service time. So, this is why we have no improvement in the response time.

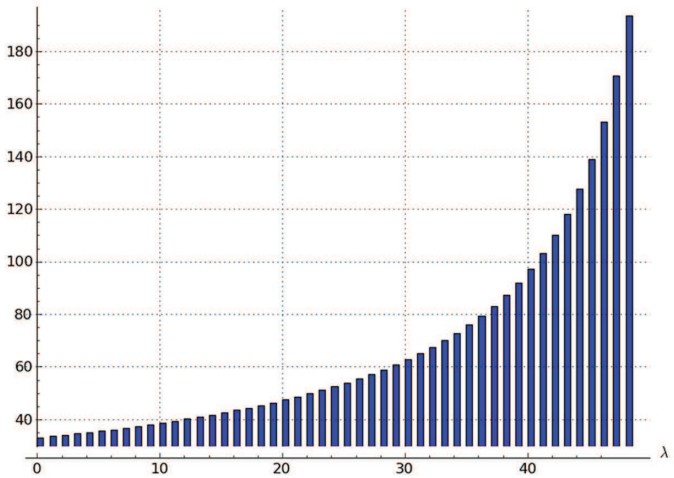


(a) $m = 1$.

Figure 3. The response time with respect to λ

Fig. 4 shows that the response time growth has a direct relation to the utilization of the server: it increases when the server is more utilized. When the utilization is less than 1 and close to 0, it is equal to the service time. When the utilization passes one, the rate of grow of the response time becomes exponential and it increases exponentially with respect to increases of utilization.

The relation between the Total Response Time regarding the mean



(b) $m = 2$.

Figure 4. The response time with respect to λ

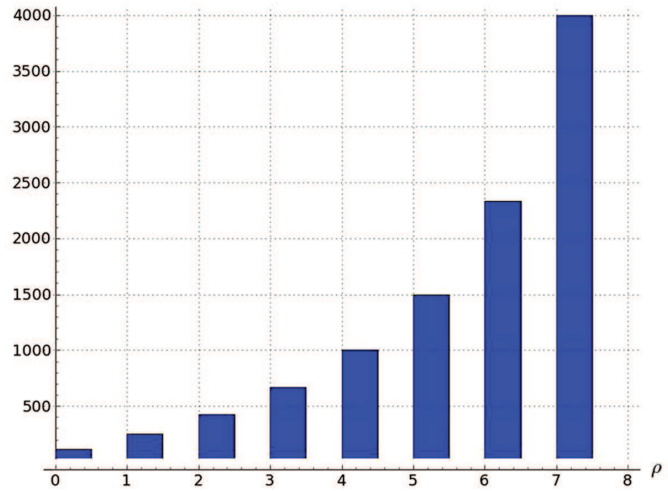


Figure 5. The response time with respect to ρ

file size (F), arrival rate (λ) and the output server bandwidth (O) is shown in Fig.5. It is obvious that the total response time has direct effect from the mean file size (F) and increases exponentially when the mean file size increases. Also it has inverse effect regarding the server bandwidth (O) and decreases when the server bandwidth (O) increases. It is because the OS needs a bigger bandwidth to send the large files to clients over the network. It means files are transferred at low speed when the server bandwidth decreases; and, as a result, the total response time (T) increases.

4 Conclusions and future work

In this paper, we considered a queuing model to design cloud computing architectures to guarantee the quality of service. We used the open Jackson's networks to model the cloud architecture which is the best fit model. Then we used analysis of the Jackson's networks to study the system performance and to analyze the response time.

First, in order to study QoS requirements, a sequence of two queuing systems, M/M/1 and M/M/N, is considered to model the cloud platform. This study showed us, regarding to have good QoS (have proper response time), where a bottleneck can occur in the system and which parameter can solve the problem. This model is an applying model and very useful to tune up the service performance and thus to guarantee the SLA contract between the client and the service provider.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2009-28, February 10, 2009. [Online]. Available:

http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[2] Amazon, "Amazon Elastic Compute Cloud (Amazon EC2)," 2010. [Online]. Available: http://aws.amazon.com/ec2/.

[3] Google, "Google App Engine," 2010. [Online]. Available: http://code.google.com/intl/en/appengine/.

[4] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *Proceedings of IEEE World Conference Services*, 2009, pp. 693–700.

[5] J. Martin and A. Nilsson, "On service level agreements for IP networks," in *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002, vol. 2, pp. 855–863. DOI: 10.1109/INFCOM.2002.1019332.

[6] A. Madankan, A. Delavarkhalafi, S.M. Karbassi, and F. Adibnia, "Resource Allocation in Cloud Computing Via Optimal Control to Queuing Systems," *Vestnik YuUrGU. Ser. Mat. Model. Progr.*, vol. 12, no. 4, pp. 67–81, 2019. DOI: 10.14529/mmp190405.

[7] A. Madankan and A. Delavar Khalfi, "Performance Management on Cloud Using Multi-Priority Queuing Systems," in *Advanced Research on Cloud Computing Design and Applications*, 2015, ch. 15, pp. 229–244. DOI: 10.4018/978-1-4666-8676-2.ch015.

[8] A. Madankan, "Performance Bounds and Suboptimal Policies for Multi-Class queue," *Vestnik YuUrGU. Ser. Mat. Model. Progr.*, vol. 12, no. 1, pp. 44–54, 2019. DOI: https://doi.org/10.14529/mmp190104.

[9] J.R. Jackson, "Networks of waiting lines," *Operations Research*, vol. 5, no. 4, pp. 518–521, 1957.

[10] J.R. Jackson, "Jobshop-like queueing systems," *Management Science*, vol. 10, no. 1, pp. 131–142, 1963.

[11] M. Martinello, M. Kaâniche, and K. Kanoun, "Web service availability: impact of error recovery and traffic model," *J Reliab Eng Syst Saf*, vol. 89, no. 1, pp. 6–16, 2005.

[12] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurr Comput Pract Exp*, vol. 24, no. 13, pp. 1397–1420, 2012.

[13] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *11th IEEE/ACM international symposium on cluster, cloud and grid, computing (CCGrid'2011)*, 2011, pp. 104–113.

[14] K.V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10)*, 2010, pp. 193–204.

[15] M. Barbeau and E. Kranakis, *Principles of ad-hoc networking*, New York: Wiley, 2007.

[16] L. Kleinrock, *Queueing systems: theory*, vol. 1, New York: Wiley-Interscience, 1975.

Ali Madankan,

University of Zabol
Zabol, Iran
E–mail: Amadankan@uoz.ac.ir

Received April 16, 2021
Accepted November 21, 2021

Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

Solving the non-linear multi-index transportation problems with genetic algorithms

Tatiana Pașa

Abstract

In this paper we study the non-linear multi-index transportation problem with concave cost functions. We solved the non-linear transportation problem on a network with 5 indices (NTPN5I) described by sources, destinations, intermediate nodes, types of products, and types of transport, that is formulated as a non-linear transportation problem on a network with 3 indices (NTPN3I) described by arcs, types of products, and types of transport. We propose a genetic algorithm for solving the large-scale problems in reasonable amount of time, which was proven by the various tests shown in this paper. The convergence theorem of the algorithm is formulated and proved. The algorithm was implemented in Wolfram Language and tested in Wolfram Mathematica.

Keywords: non-linear programming, concave function, transport problem, index.

MSC 2010: 68W25, 68W50, 90B06, 90C06, 90C08, 90C26, 90C30, 90C35, 90C59.

1 Introduction

Nowadays, the transportation problem is relevant, because of the globalization and the fact that people tend to buy products from different parts of the world, and the price depends on several factors that must be taken into account. The minimum cost of transporting products varies depending on the types of products (construction materials, food, petroleum products), the number and types of transport used [1]

(train, ship, truck, plane), the number of sources and destinations of the transportation network, and also on the transportation route (air, water, roads, pipes, cables) chosen by the carrier. To model these dependencies, transportation costs described by non-linear functions are associated with the paths between intermediate nodes in the transportation network.

The transportation problems on networks with concave cost functions with 5 indices, described by sources, destinations, intermediary nodes, types of products, and types of transport, are formulated and solved in this paper. In order to simplify the mathematical model of the problem, it is formulated as a non-linear transportation problem with 3 indices described by arcs, products, and transport types. Therefore, it is required to minimize the costs of transportation from sources to destinations of several types of products using several types of transport. Results that refer to the solution of the non-linear transportation problem with 4 indices with genetic algorithms are presented in the works [2]–[4]. In [5], the heuristic algorithms to the solution of the non-linear transportation problem with 4 and 5 indices are presented.

This problem is part of the group of transportation problems with n indices. The general case of TPnI was formulated by Ph.-X. Ninh [6], who also proposed an exact method of solving, an extension of the method of potentials by coordinating the solving of the primary and dual problem. An important result is the formulation and proof of the theorem of the necessary and sufficient condition of the existence of the solution. Among the authors who studied the multi-index transportation problem are K. B. Hayley [7], who presented an algorithm, and W. Junginer [8], who conducted a study on the characteristics of the multi-index problem and proposed a set of logical problems for solving transportation problems with several indices.

Solving this problem is difficult, because of the existence of several extremas and the impossibility of differentiating a global minimum from a local one. Because this problem belongs to the class of NP-hard problems, there are no polynomial algorithms to determine the global solution. Heuristic algorithms can be used to solve this problem, in particular, genetic algorithms that allow the solution of the problem to

be generated in a reasonable amount of time, even for large-scale problems. The use of these algorithms in favor of other heuristic algorithms is recommended, because they do not need the gradient or Hessian information. They are also resistant to locks in a local minimum and can be used to solve large-scale non-linear optimization problems.

When a genetic algorithm is applied, it will take into account the fact that each chromosome obtained at the generation of the initial population and after the application of the crossover or mutation operator must correspond to a single admissible solution in the codomain. Another aspect that must be considered is the number of iterations after which the algorithm is stopped and the solution associated with the chromosome with the best characteristics is accepted as the optimal solution of the formulated problem. A large number of iterations means that the generated solution is very close to the global optimum, but in the case of large-scale problems, the large number of iterations also means longer execution time. For this reason, a balance must be found that would allow the solution of the problem to be obtained in a reasonable amount of time, even for large-scale problems.

2 Problem formulation. Main results

In the following, we consider the non-linear transportation problem described by: n sources $A_1, A_2, ..., A_n$ with the respective product capacities $\alpha_1, \alpha_2, ..., \alpha_n$; m destinations $B_1, B_2, ..., B_m$ with the respective product capacities $\beta_1, \beta_2, ..., \beta_m$; p product types $P_1, P_2, ..., P_p$ of respective quantities $\gamma_1, \gamma_2, ..., \gamma_p$; q transport types $T_1, T_2, ..., T_q$ with the respective capacities $\delta_1, \delta_2, ..., \delta_q$; g intermediate nodes $v_r \in V_{int}$, where neither production nor consumption of the flow takes place. Each $e \in E$ arc of the transportation network is associated with a non-linear function that describes the transportation cost that depends on the volume of flow transported over that arc.

In this case, the transportation network is described by an acyclic oriented graph $G = (V, E)$, with the set of nodes $V = V_s \bigcup V_t \bigcup V_{int}$, where V_s – the set of sources, $|V_s| = n$; V_t – the set of destinations, $|V_t| = m$; V_{int} – the set of intermediate nodes, $|V_{int}| = g$; E – the

set of arcs, $|E| = u$. This problem is part of the group of non-linear transportation problems with 5 indices described by sources, destinations, intermediate nodes, product types, and types of transport that is formulated as a non-linear transportation problem on a network with 3 indices (NTPN3I) described by arcs, types of products, and types of transport.

In order to determine the minimum transportation cost, the restrictions on the conservation of the product flow in the network must be observed:

- The total quantity of product of the type P_k, $k = 1, 2, ..., p$ of quantity γ_k, $k = 1, 2, ..., p$ which comes out of all the network sources is described by the relation:

$$\sum_{e \in E^-(V_s)} \sum_{l=1}^{q} x_{ekl} = \gamma_k, k = 1, 2, ..., p, \tag{1}$$

  where $E^-(V_s)$ – the set of outgoing arcs of the network sources.

- The total quantity of transport of the type T_l, $l = 1, 2, ..., q$ with the capacity δ_l, $l = 1, 2, ..., q$ coming out of all the network sources is described by the relation:

$$\sum_{e \in E^-(V_s)} \sum_{k=1}^{p} x_{ekl} = \delta_l, l = 1, 2, ..., q, \tag{2}$$

  where $E^-(V_s)$ – the set of outgoing arcs of the network sources.

- The total quantity of product coming out of each source A_i, $i = 1, 2, ..., n$ is α_i, $i = 1, 2, ..., n$ and described by the relation:

$$\sum_{e \in E^-(s_i)} \sum_{k=1}^{p} \sum_{l=1}^{q} x_{ekl} = \alpha_i, i = 1, 2, ..., n, \tag{3}$$

  where $s_i \in V_s$ – source i of the network, $E^-(s_i)$ – the set of outgoing arcs of the source i of the network.

- The total quantity of product entering each destination B_j, $j = 1, 2, ..., m$ is β_j, $j = 1, 2, ..., m$ and described by the relation:

$$\sum_{e \in E^+(d_j)} \sum_{k=1}^{p} \sum_{l=1}^{q} x_{ekl} = \beta_j, j = 1, 2, ..., m, \tag{4}$$

  where $d_j \in V_t$ – the destination j of the network, $E^+(d_j)$ – the set of incoming arcs of the destination j of the network.

- The quantity of product γ_k, $k = 1, 2, ..., p$ that enters an intermediate node $v_r \in V_{int}$ is equal to the quantity of product that leaves that node and described by the relation:

$$\sum_{e \in E^+(v_r)} \sum_{l=1}^{q} x_{ekl} - \sum_{e \in E^-(v_r)} \sum_{l=1}^{q} x_{ekl} = 0, \begin{array}{l} r = 1, 2, ..., g, \\ k = 1, 2, ..., p, \end{array} \tag{5}$$

  where $v_r \in V_{int}$ – intermediate node of the network, $E^-(v_r)$ – the set of outgoing arcs of the node v_r, $E^+(v_r)$ – the set of incoming arcs of the node v_r.

- The transport quantity δ_l, $l = 1, 2, ..., q$ entering an intermediate node $v_r \in V_{int}$ is equal to the transport quantity leaving that node and described by the relation:

$$\sum_{e \in E^+(v_r)} \sum_{k=1}^{p} x_{ekl} - \sum_{e \in E^-(v_r)} \sum_{k=1}^{p} x_{ekl} = 0, \begin{array}{l} r = 1, 2, ..., g, \\ l = 1, 2, ..., q, \end{array} \tag{6}$$

  where $v_r \in V_{int}$ – intermediate node of the network, $E^-(v_r)$ – the set of outgoing arcs of the node v_r, $E^+(v_r)$ – the set of incoming arcs of the node v_r.

- The values that describe the product flow on the network arcs are non-negative numbers:

$$x_{ekl} \geq 0, \forall (e, k, l), \tag{7}$$

NTPN3I consists in determining a flow x^* that minimizes the cost function:

$$F(x) = \sum_{e\in E}\sum_{k=1}^{p}\sum_{l=1}^{q}\varphi_{ekl}(x_{ekl}), \tag{8}$$

where $\varphi_{ekl}(x_{ekl})$ are non-decreasing concave functions, so the solution of the following non-linear problem is required:

$$F(x) \longrightarrow \min \tag{9}$$

$$\begin{cases} \displaystyle\sum_{e\in E^-(V_s)}\sum_{l=1}^{q} x_{ekl} = \gamma_k, & k = 1,2,...,p, \\ \displaystyle\sum_{e\in E^-(V_s)}\sum_{k=1}^{p} x_{ekl} = \delta_l, & l = 1,2,...,q, \\ \displaystyle\sum_{e\in E^-(s_i)}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ekl} = \alpha_i, & i = 1,2,...,n, \\ \displaystyle\sum_{e\in E^+(d_j)}\sum_{k=1}^{p}\sum_{l=1}^{q} x_{ekl} = \beta_j, & j = 1,2,...,m, \\ \displaystyle\sum_{e\in E^+(v_r)}\sum_{l=1}^{q} x_{ekl} - \sum_{e\in E^-(v_r)}\sum_{l=1}^{q} x_{ekl} = 0, & r = 1,2,...,g, \; k = 1,2,...,p, \\ \displaystyle\sum_{e\in E^+(v_r)}\sum_{k=1}^{p} x_{ekl} - \sum_{e\in E^-(v_r)}\sum_{k=1}^{p} x_{ekl} = 0, & r = 1,2,...,g, \; l = 1,2,...,q, \\ x_{ekl} \geq 0, & \forall(e,k,l), \end{cases} \tag{10}$$

for which the non-negativity conditions are satisfied, so the quantities $\alpha_1, \alpha_2, ..., \alpha_n$, $\beta_1, \beta_2, ..., \beta_m$, $\gamma_1, \gamma_2, ..., \gamma_p$, and $\delta_1, \delta_2, ..., \delta_q$ are non-negative.

The general form of the optimal solution of NTPN3I described by (9)-(10) has the form $x^* = (x^*_{111}, x^*_{112}, ..., x^*_{upq})$ for a transportation network described by arcs. It is assumed that all data are real values and it is desired to obtain a feasible real value as the optimal solution.

2.1 Genetic algorithm MGA

In the following, the NTPN3I problem is solved, using a genetic algorithm, which involves coding the problem so that each chromosome in the population can be associated with only one admissible solution. Each crossover and mutation operator are described so that whenever they are applied to the chromosomes that describe the population, there is only one admissible solution that is associated with it after decoding. The notion of a spanning tree that correctly describes an admissible solution to the problem is used to code the admissible solutions of the problem. Each newly created population retains the chromosomes with the best characteristics from the previous population, so that during the selection the solutions associated with low costs are not lost.

Each chromosome is described by 5 compartments:

- the *first* contains spanning trees, each with the root in one of the sources of the transportation network;
- the *second* contains a permutation of the indices $i = 1, 2, ..., n$ of the sources and describes their service order;
- the *third* contains a permutation of the indices $j = 1, 2, ..., m$ of the destinations and describes their service order;
- the *fourth* contains a permutation of the indices $k = 1, 2, ..., p$ of the types of products and describes in what order they are transported;
- the *fifth* contains a permutation of the indices $l = 1, 2, ..., q$ of the types of transport and describes in what order they can be used.

The proposed ***MGA genetic algorithm*** for solving NTPN3I with concave cost functions is described by the following steps:

Step 1. *Initialization.* The initial population is formed from $4|V|$ randomly generated chromosomes, each described by the set $P = \{\{T_r, r = 1, 2, ..., n\}, P_n, P_m, P_p, P_q\}$, where $T_r = \{(i, j) \,|i, j = 1, 2, ..., n + m + g\}$ – spanning trees with root at source $r = 1, 2, ..., n$ of the transportation network, $P_n = \{p(1), p(2), ..., p(n)\}$ – a permutation of

the indices $i = 1, 2, ..., n$ of the sources, $P_m = \{p(1), p(2), ..., p(m)\}$ – a permutation of the indices $j = 1, 2, ..., m$ of the destinations, $P_p = \{p(1), p(2), ..., p(p)\}$ – a permutation of the indices $k = 1, 2, ..., p$ of the types of products, $P_q = \{p(1), p(2), ..., p(q)\}$ – a permutation of the indices $l = 1, 2, ..., q$ of the types of transport.

Step 2. *Decoding and Chromosome Evaluation.* Decoding assumes that each chromosome is associated with an admissible solution of the form $x = (x_{111}, x_{112}, ..., x_{epq})$, where the flow of k products with l types of transport from each source to destinations on the arcs corresponding to the spanning trees is transported. For this purpose:

1. One-dimensional arrays are created which contain respectively the quantities of products in sources, the product capacities in destinations, the types of products with their respective quantities and the types of transport with the respective capacities as follows: $\alpha' = [\alpha'_1, \alpha'_2, ..., \alpha'_n]$, $\alpha' = [\beta'_1, \beta'_2, ..., \beta'_m]$, $\gamma' = \left[\gamma'_1, \gamma'_2, ..., \gamma'_p\right]$, $\delta' = \left[\delta'_1, \delta'_2, ..., \delta'_q\right]$.

2. For each index $r \in P_{ij}$ the intermediate solution is determined as follows:

   - for each index $j \in P_m$, each index $k \in P_p$ and each index $l \in P_q$ is determined: $x_{ekl} = min\left\{\alpha'_i, \beta'_j, \gamma'_k, \delta'_l\right\}$, after which the update $\alpha'_i := \alpha'_i - x_{ekl}$, $\beta'_j := \beta'_j - x_{ekl}$, $\gamma'_k := \gamma'_k - x_{ekl}$, $\delta'_l := \delta'_l - x_{ekl}$ is applied, where e – the index of the arc $(i, j) \in T_r$;
   - the flow associated to the arc (i, j) with the index e, where $j \notin V_t$, for each spanning tree T_r:

$$x_{rekl} = \begin{cases} 0, & (i,j) \notin T_r, \\ \sum\limits_{(j,j') \in T_r} x_{re'kl}, & e'-\text{the index of the arc } (j,j') \end{cases}$$

   where $k = 1, 2, ..., p, l = 1, 2, ...q$.

As a result, intermediate solutions of the following form are obtained:

$$x_{rekl} = (x_{r111,}, x_{r112}, ..., x_{rupq}), r = 1, 2, ..., n$$

and the final admissible solution associated to the chromosome has the form:

$$x = \left(\sum_{i=1}^{n} x_{i111}, \sum_{i=1}^{n} x_{i112}, ..., \sum_{i=1}^{n} x_{iupq} \right).$$

Chromosome evaluation involves determining the value of the objective function for each of the admissible solutions associated with the chromosomes.

Step 3. *Chromosome selection.* Chromosomes are sorted in order of increasing value of the objective function of the admissible solution associated with the chromosome. Chromosomes from the first half of the population are transferred to the new population.

Step 4. *Crossover.* Crossover is applied to chromosomes transferred to the population $P(i)$ from the population $P(i-1)$ by selection. The same cut is randomly applied to compartment 1 of both parent-chromosomes. The offspring-chromosomes are obtained as follows:

- the first offspring-chromosome receives the spanning trees up to the cut of the mother-chromosome with the respective order of sources and types of products and the spanning trees after the cut of the father-chromosome with the respective order of destinations and types of transport;

- the second offspring-chromosome receives the spanning trees up to the cut of the father-chromosome with the respective order of sources and types of products and the spanning trees after the cut of mother-chromosome with the respective order of destinations and types of transport.

Each pair of parent-chromosomes generates two offspring-chromosomes and the size of the new population remains constant.

Step 5. *Mutation.* A mutation with probability $\epsilon \in [0.1, 0.5]$ is applied to a gene on the offspring-chromosome and involves:

- for each compartment, *two* – sources, *three* – destinations, *four* – types of products, and *five* – types of transport, it is randomly decided whether the mutation is applied;
- in each of the selected compartments the values are changed between two randomly selected elements.

Step 6. *Check the stop condition.* It involves stopping the algorithm after k iterations. The final solution to the problem is the one that corresponds to the chromosome with the minimum objective function value in the last generated population. Go to *Step 2* if the stop condition is not met.

In the case of large-scale problems, the execution time can serve as a condition for stopping the algorithm. Therefore, after a certain period of time, the generation of new populations is stopped, and the solution that corresponds to the chromosome with the best characteristics from the last population serves as a final solution to the problem.

2.2 Theoretical results

The MGA algorithm is applied to transportation networks that meet the following conditions:

1. The graph which describes the transportation network is connected and acyclic;
2. There are at least 2 sources, 2 destinations, 2 types of products, 2 types of transport, and at least one intermediate node;
3. There is at least one path from each source to each destination;
4. Source nodes do not contain ascending arcs, and destination nodes do not contain descending arcs.

The following theorems are formulated and proved:

Theorem 1. *The MGA algorithm requires* $O(|V|(n|V|+n+m+p+q))$ *memory.*

Proof. The input data described by the restriction table is of size $O(upq)$. $O(upq)$ memory is required for the solution associated to the chromosome with the best characteristics of the population. A chromosome requires $O(n|V| + n + m + p + q)$ memory, and for the entire population – $O(4|V|(n|V| + n + m + p + q))$ memory is required. Therefore, the MGA algorithm requires $O(|V|(n|V| + n + m + p + q))$ memory. □

Theorem 2. *The complexity of an iteration of the MGA algorithm is* $O(npq|V|(m + |V| + u))$.

Proof. $O(upq)$ operations are required to initialize the input data described by the constraint table. Generating a chromosome requires $O(n|V| + n + m + p + q)$ operations, so generating the entire population requires $O(|V|(n|V| + n + m + p + q))$ operations. Evaluating the solution associated with a chromosome requires $O(n(mpq + |V|pq + upq))$ operations. Because there are $4|V|$ chromosomes in the population, the complexity of evaluating all solutions associated to the chromosomes is $O(npq|V|(m|V| + u))$. The crossover is of an $O(n|V| + n + m + p + q)$ complexity for one chromosome and $O(|V|(n|V| + n + m + p + q))$ for the entire population. Therefore, the time complexity of an iteration of the MGA algorithm is $O(npq|V|(m + |V| + u))$. □

From Theorem 2, it results that the complexity of the MGA algorithm is $O(U(n|V|+n+m+p+q))$*, where* U *is the number of iterations necessary to obtain a final solution associated with the chromosome with good characteristics.*

Theorem 3. *The MGA algorithm converges to the local optimum.*

Proof. From the population $P(i-1)$ chromosomes are transferred to the population $P(i)$ for which the value of the objective function in the associated solution is minimal. The chromosomes of the populations generated at each iteration are associated with an admissible solution,

so it can be said that after the execution of a finite number of iterations the chromosome is detected to which is associated the solution that describes the local optimum. Therefore, the MGA algorithm converges to the local optimum. □

2.3 Practical results

The MGA algorithm was implemented in the Wolfram language and tested based on a series of examples, transportation problems of different sizes, that is, different number of sources, destinations, intermediate nodes, types of products, and types of transport that can be used.

Tables 1 and 2 present the results of solving a series of problems of different sizes, where cost functions are defined by concave non-decreasing functions. The size of the transportation problem is described by sources, destinations, types of products transported, types of transport used, and intermediate nodes. The total of the products available in the sources is equal to the total of the products needed in the destinations. The total of products of different types coincides with the total capacity of transport types used.

The tests were performed on an Intel i5-2500 machine with 4 Cores and 8 GB of DDR3 memory in Wolfram Mathematica 12. The time being given after the generation of 10 populations.

Table 1 presents the execution time of the MGA algorithm for problems of different sizes: number of sources, destinations, intermediate nodes, product types, and transport types, the solution of which uses from 48 unknowns in the case of the problem with 2 sources, 2 destinations, 2 types of products, 2 types of transport, and 2 intermediate nodes, up to 34500 unknowns in the case of the problem with 10 sources, 10 destinations, 10 types of products, 10 types of transport, and 10 intermediate nodes. As can be seen, the algorithm allows obtaining pseudo-optimal solutions in a reasonable amount of time even in the case of large-scale transportation problems.

Table 2 presents the results of solving a series of network transportation problems of different sizes, where for each of the 10 populations generated, there can be seen the changes in the value of the objective

Table 1. Execution time of the MGA algorithm

n/m/p/q	*2/2/2/2*	*3/3/3/3*	*4/4/4/4*	*4/5/4/5*	*5/5/5/5*
1	0.2500	1.3125	4.3750	6.6406	12.2656
2	0.2656	1.2500	4.3437	6.5781	12.4375
3	0.2812	1.2500	4.5468	6.6562	12.5781
4	0.3437	1.2968	4.4531	6.5937	12.4063
5	0.3281	1.2500	4.4531	6.5468	12.4844
Unkn.	48	261	864	1240	2125
n/m/p/q	*6/6/6/6*	*7/7/7/7*	*8/8/8/8*	*9/9/9/9*	*10/10/10/10*
1	29.6719	62.1563	118.938	213.656	353.234
2	29.5156	62.0000	117.906	211.906	359.266
3	29.4531	62.7344	118.922	216.281	361.453
4	29.6406	62.0625	117.375	211.656	356.750
5	20.9531	62.6406	117.031	216.406	355.703
Unkn.	4428	8232	14080	22599	34500

function calculated for the solution associated with the chromosome with the best characteristics in that population and the value of the total objective function for that population.

From the data presented in Table 2 it can be observed that $F_T(x)$ generally decreases from one iteration to another, with some exceptions – when crossover and mutation operations result in solutions that correspond to higher costs than those of the previous population. The insignificant increase of $F_T(x)$ is followed by decreases, which suggests that the majority of chromosomes in new populations have better characteristics than those in the population obtained in the previous iteration.

Although at several consecutive iterations the same value of $F(x)$ is repeated, which means that at several consecutive iterations a chromosome with better characteristics is not found, the algorithm is able to exit such a lock in a local optimum by determining a new chromosome with better characteristics and therefore an admissible solution for which the value of the objective function is lower.

Table 2. $F(x)$ and $F_T(x)$ of the MGA algorithm

n/m/p/q Iteration	*4/4/4/4*	*5/5/5/5*	*6/6/6/6*	*7/7/7/7*	*10/10/10/10*
I	30/2431	43/4078	60/6301	73/8897	94/17495
II	30/2263	43/3767	51/5837	73/8560	94/17176
III	29/2113	43/3584	51/5691	69/8059	94/16846
IV	29/2117	42/3343	51/5317	61/7774	94/16546
V	28/1885	41/3208	50/5145	61/7657	88/16157
VI	28/1890	40/3033	45/4878	55/7529	88/15799
VII	26/1851	37/2923	45/4744	55/7219	88/15839
VIII	17/1831	32/2801	45/4619	55/7286	85/15480
IX	17/1826	32/2775	44/4559	55/7126	85/15403
X	17/1819	32/2774	40/4551	55/6865	85/15299

The algorithm allows the determination of an optimal local solution for which the value of the objective function is lower the more iterations of the algorithm are executed. The number of iterations depends on the size of the problem being solved, the time available to provide the solution, and also the technological capabilities available.

3 Conclusions

For the formulated problem, the MGA genetic algorithm is proposed, which can obtain the solution in a reasonable amount of time for large-scale problems. In connection with other studies of non-linear transportation problems with several indices, the results have been obtained that complement the currently known results in this field. In particular:

1. the formulation of the non-linear transportation problem on a network with 5 indices described by sources, destinations, intermediate nodes, product types, and transport types as a non-linear transportation problem on a network with 3 indices described by

products, transport types, and the set of arcs;

2. building the MGA genetic algorithm for solving the non-linear transportation problem on a network with 5 indices formulated as a non-linear transportation problem on a network with 3 indices, described by concave cost functions, which is based on the application of the selection, crossover, and mutation operators of the genetic algorithm;

3. the description of a correct coding of the respective problem for the MGA genetic algorithm, so that after decoding only one admissible solution is obtained for each chromosome;

4. description of crossover and mutation operators, so that whenever they are applied within the MGA genetic algorithm, after decoding the newly created chromosome, an admissible solution of the problem is obtained.

References

[1] O. Diaz-Para, A. Ruiz-Vanoye, B. B. Loranca, A. Fluentes-Penna, and R. A. Barrera-Camara, "A survey of transportation problem," *Journal of Applied Mathematics*, vol. 2014, Article ID: 848129, 17 p., Hindawi Publishing Corporation, DOI: http://dx.doi.org/10.1155/2014/848129. [Online] Available: http://downloads.hindawi.com/journals/jam/2014/848129.pdf.

[2] T. Pașa, "Multi-index transport problem with non-linear cost functions," *Romai J.*, vol. 14, no. 2, pp. 129–137, 2018. [Online] Available: https://rj.romai.ro/arhiva/2018/2/Pasa.pdf.

[3] T. Pașa and V. Ungureanu, "Solving the non-linear 4-index transportation problem," in *The Fifth Conference of Mathematical Society of the Republic of Moldova*, (Chișinău: Vladimir Andrunachievici Institute of Mathematics and Computer Science), 2019, pp. 221–224. [Online] Available: https://ibn.idsi.md/sites/default/files/imag_file/221-224_9.pdf.

[4] T. Pașa, "Solving non-linear multi-index transportation problems," *Romai J.*, vol. 15, no. 2, pp. 91–99, 2019. [Online] Available: https://rj.romai.ro/arhiva/2019/2/Pasa.pdf.

[5] T. Pașa, "Algorithms for solving nonlinear multi-index transportation problems," *Studia Universitatis Moldaviae, Seria Științe Exacte*, vol. 132, no. 2, pp. 36–44, 2020. ISSN: 1857-2073. Online ISSN: 2345-1033. (in Romanian)

[6] P.-X. Ninh, "N index transportation problem," *Mathematical Journal*, vol. 7, no. 1, pp. 18–25, 1979.

[7] K. B. Haley, "The solid transportation problem," *Operat. Research*, no. 10, pp. 448–463, 1962.

[8] W. Junginger, "On reprezentative of multi-index transportation problems," *European Journal of Operational Research*, no. 66, pp. 353–371, 1993. DOI:10.1016/0377-2217(93)90223-A.

Tatiana Pașa,

Moldova State Univerity
Faculty of Mathematics and Computer Science
60 A. Mateevici, MD-2009, Chișinău
Republic of Moldova
E–mail: pasa.tatiana@yahoo.com

Received August 16, 2021
Accepted January 30, 2022

Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

A hybrid deep learning and handcrafted feature approach for the prediction of protein structural class

Rached Yagoubi, Abdelouahab Moussaoui,
Mohamed Bachir Yagoubi

Abstract

The knowledge of the protein structural class is one of the most important sources of information related to protein structure or that about function analysis and drug design. But researchers still face difficulties to predict the protein structural class when it is a question about low-similarity sequences. In this paper, we propose to make the prediction using a hybrid deep learning method and handcrafted features instead of shallow classifier. We input only nine features, mostly from predicted secondary structure information, to a feed-forward deep neural network. The latter will automatically explore and extend those features through many layers and discover the representations needed for classification. The obtained results, when applying the jackknife test on two low-similarity benchmark datasets (25PDB and FC699), proved to be very significant. After comparing our method to others, it has turned out that using deep learning methods affords satisfactory performance in the field of protein structural class prediction.

Keywords: Protein structural class, Deep Learning, feed-forward deep neural network, predicted secondary structure information.

MSC 2010: 97R40, 92B20, 68T05, 92D20.

1 Introduction

The importance of the protein structural class is displayed in multiple domains. For instance, in the field of protein structure, function analy-

sis, drug design, and a lot of other biomedical applications [1], [2]. Protein structural class was introduced firstly by Levitt and Chothia [3] in 1976. The study they led was related to the polypeptide chain topologies. It consisted in studying a dataset composed of 31 globular proteins. The results yielded to the categorization of the known structure protein domains into four structural classes: $all-\alpha, all-\beta, \alpha/\beta$ and $\alpha+\beta$ classes. The $all-\alpha$ classes represent structures that consist of mainly α-helices, and $all-\beta$ classes represent structures that consist of mainly β-strands. The α/β and $\alpha+\beta$ classes contain both α-helices and β-strands, where the α/β class includes mainly parallel β-sheets, and the $\alpha+\beta$ class includes anti-parallel β-sheets. Since then, many other studies in the field emerged based on the four main classes. However, due to the rapid advance of sequencing technology, genomics, and proteomics, the number of newly discovered protein sequences has grown exponentially. This growth has led to a substantial gap between the number of sequence-known and the number of structure-known proteins. As a result of such inquiry, efforts have been made to reduce this gap, by setting newly developed methods that allow fast and accurate determination of the protein structure classes. In the same line of thought and to perform the structural class prediction, we generally need to follow two steps. The first one is linked to the transformation of the Amino Acid sequences into fixed-length feature vectors, whereas the second step consists of the feeding of the feature vectors to a classification algorithm.

In a parallel way during the last three decades, many efforts have followed trying to improve the accuracy of the prediction. In that, multiple sequence features representing protein sequences have been applied. We may cite the amino acid composition (AAC) [4], pseudo amino acid composition (PseAA) [5], polypeptide composition [6], functional domain composition [7], predicted secondary structure information [8], [9], and PSI-BLAST profile [10]. In these terms, a series of classification algorithms have been used to make the prediction. As example, we can name Bayesian classification [11], information discrepancy [12], fuzzy clustering [13], rough sets [14], logistic regression [15], k-nearest neighbors [16], Fisher's linear discriminate algorithm [17],

support vector machine (SVM) [18], [19], [2], classifier ensembles [20], and Convolutional Neural Network [21] .

The year 2006 has witnessed the emergence of deep learning, also called hierarchical learning, on behalf of machine learning research [22]. The principal aim of deep learning methods is to learn feature hierarchies. This learning will be achieved through features from the higher levels of the hierarchy already composed of lower-level features [23]. We may add that deep learning methods are seen as a clue in finding solutions to problems that had displayed resistance to the artificial intelligence community for a long period [24]. For instance, deep learning has proved its high efficiency in image recognition [25] and speech recognition [26], also in predicting activities in the fields of protein structure prediction [27], prediction of protein subcellular localization [28].

In this paper, we will exploit the power of deep neural networks for the prediction of protein structural class problem. Firstly, we have used a previous custom-designed feature vector from [9] that includes only nine hand-crafted features (features that are designed beforehand by human experts). After that, the vector is fed to a feed-forward deep neural network to automatically explore those features and discover the representations needed for classification. For more objectivity, we have chosen the jackknife cross-validation test as a means of evaluation to assess the proposed prediction method. The test is used on a couple of largely adopted datasets (25PDB and FC699) both related to be low-similarity benchmark datasets.

The paper is organized as follows. Section 2 gives a small description of the two datasets, the feature vector, the proposed feed-forward deep neural network architecture, and the different performance measure indexes. The obtained results are detailed in Section 3. Section 4 concludes the paper.

2 Materials and methods

2.1 Datasets

One has to know that the effect of sequence similarity on prediction accuracy weighs heavily, since less similarity makes prediction too dif-

ficult to perform [29], [30]. This is obvious regarding the definition of similarity by the amount of amino acids existing in the protein sequences which present resemblance after aligning that sequence with or to other sequences from a given dataset [30]. To test the performance of our approach, two widely used benchmark datasets were selected. The first is the 25PDB dataset [29]. It contains 1673 low similarity proteins (Lower than 25%). The other dataset is FC699 [9] including 858 sequences with similarity lower than 40%. Table 1 presents detailed compositions of each dataset. 1 presents detailed compositions of each dataset.

Table 1. The composition of 25PDB and FC699 datasets

Dataset	$all-\alpha$	$all-\beta$	α/β	$\alpha+\beta$	Total
25PDB	443	443	346	441	1673
FC699	130	269	377	82	858

2.2 Feature vector

In this work, we have used a custom-designed feature vector that includes 9 features previously defined on [9]. 8 out of them are relaying on information obtained from secondary structure predicted with PSIPRED [31], [32]. The last feature is taken directly from the amino acid sequence. Table 2 gives a small description of each feature.

For example, if we have the following amino acid sequence:
MDPFLVLLHSVSSSLSSSELTELKYLCLGRVGKRKLERVQSGLD
LFSMLLEQNDLEPGHTELLRELLASLRRHDLLRRVDDFELEHH
HHHH

The predicted secondary structure sequence is:
CCHHHHHHHHHHHCCCCHHHHHHHHHHHHHHCCCCCCCCCC
CCHHHHHHHHHHHHHCCCCCCHHHHHHHHHHHHCHHHHHHH
HHHHHHHHHHCCC

The obtained features are:
PSIPRED-$NCount_H{}^6 = 1$
PSIPRED-$NCount_H{}^8 = 1$

Table 2. The description of the used features

Feature	Description
PSIPRED-$NCount_H{}^6$	Normalized count of α-helix segments (including at least 6 residues)
PSIPRED-$NCount_H{}^8$	Normalized count of α-helix segments (including at least 8 residues)
PSIPRED-$CMV_H{}^1$	Composition moment vector of α-helix segments (first order)
PSIPRED-$NAvgSeg_H$	Normalized average length of α-helix segments
PSIPRED-CV_E	Composition vector of β-strand segments
PSIPRED-$NCount_E{}^5$	Normalized count of β-strand segments (including at least 5 residues)
PSIPRED-$MaxSeg_E$	Length of the longest β-strand segment
PSIPRED-$NAvgSeg_E$	Normalized average length of β-strand segments
CV_{L---G}	Count of collocated amino acid pair (L, G) separated by 3 gaps

PSIPRED-$CMV_H{}^1 = 0.3543$
PSIPRED-$NAvgSeg_H = 0.3659$
PSIPRED-$CV_E = 0$
PSIPRED-$NCount_E{}^5 = 0$
PSIPRED-$MaxSeg_E = 0$
PSIPRED-$NAvgSeg_E = 0$
$CV_{L---G} = 1$

2.3 Deep neural network

A standard neural network (NN) is composed of linked processors. The latter are called neurons. Each neuron produces a sequence of real-valued activations. In this part of the definition, we point at the fact that neuron activation may differ from a neuron to another. For

instance, Input neurons need sensors that perceive the environment; others get activated via well-arranged connections, with the help of former active neurons, favoring the process of activation. Another set of neurons may affect the environment due to their triggering activities. Looking for the exact weights that allow the NN display the wished comportment is considered as the credit assignment main function. In this context, such comportment will probably need long causal series of computational phases. The phases take on their charge to change the network aggregate activation, usually in a non-linear way. The essential about Deep Learning is to assign credit accurately through such phases [33].

After multiple experimental tests, the best feed-forward deep neural network model used is described in detail below. Figure 1 and the referred description offer an outline of the architecture used: as input data, we use only 9 features described previously.

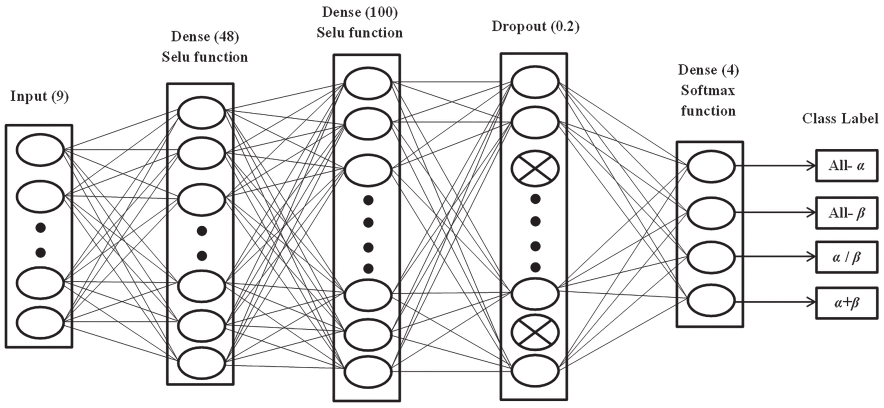


Figure 1. The proposed Deep Neural Network architecture

We have used a dense (48) layer (a fully-connected layer with 48 hidden units) which has been followed by a second dense (100) layer. In both layers, we have adopted the scaled exponential linear units (SELUs) function [34] as activation. In order to prevent our deep neural network from overfitting, we have applied a dropout [35] with a rate

of 0.2. Whereas the output layer is a fully-connected layer with four hidden units (dense (4)) that uses, as activation function, the softmax regression function, as we are dealing with a multiclass classification problem. Our model was run for 150 epochs (epoch = full pass over the training set), and the stochastic gradient descent (Adam) [36] was chosen as an optimization algorithm. Our method is implemented in Python programming language using Kears library under TensorFlow backend.

2.4 Performance measures

To obtain an accurate statistical prediction, three tests cannot be circumvented. It is the question of the independent dataset test, the subsampling test, and the jackknife test. The three cross-validation methods are largely used. They can easily examine the predictor effectiveness in a given practical situation [1], [30]. But according to the results, the jackknife test is considered the most objective. Its efficiency lays in the fact that it always gives the same result for a given dataset [8]. This is why the jackknife test has been chosen to test the validity of our method. Along the adopted test, every protein sequence in the dataset is picked out, in turn, as a test sample, when the predictor is trained by the rest of the protein sequences. In these terms and to measure the truthfulness and the strength of our predictor, five indexes have been selected for this purpose. The indexes in question are: sensitivity *(Sens)*, precision (Prec), F-measure *(F)*, Area Under ROC Curve *(AUC)* and Overall accuracy *(OA)*. AUC is the area calculated under the Receiver Operating Characteristic (ROC). AUC is used to measure the average performance of the classification model. Its value is between 0 and 1. The value 1 is considered as perfect, whereas the value 0 seems to be untruthful. So, the more the value draws near 1 means that the classification model is more truthful [30]. Here are five formulas representing respectively the five indexes:

$$Sens = \frac{TP}{TP + FN};$$

$$Prec = \frac{TP}{TP + FP};$$

$$F = 2 \times \frac{Prec \times Sens}{Prec + Sens};$$

$$AUC = \frac{1}{2} \times \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right);$$

$$OA = \frac{TP + TN}{TP + FP + TN + FN}.$$

In these formulas, the abbreviations *TP, FP, TN,* and *FN* stand respectively for the following: true positives, false positives, true negatives, and false negatives.

3 Results and discussion

3.1 Prediction performances of our method

The results of protein structural class prediction on the datasets 25PDB and FC699, obtained by jackknife test and performance evaluation indicators Sens, Prec, F-measure, AUC, and OA, are shown in Table 3.

Table 3. The prediction quality of our method on 25PDB and FC699 datasets

Dataset	Structural class	*Sens*(%)	*Prec*(%)	*F-measure*	*AUC*
25PDB	$all-\alpha$	96.8	95.6	0.96	0.98
	$all-\beta$	96.3	93.4	0.95	0.95
	α/β	95.1	92.2	0.94	0.93
	$\alpha+\beta$	90.9	86.9	0.89	0.88
	OA	94.8			
FC699	$all-\alpha$	98.9	99.1	0.99	0.99
	$all-\beta$	97.7	96.2	0.97	0.98
	α/β	92.8	93.4	0.93	0.98
	$\alpha+\beta$	97.8	94.5	0.96	0.94
	OA	96.8			

Respectively, the results of 94.8% and that of 96.8% represent the overall prediction accuracy of the datasets 25PDB and FC699. As seen in the same table after comparing the prediction accuracy of four protein structural classes, we have noticed that the performance evaluation indexes of $all-\alpha$ class are about 95.6% – 98.0% on the dataset 25PDB, whereas it was 98.9% – 99.1% on dataset FC699. On the other hand, the performance indexes of $all-\beta$ class are about 93.4% – 98.0% on both protein structural class datasets. This demonstrates that we archive a very high performance for the previous structural classes. The explanation laying behind the fact that $all-\alpha$ class and $all-\beta$ class have been so performing is that these proteins are helices or stands, in addition to being easily predictable since they are single and highly repetitive structures [37]. In parallel, the result of α/β and $\alpha+\beta$ classes have revealed so satisfactory. On 25PDB and FC699 datasets, the performances indexes of α/β are 92.2% – 98.0% and about 86.9% – 97.8% for $\alpha+\beta$ class.

3.2 Comparison with other prediction methods

In order to demonstrate the adopted method efficiency, we have compared the performance of the proposed method with the recently reported protein structural class prediction methods on the same datasets, as shown in Table 4.

Seeing the results displayed in Table 4 about the two datasets, our selected method proves to be the best performing on overall accuracies. It is striking, when analyzing the results, that the overall accuracies on 25PDB and FC699 datasets, respectively scored 1.4% and 0.3% higher than the previous top performed results. We have also noticed that the significant improvement on 25PDB is particularly made for the α/β and $\alpha+\beta$ classes, which tend to be difficultly predictable classes. In this way, we have improved the accuracy of α/β by 0.3% and by 1.3% on $\alpha+\beta$ class. Concerning FC699, our deep neural network overpasses the previous best-performing method by 8.8% on the $\alpha+\beta$ class.

Table 4. Performance comparison of different methods on two datasets

Dataset	Method	Accuracy (%)				
		$all-\alpha$	$all-\beta$	α/β	$\alpha+\beta$	Overall
25PDB	[9]	92.6	80.1	74.0	71.0	79.7
	[18]	-	-	-	-	83.0
	[38]	94.1	87.1	84.1	74.4	85.0
	[39]	95.0	91.4	77.5	88.7	88.8
	[19]	98.9	89.6	85.6	78.9	88.4
	[2]	96.4	94.8	92.5	89.6	93.4
	[40]	95.6	89.5	88.1	87.0	90.1
	[30]	**97.3**	88.9	90.8	79.4	89.0
	[37]	95.7	**97.7**	94.8	84.4	93.1
	[20]	91.4	82.4	78.6	74.2	81.8
	This paper	96.8	96.3	**95.1**	**90.9**	**94.8**
FC699	[9]	-	-	-	-	87.5
	[41]	96.2	90.7	96.3	69.5	92.0
	[18]	-	-	-	-	94.5
	[38]	96.9	94.8	97.1	78.1	94.5
	[39]	98.5	**98.1**	**97.6**	81.7	96.5
	[19]	97.7	97.4	97.1	79.3	95.6
	[42]	96.9	89.2	89.4	89.0	90.4
	[20]	98.5	94.8	95.8	75.6	93.9
	This paper	**98.9**	97.7	92.8	**97.8**	**96.8**

4 Conclusion

Protein structural class prediction is considered as an important problem in bioinformatics. Despite all the attempts that have been done to solve this problem, the prediction accuracy improvement for low-similarity protein datasets is still a challenge. The main reason for the inadequate improvement is the unsatisfactory prediction accuracies for protein from α/β and $\alpha+\beta$ classes. In this paper, we have proposed to apply deep learning for the prediction of protein structural class. A 9-dimensional feature vector (8 features based on information extracted

from the secondary structure and one feature computed from the sequence) is used as input to a feed-forward deep neural network. The deep neural network and the jackknife test are employed to predict and evaluate the model over two benchmark datasets: 25PDB and FC699 datasets with sequence similarity lower than 25% and 40%, respectively. The results show that our approach outperforms the existing methods, especially for proteins from the α/β and $\alpha + \beta$ classes.

References

[1] K.-C. Chou and C.-T. Zhang, "Prediction of Protein Structural Classes," *Critical Reviews in Biochemistry and Molecular Biology*, vol. 30, no. 4, pp. 275–349, jan 1995. [Online]. Available: https://doi.org/10.3109/10409239509083488

[2] L. Zhang, L. Kong, X. Han, and J. Lv, "Structural class prediction of protein using novel feature extraction method from chaos game representation of predicted secondary structure," *Journal of Theoretical Biology*, vol. 400, pp. 1–10, 2016.

[3] M. Levitt and C. Chothia, "Structural patterns in globular proteins," *Nature*, vol. 261, p. 552, jun 1976. [Online]. Available: http://dx.doi.org/10.1038/261552a0http://10.0.4.14/261552a0

[4] K.-C. Chou, "A novel approach to predicting protein structural classes in a (20-1)-D amino acid composition space," *Proteins: Structure, Function, and Genetics*, vol. 21, no. 4, pp. 319–344, apr 1995. [Online]. Available: http://doi.wiley.com/10.1002/prot.340210406

[5] C. Chen, Y.-X. Tian, X.-Y. Zou, P.-X. Cai, and J.-Y. Mo, "Using pseudo-amino acid composition and support vector machine to predict protein structural class," *Journal of Theoretical Biology*, vol. 243, no. 3, pp. 444–448, dec 2006.

[6] R. Y. Luo, Z. P. Feng, and J. K. Liu, "Prediction of protein structural class by amino acid and polypeptide composition," *European Journal of Biochemistry*, vol. 269, no. 17, pp. 4219–4225,

sep 2002. [Online]. Available: http://doi.wiley.com/10.1046/j.1432-1033.2002.03115.x

[7] K.-C. Chou and Y.-D. Cai, "Predicting protein structural class by functional domain composition," *Biochemical and Biophysical Research Communications*, vol. 321, no. 4, pp. 1007–1009, sep 2004.

[8] L. A. Kurgan, T. Zhang, H. Zhang, S. Shen, and J. Ruan, "Secondary structure-based assignment of the protein structural classes," *Amino Acids*, vol. 35, no. 3, pp. 551–564, oct 2008.

[9] L. Kurgan, K. Cios, and K. Chen, "SCPRED: Accurate prediction of protein structural class for sequences of twilight-zone similarity with predicting sequences," *BMC Bioinformatics*, vol. 9, no. 1, p. 226, 2008. [Online]. Available: https://doi.org/10.1186/1471-2105-9-226

[10] T. Liu, X. Zheng, and J. Wang, "Prediction of protein structural class for low-similarity sequences using support vector machine and PSI-BLAST profile," *Biochimie*, vol. 92, no. 10, pp. 1330–1334, 2010.

[11] Z.-X. Wang and Z. Yuan, "How good is prediction of protein structural class by the component-coupled method?" *Proteins: Structure, Function, and Genetics*, vol. 38, no. 2, pp. 165–175, feb 2000.

[12] L. Jin, W. Fang, and H. Tang, "Prediction of protein structural classes by a new measure of information discrepancy," *Computational Biology and Chemistry*, vol. 27, no. 3, pp. 373–380, jul 2003.

[13] H.-B. Shen, J. Yang, X.-J. Liu, and K.-C. Chou, "Using supervised fuzzy clustering to predict protein structural classes," *Biochemical and Biophysical Research Communications*, vol. 334, no. 2, pp. 577–581, 2005.

[14] Y. Cao, S. Liu, L. Zhang, J. Qin, J. Wang, and K. Tang, "Prediction of protein structural class with Rough Sets," *BMC Bioinformatics*, vol. 7, 2006.

[15] S. Jahandideh, P. Abdolmaleki, M. Jahandideh, and S. H. S. Hayatshahi, "Novel hybrid method for the evaluation of param-

eters contributing in determination of protein structural classes," *J Theor Biol*, vol. 244, 2007.

[16] T.-L. Zhang, Y.-S. Ding, and K.-C. Chou, "Prediction protein structural classes with pseudo-amino acid composition: Approximate entropy and hydrophobicity pattern," *Journal of Theoretical Biology*, vol. 250, no. 1, pp. 186–193, jan 2008.

[17] J.-Y. Yang, Z.-L. Peng, and X. Chen, "Prediction of protein structural classes for low-homology sequences based on predicted secondary structure," *BMC Bioinformatics*, vol. 11, no. 1, p. S9, 2010. [Online]. Available: https://doi.org/10.1186/1471-2105-11-S1-S9

[18] L. Nanni, S. Brahnam, and A. Lumini, "Prediction of protein structure classes by incorporating different protein descriptors into general Chou's pseudo amino acid composition," *Journal of Theoretical Biology*, vol. 360, pp. 109–116, 2014.

[19] J. Wang, C. Wang, J. Cao, X. Liu, Y. Yao, and Q. Dai, "Prediction of protein structural classes for low-similarity sequences using reduced PSSM and position-based secondary structural features," *Gene*, vol. 554, no. 2, pp. 241–248, 2015.

[20] S. Bankapur and N. Patil, "Enhanced Protein Structural Class Prediction using Effective Feature Modeling and Ensemble of Classifiers," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, p. 1, 2020.

[21] S. Kaur, A. Sharma, and P. Singh, "Protein structural classes prediction based on convolutional neural network classifier with feature selection of hybrid pso-fa optimization approach," *European Journal of Molecular & Clinical Medicine*, vol. 7, no. 10, pp. 252–265, 2020.

[22] L. Deng and D. Yu, "Deep learning: Methods and applications," pp. 197–387, 2013.

[23] D. Erhan, A. Courville, and P. Vincent, "Why Does Unsupervised Pre-training Help Deep Learning ?" *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, p. 436, may 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14539http://10.0.4.14/nature14539

[25] C. Farabet, C. Couprie, L. Najman, and Y. Lecun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, aug 2013.

[26] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, nov 2012.

[27] M. Torrisi, G. Pollastri, and Q. Le, "Deep learning methods in protein structure prediction," *Computational and structural biotechnology journal*, vol. 18, pp. 1301–1310, jan 2020. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32612753https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7305407/

[28] J. J. Almagro Armenteros, C. K. Sønderby, S. K. Sønderby, H. Nielsen, and O. Winther, "Erratum: DeepLoc: prediction of protein subcellular localization using deep learning (Bioinformatics (Oxford, England) (2017))," p. 4049, dec 2017.

[29] L. A. Kurgan and L. Homaeian, "Prediction of structural classes for protein sequences and domains—Impact of prediction algorithms, sequence representation and homology, and test procedures on accuracy," *Pattern Recognition*, vol. 39, no. 12, pp. 2323–2343, 2006.

[30] Y. Liang and S. Zhang, "Predict protein structural class by incorporating two different modes of evolutionary information into Chou's general pseudo amino acid composition," *Journal of Molecular Graphics and Modelling*, vol. 78, pp. 110–117, 2017.

[31] D. T. Jones, "Protein secondary structure prediction based on position-specific scoring matrices11Edited by G. Von Heijne," *Journal of Molecular Biology*, vol. 292, no. 2, pp. 195–202, 1999.

[32] D. W. A. Buchan and D. T. Jones, "The PSIPRED Protein

Analysis Workbench: 20 years on," *Nucleic Acids Research*, apr 2019. [Online]. Available: https://doi.org/10.1093/nar/gkz297

[33] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[34] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, jun 2017, pp. 972–981. [Online]. Available: http://arxiv.org/abs/1706.02515

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[36] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, dec 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[37] B. Yu, L. Lou, S. Li, Y. Zhang, W. Qiu, X. Wu, M. Wang, and B. Tian, "Prediction of protein structural class for low-similarity sequences using Chou's pseudo amino acid composition and wavelet denoising," *Journal of Molecular Graphics and Modelling*, vol. 76, pp. 260–273, 2017.

[38] L. Kong and L. Zhang, "Novel structure-driven features for accurate prediction of protein structural class," *Genomics*, vol. 103, no. 4, pp. 292–297, 2014.

[39] J. Wang, Y. Li, X. Liu, Q. Dai, Y. Yao, and P. He, "High-accuracy prediction of protein structural classes using PseAA structural properties and secondary structural patterns," *Biochimie*, vol. 101, pp. 104–112, 2014.

[40] M. H. Olyaee, A. Yaghoubi, and M. Yaghoobi, "Predicting protein structural classes based on complex networks and recurrence analysis," *Journal of Theoretical Biology*, vol. 404, pp. 375–382, 2016.

[41] L. Kong, L. Zhang, and J. Lv, "Accurate prediction of protein

structural classes by incorporating predicted secondary structure information into the general form of Chou's pseudo amino acid composition," *Journal of Theoretical Biology*, vol. 344, pp. 12–18, 2014.

[42] L. Liu, J. Cui, and J. Zhou, "A Novel Prediction Method of Protein Structural Classes Based on Protein Super-Secondary Structure," *Journal of Computer and Communications*, vol. 04, no. 15, pp. 54–62, nov 2016. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jcc.2016.415005

Rached Yagoubi, Abdelouahab Moussaoui,
Mohamed Bachir Yagoubi

Received August 26, 2020
Revised version October 30, 2021
Accepted January 11, 2022

Rached Yagoubi
Computer Science and Mathematics Laboratory,
Amar Telidji University of Laghouat,
Ghardaia Road, 03000 Laghouat, Algeria
E–mail: rached.yagoubi@gmail.com

Abdelouahab Moussaoui
Department of Computer Science,
Setif 1 University, El Bez Setif,
19000 Setif, Algeria
E–mail: moussaoui.abdel@gmail.com

Mohamed Bachir Yagoubi
Computer Science and Mathematics Laboratory,
Amar Telidji University of Laghouat,
Ghardaia Road, 03000 Laghouat, Algeria
E–mail: m.yagoubi@lagh-univ.dz

Computer Science Journal of Moldova, vol.**30**, no.1(88), 2022

Triple Roman domination subdivision number in graphs

J. Amjadi H. Sadeghi

Abstract

For a graph $G = (V, E)$, a triple Roman domination function is a function $f : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ having the property that for any vertex $v \in V(G)$, if $f(v) < 3$, then $f(\mathrm{AN}[v]) \geq |\mathrm{AN}(v)|+3$, where $\mathrm{AN}(v) = \{w \in N(v) \mid f(w) \geq 1\}$ and $\mathrm{AN}[v] = \mathrm{AN}(v) \cup \{v\}$. The weight of a triple Roman dominating function f is the value $\omega(f) = \sum_{v \in V(G)} f(v)$. The triple Roman domination number of G, denoted by $\gamma_{[3R]}(G)$, equals the minimum weight of a triple Roman dominating function on G. *The triple Roman domination subdivision number* $\mathrm{sd}_{\gamma_{[3R]}}(G)$ of a graph G is the minimum number of edges that must be subdivided (each edge in G can be subdivided at most once) in order to increase the triple Roman domination number. In this paper, we first show that the decision problem associated with $\mathrm{sd}_{\gamma_{[3R]}}(G)$ is NP-hard and then establish upper bounds on the triple Roman domination subdivision number for arbitrary graphs.

Keywords: Triple Roman domination number, Triple Roman domination subdivision number.

MSC 2010: 05C69.

1 Introduction

In this paper, G is a simple graph with vertex set $V(G)$ and edge set $E(G)$ (briefly V and E). For every vertex $v \in V(G)$, the *open neighborhood* of v is the set $N_G(v) = N(v) = \{u \in V(G) \mid uv \in E(G)\}$ and its *closed neighborhood* is the set $N_G[v] = N[v] = N(v) \cup \{v\}$. Similarly, the *open neighborhood* of a set $S \subseteq V$ is the set $N(S) =$

$\cup_{v\in S}N(v)$ and the *closed neighborhood* of S is $N[S] = N(S) \cup S$. The *degree* of a vertex $v \in V$ is $\deg_G(v) = |N(v)|$. The *minimum degree* and the *maximum degree* of a graph G are denoted by $\delta = \delta(G)$ and $\Delta = \Delta(G)$, respectively. The *distance* between two vertices u and v is the length of a shortest path joining them. We denote by $N_2(v)$ the set of vertices at distance 2 from the vertex v and put $d_2(v) = |N_2(v)|$. For a more thorough treatment of domination parameters and for terminology not present here, see [17].

A *[k]-Roman domination function* ([k]-RDF) is a function $f : V(G) \longrightarrow \{0, 1, 2, 3, \ldots, k + 1\}$ such that for any vertex $v \in V(G)$, if $f(v) < k$, then $f(\mathrm{N}[v]) \geq |\mathrm{AN}(v)| + k$, where $\mathrm{AN}(v) = \{w \in N(v) \mid f(w) \geq 1\}$. The weight of a [k]-RDF is the value $\omega(f) = \sum_{v\in V(G)} f(v)$. The *[k]-Roman domination number* $\gamma_{[kR]}(G)$ is the minimum weight of a [k]-RDF of G. A [k]-Roman domination function $f : V(G) \longrightarrow \{0, 1, 2, 3, \ldots, k + 1\}$ can be represented by the order partition $(V_0^f, V_1^f, V_2^f, V_3^f, \ldots, V_{k+1}^f)$ of V, where $V_i^f = \{v \in V(G) \mid f(v) = i\}$ for $i \in \{0, 1, 2, 3, \ldots, k + 1\}$. The [k]-Roman domination number was introduced by Abdollahzadeh Ahangar et al. in [1]. The case $k = 1$ is Roman domination which was introduced by Cockyne et al. in [13], the case $k = 2$ is the double Roman domination which was investigated in [9], the case $k = 3$ is *triple Roman domination number* (TRD-number) and has been studied in [1, 2, 16], and the case $k = 4$ is the *Quadruple Roman domination* and has been investigated in [4]. The literature on Roman domination and its variants has been detailed in two chapters of the books and a surveys paper (see [10–12]). Here, we restrict our attention to triple Roman domination number.

The *triple Roman domination subdivision number* $\mathrm{sd}_{\gamma_{[3R]}}(G)$ of a graph G is the minimum number of edges that must be subdivided (where each edge in G can be subdivided at most once) in order to increase the triple Roman domination number of G.

The domination subdivision number was first introduced in Velmmal's thesis [19], and since then many papers has been published in domination subdivision parameters, see for instance [3, 5–8, 14, 18].

If $G_1, G_2, \ldots, G_s$ are the components of G, then $\gamma_{[3R]}(G) = \sum_{i=1}^{s} \gamma_{[3R]}(G_i)$ and $\mathrm{sd}_{\gamma_{[3R]}}(G) = \min\{\mathrm{sd}_{\gamma_{[3R]}}(G_i) \mid 1 \leq i \leq s\}$. Hence,

it is sufficient to study $\mathrm{sd}_{\gamma_{[3R]}}(G)$ for connected graphs. Since the triple Roman domination subdivision number of the graph K_2 does not change when its only edge is subdivided, in the study of the triple Roman domination subdivision number, we must assume that the graph has order at least 3.

In this paper, we first show that the decision problem associated with $\mathrm{sd}_{\gamma_{[3R]}}(G)$ is NP-hard and then establish upper bounds on the triple Roman domination subdivision number for bipartite graphs.

We make use of the following results in this paper.

Proposition A. [2] *In a triple Roman dominating function of weight $\gamma_{[3R]}(G)$, no vertex needs to be assigned the value 1.*

Proposition B. [2] *There is no connected graph of order n such that $\gamma_{[3R]}(G) = 5$.*

Proposition C. *Let G be a connected graph of order n.*

(i) If $n \geq 2$, then $\gamma_{[3R]}(G) = 4$ if and only if $\Delta(G) = n - 1$.

(ii) If $n \geq 4$, then $\gamma_{[3R]}(G) = 6$ if and only if there are two non adjacent vertices in $V(G)$ with degree $\Delta(G) = n - 2$.

Proposition D. [1] *For $n \geq 2$,*

$$\gamma_{[3R]}(P_n) = \begin{cases} 4\lfloor \frac{n}{3} \rfloor & \text{if } n \equiv 0 \pmod 3 \\ 4\lfloor \frac{n}{3} \rfloor + 3 & \text{if } n \equiv 1 \pmod 3 \\ 4\lfloor \frac{n}{3} \rfloor + 4 & \text{if } n \equiv 2 \pmod 3. \end{cases}$$

Proposition E. [1] For $n \geq 3$,

$$\gamma_{[3R]}(C_n) = \begin{cases} \lceil \frac{4n}{3} \rceil & \text{if either } n = 4, 5, 7, 10 \text{ or } n \equiv 0 \pmod 3 \\ \lceil \frac{4n}{3} \rceil + 1 & \text{if } n \neq 4, 5, 7, 10 \text{ and } n \equiv 1, 2 \pmod 3. \end{cases}$$

As the results of Propositions D and E, we have:

Corollary 1. For $n \geq 3$,

$$\mathrm{sd}_{\gamma_{[3R]}}(P_n) = \begin{cases} 1 & \text{if } n \equiv 0, 1 \pmod 3 \\ 2 & \text{if } n \equiv 2 \pmod 3. \end{cases}$$

Corollary 2. For $n \geq 3$,

$$\mathrm{sd}_{\gamma_{[3R]}}(C_n) = \begin{cases} 1 & \text{if either } n = 5 \text{ or } n \equiv 0, 1 \pmod 3 \\ 2 & \text{if } n \equiv 2 \pmod 3 \ n \neq 5. \end{cases}$$

The proofs of the following observations are straightforward and therefore omitted.

Observation 3. *If K_n is the complete graph of order n and $n \geq 3$, then* $\mathrm{sd}_{\gamma_{[3R]}}(K_n) = 1$.

Observation 4. *If $K_{n,m}$ is the complete bipartite graph and $m, n \geq 3$, then* $\mathrm{sd}_{\gamma_{[3R]}}(K_{n,m}) = 2$.

2 Some preliminary results

In this section, we present some upper bounds on $\mathrm{sd}_{\gamma_{[3R]}}(G)$ in terms of the vertex degree and the minimum degree of G. Our first result shows that subdividing an edge does not decrease the triple Roman domination number.

Lemma 1. Let G be a simple connected graph of order $n \geq 3$ and $e = uv \in E(G)$. If G' is obtained from G by subdivision the edge e, then $\gamma_{[3R]}(G') \geq \gamma_{[3R]}(G)$.

Proof. Let x be the subdivision vertex and let f be a $\gamma_{[3R]}(G')$-function. Since f is a TRDF on G', we have that $f(u) + f(v) + f(x) \geq 4$. Let $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ be a function defined by $g(u) = \min\{4, f(u) + f(x)\}$ and $g(z) = f(z)$ whenever $z \in V(G) \setminus \{u\}$. Notice that g is a TRDF on G and $\omega(g) \leq \omega(f)$. Hence $\gamma_{[3R]}(G') \geq \gamma_{[3R]}(G)$, which completes the proof. □

We proceed with three propositions giving some sufficient conditions for a graph to having small triple Roman domination subdivision number.

Proposition 1. If G contains a strong support vertex, then $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$.

Proof. Let u, v be two leaves adjacent to w and let G' be obtained from G by subdividing the edge uw with vertex x. Let f be a TRDF on G', then $f(u) + f(v) + f(w) + f(x) \geq 7$. Define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(w) = 4$, $g(u) = g(v) = 0$ and $g(z) = f(z)$ for each $z \in V(G) \setminus \{u, v, w\}$. Clearly, g is a TRDF of G with $\omega(g) < \omega(f)$ and hence $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$. □

Proposition 2. *Let G be a connected graph of order $n \geq 3$. If $\gamma_{[3R]}(G) = 4, 6$, then $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$.*

Proof. First assume $\gamma_{[3R]}(G) = 4$. By Proposition C, we have $\Delta(G) = n-1$. Suppose v is a vertex with maximum degree $\Delta(G)$ and $w \in N(v)$. Let G' be obtained from G by subdivision the edge vw with vertex x. Then $\Delta(G') < n$ and so $\gamma_{[3R]}(G') > 4$ by Proposition C, which implies $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$.

Now assume $\gamma_{[3R]}(G) = 6$, then $G = \overline{K_2} \vee H$ by Proposition C. Let $V(\overline{K_2}) = \{u, w\}$ and $e = uv \in E(G)$, where $v \in V(H)$. Let G' be obtained from G by subdividing the edge e with vertex x. If $\gamma_{[3R]}(G') = 6$, then, as above, $G' = \overline{K_2} \vee H'$ in where $V(K_2) = \{a, b\}$. If $x = a$ ($x = b$ is similar), then since $N_{G'}(x) = \{u, v\}$ and $w \notin N_{G'}(x), w = b$. Hence $w \in N_G(u)$ which is a contradiction. Thus $x \notin \{a, b\}$, this implies that $a, b \in N_{G'}(x) = \{u, v\}$, hence $\{a, b\} = \{u, v\}$ and so w and u are adjacent in G, a contradiction.Therefore $\gamma_{[3R]}(G) < \gamma_{[3R]}(G')$, implying that $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$. This completes the proof. □

By Proposition 2, we have:

Corollary 5. *Let G be a simple connected graph of order $n \geq 3$. If $\mathrm{sd}_{\gamma_{[3R]}}(G) \geq 2$, then $\gamma_{[3R]}(G) \geq 7$.*

Proposition 3. *For every connected graph G of order $n \geq 3$ with $\delta(G) = 1$, $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 2$.*

Proof. Let v be a support vertex of $G, u \in V(G)$ be the leaf adjacent to v and $w \in N(v) - \{u\}$. Assume G' be obtained from G by subdividing the edges uv, vw with vertices x, y, respectively. Let f be a $\gamma_{[3R]}(G')$-function. Without loss of generality, assume that $f(z) \neq 1$ for all

vertices $z \in V(G')$ by Proposition A. It is easy to see that $f(u) + f(x) + f(v) + f(y) \geq 4$. If $f(w) \neq 0$, then the function $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ defined by $g(v) = 0$, $g(u) = 3$ and $g(x) = f(x)$ otherwise, is clearly TRDF of G of weight less than $\omega(f)$. If $f(w) = 0$, then $f(u) + f(x) + f(v) + f(y) \geq 7$. Define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v) = 4, g(u) = 0$ and $g(z) = f(z)$ for each $z \in V(G) \setminus \{u, v\}$. Observe that g is a TRDF on G with $\omega(g) < \omega(f)$. Hence $sd_{\gamma_{[3R]}}(G) \leq 2$. □

Next result is an immediate consequence of Proposition 3 and Corollary 1.

Theorem 6. *For every tree T of order at least 3, $\mathrm{sd}_{\gamma_{[3R]}}(T) \leq 2$. Furthermore, this bound is sharp for the paths of order n for which $n \equiv 2 \pmod 3$.*

3 Complexity of the triple Roman domination subdivision problem

In this section, we will show that the triple Roman domination subdivision number problem in bipartite graphs is NP-hard. We first state the problem as the following decision problem.

Triple Roman domination subdivision number problem (TRDSN):
Instance: *A nonempty graph G and a positive integer k.*
Question: *Is* $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq k$?

Following Garey and Johnson's techniques for proving NP-completeness given in [15], we prove our results by describing a polynomial transformation from the well-known NP-complete problem: **3-SAT**. To state 3-SAT, we recall some terms.

Let U be a set of Boolean variables. A truth assignment for U is a mapping $t : U \longrightarrow \{T, F\}$. If $t(u) = T$, then u is said to be "true" under t; if $t(u) = F$, then u is said to be "false" under t. If u is a variable in U, then u and $\bar{u}$ are *literals* over U. The literal u is true

under t if and only if the variable $\bar{u}$ is false under t; the literal $\bar{u}$ is true if and only if the variable u is false.

A *clause* over U is a set of literals over U. It represents the disjunction of these literals and is satisfied by a truth assignment if and only if at least one of its members is true under that assignment. A collection $\mathscr{C}$ of clauses over U is satisfiable if and only if there exists some truth assignment for U that simultaneously satisfies all the clauses in $\mathscr{C}$. Such a truth assignment is called a *satisfying truth assignment* for $\mathscr{C}$. The 3-SAT is specified as follows.

3-satisfiability problem (3-SAT):

Instance: A collection $\mathscr{C} = \{C_1, C_2, \ldots, C_m\}$ of clauses over a finite set U of variables such that $|Cj| = 3$ for $j = 1, 2, ..., m$.

Question: *Is there a truth assignment for U that satisfies all the clauses in $\mathscr{C}$?*

Theorem 7. ([15] Theorem 3.1). *3-SAT is NP-complete.*

Theorem 8. *3RSN is NP-hard even for bipartite graphs.*

Proof. The transformation is from 3-SAT. Let $U = \{u_1, u_2, \ldots, u_n\}$ and $\mathscr{C} = \{C_1, C_2, \ldots, C_m\}$ be an arbitrary instance of 3-SAT. We will construct a bipartite graph G and choose an integer k such that $\mathscr{C}$ be satisfiable if and only if $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq k$. We construct such a graph G as follows.

For each $i = 1, 2, \ldots, n$, corresponding to the variable $u_i \in U$, associate a complete bipartite graph $H_i = K_{3,5}$ with bipartite sets $X = \{x_i, y_i, z_i\}$ and $Y = \{v_i, u_i, w_i, \overline{u}_i, r_i\}$. For each $j = 1, 2, \ldots, m$, corresponding to the clause $C_j = \{p_j, q_j, r_j\} \in \mathscr{C}$, associate a single vertex c_j and add the edge set $c_j u_i$ if $u_i \in C_j$ and $c_j \bar{u}_i$ if $\bar{u}_i \in C_j$. Finally, add a graph H with vertex set $V(H) = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$ and $E(H) = \{s_1s_2, s_2s_3, s_3s_4, s_4s_5, s_5s_6, s_6s_7\}$, join s_1 and s_7 to each vertex c_j with $1 \leq j \leq m$ and set $k = 1$.

Figure 1 shows that on the graph obtained when $U = \{u_1, u_2, u_3, u_4\}$ and $\mathscr{C} = \{C_1, C_2, C_3\}$, where $C_1 = \{u_1, u_2, \bar{u}_3\}, C_2 = \{\bar{u}_1, u_2, u_4\}, C_3 = \{\bar{u}_2, u_3, u_4\}$. To prove that this is indeed a transformation, we only need to show that $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$ if and only if there is a truth assignment for

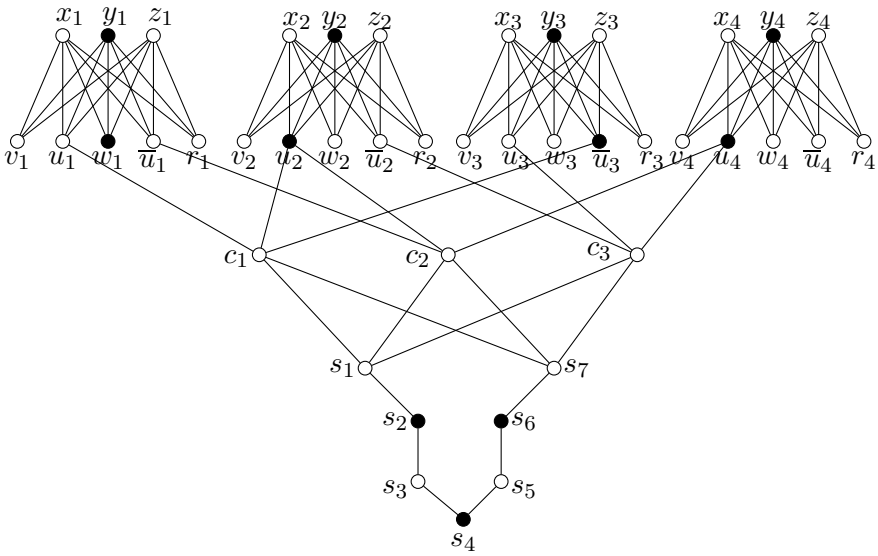


Figure 1. An instance of the triple Roman subdivision number problem resulting from an instance of 3-SAT. Here $k = 1$ and $\gamma_{[3R]}(G) = 43$, where the bold vertex p means there is a TRDF f with $f(p) = 4$, with the exception of s_4, where $f(s_4) = 3$.

U that satisfies all clauses in $\mathscr{C}$. This aim can be obtained by proving the following four claims.

Claim 1. $\gamma_{[3R]}(G) \geq 8n + 11$. Moreover, if $\gamma_{[3R]}(G) = 8n + 11$, then for any $\gamma_{[3R]}$-function f on G and for each $1 \leq i \leq n, f(H_i) = 8, f(V(H)) = 11, \max\{f(s_1), f(s_7)\} \leq 2, \min\{f(s_1), f(s_7)\} = 0$ and $f(c_j) = 0$ for each $1 \leq j \leq m$.

Proof. Let f be a $\gamma_{[3R]}$-function of G. Without loss of generality assume that $f(z) \neq 1$ for all vertices $z \in V(G)$ by Proposition A. For each $i = 1, 2, \ldots, n$, it is also clear that $f(V(H_i)) \geq 8$ and $f(N_G[H]) \geq 11$, implying that $\gamma_{[3R]}(G) \geq 8n + 11$.

Now suppose that $\gamma_{[3R]}(G) = 8n + 11$. Since $H_i := K_{m,n}$ with $m, n \geq 3$, $f(V(H_i)) = 8$. We show that $f(s_1) + f(s_7) < 5$ and

$\max\{f(s_1), f(s_7)\} \neq 4$. To contradiction, let $f(s_1) + f(s_7) \geq 5$. If $f(s_1) = 4$ ($f(s_7) = 4$ is similar), then since $\sum_{i=3}^{5} f(s_i) \geq 4$ and $f(s_6) + f(s_7) \geq 4$, we have $\gamma_{[3R]}(G) \geq 8n + 12$, a contradiction. Thus, $\max\{f(s_1), f(s_7)\} \neq 4$ and $f(s_1) + f(s_7) \leq 6$. If $f(s_1) + f(s_7) = 6$, then $f(s_1) = f(s_7) = 3$. This implies that $\sum_{i=2}^{6} f(s_i) \geq 6$ and so $\gamma_{[3R]}(G) \geq 8n+12$, which is a contradiction. If $f(s_1)+f(s_7) = 5$, without loss of generality, let $f(s_1) = 2$ and $f(s_7) = 3$ ($f(s_1) = 3, f(s_7) = 2$ is similar), then $f(N[s_1]) + f(s_3) + f(N[s_5]) \geq 9$. Thus, $\gamma_{[3R]}(G) \geq 8n + 12$, which is a contradiction again. Therefore, $f(H_i) = 8$ for each $1 \leq i \leq n, \max\{f(s_1), f(s_7)\} \leq 2, \min\{f(s_1), f(s_7)\} = 0$. Now assume $f(c_j) \neq 0$ for some $j \in \{1, 2, ..., n\}$. Since $\max\{f(s_1), f(s_7)\} \leq 2$ and $\min\{f(s_1), f(s_7)\} = 0$, we have $f(s_1) = f(s_7) = 0, f(s_1) = 2$ and $f(s_7) = 2$ or $f(s_1) = 2$ and $f(s_7) = 0$. First assume that $f(s_1) = f(s_7) = 0$. If $f(s_2) = 0$ or $f(s_6) = 0$, then $\sum_{j+1}^{m} f(c_j) \geq 4$ and $\sum_{i=2}^{6} f(s_i) \geq 8$. Hence $\gamma_{[3R]}(G) \geq 8n + 12$, a contradiction. If $f(s_2) \neq 0$ and $f(s_6) \neq 0$, then $\sum_{j+1}^{m} f(c_j) \geq 2$ and $\sum_{i=2}^{6} f(s_i) \geq 10$, which is a contradiction. Now, assume that $f(s_1) = 0$ and $f(s_7) = 2$ or $f(s_1) = f(s_7) = 2$. Then, similar as above, it is easy to see that $\gamma_{[3R]}(G) \geq 8n + 12$, a contradiction. Therefore, $f(c_j) = 0$ for each $1 \leq j \leq m$, as desired. □

Claim 2. $\mathscr{C}$ is satisfiable if and only if $\gamma_{dR}(G) = 8n + 11$.

Proof. Suppose that $\gamma_{[3R]}(G) = 8n + 11$ and let f be a $\gamma_{[3R]}$-function of G. By Claim 1, $f(H_i) = 8$ for each $i = 1, 2, \ldots n$, and since H_i is a complete bipartite graph, at most one of $f(u_i)$ and $f(\bar{u}_i)$ is 4 for each i. Define a mapping $t : U \longrightarrow \{T, F\}$ by

$$t(u_i) = \begin{cases} T & \text{if either } f(u_i) = 4 \text{ or } f(u_i), f(\overline{u_i}) \neq 4 \\ F & \text{if } f(\bar{u}_i) = 4. \end{cases} \quad i = 1, \ldots, n \qquad (1)$$

Suppose that $\gamma_{[3R]}(G) = 8n+11$ and let f be a $\gamma_{[3R]}$-function of G. By Claim 1, $f(H_i) = 8$ for each $i = 1, 2, \ldots .n$, and since H_i is a complete bipartite graph, at most one of $f(u_i)$ and $f(\bar{u}_i)$ is 4 for each i. We now show that t is a satisfying truth assignment for $\mathscr{C}$. It is sufficient to

show that every clause in $\mathscr{C}$ is satisfied by t. To this end, we arbitrarily choose a clause $C_j \in \mathscr{C}$ with $1 \leq j \leq m$.

By Claim 1, we have $f(s_1) + f(s_7) < 5, \max\{f(s_1), f(s_7)\} \leq 2, \min\{f(s_1), f(s_7)\} = 0$ and $f(c_j) = 0$ for each $1 \leq j \leq m$. Besides, since c_j is not adjacent to s_i for $i = 2, \ldots, 6$, then there exists some i with $1 \leq i \leq n$ such that c_j is adjacent to u_i or $\bar{u}_i$. Suppose that c_j is adjacent to u_i, where $f(u_i) = 4$. Since u_i is adjacent to c_j in G, the literal u_i is in the clause C_j by the construction of G. Since $f(u_i) = 4$, it follows that $t(u_i) = T$ by (1), which implies that the clause C_j is satisfied by t. Suppose that c_j is adjacent to $\bar{u}_i$, where $f(\bar{u}_i) = 4$. Since $\bar{u}_i$ is adjacent to c_j in G, the literal $\bar{u}_i$ is in the clause C_j. Since $f(\bar{u}_i) = 4$, it follows that $t(u_i) = F$ by (1). Thus, t assigns $\bar{u}_i$ the truth value T, that is, t satisfies the clause C_j. By the arbitrariness of j with $1 \leq j \leq m$, we show that t satisfies all the clauses in $\mathscr{C}$, so $\mathscr{C}$ is satisfiable.

Conversely, suppose that $\mathscr{C}$ is satisfiable, and let $t : U \longrightarrow \{T, F\}$ be a satisfying truth assignment for $\mathscr{C}$. Create a function f on $V(G)$ as follows: if $t(u_i) = T$, then let $f(u_i) = 4$, and if $t(u_i) = F$, then let $f(\bar{u}_i) = 4$. Let $f(y_i) = f(s_2) = f(s_6) = 4$ for each $1 \leq i \leq n, f(s_4) = 3$ and the remaining vertices of G assigned a 0 under f. Clearly, $f(G) = 8n + 11$. Since t is a satisfying truth assignment for $\mathscr{C}$, for each $j = 1, 2, \ldots, m$, at least one of literals in C_j is true under the assignment t. It follows that the corresponding vertex c_j in G is adjacent to at least one vertex p with $f(p) = 4$. Since c_j is adjacent to each literal in C_j by the construction of G, thus f is a TRDF of G, and so $\gamma_{[3R]}(G) \leq f(G) = 8n + 11$. Hence $\gamma_{[3R]}(G) = 8n + 11$, by Claim 1. □

Claim 3. Let G' be obtained from G by subdividing any edge e of $E(G)$, then $\gamma_{[3R]}(G') \leq 8n + 12$.

Proof. Let $e = uv \in E(G)$ and let G' be obtained from G by subdividing the edge e with vertex w. If $e \in \{s_1s_2, s_2s_3\}$, consider the function $f : V(G') \longrightarrow \{0, 1, 2, 3, 4\}$ defined by $f(s_1) = f(s_3) = f(s_6) = f(x_i) = f(v_i) = 4$ for each $1 \leq i \leq n$ and $f(x) = 0$ for all other $x \in V(G')$. If

$e \in \{s_4s_5, s_5s_6\}$, consider the function $f : V(G') \longrightarrow \{0,1,2,3,4\}$ defined by $f(s_1) = f(w) = f(s_6) = f(x_i) = f(v_i) = 4$ for each $1 \leq i \leq n$ and $f(x) = 0$ for all other $x \in V(G')$. If $e = s_6s_7$, consider the function $f : V(G') \longrightarrow \{0,1,2,3,4\}$ defined by $f(s_1) = f(s_4) = f(w) = f(x_i) = f(v_i) = 4$ for each $1 \leq i \leq n$ and $f(x) = 0$ for all other $x \in V(G')$. If $e = s_6s_7$, consider the function $f : V(G') \longrightarrow \{0,1,2,3,4\}$ defined by $f(s_2) = f(s_4) = f(w) = f(x_i) = f(v_i) = 4$ for each $1 \leq i \leq n$ and $f(x) = 0$ for all other $x \in V(G')$. If $e \in \{s_1c_j, s_7c_j$, for each $j = 1, 2, \ldots, m\}$, consider the function $f : V(G') \longrightarrow \{0,1,2,3,4\}$ defined by $f(s_1) = f(s_4) = f(s_7) = f(x_i) = f(v_i) = 4$ for each $1 \leq i \leq n$ and $f(x) = 0$ for all other $x \in V(G')$. If $e \in \{c_ju_i$, for each $1 \leq i \leq n$, $1 \leq j \leq m\}$ or $e \in \{c_j\overline{u}_i$, for each $1 \leq i \leq n$, $1 \leq j \leq m\}$, consider the function $f : V(G') \longrightarrow \{0,1,2,3,4\}$ defined by $f(s_1) = f(s_4) = f(s_7) = f(x_i) = f(u_i) = 4$ or $f(s_1) = f(s_4) = f(s_7) = f(x_i) = f(\bar{u}_i) = 4$, respectively, for each $1 \leq i \leq n$ and $f(x) = 0$ for all other $x \in V(G')$. If $e = uv$ such that $u \in \{x_i, y_i, z_i\}$ and $v \in \{v_i, u_i, w_i, \overline{u}_i, r_i\}$, consider the function $f : V(G') \longrightarrow \{0,1,2,3,4\}$ defined by $f(s_1) = f(s_4) = f(s_7) = f(u) = f(v) = 4$ and $f(x) = 0$ for all other $x \in V(G')$, then in each case, f is a TRDF on G' with $\omega(f) = 8n + 12$. Therefore, $\gamma_{[3R]}(G') \leq 8n + 12$. □

Claim 4. *$\gamma_{[3R]}(G) = 8n + 11$ if and only if $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$.*

Proof. Assume $\gamma_{[3R]}(G) = 8n + 11$. Let G' be obtained from G by subdivision the edge $e = s_1s_2$ with vertex w.

Suppose for a contradiction that $\gamma_{[3R]}(G) = \gamma_{[3R]}(G')$. Let f' be a $\gamma_{[3R]}$-function of G'. It is easy to see that $f'(V(H) \cup \{w\}) \geq 12$. On the other hand, since $f'(H_i) \geq 8$ for each $1 \leq i \leq n$, we have $\gamma_{[3R]}(G') = \omega(f') \geq 8n + 12$. This implies that $\gamma_{[3R]}(G) \geq 12$, which is a contradiction. Therefore, $\gamma_{[3R]}(G) < \gamma_{[3R]}(G')$ and so $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$.

Conversely, assume that $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$. Let G' be obtained from G by subdivision the edge e such that $\gamma_{[3R]}(G) < \gamma_{[3R]}(G')$. By Claim 3, we have $\gamma_{[3R]}(G') \leq 8n + 12$. Now the result follows by Claim 1. □

By Claims 2 and 4, we prove that $\mathrm{sd}_{\gamma_{[3R]}}(G) = 1$ if and only if there is a truth assignment for U that satisfies all clauses in $\mathscr{C}$. Since

the construction of the triple Roman subdivision number instance is straightforward from a 3-satisfiability instance, the size of the triple Roman subdivision number instance is bounded above by a polynomial function of the size of 3-satisfiability instance. It follows that this is a polynomial reduction and the proof is complete. □

4 Bounds in terms of order and maximum degree

In this section, we present some upper bounds on $\mathrm{sd}_{\gamma_{[3R]}}(G)$ in terms of the vertex degree and the minimum degree of G.

Theorem 9. *Let G be a connected graph. If $v \in V(G)$ has the degree at least two, then* $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq \deg(v)$.

Proof. Let $t = \deg(v)$ and $N(v) = \{v_1, v_2, \ldots, v_t\}$ and G' be obtained from G by subdividing the edges $vv_1, vv_2, \ldots, vv_t$ with vertices $x_1, x_2, \ldots, x_t$, respectively. Let f be a $\gamma_{[3R]}(G')$-function. Without loss of generality, assume that $f(z) \neq 1$ for all vertices $z \in V(G')$ by Proposition A. If $\Sigma_{i=1}^{t} f(x_i) + f(v) \geq 5$, then define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v) = 4$ and $g(z) = f(z)$ for $z \in V(G) \setminus \{v\}$. Clearly, g is a TRDF of G with $\omega(g) < \omega(f)$ and hence $sd_{[3R]}(G) \leq \deg(v)$. We assume that $\Sigma_{i=1}^{t} f(x_i) + f(v) \leq 4$. (Note that f is a TRDF on G' and so $f(N[v]) \geq 3$). If $f(v) = 2$, then there exists some x_i, say without loss of generality, x_1, such that $f(x_1) = 2$ and $\sum_{i=2}^{t} f(x_i) = 0$. Hence $f(v_i) \geq 3$ for each $2 \leq i \leq t$. Define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v) = 3$ and $g(z) = f(z)$ for $z \in V(G) \setminus \{v\}$. If $f(v) = 3$, then $\sum_{i=1}^{t} f(x_i) = 0$ and $f(v_i) \geq 2$ for each $1 \leq i \leq t$. Define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v) = 2$ and $g(z) = f(z)$ for $z \in V(G) \setminus \{v\}$. Finally, assume that $f(v) = 4$. Hence $\sum_{i=1}^{t} f(x_i) = 0$. Define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v) = 3$ and $g(z) = f(z)$ for $z \in V(G) \setminus \{v\}$. Clearly, in each case, g is a TRDF of G with $\omega(g) < \omega(f)$ and so $\gamma_{[3R]}(G) \leq \omega(g) < \omega(f) \leq \gamma_{[3R]}(G')$.

Now assume that $f(v) = 0$, then to be triple Roman dominate the vertex v, we must have $f(x_i) = 4$ for some $1 \leq i \leq t$, say $i = 1$.

Thus, $f(x_j) = 0$, and thus, $f(v_j) = 4$ for all $2 \leq j \leq t$. Define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v_1) = \min\{f(v_1)+3, 4\}$ and $g(z) = f(z)$ for $z \in V(G) \setminus \{v_1\}$. Since $t \geq 2$, clearly g is a TRDF of G with $\omega(g) < \omega(f)$. Thus, $\gamma_{[3R]}(G) \leq \omega(g) < \omega(f)$, and this implies that $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq \deg(v)$. □

A consequence of Theorem 9 is that $\mathrm{sd}_{\gamma_{[3R]}}(G)$ is defined for every connected graph G of order $n \geq 3$. In addition:

Corollary 10. *For every connected graph G with $\delta \geq 2$, $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq \delta$.*

It is well known that every planar graph contains at least one vertex of degree at most five. Thus, the following result is an immediate consequence of Corollary 10.

Corollary 11. *For every planar graph G, $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 5$.*

In the sequel, we present an upper bound on the triple Roman domination subdivision number in terms of δ_2. We make use of the following lemmas in the proof of Theorem 12.

Lemma 2. *Let G be a connected graph of order $n \geq 3$ and let G have a vertex $v \in V(G)$ which is contained in a triangle uvw such that $N(u) \cup N(w) \subseteq N[v]$. Then $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 3$.*

Proof. Let G' be obtained from G by subdivision the edges vu, vw, uw with vertices x, y, z, respectively. Let f be a $\gamma_{[3R]}(G')$-function. Without loss of generality, assume that $f(z) \neq 1$ for all vertices $z \in V(G')$ by Proposition A. We claim that $f(v)+f(u)+f(w)+f(x)+f(y)+f(z) \geq 6$. If $0 \notin \{f(x), f(y), f(z)\}$, then the claim is directly correct. Hence, we assume that $0 \in \{f(x), f(y), f(z)\}$, say without loss of generality, $f(x) = 0$, then $f(u) + f(v) \geq 4$. Now, if $f(y) \neq 0$ or $f(z) \neq 0$, then $f(v) + f(u) + f(w) + f(x) + f(y) + f(z) \geq 6$, as desired. Hence, we assume that $f(y) = f(z) = 0$, therefore, to triple Roman dominate x, y and z we must have $f(u) + f(v) + f(w) \geq 8$. Now define the function $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v) = 4$, $g(u) = g(w) = 0$ and $g(s) = f(s)$ for each $s \in V(G) \setminus \{v, u, w\}$. Obviously g is a TRDF of G of weight less than $\gamma_{[3R]}(G)$. This completes the proof. □

Lemma 3. *Let G be a connected graph of order $n \geq 3$ and let G have a vertex $v \in V(G)$ which is contained in a triangle uvw such that $N(u) \subseteq N[v]$ and $N(w) \setminus N[v] \neq \emptyset$. Then*

$$\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 3 + |N(w) \setminus N[v]|.$$

Proof. Let $N(w) \setminus N[v] = \{w_1, w_2, \ldots, w_k\}$ and let G' be obtained from G by subdividing the edges vu, vw, uw with x, y, z, respectively, and for each $1 \leq i \leq k$ the edge ww_i with vertex z_i. Assume g is a $\gamma_{[3R]}(G')$-function. Similar as the proof of Lemma 2, we have $g(v) + g(u) + g(w) + g(x) + g(y) + g(z) \geq 6$. Define $h : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $h(v) = 4, h(u) = h(w) = 0, h(w_i) = \min\{g(w_i) + g(z_i), 4\}$ for each $1 \leq i \leq k$ and $h(s) = g(s)$ for each $s \in V(G) \setminus \{v, u, w, w_1, \ldots, w_k\}$. It is easy to see that h is a TRDF of G of weight less than $\gamma_{[3R]}(G')$ and the proof is complete. □

Lemma 4. *Let G be a simple connected graph of order $n \geq 3$ and v a vertex of degree at least 2 of G such that*

(i) $N(y) \setminus N[v] \neq \emptyset$ for each $y \in N(v)$,

(ii) there exists a pair α, β of vertices in $N(v)$ such that $(N(\alpha) \cap N(\beta)) \setminus N[v] = \emptyset$,

Then $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 3 + |N_2(v)|$.

Proof. Let $N(v) = \{v_1, v_2, \ldots, v_{\deg(v)}\}$. Without loss of generality, assume that $\alpha = v_1$ and $\beta = v_2$. Moreover, we will assume that the pair α, β is chosen first among the adjacent vertices in $N(v)$. Hence, if $\alpha\beta \in E(G)$, then we assume also that $v_1v_2 \in E(G)$. In addition, let $S = \{v_1, v_2, \ldots, v_k\}$ be one of the largest subset of $N(v)$ containing v_1, v_2 and such that every pair α, β of vertices of S satisfies (ii). According to item (i), let $N(v_i) \setminus N[v] = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{l_i}}\}$ for each $i \in \{1, 2, \ldots, k\}$. Now consider the graph G_1 obtained from G by subdividing the edges vv_1 and vv_2 with new vertices x_1 and x_2, respectively, and for all $i \in \{1, 2, \ldots, k\}$, each of the edges $v_iv_{i_j}, 1 \leq j \leq l_i$, with a new vertex v^{i_j}. We put $T_i = \{v^{i_j} \mid 1 \leq j \leq l_i\}$ and $T = \cup_{1 \leq i \leq k} T_i$. Furthermore, if v_1 and v_2 are adjacent, then we also subdivide the edge v_1v_2 with a

vertex u. Let f be a $\gamma_{[3R]}(G_1)$-function. Without loss of generality, assume that $f(z) \neq 1$ for all vertices $z \in V(G_1)$ by Proposition A. Assume first that $v_1v_2 \in E(G)$. Then, as seen in the proof of Lemma 2, we have $f(v) + f(v_1) + f(v_2) + f(x_1) + f(x_2) + f(u) \geq 6$. Define $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(v) = 4$, $g(v_1) = g(v_2) = 0$, $g(v_{i_j}) = \min\{f(v_{i_j}) + f(v^{i_j}), 4\}$ for all $i \in \{1, 2, \ldots, k\}$ and all $j \in \{1, 2, \ldots, l_i\}$ and $g(x) = f(x)$ otherwise. It is easy to see that g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$.

Assume now that $v_1v_2 \notin E(G)$. By the choice of v_1, v_2, we conclude that S is an independent set. Also to dominate x_1, x_2, we must have $f(v) + f(x_1) + f(x_2) + f(v_1) + f(v_2) \geq 4$. If $f(v) + f(x_1) + f(x_2) + \sum_{i=1}^{k} f(v_i) \geq 5$, then the function g defined on $V(G)$ by $g(v) = 4$, $g(v_i) = 0$ for $i \in \{1, 2, \ldots, k\}, g(v_{i_j}) = \min\{f(v_{i_j}) + f(v^{i_j}), 4\}$ for all $i \in \{1, 2, \ldots, k\}$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Thus, we may assume that $f(v) + f(x_1) + f(x_2) + \sum_{i=1}^{k} f(v_i) = 4$. Hence, $f(v) = 0$ or $f(v) = 4$.

If $f(v) = 0$, then $f(x_1) + f(x_2) + f(v_1) + f(v_2) \geq 6$, which is a contradiction. Thus, $f(v) = 4$ and so $f(x_1) + f(x_2) + \sum_{i=1}^{k} f(v_i) = 0$.

If $\sum_{j=1}^{l_i} f(v^{i_j}) \geq 5$ for some $1 \leq i \leq k$, say $i = 1$, then the function g defined on $V(G)$ by $g(v_1) = 4$, $g(v_{i_j}) = \min\{f(v_{i_j}) + f(v^{i_j}), 4\}$ all $i \in \{2, 3, \ldots, k\}$ and all $j \in \{1, 2, \ldots, l_i\}$, and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Hence, suppose that $\sum_{j=1}^{l_i} f(v^{i_j}) \leq 4$, for each $i \in \{1, 2, \ldots, k\}$. If $2 \leq f(v^{i_j}) \leq 4$ for some $i \in \{1, 2, \ldots, k\}$ and for some $j \in \{1.2, \ldots, l_i\}$, say $i = j = 1$, then define g by

$$g(v_{1_1}) = \begin{cases} \min\{4, 3 + f(v_{1_1})\} & \text{if} \quad f(v^{1_1}) = 4 \\ \min\{4, 2 + f(v_{1_1})\} & \text{if} \quad f(v^{1_1}) = 3 \\ \min\{4, 1 + f(v_{1_1})\} & \text{if} \quad f(v^{1_1}) = 2, \end{cases}$$

$g(v_{i_j}) = \min\{4, f(v_{i_j}) + f(v^{i_j})\}$ when $i_j \neq 1_1$ and $g(x) = f(x)$ otherwise. Clearly, g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Hence, we can assume that $f(v^{i_j}) = 0$ for each i and j. This implies that $f(v_{i_j}) = 4$ for every i and j. In this case, we can define the function g

on $V(G)$ by $g(v) = 3$ and $g(x) = f(x)$ otherwise. Clearly, g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$.

In each of the situation we saw, graph G has a TRDF of weight less than $\gamma_{[3R]}(G_1)$. Moreover, since G_1 was obtained by inserting at most $3+|T| \leq 3+|N_2(v)|$ new vertices, we obtained $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 3+|N_2(v)|$. This completes the proof. □

Lemma 5. *Let G be a simple connected graph of order $n \geq 3$ and v be a vertex of degree at least 2 in G such that*

(i) $N(y) \setminus N[v] \neq \emptyset$ for each $y \in N(v)$

(ii) for every pair of vertices α, β in $N(v)$, $(N(\alpha) \cap N(\beta)) \setminus N[v] \neq \emptyset$.

Then $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 3 + |N_2(v)|$.

Proof. The result follows from Theorem 9 if $\deg(v) \leq 3 + |N_2(v)|$. Hence, assume that $\deg(v) \geq 4 + |N_2(v)|$. Let $N(v) = \{v_1, v_2, \ldots, v_k\}$ and let $M = N(v_1) - N[v] = \{w_1, w_2, \ldots, w_p\}$. It follows from the hypothesis that each $y \in N(v) \setminus \{v_1\}$ has a neighbor in M. Let T be one of the largest subsets of $N(v) \setminus \{v_1\}$ such that for each $T_1 \subseteq T, |N(T_1) \setminus (N[v] \cup M)| \geq |T_1|$. By the definition of $T, |N_2(v)| \geq |M| + |T|$. Moreover, every vertex u in $U = N(v) \setminus (T \cup \{v_1\})$, has a neighbor in M and $N(u) \setminus N[v] \subseteq M \cup N(T)$. Also M dominates $N(v)$ (by item (ii)). We deduced from $4 + |M| + |T| \leq 4 + |N_2(v)| \leq \deg(v) = |T| + 1 + |U|$ that $|U| \geq 4$. If $T \neq \emptyset$, then without loss of generality, let $T = \{v_2, v_3, \ldots, v_s\}$.

Let G_1 be the graph obtained from G by subdividing the edges v_1w_j with new vertices y_j for all $j \in \{1, 2, \ldots, p\}$ and vv_i with vertices x_i for $1 \leq i \leq s + 2$ (when $T \neq \emptyset$) or $1 \leq i \leq 3$ (when $T = \emptyset$). Hence, $|M| + |T| + 3$ edges of G are subdivided. Let f be a $\gamma_{[3R]}(G_1)$-function. Without loss of generality, assume that $f(z) \neq 1$ for all vertices $z \in V(G_1)$ by Proposition A. If $f(v) + f(v_1) + \sum_{i=1}^{s+2} f(x_i) \geq 5$, then define $g : V(G) \longrightarrow \{0, 1, 2, 3, 4\}$ by $g(v) = 4$, $g(v_1) = 0$, $g(w_j) = \min\{4, f(w_j) + f(y_j) \mid 1 \leq j \leq p\}$ and $g(x) = f(x)$ otherwise. It is easy to see that g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Hence, we

assume that

$$f(v) + f(v_1) + \sum_{i=1}^{s+2} f(x_i) \leq 4. \tag{2}$$

If $f(v) = 3$, then it follows from (2) that $f(x_1) = f(v_1) = 0$ which is a contradiction. Consider the following three cases depending on the values of v under f.

Case 1. $f(v) = 4$.
It follows from (2) that $f(v_1) = \sum_{i=1}^{s+2} f(x_i) = 0$. Assume first that $\sum_{i=1}^{p} f(y_i) \geq 5$, then the function $g : V(G) \to \{0, 1, 2, 3\}$ defined by $g(v_1) = 4$, and $g(x) = f(x)$ otherwise, is a TRDF of weight less than $\gamma_{[3R]}(G_1)$. Assume now that $\sum_{i=1}^{p} f(y_i) = 4$. If $f(y_j) = 4$ for some $j \in \{1, 2, \ldots, p\}$, say $j = 1$, then $f(y_j) = 0$ for all $j \in \{2, 3, \ldots, p\}$ and the function $g : V(G) \to \{0, 1, 2, 3, 4\}$ defined by $f(w_1) = \min\{4, 3 + f(w_1)\}$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Now assume that $f(y_j) = f(y'_j) = 2$ for some $j, j' \in \{1, 2, \ldots, p\}$, say $j = 1$ and $j' = 2$, then $f(y_j) = 0$ for all $j \in \{3, \ldots, p\}$. Moreover, the definition of f implies that $f(w_1) \geq 2$ and $f(w_2) \geq 2$. Define the function $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(w_1) = g(w_2) = 3$ and $g(x) = f(x)$ otherwise. It is easy to see that g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Suppose now that $\sum_{i=1}^{p} f(y_i) = 3$. Thus, there exists some $j \in \{1, 2, \ldots, p\}$, say $j = 1$, such that $f(y_1) = 3$ and $f(y_j) = 0$ for all $j \in \{2, 3, \ldots, p\}$. Then the function g defined on $V(G)$ by $g(w_1) = \min\{3, f(w_1) + 2\}$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Now assume that $\sum_{i=1}^{p} f(y_i) = 2$, then $f(y_i) = 2$ for some $i \in \{1, 2, \ldots, k\}$, say $i = 1$, so by the definition of f, $f(w_1) \geq 2$. Then the function g defined on $V(G)$ by $g(w_1) = \min\{3, f(w_1) + 1\}$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Finally, assume that $\sum_{i=1}^{p} f(y_i) = 0$, the $f(w_i) = 4$ for each $1 \leq j \leq p$. Since every vertex of U has a neighbor in M, define $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(v) = 3$ and $g(x) = f(x)$ otherwise. Since every vertex of U has a neighbor in M, we deduced that g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$.

Case 2. $f(v) = 0$.

To dominate x_1, we must have $f(x_1) + f(v_1) \geq 3$. On the other hand, $f(x_1) + f(v_1) \leq 4$ by (2). If $\sum_{i=1}^{p} f(y_i) \geq 2$, then define function $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(v_1) = 4$ and $g(x) = f(x)$ otherwise. Clearly, g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Now let $\sum_{i=1}^{p} f(y_i) = 0$. If $f(v_1) = 4$, then $f(x_1) = 0$ and the function $g : V(G) \to \{0, 1, 2, 3, 4\}$ defined by $g(v_1) = 3$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$.

Now let $3 \leq f(x_1) \leq 4$, then $f(v_1) = f(x_i) = 0$ and so $f(v_i) = f(w_j) = 4$ for each $2 \leq i \leq s + 2$ and $1 \leq j \leq p$, respectively. Define $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(v) = g(v_1) = 4, g(v_{s+1}) = g(v_{s+2}) = 0$ and $g(x) = f(x)$ otherwise. Since for each $s + 1 \leq i$, v_i has a neighbor in $M = \{w_1, w_2, \ldots, w_p\}$ and $N(v_i) \setminus N[v] \subseteq M \cup N(T)$, we deduced that g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$.

Now let $f(x_1) = 2$ and $f(v_1) = 2$. Since $\sum_{j=1}^{p} f(y_j) = \sum_{i=2}^{s+2} f(x_i) = 0$, $f(w_j) \geq 3$ and $f(v_i) = 4$ for each $1 \leq j \leq p$ and $2 \leq i \leq s + 2$, respectively. Define $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(v_1) = 3$ and $g(x) = f(x)$ otherwise. Clearly, g is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$.

Case 3. $f(v) = 2$.

To dominate x_1, we must have $f(x_1) + f(v_1) \geq 2$. On the other hand, $f(x_1) + f(v_1) \leq 2$ by (2). It implies that $f(x_1) + f(v_1) = 2$ and so $f(x_1) = 2$, $f(v_1) = 0$. If $\sum_{i=1}^{p} f(y_i) \geq 3$, then define $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(v_1) = 4$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Assume now that $\sum_{i=1}^{p} f(y_i) = 2$, then $f(y_j) = 2$ for some j, say $j = 1$. Thus, $f(w_1) \geq 2$ and $f(y_j) = 0$ for each $2 \leq j \leq p$. Define $g : V(G) \to \{0, 1, 2, 3, 4\}$, by $g(w_1) = \min\{f(w_1) + 1, 4\}, f(v) = 3$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$. Finally, suppose that $\sum_{i=1}^{p} f(y_i) = 0$, then $f(w_j) = 4$ for each $1 \leq j \leq p$. Define $g : V(G) \to \{0, 1, 2, 3, 4\}$ by $g(v) = 3$ and $g(x) = f(x)$ otherwise, is a TRDF of G of weight less than $\gamma_{[3R]}(G_1)$.

In either case, $\gamma_{[3R]}(G) < \gamma_{[3R]}(G_1)$, and this completes the proof. □

We are now ready to prove our main result.

Theorem 12. *Let G be a simple connected graph of order $n \geq 3$. Then*

$$sd_{\gamma_{[3R]}(G)} \leq 3 + \min\{d_2(v) \mid v \in V \text{ and } \deg(v) \geq 2\}.$$

Proof. If G is a star $K_{1,n-1}$, then $sd_{\gamma_{[3R]}(G)} = 1$, and the result is valid. Hence, assume that G is different from a star. If G has a leaf, then by Proposition 3, the result is also valid. Now, assume that G is a graph with $\delta(G) \geq 2$. According to Lemmas 2, 3, 4, and 5, we have

$$sd_{\gamma_{[3R]}(G)} \leq 3 + \min\{d_2(v) \mid v \in V \text{ and } \deg(v) \geq 2\}.$$

□

The following example of graphs shows that the bound in Theorem 12 is better than the one in Theorem 9. Consider $r \geq 3$ copies of the complete graph K_n with $n \geq 7$, and let x_i be a vertex of the i-th copy of K_n. Let G be the connected graph obtained from the r copies of K_n by adding edges between vertices x_i's so that they induce cycle C_r. One can easily check that by Theorem 9, $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq n - 1$, while by Theorem 12, we have $\mathrm{sd}_{\gamma_{[3R]}}(G) \leq 3 + \deg_2(v) = 5$ for any vertex $v \notin \{x_1, \ldots, x_r\}$.

Let $\delta_2(G) = \min\{\deg_2(v) \mid v \in V \text{ and } \deg(v) \geq 2\}$ and observe that for every vertex v of degree at least two, $\delta_2(G) \leq |N_2(v)| \leq n - \Delta - 1$.

The following two Corollaries are immediate consequences of Theorem 12.

Corollary 13. For any connected graph G with $\delta(G) \geq 2$, $sd_{\gamma_{[3R]}(G)} \leq \delta_2(G) + 3$.

We observe that for a vertex v of degree Δ, $|N_2(v)| \leq n - \Delta - 1$ and thus we obtain the following result.

Corollary 14. Let G be a connected graph of order $n \geq 3$. Then $sd_{\gamma_{[3R]}(G)} \leq n - \Delta + 2$.

References

[1] H. Abdollahzadeh Ahangar, M.P. Alvarez, M. Chellali, S.M. Sheikholeslami, and J.C. Valenzuela-Tripodoro, "Triple Roman Domination in Graphs," *Appl. Math. Comput.*, vol. 391, no. 3, 2021, Art no. 125444.

[2] H. Abdollahzadeh Ahangar, M. Hajjari, R. Khoeilar, Z. Shao, and S.M. Sheikholeslami, "An upper bound on triple Roman domination," *J. Combin. Math. Comput. Combin.*, to be published.

[3] J. Amjadi, "Total Roman domination subdivision number in graphs," *Commun. Comb. Optim.*, vol. 5, pp. 157–168, 2020.

[4] J. Amjadi and N. Khalili, "Quadruple Roman domination in graphs," *Discrete Math. Algorithms Appl.*, 2021, DOI: 10.1142/S1793830921501305.

[5] J. Amjadi, R. Khoeilar, M. Chellali, and Z. Shao, "On the Roman domination subdivision number of a graph," *J. Comb. Optim.*, vol. 40, pp. 501–511, 2020.

[6] J. Amjadi, H. Sadeghi, "Double Roman domination subdivision number in graphs," *Asian-Eur. J. Math.*, 2021, DOI: 10.1142/S179355712250125X.

[7] H. Aram, S.M. Sheikholeslami, and O. Favaron, "Domination subdivision numbers of trees," *Discrete Math.*, vol. 309, pp. 622–628, 2009.

[8] M. Atapour, S.M. Sheikholeslami, and A. Khodkar, "Roman domination subdivision number of graphs," *Aequationes Math.*, vol. 78, no. 3, pp. 237–245, 2009.

[9] R.A. Beeler, T.W. Haynes, and S.T. Hedetniemi," Double Roman domination," *Discrete Appl. Math.*, vol. 211, pp. 23–29, 2016.

[10] M. Chellai, N. Jafari Rad, S.M. Sheikholeslami, L. Volkmann, "Varieties of Roman domination II," *AKCE Int. J. Graphs Combin.*, vol. 17, pp. 966–984, 2020.

[11] M. Chellali, N. Jafari Rad, S.M. Sheikholeslami, and L. Volkmann, "Roman domination in graphs," in *Topics in Domination in Graphs*, T.W. Haynes, S.T. Hedetniemi, and M.A. Henning, Eds. Berlin/Heidelberg: Springer, 2020, pp. 365–409.

[12] M. Chellali, N. Jafari Rad, S.M. Sheikholeslami, and L.Volkmann, "Varieties of Roman domination," *Structures of Domination in Graphs*, T.W. Haynes, S.T. Hedetniemi, and M.A. Henning, Eds. Berlin/Heidelberg: Springer, 2021, pp. 273–307.

[13] E.J. Cockayne, P.A. Dreyer, S.M. Hedetniemi, and S.T. Hedetniemi, "Roman domination in graphs," *Discrete Math.*, vol. 278, pp. 11–22, 2004.

[14] M. Dettlaff, S. Kosari, M. Lemańska, and S.M. Sheikholeslami, "The convex domination subdivision number of a graph", *Commun. Comb. Optim.*, vol. 1, no. 1, pp. 43–56, 2016.

[15] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completness*, San Francisco: W. H. Freeman, 1979, 340 p. ISBN-10: 0716710455, ISBN-13: 978-0716710455.

[16] M. Hajjari, H. Abdollahzadeh Ahangar, R. Khoeilar, Z. Shao, and S.M. Sheikholeslami, "New bounds on the triple Roman domination number of graphs," *J. Math.*, vol. 2022, ID: 9992618, 2022.

[17] T.W. Haynes, S.T. Hedetniemi, and P.J. Slater, *Fundamentals of Domination in graphs*, New York, USA: Marcel Dekker, Inc., 1998, 464 p. ISBN: 0824700333.

[18] N. Meddah, M. Blidia and M. Chellali, "On the 2-independence subdivision number of graphs," *Commun. Comb. Optim.*, vol. 7, no. 1, pp. 105–112, 2022.

[19] S. Velammal, "Studies in Graph Thory: Covering, Independence, Domination and Related Topics," Ph.D. Thesis, Manonmaniam Sundaranar University, Tirunelveli, 1997.

[20] L. Volkmann, "Double Roman domination and domatic numbers of graphs," *Commun. Comb. Optim.*, vol. 3, no. 1, pp. 71–77, 2018.

J. Amjadi, H. Sadeghi

Received October 10, 2020
Accepted September 18, 2021

Jafar Amjadi
Department of Mathematics
Azarbaijan Shahid Madani University
Tabriz, I.R. Iran
E–mail: j-amjadi@azaruniv.ac.ir

Hakimeh Sadeghi
Department of Mathematics
Azarbaijan Shahid Madani University
Tabriz, I.R. Iran
E–mail: h.sadeghi@azaruniv.ac.ir

Workshop on Intelligent Information Systems (WIIS2022)

Vladimir Andrunachievici Institute of Mathematics and Computer Science organizes *the third edition of the Workshop on Intelligent Information Systems (WIIS2022)*, which will be held on **October 6-8, 2022**, in Chisinau, Republic of Moldova.

The workshop encompasses a broad spectrum of intelligent information systems related subjects. It is devoted to discussion of current research and applications regarding development, implementation, and promotion of advanced emergent Information Systems and Technologies.

Topics of interest

Topics of interest include, but are not restricted to the following areas:

- Theoretical Computer Science;
- Formal Models of Computing;
- Information Technologies Applications;
- Decision Support Systems;
- Intelligent Agents and Multi-Agent Systems;
- Knowledge Engineering and Management;
- Medical & Diagnostic Systems;
- Natural Language Processing;
- E-Learning Tools and Systems;
- Cyber Security;
- Intelligent User Interfaces.
- Neural Networks.

On can find the information about the current and previous editions on the Institute's webpage: http://www.math.md/conferences/.

WIIS2022 Organizing Committee

The International Conference on Linguistic Resources and Tools for Natural Language Processing – ConsILR-2022

The Academy of Sciences of Moldova, Chișinău; Vladimir Andrunachievici Institute of Mathematics and Computer Science, Chișinău; Technical University of Moldova, Chișinău; Romanian Academy, by the Institute of Computer Science, Iași branch; the Institute for Artificial Intelligence "Mihai Drăgănescu" Bucharest; the "A. Philippide" Institute of Romanian Philology; "Alexandru Ioan Cuza" University of Iași, by the Faculty of Computer Science; Romanian Association of Computational Linguistics in collaboration with the Technical Sciences Academy of Romania organize *the 17th edition of The International Conference on Linguistic Resources and Tools for Natural Language Processing – ConsILR-2022*, which will be held on **November 10-12, 2022**, in Chisinău, Republic of Moldova and online.

Conferences topics refer to all areas related to Linguistic Resources and Tools for Natural Language Processing. Contributions that range from theoretical, empirical and applied linguistics to computational models, from development of resources to their dedicated use for the improvement of the natural language technology, research related to any language or no one in particular; case studies, demos, and review papers are expected.

One can find the information about the current and previous editions on the conference's webpage:

https://profs.info.uaic.ro/ consilr/2022/.

ConsILR-2022 Organizing Committee