Acta Universitatis Sapientiae

Informatica

Volume 12, Number 2, 2020

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Acta Universitatis Sapientiae Informatica
is covered by the following services:

DOAJ (Directory of Open Access Journals)
EBSCO (relevant databases)
EBSCO Discovery Service
io-port.net
Japan Science and Technology Agency (JST)
Micosoft Academic
Ulrich's Periodicals Directory/ulrichsweb
Web of Science – Emerging Sources Citation Index
Zentralblatt für Mathematik

Contents

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 159–182

DOI: 10.2478/ausi-2020-0010

Development of ATmega 328P micro-controller emulator for educational purposes

Michal ŠIPOŠ
IBM Slovakia, Ltd., branch office Košice
Aupark Tower, Protifašistických
bojovníkov 11, Košice, Slovak Republic
email: michal.sipos@ibm.com

Slavomír ŠIMOŇÁK
Technical University of Košice
Košice, Slovak Republic
email: slavomir.simonak@tuke.sk

Abstract. The paper presents some of our recent results in the field of computer emulation for supporting and enhancing the educational processes. The ATmega 328P micro-controller emulator has been developed as a set of emuStudio emulation platform extension modules (plug-ins). The platform is used at the Department of Computers and Informatics as a studying and teaching support tool. Within the Assembler course, currently, the Intel 8080 architecture and language is briefly described as a preliminary preparation material for the study of Intel x86 architecture, and the Intel 8080 emuStudio emulator module is used here. The aim of this work is to explore the possibility to enrich the course by introducing a more up-to-date and relevant technology and the ATmega is the heart of Arduino – a popular hardware and software prototyping platform. We consider the options to make the process of studying the assembly language principles more attractive for students and using the ATmega AVR architecture, which is broadly deployed in embedded systems, seems to be one of them.

Computing Classification System 1998: K.3.2, C.1.0
Mathematics Subject Classification 2010: 68U20, 68M01
Key words and phrases: emuStudio, emulation, Atmega, Arduino

1 Introduction

Emulation [13] can be described as a technique of imitating a software or hardware product by another software [21]. Emulation currently is widely used mainly as a technique for running a software written for computer system different from the host computer operating environment. It provides the possibility of cross-platform compatibility between different computer systems [31], so it can also be considered as a preservation strategy for digital content [32], [33]. Our motivation behind the development of the ATmega emulator was slightly different however, as it was mainly intended for educational purposes.

At the present time, there are many emulators of computer systems available [34]. Choosing the emuStudio as the platform for which we developed our emulator was straightforward, since it provides many features, which are essential for its successful application in educational process. emuStudio is a platform for emulation of computer architectures that integrates, as a form of an IDE, also source code editing, compiling and debugging features.

Within the platform, programs for emulated machines are usually written using assembly language of the particular architecture. Another significant advantage of the emuStudio is the fact that it does not only serve as a one-purpose emulator. Essentially, it provides a framework[1] in the form of a Java API. By utilizing it, programmers are enabled to design and implement their own computer emulators as a set of plug-in modules. The above-mentioned framework is intended to help to standardize the process of emulation [11], i.e. to define the key responsibilities, functionalities and types of components that are common for most emulators. In particular, the task of a programmer is to implement modules for assembler source code compilation, CPU and memory emulation, but optional peripheral devices can also be emulated.

With its ability of illustrative exploring the internal operation of emulated architecture, emuStudio is well suited for educational purposes. The platform has been designed to be easily extendable and once a component is implemented, it can be reused effectively. As a result, possibilities of enriching the emuStudio by new modules are very broad.

At the time of writing this paper, the emuStudio is used at the Department of Computers and Informatics in Assembler and Data Structures and Algorithms courses. In the Assembler course, Intel 8080 emulator is utilized and within the Data Structures and Algorithms course, emulators of RAM and RASP [29] abstract machines are used. The 8080 as a predecessor of x86 ar-

[1] `http://www.emustudio.net`

chitecture and a relevant example of an 8-bit processor architecture is used as a simple preparation for studying Intel x86 architecture and language. However, being introduced to the microprocessors market in 1974 [20], despite of its impact on the industry, it can cause some students to become slightly demotivated by using a less up-to-date technology.

In the thesis [18], an idea of incorporating Arduino with its ATmega microcontroller instead of the Intel 8080 in the assembly language course curriculum has been investigated. Some similarities between Intel 8080 and ATmega 328P assembly languages has been observed and the possibilities of such substitution are discussed there. Furthermore a simple library providing students with the possibility to use basic input and output operations has been developed within the thesis and further enhanced later.

2 Related work

ATmega is the core of the popular Arduino Uno platform[2]. In recent years, Arduino has been introduced to several computer science courses at the Department of Computers and Informatics. This tendency has also been stimulated by the growing demand for knowledge in the area of *embedded systems* [1]; the popularity of the term *Internet of Things* [19] can be observed for being rapidly increasing, too.

Globally, we can see that teaching assembler courses at some universities is continually being shifted aside in favor of new subjects. Nowadays, as the author of [17] suggests, operating systems and virtual machines on the top of them pose a certain problem for teachers and students when it comes to studying internal operation of computer systems as they act as a form of a shield that hides the inner mechanisms. It can be concluded that it is one of the causes for the phenomena of leaving assembler courses out from the curriculum that students might not see enough reasons for studying it.

On the other hand, when using embedded systems, a virtual machine as an abstraction layer is often missing and direct access to hardware is much easier, but it requires some knowledge of its internal operation. And this is where, in addition to C language, assembler becomes much more relevant. The topic is deeper discussed in [17].

An approach similar to the one presented in this paper (but based on 8051 micro-controller emulation) has been taken with the EdSim51 - the 8051 simulator for teachers and students [25]. The EdSim51 is a simulator of 8051

[2] `https://store.arduino.cc/arduino-uno-rev3`

micro-controller, which is interfaced with virtual peripheral devices like keypad, DC motor, 7-segment display, UART, LEDs, etc. A nice advantage of the EdSim51 over some other available 8051 simulators is that it provides graphical representation of several peripheral devices, which can be used interactively [26]. It is a Java-based application, so it can be used in multiple operating environments.

Another example of very popular processor within the computer architecture academic community and one often utilized for educational purposes [22] is MIPS. Several simulators have been developed to date for this architecture [37]. We can mention DrMIPS, an educational MIPS simulator [23, 22], which can simulate the execution of an assembly program and display the datapath graphically. Moreover it can display the values of inputs and outputs of several components, which are relevant for the execution of the current instruction. The simulator was developed in Java and is available not only for PC, but also for Android devices. On the other hand, since the emulator is developed mainly for educational purposes, it supports rather limited set of instructions. Several instructions like syscalls, floating-point operations and shifts are not supported [23].

CPUlator is a Nios II, ARMv7, and MIPS computer system simulator and debugger running in a web browser [37]. It allows running and debugging programs without corresponding hardware board. Systems simulated by CPUlator are based on the computer systems from the Altera University Program (Nios II, ARMv7) and SPIM (MIPS) respectively.

Gerd's AVR simulator [27] is a complex solution for simulating AVR 8-bit micro-controllers. It provides an editor, assembler, simulator, overview of I/O ports and timers, memories, etc. Lazarus Pascal source code and executable 64-bit versions for Windows and for Linux are available from [27]. The advantage of this solution is the support for many types of AVR 8-bit micro-controllers and complex support of built-in peripheral devices. As an advantage of our solution, described in this paper, can be considered the fact that it is developed using Java and thus could be more portable. It is developed as a set of plug-in modules for emuStudio emulation platform, so it could be easier to develop and enhance in the future. Modules for some simple external devices are also available within our solution (like USART terminal and LED diode emulation modules).

An interesting approach has been employed in [24], where a methodology is proposed for teaching the microprocessors interface course based on the idea of emulating the microprocessor operation. Students are instructed to use the parallel port for emulating the signals generated by 8088/8086 micro-

processors. According to authors of the paper, emulating a working system by generating the necessary control signals leads to a good knowledge of the system.

An educational approach for bridging the gap between low-level and higher level programming, based on usage of 8-bit microcontrollers has been proposed in [6]. The approach proposed aims in simplification the students' learning by making the parallelization between the assembly language programming and higher level programming.

But what is the purpose of using an emulator in addition to original hardware? When studying the above-mentioned inner workings of computers, emulation tools can be a way of a deeper insight into them [36], as the complexity of hardware might sometimes exceed the limit of what can be effectively studied or taught [38], respectively. Also, emulator gives students the opportunity to exercise what they have learned at the classes, at home, without owning the hardware components physically. What is more, emulating an ATmega device in software brings another advantage – it can be used to automate the testing of students' assignments, as has also been suggested in [18].

In addition to using emulators, there are also other efficient approaches that are used within the education in the field of computer architecture and organization, such as hardware-description languages and reconfigurable circuits [7, 16]. An interesting case study is described in [8] where critical investigation of existing course on digital electronics revealed that it mostly produces surface understanding of digital systems and students lack practical skills to develop complex digital designs. The course outline has been improved and an enhanced delivery method was proposed. The study of students' performance over a period of six years was evaluated and the results indicate that students had developed good level of understanding of basic principles and were able to employ system modeling using VHDL.

3 Emulation techniques

Before proceeding to description of the design of ATmega emulator modules set, several emulation-related terms will be introduced. According to the study [21] two basic types of emulation can be distinguished, as they correspond to the format of the emulated program:

- *interpretation* – the program is represented in the form that is native for the original processor it has been written for; this enables the emulator, in this case *interpreter*, to work in the "fetch-decode-execute" loop,

- *binary translation* – the code of the program is translated into the form native for the architecture it is being emulated on, which usually means x86-like binary. The translation, can either take place at runtime (dynamic) or in advance, before emulation (static).

In the study also advantages and disadvantages of both of the approaches are mentioned. Interpretation is generally less complex to implement than binary translation. On the contrary, as during binary translation a native code is generated, the resulting performance of the emulation is usually better. One of the techniques to increase the performance of interpreter emulators is so-called *threaded code*. James R. Bell explains its basic principles in his paper [5]. The algorithm begins by reading the instruction at the PC-th address from the emulated program storage, where the PC is the program counter register. The retrieved opcode serves as an index into a so-called jump table. Each particular table entry contains a reference to the function that implements the corresponding instruction emulation. The only step left is to call the function, which is analogous to executing the instruction on the emulator. The jump table can be filled by function references in advance, when the emulator is initialized, which unburdens the emulation loop from the time consuming task of decoding instruction opcodes.

By performing the above-mentioned procedure, we have eliminated the effort needed for the *decode* step from the *fetch-decode-execute* loop. In addition to this, it is also possible to minimize the *fetch* part, if we cache the instructions already fetched from the memory. These optimizations lead to the algorithm presented in Figure 1 [30].

4 ATmega 328P micro-controller architecture

ATmega 328P micro-controller is based on the AVR architecture, which is one of the leading 8-bit architectures [4]. From instructions complexity point of view, it is a RISC (Reduced Instruction Set Computer) architecture [4]. From the perspective of an emulator programmer, this can be considered as an advantage as it is characteristic for RISC computers to have simpler instructions. The fixed-length instructions also make the decoding part of the execution loop less complex. On the other hand, from the point of view of the one developing programs for the micro-controller, the CISC (Complex Instruction Set Computer) addressing modes flexibility might be missing. Also, a program size might be bigger in case of RISC binary [35] in comparison to a

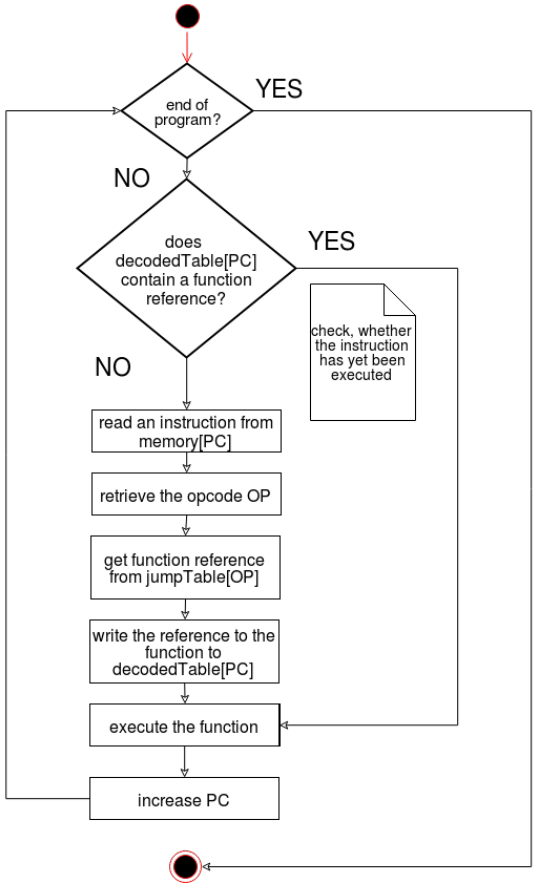


Figure 1: *Threaded code* execution algorithm.

CISC one as for one operation, a sequence of several simpler instructions can be required.

ATmega is a representative of Harvard architecture [4], which affects the structure of the memory subsystem – separate modules are used for storing program and data. When compared to von Neumann architecture, Harvard computers enable more effective pipelined execution [9] – while one instruction is being executed and is retrieving its operands from the data memory, the next one can be pre-fetched from the program memory module.

The ATmega 328P micro-controller is equipped with a 32 kilobyte flash program memory [2]. It is organized in a less conventional way – each memory cell contains an instruction word, i.e. two bytes. The reason for such a distribution

[2] is that the length of all opcodes is either 16 or 32 bits. As a program is addressed by words, program counter is 14 bits wide, which is enough for the whole range.

In case of the data memory, addressing is conventional – one byte for one cell. ATmega chip includes a SRAM (Static Random Access Memory) data memory module with all general purpose registers (GPR) and input/output registers (IOR) mapped to its address space. The organization is depicted in Figure 2 and the capacity of the internal SRAM, i.e. excluding the GPRs and IORs, is 2 KB.

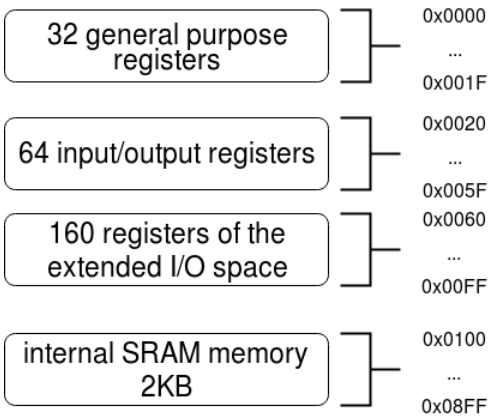


Figure 2: Data memory organization.

An inevitable part of the micro-controller is the input/output subsystem. On the ATmega, it includes digital and analog input/output pins, USART (Universal Synchronous-Asynchronous Receiver/Transmitter) serial interface, timers and a lot of other peripherals. The peripheral devices communication with the micro-controller is realized by using of so called *ports*. They serve as *gates* [28] between the CPU core and the other parts of the micro-controller, or between the CPU and devices out of the chip. These might be various types of sensors or mechanical equipment, e.g. servo motors or relays. For digital input/output, there are three ports – B, C and D available. Each of those ports is controlled by three registers: the DDR register for determining whether the pin has INPUT or OUTPUT direction, the PORT register for setting the pin to HIGH or LOW level, and the PIN register for reading the state of INPUT pins (or toggling the value of particular PORT bit by writing '1' to the corresponding bit of the PIN register) [2]. So to send data e.g. to port B, writing to the register PORTB is required. Each of its bits represents

the corresponding pin of the port. To read its current value, the PINB register can be read and the relevant bit needs to be evaluated.

5 ATmega 328P - emuStudio extensions design

The initial step to consider when designing the support for a new architecture in emuStudio is so called *abstract scheme*[3]. The scheme is always based on the von Neumann model [12], which represents a certain complication in our design as the ATmega is a computer of the Harvard type – it contains separate memory modules for data and for program. Furthermore, also EEPROM (Electrically Erasable Programmable Read-Only Memory) memory module for permanent data storage is included on the chip.

5.1 Memory emulation

There has been a discussion with the author of the emuStudio platform regarding the memory subsystem, in which several possible solutions has been considered. One of them was to extend the emuStudio to support Harvard architecture computers by adding a new type of component in the abstract scheme – program memory. However, such component would be limited only to store programs, which would mean to narrow the functionality of a storage component only for this purpose. Another considered option was to use the concept of a Memory Management Unit (MMU) that would provide access to all the three memory modules. This would, however, require them to be mapped to a single memory space.

Finally, the solution reducing Harvard architecture of ATmega to von Neumann type has been chosen. To perform this, one of the memory modules needed to be considered as *memory*, since von Neumann computers, and therefore also emuStudio abstract scheme include at most one memory. For practical purposes, such as the fact that the *compiler* module needs storage to load a program to and also that the *CPU* needs it to fetch instructions, flash program memory has been chosen.

For the data memory, as it could not be considered as *memory* after this decision, it has been chosen to include it within the *CPU* module. Such an organization is a compromise. Both of the memory modules are integral parts of the chip on a real device, so including data memory in the *CPU* module is acceptable. As for the program memory, it has been chosen, from the logical

[3] `http://www.emustudio.net/docuser/main_module/index/`

point of view of a von Neumann computer, as the only memory in the scheme. EEPROM data permanent storage is not implemented within our solution yet, however, in the future, it can be added as a *device*, i.e. peripheral device type of module.

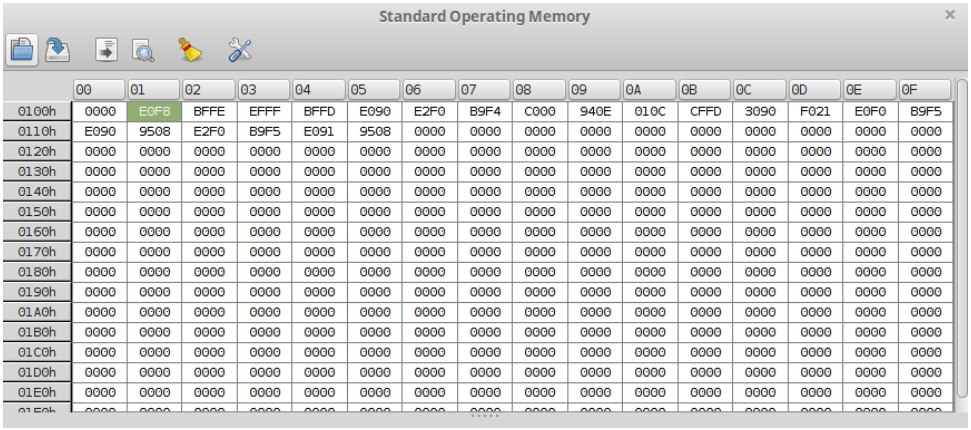
For the purpose of storing program, *Standard Memory*, an already implemented emuStudio module has been chosen. This component has been part of MITS Altair 8800 computer emulator and the fact makes it a sufficient candidate for reuse. Though, we had to adapt it for two-byte cells addressing. Updates in *Standard Memory* module could have been done, for example, by designing a new *context* to support different cell sizes. This would, however, require changes in the *emuLib*[4] library. Also, an already existing utility *HexfileManager*, responsible for loading contents of an Intel HEX files into memory, would be needed to be adapted as it currently depends on memories with one byte at a cell.

Therefore, we had to choose a solution with as little changes in *Standard Memory* as possible. It has been decided to adapt just the visual representation of the memory content in the GUI (Graphical User Interface). As a result, internally, in our solution, the content is still an array of one byte values, while in the GUI, user sees two bytes at a cell. The interface of the *StandardMemory* context has been extended by the option to set the cell size by a programmer. He or she only needs to do this configuration when requesting it from the *context pool*. Also, endianness can be set. The resulting view of the memory content can be seen in Figure 3.

5.2 Compiler module

Another module that is part of the set presented in this paper is the *compiler* module. All compilers currently present in emuStudio use automation tools to generate lexical (lexer) and syntactic (parser) [10] analyzer. For lexical analysis, JFlex[5] is used. This phase of compilation is also needed for the purpose of syntax highlighting in the emuStudio source code editor. Thanks to lexical analyzer generated on the basis of the JFlex specification file, the emuStudio main module takes care of this task, we only needed to specify the types of tokens. This fragment from the JFlex file illustrates it:

[4] `https://github.com/vbmacher/emuLib`
[5] `http://jflex.de/`



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0100h	0000	E0F8	BFFE	EFFF	BFFD	E090	E2F0	B9F4	C000	940E	010C	CFFD	3090	F021	E0F0	B9F5
0110h	E090	9508	E2F0	B9F5	E091	9508	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0120h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0130h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0140h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0150h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0160h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0170h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0180h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0190h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
01A0h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
01B0h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
01C0h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
01D0h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
01E0h	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Figure 3: GUI of *StandardMemory* after adding a support for different cell sizes.

```
"ADD" {
    return token(Token.RESERVED);
}

".DB" {
    return token(Token.PREPROCESSOR);
}
```

For the core task of compiling assembler source code of programs written for ATmega, we call an external tool – GAVRASM[6] from the compiler module. From Java code, it is invoked by utilizing *ProcessBuilder* provided by the Java API:

```
ProcessBuilder processBuilder = new ProcessBuilder(command);
Process process = processBuilder.start();
process.waitFor();
```

The `command` depends on the underlying operating system, since the GAVRASM is available for MS Windows as well as for Linux OS.

[6] `http://www.avr-asm-tutorial.net/gavrasm/index_en.html`

5.3 CPU emulator module

Within this subsection we will continue to the core part of the emulator – the *CPU* module. The basic algorithm of a CPU emulator in general is presented in [21]. The diagram is depicted in Figure 4. The algorithm will also be applied in our solution.

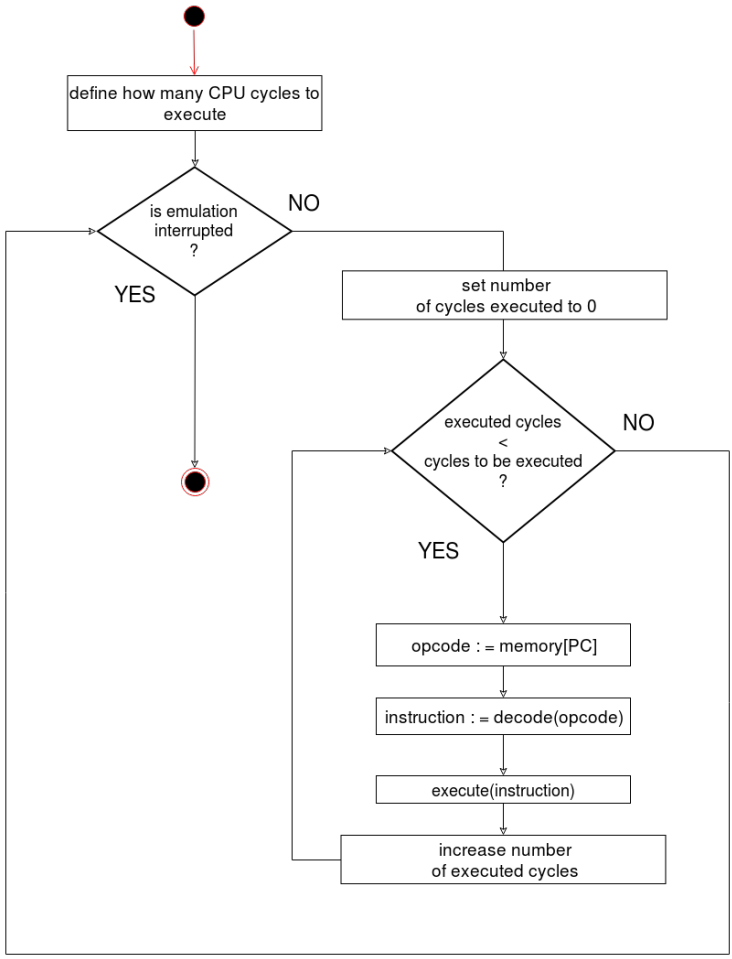


Figure 4: General CPU core emulator algorithm.

Let us explore the steps in deeper detail and discuss how they will be implemented. At first, we need to determine how many CPU clock cycles will be executed. For this purpose, it is necessary to define a *synchronization interval*

for which the number will be calculated. The time range will be figured out by the following formula [15]:

$$\mathsf{timeSlice} = \mathsf{T} \cdot \mathsf{numberOfCycles}\,. \tag{1}$$

This will be equal, as the formula suggests, to the time of execution of *numberOfCycles* clock cycles on the real CPU; T is the clock cycle period, i.e. multiplicative inverse of the CPU frequency. Therefore, *timeSlice* will be calculated according to this formula:

$$\mathsf{timeSlice} = \frac{\mathsf{numberOfCycles}}{\mathsf{f}}\,. \tag{2}$$

The *numberOfCycles* value can be then calculated as follows:

$$\mathsf{numberOfCycles} = \mathsf{timeSlice} \cdot \mathsf{f}\,. \tag{3}$$

Now, what is still missing is the *timeSlice* value. It can be an empirically chosen constant [15]. In the Intel 8080 emuStudio extension, it is 100 ms (0.1 s). We will also use this time range and assuming that clock frequency of ATmega 328P is 16 MHz[7], the following formula holds:

$$\mathsf{numberOfCycles} = 0.1 \cdot 16 \cdot 10^6 = 16 \cdot 10^5\,. \tag{4}$$

The next step is to determine whether the number of executed cycles is less than how many are to be executed at all within the 100 ms interval. To evaluate this condition, we need to know the number of clock cycles of specific instructions. These are constant values and can be obtained from the AVR Instruction Set Manual [3]. To keep the number of cycles executed so far, we need to return the number of clock cycles that the instruction execution took from the `execute(instruction)` method.

The following part of the algorithm is reading the instruction `opcode` from the program memory. After fetching the instruction, we need to decode it. This can be done by using an extensive *switch* statement for all the 131 instructions, not even counting all the possible combinations of opcodes that have their operand encoded in them. The idea is illustrated by the following code fragment:

[7] `https://www.arduino.cc/en/Products/Compare`

```
switch(opcode){
 case 0x0:
 execute(instruction1);
 case 0x1:
 execute(instruction2);
 ...
}
```

In contrast to this, it is more efficient to use the concept of a *jump table*, that has been introduced in the *Introduction* section and can be seen in the Figure 1. In terms of Java, which is the implementation language of our solution, we can use so called *functional interface* for this purpose, let us call it `ExecutableInstruction`, with one method – `execute(Short[] opcodeWord)` that takes a two-bytes opcode and returns the number of CPU clock cycles executed by the particular instruction. The *jumpTable* needs to be initialized. It is relevant to do so outside of the emulation loop itself and have it prepared in advance. The number of all possible distinct opcodes is

$$\mathrm{numOfDistinctOpcodes} = 2^{16} = 65536\,. \tag{5}$$

During the emulator initialization, we will go through all these possibilities, i.e. from 0 to 65535, and in all the *case* branches of the *switch* statement, we will apply assignments in the following form:

```
executableInstructions[OP] = EmulatorEngine.this::add;
```

As we can see, a reference to the method `add()` that implements emulation of the ADD instruction is put at the *OP-th* position in the `executableInstructions[]` array, which represents the jump table and the *OP* is the operation code of the instruction. The last step of the algorithm is to execute the instruction at the *OP-th* item of the *jump table*. As we have already mentioned, the table items are of the type `ExecutableInstruction` and executing the instruction means calling its `execute()` method:

```
cycles = executableInstructions[opcode].execute(opcodeWord);
cyclesExecuted += cycles;
```

However, one additional adjustment is needed. If we have already reached the number of cycles to be executed but the time range of 100 ms still has not passed, we have to *wait* for the remaining part of the time interval. For this purpose, Java API provides a utility method to stop current thread – emulation thread (in emuStudio, there is a separate thread for emulation):

```
LockSupport.parkNanos(timeSliceNanos - endTime);
```

5.4 Input/output subsystem emulation

As one of suitable representatives of the input/output subsystem of ATmega we have chosen the USART serial interface. We will not explain its functionality and operation here, more can be found in the analytic part of the Master's thesis [30] and also in the manual [2]. USART interface is usually used for communication between the micro-controller and external peripheral devices, or even an another micro-controller. A frequent application is also data exchange between ATmega and a personal computer via a *terminal*.

On a real ATmega device, the CPU core communicates with the USART module on the chip via data bus. On the other side, USART is connected with the outside world via RX and TX micro-controller pins. Communication with the PC is enabled by a special one-purpose integrated circuit that converts the data stream between the format used by USART and USB (Universal Serial Bus) format. The authors of article [14] explain that a suitable level of abstraction must be agreed on when implementing emulators. One-purpose auxiliary chips together with buses and also, in our case, RX and TX pins as communication channels can be omitted in the emulator design.

As a result of abstracting away from these hardware details, two components in the abstract scheme – *USART* and *Terminal* modules can be used in our solution. The design of the scheme is depicted in Figure 5.

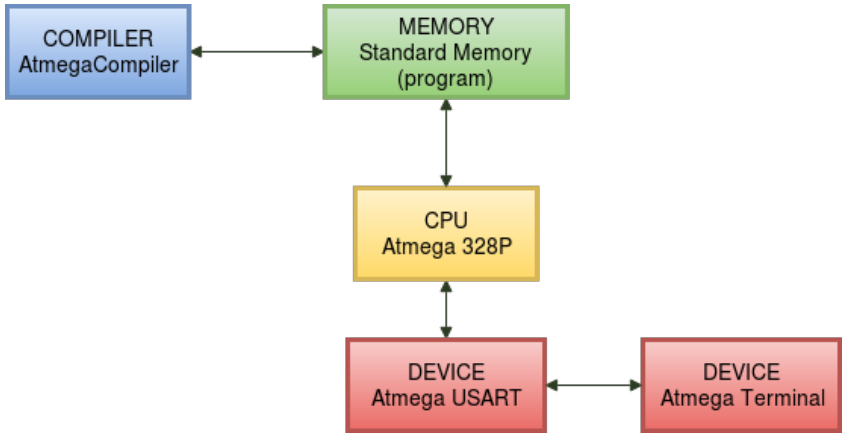


Figure 5: Abstract scheme of the solution.

As we have already stated, bit-after-bit communication via RX and TX pins will not be emulated exactly as on a real device. We will use a different approach here. In the section 4, we explained that I/O registers are mapped

into the SRAM data memory space and this can be effectively exploited also in our solution.

emuStudio modules use a special component – *context*, that serves as their communication interface to exchange data and commands with the other modules. In our implementation, all the SRAM data memory is integrated in the *CPU* module. We will use the context of *CPU* in modules of peripheral devices to subscribe them to observe changes in I/O registers. In the case of USART module, the most important register is *UDR0*. When writing or reading from it using `OUT` and `IN` instructions, on a real device it is used as a temporary storage for the byte (character) being transmitted or received. For the subscribing, the `CPUContext` interface provided by the *emuLib* library must be extended, as it can be seen in Figure 6. Here, *UDR0* will be the device context, in our case.

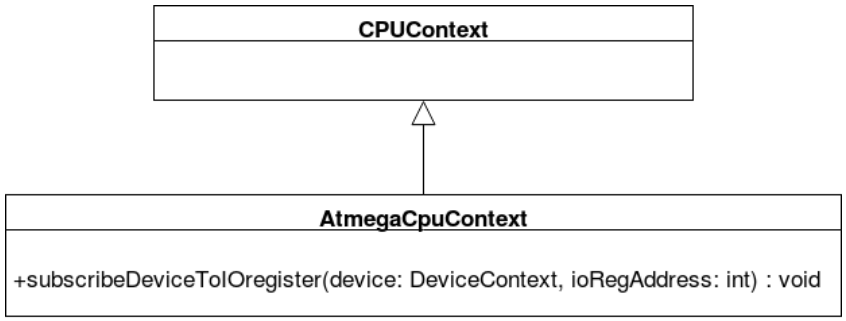


Figure 6: The CPUContext extension.

The interconnection of the *CPU* and *USART* modules via subscribing to changes in I/O registers (mapped to the data memory, which is integrated in the *CPU* module) can be seen in Figure 7.

On the other side of communication, *USART* and *terminal* modules are interconnected, again, by using their contexts, as the component diagram in Figure 8 depicts.

Graphical user interface of the terminal implemented in our solution can be seen in Figure 9.

For the purpose of effective testing of students' assignments, automatic emulation support has been added to the *terminal* module. The functionality is enabled by redirecting input from a text file with prepared inputs into the *terminal* and outputs are then written to a separate output file instead of being printed to the terminal GUI. Automatic emulation without GUI and user interaction is already one of the emuStudio features, so it what was only

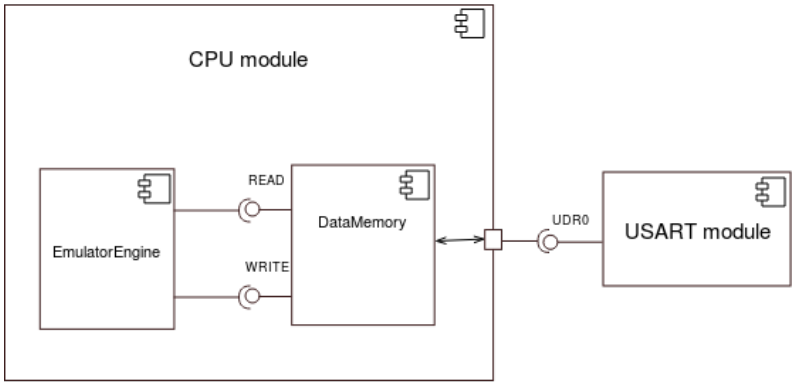


Figure 7: Interconnection of CPU and USART modules.

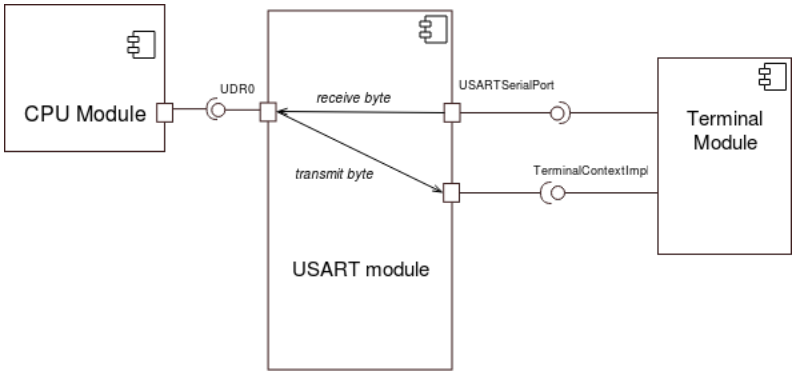


Figure 8: Interconnection of CPU, USART and terminal modules.

needed to add support for it in the *terminal* module. To run the emulation using USART and terminal in the automatic mode, the following command can be used:

```
java -jar emuStudio.jar --auto --nogui --config "AVR"
  --input example.s
```

What needs to be specified is the name of the file with source code of the program. Path to the file with input values and also to the file where outputs will be redirected, can be set in the *terminal* module settings.

Similar concept of observing specific I/O registers for changes has also been used in another extension – LED (Light-Emitting Diode) diode emulation module. It visualizes the communication via digital ports of the micro-

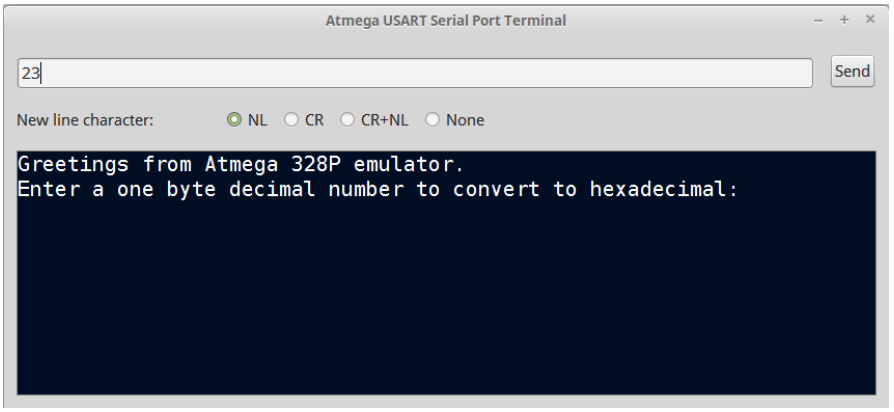


Figure 9: Graphical user interface of the USART terminal.

controller. However, now, the LED module needs to subscribe to a specific pin of a digital port. It is represented by the corresponding bit in the PORTx register, where "x" stands either for B, C or D. To enable this, one more operation needed to be added to CPU context extension – the resulting interface is depicted at diagram in Figure 10.

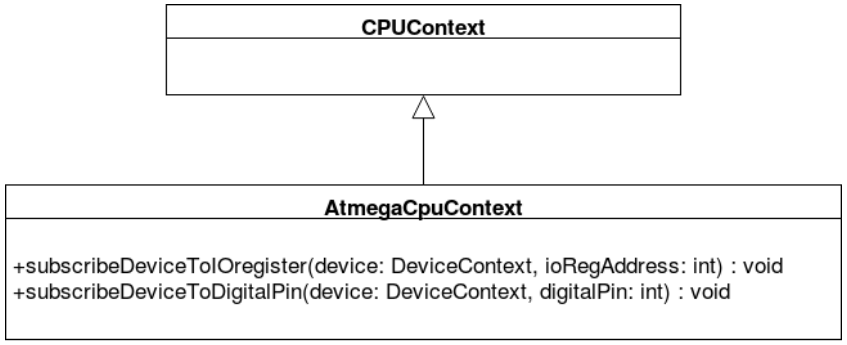


Figure 10: The CPUContext extension for pin subscribing.

One more peripheral device module has been implemented within our solution – the timer. ATmega chip includes three timers [2]; since our emulator is mainly intended for usage as a study supporting tool, one of them is sufficient to be emulated at the moment – *Timer/Counter0*. Again, timer uses, similarly to USART, its dedicated I/O registers. Just to mention some of them, *TCNT0* register holds current counter value and *TIMSK0* is used to

enable/disable timer interrupts. As in the case of the USART module, we can apply the concept of subscribing the timer to relevant I/O registers within the *data memory*, which is included in the *CPU* module.

According to the documentation[8], within emuStudio main module, there is a dedicated execution thread for emulation. It is not suitable to burden it by other auxiliary tasks. One of these is the process of counting timer/counter cycles. Therefore, we should create a separate thread for this purpose. It will take care of updating the counter register *TCNT0* and check for situations when it should signal timer interrupts. One of them is timer overflow interrupt. Since the *Timer/Counter0* uses an 8-bit counter, the highest value is therefore 255 and in the clock cycle when the counter reaches it, interrupt signal is generated and sent to the CPU. To configure how often it will occur, *TCCR0B* register can be used – by assigning it a specific value, the *prescaler* will be set accordingly. Prescaler serves as a frequency divider – it divides 16 MHz frequency of ATmega micro-controller by given constant. E.g., if we set *TCCR0B* to 0b101, the divisor will be 1024 and the resulting frequency of the timer will be equal to 15625 Hz. As a result, the period between timer ticks will be longer and overflow of the counter register will be less frequent.

Moving on to the interrupts signalization. For this purpose, it is enough to keep a flag boolean variable – if its value is TRUE, it means that a pending, not yet handled interrupt is present. We need to, however, ensure synchronized access to it from both threads that use it – emulation and timer thread.

In each emulation step, it is then required to check the flag variable and if it is set to TRUE, it is needed to execute corresponding interrupt handling subroutine. Interrupts currently supported by our CPU emulator are *Timer/Counter0* overflow interrupt and *Timer/Counter0* compare match interrupt that occurs when *TCNT0* counter register becomes equal to either of two registers values set by the programmer – *OCR0A* or *OCR0B*. If an unserved interrupt is present, program counter register is set to the corresponding vector address. The algorithm of handling the timer interrupts is depicted in Figure 11.

6 Conclusion

In this paper we presented a set of extension modules implemented for emuStudio platform that provide support for emulation of ATmega 328P microcontroller. During the design and implementation, thematic areas covered in

[8] `http://www.emustudio.net/docdevel/emulator_tutorial/index/`

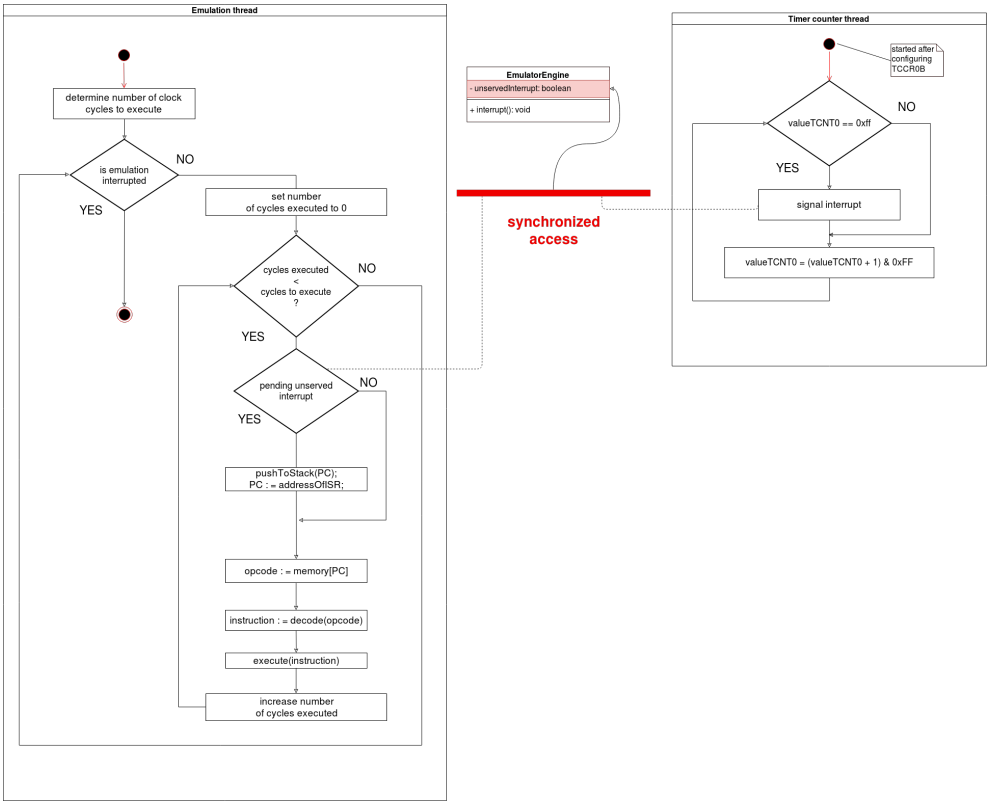


Figure 11: Algorithm of signaling interrupts from the timer.

a collection of exercises that was a part of thesis [18] have been taken into account as relevant for application in the Assembler course. Therefore the areas included work with USART serial interface, digital output, timer and interrupts handling.

As a result, the set of extensions includes the following modules:

- module for assembler source code compilation using an external tool (GAVRASM),
- module for emulation of ATmega micro-controller CPU core with integrated SRAM data memory with general purpose and I/O registers mapped into its address space; also a disassembler is included for the purpose of more comfortable program debugging,
- module emulating USART serial interface,
- USART terminal module,

- module emulating *Timer/Counter0* – one of three ATmega timers, including interrupt generation in cases of counter overflow and counter match,
- LED diode emulation module to visualize voltage changes on digital pins of the micro-controller.

As a program storage (ATmega flash program memory), an already existing module *Standard Memory* has been reused. What needed to be adjusted was the visual representation of values in memory cells as the ATmega program memory is addressed by words (pairs of bytes). In comparison with the official Atmel Studio IDE[9] from the manufacturer of the micro-controller, it removed the source of misinterpretation of addressing – the Studio displayed one byte per cell.

As for the CPU emulation, the whole ATmega 328P instruction set is supported in our solution, except for instructions for switching the device into sleep mode (SLEEP), breaking program execution for debugging purposes (BREAK), resetting watchdog timer which is not implemented (WDR) and writing into the program memory as we do not support program self-modification (SPM).

From peripheral devices, USART and terminal modules can be used in mathematical and text-oriented exercises for the purpose of providing inputs and retrieving outputs from the programs. To support students' assignments testing, automatic emulation feature has been added to the terminal module, too. Students can also practically learn how to handle interrupts thanks to available timer/counter module. As a reaction to timer events, the LED diode module can be effectively used. It also can visualize the voltage changes on the ATmega digital pins.

The solution presented within the paper can be practically applied in the Assembler course at the Department of Computers and Informatics as a studying and teaching support tool. It would help the students to better understand the internal operation of the ATmega 328P micro-controller.

However, lot of the features of the ATmega micro-controller is currently not included in our solution. To fulfill the goal of making the Assembler course more attractive, there is still a large scale of possibilities of how our set of emuStudio modules can be further improved and extended. What could be added is the support for EEPROM permanent data storage memory that has not been implemented within our solution. Within the available timer/counter

[9] `http://www.microchip.com/avr-support/atmel-studio-7`

module, additional modes of operation could be implemented. The next alternative is the emulation of additional peripheral devices, e.g. the remaining two timers/counters. Also, an input on digital pins of the micro-controller could be considered in addition to currently supported digital output.

Some from the available external peripheral devices could be emulated as well, like an LCD display or boards for network communication. In addition to these, servo motors, relays and various sensors could be emulated, too. Our solution, as explained in the text above, is open for further development and extension to continue with the effort of making the educational process at our department more attractive for students by supporting their interest in the field of machine-oriented languages and computer organization.

References

[1] N. Ádám, Interconnection of computer and software engineering courses (Prepojenie predmetov počítačového a softvérového inžinierstva), *Proceedings of the 10th Workshop on Intelligent and Knowledge oriented Technologies WIKT 2015*, Center of Business Informatics, FEI TUKE, 7 2015. ⇒161

[2] Atmel Corporation, Atmega328/P – datasheet complete, 2016. ⇒165, 166, 173, 176

[3] Atmel Corporation, AVR instruction set manual, 2016. ⇒171

[4] Atmel Corporation, AVR microcontrollers for high-performance and power-efficient 8-bit processing, 2013. ⇒164, 165

[5] J. R. Bell, Threaded code, *Communications of the ACM* **16**, 6 (1973) 370–372. ⇒164

[6] D. E. Bolanakis, G. A. Evangelakis, E. Glavas, K. T. Kotsis, A teaching approach for bridging the gap between low-level and high-level programming using assembly language learning for small microcontrollers, *Computer Applications in Engineering Education* **19**, 3 (2011) 525–537. ⇒163

[7] F. Cancare, D. B. Bartolini, M. Carminati, D. Sciuto, M. D. Santambrogio, On the Evolution of Hardware Circuits via Reconfigurable Architectures, *ACM Trans. Reconfigurable Technol. Syst.* **5**, 4 (2012). ⇒163

[8] C. V. Eguzo, B. J. Robert, O. C. Ihemadu, P. A. Avong, Integrating hardware descriptive language (HDL) in teaching digital electronics-a case of Nigerian polytechnics, *2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON)*, Owerri, 2017, pp. 650-655. ⇒163

[9] R. Eigenmann, D. J. Lilja, Von Neumann Computers. John Wiley & Sons, Inc., 2001. ⇒165

[10] S. Chodarev, J. Porubän, Development of custom notation for XML-based language: a model-driven approach, *Computer Science and Information Systems (ComSIS)* **14**, 3 (2017) 939–958. ⇒168

[11] P. Jakubčo, M. Domiter, Standardization of computer emulation, *Applied Machine Intelligence and Informatics (SAMI), 2010 IEEE 8th International Symposium*, IEEE, 2010, pp. 221–224. ⇒160

[12] P. Jakubčo, S. Šimoňák, emuStudio - a plugin-based emulation platform, *Journal of Information, Control and Management Systems* **7**, 1 (2009) 33–45. ⇒167

[13] P. Jakubčo, S. Šimoňák, Utilizing GPGPU in computer emulation, *Journal of Information and Organizational Sciences* **36**, 1 (2012) 39–53. ⇒160

[14] P. Jakubčo, S. Šimoňák, N. Ádám, Communication model of emuStudio emulation platform, Acta Univ. Sapientiae, Informatica **2**, 2 (2010) 117-134. ⇒173

[15] P. Jakubčo, L. Vokorokos, Preserving host independent emulation speed, *CSE'2010 International Scientific Conference on Computer Science and Engineering*, Department of Computers and Informatics, FEEI, Technical University of Košice, 2010. ⇒171

[16] B. Madoš, Z. Bilanová, E. Chovancová, N. Ádám, Field Programmable Gate Array Hardware Accelerator of Prime Implicants Generation for Single-Output Boolean Functions Minimization, *ICETA 2019 - 17th IEEE International conference on emerging elearning technologies and applications*, Starý Smokovec, Slovakia, 2019, pp. 493-498. ⇒163

[17] T. S. Margush, Using an 8-bit RISC microcontroller in an assembly language programming course, *Journal of Computing Sciences in Colleges* **22**, 1 (2006) 15–22. ⇒161

[18] O. Matija, Using the Arduino platform within the Assembler subject (Využitie platformy Arduino v rámci predmetu Asembler), Bachelor's Thesis, Department of Computers and Informatics, FEEI, Technical University of Košice, Košice, 2015. ⇒161, 163, 178

[19] O. Mavropoulos, H. Mouratidis, A. Fish, E. Panaousis, C. Kalloniatis, A conceptual model to support security analysis in the internet of things, *Computer Science and Information Systems (ComSIS)* **14**, 2 (2017) 557–578. ⇒161

[20] S. P. Morse, B. W. Ravenel, S. Mazor, W. B. Pohlman, Intel microprocessors – 8008 to 8086, *IEEE Computer* **13**, 10 (1980) 42–60. ⇒161

[21] V. Moya del Barrio, Study of the techniques for emulation programming, *Proyecto fin de carrera. Universidad Politécnica de Cataluña, España*, 2001. ⇒160, 163, 170

[22] B. Nova, J.C. Ferreira, A. Araújo, Tool to Support Computer Architecture Teachingand Learning, *2013 1st International Conference of the Portuguese Society for Engineering Education (CISPEE)*, 2013. ⇒162

[23] B. Nova, DrMIPS Educational MIPS simulator, 2013-2015. ⇒162

[24] E. A. Qaralleh, K. A. Darabh, A new method for teaching microprocessors course using emulation, *Computer Applications in Engineering Education* **23**, 3 (2014) 455–463. ⇒162

[25] J. Rogers, EdSim51's Guide to the 8051: core of the popular 51 series of 8-bit micro-controllers, CreateSpace Independent Publishing Platform, 2009. ⇒161

[26] J. Rogers, The 8051 Simulator for Teachers and Students, 2005-2016. ⇒162

[27] G. Schmidt, Gerd's AVR simulator, 2017-2020. ⇒162
[28] G. Schmidt, Beginners introduction to the assembly language of ATMEL AVR microprocessors, 2016. ⇒166
[29] M. Šipoš, S. Šimoňák, RASP abstract machine emulator - extending the emuStudio platform, *Acta Electrotechnica et Informatica* **17**, 3 (2017) 33–41. ⇒160
[30] M. Šipoš, Extension of the emuStudio platform for emulation of computer architectures (in slovak), Diploma Thesis, Department of Computers and Informatics, FEEI, Technical University of Košice, Košice, 2018. ⇒164, 173
[31] K. Stevens, *The Emulation User's Guide*, Lulu.com, 2008. ⇒160
[32] R. K. Dirk von Suchodoletz, B. van der Werf, Long-term preservation in the digital age - emulation as a generic preservation strategy, *PIK - Praxis der Informationsverarbeitung und Kommunikation* **35**, 4 (2012) 225–226. ⇒160
[33] D. von Suchodoletz, K. Rechert, I. Valizada, A. Strauch, Emulation as an alternative preservation strategy – use-cases, tools and lessons learned, *INFORMATIK 2013 – Informatik angepasst an Mensch, Organisation und Umwelt*, 2013. ⇒160
[34] Wikipedia, List of computer system emulators, 2020. ⇒160
[35] A. Wolfe, A. Chanin, Executing compressed programs on an embedded RISC architecture, *ACM SIGMICRO Newsletter* **23**, 1-2 (1992) 81–91. ⇒164
[36] G. S. Wolffe, W. Yurcik, H. Osborne, M. A. Holliday, Teaching computer organization/architecture with limited resources using simulators, *ACM SIGCSE Bulletin* **34**, 1 (2002) 176–180. ⇒163
[37] H. Wong, CPUlator Computer System Simulator, University of Toronto, 2019. ⇒162
[38] C. Yehezkel, W. Yurcik, M. Pearson, D. Armstrong, Three simulator tools for teaching computer architecture: EasyCPU, Little Man Computer, and RTLSim, *J. Educ. Resour. Comput.* **1**, 4 (2001) 60–80. ⇒163

Received: February 18, 2020 • Revised: July 1, 2020

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 183–204

DOI: 10.2478/ausi-2020-0011

Improved balance in multiplayer online battle arena games

Chailong HUANG
Department of Computer Science
Bishop's University
Sherbrooke, Quebec, Canada
email: huang@cs.ubishops.ca

Stefan D. BRUDA
Department of Computer Science
Bishop's University
Sherbrooke, Quebec, Canada
email: stefan@bruda.ca

Abstract. The Multiplayer Online Battle Arena (MOBA) game is a popular type for its competition between players. Due to the high complexity, balance is the most important factor to secure a fair competitive environment. The common way to achieve dynamic data balance is by constant updates. The traditional method of finding unbalanced factors is mostly based on professional tournaments, a small minority of all the games and not real time. We develop an evaluation system for the DOTA2 based on big data with clustering analysis, neural networks, and a small-scale data collection as a sample. We then provide an ideal matching system based on the Elo rating system and an evaluation system to encourage players to try more different heroes for a diversified game environment and more data supply, which makes for a virtuous circle in the evaluation system.

1 Introduction

Electronic games [10], especially e-sports games, have become an important part of people's lives. Multiplayer Online Battle Arena (MOBA) games [2] in particular are very popular among young people because of their interesting

Computing Classification System 1998: H.3.3
Mathematics Subject Classification 2010: 62H30, 62M45
Key words and phrases: mupliplayer online battle arena game, game balance, matching system, clustering, neural network

and playful features. The reason why there are so many MOBA game lovers is the competition in all kinds of areas such as operation, strategy, and teamwork. The basis of this competitive pleasure lies in its balance.

DOTA (Defense of the Ancient) is the first independent MOBA game. A classic game with a 15-year history, it is still very popular as the second generation DOTA2. Witnessing its enduring popularity, more companies have seen the business opportunities, so a batch of similar MOBA games came out. Even though they are not as balanced as DOTA, they still have a large number of players.

Success and failure always exist simultaneously, so not all games are good. There are even more MOBA games in China, but most are commonly criticized for their low quality, unbalanced setting, even plagiarism. Game design is actually a process of producing artwork: beside creativity and inspiration, exquisite workmanship is also required. Balance is the workmanship of games, and is even more important in a competitive MOBA game. An imbalanced game cannot guarantee the loyalty of the fans; players will get bored easily if they only have a few options to win a game. Thus a deep understanding of the balance of the game as well as the way to achieve it are both necessary.

This paper develops a new method for achieving a better implementation of dynamic balance in DOTA2, and then an ideal matching system is designed as an improvement over the original. That is, the main contribution of this paper is a new method to achieve a better implementation of dynamic balance, with small-scale data collection as a sample. As a multiplayer online game, a fair matching system also plays an important role in game balance. Based on DOTA2 original rank/matching system (which uses the same rank system based on Elo ratings as League of Legends), we also design an ideal algorithm for the matching system. Finally we come up with an improvement on the rank/matching system based on the analysis of dynamic data balance.

2 MOBA games overview

Mutiplayer Online Battle Arena (MOBA) games are also called Action Real-time Strategy Games (Action RTS, ARTS), or DOTA-like Games. This in turn is all a subclass of Real-time Strategy (RTS) Games. In a MOBA game players are usually divided into two camps five versus five, and fight for more gold to buy items and for experience to level up. The ultimate goal is to destroy a certain building of the other side. A MOBA game player usually controls one character only called “hero”, which has specific abilities and slots to equip with

items. Our work focuses on DOTA2, which is a complex game. This section will only introduce (briefly) the elements that are related to our research.

Every DOTA2 game has generally 10 players, divided into two camps. Every player needs to pick a hero to control at the beginning of the game. The two camps have one ancient structure in each of their bases. The bases are located on each side's high ground, with three lanes to the other side. For each side, there are two barracks (melee and range) and tree defense towers on each lane. Barracks produce three types of "creeps" (melee, range and siege) every 30 seconds, who automatically attack all the enemy units along the lane. Defense towers automatically attack enemies within their range. The strength and number of creeps grow over time. Each side also has two jungles, with some neutral creeps in them. Players control their heroes to kill creeps and enemy heroes to gain gold and experience. Gold can be used to build items strengthening heroes. Experience is for leveling up; heroes can get one spell point each level for one of their four abilities (three basic abilities and one ultimate ability which can only be gained on certain levels). The only way to win is to destroy enemy's ancient structure at the center of their base.

There are 113 different *heroes*. Based on their major attributes, each hero is one of the following three types: Strength, Agility and Intelligence. There are more attributes combined in every hero beside these three types, including Armor, Move speed, Attack speed, Magic resistance, Health/Mana points, Damage, four abilities with different cool down and damage types, etc.

Most heroes have four *abilities*, three ordinary abilities and one ultimate. Each ordinary ability can be leveled up with a maximum level 4. The ultimate ability can be leveled up only at hero's level 6, 12 and 18. Every ability has a different effect such as dealing damage, stun, slow the enemy units, or provide beneficial status for the hero and its allies.

3 Game balance overview

> "A game is a series of meaningful choices." — Sid Meier

The quote above reveals the nature of game balance, namely that every player is supposed to have multiple choices to achieve their goals. Since there are more than a hundred heroes and items in the game, countless choices like team composition, item choice and combat strategy are made by every player every minute everywhere. Some of them are wise and good, others are not. But if only one choice is correct at every crossroad the game will soon become a meaningless repetition, and so becomes gradually boring; this is imbalance.

By contrast, a balanced MOBA game has bad choices, but it also has several good choices every time. In a balanced MOBA game you can try everything to win, unlike a robot who follows the same rules all the time.

The fundamental purpose for a player to play a game is to gain pleasure, so the playability of a game decides how long its life will be. In MOBA games balance is even more important for playability than in other types of games. Without balance, it doesn't even matter if you have more experience or better skill, the one who grabs the "perfect choice" always win the game. This situation could be caused by an unbalanced hero or by a bug in the matching system. In all, balance is always the most important factor for every game designer and developer, from beginning to the end.

Game data balance Data balance is subdivided into static and dynamic balance. Static data balance means that after the design and development, but before the release of the game, all the parameters in the game are in balance. For dynamic data balance, after feedback during internal test and public beta, the production team adjusts data and adds new elements through updates to achieve a better balance. Dynamic data balance is the interaction between players and game designers through feedback and adjustments to keep the game in a healthy balance all the time. Section 5 will elaborate on the process of achieving data balance in a MOBA game like DOTA2.

Focusing again on DOTA2 as an example, Valve has a dedicated team to accept reports from players around the world in the DOTA2 community. If too many players report a single imbalanced problem (or a bug), analysis and adjustment will be considered for next update. Another important reference is Professional Tournaments. There are around 100 games in a DOTA2 professional tournament [14]. From the behaviour of professional players, most based on the popularity of the heroes and items, Valve analyzes which heroes or items are picked the most and which the least. With the principle "balance every single hero", they will strengthen the most unpopular heroes and nerf (weaken) the hottest. However, with such a limited data collection only significant imbalance can be identified. Professional players are a very small part of the DOTA2 community, so this method cannot reflect the imbalanced factors comprehensively.

The above methodology makes the evaluation one-sided and not real time (big professional tournaments only happen every 2 to 3 months). This is the main problem we try to address in this paper; a new method based on data analysis for all players is developed in Section 5.

Matching system balance A player's level depends on experience, personal reaction time, teamwork awareness and physical/mental status. Without a reasonable matching system that can give every player an appropriate evaluation on their level it would be impossible to set up both team to have a similar overall level before the game starts. Fortunately, most MOBA games' matching systems have a general judgement for every player according to their performance in a large number of past games. After awhile, the evaluation system can objectively reflect the player's level.

DOTA2 has an excellent matching system with MMR (Match Making Rating). In Section 5, through a detailed analysis of DOTA2 matching system, an ideal matching system with improvements will be developed.

4 Further preliminaries

K-means cluster analysis will be used in this paper to cluster all the heroes into different types according to their data, so that the same type of heroes can be assessed with the same standard. Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a d-dimensional real vector, K-means clustering [8] aims to partition the n observations into $k \leq n$ sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares or WCSS (i.e., variance). This is accomplished by randomly selecting k cluster centroids μ_j, $1 \leq j \leq k$, determine cluster membership based on the distance from centroids, calculate new centroids by averaging the coordinates of the vectors in the current clusters, and repeat this process until the centroid selection converges.

An artificial neural network [3] consists of numerous simple units connect as a network. There are several kinds of neural networks, but based on our data processing requirement this paper will use a *BP-neural network* [11]. Such a network will be used to determine the specific weights of each type of data for every type of heroes respectively, for the final complete evaluation system.

The BP-neural network is based on the back-propagation algorithm. It is a multi-level learning network with supervised learning, featuring input and output neurons but also a "hidden" layer of neurons. The learning algorithm adjusts the parameters of the neurons based on the training data. Thus, a BP-neural network converts the input/output problem of a group of samples into a nonlinear optimization problem, which uses a gradient descent algorithm most commonly used in optimization techniques.

The Elo rating system [7] is a method for calculating the relative skill levels of players in zero-sum games such as chess. The current hierarchical scoring

system usually uses the logistic distribution $f(x) = L/(1 + e^{-k(x-x_0)})$, where x_0 is the x-value of the sigmoid's midpoint, L is the curve's maximum value, and k the steepness of the curve.

If Player A has a rating of R_A and Player B a rating of R_B, then the expected score of Player A is $E_A = 1/(1 + 10^{(R_B - R_A)/400})$. Similarly the expected score for Player B is $E_A = 1/(1 + 10^{(R_A - R_B)/400})$. Supposing Player A was expected to score of E_A points but actually scored S_A points, the formula for updating their rating is $R'_A = R_A + K(S_A - E_A)$. The factor K is based on the scoring rules and depends on what is the score unit for each game (10, 50, 100, etc.). $S_A = 1$ if player won the game, else $S_A = 0$.

In the Elo rating system the new updated rating for a player is only related to his original rating, the outcome of the game (win/lose) and the opponent's rating before the game, which satisfies the basic ranking/matching system of DOTA2. Players' behaviour is similar in competitive games. Finally, official description of another MOBA game League of Legends confirms the fact that its rank/match system is based on Elo ratings [13]. Considering their high level of similarity, we thus assume that DOTA2 follows the Elo rating system in the same way.

5 Balance implementation in DOTA2

In order to accurately evaluate the balance/ imbalance factors, we will establish an evaluation system for Heroes. Inspired by this evaluation system we then introduce an ideal matching system based on the Elo rating system together with some other improvements to achieve a better balance.

Win rate and damage dealt would be good choices to judge if a hero is too strong [1]. We consider 17 different heroes. Zeus, Huskar and Outworld Devourer appear in the statistics collected (using an API as described later) for both win rate and damage dealt, so we primarily focus on them to simplify data processing. We then collected a 10-player sample (randomly selected from one of the aurhor's friends list) playing with these 17 heroes as shown in Figure 1 in all the games played in January 2018 (ranging from 2 to 23).

The distribution shown in Figure 2 shows that different heroes' ability to deal damage differ substantially, so that damage dealt appears to be important in evaluating whether a specific hero is too strong or not. However there are obviously more factors that must be considered for a complete picture. Different heroes have different positions, attributes and abilities, which in turn have different effects in every DOTA2 game. For example, some heroes are meant

	A	B	C	D	E	F	G	H	I	J	K
1	GAME ID	Sccc	AI(EASY)	Rapier Rapier	Xiyan	Xphotograph	报复社会MO	Angelina	K_ASS	LuciferShana	NineG
2	Lycan	13850	10582	16854	11258	9825	11367	7845	18451	14526	10486
3	Zeus	26482	19853	22684	13698	19874	23658	12548	19269	21256	17987
4	Vegenful Spirit	7654	9845	4856	16504	5482	12574	8416	4019	14253	6874
5	Chaos Knight	11263	8764	12541	4216	9841	9331	8949	12577	10471	9096
6	Underlord	8945	5698	9874	11025	5476	9841	13024	9006	10104	4169
7	Omniknight	5612	4875	3641	4028	6214	3210	3987	4699	4251	5966
8	Huskar	10582	7685	18752	8481	7985	11374	10244	6934	17772	9424
9	Shadow Shaman	4012	4862	3684	5024	4687	4024	3940	3169	4127	5874
10	Outworld Devourer	14588	15862	12485	11487	12368	13674	13024	10147	15487	9871
11	Centaur Warrunner	9625	7685	11254	8947	7587	8714	5922	10478	6988	8744
12	Tinker	24960	11253	9874	20147	9897	14782	8770	18973	24873	14876
13	Spectre	26630	24563	17845	14793	18972	11258	20481	18633	10474	22103
14	Bristleback	18542	12485	15846	9877	15693	14870	11254	17451	12544	10024
15	Sniper	12368	11257	16485	19832	10166	11985	14120	9046	16870	12205
16	Gyrocopter	18963	9632	8746	20460	14885	7966	10441	7012	18969	17543
17	Arc Warden	8624	5762	20148	4980	7691	9822	22687	19890	10146	23662
18	Ember Spirit	14632	16875	11684	12486	13633	19890	9923	10207	13362	17439

Figure 1: Average damage dealt per game using 17 heroes and 10 players (January 2018).

to deal a huge amount of damage; some on the other hand are good at limiting enemy heroes' actions with stuns, slow, silence, etc; some others are supposed to help teammates by healing and take damage from the enemy.

Figure 3 show three different types of heroes: Zeus has a damage-dealing abilities of 4, so that even a bad player can deal a lot of damage with it. Centaur Warrunner, a representative tank, is supposed to take the most damages from enemy in every fight, and also deals some damage at the same time. Shadow Shaman, usually played as a support, helps cores (damage dealers) have a better environment to farm, and stuns enemy heroes to let allies have easy kills.

These samples show that the evaluation should be done in a comprehensive way considering all the abilities of a hero. In this aspect even the data analysis website dotamax.com has many deficiencies in that it features too few types of data. Fortunately, DOTA2 itself has an application programming interface (API) for its database including every single game's detailed data [5, 12]. This API allows access to many more types such as "damage taken" or "stun time" for every single game. With all these data, comprehensive analysis becomes possible. The following factors are significant in the assessment of a hero:

1. *Win rate* is the most important factor to evaluate, weight S is given.

2. *Damage dealt* can be divided into Building damage dealt, and Hero damage dealt. The only way to win a DOTA2 game is to destroy the enemy's Ancient base, and killing enemy's heroes would lead to an easy push, so these two factors are both important. Weight A is given.

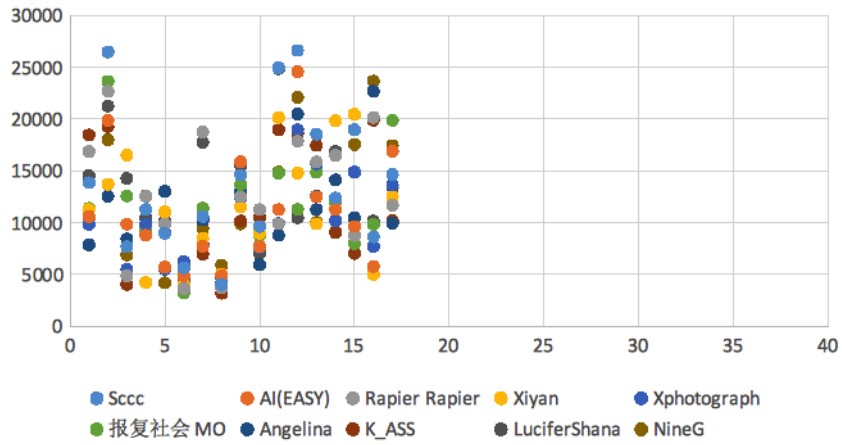


Figure 2: Different heroes' performance on dealing damage.

3. *Time of stun and hex:* Stun and hex can totally restrict any of enemy heroes' actions, which lead to an easy kill. However stun and hex themselves cannot make a killing happen, so they have a lighter weight B.

4. *Time of debuff* includes silence, root, slow and mute. These debuffs can only restrict one of the abilities such as move, attack, using ability of items, so they are even weaker than stun and hex. Weight C is given.

5. *Buff and heal* provide a beneficial effect for allies including speed up and extra damage (buff), and help allies regenerate their health/mana points (heal). These two factors have the same weight C as debuff.

6. *Damage taken* is helpful but it is not necessary all the time. Sometimes having a high capacity to take damage may even promote mistakes. In all this is weaker than all the factors above; weight D is given.

7. *Support ability* is mainly reflected in purchasing supportive items for the whole team, such as wards to provide vision and dust/sentry for detection. Supportive behaviour is very important in MOBA games including DOTA2; we give the weight B for it.

Obtaining accurate weights requires massive computation and iterative verification based on big data. Our aim is to provide a method rather than complete the calculation. Once the weights are determined we construct the usual

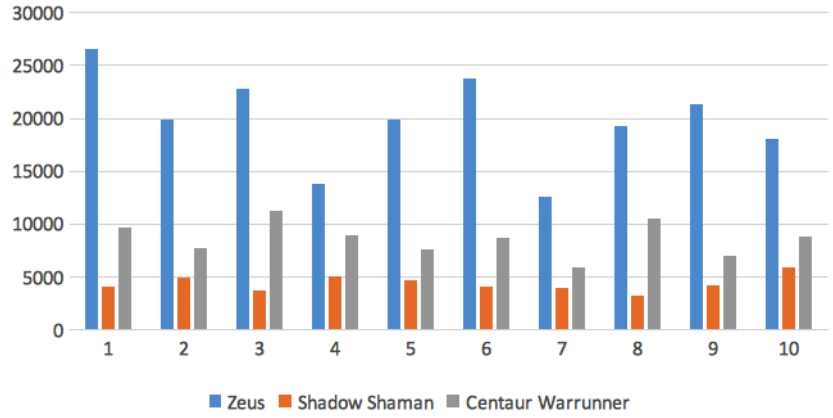


Figure 3: Different types of heroes.

formula for the assessment of heroes:

$$M = \sum_{i-1}^{n} W_i \times F_i \tag{1}$$

where F_i represent the normalized average values for the factors listed above based on the data from all DOTA2 games over a period of time, and W_i are the corresponding weights listed above. M thus becomes the *Balance value*. In a certain period of time, if a specific hero has a significantly abnormal balance value (too high or low), it is very likely that the last update had a very imbalanced effect on this hero. A fix may be needed after more tests to decide whether a new update is necessary.

However, based on the general idea outlined above it is easy to see that the "damage dealt" (to heroes or towers) type heroes always have the best assessment since these two data have the highest weights all the time. In practice things are not that simple. Many imbalanced heroes are so because of their different functional abilities, such as long-time stuns and huge amounts of healing. The assessment above is therefore not suitable for every unique hero. A better classification is required.

Besides damage dealt, two more data are now considered: Stun Time and Hero Healing (as a representative of Buff). There is no big data directly available for these two on the statistics website. Instead we obtained game data from the combat summary of the 10 players mentioned earlier, using 17 heroes in January 2018. Figure 4 shows the average statistics for stun time, hero healing, and damage dealing. We then performed clustering analysis [9] in Weka.

	A	B	C	D
1		Average Damage	Average Stun(s)	Average Healing
2	Lycan	12504.4	22.4	3425.5
3	Zeus	19730.9	42.8	80
4	Vegenful Spirit	9047.7	87.6	194
5	Chaos Knight	9704.9	99	864.2
6	Underlord	8716.2	63.5	1109.6
7	Omniknight	4648.3	18.3	5804.8
8	Huskar	10923.3	8.6	1095.2
9	Shadow Shaman	4340.3	156.4	92
10	Outworld Devourer	12899.3	18.9	0
11	Centaur Warrunner	8594.4	105.2	889.6
12	Tinker	15840.5	39.5	162.5
13	Spectre	18575.2	10.4	620.6
14	Bristleback	13858.6	12.3	1824.5
15	Sniper	13433.4	8	0
16	Gyrocopter	13461.7	48.8	0
17	Arc Warden	13341.2	22.9	135
18	Ember Spirit	14013.1	9.5	0

Figure 4: Average statistics of 17 heroes.

The first step is to determine the number of clusters. According to the rules of DOTA2, Heroes are divided into two primary roles, known as the Carry and the Support. Carries (or "cores") begin each match as weak and vulnerable, but are able to become more powerful later in the game, thus becoming able to "carry" their team to victory. Supports generally lack abilities that deal heavy damage, instead having functionality and utility that provide assistance for their carries. Basically a Carry is responsible for dealing huge amount of damage, while Supports create better chances for their Carries. However in our opinion, two classifications are not enough. Some heroes in DOTA2 can deal some lower amount of damage, while offering stun and healing at the same time. Most players call this kind of heroes Functional Cores. We thus submit that three types ("Carry" "Support" and "Functional Core") are needed.

Figure 5 shows the summary of our clustering analysis. Clusters 0, 1, 2 represent Functional Core, Support, and Carry, respectively. Hence Cluster 1 mainly contributes stun, Cluster 2 mainly contributes damage, and Cluster 0 does pretty well on damage, stun, and healing at the same time. The only element in Cluster 1 is Shadow Shaman, which is indeed an excellent support hero. Vegenful Spirit, Chaos Knight, Underlord, Omiknight and Centaur Warrunner are indeed able to deal some amount of damage and offer healing and stun at the same time. Others are purely damage dealers. The reason there

```
Final cluster centroids:
                              Cluster#
Attribute        Full Data           0          1          2
                     (17.0)      (5.0)      (1.0)     (11.0)
============================================================
AverageDamage    11978.4353     8142.3     4340.3 14416.5091
AverageStun         45.5353      74.72      156.4    22.1909
AverageHealing     735.1471    1612.44         92   394.8455
```

Figure 5: Result of clustering analysis on 17 DOTA2 heroes data.

is only one support hero is that the sample was picked as 10 most damage dealing heroes and 7 highest win rate heroes, so the data is supposed to have most elements in Cluster 2 and least elements in Cluster 1, which reflects the real world. Notice that determining the type of a hero should be based not only on the three types of data considered here, but also on the other factors discussed earlier. We limit our cluster analysis only on these three data points due to the limited resources allocated to this work. Extending our analysis to more data is however immediate.

Based on the cluster analysis, we refine the weights for our heroes' abilities in Equation (1) as follows:

1. *Carry heroes*: damage dealt $>$ healing $\geq$ time of stun and hex
2. *Support heroes*: time of stun and hex $>$ damage dealt $\geq$ healing
3. *Functional cores*: the weights of damage dealt, healing and time of stun and hex are almost at the same level.

We used the following principle: Each type of heroes has its specific duty, so the most important capability for a hero is based on what it is supposed to do (deal damage or support or limit enemy). The other abilities are not that important compared with its main purpose. This way, the balance of every single hero can be quantified in the same evaluation system. A better balanced update can be completed afterward based on big data on thousands million games rather than only professional tournaments.

Suppose the weights for damage, stun, and heal, respectively are 0.5, 0.3, 0.2 for the 11 Carry heroes, and 0.35, 0.35, 0.3 for the 5 Functional Cores. Since the Support cluster contains a single sample, this type will be ignored here. We then normalize our values using a linear function ($y = (x - min)(max - min)$). The Balance values for these two types are then shown in Figure 6.

Carry	BalanceValue	FunctionalCores	BalanceValue
Lycan	0.305882353	Vengeful Spirit	0.603650768
Zeus	0.760553204	Chaos Knight	0.675028769
Huskar	0.178206232	Underlord	0.478515032
Outworld Devourer	0.107470106	Omniknight	0.3
Tinker	0.471929363	Centaur Warrunner	0.624677433
Spectre	0.473918548		
Bristleback	0.231839052		
Sniper	0.202656888		
Gyrocopter	0.366235384		
Arc Warden	0.175338899		
Ember Spirit	0.188945456		

Figure 6: Rough balance values for carry and functional cores.

The Balance value ranges from 0.107 to 0.76. Based on these values all types of heroes can be evaluated at the same level. When there are 113 heroes with all the 7 attributes listed above considered, this range will be smaller and more precise. The ideal model will have a range from 0.4 to 0.6.

Based on the rough estimate above, we then used a BP-neural network to determine the weight for each type. We only consider the Carry heroes in this paper as the type has 11 samples, but the extension for other types is immediate. The inputs are the three types of data namely damage, stun and heal. Each input datum has a neuron (1, 2, 3, respectively) and the output is the evaluation for hero's balance (balanced or not) based on the Balance Value M. The activation function is $\mathsf{O} = 0$ for $0.4 \leq \mathsf{M} \leq 0.6$ and $\mathsf{O} = 1$ otherwise.

To obtain a complete training set some method of determining the output value O is needed. We propose one of the following two methods:

1. Refer to the previous update; if some heroes were modified, then the data before was unbalanced and the data after modifications is balanced.

2. Wait for the next update; heroes with modifications are assumed to have transitioned from unbalanced to balanced.

The logs for DOTA2 replays are only kept for 30 days, which makes it difficult to collect data from last update (it is also the reason why only the data from January 2018 is collected). We will therefore determine which heroes are balanced and which are not using the following assumption for a rough training of the neural network: we suppose that the heroes who just got modifications in the latest update are balanced, while the others are not. According to this

Hidden neurons	Input neurons			Output neurons
	1	2	3	
1	0.205612	0.102755	0.768041	0.5719874
2	0.672780	0.207483	0.344029	-0.263680
3	0.980589	0.462451	0.016968	0.585587

Figure 7: Weight coefficients between neurons.

Data type:	Damage	Stun	Heal
Weight (S):	0.446598302	0.240285127	0.31311657

Figure 8: Weight coefficients of every type of data for carry heroes.

estimate the balanced heroes are Lycan, Ember Spirit, Gyrocopter and Tinker, while the other seven are not balanced.

With the three types of data as input, 3 neurons in the hidden layer and initial weight as 0.5, 0.3, 0.2, after the training with 11 samples, the weights for every neuron are as shown in Figure 7.

We refer to the neurons using the indices i for input neurons, j for output neurons, and k for neurons in the hidden layer, with $1 \leq i \leq m$, $1 \leq j \leq n$, and $1 \leq k \leq P$. The relations between every 2 neurons are given by the method of Wang and Sun [4]: $r_{ij} = \sum_{k-1}^{P} W_{ki}(1 - e^{-x})/(1 + e^{-x})$ with $x = w_{jk}$, and $R_{ij} = |(1 - e^{-x})/(1 + e^{-y})|$ with $y = r_{ij}$. W_{ki} is the weight between the hidden layer neuron k to the input neuron i, and w_{jk} is the weight between the output neuron j and the hidden layer neuron k. The absolute influence coefficient is then $S_j = R_{ij} / \sum_{i=1}^{m} R_{ij}$, which is the required weight.

Based on this method and the data collected, the weights for every type of data were calculated as shown in Figure 8. We can establish a relatively complete evaluation system for Carry heroes by plugging in these weights into Equation (1), with $(W_i) = [0.446598302, 0.240285127, 0.31311657]$, and the normalized $F_i = [AverageDamage, AverageStunTime, AverageHeal]^T$. Then a Carry hero is balanced if and only if $0.4 \leq M \leq 0.6$.

It should be emphasized again that to simplify the calculation only three types of data are considered. A complete evaluation system will require all the types of data listed earlier.

The process above assumes that the latest updated heroes are balanced, which is likely an inaccurate method. The ideal accurate method is to always keep all the data for every hero for the latest month. After the new update is released (with the traditional methods mentioned above), we can pick the heroes that were modified and separate them into the old version set and the

new version set. These sets can be used for training the BP-neural network, where the old version set is unbalanced and the new version set is balanced. Once data is available for a new version we can retrain the system, making it more accurate. The final ("production") system should be established after several training sessions with different updates.

In all, the sample experiment described above is based on an inaccurate assumption (because as mentioned logs are only kept for 30 days, which makes it impossible to collect data from other updates), which makes it of reduced utility for validating more heroes. The ideal model requires persistent data collection for a long time, covering several updates. This being said, we believe that our system is both feasible and accurate if it is fed with accurate and complete big data. Our system has the following advantages:

1. Updates no longer rely on the limited data from professional tournaments; instead all player around the world can be part of it.

2. The new game changes (like new heroes and new strategies) can be balanced through the reevaluation of the related heroes in real time; any new training can be finished in a short time, with assessment readily available for the next update.

3. With more training, this evaluation system will become increasingly accurate and stable.

Since the third point requires substantially more data for training and test in the future, we only elaborate on the first and second advantages.

While 113 unique heroes seem to be enough, this is not actually true for an energetic MOBA game. In fact, Valve Corporation is still designing and releasing new heroes every year to maintain freshness. Some existing heroes are also reconstructed with brand new abilities, which in effect create another kind of new heroes. New (or reconstructed) heroes can be analyzed and clustered in a short time, in respect to their balance for a quick update, rather than waiting for feedback or data from professional tournaments.

With some modifications on its abilities, the position of a certain hero may change. For example, the Hero Monkey King used to be known as a Support, and Naga Siren as Carry. After an update on their ability values (mostly damage and time of stun), Monkey King became Carry and Naga Siren Support. Using our evaluation system new clustering analysis and balance evaluation can also be finished in a very short time, as soon as the data have changed.

When new strategies appear, there may be some heroes who can take full (and unexpected) advantage of them. For example, there used to be a popular

	Average Damage	Average Stun(s)	Average Healing
Sven	16187.5	29.8	0
Troll Lord	14982.3	13.2	1195.2

Figure 9: Average statistics of two heroes.

strategy in mid 2014 named the Pushing Strategy. This strategy requires all five heroes as a group and destroys enemy's towers early in the game. Only a few heroes are suitable for this strategy such as Pugna, Nature Prophet and Undying. With increasing popularity and higher win rate than other strategies, all types of data of pushing heroes rose rapidly. If this strategy is too strong in most conditions, then the new balance value of the respective heroes will go beyond the balanced range (> 0.6). Then a nerf on the heroes in combination with this strategy must be considered, and can be considered in the next update rather than waiting for the results of the tournaments.

This evaluation system based on big data from daily games all over the world can be more accurate and comprehensive, and a real-time reflection of the dynamic data balance of every single hero. It can work even more effectively on a new MOBA game, for which keeping the balance in an ideal range in a short time would be important to seize the players and the market.

Example Suppose that two new heroes are released in an update. We use Sven and Troll Lord as examples, since they are pure damage dealer Carry according to experience.

Figure 9 shows the data in the last 15 days from the 10 players sample. Assume that this is the first 2-weeks worth of data for the two new heroes. With our classification, they are classified as Carry hero. To compare them with others, we use Equation (1) to calculate their Balance value with weight coefficients $(W_i) = [0.446598302, 0.240285127, 0.31311657]$. The outcome is that the Balance values are 0.356003529 for Sven and 0.293009308 for Troll Lord.

Assuming that the weight coefficients are accurate enough (which will happen with enough training), then we can say that these 2 new heroes are a bit weaker than the average level (< 0.4). Some positive modifications are therefore necessary in the next update. All this analysis can be finished in a short time (2 weeks after the new heroes were released).

6 Ideal matching based on the Elo rating system

Inspired by the principle of the DOTA2 matching and rank system and the evaluation system described earlier, we introduce an ideal matching system based on Elo ratings together with other improvements to achieve a system that provides a better balance in support of the MOBA game players' experience. Notice that the DOTA2 matching system is not open-sourced, so we start from an ideal matching system based on matching rules instead.

6.1 An ideal matching system

The ideal matching system follows the Elo rating system as follows: Let $K = 50$ (standard points a player may earn or lose after a DOTA2 game) and $S_A = 1$ (player wins this game) or 0 (player loses this game). Based on DOTA2 rank/matching system, there are two Teams A and B with different average rank scores (R_A and R_B) in the same game, 5 players on each side. With $D = R_B - R_A$, we have $E_A = P(D) = 1/(1 + 10^{D/400})$ and $E_B = P(-D) = 1/(1 + 10^{-D/400})$. For convenience we use the Percentage Expectancy Tables [6] provided in the appendix to simplify the calculation process.

As an example, suppose Team A has an average rank score of 3890, and Team B an average rank of 3700. We have $D = 190$ and so $E_A = 0.75$ and $E_B = 0.25$. Therefore if Team A wins the game then every player in Team A will gain 12.5 points and every player in Team B will lose 12.5 points. On the other hand, if Team A loses the game then every player in Team A will lose 47.5 points and every player in Team B will gain 47.5 points.

Based on the Elo rating system, matching rules can be set up so that they observe the following properties:

1. Both sides have a similar average score.
2. The gap between the players with the highest and lowest score is small.
3. Both sides have a similar experience that is, a similar number of played games for players on both sides.
4. Scores of highest players on both sides are similar.
5. Both sides have similar number of solo/party players.
6. Complete the matching as fast as possible.

We then propose the following ideal matching system. Every team is a node, and every match is a queue with maximum 10 players. A team can be a solo player with his MMR, or a 2-5 party players with their average party MMR.

1. Once a new node comes in, detect if there is an eligible queue for it (the MMR is in range, there is space available for the node, etc.),
2. If yes, then add the node to the queue with minimal score difference.
3. Otherwise, add this node to a empty queue, with the new matching parameters based on the Elo rating system then wait.
4. Once a new node is added to a queue, check if this queue is full.
5. If yes, then find a sever to start this game and empty the queue, otherwise, keep waiting.
6. Every 30 seconds, check if there are new nodes added into the waiting queue.
7. If yes, then keep waiting.
8. Otherwise, expand the condition to a larger acceptable score difference based on the Elo rating system.
9. Repeat from 7.
10. If a queue has waited for 5 minutes, remove all the nodes, repeat from 1 to 6, and empty the queue.

The corresponding flow chart is shown in Figure 10.

6.2 An improved scoring method for the matching system

The rank system is suitable for most competitive games, in real life as well as in e-sports. However for MOBA games balance is the most important factor. With 113 different heroes, everyone is supposed to have a seat in this game. The Elo rating system only focuses on the outcome of a game, with no concern about what heroes players use.

What if all the players only pick certain strong heroes to play all the time? In this situation the balance of the game will be destroyed, together with the player experience. On other hand, some heroes are meant to be harder to get started than most others; if the designer doesn't encourage players to start

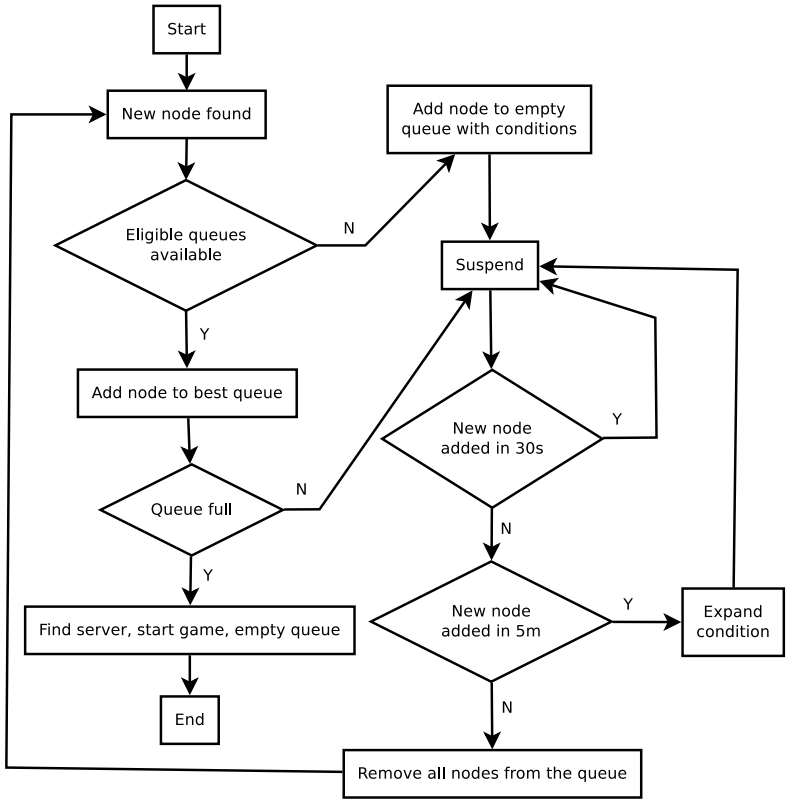


Figure 10: Flow chart for the ideal matching system.

with them, then the game will lack variety. In order to encourage players to play different heroes for a diverse environment, we propose an improvement on the scoring method discussed earlier.

First, we introduce a *hero rank*. This rank reflects the players' level while playing a certain hero, and is similar to the assessment of heroes in Equation (1). We establish it as $H = (\sum_{i=1}^{n} K_i \times E_i)/n$. H is the rank score for a certain hero of a player, E_i ranges over the data from all the games with this player using the given hero, including Win rate, Damage dealt, Stun/Slow Time, K/D/A, Damage Taken, etc. K_i are the weights for each type of data, which is the same concept as the weights in Equation (1) (different weights between heroes' types reflect how important the respective ability is for a certain hero), and n is the number of games the player has played with the respective hero. n is effective only when it is greater than a certain, small but not too

small a number (such as 10), to make sure the player is familiar enough to this hero for a precise and stable evaluation; this is the same as the calibration of the DOTA2 rank system.

We then compute the Hero-Pool value $T = (\sum_{i=1}^{n} H_i \times U_i)/\sum_{i=1}^{n} U_i$ to evaluate a single player's ability to use different heroes. H_i is the Hero rank for a certain hero, U_i is the number of games this player has played with this certain hero, and n is the number of heroes this player has used.

Notice that it is not realistic to assume that every single player can operate every single hero, so n should be set between 30 and 50, meaning that the Hero-Pool value only considers a player's highest top 30 to 50 unique heroes, encouraging players to play as many heroes as they reasonably can.

With this value, together with the old formula, we compute the improved final rank O as follows:

$$O = T \times k + S \tag{2}$$

where S is the old-version rank score based on the Elo rating system, T is the Hero-Pool value, and k is a constant coefficient given by statistical calculations.

With this improvement, the outcome of the game is no longer the only element that affects a players' rank score in DOTA2. Players will be encouraged to try more heroes with more combinations in a team. Therefore more heroes will be used in every play, more data will become available, problems and imbalance are easier to find, for an overall better balanced environment.

The improved matching system developed above is a bonus on top of the evaluation system (the most important contribution of this paper). Indeed, the evaluation system makes it possible to assess players' ability using different heroes. At the same time, according to statistics some most popular heroes are played 20 times more than the least popular ones, and this gap has a bad influence both on the game environment and on the data analysis of our evaluation system. Therefore the improvement given by Equation (2) on the matching/rank system will directly encourage players to use more heroes for a balanced environment and more data supply for an accurate clustering analysis and evaluation of all the heroes. This is all a virtuous circle.

7 Conclusions

We analyzed the importance of balance and the way to achieve it in MOBA, based on DOTA2 as an outstanding representative. We studied ways of improving balance in DOTA2, both on data and rank/matching system. Balance directly determines the diversity, playability and even life of a game, and the

traditional method of determining what data needs to be modified in the next patch in DOTA2 basically relies on players' reports and professional tournaments. Thousands of millions players' game data are not considered in this traditional method, which in turn makes it unsuitable for real-time analysis and also not accurate or comprehensive enough.

We developed a new method which consists of data collection, cluster analysis and neural network classification to quantify the 113 unique heroes in DOTA2, and to measure balance. With original data collected from the DOTA2 API, heroes are clustered into three types based on their features and data. A neural network is then used to determine the weights for every piece of data. A Balance value is then computed as a standard to measure whether a certain hero is balanced. Further updates can then target the unbalanced factors.

Although applied on limited data (30 days worth of logs, only three representative types of data, some inaccurate assumptions), this evaluation is still feasible and effective, especially with more time and data support. This method would also play a better and more important role in newborn MOBA games than in a mature game that is already balanced such as DOTA2.

Based on the Elo rating system, we also designed an ideal matching system for DOTA2, together with an improvement based on the hero evaluation system for a better balanced environment and more data supply. This is the second contribution of this paper.

An immediate continuation of this work would be to investigate how to accurately calculate all the weights from Equation (1) for each type of data. This paper used the strategy to assume that the latest updated heroes are balanced, which diminishes the accuracy of the results of training the neural network. An ideal and accurate method is to always keep all the data for every hero for the last month. After the new update is released (with the traditional methods mentioned above), one can pick the heroes with modifications, and separate them into two sets (old version and new version sets). We can then train the neural network with the old version set as "unbalanced" and new version set as "balanced"; note that this method requires more time and data collection. With enough training and our adjustment of the weights (ideally thought 3–4 patches), we believe that this system can become a standard.

This paper only focuses on heroes. Data on items can and should be considered as a balance factor. Items have the same types of data to heroes' abilities such as damage, stun, buff, plus price as one more factor to be considered.

References

[1] K. Conley, D. Perry, *How does he saw me? A recommendation engine for picking heroes in Dota 2*, Technical report, Stanford University, 2013. ⇒188

[2] J. Funk, MOBA, DOTA, ARTs: A brief introduction to gaming's biggest, most impenetrable genre, *Polygon*, December 2013. ⇒183

[3] M. van Gerven, Computational foundations of natural intelligence, *Frontiers in Computational Neuroscience*, **11** (2017), 112. ⇒187

[4] S. Huijun, W. Xinhua, Determination of the weight of evaluation indexes with artificial neural network method, *Journal of Shandong University of Science and Technology*, 20:84, 2001. ⇒195

[5] F. Johansson, J. Wikstrom, *Result prediction by mining replays in DOTA2*. Master's thesis, Blekinge Institute of Technology, Faculty of Computing, 2015. ⇒189

[6] P. Kannan, *Elo percentage expectancy table* ⇒198, 204

[7] H. P. Kriegel, M. Schubert, A. Züfle, Managing and mining multiplayer online games, *Advances in Spatial and Temporal Databases (SSTD 2011)*, *Lecture Notes in Computer Science*, **6849**, (2011), 441–444. ⇒187

[8] J. MacQueen. Some methods for classification and analysis of multivariate observations, *Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, 1967, 281–297. ⇒187

[9] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, K. Hornik, *Cluster: Cluster analysis basics and extensions*, 2016, R package version 2.0.4. ⇒191

[10] R. L. D. Mandryk, D. S. Maranan, False prophets: Exploring hybrid board/video games, *CHI: Conference on Human Factors in Computing Systems*, Minneapolis, Minnesota, 2002, pp. 640—641. ⇒183

[11] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, *Nature*, **323** (1986), 533–536. ⇒187

[12] B. G. Weber and M. Mateas. A data mining approach to strategy prediction. In *Proc. Computational Intelligence and Games*, pages 140–147, 2009. ⇒189

[13] Roit Games, *League of legends matchmaking explained*, 2009. ⇒188

[14] Valve Corporation, The international DOTA2 championships official website. ⇒186

A Elo percentage expectancy table

Based on the Elo rating and on $D = R_B - R_A$, E_A and E_B can be easily completed from the Percentage Expectancy Table [6] (Table 1). Further score differences can be obtained from Table 2 with the results in Table 1 as the median.

P(D)	D	P(D)	D	P(D)	D	P(D)	D	P(D)	D	P(D)	D
1.00	*	0.83	273	0.66	117	0.49	-7	0.32	-133	0.15	-296
0.99	677	0.82	262	0.65	110	0.48	-14	0.31	-141	0.14	-309
0.98	589	0.81	251	0.64	102	0.47	-21	0.30	-149	0.13	-322
0.97	538	0.80	240	0.63	95	0.46	-29	0.29	-158	0.12	-336
0.96	501	0.79	230	0.62	87	0.45	-36	0.28	-166	0.11	-351
0.95	470	0.78	220	0.61	80	0.44	-43	0.27	-175	0.10	-366
0.94	444	0.77	211	0.60	72	0.43	-50	0.26	-184	0.09	-383
0.93	422	0.76	202	0.59	65	0.42	-57	0.25	-193	0.08	-401
0.92	401	0.75	193	0.58	57	0.41	-65	0.24	-202	0.07	-422
0.91	383	0.74	184	0.57	50	0.40	-72	0.23	-211	0.06	-444
0.90	366	0.73	175	0.56	43	0.39	-80	0.22	-220	0.05	-470
0.89	351	0.72	166	0.55	36	0.38	-87	0.21	-230	0.04	-501
0.88	336	0.71	158	0.54	29	0.37	-95	0.20	-240	0.03	-538
0.87	322	0.70	149	0.53	21	0.36	-102	0.19	-251	0.02	-589
0.86	309	0.69	141	0.52	14	0.35	-110	0.18	-262	0.01	-677
0.85	296	0.68	133	0.51	7	0.34	-117	0.17	-273	0.00	*
0.84	284	0.67	125	0.50	0	0.33	-125	0.16	-284		

Table 1: P(D) Table according to the Elo percentage expectancy table.

SD = Score difference, EH = Expected score rate for high scorers, EL = Expected score rate for low scorers											
SD	EH	EL	SD	EH	EL	SD	EH	EL	SD	EH	EL
0-3	0.50	0.50	92-98	0.63	0.37	198-206	0.76	0.24	345-357	0.89	0.11
4-10	0.51	0.49	99-106	0.64	0.36	207-215	0.77	0.23	358-374	0.90	0.10
11-17	0.52	0.48	107-113	0.65	0.35	216-225	0.78	0.22	375-391	0.91	0.09
18-25	0.53	0.47	114-121	0.66	0.34	226-235	0.79	0.21	392-411	0.92	0.08
26-32	0.54	0.46	122-129	0.67	0.33	236-245	0.80	0.20	412-432	0.93	0.07
33-39	0.55	0.45	139-137	0.68	0.32	246-256	0.81	0.19	433-456	0.94	0.06
40-46	0.56	0.44	138-145	0.69	0.31	257-267	0.82	0.18	457-484	0.95	0.05
47-53	0.57	0.43	146-153	0.70	0.30	268-278	0.83	0.17	485-517	0.96	0.04
54-61	0.58	0.42	154-162	0.71	0.29	279-290	0.84	0.16	518-559	0.97	0.03
62-68	0.59	0.41	163-170	0.72	0.28	291-302	0.85	0.15	560-619	0.98	0.02
69-76	0.60	0.40	171-179	0.73	0.27	303-315	0.86	0.14	620-735	0.99	0.01
77-83	0.61	0.39	180-188	0.74	0.26	316-328	0.87	0.13	735	1.00	0.00
84-91	0.62	0.38	189-197	0.75	0.25	329-344	0.88	0.12			

Table 2: Corresponding expected score rate based on scores difference

Received: May 18, 2020 • Revised: September 15, 2020

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 205–216

DOI: 10.2478/ausi-2020-0012

Machine learning methods for toxic comment classification: a systematic review

Darko ANDROČEC
Faculty of Organization and Informatics, University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
email: dandrocec@foi.unizg.hr

Abstract. Nowadays users leave numerous comments on different social networks, news portals, and forums. Some of the comments are toxic or abusive. Due to numbers of comments, it is unfeasible to manually moderate them, so most of the systems use some kind of automatic discovery of toxicity using machine learning models. In this work, we performed a systematic review of the state-of-the-art in toxic comment classification using machine learning methods. We extracted data from 31 selected primary relevant studies. First, we have investigated when and where the papers were published and their maturity level. In our analysis of every primary study we investigated: data set used, evaluation metric, used machine learning methods, classes of toxicity, and comment language. We finish our work with comprehensive list of gaps in current research and suggestions for future research themes related to online toxic comment classification problem.

1 Introduction

Toxic comments are defined as comments that are rude, disrespectful, or that tend to force users to leave the discussion. If these toxic comment can be auto-

Computing Classification System 1998: I.2.7
Mathematics Subject Classification 2010: 68T50
Key words and phrases: machine learning, toxic comment, deep learning, systematic review

matically identified, we could have safer discussions on various social networks, news portals, or online forums. Manual moderation of comments is costly, ineffective, and sometimes infeasible. Automatic or semi-automatic detection of toxic comment is done by using different machine learning methods, mostly different deep neural networks architectures.

Recently, there is a significant number of research papers on the toxic comment classification problem, but, to date, there has not been a systematic literature review of this research theme, making it difficult to assess the maturity, trends and research gaps. In this work, our main aim was to overcome this by systematically listing, comparing and classifying the existing research on toxic comment classification to find promising research directions. The results of this systematic literature review are beneficial for researchers and natural language processing practitioners.

This work is organized as follows: Section 2 describes in detail the research methodology used for our systematic literature review. The next section lists the results and provides a discussion about obtained results of the systematic review. Our conclusions and future research ideas are provided in the final section.

2 Research methodology

This study has been carried out using the systematic literature review (SLR) methodology described in [16]. First, we have defined the SLR protocol. Then, we performed the study selection and the data extraction process whose outcome is the final list of papers. The main steps of the SLR protocol are listed and elaborated in the next subsections.

2.1 Planning

Planning starts with the identification of the needs for a specific systematic review. We have explained the needs for a systematic review on machine learning methods for toxic comment classification in Introduction section. Next, we have defined the following main research questions:
RQ1: When did the research on toxic comment classification become active in the research community?
RQ2: How is toxic comment classification research reported and what is the maturity level of the research in this field?
RQ3: Which data sets are used to classify toxic comments?

RQ4: Which machine learning methods are used to classify toxic comments?
RQ5: What are main evaluation metrics used to classify toxic comments?

Based on the objectives and research questions of this study, we have defined the review protocol. We have decided to include the following electronic databases: ACM Digital Library, IEEE Xplore, Scopus, and Web of Science Core Collection. These sources are chosen because they represent comprehensive literature in the machine learning field. We also included arXiv, because some highly-cited machine learning papers from researchers of commercial organizations is sometimes published only at this open-access archive service. The search string was simply defined as "toxic comment classification". We defined the following inclusion criteria (IC):

- IC1 - The main objective of the paper must discuss or investigate an issue related to toxic comment classification.
- IC2 - The work must be a research (scientific) paper.
- IC3 - The paper must be written in English.
- IC4 - The study should be published as a conference or a journal paper or a book chapter or an arXiv document.

We excluded papers based on the following exclusion criteria (EC):

- EC1 - Studies that are not related to the research questions.
- EC2 - Studies in which claims are non-justified or studies that had ad hoc statements instead of evidence-based statements.
- EC4 - Papers reported only by abstracts or slides.
- EC5 - Duplicate studies.
- EC6 - Demonstrations, preliminary studies, position papers, technical reports, posters and proof-of-concept papers were excluded.

2.2 Conducting

Conducting is the second step of the systematic review procedure. We have performed the search operation on the mentioned five electronic sources using search string *"toxic comment classification"* on 3th July 2020. We have used a reference management system *Zotero* where we added full texts of the articles to easy our systematic literature review. Our first search resulted in 202 extracted papers. After performing inclusion/exclusion criteria on titles and abstracts, 63 papers remained. After excluding the unrelated and duplicate works, 40 papers remained. The final selection was done by reading the whole text of the papers, and after this phase, we have selected 31 primary studies for our systematic review (Table 1).

ID	Paper title and reference
S1	A supervised multi-class multi-label word embeddings approach for toxic comment classification [5]
S2	Adversarial Text Generation for Googleś Perspective API [13]
S3	Are These Comments Triggering? Predicting Triggers of Toxicity in Online Discussions [1]
S4	Automation in Social Networking Comments With the Help of Robust fastText and CNN [18]
S5	Avoiding Unintended Bias in Toxicity Classification with Neural Networks [21]
S6	Bangla Toxic Comment Classification (Machine Learning and Deep Learning Approach) [15]
S7	BEEP! Korean Corpus of Online News Comments for Toxic Speech Detection [20]
S8	Challenges for Toxic Comment Classification: An In-Depth Error Analysis [32]
S9	Classification of Abusive Comments in Social Media using Deep Learning [2]
S10	Classification of Online Toxic Comments Using Machine Learning Algorithms [24]
S11	Classification of Online Toxic Comments Using the Logistic Regression and Neural Networks Models [28]
S12	Convolutional Neural Networks for Toxic Comment Classification [9]
S13	Cyberbullying ends here: Towards robust detection of cyberbullying in social media [33]
S14	Detecting Aggression and Toxicity using a Multi Dimension Capsule Network [30]
S15	Detecting Toxicity with Bidirectional Gated Recurrent Unit Networks [17]
S16	Detection of social network toxic comments with usage of syntactic dependencies in the sentences [29]
S17	Empirical Analysis of Multi-Task Learning for Reducing Model Bias in Toxic Comment Detection [31]
S18	Ensemble Deep Learning for Multilabel Binary Classification of User-Generated Content [10]
S19	Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning [12]
S20	Is preprocessing of text really worth your time for toxic comment classification? [19]
S21	LSTM neural networks for transfer learning in online moderation of abuse context [3]
S22	Machine Learning Suites for Online Toxicity Detection [22]
S23	On the Design and Tuning of Machine Learning Models for Language Toxicity Classification in Online Platform [26]
S24	Overlapping Toxic Sentiment Classification Using Deep Neural Architectures [27]
S25	Practical Significance of GA PartCC in Multi-Label Classification [23]
S26	Reading Between the Demographic Lines: Resolving Sources of Bias in Toxicity Classifiers [25]
S27	Stop illegal comments: A multi-task deep learning approach [8]
S28	Tackling Toxic Online Communication with Recurrent Capsule Networks [6]
S29	Towards non-toxic landscapes: Automatic toxic comment detection using DNN [7]
S30	Toxic comments identification in arabic social media [11]
S31	Using Sentiment Information for Preemptive Detection of Toxic Comments in Online Conversations [4]

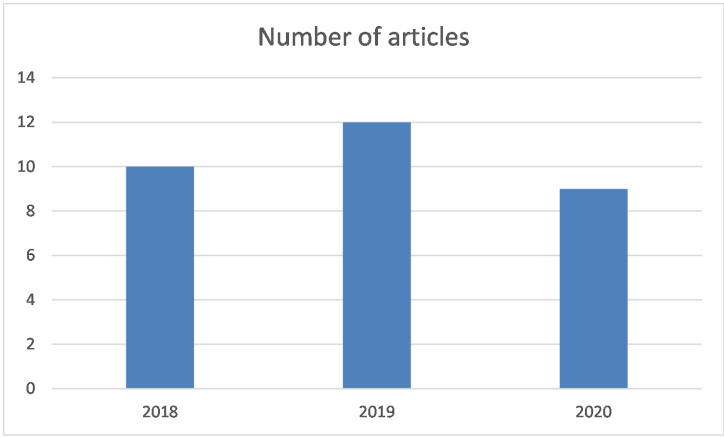


Figure 1: Selected primary studies per year

In the end, we extracted data from the 31 selected primary studies and did a synthesis taking into consideration the stated research questions. The results of our systematic literature review on toxic comment classification are shown in the next sections.

3 Results and discussion

3.1 Temporal overview of studies

The earliest relevant works on toxic comment classification were published in 2018 and their number significantly increased in 2019 (see Fig.1). There is a decrease in the number of studies in 2020, but this is due to the date when we performed the SLR search (3th July 2020, data are actually only for first halve of 2020).

3.2 Types of publications

The number of studies per publication type is demonstrated in Fig. 2. Most of the primary studies are conference papers (twenty), followed by arXiv papers (seven), three journal papers, and one book chapter.

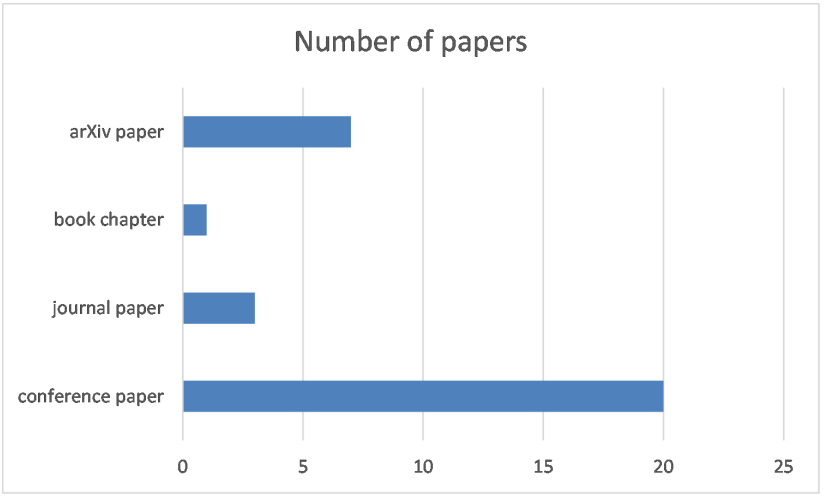


Figure 2: Papers per publication type

3.3 Used data set

Most of the primary studies have used one or two data sets with labelled toxic comments for a supervised machine learning. The most used data set is Jigsaw's data set hosted on Kaggle for Toxic Comment Classification Challenge competition [14]. It is used on twenty-two selected primary studies. This data set was later updated in Jigsaw Unintended Bias in Toxicity Classification Kaggle's competition. The mentioned data set contains a large number of Wikipedia comments which have been labelled by human raters for toxic behaviour. The types of toxicity are: toxic, severe toxic, obscene, threat, insult, and identity hate. Other used data sets are specific to a certain primary studies. Some of these other data sets are created by third parties: Twitter dataset by Davidson, Instagram dataset collected by Hosseinmardi et al., Semeval2018-Task 1 with almost 7000 tweets, The Twitter Hate Speech dataset, dataset of tweets made by members of the U.S. House of Representatives, Wikipedia Detox corpus, and live in-game chat conversations from a video game. The rest of datasets used in primary studies where created by authors of these studies: reviews taken from Udemy, synthetic training data by using Facebook comments that were posted in response to popular news articles, extensive collection of more than 104 million Reddit comments, a dataset taking comments from Facebook pages posts, 9.4K manually labelled entertainment news comments for identifying Korean toxic speech, comments in Hindi and English

both scraped from Facebook and Twitter, and custom prepared data in Arabic language from Facebook, Twitter, Instagram, and WhatsApp.

3.4 Used machine learning methods

Most of the selected primary studies have used more than one machine learning method to classify toxic comments from datasets mentioned in the previous subsection of this work. Table 2 shows in how many primary studies a specific machine learning method was used. The most used and effective methods are different deep neural networks, but often simpler and faster methods such as a logistic regression were used for baseline approaches.

Machine learning method	**Number of papers**
Convolutional neural network (CNN)	12
Logistic regression classifier	9
Bidirectional long short-term memory (BiLSTM)	8
Bidirectional Gated Recurrent Unit Networks (Bidirectional GRU)	6
Long Short Term Memory (LSTM)	5
Support Vector Machine (SVM)	5
Bidirectional Encoder Representations from Transformers (BERT)	4
Naive Bayes	4
Capsule Network	3
Random Forest	2
Decision tree	2
KNN classification	2
Gated Recurrent Unit (GRU)	2
Extreme Gradient Boosting (XGBoost)	2
Recurrent Neural Network (RNN)	2
Bi-GRU-LSTM	1
Gaussian Naive Bayes	1
Genetic Algorithms (GA)	1
Partial Classifier Chains (PartCC)	1

Table 2: Machine learning methods used in primary studies

3.5 Evaluation metrics

To evaluate results of using different machine learning methods to tackle problem of toxic comment classification, authors of primary studies use one or more evaluation metrics. Most used evaluation metrics are F1 score, accuracy, and area under the ROC curve (AUC ROC). All evaluation metrics used in selected primary studies are listed in Table 3.

Evaluation metric	Number of papers
F1 score	15
Accuracy	14
Area Under the ROC Curve (AUC ROC)	9
Custom AUC bias metric	2
Log loss	2
Hamming loss	2
False discovery rate	2
Mean precision	2
Mean recall	2
Pearson correlation coefficients	1
Specificity	1
Mean of the error rates	1
Generalized Mean Bias AUC	1
Subgroup AUC	1
BPSN AUC	1

Table 3: Used evaluation metrics

3.6 Classes of toxicity

Twenty-three of the primary studies have used six classes of toxicity defined in Jigsaw's data set hosted on Kaggle for Toxic Comment Classification Challenge competition [14]: toxic, severe toxic, obscene, threat, insult, and identity hate. Three papers used two classes (toxic or non-toxic). All used classes of toxicity are listed in Table 4.

Classes of toxicity	Number of papers
toxic, severe toxic, obscene, threat, insult, and identity hate	23
toxic or non-toxic	3
hate, offensive, and none	2
overtly aggressive, covertly aggressive, and non-aggressive	1
anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise and trust	1
racist content, sexist, and neutral	1

Table 4: Classes of toxicity identified in primary studies

3.7 Language of toxic comments

Only one paper (S14) analyse the toxic comment for two languages (in this case Hindi and English). Toxic comments in English are used the most (28

primary studies). The rest of studies dealt with Bangla (Bengali), Korean, and Arabic language.

4 Conclusions and future research ideas

Toxic comment classification is a complex research problem tackled by several machine learning methods. That is illustrated by many recent works in literature. After conducting a systematic review literature protocol proposed by Kitchenham and Charters [1], we have selected and analysed 31 primary studies. Our main conclusions are presented as answers to research questions as follows:
RQ1: When did the research on toxic comment classification become active in the research community? – The research on toxic comment classification become active recently, from 2018. It is due to release of Jigsaw's data set [33] that is mostly used in current related papers. From this year the number of paper is growing and this is an indicator that this research topic is actual and trendy.
RQ2: How is toxic comment classification research reported and what is the maturity level of the research in this field? – The most of the work are conference papers, so toxic comment classification is still a novel research topic.
RQ3: Which data sets are used to classify toxic comments? - The most used data set is the Jigsaw's data set hosted on Kaggle for Toxic Comment Classification Challenge competition [33]. Other data sets are mostly created from comments on popular social networks.
RQ4: Which machine learning methods are used to classify toxic comments? - The most used and effective methods are different architectures of deep neural networks, but often simpler and faster methods such as a logistic regression were used for baseline approaches.
RQ5: What are main evaluation metrics used to classify toxic comments? – Main evaluation metrics used to classify toxic comments are: F1 score, accuracy, and area under the ROC curve (AUC ROC).

The toxic comment classification research topic is a very active and challenging theme. Different transformers have recently shown a superior performance in many natural language processing tasks, so we recommend use of transformers for toxic comment classification in future works. Our systematic literature review identified three uses of BERT, but other transformers were not used yet (e.g. Hugging Face Transformers such as GPT-2, RoBERTa, XLM, DistilBert, XLNet... with pre-trained models in Tensor-

Flow 2.0 and PyTorch). Next, multilingual toxic comment classification is yet unsolved problem. In 2020, Jigsaw released multilingual toxic comment classification data set (https://www.kaggle.com/c/jigsaw-multilingual-toxic-comment-classification/data) that will be a basis for future work on this topic. The dataset is released under CC0, with the underlying comment text being governed by Wikipedia's CC-SA-3.0.

References

[1] H. Almerekhi, H. Kwak, J. Salminen, B. J. Jansen, Are These Comments Triggering? Predicting Triggers of Toxicity in Online Discussions, *Proceedings of The Web Conference 2020*, Taipei, Taiwan, Apr. 2020, pp. 3033–3040. ⇒208

[2] M. Anand, R. Eswari, Classification of Abusive Comments in Social Media using Deep Learning, *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India, Mar. 2019, pp. 974–977. ⇒208

[3] A. Bleiweiss, LSTM neural networks for transfer learning in online moderation of abuse context, *ICAART 2019 - Proceedings of the 11th International Conference on Agents and Artificial Intelligence*, Prague, Czech Republic,2019, pp. 112–122. ⇒208

[4] É. Brassard-Gourdeau, R. Khoury, Using Sentiment Information for Preemptive Detection of Toxic Comments in Online Conversations, ArXiv200610145 Cs, Jun. 2020, Accessed: Jul. 03, 2020. http://arxiv.org/abs/2006.10145. ⇒208

[5] S. Carta, A. Corriga, R. Mulas, D. R. Recupero, R. Saia, A supervised multi-class multi-label word embeddings approach for toxic comment classification, *IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, Vienna, Austria, 2019, pp. 105–112. ⇒208

[6] A. G. D'Sa, I. Illina, D. Fohr, Towards non-toxic landscapes: Automatic toxic comment detection using DNN, ArXiv191108395 Cs Stat, Nov. 2019, Accessed: Jul. 03, 2020. http://arxiv.org/abs/1911.08395. ⇒208

[7] S. Deshmukh, R. Rade, Tackling Toxic Online Communication with Recurrent Capsule Networks, *2018 Conference on Information and Communication Technology (CICT)*, Jabalpur, India, 2018. ⇒208

[8] A. Elnaggar, B. Waltl, I. Glaser, J. Landthaler, E. Scepankova, F. Matthes, Stop Illegal Comments: A Multi-Task Deep Learning Approach, *ACM International Conference Proceeding Series*, 2018, pp. 41–47. ⇒208

[9] S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, V. P. Plagianakos, Convolutional Neural Networks for Toxic Comment Classification, *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, Patras, Greece, Jul. 2018, pp. 1–6. ⇒208

[10] G. Haralabopoulos, I. Anagnostopoulos, D. McAuley, Ensemble Deep Learning for Multilabel Binary Classification of User-Generated Content, *Algorithms*, **13**, 4 (2020). ⇒208

[11] O. Hosam, Toxic comments identification in arabic social media, *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, **11**, (2019) 219–226. ⇒208

[12] M. Ibrahim, M. Torki, N. El-Makky, Imbalanced Toxic Comments Classification using Data Augmentation and Deep Learning, *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, Orlando, USA, 2018, pp. 875–878. ⇒208

[13] E. Jain et al., Adversarial Text Generation for Google's Perspective API, *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, USA, Dec. 2018, pp. 1136–1141. ⇒208

[14] Jigsaw, Data for Toxic Comment Classification Challenge. `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data`

[15] A. N. M. Jubaer, A. Sayem, Md. A. Rahman, Bangla Toxic Comment Classification (Machine Learning and Deep Learning Approach), *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)*, Moradabad, India, Nov. 2019, pp. 62–66. ⇒210, 212

[16] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering(2007). ⇒208

[17] V. Kumar, B. K. Tripathy, Detecting Toxicity with Bidirectional Gated Recurrent Unit Networks, *Adv. Intell. Syst. Comput.*, vol. **1034**,(2020) 591–600. ⇒206

[18] S. Mestry, H. Singh, R. Chauhan, V. Bisht, K. Tiwari, Automation in Social Networking Comments With the Help of Robust fastText and CNN, *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, Chennai, India, Apr. 2019, pp. 1–4. ⇒208

[19] F. Mohammad, Is preprocessing of text really worth your time for toxic comment classification?, *CSCE 2018 - Proceedings of the 2018 International Conference on Artificial Intelligence, ICAI 2018*, Las Vegas, USA, 2018, pp. 447–453. ⇒208

[20] J. Moon, W. I. Cho, J. Lee, BEEP! Korean Corpus of Online News Comments for Toxic Speech Detection, ArXiv200512503 Cs, May 2020, Accessed: Jul. 03, 2020. http://arxiv.org/abs/2005.12503. ⇒208

[21] S. Morzhov, Avoiding Unintended Bias in Toxicity Classification with Neural Networks, *2020 26th Conference of Open Innovations Association (FRUCT)*, Yaroslavl, Russia, Apr. 2020, pp. 314–320. ⇒208

[22] D. Noever, Machine Learning Suites for Online Toxicity Detection, ArXiv181001869 Cs Stat, Oct. 2018, Accessed: Jul. 03, 2020. http://arxiv.org/abs/1810.01869. ⇒208

[23] A. P. Patil, A. Mohammed, G. Elachitaya, M. Tiwary, Practical Significance of GA PartCC in Multi-Label Classification, *Proceedings of the 2019 Ieee Region 10 Conference (tencon 2019): Technology, Knowledge, and Society*, Kerala, India, 2019, pp. 2487–2490. ⇒208

[24] Rahul, H. Kajla, J. Hooda, G. Saini, Classification of Online Toxic Comments Using Machine Learning Algorithms, *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, May 2020, pp. 1119–1123. ⇒208

[25] E. Reichert, H. Qiu, J. Bayrooti, Reading Between the Demographic Lines: Resolving Sources of Bias in Toxicity Classifiers, ArXiv200616402 Cs, Jun. 2020, Accessed: Jul. 03, 2020. http://arxiv.org/abs/2006.16402. ⇒208

[26] M. Rybinski, W. Miller, J. Del Ser, M. Nekane Bilbao, J. F. Aldana-Montes, On the Design and Tuning of Machine Learning Models for Language Toxicity Classification in Online Platforms, *Intelligent Distributed Computing Xii*, **798** (2018), pp. 329–343. ⇒208

[27] H. H. Saeed, K. Shahzad, F. Kamiran, Overlapping Toxic Sentiment Classification using Deep Neural Architectures, *2018 18th Ieee International Conference on Data Mining Workshops (icdmw)*, Sentosa, Singapore, 2018, pp. 1361–1366. ⇒208

[28] M. A. Saif, A. N. Medvedev, M. A. Medvedev, T. Atanasova, Classification of Online Toxic Comments Using the Logistic Regression and Neural Networks Models, *Proceedings of the 44th International Conference Applications of Mathematics in Engineering and Economics*, Sozopol, Bulgaria, 2018. ⇒208

[29] S. Shtovba, O. Shtovba, M. Petrychko, Detection of social network toxic comments with usage of syntactic dependencies in the sentences, *CEUR Workshop Proceedings*, Otzenhausen, Germany, 2019, pp. 313–323. ⇒208

[30] S. Srivastava, P. Khurana, Detecting Aggression and Toxicity using a Multi Dimension Capsule Network. Stroudsburg: Assoc Computational Linguistics-Acl, 2019, pp. 157–162. ⇒208
⇒208

[31] A. Vaidya, F. Mai, Y. Ning, Empirical Analysis of Multi-Task Learning for Reducing Model Bias in Toxic Comment Detection, ArXiv190909758 Cs, Mar. 2020, Accessed: Jul. 03, 2020. http://arxiv.org/abs/1909.09758. ⇒208

[32] B. van Aken, J. Risch, R. Krestel, A. Löser, Challenges for Toxic Comment Classification: An In-Depth Error Analysis, ArXiv180907572 Cs, Sep. 2018, Accessed: Jul. 03, 2020. http://arxiv.org/abs/1809.07572. ⇒208

[33] M. Yao, C. Chelmis, D.-S. Zois, Cyberbullying Ends Here: Towards Robust Detection of Cyberbullying in Social Media, *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, San Francisco, USA, 2019, pp. 3427–3433. ⇒208

Received: July 23, 2020 • Revised: October 5, 2020

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 217–231

DOI: 10.2478/ausi-2020-0013

Degree tolerant coloring of graph

Johan KOK
Independent Mathematics Researcher, City of Tshwane, South Africa & Visiting Faculty at CHRIST (Deemed to be a University), Bangalore, India
email: jacotype@gmail.com

Abstract. This paper initiates a study on a new coloring regime which sets conditions in respect of the degrees $\deg(v)$ and $\deg(u)$ where, $v, u \in V(G)$ and $vu \in E(G)$. This new coloring regime is called, *"degree tolerant coloring"*. The degree tolerant chromatic number is defined. A number of interesting introductory results are presented. Amongst others, new Nordhaus-Gaddum type bounds are provided.

1 Introduction

For general notation and concepts in graphs see [2, 5, 8]. Throughout only finite, undirected, simple graphs will be considered. It is assumed that the reader is familiar with the concept of graph coloring. Recall that in a proper coloring of G all edges are good i.e. $\forall\ uv \in E(G)$, $c(u) \neq c(v)$. The set of colors assigned in a proper graph coloring is denoted by $\mathcal{C}$ and a subset of colors assigned to a subset of vertices $X \subseteq V(G)$ is denoted by $c(X)$. In an improper (or defect) coloring it is permitted that for some $uv \in E(G)$, the coloring is $c(u) = c(v)$. It is evident that improper coloring has been formally defined and studied by [3, 4, 6].

Since any graph G has the parameters, $\delta(G)$ and $\Delta(G)$, an integer degree condition related to an integer k, $\delta(G) \leq k \leq \Delta(G)$ will be introduced. For

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15, 05C38, 05C75, 05C85
Key words and phrases: Degree tolerant coloring, degree tolerant chromatic number, proper coloring, defect coloring

a *"degree tolerant coloring"* abbreviated as, DT-coloring of a graph G the following conditions are set:

(i) If $uv \notin E(G)$ then, either $c(u) = c(v)$ or $c(u) \neq c(v)$;

(ii) If $uv \in E(G)$ and $deg(u) = deg(v)$ then, $c(u) = c(v)$ else, $c(u) \neq c(v)$.

Alternative formulation for condition (ii). If $uv \in E(G)$ then, $c(u) = c(v)$ if and only if $deg(u) = deg(v)$.

The motivation for this study is that, it is fundamentally acceptable to *"do mathematics for the sake of mathematics"*. From an application point of view the vertices could represent communication elements (graph vertices in graph context) in a communication network. Failure of a communication vertex u could be replaced by a neighbor v if and only $deg(u) = deg(v)$. This can be made possible by imbedding equivalent or identical technology in both u and v. The technology equivalence is characterised by equal degrees for such pair of neighbors. The aforesaid could be viewed as spontaneous merging of such vertices on failure of any one vertex. Hence, a maximum number of such merging operations exists for a communication network before complete communication failure occurs. Analogy is found in electrical networks where one network relies on say, conventional fossil fuel generation and another relies of solar generation which is then distributed via an inverter plant. On failure of either, the neighbor can provide electricity to the other through a switch over protocol. The author foresees that interesting informatica research can result from this introductory paper.

The minimum number of colors which yields a DT-coloring is called the *degree tolerant chromatic number* of G and is denoted by, $\chi_{dt}(G)$. For certain classes of connected graphs the value of $\chi_{dt}(G)$ follows immediately. For example, for paths we have $\chi_{dt}(P_1) = \chi_{dt}(P_2) = 1$ and $\chi_{dt}(P_n) = 2$, $n \geq 3$. For all cycles, $\chi_{dt}(C_n) = 1$, $n \geq 3$. All regular graphs G have $\chi_{dt}(G) = 1$. Also, the null graph (edgeless) $\mathfrak{N}_n$ has $\chi_{dt}(\mathfrak{N}_n) = 1$. Clearly there exist graphs for which, $\chi_{dt}(G) \leq \chi(G)$. However, it is easy to verify that the graph G in Figure 1 has, $\chi_{dt}(G) > \chi(G)$.

As an introductory paper a variety as aspects are considered. Section 2 deals with preliminaries on general results and with elementary graph operations. Section 3 deals with three standard graph products. In addition, new Nordhaus-Gaddum type bounds are provided. In the conclusion some further research avenues are mentioned.

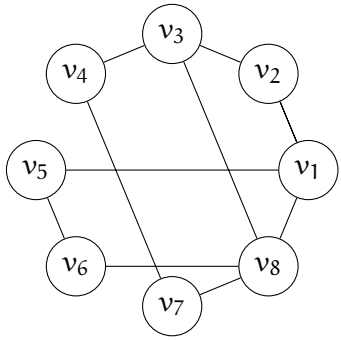


Figure 1: Graph G for which $\chi(G) = 2$ and $\chi_{dt}(G) = 3$

2 Some general results

In the literature a wide range of coloring regimes have been defined and many have been well researched. It is rare to find a formal result to show that all graphs permit a particular coloring regime. The aforesaid is mostly stated or silently assumed to be true. However, there does exist at least one coloring regime called, *Johan coloring* (or $\mathcal{J}$-coloring) which is not permitted by all graphs. See [7] and related references. It cannot be assumed that all graphs permit a DT-coloring but this is true in this case.

Theorem 1 *Any graph permits a* DT*-coloring.*

Proof. We proof the result through induction on n. At first, assume the graph G is a connected graph of order n. It is known that $\chi_{dt}(K_1) = 1$ hence, the result holds a connected graphs of order 1. Also, $\chi_{dt}(P_2) = 1$ hence, the result holds for all connected graphs of order 2. For $n = 3$ we have two cases to consider.

Case 1_3. Let $G = P_3$. It is known that, $\chi_{dt}(P_3) = 2$.

Case 2_3. Let $G = K_3$. Since K_3 is a regular graph, $\chi_{dt}(K_3) = 1$.

Thus far the result holds for all connected graphs of order 3.

For connected graphs of order 4, six cases up to isomorphism must be considered.

Case 1_4. For P_4 we have, $\chi_{dt}(P_4) = 2$.

Case 2_4. For the star $S_{1,3}$ it easily follows that, $\chi_{dt}(S_{1,3}) = 2$.

Case 3_4. For the graph G obtained from K_3 on vertices v_1, v_2, v_3 with a single pendant vertex u attached to any vertex of K_3 say, to v_1 the coloring, $c(v_1) = c_1$, $c(v_2) = c(v_3) = c(u) = c_2$ is a permissible DT-coloring.

Case 4_4. For the graph C_4 a DT-coloring is permissible because C_4 is regular.

Case 5_4. For the graph G obtained from C_4 on vertices v_1, v_2, v_3, v_4 with the addition of a chord say, v_1v_3, the coloring $c(v_1) = c(v_3) = c_1$ and $c(v_2) = c(v_4) = c_2$ is a permissible DT-coloring.

Case 6_4. It is known that K_4 permits a DT-coloring.

The basis for the induction assumption is now well established. Assume the result holds for all connected graphs on $1 \leq q \leq n$ vertices. Let the distinct connected graphs of order n up to isomorphism be, G_1, G_2, G_3, ..., G_κ. Consider any connected graph G_i, $1 \leq i \leq \kappa$ on $q = n$ vertices. To reason for the case $q = n + 1$ begin with the disconnected graph $G_i \cup K_1$. Let $V(K_1) = \{v_{n+1}\}$. Also assume the color set $\mathcal{C} = \{c_i : i = 1, 2, 3, \ldots, k\}$ is a DT-coloring set of G_i.

Construct G_i' by adding any number of edges say, t edges, $1 \leq t \leq n$ given by,

$$X = \underbrace{\{v_{n+1}v_i, v_{n+1}v_j, v_{n+1}v_k, \ldots, v_{n+1}v_s\}}_{(t \text{ edges})}.$$

Clearly, $\deg(v_{n+1}) = t$. Also the open neighborhood of v_{n+1} is, $N(v_{n+1}) = X = \{v_i, v_j, v_k, \ldots, v_s\}$.

If $\deg(v_{n+1}) \neq \deg(v_i)$, $v_i \in X$ then let $c(v_{n+1})$ be any color in $\mathcal{C} \backslash c(X)$ if possible else, assign a new color say c^*. In both cases, be it $\mathcal{C}$ or $\mathcal{C} \cup \{c^*\}$ assigned, a minimum DT-coloring is obtained.

If $\deg(v_{n+1}) = \deg(v_i)$, for some $v_i \in X$ we have the following subcases.

(i) If $\deg(v_{n+1}) = \deg(v_i)$, for exactly one $v_i \in X$, let $c(v_{n+1}) = c(v_i)$ to yield a permissible DT-coloring of G_i'.

(ii) Let $\deg(v_{n+1}) = \deg(v_i)$, for two or more vertices in X. If the said two or more adjacent vertices have identical coloring say, color c_r then assign $c(v_{n+1}) = c_r$. If at least one of the said adjacent vertices has a coloring other than the rest then recolor to $c(X \cup \{v_{n+1}\})$ any color in $\mathcal{C}$ and through iterative neighborhood re-assigned coloring obtain a DT-coloring, $\varphi : V(G_i') \mapsto \mathcal{C}$ if possible. Otherwise, assign $c(X \cup \{v_{n+1}\}) = c^*$. Therefore, $\mathcal{C} \cup \{c^*\}$ is a permissible DT-coloring of G_i'.

Through immediate induction the results holds for all connected graphs G_i', $1 \leq i \leq \kappa$. Put differently, it holds for all connected graphs of order $n + 1$. Hence, for all connected graphs of order q, $1 \leq q \leq n + 1$. Therefore, through mathematical induction it hold for all connected graphs of finite order.

Assume the graph H has t connected components i.e. $H_1, H_2, H_3, \ldots, H_t$. Clearly, $\chi_{dt}(H) = max\{\chi_{dt}(H_i) : \text{for some } i,\ 1 \leq i \leq t\}$. Since all H_i permit a

DT-coloring it follows that H permits a DT-coloring. Hence, the result holds for any simple graph of order n, $n \in \mathbb{N}$. □

Remark: The proof of Theorem 1 can be achieved through a *reconstructive* technique. Recall that the formal definition of a graph G of order n is, an ordered triple $(V(G), E(G), \iota_G)$ consisting of an non-empty set $V(G)$ of vertices say, $\{v_1, v_2, v_3, \ldots, v_n\}$ and a set $E(G)$, disjoint from $V(G)$, of edges and an *incidence function*, ι_G that associates with each edge of G an unordered pair of vertices of G (vertices in an unordered pair are not necessarily distinct). Begin with the null graph (edgeless) denoted by, $\mathfrak{N}_n$ on the vertices $\{v_1, v_2, v_3, \ldots, v_n\}$. Note that $\chi_{dt}(\mathfrak{N}_n) = 1$. Let $E(G) = \{e_1, e_2, e_3, \ldots, e_q\}$. Reconstruct G by iteratively adding the edges e_i, $i = 1, 2, 3, \ldots, q$ and by assigning a DT-coloring iteratively. Observe that after adding the edge e_1 the result, $\chi_{dt}(\mathfrak{N}_n + e_1) = 1$ follows. If edge e_2 is added such that e_1, e_2 are incident then the result, $\chi_{dt}((\mathfrak{N}_n + e_1) + e_2) = 2$ follows. Else, $\chi_{dt}((\mathfrak{N}_n + e_1) + e_2) = 1$. The observation is that G can be reconstructed iteratively and after each iteration a DT-coloring can be assigned. Formalising this approach as a proof is left to the reader.

Henceforth, only simple connected graphs will be considered. We now present some general results for $\chi_{dt}(G)$. Note that $\chi_{dt}(K_n) = 1$ because K_n is a regular graph.

Theorem 2 *For $n \in \mathbb{N}$ there exists a graph G with, $\chi_{dt}(G) = n$.*

Proof. Obviously, $\chi_{dt}(K_1) = 1$. Also, $\chi_{dt}(P_3) = 2$. For $n \geq 3$ begin with the complete graph K_n. Let $V(K_n) = \{v_1, v_2, v_3, \ldots, v_n\}$. Construct a new graph G by attaching to each vertex v_i an arbitrary number $k_i \geq 0$ pendent vertices such that, $k_i \neq k_j$ if and only if $i \neq j$. Clearly $d_G(v_i) \neq d_G(v_j)$ if $i \neq j$. However, $v_i v_j \in V(G)$ hence, $c(v_i) \neq c(v_j)$. Therefore, $\chi_{dt}(G) \geq n$. Without loss of generality let the pendent vertices adjacent to v_i be labeled $u_{i,\ell}$, $1 \leq \ell \leq k_i$. Because $d_G(u_{i,\ell}) = 1 \leq d_G(v_i)$ and $u_{i,\ell} v_i \in E(G)$ the coloring $c(u_{i,\ell}) = c_j$, $j \neq i$ is permissible. Thus $\chi_{dt}(G) \leq n$ and the result, $\chi_{dt}(G) = n$ follows. □

Recall that the order and size (number of edges) of a graph is denoted by $\nu(G)$ and $\varepsilon(G)$. For a graph parameter $p(G)$ of specific value say, $k \in \mathbb{N}$ a minimal graph G is a graph with $\min\{\nu(G) + \varepsilon(G)\}$ which yields, $p(G) = k$. Observe that for $k = 1$ the minimal graph $G = K_1$ yields, $\chi_{dt}(G) = 1$. For $k = 2$ the minimal graph $G = P_3$ yields, $\chi_{dt}(G) = 2$. For $k = 3$ the minimal graph is the *dart* graph. The dart graph has order 5 and size 6. These observations can be verified exhaustively against the list of small graphs at, www.graphclasses.org/smallgraphs.html We claim that, to construct such minimal graph the *founding or initial* graph is the complete graph of order

k. Firstly, we see that K_1, K_2 (or P_2) and K_3 (or C_3) are complete as well as regular graphs. Note that for $k = 4$ one could reasonably consider to begin with the trivial founding graph C_4 on vertices v_1, v_2, v_3, v_4, which is 2-regular. However, to achieve distinct colorings i.e. $c(v_1) \neq c(v_j)$ if $i \neq j$ with the minimal constructive additions of graph elements (vertices or edges), a K_4 will inevitably result through any construction methodology. Inductive reasoning is the basis of our claim.

Theorem 3 *For $k \in \mathbb{N}$ there exists a minimal graph G of order $n = 2k-1$ (or $\nu(G) = 2k-1$) and size $\varepsilon(G) = k(k-1)$ for which, $\chi_{dt}(G) = k$. Also, this minimal graph is unique.*

Proof. Obviously for $k \in \mathbb{N}$ and $\chi_{dt}(G) = k$ we must have, $\nu(G) \geq k$. It follows that for $k = 1$, the graph K_1 is minimal of order, $2 \times 1 - 1 = 1$.

For $k \geq 2$, begin with a graph putting all vertices *on equal footing* with regards to degree, i.e. having equal degree. Let the complete graph be on vertices $v_1, v_2, v_3, \ldots, v_k$. For v_i, $i = 2, 3, 4, \ldots, k$ add a distinct pendent vertex u_i. Also add the edges $u_i v_{i+j}$, $i = 2, 3, 4, \ldots, (k-1)$, $j = 1, 2, 3, \ldots, (k-i)$ to obtain G. Clearly, the aforesaid addition of pendent vertices and edges is the minimum constructive additions of graph elements to ensure, $deg(v_i) \neq deg(v_j)$ for $i \neq j$. Clearly, from conditions (i) and (ii) the minimum color set $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_k\}$ is required to assign a DT-coloring to G. From the construction of G it follows that G is a minimal graph hence, $min\{\nu(G)+\varepsilon(G)\}$, $\forall\ \nu, \varepsilon \in \mathbb{N}$ to yield, $\chi_{dt}(G) = k$. Furthermore, $\nu(G) = 2k-1$ and $\varepsilon(G) = \frac{k(k-1)}{2} + \frac{(k-1)k}{2} = k(k-1)$. Minimality and uniqueness of G follow directly from the stringent and unambiguous construction methodology. □

Figure 2 depicts the unique minimal graph for which $\chi_{dt}(G) = 3$. For example let, $c(v_1) = c_1$, $c(v_2) = c_2$, $c(v_3) = c_2$, $c(u_2) = c_1$, $c(u_3) = c_1$.

Theorem 3 leads to an important inverse result.

Theorem 4 *For a graph G of order $n \geq 1$ it follows that,*

$$\chi_{dt}(G) \leq \left\lfloor \tfrac{n+1}{2} \right\rfloor.$$

Proof. Observe that for $k = 1, 2, 3, 4 \ldots$ Theorem 3 provides the minimum order of graphs i.e. $n = 1, 3, 5, 7, \ldots, 2k-1, \ldots$ for which a minimal graph G can be constructed such that, $\chi_{dt}(G) = k$. It follows that for any graph G' of order ℓ, $2k-1 < \ell < 2k+1$ we have, $\chi_{dt}(G') < k+1$. Thus $\chi_{dt}(G') \leq k$.

Since for $k \in \mathbb{N}$, the minimal graph G is of order $n = 2k-1$, it implies that:

$$n - 2k + 1 = 0 \text{ has root at, } k = \tfrac{n+1}{2}.$$

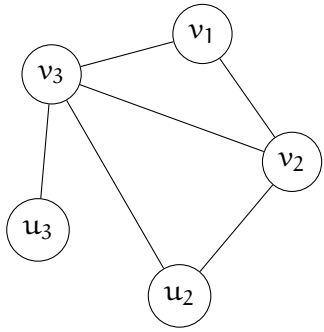


Figure 2: Unique minimal graph for which $\chi_{dt}(G) = 3$

For an integer solution it follows that,

$$\chi_{dt}(G) \leq \left\lfloor \tfrac{n+1}{2} \right\rfloor.$$

□

Theorem 5 *For a graph* G *of size* $\varepsilon(G) = q \geq 1$ *it follows that,*

$$\chi_{dt}(G) \leq \left\lfloor \tfrac{1+\sqrt{1+4q}}{2} \right\rfloor.$$

Proof. We know that for $k = 1, 2, 3, 4 \ldots$, Theorem 3 provides the minimum size of graphs i.e. $q = 0, 2, 6, 12, \ldots, k(k-1), \ldots$ for which a minimal graph G can be constructed such that, $\chi_{dt}(G) = k$. It follows that for any graph G' of size ℓ, $k(k-1) < \ell < (k+1)k$ we have, $\chi_{dt}(G') < k+1$. Thus, $\chi_{dt}(G') \leq k$.

Since for $q \in \mathbb{N}$, the minimal graph G is of size $q = k(k-1)$, it implies that:

$$q - k(k-1) = 0 \text{ has roots at, } \tfrac{1 \pm \sqrt{1+4q}}{2}.$$

For an integer solution it follows that,

$$\chi_{dt}(G) \leq \left\lfloor \tfrac{1+\sqrt{1+4q}}{2} \right\rfloor.$$

□

For n as determined by Theorem 3 in respect of $k \in \mathbb{N}$, let $C_1(G)$ be all graphs of order n.

Corollary 6 *A graph* $G \in C_1(G)$ *has* $\chi_{dt}(G) \leq k$.

Proof. The result is a direct consequence of Theorems 3 and 4. □

For q as determined by Theorem 3 in respect of $k \in \mathbb{N}$, let $C_2(G)$ be all graphs of of size q.

Corollary 7 *A graph* $G \in C_2(G)$ *has* $\chi_{dt}(G) \leq k$.

Proof. The result is a direct consequence of Theorems 3 and 5. □

Corollary 8 *A minimal graph* G *in respect of* k*-degree tolerant coloring is always of odd order and even size.*

A more significant result which follows directly from Theorems 4 and 5 is presented.

Theorem 9 *For a graph* G *of order* n *and size* q *we have,*

$$\chi_{dt}(G) \leq \min\{\left\lfloor \tfrac{n+1}{2} \right\rfloor, \left\lfloor \tfrac{1+\sqrt{1+4q}}{2} \right\rfloor\}.$$

2.1 On three elementary graph operations

Consider non-trivial graphs G and H and the elementary graph operations known as the disjoint union $G \cup H$, the corona $G \circ H$ and the join $G + H$.

Proposition 10 *For graphs* G *and* H *on order* n *and* m *respectively:*
(a) $\chi_{dt}(G \cup H) = \max\{\chi_{dt}(G), \chi_{dt}(H)\}$.
(b)

$$\chi_{dt}(G \circ H) = \begin{cases} \chi_{dt}(G), & \text{if } \chi_{dt}(G) > \chi_{dt}(H); \\ \chi_{dt}(H) + 1, & \text{if } \chi_{dt}(G) \leq \chi_{dt}(H). \end{cases}$$

Proof.
(a) Trivial.
(b) From a $\chi_{dt}(G)$-color set $\mathcal{C}$ assign a DT-coloring to graph G. In the graph $G \circ H$ a vertex u in a copy of H has, $\deg(u) \leq m$. In the graph $G \circ H$ a vertex v in G has, $\deg(v) \geq 1 + m$. Hence, $c(v)$ is distinct from any vertex color in the copy of H. For purposes of reasoning, begin by coloring each copy of H with its independent DT-coloring (ignoring adjacency of the corresponding vertex in $V(G)$.)

Case 1: If $\chi_{dt}(G) > \chi_{dt}(H)$ and $c(v) = c_i$ then all vertices in $u \in V(H)$ for which $c(u) = c_i$, may be recolored with any color in the set $\mathcal{C} \backslash \{c_i\}$. The coloring obtained is a permissible DT-coloring of $G \circ H$. Hence, $\chi_{dt}(G \circ H) = \chi_{dt}(G)$.

Case 2. If $\chi_{dt}(G) \leq \chi_{dt}(H)$ and $c(v) = c_i$ then all vertices in $u \in V(H)$ for which $c(u) = c_i$, must be recolored with a new color $c_{\chi_{dt}(H)+1}$. Hence, $\chi_{dt}(G \circ H) = \chi_{dt}(H) + 1$. □

Lemma 11 *For a graph* G *partition* $V(G)$ *into vertex subsets,* $P = \{X_1, X_2, X_3, \ldots, X_t\}$ *such that* $v_j, v_k \in X_i$ *if and only if* $\deg(v_j) = \deg(v_k)$ *else,* $|X_i| = 1$. *It follows that,* $\chi_{dt}(G) \leq |P|$.

Proof. Let a set of distinct colors be $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_{|P|}\}$ together with the mapping, $c(X_i) \mapsto c_i$, $\forall\ i$. Assume that the partition coloring does not correspond to a χ_{dt}-coloring of G and that $\chi_{dt}(G) > |P|$. It implies that for at least one $X_i \in P$ there exists at least one vertex $v_j \in X_i$. Therefore, $c(v_j) = c_l$, $c_l \notin \mathcal{C}$ is required. If $v_jv_k \in E(G)$, $c(v_k) = c_k \in \mathcal{C}$, it is a contradiction in terms of condition (ii). Hence, $c(v_j) = c_i$ is permissible. If $v_jv_k \notin E(G)$, then by condition (i) the coloring $c(v_j) = c_i$ is permissible. Therefore, $\chi_{dt}(G) \leq |P|$. □

Corollary 12 *For graph* G *the degree tolerant chromatic number is bounded by,* $1 \leq \chi_{dt}(G) \leq |P|$.

Association with vertex partition P**.** Obviously P is a partition of the vertex set of a graph. The parameter, $\deg_G(X_i) = \deg_G(v)$, $v \in X_i$ is also associated with P. Finally, for $P = \{X_i : 1 \leq i \leq t\}$ of $V(G)$ and $P' = \{Y_1, Y_2, Y_3, \ldots, Y_l\}$ of $V(H)$ we define the operation, $P \ominus P' = k$, where k is the number of vertex subsets, Y_j with $\deg_H(Y_j) \neq \deg_G(X_i)$, $\forall i$.

Reduction procedure: Reflecting on Lemma 11 it is observed that, if a vertex subset, $X_i \in P$ can itself be partitioned such that, $X_i = \bigcup_{j=1}^{r} X_{i,j}$ and for each $X_{i,j}$ there exists some vertex subset $X_k \in P$, $i \neq k$ such that, $\forall\ v_l \in X_{i,j}$ and $\forall\ v_t \in X_k$ the edge $v_lv_t \notin E(G)$, then the upper bound can be decreased (improved) by 1.

Example. Consider the path P_3 on vertices v_1, v_2, v_3. Attach pendent vertices u_1, u_2, u_3, u_4 to v_1 to obtain the graph G. Utilising Lemma 11 the vertex partition, $P = \{\{v_1\}, \{v_2\}, \{v_3, u_1, u_2, u_3, u_4\}\}$ is obtained. Hence, $\chi_{dt}(G) \leq 3$. Note that the vertex subset $\{v_3, u_1, u_2, u_3, u_4\}$ itself can be partitioned into $\{\{v_1\}, \{u_1, u_2, u_3, u_4\}\}$. Observe that edge, $v_1v_3 \notin E(G)$ and edges, $u_1v_2, u_2v_2, u_3v_2, u_4v_2 \notin E(G)$. Thus, $\chi_{dt}(G) \leq 2$. When the stated reduction procedure is exhausted, equality is attained. In the example, $\chi_{dt}(G) = 2$.

Proposition 13 *For graphs* G *and* H *of order* n *and* m *respectively, we have:* $\chi_{dt}(G + H) = \chi_{dt}(G) + (P \ominus P')$.

Proof. Without loss of generality let $n \geq m$. Assign a DT-coloring to G. Such DT-coloring exists by Theorem 1. Let the DT-color set be, $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_t\}$

and let $V(G) = \{v_1, v_2, v_3, \ldots, v_n\}$. Clearly, in the join $G + H$ the vertex degrees of $V(G)$ increases by the constant m hence, $deg_{G+H}(v_i) = deg_G(v_i) + m$, $1 \leq i \leq n$. Therefore, the color set $\mathcal{C}$ remains a DT-color set for the induced subgraph $\langle V(G)\rangle$. Also, $V(G)$ can be partitioned into $P = \{X_1, X_2, X_3, \ldots, X_t\}$ such that, $c(X_i) = c_i \in \mathcal{C}$, $i = 1, 2, 3, \ldots, t$.

Similarly, partition $V(H) = \{u_1, u_2, u_3, \ldots, u_m\}$ in accordance to an assigned DT-coloring of H. Let this partition be $P' = \{Y_1, Y_2, Y_3, \ldots, Y_l\}$. Note that each vertex degree in $V(H)$ has increased with a constant i.e. $deg_{G+H}(u_i) = deg_G(u_i) + n$, $1 \leq i \leq m$. Clearly, the vertices of $V(H)$ in a partition subset, $Y_i \in P'$ which has $deg_{G+H}(Y_i) = deg_{G+H}(X_j)$, must be colored $c(Y_i) = c(X_j)$. See condition(ii). Those which have $deg_{G+H}(Y_i) \neq deg_{G+H}(X_j)$, $\forall\ i$ must be colored with a new color not in $\mathcal{C}$. See condition (ii). Hence, by Lemma 11, $\chi_{dt}(G+H) \leq \chi_{dt}(G) + (P \ominus P')$. Because DT-colorings were assigned to both G and H the reasoning of Lemma 11 has been met. Also, the reduction procedure has implicitly been exhausted. Therefore, $\chi_{dt}(G + H) = \chi_{dt}(G) + (P \ominus P')$. $\square$

Due to the commutative property of $G + H$ it follows that, $\chi_{dt}(G + H) = \chi_{dt}(G) + (P \ominus P') = \chi_{dt}(H) + (P' \ominus P)$.

3 On graph products

Consider two graphs G and H of order n and m, respectively. Let the respective vertex sets be, $V(G) = \{v_1, v_2, v_3, \ldots, v_n\}$ and $V(H) = \{u_1, u_2, u_3, \ldots, u_m\}$. Recall that in general, a graph product is defined on the vertices $V(G) \times V(H)$. Adjacency (symbolised by, $\sim$) in the graph product is defined by conditions of adjacency (or non-adjacency) or equality, between pairs of distinct vertices in $V(G)$ and/or $V(H)$. It is known that $2^8 = 256$ products can be defined. This section will address three *standard* graph products. Note that in the literature there are different names corresponding to some of these graph products.

Convention: It is agreed that a comparable copy of a graph G means, comparable *diagrammatic presentation* including *self evident* comparable vertex labeling. For example, if the path P_3 is sketch horizontally with the vertices labeled from left to right as, v_1, v_2, v_3 then, a comparable copy (just copy for brevity) could be on the vertices labeled from left to right as, v_i, v_{i+1}, v_{i+2}, $i \geq 4$. It also implies that v_1 corresponds to v_i and it does not correspond to, v_{i+2}.

(i) Recall that adjacency in the Cartesian product denoted by, $G\square H$ is defined by $(v_i, u_j) \sim (v_k, u_t)$ if, $v_i = v_k$ and $u_j \sim u_t$ or, $v_i \sim v_k$ and $u_j = u_t$. To visualize this adjacency definition, simply replace each vertex of G with

a copy of H. For the vertex $v_i \in V(G)$ label such copy, H_{v_i}. If the edge v_iv_j exists then add an edge between *corresponding vertices* of the copies, H_{v_i} and H_{v_j}.

(ii) Recall that adjacency in the tensor product (also called, categorical product) denoted by, $G \times H$ is defined by $v_i \sim v_k$ and $u_j \sim u_t$. To visualize this adjacency definition is not easy. What is important to note is that, if the respective degree sequences are,

$$(deg(v_1), deg(v_2), deg(v_3), \ldots, deg(v_n))$$

and,

$$(deg(u_1), deg(u_2), deg(u_3), \ldots, deg(u_m))$$

then die degree sequence of $G \times H$ is given by,
$(deg(v_1)deg(u_1), deg(v_1)deg(u_2), deg(v_1)deg(u_3), \ldots, deg(v_1)deg(u_m),$
$deg(v_2)deg(u_1), deg(v_2)deg(u_2), deg(v_2)deg(u_3), \ldots, deg(v_2)deg(u_m),$
$\ldots$
$\ldots$
$\ldots$
$deg(v_n)deg(u_1), deg(v_n)deg(u_2), deg(v_n)deg(u_3), \ldots, deg(v_n)deg(u_m))$.

(iii) Recall that adjacency in the lexicographical product denoted by, $G \bullet H$ is defined by $v_i \sim v_k$ or, $v_i = v_k$ and $u_j \sim u_t$. To visualize this adjacency definition, simply replace each vertex of G with a copy of H. For the vertex $v_i \in V(G)$ label such copy, H_{v_i}. If the edge v_iv_j exists then add all edges of the join, $H_{v_i} + H_{v_j}$.

Definition 14 *Let* $(deg(v_1), deg(v_2), deg(v_3), \ldots, deg(v_n))$ *be the degree sequence of graph* G. *The degree index of graph* G *denoted by* $di(G)$, *is the number of distinct vertex degree values.*

Note that for any path P_n), $n \geq 3$ we have, $di(P_n) = 2$. For any cycle C_n, $n \geq 3$ we have, $di(C_n) = 1$.

Theorem 15 *For graphs* G *and* H *it follows that,*

$$\chi_{dt}(G \square H) = min\{di(H)\chi_{dt}(G), di(G)\chi_{dt}(H)\}.$$

Proof. It is known that the Cartesian product is commutative to isomorphism. Hence, $G \square H \cong H \square G$.

Case 1. Consider $G \square H$. Replace each vertex $v_i \in V(G)$ with H_{v_i}. The adjacency condition, $v_i = v_k$ and $u_j \sim u_t$ is immediately satisfied. Do the following

in respect of each $(v_i, u_j)_{j=1,2,3,\ldots,m}$, $i = 1, 2, 3, \ldots, n-1$. For each neighbor of v_i in G add an edge between the vertices (v_i, u_j), $j = 1, 2, 3, \ldots, m$ and between the respective corresponding vertices in the neighboring copy of H, if such edges do not exist. After the stated procedure the condition $v_i \sim v_k$ and $u_j = u_t$ has been satisfied. Hence, $G \square H$ has been obtained. Clearly, $deg((v_i, u_j)) = deg(u_j) + |N(v_i)|$. Following from Lemma 11, $\chi_{dt}(G \square H) \leq di(G)\chi_{dt}(H)$.

Case 2. Consider $H \square G$. By similar reasoning as in Case 1 it follows that, $\chi_{dt}(H \square G) \leq di(H)\chi_{dt}(G)$.

Thus far by Lemma 11, we showed that,

$$\chi_{dt}(G \square H) \leq min\{di(H)\chi_{dt}(G), di(G)\chi_{dt}(H)\}.$$

Since the reduction procedure has implicitly been exhausted for at least one of the two cases it follows that,

$$\chi_{dt}(G \square H) = min\{di(H)\chi_{dt}(G), di(G)\chi_{dt}(H)\}.$$

□

Theorem 16 *For graphs* G *and* H *it follows that,* $\chi_{dt}(G \times H) \leq di(G \times H)$.

Proof. It is known that the tensor product is commutative to isomorphism. Hence, $G \times H \cong H \times G$.

The degree sequence of $G \times H$ is given by:
$(deg(v_1)deg(u_1), deg(v_1)deg(u_2), deg(v_1)deg(u_3), \ldots, deg(v_1)deg(u_m),$
$deg(v_2)deg(u_1), deg(v_2)deg(u_2), deg(v_2)deg(u_3), \ldots, deg(v_2)deg(u_m),$
$\ldots$
$\ldots$
$\ldots$
$deg(v_n)deg(u_1), deg(v_n)deg(u_2), deg(v_n)deg(u_3), \ldots, deg(v_n)deg(u_m))$.
Hence, the minimum number of colors needed to be assigned is $di(G \times H)$. The reduction procedure can be used to yield equality on a case by case basis. Therefore, $\chi_{dt}(G \times H) \leq di(G \times H)$. □

Theorem 17 *For graphs* G *and* H *it follows that,* $\chi_{dt}(G \bullet H) \leq di(G)\chi_{dt}(H)$.

Proof. It is known that the lexicographical product is associative but not commutative. Therefore, the reasoning is similar to that stated in the proof of Theorem 15, Case 1. However, exhaustion of the reduction procedure is not self-evident. Hence, $\chi_{dt}(G \bullet H) \leq di(G)\chi_{dt}(H)$. □

3.1 Nordhaus-Gaddum type bounds

Relations between a graph G and its complement $\overline{G}$ have been studied since the inception of graph theory. Perhaps the most interesting category of relations between the two graphs are those with regards to the sum and products of graph parameters. The first known such relations were introduced by Nordhaus and Gaddum in 1956. The relations provide lower and upper bounds on the sum and the product of the chromatic number of a graph and its compliment. A comprehensive survey of the wide field which developed over the years can be found in [1].

Definition 14 read together with Lemma 11 imply that for a graph G, $di(G) = t$. Also, $deg_{\overline{G}}(v_i) = (n-1) - deg_G(v_i) \Rightarrow di(\overline{G}) = di(G)$.

Theorem 18 *For a graph* G *of order* n *and size* q *it holds that,*
(a)

$$2 \leq \chi_{dt}(G) + \chi_{dt}(\overline{G}) \leq 2di(G),$$
$$1 \leq \chi_{dt}(G) \cdot \chi_{dt}(\overline{G}) \leq di(G)^2.$$

Weaker bounds are;
(b)

$$2 \leq \chi_{dt}(G) + \chi_{dt}(\overline{G}) \leq 2(\lfloor \tfrac{n+1}{2} \rfloor),$$
$$1 \leq \chi_{dt}(G) \cdot \chi_{dt}(\overline{G}) \leq \lfloor \tfrac{n+1}{2} \rfloor^2.$$

(c)

$$2 \leq \chi_{dt}(G) + \chi_{dt}(\overline{G}) \leq \left\lfloor \tfrac{1+\sqrt{1+4q}}{2} \right\rfloor + \left\lfloor \tfrac{1+\sqrt{1+4(\frac{n(n-1)}{2}-q)}}{2} \right\rfloor,$$
$$1 \leq \chi_{dt}(G) \cdot \chi_{dt}(\overline{G}) \leq \left\lfloor \tfrac{1+\sqrt{1+4q}}{2} \right\rfloor \cdot \left\lfloor \tfrac{1+\sqrt{1+4(\frac{n(n-1)}{2}-q)}}{2} \right\rfloor.$$

Proof. (a) The result is a direct consequence of Lemma 11.
(b) The result is a direct consequence of Theorem 4.
(c) The result is a direct consequence of Theorem 5. □

Any simple graph G of order n has at most, $\frac{n(n-1)}{2}$ edges. From Theorem 3 it follows that the minimal graph G has order $2k-1$ and size $k(k-1)$ for $k \in \mathbb{N}$. Hence, $\overline{G}$ has $\varepsilon(\overline{G}) = k^2 - 2k + 1$. The aforesaid implies that $\varepsilon(G) - \varepsilon(\overline{G}) = k - 1 > 0$, $\forall\ k \geq 2$. Clearly G is not self-complementary. This leads to a corollary.

Corollary 19 *For the minimal graph* G *from Theorem 3 and for* $k \geq 2$ *we have,* $\chi_{dt}(G) > \chi_{dt}(\overline{G})$.

Proof. Theorem 5 read together with the fact that, $\varepsilon(G) - \varepsilon(\overline{G}) = k - 1 > 0$, $\forall\ k \geq 2$, suffice. □

4 Conclusion

The degree tolerant chromatic number was introduced subject to conditions (i) and (ii). Different conditions can be formulated to study derivatives of the notion of degree tolerance coloring. For example, for a given k, $1 \leq k \leq n-1$ a condition such as; if $deg(u) \leq k$ and $deg(v) \leq k$ and $uv \in E(G)$ then, $c(u) = c(v)$ else, $c(u) \neq c(v)$ is a derivative for further study.

Problem 1. The procedure to improve the upper bound stated in Lemma 11 follows from condition (i). The fact that applying the procedure exhaustively will yield equality has been stated without proof. Formalise the statement.

Problem 2. Utilise the result in problem 1 to determine exact values of $\chi_{dt}(G)$ for various classes of graphs.

Problem 3. If possible find improved results for Theorems 16 and 17.

Conjecture. For graphs G and H it follows that,

$$\chi_{dt}(G \times H) = min\{di(H)\chi_{dt}(G), di(G)\chi_{dt}(H)\}.$$

A worthy avenue for further research would be to consider all known graph products such as, strong product, co-normal product, modular product, rooted product and so on.

It is observed that with regards to the clique number the minimal graph G constructed in the proof of Theorem 4 is, $\omega(G) = k$. Furthermore, exactly two such induced cliques exist in G. We suggest that studying other graph parameter specific to this minimal graph could be a worthy avenue.

Let $f(k), g(k), h(k)$ be functions such that $f(k) = \min\{g(k), h(k)\}$. If for some $k \in \mathbb{N}$ we have that, $g(k) = h(k)$ then $f(k)$ is said to be *tied* or equal-valued. If $g(k) \neq h(k)$ then $f(k)$ is said to be *non-tied* or decisive. A graph of order n and size q can be called a (n, q)-graph. It is obvious from Theorems 4 and 5 that the function, $f_1(n, q) = \min\{\left\lfloor \frac{n+1}{2} \right\rfloor, \left\lfloor \frac{1+\sqrt{1+4q}}{2} \right\rfloor\}$ is *tied* (equal-valued) for all $(2k-1, k(k-1))$-graphs.

Problem 4. Prove that $f_2(n, q)$ is *non-tied* (or decisive) for all $(g(k), h(k))$-graphs if the functions, $n = g(k)_{k\in\mathbb{N}} \neq 2k-1$ or $q = h(k)_{k\in\mathbb{N}} \neq k(k-1)$.

References

[1] M. Aouchiche, P. Hansen, A survey of Nordhaus-Gaddum type relations, *Discrete Applied Mathematics*, **161** (2013) 466–546. ⇒229

[2] J. A. Bondy, U.S.R. Murty, *Graph Theory with Applications,* Macmillan Press, London, (1976). ⇒217

[3] L. Cowen, R. Cowen, D. Woodall, Defective colorings of graphs in surfaces: partitions into subgraphs of bounded valence, *Journal of Graph Theory*, **10** (1986) 187–195. ⇒217

[4] L. Cowen, W. Goddard, C. Jesurum, Coloring with defect, *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, 1997, pp. 548–557. ⇒217

[5] F. Harary, *Graph Theory*, Addison-Wesley, Reading MA, 1969. ⇒217

[6] F. Harary, K. Jones, Conditional colorability II: Bipartite variations, *Congressus Numer.*, **50** (1985) 205–218. ⇒217

[7] N. K. Sudev, On certain $\mathcal{J}$-colouring parameters of graphs, *Nat. Acad. Sci. Lett.*, **43** (2020) 53–57. ⇒219

[8] B. West, *Introduction to Graph Theory*, Prentice-Hall, Upper Saddle River, (1996). ⇒217

Received: June 9, 2020 • Revised: October 13, 2020


Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 232–250

DOI: 10.2478/ausi-2020-0014

Enhanced type inference for binding-time analysis

Mátyás SZOKOLI
Eötvös Loránd University
Budapest, Hungary
email: vyu4a5@inf.elte.hu

Attila KISS
J. Selye University
Komárno, Slovakia
email: kissae@ujs.sk

Abstract. In this paper we will be taking a look at type inference and its uses for binding-time analysis, dynamic typing and better error messages. We will propose a new binding-time analysis algorithm $\mathcal{B}$, which is a modification of an already existing algorithm by Gomard [4], and discuss the speed difference.

1 Introduction

Binding-time analysis (BTA) is used to guide partial evaluation, which is an optimization technique that from an input program produces an output program behaving in the same way, with the output having faster run time. We divide the program into separate parts based on two or more stages. In the case of two stages these are the static and dynamic parts. The compiler can perform the static computations in compile time, yielding a more efficient residual program. This analysis can be done from multiple angles and approaches: through abstract interpretation, type inference, temporal logic and type-based search. One such method developed by Gomard uses the widely-known W type inference algorithm. We replace this inference algorithm by M. We show that this modified algorithm is correct, and is faster then using W in many cases.

Computing Classification System 1998: D.1.1, F.4.1, F.3.2, F.3.3
Mathematics Subject Classification 2010: 68N18, 68N15, 03B38
Key words and phrases: type inference, dynamic typing, binding-time analysis, partial evaluation, functional languages

2 Related works

The value of type systems for binding-time analysis was found early, both Nielson [10] and Gomard [3] formalized it as a type inference problem for the two-level typed λ-calculus. Algorithm $\mathcal{W}$ was already used by both Gomard and Nielsen for their respective λ-calculus variants. For a detailed look into types and type systems beyond what is included in this paper, see the works of Benjamin C. Pierce [15, 16].

Tim Sheard and Nathan Linger introduced a search-based method to perform BTA [11], and they implemented it for MetaML. MetaML already had manual staging annotations, and they integrated automatic BTA to get a unified system to get the advantages from both manual and automated annotations, as they felt that both ways are beneficial to the programmer. Their proposal integrates automatic BTA as a part of the language, not an external tool.

Takuma Murakam et al. also worked with binding-time analysis based on MetaML [9], but they solved it using the maximum marking problem, and used the program transformation *optimization theorem* for maximum marking to achieve an effective algorithm. This also had the advantage of having a formal computational complexity and guaranteeing the optimality of the solutions based on a given weight function. They also allow the user to define their own weight measurements.

Kenichi Asai introduced binding-time analysis to MetaOCaml [1], a staged language. He relates the 2-level λ-calculus with staged λ-calculus. The advantage of his approach is that the optimal binding times can be found easier in the 2-level λ-calculus, avoiding searching or complex constraint solving. This is in opposition to Sheard and Linger, who also considered multiple stages and polymorphism. Kenichi Asai thinks it is not clear if his approach supports these two features.

These works show that even the type-theoretic viewpoint has several separate solutions. Our contribution concerns the use of classic type inference algorithms for this purpose.

3 Additional uses of type inference

3.1 Binding-time analysis

Binding-time analysis is used to reason about the availability of the data, that is, which parts of a program we can evaluate at compile time. Guided

by this information, we can produce a residual program, which only contains computations that cannot be executed at compile time. We will be using the terms "dynamic", "late-bound" interchangeably.

Binding-time analysis aims to produce a minimal completion - the least amount of program parts should be dynamic. Different solutions were proposed for this problem, one of them being the use of type inference and of two-level expressions.

We can view this analysis as a type inference problem, where we are trying to produce a completion which satisfies some typing rules.

However, as mentioned, these can be also used for other problems, namely type checking dynamic types, and providing more comprehensive type errors in statically typed languages.

3.2 Dynamic typing

We can split functional programming languages into two separate groups, statically typed ones like Haskell, ML and dynamically typed languages, e. g. Lisp, Scheme, Clojure and Erlang. In statically typed languages types are known at compile-time. This means that we don't need typechecking in the runtime, which can be a performance bonus. This can also replace some basic tests, as it narrows down the domain of possible input values. In a rich type system types can also act as documentation, detailing the computation itself.

On the other hand dynamic languages can offer great flexibility at the cost of some performance. In the former languages through many so-called typechecking algorithms we have assurances, that no type errors will arise at runtime. This analysis can be done in compile time. In dynamically typed languages we must keep track of types in the runtime, and type errors often only can be found at runtime. However there exist many separate approaches to provide as much type safety as we can while keeping dynamic typing.

Staged type inference is run multiple times. One possibility is to separate dynamic and static parts, and run checks on them in compile time and runtime.

Success typing finds definite type clashes without changing the source language fundamentally, and was implemented for Erlang in the popular static analysis tool Dialyzer [7].

We will present an approach in this paper, which finds all possible sources of type errors in the program, and underline them. Later, we will discuss ways we can improve the performance and enrich the languages we can examine, and other possible uses. Our algorithm, $\mathcal{B}$ is based on the one presented by Gomard [4].

4 Overview

These problems are quite similar: we are searching for points where the expression is not well-formed in regards to some predefined rules. In the former it means that these conflicting parts cannot be executed at compile time, and in the latter that we don't know the types statically.

Viewed from a type-theoretic angle these can be handled as type inference problems: inferring a *completion* of a term with annotations so that it is well-typed in regards to some type rules. Gomard proved that for every term of his two-level λ-calculus exists a completion that minimizes late binding.

Figures 1 and 2 show such a system. This is almost directly lifted from Gomard [4], the difference being that we use Λ to mean the type *Untyped* for the sake of brevity.

For our purposes we will be using lambda calculus with constants, conditional branching, and an explicit fixpoint-operator. We will be using a Curry-style type system. There are several implications of this: firstly, we can view untyped programs as potentially typed programs without annotations, and the expressions whose types can be inferred as well-typed.

Naturally, this technique can be used with other type systems that have decidable type inference capabilities. We will be using a modified version of Algorithm $\mathcal{M}$ ($\underline{\mathcal{M}}$) instead of $\mathcal{W}$. $\mathcal{M}$ is also sound and complete, and was shown to always find type errors earlier than $\mathcal{W}$ [6]. We will examine how this behaviour is affected by the modified type system.

The extended lambda calculus with annotations is defined by the following production:

$$\begin{aligned} e ::= & x \mid \lambda x.e' \mid e' @ e'' \mid if\ e'\ e''\ e''' \mid fix\ e \mid const_n\ base{-}value_n \mid \\ & \underline{x} \mid \underline{\lambda} x.e' \mid e' \underline{@} e'' \mid \underline{if}\ e'\ e''\ e''' \mid \underline{fix}\ e \mid \underline{const}_n\ base{-}value_n \end{aligned}$$

where x ranges over a class of variables, n is a natural number, $base-value_n$ is member of the set of curried n-ary base functions on first order values. and fix is a fix-point operator.

Binding time information is represented by types. Our types τ are generated like so:

$$\tau ::= Base_\omega \mid \Lambda \mid \alpha \mid \tau' \rightarrow_\omega \tau''$$

Base is a type constant, it represents concrete static values - values known at compile-time. Λ stands for values not known at compile-time, or in the second use case it means an unknown type at compile-time. α denotes the set of type variables, and $\tau_1 \rightarrow_\omega \tau_2$ is a function type. ω is a sequence describing

where the type was inferred in the examined expression. For instance in the expression $(\lambda x.x)@const$ the right hand side's type is $Base_{Arg::*}$.

$$\omega ::= * \mid Fun :: \omega \mid Arg :: \omega \mid Body :: \omega \mid FixBody :: \omega \mid$$
$$Cond :: \omega \mid Then :: \omega \mid Else :: \omega$$

$$\frac{\Gamma \vdash e : \tau \to \tau}{\Gamma \vdash fix\ e : \tau} \quad [\text{FIX1}]$$

$$\frac{\Gamma \vdash e_1 : Base \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash if\ e_1\ e_2\ e_3 : \tau} \quad [\text{IF1}]$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ @\ e_2 : \tau_1} \quad [\text{APP1}]$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \to \tau_2} \quad [\text{ABS1}]$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad [\text{VAR1}]$$

$$\Gamma \vdash const : Base \quad [\text{CON1}]$$

$$\Gamma \vdash const_1 : Base \to Base \quad [\text{CONFN1}]$$

Figure 1: Basic expressions

$$\frac{\Gamma \vdash e : \Lambda}{\Gamma \vdash \underline{fix}\ e : \Lambda} \quad [\text{FIX2}]$$

$$\frac{\Gamma \vdash e_1 : \Lambda \quad \Gamma \vdash e_2 : \Lambda \quad \Gamma \vdash e_3 : \Lambda}{\Gamma \vdash \underline{if}\ e_1\ e_2\ e_3 : \Lambda} \quad [\text{IF2}]$$

$$\frac{\Gamma \vdash e_1 : \Lambda \quad \Gamma \vdash e_2 : \Lambda}{\Gamma \vdash e_1\ \underline{@}\ e_2 : \Lambda} \quad [\text{APP2}]$$

$$\frac{\Gamma, x : \Lambda \vdash e : \Lambda}{\Gamma \vdash \underline{\lambda} x.e : \Lambda} \quad [\text{ABS2}]$$

$$\Gamma \vdash \underline{x} : \Lambda \quad [\text{VAR2}]$$

$$\Gamma \vdash \underline{const} : \Lambda \quad [\text{CON2}]$$

$$\Gamma \vdash \underline{const_1} : \Lambda \quad [\text{CONFN2}]$$

Figure 2: Underlined expressions

Binding-time information is represented syntactically: a late-bound/dynamic operator is different from it's matching basic version by being underlined. A λ-term with these annotations is called an annotated λ-term. If a term satisfies the inference rules in Figures 2 and 1 we also call it "well-annotated".

We call an untyped λ-term the erasure of an annotated λ-term e if the untyped term has no operator and type annotations, but otherwise is identical to e. A *completion* of a non-annotated term e with reference to some typing assumptions $\mathcal{A}$ is a well-annotated term w.r.t $\mathcal{A}$ with the erasure e.

We can get a trivial completion for every untyped λ-term by underlining all of the subexpressions. A minimal completion is a completion with the minimal amount of late-binding operators. Gomard presented a modified $\mathcal{W}$ ($\underline{\mathcal{W}}$) algorithm, and conjectures that using it we can compute minimal completions, which we will compare to our $\underline{\mathcal{M}}$ algorithm.

We give a possible definition of Gomard's algorithm ($\mathcal{B}'$). Then we will detail ours ($\mathcal{B}$).

$$\mathcal{B}'(e,\Gamma) = \begin{cases} \mathcal{B}'(U(e,\omega),\Gamma),\ if\ \underline{\mathcal{W}}(e,\Gamma,*) = FAIL(\omega) \\ e\ otherwise \end{cases}$$

$$\mathcal{B}(e,\Gamma) = \begin{cases} \mathcal{B}(U(e,\omega),\Gamma),\ if\ \underline{\mathcal{M}}(\Gamma,e,\beta,*) = FAIL(\omega),\ new\ \beta \\ e\ otherwise \end{cases}$$

We provide no definition for the U function, which has the sole purpose of underlining the operator in the e parameter which is pointed to by the ω occurrence parameter. Both $\underline{\mathcal{M}}$ and $\underline{\mathcal{W}}$ are extended to support underlined operators. The $\underline{\mathcal{M}}$ algorithm is detailed in Fig. 4.

To summarize both $\mathcal{B}$ and $\mathcal{B}'$: both algorithms run their respective type inference algorithms first, to check if the e input is well typed w.r.t the Γ type environment. If the type of e is inferred successfully, then it is finished. If $\underline{\mathcal{M}}$ or $\underline{\mathcal{W}}$ announce failure, we iterate once more, but with a modified e, which has the offending operator underlined.

Theorem 1 (Oukseh Lee, Kwangkeun Yi [6]) *Let Γ be a type environment, e an expression and β a new type variable. Then*

$$\left|[\![\mathcal{M}(\Gamma,e,\beta)]\!]\right| \leq \left|[\![\mathcal{W}(\Gamma,e)]\!]\right|$$

$\mathcal{M}$'s call string is shorter or equal to $\mathcal{W}$'s. Put in other words, the first type inference algorithm makes fewer or the same amount of recursive calls. We will see later what this means in practice

$\mathcal{W}$ is bottom-up, and $\mathcal{M}$ is top-down, they process expressions in a different order. The former infers the type of every subexpression, and then tries to unify them, while the latter tries to check each subexpression after another with a constraint, which will be unified at the "leaf nodes". This means that it often catches errors earlier that violate the constraint. The main difference can be seen with e_1 @ e_2: it is possible, that a not well-typed subexpression is typechecked by $\mathcal{W}$ before it gets into conflict at the application. In the cases where the left-hand side of the application is not a valid function, $\mathcal{M}$ only has to examine it, while $\mathcal{W}$ has to check both and then unify.

$$mgu : Type \times Type \to Subst$$

$$\begin{aligned}
mgu(Base_\omega, Base_{\omega'}) &= nullSubst \\
mgu(\Lambda, \Lambda) &= nullSubst \\
mgu(\alpha, t) &= FAIL(\omega),\ if\ t\ is\ l \to_\omega r\ and\ \alpha \in ftv(t), \\
&\quad \{n \mapsto t\}\ otherwise \\
mgu(t, \alpha) &= symmetric\ to\ the\ case\ above \\
mgu(l \to_\omega r, l' \to_{\omega'} r') &= let\ s_1 = mgu(l, l') \\
&\qquad\qquad s_2 = mgu(s_1 r, s_1 r') \\
&\quad in\ s_1 s_2 \\
mgu(t, t') &= FAIL(\omega),\ as\ one\ of\ the\ types\ must\ be\ Base_\omega \\
&\quad or\ l \to_\omega r
\end{aligned}$$

Figure 3: The unification used. $t,\ t',\ l,\ l',\ r,\ r' \in \tau$ are arbitrary types, α is a type variable.

$$\underline{\mathcal{M}} : TypeEnv \times Expr \times Type \times Occurrence \to Subst$$

$$\begin{array}{lll}
\underline{\mathcal{M}}(\Gamma, e_1 \ @ \ e_2, \tau, \omega) & = let\ s_1 = \underline{\mathcal{M}}(\Gamma, e_1, \beta \to_\omega \tau, Fun::\omega),\ new\ \beta & (1.1)\\
& \quad\ s_2 = \underline{\mathcal{M}}(s_1\ \Gamma, e_2, s_1\ \beta, Arg :: \omega) & (1.2)\\
& \ in\ s_2\ s_1 & \\
\underline{\mathcal{M}}(\Gamma, e_1\ \underline{@}\ e_2, \tau, \omega) & = let\ s_1 = mgu(\tau, \Lambda) & (2.1)\\
& \quad\ s_2 = \underline{\mathcal{M}}(s_1\Gamma, e_1, \Lambda, Fun :: \omega) & (2.2)\\
& \quad\ s_3 = \underline{\mathcal{M}}(s_2\ s_1\ \Gamma, e_2, \Lambda, Arg :: \omega) & (2.3)\\
& \ in\ s_3\ s_2\ s_1 & \\
\underline{\mathcal{M}}(\Gamma, \lambda.e, \tau, \omega) & = let\ s_1 = mgu(\tau, \beta_1 \to_\omega \beta_2),\ new\ \beta_1,\ \beta_2 & (3.1)\\
& \quad\ s_2 = \underline{\mathcal{M}}(s_1\Gamma + x : s_1\beta_1, e, s_1\beta_2, Body :: \omega) & (3.2)\\
& \ in\ s_2\ s_1 & \\
\underline{\mathcal{M}}(\Gamma, \underline{\lambda}.e, \tau, \omega) & = let\ s_1 = mgu(\tau, \Lambda) & (4.1)\\
& \quad\ s_2 = \underline{\mathcal{M}}(s_1\ \Gamma + x : \Lambda, e, \Lambda, Body :: \omega) & (4.2)\\
& \ in\ s_2\ s_1 & \\
\underline{\mathcal{M}}(\Gamma, var\ n, \tau, \omega) & = mgu(\tau, \Gamma(n)) & \\
\underline{\mathcal{M}}(\Gamma, \underline{var}\ n, \tau, \omega) & = mgu(\tau, \Lambda) & \\
\underline{\mathcal{M}}(\Gamma, const, \tau, \omega) & = mgu(\tau, Base_\omega) & \\
\underline{\mathcal{M}}(\Gamma, \underline{const}, \tau, \omega) & = mgu(\tau, \Lambda) & \\
\underline{\mathcal{M}}(\Gamma, const_1, \tau, \omega) & = mgu(\tau, Base_\omega \to_\omega Base_\omega) & \\
\underline{\mathcal{M}}(\Gamma, \underline{const}_1, \tau, \omega) & = mgu(\tau, \Lambda) & \\
\underline{\mathcal{M}}(\Gamma, if\ e_1\ e_2\ e_3, \tau, \omega) & = let\ s_1 = \underline{\mathcal{M}}(\Gamma, e_1, Base_\omega, Cond :: \omega) & (11.1)\\
& \quad\ s_2 = \underline{\mathcal{M}}(s_1\ \Gamma, e_2, s_1\tau, Then :: \omega) & (11.2)\\
& \quad\ s_3 = \underline{\mathcal{M}}(s_2\ s_1\ \Gamma, e_3, s_2\ s_1\tau, Else :: \omega) & (11.3)\\
& \ in\ s_3\ s_2\ s_1 & \\
\underline{\mathcal{M}}(\Gamma, \underline{if}\ e_1\ e_2\ e_3, \tau, \omega) & = let\ s_1 = mgu(\tau, \Lambda) & (12.1)\\
& \quad\ s_2 = \underline{\mathcal{M}}(s_1\Gamma, e_1, \Lambda, Cond :: \omega) & (12.2)\\
& \quad\ s_3 = \underline{\mathcal{M}}(s_2 s_1\ \Gamma, e_2, \Lambda, Then :: \omega) & (12.3)\\
& \quad\ s_4 = \underline{\mathcal{M}}(s_3 s_2 s_1\ \Gamma, e_3, \Lambda, Else :: \omega) & (12.4)\\
& \ in\ s_4\ s_3\ s_2\ s_1 & \\
\underline{\mathcal{M}}(\Gamma, fix\ e, \tau, \omega) & = \underline{\mathcal{M}}(\Gamma, e, \tau \to_\omega \tau, FixBody :: \omega) & (13.1)\\
\underline{\mathcal{M}}(\Gamma, \underline{fix}\ e, \tau, \omega) & = let\ s_1 = mgu(\tau, \Lambda) & (14.1)\\
& \quad\ s_2 = \underline{\mathcal{M}}(s_1\Gamma, e, \Lambda, FixBody :: \omega) & (14.2)\\
& \ in\ s_2\ s_1 &
\end{array}$$

Figure 4: The modified M algorithm

5 Proof of correctness

Lemma 2 (Damas and Milner) *If S is a substitution and $\Gamma \vdash e : \tau$, then $S\Gamma \vdash e : S\tau$* [8]

Lemma 3 ($\underline{\mathcal{M}}$ is sound) *Let e be an expression and Γ a type environment. If there exist a type τ such that $\underline{\mathcal{M}}(\Gamma, e, \tau, *) = S$, then $S\Gamma \vdash e : S\tau$.*

This section heavily builds upon the work of Oukseh Lee and Kwangkeun Yi [6]. The only differences are that we have a new constant type, Λ, and that we are not using polymorphism in this instance. We will use structural induction on e.

Proof.

- **case** *const*: $S\tau = SBase_\omega = Base_\omega$. So $S\Gamma \vdash const : S\tau$ by [CON1].
- **case** $\underline{const}$: $S\tau = S\Lambda = \Lambda$. So $S\Gamma \vdash \underline{const} : S\tau$ by [CON2].
- **case** $const_1$: $S\tau = S(Base_\omega \to Base_{\omega'}) = Base_\omega \to Base_{\omega'}$.
  So $S\Gamma \vdash const_1 : S\tau$ by [CONFN1].
- **case** $\underline{const}_1$: $S\tau = S\Lambda = \Lambda$. So $S\Gamma \vdash \underline{const}_1 : S\tau$ by [CONFN2].
- **case** x: $S\tau = S\Gamma(x)$. So $S\Gamma \vdash x : S\tau$ by [VAR1].
- **case** $\underline{x}$: $S\tau = S\Lambda = \Lambda$. So $S\Gamma \vdash \underline{x} : S\tau$ by [VAR2].
- **case** $\lambda x.e$:
  1. By induction, (3.2) implies
     $$S_2S_1\Gamma + x : S_2S_1\beta_1 \vdash e : S_2S_1\beta_2.$$
  2. By [ABS1] $S_2S_1\Gamma \vdash \lambda x.e : S_2S_1\beta_1 \to S_2S_1\beta_2$; which is, by (3.1)
     $$S_2S_1\Gamma \vdash \lambda x.e : S_2S_1\tau.$$
- **case** $\underline{\lambda}x.e$:
  1. By induction, (4.2) implies
     $$S_2S_1\Gamma + x : S_2\Lambda \vdash e : S_2\Lambda. \text{ So } S_2S_1\Gamma + x : \Lambda \vdash e : \Lambda.$$

2. By [ABS2] $S_2S_1\Gamma \vdash \underline{\lambda}x.e : S_2S_1\Lambda$; which is, by (4.1)

$$S_2S_1\Gamma \vdash \underline{\lambda}x.e : S_2S_1\tau.$$

- **case** $e_1@e_2$:

1. By induction, (1.1) implies $S_1\Gamma \vdash e_1 : S_1(\beta \to \tau)$. By Lemma 2, we can apply S_2 to both sides.

$$S_2S_1\Gamma \vdash e_1 : S_2S_1\beta \to S_2S_1\tau$$

2. By induction, (1.2) implies

$$S_2S_1\Gamma \vdash e_2 : S_2S_1\beta.$$

3. By [APP1],

$$S_2S_1\Gamma \vdash e_1@e_2 : S_2S_1\tau.$$

- **case** $e_1\underline{@}e_2$:

1. By induction, (2.2) implies $S_2S_1\Gamma \vdash e_1 : S_2S_1\Lambda$. By Lemma 2, we can apply S_3 to both sides.

$$S_3S_2S_1\Gamma \vdash e_1 : S_3S_2S_1\Lambda. \text{ So } S_3S_2S_1\Gamma \vdash e_1 : \Lambda.$$

2. By induction, (2.3) implies

$$S_3S_2S_1\Gamma \vdash e_2 : S_3\Lambda. \text{ So } S_3S_2S_1\Gamma \vdash e_2 : \Lambda.$$

3. By [APP2] $S_3S_2S_1\Gamma \vdash e_1\underline{@}e_2 : \Lambda$. So $S_3S_2S_1\Gamma \vdash e_1\underline{@}e_2 : S_3S_2S_1\Lambda$. That is, by (2.1),

$$S_3S_2S_1\Gamma \vdash e_1\underline{@}e_2 : S_3S_2S_1\tau.$$

- **case** $if\ e_1\ e_2\ e_3$:

1. By induction, (11.1) implies $S_1\Gamma \vdash e_1 : S_1Base$. By Lemma 2, we can apply S_2 and S_3 to both sides:

$$S_3S_2S_1\Gamma \vdash e_1 : S_3S_2S_1Base.$$

2. By induction, (11.2) implies $S_2S_1\Gamma \vdash e_2 : S_2S_1\tau$. By Lemma 2, we can apply s_3 to both sides:

$$S_3S_2S_1\Gamma \vdash e_2 : S_3S_2S_1\tau.$$

3. By induction, (11.3) implies

$$S_3S_2S_1\Gamma \vdash e_3 : S_3S_2S_1\tau.$$

4. By [IF1],

$$S_3S_2S_1\Gamma \vdash if\ e_1\ e_2\ e_3 : S_3S_2S_1\tau$$

- **case** $\underline{if}\ e_1\ e_2\ e_3$:

1. By induction, (12.2) implies $S_2S_1\Gamma \vdash e_1 : S_2\Lambda$. By Lemma 2, we can apply S_3 and S_4 to both sides:

$$S_4S_3S_2S_1\Gamma \vdash e_1 : S_4S_3S_1\Lambda. \text{ So } S_4S_3S_2S_1\Gamma \vdash e_1 : \Lambda.$$

2. By induction, (12.3) implies $S_3S_2S_1\Gamma \vdash e_2 : S_3\Lambda$. By Lemma 2, we can apply S_4 to both sides:

$$S_4S_3S_2S_1\Gamma \vdash e_2 : S_4S_3\Lambda. \text{ So } S_4S_3S_2S_1\Gamma \vdash e_2 : \Lambda.$$

3. By induction, (12.4) implies

$$S_4S_3S_2S_1\Gamma \vdash e_3 : S_4\Lambda. \text{ So } S_4S_3S_2S_1\Gamma \vdash e_3 : \Lambda.$$

4. By [IF2] $S_4S_3S_2S_1\Gamma \vdash \underline{if}\ e_1\ e_2\ e_3 : \Lambda$. So $S_4S_3S_2S_1\Gamma \vdash \underline{if}\ e_1\ e_2\ e_3 : S_4S_3S_2S_1\Lambda$, that is, by (12.1)

$$S_4S_3S_2S_1\Gamma \vdash \underline{if}\ e_1\ e_2\ e_3 : S_4S_3S_2S_1\tau$$

- **case** $fix\ e$:

1. By induction, (13.1) implies $S_1\Gamma \vdash e : S_1(\tau \to \tau)$, which is by definition

$$S_1\Gamma \vdash e : S_1\tau \to S_1\tau.$$

2. By [FIX1],

$$S_1\Gamma \vdash fix\ e : S_1\tau.$$

- **case** $\underline{fix}\ e$:

1. By induction, (14.2) implies

$$S_2S_1\Gamma \vdash e : S_2\Lambda. \text{ So } S_2S_1\Gamma \vdash e : \Lambda.$$

2. By [FIX2] $S_2S_1\Gamma \vdash \underline{fix}\ e : \Lambda$. So $S_2S_1\Gamma \vdash \underline{fix}\ e : S_2S_1\Lambda$, that is, by (14.1)
$$S_2S_1\Gamma \vdash \underline{fix}\ e : S_2S_1\tau.$$

□

Lemma 4 *No operator is underlined twice.*

Proof. To be subject to underlining, an operator has to have their occurrence in the exception raised by the *mgu* algorithm. This can only happen if the occurrence is in one of *mgu*'s parameters. There are two ways some o operator's occurrence can be one of the the parameters of an *mgu* call:

- This can happen directly if we call *mgu* with o's occurence when o is processed (see *const*).
- If we make a recursive call to $\underline{\mathcal{M}}$, where the type parameter has o's occurrence (see 1.1), which is unified later.

If we take a look at $\underline{\mathcal{M}}$'s branches where the expression parameter is underlined, we can see that neither of these conditions are met. □

Theorem 5 (The $\mathcal{B}$ algorithm is correct) *Let e be some starting lambda expression and Γ a starting type environment. $\mathcal{B}(e,\Gamma)$ completes in finite steps, and for the expression e', where $\mathcal{B}(e,\Gamma) = e'$:*

- *There exist some τ type and S substitution, such that $S\Gamma \vdash e' : Sp$.*
- *e is the erasure of e'.*

Proof. Let $n(e) \in \mathbb{N}$ be the number of underlined operators in e, and $s(e) \in \mathbb{N}$ be the total number of operators in the expression in e, underlined or otherwise. It is trivial, that for every e λ-term $n(e) \leq s(e)$. On each recursive step, based upon $\underline{\mathcal{M}}$'s result either:

- $\underline{\mathcal{M}}(\Gamma, e, \tau, *) = FAIL(\omega)$ for some new τ. We underline the indicated operator, $e' = U(e,\omega)$. Because of Lemma 4 we can say that this is a new underlining, so $n(e') = n(e) + 1$. We continue the recursion with e'.
- If $\underline{\mathcal{M}}(\Gamma, e, \tau, *) = S$, then according to Lemma 3 we get $S\Gamma \vdash e : Sp$. This means that the expression e in the environment Γ is well typed w.r.t. the type inference system presented in figures 1 and 2, so e is correctly annotated.

If follows, that if $\mathcal{B}$ stops, the first case is true, so we have the correct output. This means that the only way $\mathcal{B}$ be incorrect, is if it doesn't halt. But because $n(e)$ grows by one between each recursive call, we must reach the state where $n(e) = s(e)$. If $\underline{\mathcal{M}}(\Gamma, e, \tau, *) = FAIL(\omega)$, then we must underline a new non-underlined operator, but because $n(e) = s(e)$, there is no operator that is lined.

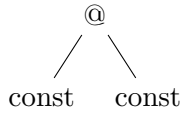
We defined U as a function that only underlines operators. Between every recursion we modify the expression in no other way, meaning that the output expression's erasure is the initial input expression. □

6 Examples

In this section we will be taking a look at a few examples, and how our algorithm's behaviour differs from that of Gomard's.

6.1 Simple expression

Let us examine the expression *const* @ *const*, where *const*-s are some arbitrary constants. The current analysis doesn't differentiate between constant types, so this could consist of any two expressions with such types like Int, $Bool$. $\underline{\mathcal{W}}$ infers the type of the left-hand side, then the right-hand side, then fails at unification:



$$t_{left} = t_{right} \to T$$

(where T is a new type variable), because t_{left} and t_{right} are both $Base$. In case of a unification failure, the occurrence of one of the types is sent, which in our implementation's case is that of the left. We can see that on each round we must check both subexpressions.

$\underline{\mathcal{M}}$ however finds this problem one step earlier. According to the algorithm above, it checks if the left subexpression has a function type, and here it fails, because the type found is $Base_\omega$ and fails with the same occurrence as earlier.

After underlining the expression we try again. We fail at the same place, but this time it is with the constraint

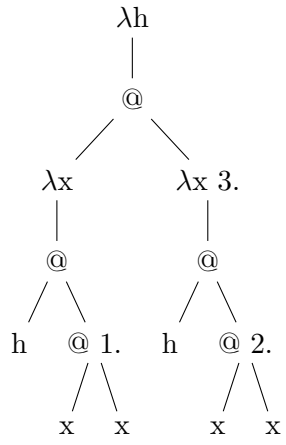
$$\Lambda = \tau \to T$$

We have the occurrence of the equation's right side as error message. The selector in question is empty, because the application is the root of the whole

expression. This means that we will underline the application. Upon restarting, we branch again at the underlined application. This time left-hand side checks out, because we both expect and get Λ. There is conflict on the right however: we expect Λ, but have $Base$. The second *const* gets underlined, then we restart the algorithm again, which is finally successful.

6.2 Y combinator

For this second example we will take a look at the Y combinator. This fixpoint-combinator is not well-typed, so we will need to underline the problem areas to conform to the type rules. We have numbered these in the figure for an easier understanding. The numbering is according to the order in which $\underline{\mathcal{W}}$ finds them. Both algorithms find 1. first. Here $\underline{\mathcal{M}}$ examines the same number of subexpressions, we don't gain any advantage at the application. This is because the left-hand side's type is a type variable that hasn't had any constraints or substitutions placed upon it, so it can be unified with a function type. [1] Conflict only arises after visiting the other side with



$$T = \tau \to T$$

The rest of the errors are found earlier. In these two cases the different ways of inferring application types do come in to play. To summarize, the main cause of difference is that at applications $\underline{\mathcal{W}}$ has to infer both children, while in some cases $\underline{\mathcal{M}}$ can do it by only one.

7 Implementation and random testing

Random testing compiler optimizations have become more common practice in the last decades as computing power grew and the necessary frameworks and algorithms were established. We will be using the package QuickCheck [2]

[1] This is why $\lambda x.x@x$ takes the same number of visited nodes despite being similar to the first example.

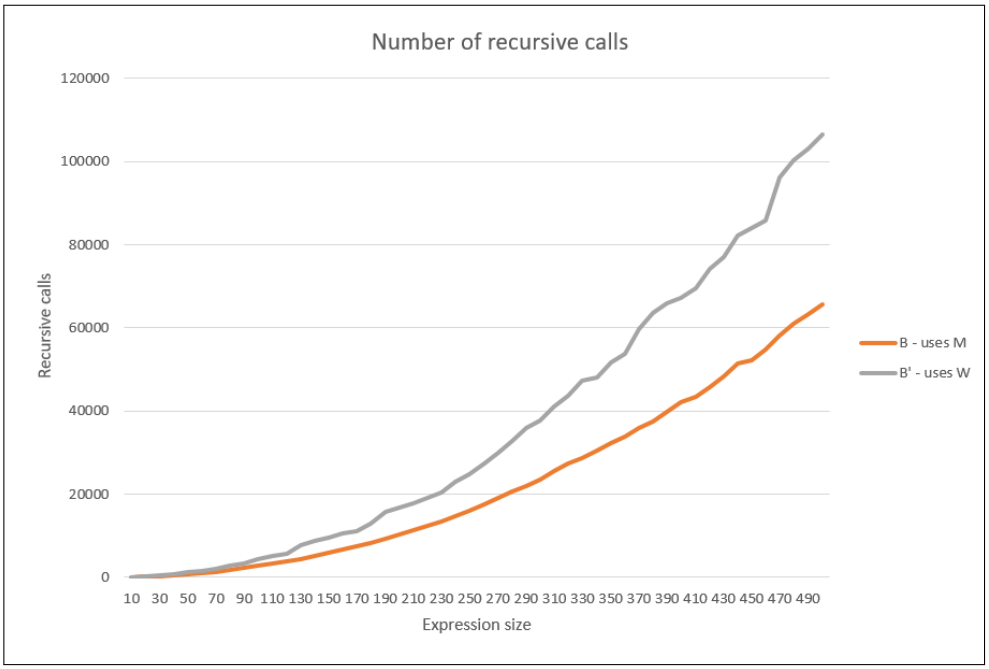


Figure 5: Number of recursive calls for a set expression size

with a random expression generator to test the two approaches discussed in this paper.

The size of an expression e is:

$$size(e) = \begin{cases} 1, & \text{if } e \equiv x \\ 1 + size(f) + size(g), & \text{if } e \equiv (f@g) \\ 1 + size(f), & \text{if } e \equiv (\lambda x.f) \end{cases}$$

We will generate random untyped lambda expressions and calculate the total number recursive calls made by the inference algorithms depending on the size of the input. We will be using and algorithm developed by Jue Wang [12]. This algorithm generates random λ-terms of a given size, assuming a uniform distribution over all terms of a given size.

These calculations will be run with empty starting type environments. For each size we will generate a hundred expressions, and take the average of the visited nodes/recursive steps made by the algorithms.

We can see that the numbers of recursive calls are super linear in both cases but there is a noticeable difference in the multiplier. This data corresponds only to the number of recursive calls made by $\underline{\mathcal{M}}$ and $\underline{\mathcal{W}}$. It is important to keep in mind that this doesn't take the cost of unification into consideration. This is why we have also included the time spent by both algorithms in the following figure.

Mean execution time		
Size	$\mathcal{B}$	$\mathcal{B}'$
100	1.09 ms	1.91 ms
250	3.93 ms	6.84 ms
500	88.4 ms	113 ms
1000	1.10 s	1.58 s

Both $\mathcal{B}$ and $\mathcal{B}'$ were implemented in Haskell. The unification algorithm and the definitions describing the type system and the expressions are common between the two algorithms. The code for generating λ-terms is also available.[2] To facilitate exception throwing and the calls for new type variables, we used a monad stack in the type inference algorithms.

```
type TyVar = Int

data Type = TBase [Occurrence]
          | TArrow Type Type [Occurrence]
          | TVar TyVar
          | TUntyped

type TI a =  ExceptT [Occurrence] (State TyVar) a
```

The core types and definitions are located in *Common*, $\mathcal{B}$ and $\mathcal{B}'$ are found in *AlgoM* and *AlgoW* respectively. A basic parser is provided in *Parser* for interactive testing. *Testing* includes the functions and generators used for random testing.

8 Conclusion

Binding time analysis and partial evaluation are well-researched fields. Possible advances can only mostly be done to better efficiency and support a wider

[2]Link to publicly available repository: https://github.com/szokolimatyas/Type-Inference-for-BTA

range of language features. This paper proposes some modest improvements to an existing algorithm without great changes. These improvements speed up the algorithm, but not by orders of magnitude, still having a super-linear complexity.

It is not possible to achieve linearity for these types of "restarting" inference algorithms, as in the worst case - which means every operator needs to be underlined - for an expression with a size s the complexity is $s*\theta(k)$. One run of the algorithm tries to find the next operator that needs to be underlined, which $(\theta(k))$ is almost linear in practice. If we used the Hindley-Milner type system, the inference algorithm would mostly have a polynomial complexity. However, there are some pathological cases where it is non-linear [14]. Intuitively, we run the algorithm s times, with the time complexity proportional to s each time.

It would be remiss if we did not mention the work of Henglein [5], who took a more constraint-oriented approach based on Gomard, proposing a near linear time solution to binding time analysis.

There are some more efficient algorithms provided for BTA as mentioned in the related works, but this direction still can be more thoroughly explored. Neither was the use of type inference for detecting type conflicts in untyped programs deeply examined formerly.

We feel that this algorithm has legitimacy, because it can be applied to a wide range of problems, promises good prospects for further advancement, and is easily derived from existing approaches, providing an easy to understand solution to binding time analysis and analyzing dynamically typed programs.

9 Future work

Type inference algorithms have been developed in more descriptive type systems and we think that the same type of modifications could also be applied there to produce similar results. Namely, the two algorithms described in this paper can also be used in the Hindley-Milner system, even with let-polymorphism. One hindrance is that in some cases inference is not decidable [13]. This can cause problems if we wanted to use it for better error messages in statically typed languages. Binding time analysis, however, could fare better, as we don't differentiate between type constants, so we could avoid some of the ambiguity.

We have not implemented the proposed changes by Gomard: the addition of lifting and reporting more than one occurrences is possible avenue for improvement. It is also trivial to change $\underline{\mathcal{M}}$ to support let-polymorphism.

Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] K. Asai, Toward introducing binding-time analysis to MetaOCaml, *Proceedings of the 2016 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, St. Petersburg, FL, USA, 2016, pp. 97-102. ⇒233

[2] K. Claessen, J. Hughes, Quickcheck: A lightweight tool for random testing of Haskell programs, *SIGPLAN* **35,** 9 (2000) 268—279. ⇒245

[3] C. Gomard, N. Jones, A partial evaluator for the untyped lambda calculus, *Journal of Functional Programming* **1,** 1 (1991) 21–69. ⇒233

[4] C. K. Gomard, Partial type inference for untyped functional programs, *Proceedings of the 1990 ACM Conference on LISP and Functional Programming*, Nice, France, 1990, pp. 282–287. ⇒232, 234, 235

[5] F. Henglein, Efficient type inference for higher-order binding-time analysis, *Conference on Functional Programming Languages and Computer Architecture*, *Lecture Notes in Computer Science* **523** (1991) 448—472. ⇒248

[6] O. Lee, K. Yi, Proofs about a folklore let-polymorphic type inference algorithm, *ACM Trans. Program. Lang. Syst.*, **20,** 4 (1998) 707—723. ⇒235, 237, 240

[7] T. Lindahl, K. Sagonas, Practical Type Inference Based on Success Typings, *Proc. 8th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, Venice, Italy, 2006, pp. 167–178. ⇒234

[8] R. Milner, L. Damas, Principal type-scheme for functional programs, *Proc. 9th ACM Symposium on Principles of Programming Languages* New York, USA, 1982, pp. 207–212. ⇒240

[9] T. Murakami, Z. Hu, K. Kakehi, M. Takeichi, An efficient staging algorithm for binding-time analysis, *International Symposium on Logic-Based Program Synthesis and Transformation*, *Lecture Notes in Computer Science* **3018** (2003) pp. 106-107. ⇒233

[10] H. R. Nielson, F. Nielson, Automatic binding time analysis for a typed lambda-calculus (extended abstract), *Proc. 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* San Diego, California, USA, 1988, pp. 98–106. ⇒233

[11] T. Sheard, N. Linger, Search-based binding time analysis using type-directed pruning, *Proceedings of the ASIAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, Aizu, Japan, 2002, pp. 20-–31. ⇒233
[12] J. Wang, Generating random lambda calculus terms, *Semantic Scholar*, 2005. https://www.semanticscholar.org ⇒246
[13] J. B. Wells, Typability and type checking in system F are equivalent and undecidable, *Annals of Pure and Applied Logic*, **98,** 1 (1999) 111 – 156. ⇒248
[14] H. G. Mairson, Deciding ML typability is complete for deterministic exponential time, *Proc. 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, New York, NY, United States, 1989, pp. 382-401. ⇒248
[15] B. C. Pierce, *Advanced Topics in Types and Programming Languages*, MIT Press, Cambridge, MA, 2004. ⇒233
[16] B. C. Pierce, *Types and Programming Languages*, MIT Press, Cambridge, MA, 2002. ⇒233

Received: June 9, 2020 • Revised: October 12, 2020

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 251–259

DOI: 10.2478/ausi-2020-0015

On degree sets in k-partite graphs

T. A. Naikoo
Islamia College of Science and Commerce, Srinagar, India
email: tariqnaikoo@rediffmail.com

U. Samee
Institute of Technology, University of Kashmir, Srinagar, India
email: drumatulsamee@gmail.com

S. Pirzada
University of Kashmir, Srinagar, India
email: pirzadasd@kashmiruniversity.ac.in

Bilal A. Rather
Department of Mathematics, University of Kashmir, India
email: bilalahmadrr@gmail.com

Abstract. The degree set of a k-partite graph is the set of distinct degrees of its vertices. We prove that every set of non-negative integers is a degree set of some k-partite graph.

1 Introduction

In a graph G, the degree of a vertex v_i, denoted by d_{v_i} (or simply d_i), is the number of edges which are incident on v_i. A sequence of non-negative integers $[d_1, d_2, \ldots, d_p]$ is called the degree sequence of a graph G if the vertices of G can be labelled $v_1, v_2, \ldots, v_p$ such that $\deg v_i = d_i$ for each i, $1 \leq i \leq p$. The terminology and notations used in this paper are same as in [6, 25].

The set of distinct degrees of the vertices of a graph is called its degree set. The following result can be found in [8].

Theorem 1 [8] *Any set* D *of distinct positive integers is the degree set of a connected graph and the maximum order of such a graph is* $M + 1$*, where* M *is the maximum integer in the set* D.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C20
Key words and phrases: bipartite graph, k-partite graph, degree, degree set

More on degree sets in graphs can be seen in [1, 2, 3, 4, 5, 9, 10, 11, 12, 13, 14, 15, 29, 30, 31]. Analogous results in directed graphs and signed graphs can be found in [17, 18, 19, 20, 21, 22, 23, 24, 26, 27].

2 Degree sets in k-partite graphs

A k-partite graph ($k \geq 2$) is a graph G whose vertex set can be partitioned into k nonempty disjoint sets $V_1, V_2, \ldots, V_k$, known as partite sets, such that $v_i v_j$ is an edge of G if v_i is in some V_i and v_j is in some V_j ($i \neq j$). A k-partite graph with partite sets $V_1, V_2, \ldots, V_k$ is denoted by $G(V_1, V_2, \ldots, V_k)$. For $k = 2$ and $K = 3$ we get respectively bipartite graph and 3-partite graph. Also, a k-partite graph $G(V_1, V_2, \ldots, V_k)$ is said to be connected if each vertex $v_i \in V_i$ is connected to every vertex $v_j \in V_j$ ($i \neq j$). The degree of a vertex v_i in a k-partite graph $G(V_1, V_2, \ldots, V_k)$ is the number of edges of $G(V_1, V_2, \ldots, V_k)$ which are incident to v_i and is denoted by d_{v_i} or d_i. Let $G(V_1, V_2, \ldots, V_k)$ be a k-partite graph with $V_i = \{v_{i1}, v_{i2}, \ldots, v_{ip_i}\}$, $1 \leq i \leq k$ and let $d_{i1}, d_{i2}, \ldots, d_{ip_i}$ be the respective degrees of $v_{i1}, v_{i2}, \ldots, v_{ip_i}$. Then the sequence $D_i = [d_{i1}, d_{i2}, \ldots, d_{ip_i}]$, $1 \leq i \leq k$, are called degree sequences of $G(V_1, V_2, \ldots, V_k)$.

The set of distinct degrees of the vertices of a k-partite graph $G(V_1, V_2, \ldots, V_k)$ is called its degree set.

The following result is given in [16].

Theorem 2 [16] *Every set of positive integers is a degree set of some connected bipartite graph.*

The following result can be seen in [7].

Theorem 3 [7] *Every set of positive integers, except* $\{1\}$ *is a degree set of some connected* 3*-partite graph.*

Now, we have the following observation.

Theorem 4 *Every singleton set of positive integers is a degree set of some* k*-partite graph.*

Proof. Let $D = \{d\}$, where d is a positive integer. For $d = 1$, construct a k-partite graph $G(V_1, V_2, \ldots, V_k)$ as follows

$$\begin{aligned} V_1 &= A_{11} \cup A_{12} \cup \cdots \cup A_{1(k-2)} \cup A_{1(k-1)}, \\ V_2 &= A_{21}, \\ V_3 &= A_{31}, \\ &\vdots \\ V_{k-1} &= A_{(k-1)1}, \\ V_k &= A_{k1}, \end{aligned}$$

with $A_{1p} \cap A_{1q} = \emptyset$ $(p \neq q)$, $|A_{ij}| = 1$ for all i, j where $1 \leq i \leq k$, $1 \leq j \leq k-1$. Let there be an edge from the vertex of $A_{1(i-1)}$ to the vertex of A_{i1}. Then the degrees of the vertices of $G(V_1, V_2, \ldots, V_k)$ are as follow.

For $2 \leq i \leq k$

$$d_{a_{1(i-1)}} = d_{a_{i-1}} = |A_{1(i-1)}| = 1 = d, \text{ for all } a_{1(i-1)} \in A_{1(i-1)},\ a_{i1} \in A_{i1}.$$

Therefore, degree set of $G(V_1, V_2, \ldots, V_k)$ is $D = \{d\}$.

Now we assume that $d \geq 2$. For $k = 2$, consider the bipartite graph $G(V_1, V_2)$ with $|V_1| = |V_2| = d$ and let there be an edge from each vertex of V_1 to every vertex of V_2. Then the degrees of the vertices of $G(V_1, V_2)$ are as follows.

$$d_{v_1} = d_{v_2} = |V_1| = d, \text{for all } v_1 \in V_1,\ v_2 \in V_2.$$

Therefore, degrees set of $G(V_1, V_2)$ is $D = \{d\}$.

If $k \geq 3$ is odd, say $k = 2m + 1$ where $m \geq 1$, construct a $2m + 1$-partite graph $G(V_1, V_2, \ldots, V_{2m+1})$ as follows.

Let $V_1 = A_1$, $V_2 = A_2 \cup B_2$, $V_3 = A_3, \ldots, V_{2m} = A_{2m}$, $V_{2m+1} = A_{2m+1}$ with $|A_i| = |B_2| = d - 1$ for all i, $1 \leq i \leq 2m + 1$, $A_2 \cap B_2 = \emptyset$. Let there be an edge **(i)** from each vertex of A_i to every vertex of A_{i+1} for all odd i, **(ii)** from distinct vertices of A_i to distinct vertices of A_{i+1} for all even i, **(iii)** from distinct vertices of A_1 to distinct vertices of B_2, and **(iv)** from each vertex of A_{2m+1} to every vertex of B_2. Then the degrees of the vertices of $G(V_1, V_2, \ldots, V_{2m+1})$ are as follows.

For $1 \leq i \leq 2m + 1$

$$d_{a_i} = d_{b_2} = |A_i| + 1 = d - 1 + 1 = d \text{ for all } a_i \in A_i \text{ and } b_2 \in B_2.$$

Therefore, degree set of $G(V_1, V_2, \ldots, V_{2m+1})$ is $\{d\}$.

Again, if $k \geq 4$ is even, say $k = 2m + 2$ where $m \geq 1$, consider a $2m + 2$-partite graph $G(V_1, V_2, \ldots, V_{2m+2})$ with $|V_i| = d - 1$ for all i, $1 \leq i \leq 2m + 2$.

Let there be an edge **(i)** from each vertex of V_i to every vertex of V_{i+1} for all odd i, **(ii)** from distinct vertices of V_i to distinct vertices of V_{i+1} for all odd i, and **(iii)** from distinct vertices of V_1 to distinct vertices of V_{2m+2}. Then the degrees of the vertices of $G(V_1, V_2, \ldots, V_{2m+2})$ are as follows

For, $1 \leq i \leq 2m+2$,

$$d_{v_i} = |V_i| + 1 = d - 1 + 1 = d, \text{ for all } v_i \in V_i.$$

Therefore, degree set of $G(V_1, V_2, \ldots, V_{2m+2})$ is $D = \{d\}$. □

Except for $d = 1$ and $k \geq 3$ in the proof of Theorem 4, the construction there yields a connected k-partite graph and we have the following result.

Corollary 5 *Every singleton set of positive integers is a degree set of some connected k-partite graph, except $\{1\}$ for $k \geq 3$ in which case the k-partite graph is not connected.*

Now, we obtain the following result.

Theorem 6 *Every set of positive integers is a degree set of some connected k-partite graph, except $\{1\}$ for $k \geq 3$ in which case the k-partite graph is not connected.*

Proof. Let $d_1, d_2, \ldots, d_n$ be positive integers. We will show that there is a connected k-partite graph $G(V_1, V_2, \ldots, V_k)$ with degree set $D = \{d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{n} d_i\}$, except when $d = 1$ and $k \geq 3$ in which case the k-partite graph is not connected.

The case $k = 2$ and $k = 3$ are respectively given in Theorem 2 and Theorem 3. Also the case $k = 1$ follows by Corollary 5. So, we assume $k \geq 4$ and $n \geq 2$.

For $k = 4$, construct a 4-partite graph $G(V_1, V_2, V_3, V_4)$ as follows. Let

$$\begin{aligned}
V_1 &= A_{11} \cup A_{12} \cup A_{13} \cup \cdots \cup A_{1(n-1)} \cup A_{1n},\\
V_2 &= A_{21} \cup A_{22} \cup A_{23} \cup \cdots \cup A_{2(n-1)},\\
V_3 &= A_{31} \cup A_{32} \cup A_{33} \cup \cdots \cup A_{3(n-1)} \cup A_{3n},\\
V_4 &= A_{41} \cup A_{42} \cup A_{43} \cup \cdots \cup A_{4(n-1)} \cup A_{4n},
\end{aligned}$$

with $A_{1p} \cap A_{1q} = \emptyset$, $A_{2p} \cap A_{2q} = \emptyset$, $A_{3p} \cap A_{3q} = \emptyset$, $A_{4p} \cap A_{4q} = \emptyset$ $(p \neq q)$, $|A_{1j}| = d_j$ for all j, $1 \leq j \leq n$, $|A_{2j}| = d_1 + d_2 + \cdots + d_j$ for all j, $1 \leq j \leq n-1$, $|A_{3j}| = d_j$ for all j, $1 \leq j \leq n$, $|A_{41}| = d_2$, $|A_{4j}| = d_1 + d_2 + \cdots + d_{j-1}$ for all j, $2 \leq j \leq n$. Let there be an edge **(i)** from each vertex of A_{1j} to every

vertex of A_{3r} whenever $j \geq r$, **(ii)** from each vertex of A_{11} to every vertex of A_{41}, **(iii)** from each vertex of A_{2j} to every vertex of $A_{3(j+1)}$, and **(iv)** from each vertex of A_{2j} to every vertex of $A_{4(j+1)}$. Then, the degrees of the vertices of $G(V_1, V_2, V_3, V_4)$ are as follows.

$$d_{a_{11}} = |A_{31}| + |A_{41}| = d_1 + d_2, \text{ for all } a_{11} \in A_{11},$$

$$\text{for } 2 \leq j \leq n,\ d_{a_{1j}} = \sum_{r=1}^{j} |A_{3r}| = \sum_{r=1}^{j} d_r, \text{ for all } a_{1j} \in A_{1j},$$

$$\text{for } 1 \leq j \leq n-1,\ d_{a_{2j}} = |A_{3(j+1)}| + |A_{4(j+1)}| = d_{j+1} + d_1 + \cdots + d_j = \sum_{r=1}^{j+1} d_r, \text{ for all } a_{2j} \in A_{2j},$$

$$d_{a_{31}} = \sum_{r=1}^{n} |A_{1r}| = \sum_{r=1}^{n} d_r, \text{ for all } a_{31} \in A_{31},$$

$$\text{for } 2 \leq j \leq n,\ d_{a_{3j}} = \sum_{r=j}^{n} |A_{1r}| + |A_{2(j-1)}| = \sum_{r=j}^{n} d_r + d_1 + \cdots + d_{j-1} = \sum_{r=1}^{n} d_r, \text{ for all } a_{3j} \in A_{3j},$$

$$d_{a_{41}} = |A_{11}| = d_1, \text{ for all } a_{41} \in A_{41},$$

$$\text{for } 2 \leq j \leq n,\ d_{a_{4j}} = |A_{2(j-1)}| = d_1 + d_2 + \cdots + d_{j-1} = \sum_{r=1}^{j-1} d_r, \text{ for all } a_{4j} \in A_{4j}.$$

Therefore, degree set of $G(V_1, V_2, V_3, V_4)$ is $D = \{d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{n} d_i\}$.

If $k \geq 5$ is odd say $k = 2m + 3$ where $m \geq 1$, construct a $2m + 3$-partite graph $G(V_1, V_2, \ldots, V_{2m+3})$ as follows.

Let

$$\begin{aligned}
V_1 &= A_{11} \cup A_{12} \cup A_{13} \cup \cdots \cup A_{1(n-1)} \cup A_{1n},\\
V_2 &= A_{21} \cup A_{22} \cup A_{23} \cup \cdots \cup A_{2(n-1)},\\
V_3 &= A_{31} \cup A_{32} \cup A_{33} \cup \cdots \cup A_{3(n-1)} \cup A_{3n},\\
V_4 &= A_{41} \cup B_{42} \cup A_{42} \cup A_{43} \cup \cdots \cup A_{4(n-1)} \cup A_{4n},\\
V_5 &= A_{51} \cup B_{52},\\
V_6 &= A_{61} \cup B_{62},\\
&\vdots\\
V_{2m+2} &= A_{(2m+2)1} \cup B_{(2m+2)2},\\
V_{2m+3} &= A_{(2m+3)1} \cup B_{(2m+3)2},
\end{aligned}$$

with $A_{1p} \cap A_{1q} = \emptyset$, $A_{2p} \cap A_{2q} = \emptyset$, $A_{3p} \cap A_{3q} = \emptyset$, $A_{4p} \cap A_{4q} = \emptyset$, $(p \neq q)$, $A_{4p} \cap B_{42} = \emptyset$, $A_{p1} \cap B_{p2} = \emptyset$, $|A_{1j}| = d_j$ for all j, $1 \leq j \leq n$, $|A_{2j}| = d_1 + d_2 + \cdots + d_j$ for all j, $1 \leq j \leq n-1$, $|A_{3j}| = d_j$ for all j, $1 \leq j \leq n$, $|A_{i1}| = d_2$, for all i, $4 \leq i \leq 2m+3$, $|B_{i2}| = d_1$ for all i, $4 \leq i \leq 2m+3$, $|A_{4j}| = d_1 + d_2 + \cdots + d_{j-1}$ for all $j, 2 \leq j \leq n$. Let there be an edge **(i)** from each vertex of A_{1j} to every vertex of A_{3r} whenever $j \geq r$, **(ii)** from each vertex of A_{11} to every vertex of A_{41}, **(iii)** from each vertex of A_{2j} to every vertex of $A_{3(j+1)}$, **(iv)** from each vertex of A_{2j} to every vertex of $A_{4(j+1)}$, **(v)** from each vertex of A_{i1} to every vertex of $A_{(i+1)1}$ for all even $i \geq 4$, **(vi)** from each vertex of B_{i2} to every vertex of $B_{(i+1)2}$ for all even $i \geq 4$, and **vii** from each vertex of B_{i2} to every vertex of $A_{(i+1)1}$ for all $i \geq 4$. Then the degrees of the vertices of $G(V_1, V_2, \ldots, V_{2m+3})$ are as follows.

$$d_{a_{11}} = |A_{31}| + |A_{41}| = d_1 + d_2, \text{ for all } a_{11} \in A_{11},$$

$$\text{for } 2 \leq j \leq n, \; d_{a_{1j}} = \sum_{r=1}^{j} |A_{3r}| = \sum_{r=1}^{j} d_r, \text{ for all } a_{1j} \in A_{1j},$$

$$\text{for } 1 \leq j \leq n-1, \; d_{a_{2j}} = |A_{3(j+1)}| + |A_{4(j+1)}| = d_{j+1} + d_1 + \cdots + d_j = \sum_{r=1}^{j+1} d_r, \text{ for all } a_{2j} \in A_{2j},$$

$$d_{a_{31}} = \sum_{r=1}^{n} |A_{1r}| = \sum_{r=1}^{n} d_r, \text{ for all } a_{31} \in A_{31},$$

$$\text{for } 2 \leq j \leq n, \; d_{a_{3j}} = \sum_{r=j}^{n} |A_{1r}| + |A_{2(j-1)}| = \sum_{r=j}^{n} d_r + d_1 + \cdots + d_{j-1} = \sum_{r=1}^{n} d_r, \text{ for all } a_{3j} \in A_{3j},$$

$$d_{a_{41}} = |A_{11}| + |A_{51}| = d_1 + d_2, \text{ for all } a_{41} \in A_{41},$$

$$\text{for } 2 \leq j \leq n, \; d_{a_{4j}} = |A_{2(j-1)}| = d_1 + d_2 + \cdots + d_{j-1} = \sum_{r=1}^{j-1} d_r, \text{ for all } a_{4j} \in A_{4j},$$

$$\text{for even } 4 \leq i \leq 2m+2, \; d_{b_{i2}} = |A_{(i+1)1}| + |B_{(i+1)2}| = d_2 + d_1 = d_1 + d_2, \text{ for all } b_{i2} \in B_{i2},$$

$$\text{for odd } 5 \leq i \leq 2m+1, \; d_{b_{i2}} = |B_{(i-1)2}| + |A_{(i+1)1}| = d_1 + d_2, \text{for all } b_{i2} \in B_{i2},$$

$$d_{b_{(2m+3)2}} = |B_{(2m+2)2}| = d_1, \text{ for all } b_{(2m+3)2} \in B_{(2m+3)2},$$

$$\text{for odd } 5 \leq i \leq 2m+3, \; d_{a_{i1}} = |A_{(i-1)2}| + |B_{(i-1)2}| = d_2 + d_1 = d_1 + d_2, \text{ for all } a_{i1} \in A_{i1},$$

for even $6 \leq i \leq 2m+2$, $d_{a_{i1}} = |A_{(i+1)1}|+|B_{(i-1)2}| = d_2+d_1 = d_1+d_2$,

for all $a_{i1} \in A_{i1}$.

Therefore, degree set of $G(V_1, V_2, \ldots, V_{2m+3})$ is $D = \{d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{n} d_i\}$.

Now, assume $k \geq 6$ is even, say $k = 2m + 4$ where $m \geq 1$. We add a new partition set V_{2m+4} to the above constructed $2m + 3$-partite graph $G(V_1, V_2, \ldots, V_{2m+3})$ with $|V_{2m+4}| = d_2$ and let there be an edge from each vertex of V_{2m+4} to every vertex of $B_{(2m+3)2}$ so that we obtain a $2m+4$-partite graph $G(V_1, V_2, \ldots, V_{2m+3}, V_{2m+4})$. It is clear that in $G(V_1, V_2, \ldots, V_{2m+3}, V_{2m+4})$ the degrees of all the vertices from the partite sets $V_1, V_2, \ldots, V_{2m+3}$ remain unchanged except the vertices in $B_{(2m+3)2}$ (of V_{2m+3}) whose degrees are increased to $d_1 + d_2$ and the degree of each vertex in V_{2m+4} is d_1. Therefore, the degree set of $G(V_1, V_2, \ldots, V_{2m+3}, V_{2m+4})$ is $D = \{d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{n} d_i\}$.

We note that in above construction, all the $k-partite$ graphs are connected except when $d_1 = 1$ and $k \geq 3$. □

Finally, we have the following result.

Theorem 7 *Every set of non-negative integers is a degree set of some* k*-partite graph.*

Proof. Let $d_1, d_2, \ldots, d_n$ be non-negative integers with $d_2, d_3, \ldots, d_n > 0$. We will show there is a k-partite graph $G(V_1, V_2, \ldots, V_k)$ with the degree set $D = \{d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{n} d_i\}$.

First assume that $d_1 = 0$. For $n = 1$, consider a null k-partite graph $G(V_1, V_2, \ldots, V_k)$ with $|V_i| = 1$ for all i, $1 \leq i \leq k$. Then for $1 \leq i \leq k$, $d_{v_i} = 0 = d_1$, for all $v_i \in V_i$. Therefore, degree set of $G(V_1, V_2, \ldots, V_k)$ is $D = \{d_1\}$.

Now let $n > 1$. Since $d_2, d_3, \ldots, d_n$ are positive integers, therefore by Theorem 6, there exists a k-partite graph $G(W_1, W_2, \ldots, W_k)$ with degree set $D_1 = \{d_2, \sum_{i=2}^{3} d_i, \ldots, \sum_{i=2}^{n} d_i\}$. We construct another k-partite graph $G(V_1, V_2, \ldots, V_k)$ as follows.

Let $V_1 = W_1 \cup \{v\}, V_2 = W_2, \ldots, V_k = W_k$. Then the degree of the vertex v is zero, that is, $d_v = 0 = d_1$ and the degrees of all the vertices of the partition set $W_1, W_2, \ldots, W_k$ remain unchanged in $G(V_1, V_2, \ldots, V_k)$. Therefore $G(V_1, V_2, \ldots, V_k)$ is a k-partite graph with degree set $D = \{d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{n} d_i\}$.

Now assume that $d_1 > 0$. Then $d_1, d_2, \ldots, d_n$ are the positive integers and therefore by Theorem 6, there exists a k-partite graph $G(V_1, V_2, \ldots, V_k)$ with degree set $D = \{d_1, \sum_{i=1}^{2} d_i, \ldots, \sum_{i=1}^{n} d_i\}$. □

References

[1] T. S. Ahuja, A. Tripathi, On the order of a graph with a given degree set. *J. Comb. Math. Comb. Comput.*, **57** (2006) 157–162. ⇒252

[2] G. Chartrand, R. J. Gould, S. F. Kapoor, Graphs with prescribed degree sets and girth, *Periodica Math. Hung.*, **12,** 4 (1981) 261–266. ⇒252

[3] A. A. Chernyak, Minimal graphs with a given degree set and girth (Russian), *Vestsi Akad. Navuk BSSR Ser. Fiz.-Mat. Navuk,* **1988,** 2 21–25, 123. ⇒252

[4] F. Harary, E. Harzheim, The degree sets of connected infinite graphs. *Fund. Math.*, **118,** 3 (1983) 233–236. ⇒252

[5] A. Iványi, J. Elek, Degree sets of tournaments, *Studia Univ. Babeş-Bolyai, Informatica*, **59** (2014) 150–164. ⇒252

[6] F. Harary, *Graph Theory*, Reading, MA, Addison-Wesley (1969). ⇒251

[7] A. Iványi, S. Pirzada and F. A. Dar, Tripartite graphs with given degree set, *Acta Univ. Sap. Informatica*, **7, 1** (2015) 72–106. ⇒252

[8] S. F. Kapoor, A. D. Polimeni, C. E. Wall, Degree sets for graphs, *Fund. Math.*, **95,** 3 (1977) 189–194. ⇒251

[9] S. Koukichi and H. Katsuhiro, Some remarks on degree sets for graphs, *Rep. Fac. Sci. Kogoshima Univ.*, **32** (1999) 9–14. ⇒252

[10] Y. Manoussakis, H. P. Patil, V. Shankar, Further results on degree sets for graphs, *AKCE J. Graphs Combin.*, **1,** 2 (2004) 77–82. ⇒252

[11] Y. Manoussakis, H. P. Patil, Bipartite graphs and their degree sets, *Electron. Notes on Discrete Math.,* (Proceedings of the R. C. Bose Centenary Symposium on Discrete Mathematics and Applications,) **15** (2003) 125–125. ⇒252

[12] Y. Manoussakis, H. P. Patil, On degree sets and the minimum orders in bipartite graphs, *Discussiones Math. Graph Theory,* **34,** 2 (2014) 383–390. ⇒252

[13] C. M. Mynhardt, Degree sets of degree uniform graphs, *Graphs Comb.*, **1** (1985) 183–190. ⇒252

[14] S. Osawa, Y. Sabata, Degree sequences related to degree sets, *Kokyuroki*, **1744** (2011) 151–158. ⇒252

[15] S. Pirzada and Y. Jian Hua, Degree sequences in graphs, *J. Math. Study,* **39,1** (2006) 25–31. ⇒252

[16] S. Pirzada, T. A. Naikoo and F. A. Dar, Degree sets in bipartite and 3-partite graphs, *Oriental J. Math. Sci.*, **1,1** (2007) 39–45. ⇒252

[17] S. Pirzada, F. A. Dar, Signed degree sets in signed tripartite graphs, *Matematicki Vesnik*, **59** (3) (2007) 121–124. ⇒252

[18] S. Pirzada, F. A. Dar, Signed degree sequences in signed tripartite graphs, *J. Korean Soc. Ind, Appl. Math.*, **11,** 2 (2007) 9–14. ⇒252

[19] S. Pirzada, Merajuddin, T. A. Naikoo, Score sets in oriented 3-partite graphs, *Analysis Theory Appl.*, **4** (2007) 363–374. ⇒252
[20] S. Pirzada, T. A. Naikoo, Score sets in oriented k-partite graphs, *AKCE J. Graphs Combin.*, **3,** 2 (2006) 135–145. ⇒252
[21] S. Pirzada, T. A. Naikoo, Score sets in k-partite tournaments, *J. Appl. Math. Comp.*, **22,** 1–2 (2006) 237–245. ⇒252
[22] S. Pirzada, T. A. Naikoo, Score sets in oriented graphs, *Appl. Anal. Discrete Math.*, **2** 1 (2008) 107–113. ⇒252
[23] S. Pirzada, T. A. Naikoo, T. A. Chishti, Score sets in oriented bipartite graphs, *Novi Sad J. Math*, **36,** 1 (2006) 35–45. ⇒252
[24] S. Pirzada, T. A. Naikoo, F. A. Dar, Signed degree sets in signed graphs, *Czechoslovak Math. J.*, **57,** 3 (2007) 843–848. ⇒252
[25] S. Pirzada, *An Introduction to Graph Theory*, Universities Press, Hyderabad, India, 2012. ⇒251
[26] S. Pirzada, T. A. Naikoo, F. A. Dar, A note on signed degree sets in signed bipartite graphs, *Appl. Anal. Discrete Math.*, **2,** 1 (2008) 114–117. ⇒252
[27] K. B. Reid. Score sets for tournaments, *Congressus Numer.*, **21** (1978) 607–618. ⇒252
[28] T. A. Sipka, The orders of graphs with prescribed degree sets, *J. Graph Theory*, **4,** 3 (1980) 301–307. ⇒
[29] A. Tripathi, S. Vijay, On the least size of a graph with a given degree set, *Discrete Appl. Math.*, **154** (2006) 2530–2536. ⇒252
[30] A. Tripathi, S. Vijay, A short proof of a theorem on degree sets of graphs, *Discrete Appl. Math.*, **155** (2007) 670–671. ⇒252
[31] L. Volkmann, Some remarks on degree sets of multigraphs, *J. Combin. Math. Combin. Comput.*, **77** (2011) 45–49. ⇒252

Received: October 19, 2020 • Revised: November 6, 2020

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 260–282

DOI: 10.2478/ausi-2020-0016

Comparing epidemiological models with the help of visualization dashboards

Csaba FARKAS
Sapientia Hungarian University of Transylvania, Cluj-Napoca, Romania
Dept. of Mathematics and Informatics, Târgu Mureș
email: farkascs@ms.sapientia.ro

David ICLANZAN
Sapientia Hungarian University of Transylvania, Cluj-Napoca, Romania
Dept. of Mathematics and Informatics, Târgu Mureș
email: iclanzan@ms.sapientia.ro

Boróka OLTEAN-PÉTER
Sapientia Hungarian University of Transylvania, Cluj-Napoca, Romania
Dept. of Mathematics and Informatics, Târgu Mureș
email: boroka.oltean@ms.sapientia.ro

Géza VEKOV
Babeș-Bolyai University, Cluj-Napoca, Romania
Faculty of Mathematics and Computer Science
email: geza.vekov@cs.ubbcluj.ro

Abstract. In 2020, due to the COVID − 19 pandemic, various epidemiological models appeared in major studies [16, 22, 21], which differ in terms of complexity, type, etc. In accordance with the hypothesis, a complex model is more accurate and gives more reliable results than a simpler one because it takes into consideration more parameters.

In this paper we study three different epidemiological models: a SIR, a SEIR and a SEIR − type model. Our aim is to set up differential equation models, which rely on similar parameters, however, the systems of equation and number of parameters deviate from each other. A visualization dashboard is implemented through this study, and thus, we are

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 68R15
Key words and phrases: Visualization dashboard, epidemiological model, optimization algorithm

able not only to study the models but also to make users understand the differences between the complexity of epidemiological models, and ultimately, to share a more specific overview about these that are defined by differential equations [24].

In order to validate our results, we make a comparison between the three models and the empirical data from Northern Italy and Wuhan, based on the infectious cases of COVID-19. To validate our results, we calculate the values of the parameters using the Least Square optimization algorithm.

1 Introduction

The COVID-19 pandemic has been responsible for over 24 million cases worldwide according to WHO reports. Not only has it been causing one of the biggest global health crises and the greatest challenge we have faced since World War II., but will also turn global economic growth "sharply negative" this year, based on a forecast by BBC.

Mathematical models have been employed to inform media, authorities and researchers from different areas about the effect of the virus from different perspectives. In this study, we are going to discuss three different epidemiological models, which are visualized by a reactive tool. As [24] says epidemiological models are a key tool to guide public health measures. Without having experiences in crises such as this one, modelling and simulations require assumptions and different test scenarios. Therefore, visualizing three models with different complexity and parameter numbers as well as, comparing them are crucial to be well prepared for future events and to understand the roles of different factors in this pandemic.

Our first goal is to minimize the gap between medical reports, statistics, interoperability and public health information system [1, 6, 14, 3]. However, [3] study also highlighted that the pressure from this gap has always been particularly acute for the surveillance and management of infectious diseases with pandemic or bioterrorism potential, we reckon that nowadays this issue is one of the most pressing global problems.

We think that this tool is useful not only from the perspective of the public health information system, but also from that of simulations. We use one of the newest Javascript libraries, Svelte, to create a reactive tool. Secondary, our aim is to validate our results with help of the Least Square optimization algorithm. Truncated Newton method is suitable for solving large nonlinear optimization problems [18].

The novelty of this research is that even though there exist several tools which visualize the results of mathematics models, which are presented in 3 section, we have no knowledge of tools which compare the models and validate the results with empirical data.

2 Short overview of epidemiological models and visualization dashboards

Study [3] in 2014 made a systematic review about visualization and analytics tools used for infectious disease epidemiology. In this study 247 articles were screened, and 88 articles were included in the review process. These articles primarily included descriptive reports, qualitative (e.g. interviews, focus groups) and usability studies. Although, public health workspace is extremely diverse [19] and the need for rapid access to information to support critical decisions in public health is inevitable, the public health information sources are unstandardized [12]. As a result, the visualization and analytics tools are various, especially in the case of $\mathsf{COVID-19}$ disease.

There are many studies which concentrate on the model proposal, such as [16, 22, 21], but there is a lack of visualization of these models. To our knowledge, the number of dashboards which project and simulate the population's exposure is very low, there is only one tool[1] which is a reactive data visualization based on an epidemiological model. However, the above mentioned visualization dashboard uses a basic SEIR model which cannot provide sufficiently accurate results from our point of view. We reckon that the demand for these tools which predict cases would be much higher, especially because there are a lot of different other software which visualize the empirical, measured cases in different countries and regions, such as Covidvisualizer[2], Gisanddata[3] which was presented in [9] study or Wolframcloud Visualization Dashboard[4]. Other web pages which try to inform individuals and to minimize the $\mathsf{COVID-19}$ damages are rife, for instance, the $'\mathtt{plugandplaydiagnostics}'$ software, presented in [25] study, which helps to prevent future epidemics or the COVID-19 Search Intensity Monitoring tool[5]. We also need to mention Epirisk dashboard[6]

[1] https://gabgoh.github.io/COVID/index.html

[2] https://www.covidvisualizer.com/

[3] https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6

[4] https://www.wolframcloud.com/obj/examples/COVID19Dashboard

[5] https://covid19map.uptodate.com/

[6] https://epirisk.net/

which is a computational platform estimating of the probability of exporting infected individuals from sites affected by a disease outbreak to other areas in the world through the airline transportation network and the daily commuting patterns.

As it can be seen, the above listed tools do not concentrate on mathematical models, and they do not contain a model comparison, even though different mathematical models bring various results [13, 1].

There is no accepted consensus regarding the modelling approach is considered to be the most accurate. Cooper et al. [8] use an susceptible-infected-removed (SIR) model and Wangping et al. [26] calculates with an extended SIR model, where transmission can be changed through many interventions, such as personal protective measures, community-level isolation, and city blockade. A lot of studies use susceptible-infected-exposed-removed (SEIR) models and their extended versions [29, 16, 22, 21, 23, 28, 17, 11].

In the above mentioned studies not only are the epidemiological models are different, but also the results and the parameter values are various.

3 Mathematical background for epidemiological models and optimization algorithm

Not only do we confirm that it is not possible to decide which model brings the most accurate results, but we also reckon that users' knowledge regarding the COVID − 19 pandemic and mathematical modelling is very wide-ranging. Even though we would accept the hypothesis according to which a more complex model is more accurate than a simpler one, we are faced with challenges regarding the user environment: users need to set up many parameters, which is difficult to comprehend. As one of our aims is to minimize the gap between general users and the public health information system, we reckon that by choosing a too complex model we would lose a significant part of possible users who we want to address.

For that, we have implemented three different models, with different complexity and different parameter numbers, but they use same parameters, which have the same meaning from a medical and environmental point of view.

3.1 SEIR-type model

The most complex model is a general SEIR-type model, which incorporates biological, social, environmental processes, such as governmental actions, (e.g.

school closing), weather conditions (temperature, humidity), and behavioral responses. Taking into account the above factors, we propose for the visualization the following SEIR-type model (see for more details see [10]):

$$\begin{cases} S' = -\left(\beta c\left(t\right) + c\left(t\right) q\left(1-\beta\right)\right) S\left(I + \theta A\right)/N + \lambda S_q \\ E' = \beta c\left(t\right)\left(1-q\right) S\left(I + \theta A\right)/N - \sigma E \\ I' = \sigma\rho E - \left(\delta_I + \alpha + \gamma_I\right) I \\ A' = \sigma\left(1-\rho\right) E - \gamma_A A \\ S_q' = \left(1-\beta\right) c\left(t\right) qS\left(I + \theta A\right)/N - \lambda S \\ E_q' = \beta c\left(t\right) qS\left(I + \theta A\right)/N - \delta_q E_q \\ H' = \delta_I I + \delta_q E_q - \left(\alpha + \gamma_H\right) H \\ R' = \left(\delta_I + \alpha + \gamma_I\right) I + \gamma_A A + \gamma_H H - \gamma_R R \end{cases}$$

where the functions S, E, I, A, S_q, E_q, H and R denote the proportion of the population into eight groups: susceptible ($S(t)$), exposed ($E(t)$), infectious ($I(t)$), pre-symptomatic ($A(t)$), hospitalized ($H(t)$), recovered ($R(t)$), quarantined susceptible ($S_q(t)$) and isolated exposed ($E_q(t)$) groups of population, see Figure 1 for infection dynamics:

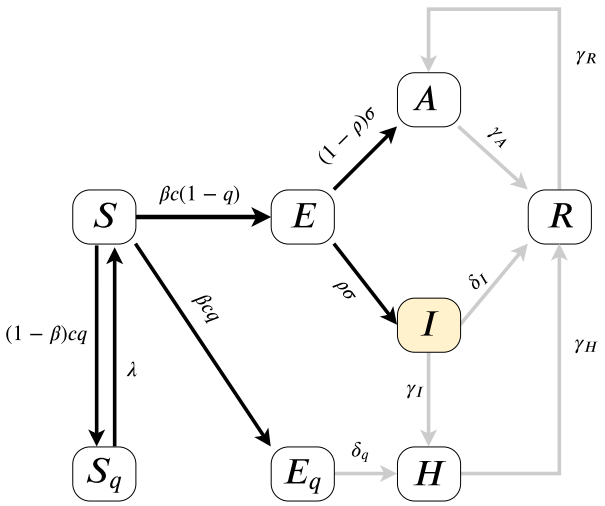


Figure 1: Model diagram for infection dynamics (SEIR-type model)

3.2 SEIR model

The $\mathsf{SEIR}-\mathsf{type}$ model's simplified version is the SEIR model which uses significantly less parameters. This model was developed from the $\mathsf{SEIR}-\mathsf{type}$ model:

$$\begin{cases} S' = -\left(\beta c\left(t\right) + c\left(t\right) q\left(1-\beta\right)\right) SI/N \\ E' = \left(\beta c\left(t\right) + c\left(t\right) q\left(1-\beta\right)\right) SI/N - \sigma\rho E \\ I' = \sigma\rho E - \left(\delta_I + \alpha + \gamma_I\right) I \\ R' = \left(\delta_I + \alpha + \gamma_I\right) I - \gamma_R R \end{cases}$$

where the functions denote the proportion of the population into four groups: susceptible ($S(t)$), exposed ($E(t)$), infectious ($I(t)$) and recovered ($R(t)$), see Figure 2.

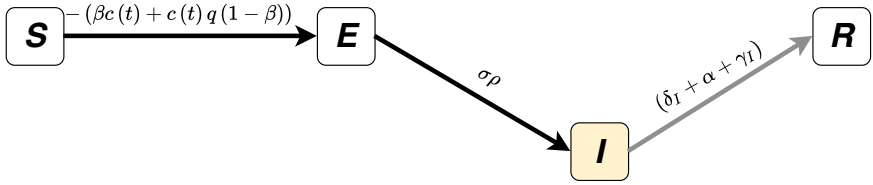


Figure 2: Model diagram for infection dynamics (SEIR model)

3.3 SIR model

By following the previous logic, we get the SIR model, if we exclude from the SEIR model the exposed population:

$$\begin{cases} S' = -\beta SI/N \\ I' = \beta SI/N - \gamma_I I \\ R' = \gamma_I I - \gamma_R R \end{cases}$$

where the population is split into 3 groups: susceptible ($S(t)$), infected ($I(t)$) and recovered ($R(t)$) proportions.

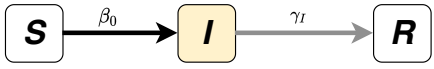


Figure 3: Model diagram for infection dynamics (SIR model)

3.4 The meaning of the parameters

The above mentioned models use same parameters in order to be comparable. These parameters are presented briefly, in case any additional information is needed, read [10] study.

The parameters σ and λ describe the transition rate of exposed individuals to the infected class and the rate at which the quarantined uninfected were released into the wider community, while the parameter ρ represents the probability of having symptoms among infected individuals. The parameters δ_I and δ_q denote the transition rate of symptomatic infected and quarantined exposed to the quarantined infected class. The γ_I, γ_A and γ_H represent the recovery rate of symptomatic, asymptomatic and quarantined infected individuals, and finally γ_R is the rate at which immunity is lost and recovered individuals move to the pre-symptomatic class (according to a recent NHK-World Japan report[7] and [15]). The parameter θ represents the relative transmission probability of pre-symptomatic individuals to infected individuals. Finally we assume that natural birth and natural death rates are equal.

The motivation of such a choice is the following (see also [5, 28]): individuals move from quarantined cases with $1-q$ proportion to S_q and with q proportion to E_q. If the transmission probability is β and the contact rate is c, then, the infected quarantined individuals move to E_q at rate of βcq and uninfected quarantined individuals move to S_q at $(1-\beta)\,cq$ rate. In case of not quarantined infected, they are going to move to E at a rate of $\beta c\,(1-q)$. When an epidemiological outbreak occurs, many preemptive actions can be taken to mitigate the spreading. Once people become informed, they can change their behavior, working from home, practicing social distancing, and take actions like washing hands more often, wearing protective clothing, disinfecting etc., all of them contributing to the prevention of the spread. The media interacts with the susceptible population, it starts influencing them to take appropriate measures to minimize the chances of getting infected. This media influence is initially low and increases as the infection increases. This observation suggests the following contact rate function:

$$c(t) = c_a + \frac{3(c_0 - c_a)}{1 + 2b^{-t}},\ t \geq 0, \tag{1}$$

where $b < 1$ and c_0 denotes the initial contact rate, while c_a denotes the minimum contact rate under the current control strategies.

[7] https://www3.nhk.or.jp/nhkworld/en/news/20200315_13/

Obviously, not every parameter appears in all of three models. This description can be seen complete only in case of the $\mathtt{SEIR-type}$ model.

3.5 Least square optimization algorithm

Optimization algorithms can be classified into two major categories: line search methods and trust region methods, as Ya-xinag Yuan confirms [30]. The trust region approach associates with approximation, assuming that we have a current guess and the model can be constructed near that point. This algorithm is proposed to solve large-scale bound constrained minimization problems. It solves trust-region subproblems iteratively, augmenting with trust-region shape determined by the distance from the bounds and the direction of the gradient and by a special diagonal quadratic term as it is formulated in Python documentation[8]. The aim of this improvement is to iterate through the whole space of variables and to avoid hitting directly the bounds.

The mathematical approach of Trust Region Reflective Algorithm was formulated based on ["A subspace, interior and conjugate gradient method for large-scale bound-constrained minimization problems"] paper by Nikolay Mayorov through article[9]. It is defined the following bound-constrained minimization problem

$$\min f(x), \; x \in \mathcal{F} = \{x : l \leq x \leq u\},$$

where $l \in \{\mathbb{R} \cup \{-\infty\}\}^n$ and $u \in \{\mathbb{R} \cup \{\infty\}\}^n$ and also the f function is a smooth function [2]. The $g(x) = \nabla f(x)$ and $H(x) = \nabla^2 f(x)$. Defining the following vector, we get

$$v(x)_i = \begin{cases} u_i - x_i & g_i < 0, u_i < \infty, \\ x_i - l_i & g_i > 0, l_i > -\infty, \\ 1 & otherwise. \end{cases}$$

Defining a matrix $D(x) = \operatorname{diag}\left(v(x)^{\frac{1}{2}}\right)$, we can formulate the first order optimality as followed:

$$D^2(x)\, g(x) = 0.$$

If $v(x)_i = 0$ the Jacobian of the left hand does not exists, so we can consider that

$$v(x)_i \neq 0,$$

[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html

[9] https://nmayorov.wordpress.com/2015/06/19/trust-region-reflective-algorithm/

for all i. This happens if x is not on the bound. In this case the Newton step for this system is

$$\left(D^2H + \operatorname{diag}(g(x))\, J_v\right) p = -D^2 g(x),$$

where J_v is the $v(x)$ vector's diagonal Jacobian matrix. Now the corresponding trust-region problem can be formulated:

$$\min_p m(p) = \frac{1}{2} p^T B p + g^T p, \text{ s.t. } \|D^{-1} p\| \leq \Delta,$$

where $B = H + D^{-1} C D^{-1}$ and $C = \operatorname{diag}(g)\, J_v$. As it is formulated in [7] study reflective algorithms are used to maintain feasibility by a piecewise linear function, which helps to avoid bounds. The following implementation (Algorithm 3.5), which was developed by Nikolay Mayorov is mainly the same as `scipy.optimize.least_squares`:

```
import numpy as np

def reflective_transformation(y,l,u):
  if l is None:
    l=np.full_like(y, -np.inf)
  if u is None:
    u=np.full_like(y,np.inf)
  l_fin=np.isfinite(l)
  u_fin=np.isfinite(u)
  x=y.copy()
  m=l_fin & ~u_fin
  x[m] = np.maximum(y[m], 2*l[m]-y[m])
  m=~l_fin & u_fin
  x[m] = np.maximum(y[m], 2*u[m]-y[m])

  m=l_fin & u_fin
  d=u-l
  t=np.remainder(y[m]-l[m], 2*d[m])
  x[m]=l[m]+np.minimum(t,2*d[m]-t)

  return x

```

Algorithm 1: Reflective transformation algorithm

4 The overview of the visualization dashboard

Wekler at al. [27] defines dashboards as: "a visual display of data used to monitor conditions and/or facilitate understanding", our aims are to monitor conditions and facilitate the understanding of the COVID − 19 pandemic and similar infectious diseases. The presented tool is a functional genre of dashboard which, as Sarikaya et al. [20] defines, means an interactive display that enables real-time monitoring of dynamically updating data.

The dashboard is a reactive, interactive tool, which is going to be presented trough the following types of interactivity [20]:

1. Construction and Composition,
2. Multipage,
3. Interactive Interface.

The role of this classification is that reactivity and interactivity can take place at a number of different places in the dashboard lifecycle.

Reactive programming means a declarative programming paradigm where variables are updated automatically whenever other values change, while the re-execution of the statements is not necessary. Reactivity is programming with asynchronous data streams[10]. The benefit of the reactivity is that the dashboard becomes highly interactive with a multitude of UI events related to data events. This benefit evolves real-time monitoring: modifying a single value, such as a parameter, can automatically trigger other contents. Using Svelte Javascript library reactivity is realized by techniques such as virtual DOM which runs at built time, converting the components into a highly efficient imperative code that updates the DOM[11].

Due to the reactivity of the tool, the differential equation solver needs to compute the model very efficiently in order to serve a reasonable re-computation time based on the current parameters. From data visualization's perspective one of the most pressing issues was to find an implementation which is accurate enough and does not cause lagging. For that, we have chosen the classic Runge-Kutta IV (RK4) method. There are a lot of Javascript libraries which implement Runge-Kutta methods, such as Runge-Kutta 4 library[12], or Cash-Karp implementation[13] which is an adaptive Runge-Kutta method and

[10] https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
[11] https://svelte.dev/blog/svelte-3-rethinking-reactivity
[12] https://www.npmjs.com/package/runge-kutta-4
[13] https://www.npmjs.com/package/ode45-cash-karp

is presented in [4] study. However, we chose almost the simplest library Runge-Kutta library[14], because this one can set up properly the interval and the step size. Our issue with adaptive Runge-Kutta method is that it gave significantly less efficiency than our final choice.

4.1 Construction and composition

These dashboards provide flexibility for the viewer to customize the placement of views, modify the visual representations inside those views, or select the particular dimensions and measures to visualize [20].

As we presented in Section 3, we visualize values of three different mathematical compartment models. As a result, users can modify the visual representations of the models. Users can choose the following three major representations:

- SIR model representation
- SEIR model representation
- $\mathsf{SEIR-type}$ model representation
- Overall trend.

As we discussed in section 1, different parameter lists help the user to find the model which is consistent with their knowledge. For instance, if the most complex model, 17 parameters and 8 initial values of functions appear (Figure 5). In case of selecting SEIR representation, the number of parameters is 11, 4 initial values are needed. For the least complex model only 5 parameters, 3 initial values are needed. These options modify drastically the user experience, because as long as the most complex representation gives a scientific view of the topic, the simplest representation can be understood by general users.

[14] `https://www.npmjs.com/package/runge-kutta`

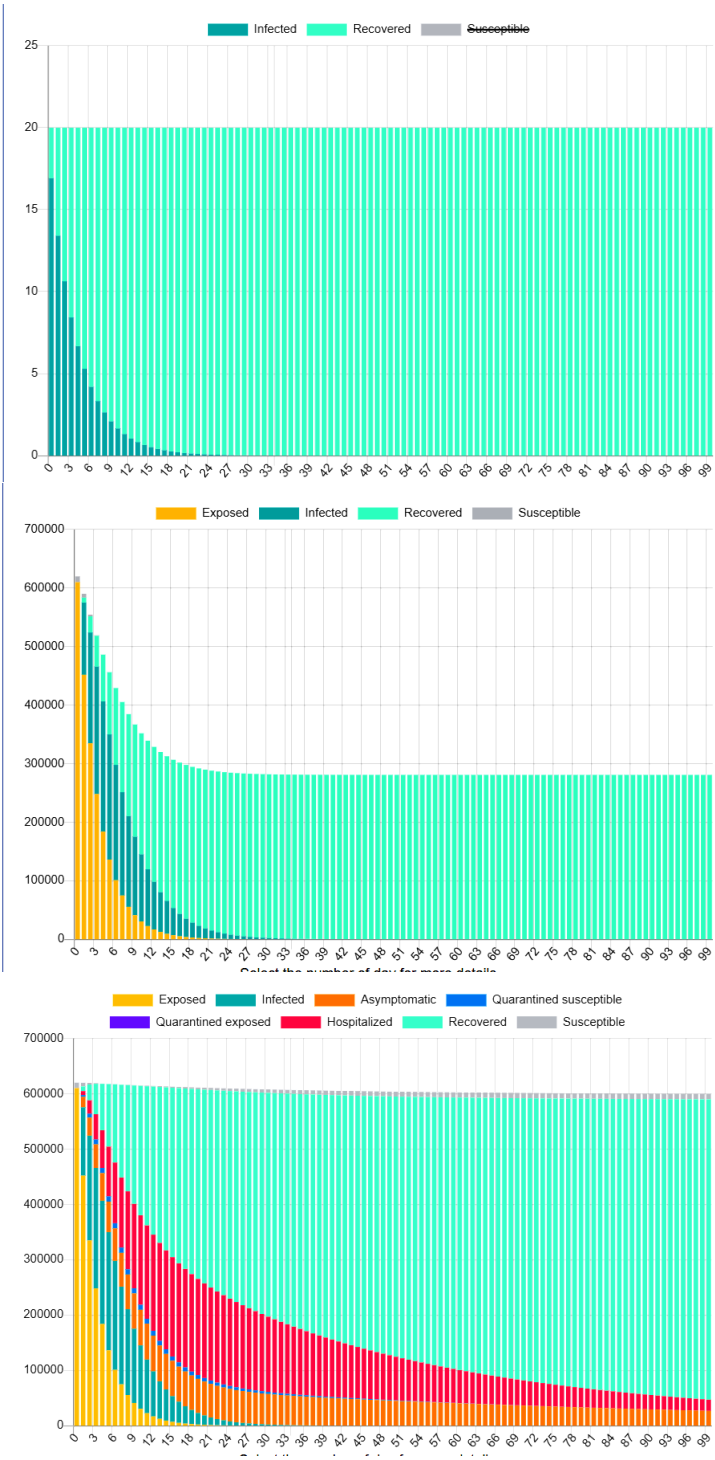


Figure 4: SIR model, SEIR model, SEIR-type models' representation, and overall trend

Definition:	Range of value:	Selected value:	Included in model:
c_0		41.85	☑
c_a		1.287039726	☑
q_1		1.484170130317585e-7	☑
β_0		2.1031894440837713e-12	☑
ϵ		0.88	☑
σ		0.3333	☑
λ		0.071428	☑
δ_I		0.121	☑
δ_q		0.03363	☑
γ_I		0.1351146	☑
γ_A		0.011795	☑
γ_H		0.028635	☑
γ_R		0.000001714578192	☑
θ		0.00016357	☑
α_b		0.0012209	☑

Definition:	Range of value:	Selected value:
S		52740000
E		19.22
I		16.93
A		17.99
S_q		15.54
E_q		1.72
H		46.01
R		3.07

Figure 5: SEIR-plus model's parameter list and initial value condition

4.2 Multipage

Usually dashboards are monopages, but some of them support tabbed layouts. These dashboards allow viewers to switch between pages, which may have visualizations that relate to a different component of decision-making or help to provide the necessary context [20].

Regarding the presented dashboard, multipage and constraction-composition relate closely. The user can navigate between the three model representations and the comparison of three models. However, the dashboard has multi-level structure, meaning that a single page, such as Overall trend page, is divided to more than one components. The user can compare the tree models based on each basic proportion of population. As it can seen on 6, the comparison of models is visualized based on Recovered (R(t)) cases.

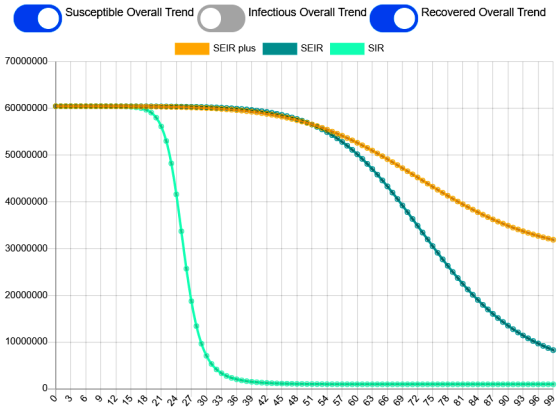


Figure 6: The user can choose which function to be visualized: S(t), I(t) or R(t)). Here, the S(t) function is displayed.

4.3 Interactive interface

Obviously, drop-down menus (Figure 7), slicers (Figure 5) appear on the dashboard which improve user-experience. But, due to the reactivity of the tool, the differential equation solver was needed the model to be computed very efficiently in order to serve a reasonable re-computation time based on the current parameters. From data visualization one of the most pressing issues was to find an implementation which is accurate enough and does not cause lagging. For that, we have chosen the classic Runge-Kutta IV (RK4) method. There are a lot of Javascript libraries which implement Runge-Kutta meth-

ods, such as Runge-Kutta 4 library[15], or Cash-Karp implementation[16] which is an adaptive Runge-Kutta method and is presented in [4] study. However, we chose one of the simplest libraries Runga Kutta library[17], because the adaptive Runge-Kutta method gave significantly less efficiency than our final choice.

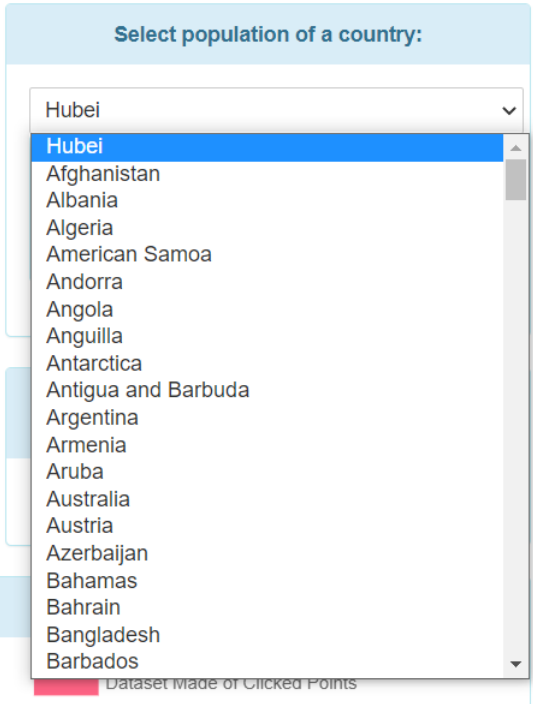


Figure 7: Drop-down list where the user can choose the country or area data the data of which they want to be set up.

5 Results

We reckon that we accomplished our aims formulated in Section 1. Our primarily one with visualization dashboard was to reduce the gap between general user's knowledge regarding mathematical modelling and public health information system which helps to make correct decisions. For that, we implemented different informative labelling and description (as shown Figure 8) regarding

[15] `https://www.npmjs.com/package/runge-kutta-4`
[16] `https://www.npmjs.com/package/ode45-cash-karp`
[17] https://www.npmjs.com/package/runge-kutta

the models, the parameters and initial conditions, which change dynamically after modifying of the models. The fact, that parameters can be modified only in predefined intervals serves the same purpose. We would like to suggest values which reflect the reality. Because of it, when the dashboard is loaded for the first time, the parameters are predefined based on Least-Square optimization algorithm and empirical data of Northern Italy which can be seen in SEIR-type column of Table 2.

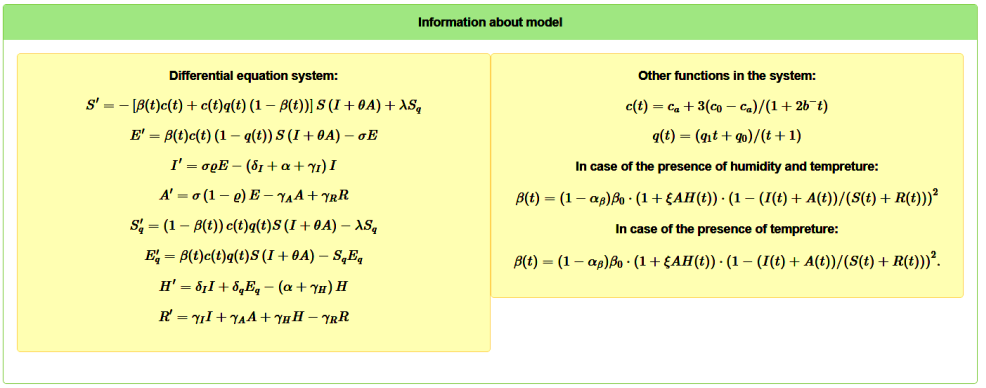


Figure 8: Model description and informative labelling

However, based on our results, we reckon that choosing the proper model and parameters' values presumes prior knowledge from users. For that, we think all visualization dashboards based on mathematical models mostly targeted users who are mathematicians investigating the discussed models, and these tools might be providing them with important insights.

5.1 Visualization dashboard

The functionalities of the visualization dashboard's presented in this study accomplish the aims formulated in 1 Section. The following functionalities serve the demands of general lay users but they also help the work of practised ones by the following components.

As it is presented on Figure 9, the web page has different components. When the user enters the web page, they see a predefined parameter settings, ebpagehich we calculated based on the cases of infection as measured in the region of Northern Italy and the parameters were obtained with the help of the

Least − Square algorithm. The user can start to change parameters manually or can import new parameter setup via a JSON file. It is given the opportunity to set up some parameters manually after importing the parameters, and the other way around, they can import some parameters after setting up the parameter manually as well. After setting up the parameters, they can be exported, resulting a JSON file with the current parameter setup, or can directly display data visualization. If the user changes a parameter, the differential equation solver automatically recalculates the result. After data visualization, the charts can be exported in the PDF format as well.

Due to reactivity, these steps are not detached strictly, they can be inverted, and other steps will respond to these changes directly and instantly.

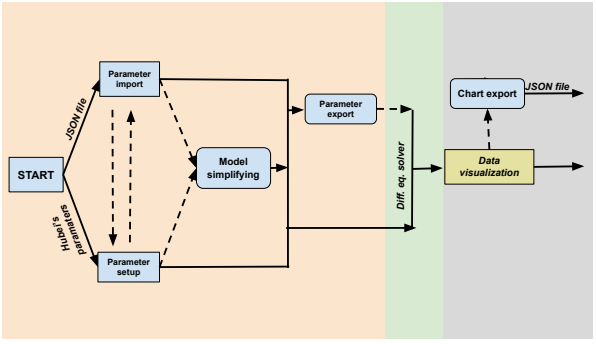


Figure 9: The stucture of visualization dashboard, the yellow segment represents the parameter setup part, the green component represents the differential equation solver and the grey one represents data visualization.

5.2 Models comparison

As we expected, the tree models usually bring slightly different results (Figures 10, 11). These figures present results with different parameter setup and initial value conditions.

We can observe that the SIR epidemiological model presented in Section 3 is not complex enough to model COVID − 19 pandemic.

The comparison of models can be performed from various perspectives. We have studied the models based on the comparison of empirical data. For that, we have used Least-Square optimization algorithm to estimate the parameters for models. To see more about this optimization algorithm, see [10].

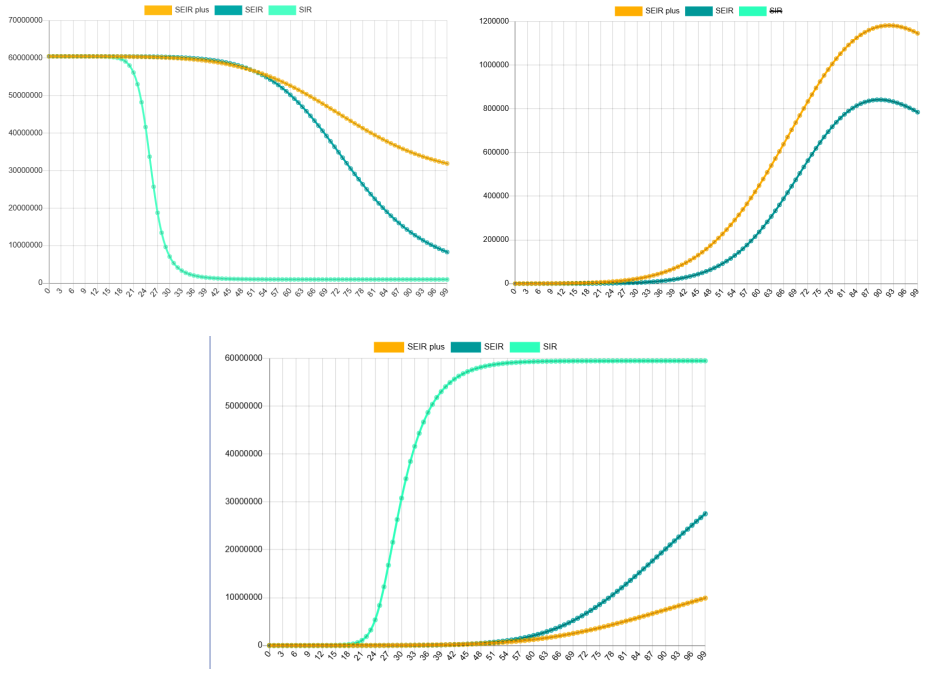
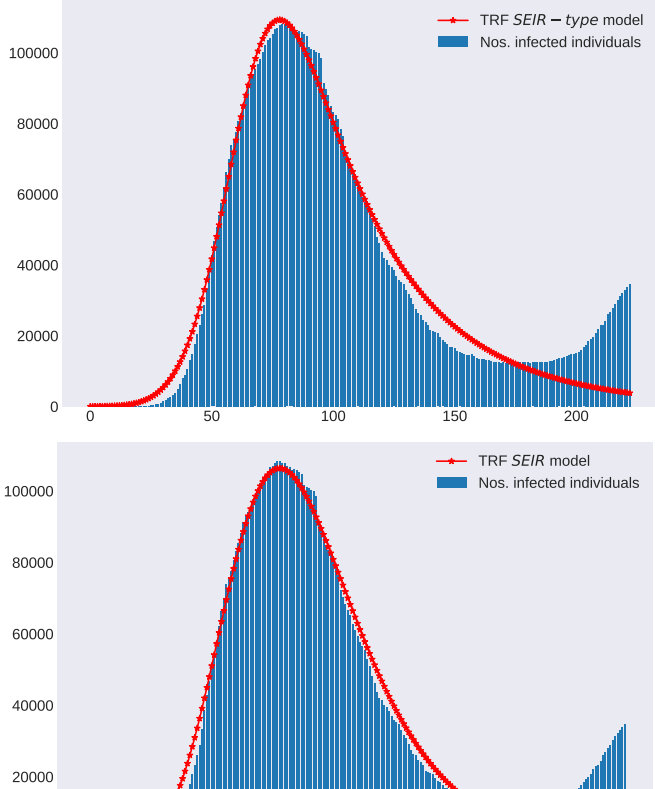


Figure 10: Model comparison (first fig. S(t), second fig. I(t), third fig. R(t))



As long as the SEIR−type and SEIR model perform quite similarly, the SIR model fails from different perspectives. As shown in Table 1, the cumulative errors between measured and estimated values are similar in case of the SEIR−type model and the SEIR model. The algorithm was run for multiple dataset, and the errors move approximately in the same interval in case of SEIR and SEIR − type models. The algorithm could not fit to measured data in case of the SIR model, as the error values suggest.

Empirical data	SIR	SEIR	SEIR-type
North-Italy	160968187	1209142	1113692
Hubei	149720737	103984	127389
Germany	215503664	1446547	1336592

Table 1: Models comparison based on error bounds

As long as the error bounds of the SEIR and the SEIR-type models are approximately similar, the parameter lists based on the estimation are various. And even tough, the estimations seem to be very close to each other, as shown Figures 11, these results were gotten from strongly dissimilar parameter lists (see Table 2).

6 Discussion

The phenomenon, which was presented at Subsection 5.2 can have multiple reasons. We need to emphasize that running an optimization algorithm does not necessarily mean that parameters from Table 2 are the most optimal parameters for the presented models.

These approaches can be further aims, but our primarily goal with this study was not parameter optimization. In accordance with the previously formulated hypothesis, we cannot decide unequivocally that a more complex model is more accurate than a simpler one. This study highlights the fact that neither the complexity of the model, nor parameter number are crucial in mathematical models. If we do not interpret the values of parameters from medical and epidemiological point of view, almost every model can fit properly to empirical values.

We also need to emphasize that these models divide the population in different groups. For instance, as long as the SEIR-type model defines 8 different proportions, associating exposed, isolated exposed, etc., the SIR model does not even take into account the exposed proportion of the population. This dif-

ference obviously comes up in the values and in the visualization as well. Based on this study we confirm that mathematical epidemiological models provide a certain way to understand infectious diseases and pandemics, but without a medical perspective it is not possible to conclude clear conclusions regarding the future-events.

Based on the above formulated affirmations, we think that this visualization dashboard is the most useful, when the user knows some parameter values and wants to check different test scenarios based on their knowledge, and not inversely.

Name of variables	SEIR-type	SEIR
S	60461744	60461744
E	194	195.656
I	10	10
A	96.69	–
S_q	151.86	–
E_q	158.28	–
H	2	–
R	0	0
c_0	33.74	33.34
c_a	10.95	2.49
q_1	0.29	0.20
β_0	0.11	0.38
ϵ	0.46	0.57
σ	0.17	0.17
λ	0.071	0.071
δ_I	0.0028	0.999
δ_q	0.16	–
γ_I	0.219	0.333
γ_R	0.219	0.023
γ_A	0.197	–
γ_H	0.326	–
θ	0.502	–
α	0.838	0.999

Table 2: Optimal parameter lists

Acknowledgment

Csaba Farkas has been supported by the Sapientia Foundation — Institute for Scientific Research, Romania, Project No. 17/11.06.2019. Boróka Oltean-Péter has been supported by the Sapientia Hungariae Foundation – Collegium Talentum project and by Accenture Student Research Scholarship.

References

[1] A. Abta, A. Kaddar, T. Hamad, Global stability for delay sir and seir epidemic models with saturated incidence rates, *Electronic Journal of Differential Equations* **2012,** 23 (2012) 1–13. ⇒261, 263

[2] M. Branch, Th. Coleman, Y. Li, A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21, 1 (1999) 1–13. doi:10.1137/S1064827595289108 ⇒ 267

[3] L. N. Carroll, A. P. Au, L. T. Detwiler, T. Ch. Fu, I. S. Painter, N. F. Abernethy, Visualization and analytics tools for infectious disease epidemiology: A systematic review, *Journal of Biomedical Informatics*, **51** (2014) 287–298. doi:10.1016/j.jbi.2014.04.006 ⇒261, 262

[4] J. Cash, A. Karp, A variable order Runge-Kutta method for value problems with rapidly varying right-hand sides, *ACM Trans. Math. Softw.*, **16,** 3 (1990) 201–222. doi:10.1145/79505.79507 ⇒270, 274

[5] C. Castillo-Chavez, C. W. Castillo-Garsow, A.-A. Yakubu, Mathematical models of isolation and quarantine, *JAMA*, 290, 21 (2003) 2876–2877. doi:10.1001/jama.290.21.2876 ⇒266

[6] H. Chen, D. Zeng, P. Yan, Data visualization, information dissemination, and alerting, In *Integrated Series in Information Systems*, vol. 21. Springer, New York, NY. 2010, pp. 73–87. doi:10.1007/978-1-4419-1278-7_5 ⇒261

[7] Th. Coleman Y Li, On the convergence of reflective newton methods for large-scale nonlinear minimization subject to bounds, *Math. Program.* **67,** 1-3 (1994) 189–224. doi: 10.1007/BF01582221 ⇒268

[8] I. Cooper, A. Mondal, Ch. G. Antonopoulos, A SIR model assumption for the spread of COVID-19 in different communities, *Chaos, Solitons & Fractals*, **139** (2020) 110057. doi:10.1016/j.chaos.2020.110057 ⇒263

[9] L. Gardner E. Dong, H. Du, An interactive web-based dashboard to track Covid-19 real time, *THE LANCET Infectious Diseases* **20,** 5 (2020) 533–534. doi:10.1016/S1473-3099(20)30120-1 ⇒262

[10] Cs. Farkas, D. Iclanzan, B. Olteán Péter, G. Vekov, Estimation of parameters for a temperature and humidity-dependent compartmental model of the Covid-19 outbreak, *Preprint*, 2020. ⇒264, 266, 276

[11] Q. Griette, Z. Liu, P. Magal, Estimating the last day for Covid-19 outbreak in mainland China, *Preprint at medRχiv.org*, July 6, 2020. doi:10.1101/2020.04.14.20064824 ⇒263

[12] B. L. Humphreys, Meeting information needs in health policy and public health: Priorities for the national library of medicine and the national network of libraries of medicine, *Journal of Urban Health*, 75, 4 (1998) 878–883. doi:10.1007/BF02344515 ⇒262

[13] A. Kaddar, A. Abta, H. T. Alaoui, A comparison of delayed SIR and SEIR epidemic models, *Nonlinear Analysis: Modelling and Control* **16,** 2 (2011) 181–190. doi:10.15388/na.16.2.14104 ⇒263

[14] M. Klompas, M. Murphy, J. Lankiewicz, J. McVetta, R. Lazarus, E. Eggleston, P. Daly, P. Oppedisano, B. Beagan, Ch. Kirby, R. Platt, Harnessing electronic health records for public health surveillance, *Online Journal of Public Health Informatics*, **3,** 3 (2011) doi:10.5210/ojphi.v3i3.3794 ⇒261

[15] G. Li, Y. Fan, Y. Lai, T. Han, Z. Li, P. Zhou, P. Pan, W. Wang, D. Hu, X. Liu, Q. Zhang, J. Wu, Coronavirus infections and immune responses, *Journal of Medical Virology*, **92,** 4 (2020) 424–432. doi:10.1002/jmv.25685 ⇒266

[16] Q. Lin, S. Zhao, D. Gao, Y. Lou, S. Yang, S. S. Musa, M. H. Wang, Y. Cai, W. Wang, L. Yang, D. He, A conceptual model for the coronavirus disease 2019 (covid-19) outbreak in wuhan, china with individual reaction and governmental action, *International Journal of Infectious Diseases*, **93** (2020) 211–216. doi:10.1016/j.ijid.2020.02.058 ⇒260, 262, 263

[17] Y. Ma, Y. Zhao, J. Liu, X. He, B. Wang, Sh. Fu, J. Yan, J. Niu, J. Zhou, B. Luo, Effects of temperature variation and humidity on the death of Covid-19 in Wuhan, China, *Science of the Total Environment*, **724,** 7 (2020) 138226. doi:10.1016/j.scitotenv.2020.138226 ⇒263

[18] S. G. Nash, A survey of truncated-Newton methods, *Journal of Computational and Applied Mathematics* **124,** 1-2 (2000) 45–59. doi:10.1016/S0377-0427(00)00426-X ⇒261

[19] D. Revere, A. M. Turner, A. Madhavan, N. Rambo, P. F. Bugni, A. Kimball, Sh. S. Fuller, Understanding the information needs of public health practitioners: A literature review to inform design of an interactive digital knowledge management system, *Journal of Biomedical Informatics*, **40,** 4 (2007) 410–421. doi:10.1016/j.jbi.2006.12.008 ⇒262

[20] A. Sarikaya, M. Correll, L. Bartram, M. Tory, D. Fisher, What do we talk about when we talk about dashboards? *IEEE Transactions on Visualization and Computer Graphics*, **29,** 1 (2019) 682–692. ⇒269, 270, 273

[21] B. Tang, N. L. Bragazzi, Q. Li, S. Tang, Y. Xiao, J. Wu, The effectiveness of quarantine and isolation determine the trend of Covid-19 epidemics in the final phase of current outbreak in China, *International Journal of Infectious Diseases* **95,** 6 (2020) 288–293. doi:10.1016/j.ijid.2020.03.018 ⇒260, 262, 263

[22] B. Tang, N. L. Bragazzi, Q. Li, S. Tang, Y. Xiao, J. Wu, An updated estimation of the risk of transmission of the novel coronavirus (2019-nCov), *Infectious Disease Modelling*, **5** (2020) 248–255. doi:10.1016/j.idm.2020.02.001 ⇒260, 262, 263

[23] B. Tang, X. Wang, Q. Li, N. L. Bragazzi, S. Tang, Y. Xiao, J. Wu, Estimation of the transmission risk of the 2019-nCov and its implication for public health interventions, *Journal of Clinical Medicine* **9,** 2 (2020) 462. doi:10.3390/jcm9020462 ⇒263

[24] R. N. Thompson, Epidemiological models are important tools for guiding COVID-19 interventions, *BMC Medicine* 18, 152 (2020). doi:10.1186/s12916-020-01628-4 ⇒261

[25] B. Udugama, P. Kadhiresan, H. N. Kozlowski, A. Malekjahani, M. Osborne, V. Y. C. Li, H. Chen, J. B. Gubbay S. Mubareka, W. C. W. Chan, Diagnosing Covid-19: The disease and tools for detection, *ACS Nano* **14,** 4 (2020) 3822–3835. doi:doi:10.1021/acsnano.0c02624 ⇒262

[26] J. Wangping, H. Ke, S. Yang, C. Wenzhe, W. Shengshu, Y. Shanshan, W. Jianwei, K. Fuyin, T. Penggang, L. Jing, L. Miao, H. Yao, Extended SIR prediction of the epidemics trend of COVID-19 in Italy and compared with Hunan, China. *Frontiers in Medicine*, **7,** 5 (2020) doi:10.3389/fmed.2020.00169 ⇒263

[27] S. Wexler, J. Shaffer, A. Cotgreave, *The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios*, Wiley Publishing, 1st edition, 2017. ISBN: 978-1-119-28271-6 ⇒269

[28] Y. Xiao, S. Tang, J. Wu, Media impact switching surface during an infectious disease outbreak, *Scientific Reports* **5,** 7838 (2015). doi:10.1038/srep07838 ⇒263, 266

[29] Z. Yang, Zh. Zeng, K. Wang, S.-S. Wong, W. Liang, M. Zanin, P. Liu, X. Cao, Zh. Gao, Zh. Mai, J. Liang, X. Liu, Sh. Li, Y. Li, F. Ye, W. Guan, Y. Yang, F. Li, Sh. Luo, Y. Xie, B. Liu, Zh. Wang, Sh. Zhang, Y. Wang, N. Zhong, J. He, Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions, *Journal of Thoracic Disease* **12,** 3 (2020) 165–174. doi:10.21037/jtd.2020.02.64 ⇒263

[30] Ya-xiang Yuan. A review of trust region algorithms for optimization, *ICM99: Proceedings of the Fourth International Congress on Industrial and Applied Mathematics*, September, 1999. ⇒267

Received: October 13, 2020 • Revised: November 11, 2020

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 283–301

DOI: 10.2478/ausi-2020-0017

Statistical complexity of the kicked top model considering chaos

Ágnes FÜLÖP
Faculty of Informatics
Loránd Eötvös University, Budapest
email: fulop@caesar.elte.hu

Abstract.

The concept of the statistical complexity is studied to characterize the classical kicked top model which plays important role in the qbit systems and the chaotic properties of the entanglement. This allow us to understand this driven dynamical system by the probability distribution in phase space to make distinguish among the regular, random and structural complexity on finite simulation. We present the dependence of the kicked top and kicked rotor model through the strength excitation in the framework of statistical complexity.

1 Introduction

In this article we study the driven systems considering the statistical complexity. The concept of statistical complexity has been introduced in different way from complexity of finite series (Lempel, Ziv) [36], algorithmic complexity (Kolmogorov) [30], the amount of information about the optimal prediction, where the future fulfills to the expected past (Crutchfield, Young) [17]

The effective entropy was published by Grassberger [26] considering the mixture of the order and disorder, regularity and randomness, since the entropy of most systems is between the maximum and minimum entropy values.

Computing Classification System 1998: F.2.1
Mathematics Subject Classification 2010: 68U20
Key words and phrases: statistical complexity, Shannon entropy, chaos, kicked top map, kicked rotor map, qbit

The definition of statistical complexity was introduced by López, Ruiz, Manchini, Calbet (LMC) [2] and Shiner, Davison, Landsberg (SDL) [55]. The generalized statistical complexity measure (Martin, Palestino, Rosso) [43] is based on the LMC's that gives a description of the finite sequence of nonlinear systems with the adequate probability distribution of the time dependent method. It was extended to Tsallis, Wootters, Renyi entropy and Kullback-Leibler, Jensen-Shannon divergence. Tsallis suggested a generalization of the Shannon-Boltzmann-Gibbs entropy measure [58]. The new entropy funtional plays an significant role along with its corresponding thermodynamics (1998). Wootters reflected on the Euclidean distance [61], because he studied this concept in a quantum mechanical field; this consideration allowed to consider an intrinsic statistical measure, this concept can be employed to any probability space.

Both experimental and theoretical sign can be evaluated by the information theory tools, as entropy, distance, statistical divergence provides an opportunity to make estimation, detection and transmission processes.

In the last two decades more kind of complexity measures and methodologies were introduced with their time evolution connected to optimal predictability, symbolic analysis [1], algorithmic data compression, number system, pseudo-random bit generator, earthquake the chaotic regim etc. [36, 17, 59, 38, 18, 55, 43, 35, 11, 22, 25, 39, 19]. The dynamics of the statistical complexity measure is formulated according to the second law of thermodynamics. According to, entropy increases monotonically at time. It follows that the quantity H is implied as an arrow of time.

We study the kicked top and kicked rotor model in this article, these driven systems are intensively researched field in quantum mechanics. These are studied in the field of quantum chaos and entanglement in ergodic and non ergodic system [48]. In the past two decades the kicked top model was an intensively researched area which contained the chaotic dynamics and quantum correlations considering in quantum information and computation. The kicked top is a suitable model for studying spins or qbits and corresponds for the studying of entanglement. We approximate the classical limit of the kicked top if the number of spins tend to infinity.

Therefore this model is an important area of research [9, 33, 42, 52] for the study of entanglement [37, 24, 45, 34, 4, 60] and its relationship to classical dynamics [57], sign of bifurcations on different quantum correlation measures [9], quantum classical transition with respect to periodic trajectories [33] and the behavior of entropy in the transition to chaos citezs. Measure of quantum correlations is strongly correlated with the qualitative nature of classical phase

space, whether it is regular or chaotic [9, 52, 37, 3, 41, 20, 62]. The importance of the kicked top model is also demonstrated by the large number in a series of papers [14, 9, 52, 37, 3, 41, 20, 47, 10].

The structure of the article contains the next parts:

In the section (2) we introduce the idea of complexity accordingly the measure of entropy and disequilibrium with the probability distributions by the by LMC functional associating to SCM family. We discuss the statistical complexity considering the Wootters, Kullback-Leibler relative entropy and the Jensen-Shannon divergency. The time evolution of the SCM associated to the evolving of entropy. In the section (3) the quantum kicked top model is investigated by the Hamiltonian functions considering the properties of the classical equation motions. The quantum kicked rotor system is derived by the Hamiltonian and we compere the chaotic behavior of these systems in the section (4). The Numerical approximation of the statistical complexity of the models is discussed in the section (5).

2 Statistical complexity measures

In this section we discuss the entropy and distance in the probability space which can be used to determine the statistical complexity measure. This plays important role in the dynamics of quantum-classical transition and the chaotic motion. We review the main futures of statistical considerations describing dynamical properties.

2.1 Information measures

The information measure I is defined by a given probability distribution. $I[P]$ refers as the measure of the uncertainty connected to probability distribution $P = \{p_j, j = 1, \ldots, N\}$, where N indicates the the number of possible states of the systems satisfying $\sum_{j=1}^{N} p_j = 1$ (micro-canonical ensemble).

If $I[P] = I_{min} = 0$ then this means that the maximum information is extracted from all possible outputs states. Otherwise the ignorance appears when $I[P] = I[P_e] \equiv I_{max}$; $P_e = \{p_i = 1/n; \forall i\}$, P_e being the uniform distribution. These are the trivial cases. We define the amount of disorder H at a given probability distribution P and considering the information measure $I[P]$:

$$H[P] = I[P]/I_{max} \tag{1}$$

The value of H is changing $0 \leq H \leq 1$.

Based on the Shannon-Kinchin paradigm I is introduced in the expression of entropy. It can arise by canonical formulation (Boltzmann-Gibbs) of statistical mechanics which is expandable to another entropy term as Renyi, Tsallis [51]. We define the disorder H for the $P \equiv \{p_i, i = 1, \ldots, N\}$ on a discrete probability distribution:

$$H[P] = S[P]/S[P_e], \tag{2}$$

where $S[P]$ means Shannon's logarithm entropy [53] by this form

$$S[P] = -\sum_{j=1}^{N} p_j \log(p_j) \tag{3}$$

and $S[P_e] = \log N$.

2.2 Distances and statistical complexity measure

In order to define Statistical Complexity Measure (SCM) we need to use some distance D [31] between given P and the uniform distribution P_e on the available states of the system [38, 43, 35].

$$Q[P] = Q_0 \cdot D[P, P_e], \tag{4}$$

where Q_0 is a normalization constant ($0 \leq Q \leq 1$) i. e. the inverse of the maximum distance $D[P, P_e]$. The value of largest distance corresponds to that one component of probability distribution P takes 1 and the others equal to zero. The disequilibrium-distance Q shows the structure of the system, because the "privileged" states differ from zero probability value.

The functional form of the SCM is introduced by Lopetoz-Ruiz, Manchini and Calbet (LMC) [38].

$$C[P] = H[P] \cdot Q[P] \tag{5}$$

This quantity represents at a given scale between the amount of information stored in the system and its disequilibrium [38]. In this article we study complex dynamics where the different regime are mixed, i.e. chaos, regular islands and trajectory that are neither periodic nor chaotic can be featured by SCM.

Different distance-forms D can be used to define the quantity Q for the SCM. In the following we apply two discrete probability distributions $P_i \equiv \{p_1^{(i)}, \ldots, p_N^{(i)}\}$, with $i = 1, 2$ consider the next options:

(I) Euclidean norm D_E in $\mathbb{R}^N$ [38]:
This is natural case for the distance D. We get

$$D_E[P_1, P_2] = \|P_1 - P_2\|_E^2 = \sum_{j=1}^{N} \left\{ p_j^{(1)} - p_j^{(2)} \right\}^2 \tag{6}$$

This is the disequlibrium term which was contained in the original complexity measure by López-Ruiz, Manchini and Calbet (LMC-complexity measure [38]). Wootter extended it to the possibility, where we also consider the shape of the probability distributions.

(II) Wootter's distance D_W [43, 61]

The concept of statistical distance is extended in a quantum mechanical field. He proposed a definition to distinguish among different preparations of a given quantum state and to take it into account that two such states differ from one another inside statistical error. It allows to consider an intrinsic statistical nature, this can be applied to any probabilistic space [61].

$$D_W[P_1, P_2] = \cos^{-1} \left\{ \sum_{j=1}^{N} \left(p_j^{(1)} \right)^{1/2} \cdot \left(p_j^{(2)} \right)^{1/2} \right\} \tag{7}$$

Two divergence classes were distinguished by Basseville [5]. The first class contains divergences defined by relative entropy, while the second one pays attention divergences related as entropy differences.

(III) Kullback-Leiber relative entropy D_K [32]:

The relative entropy of P_1 with respect to P_2 connected to Shannon measure is the relative Kullback-Leibler Shannon entropy in the discrete case follows

$$D_K[P_1, P_2] = K[P_1|P_2] = \sum_{j=1}^{N} p_j^{(1)} \log \left(\frac{p_j^{(1)}}{p_j^{(2)}} \right) \tag{8}$$

The distance between the probability distribution P and uniform distribution P_e in the Kullback-Leibler Shannon expression is given by this form

$$D_K[P, P_e] = K[P|P_e] = S[P_e] - S[P] \tag{9}$$

(IV) Jensen divergence D_j [35]:

The entropic difference $S[P_1] - S[P_2]$ does not mean an information gain (or divergence), bacause the difference is not inevitably positive definite. Jensen's

divergence is a symmetric version of the Kullback-Leibler relative entropy, which can be written in the form of the Shannon entropy as follow:

$$\begin{aligned} D_J[P_1, P_2] &= J_S[P_1, P_2] = \{K[P_1|P_2] + K[P_2|P_1]\}/2 \\ &= S[\tfrac{P_1+P_2}{2}] - S[P_1]/2 - S[P_2]/2 \end{aligned} \tag{10}$$

The Jensen-Shannon divergence verifies the following properties

$$\begin{aligned} &(i) \quad J_S[P_1, P_2] \geq 0 \\ &(ii) \quad J_S[P_1, P_2] = J_S[P_2, P_1] \\ &(iii) \quad K_S[P_1, P_2] = 0 \Leftrightarrow P_2 = P_1 \end{aligned} \tag{11}$$

It square root fulfills the triangle inequality:

$$(iv) \quad (J_S[P_1, P_2])^{1/2} + (J_S[P_2, P_3])^{1/2} = (J_S[P_1, P_3])^{1/2} \tag{12}$$

So the square root of the Jensen-Shannon divergence is a metric [12]. These entropy concepts are extensive quantities in thermodynamics, therefore the associated statistical complexity will be an intensive quantity.

Generally on the basis of LMC-functional product term we get a family of SCMs for each four disequilibrium

$$C^{\nu}[P = H[P] \cdot Q_{\nu}[P] \tag{13}$$

The index $\nu = E, W, K, J$ denoted the disequilibrium distance which is determined with the adequate distance measure (Euclidean, Wootters, Kullback-Leibler, and Jensen-Shannon) Then the SCM family for $\nu = K$ is following

$$C^{(K)}[P] = H[P] \cdot Q_K[P] = H[P] \cdot (1 - H[P]) \tag{14}$$

The generalized functional term was introduced by Davison and Landsberg [55] for the SCM. Similar results are published by these article [16, 8, 56].

We consider three members of the family C^{ν} ($\nu = E, W, J$) these are not trivial functions of the entropy [44] because they associate to two dissimilar probabilities distributions P and uniform distribution P_e. It can be seen that a given H value determines a range of SCM values from C_{min} to C_{max}. These bounds are changing during the time evolution. We obtain the range C_{min} and C_{max} relating to the generalized $C^{\nu} = H \cdot Q_{\nu}$ family which provides more information corresponding to the correlation structure between the elements of physical system.

2.3 The evolution

In statistical mechanics isolated systems [13] play important role featured by an initial discrete probability distribution going toward equilibrium. The uniform distribution P_e characterizes the equilibrium. The evolution of the SCM can be plotted on the Figure of C versus time t. Nevertheless in isolated system the entropy grows monotonically with time ($dH/dt \geq 0$) by the second law of thermodynamics [49]. It follows that H behaves as an arrow of time, i.e. the time evolution of the SCM corresponds to plot C versus H. The normalized entropy-axis equivalent with the time-axis [38, 51, 50].

3 Kicked top model

Quantum kicked top The Quantum Kicked top (QKT) is e time-dependent periodic system, which is described by an angular momentum vector $J = (J_x, J_y, J_z)$. Here we choose natural unit where the Planck's constant has been adjusted to unity. The time evolution of the model is given by Hamiltonian

$$H(t) = pJ_y + \frac{k}{2j}J_z^2 \sum_{n=-\infty}^{\infty} \delta(t - n\tau) \tag{15}$$

The first expression of the Equation (15) means the free precession of the kicked top model around y axis with angular frequency p. The second expression indicates the periodic δ kicks on the kicked top system, where each kick causes a torsion by an angle $(k/2j)J_z$ about the z axis. The components of angular momentum satisfy the commutation relations in standard algebra of angular momentum:

$$[J_i, J_j] = i\varepsilon_{i,j,k}J_k \tag{16}$$

The magnitude of total angular momentum $J^2 = j(j+1)\hbar^2$ is conserved quantity. The classical limit is obtained when $j \to \infty$. The time between periodic kicks corresponds to τ. In this article it is chosen unit ($\tau = 1$). The parameter k characterizes the chaotic behavior of the system and the strength of the kick. If $k = 0$ then the equation (15) can be integrated, which is the classical boundary of the system. As the value of k increases, the chaoticity of this model is growing.

The expression of the periodic-one Floquet operator for the Hamiltonian equation (15) is as follows:

$$U = \exp\left(-i\frac{k}{2j}J_z^2\right)\exp(-ipJ_y) \tag{17}$$

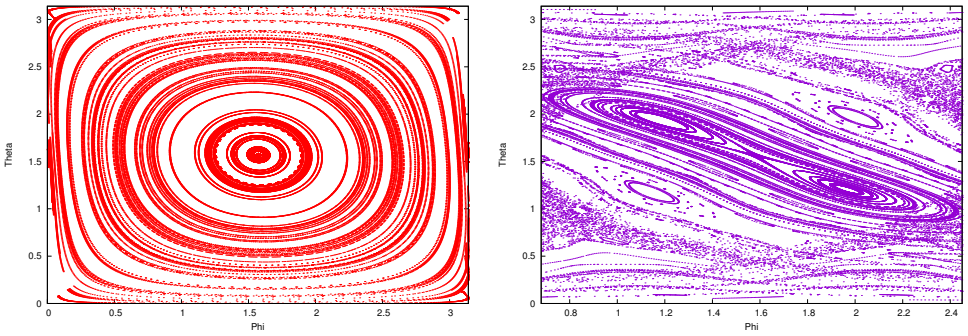


Figure 1: The Θ depends on Φ variables at the kicked top model at $k = 0.07$, $k = 2.2$.

Each time period contains a linear rotation by angle p around the y axis and a nonlinear rotation around the z axis. The dimension of Hilbert space is $2j + 1$ therefore the time dependent behavior can be described without any truncation of the Hilbert space.

The quantum simulation of a given set of qbits with $N = 2j$ is well described by the quantum kicked top model for given angular momentum j. These are half-spin particles which are confined to a subspace that is symmetrical for qbit exchange.

In the symmetric subspace the state vector is defined by the following states $\{|j, m>: (m = -j, -j + 1, \ldots, j)\}$ where $j = N/2$. The ground conditions fulfill the following conditions $S_z|j, m >= m|j, m >$ and $S_\pm|j, m >= \sqrt{(j \mp m)(j \pm m + 1)}|j, m \pm 1 >$, where S_z and $S_\pm$ are collective spin operator [46]. This is a multiqubit system and the collective properties evolve according to Hamiltonian Eq. (15).

According to the quantum mechanics, the initial state of the kicked top system is a spin coherent state(minimum-uncertainty states) pointing in the direction of Φ, Θ. The time evolution is determined by the Floquet operator. The classical map of the kicked top model is discuss below.

Classical kicked top The phase space is represented in Fig. (1) as a function of coordinates Φ and Θ.

The classic map of the kicked top model is derived in the following form [28]

$$\begin{aligned} X' &= (X\cos p + Z\sin p)\cos[k(Z\cos p - X\sin p)] - Y\sin(k(Z\cos p - X\sin p)) \\ Y' &= (X\cos p + Z\sin p)\sin[k(Z\cos p - X\sin p)] + Y\cos[k(Z\cos p - X\sin p)] \\ Z' &= -X\sin p + Z\cos p \end{aligned} \tag{18}$$

The time-dependent variables (X, Y, Z) fulfill the constraint $X^2 + Y^2 + Z^2 = 1$. The trajectories are located on a sphere of unit radius.

These equations (18) can be specified by polar coordinates i.e. with polar angle Φ and azimuth angle Θ, therefore $X = \sin\Theta\cos\Phi, Y = \sin\Theta\sin\Phi, Z = \cos\Theta$. During the time evolution of the equations, the values of Φ and Θ are determined in each step. The symmetry properties of the model are discussed below.

The phase space is reflective on $\Theta = \frac{\pi}{2}$ during the transformation $k \to -k$. This is fulfilled because $k \to -k$ transformation has the same meaning as $X \to -X$ and $Z \to -Z$ in Eq. (18). It follows that $Z' \to -Z'$ due to which $\Theta \to \pi - \Theta$. Therefore the $k \to -k$ transformation is an isomorphism in the phase space.

Further symmetry can be found in the Equations (18) studying the classical map. The classical map contains the parameter p. So we analyze the dependence on it which leads to different simpler equations. first consider the system at $p = \frac{\pi}{2}$. It was studied by wide range of articles [9, 33, 3, 41, 47, 28] in the literature. Due to newer symmetries, the shape of the mapping is simplified as follows

$$\begin{aligned} X' &= Z\cos(kX) + Y\sin(kX) \\ Y' &= Y\cos(kX) - Z\sin(kX) \\ Z' &= -X \end{aligned} \tag{19}$$

At small values k, the phase space is mainly covered by regular trajectories (Fig. (1)) at $k = 0.07$. The trivial fixed points is situated at $(\Phi, \Theta) = (\pi/2, \pm\pi/2)$. Increasing the value of k, the chaotic regions expands more and more in the phase space. For growing parameter value k the phase space contains mainly chaotic sea with a few regular regions.

The map is studied for the value of the parameter $p = 3\pi/2$. It is derived from $p = \pi/2$ by the the transformation $X' \to -X'$ and $Z' \to -Z'$. This means reflections about $\Phi = 0$ and $\Theta = \pi/2$ because $\Phi \to -\Phi$ and $\Theta \to \pi - \Theta$. In the case of phase space, we also find such a behavior when we use these reflection.

The next p value is chosen to be π, then the Equation (18) of the classical map forms:

$$\begin{aligned} X' &= Y\sin(kZ) - X\cos(kZ) \\ Y' &= Y\cos(kZ) - X\sin(kZ) \\ Z' &= -Z \end{aligned} \tag{20}$$

In this case the fully developed chaos does not appear. The angle Θ is changing between $\cos^{-1} Z$ and $\pi - \cos^{-1} Z$ at a given initial value of Z. These quantities are reflected for $\pi/2$.

Th last instance is $p = 2\pi$ which we investigate

$$\begin{aligned} X' &= X\cos(kZ) - Y\sin(kZ) \\ Y' &= X\sin(kZ) + Y\cos(kZ) \\ Z' &= Z \end{aligned} \tag{21}$$

In this situation the fully developed chaos does not evolve for a given initial value Z and the angle Θ equals to constant at $\cos^{-1} Z$.

4 Quantum kicked rotor

The kicked rotor(QKR) plays an important role in the research of chaos. The Hamiltonian of this driven system is

$$H_R = \frac{1}{2I}P^2 + k\cos\Phi \sum_{n=-\infty}^{\infty} \delta(t - nT), \tag{22}$$

where Φ is the angle operator and P is the angular momentum, canonically conjugate to Φ and T is a periodic time. The strength of the kick is denoted by k and I is the moment of inertia and the rotor operators satisfy the communication relation:

$$[P, \Phi] = -i. \tag{23}$$

From the discrete dynamics, we get the angular operator and angular momentum from driven to driven in the Heisenberg picture by these equations:

$$\begin{aligned} P' &= U_R^{\dagger} P U_R \\ \Phi' &= U_R^{\dagger} \Phi U_R, \end{aligned} \tag{24}$$

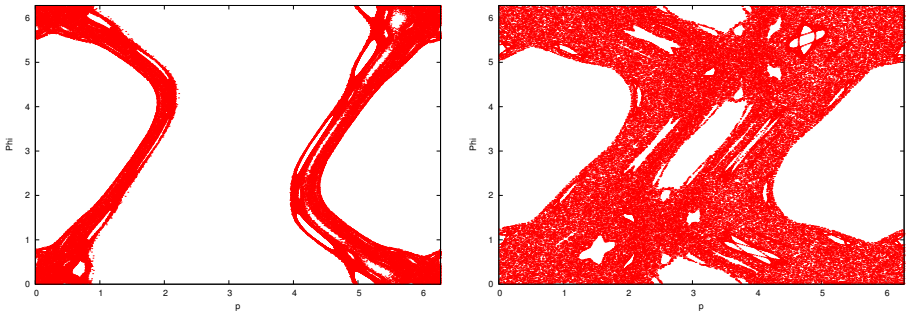


Figure 2: Kicked rotor model $k = 0.9$ $k = 1.26$.

where the Floquet uniter operator U_R is defined by this term

$$U_R = \exp\left(-i\frac{P^2}{2I}\right)\exp(-ik\cos\Phi) \tag{25}$$

The stroboscopic equations is as follows

$$\begin{aligned} P' &= P + k\sin\Phi \\ \Phi' &= \Phi + P'/I. \end{aligned} \tag{26}$$

The classical equation of motion following from Eq. (22) in the literature it is known as Chirikov's standard map [15, 23, 6] ($I = T = 1$).

The surface of the rotor phase space is a cylinder, $-\infty < P < \infty$, $0 \leq \Phi < 2\pi$. It is seem from the stroboscopic equation that the model is invariant under $2\pi I$ translations in P and 2π in Φ, even though the model is not bounded in P.

Comparing the topology of the systems rotor and kicked top model this is dissimilar because the kicked top model holds spherical phase space. The classical rotor model is plotted in Fig. (2) for $k = 0.9$, $I = 1$.

The classical phase space of the rotor model contains elliptic and hyperbolic fixed points and KAM tori (as circles) corresponding to KAM theory and Poincaré-Birkhoff theorem. The KAM tori are invariant sets therefore the chaotic trajectories can not pass through to evolve hyperbolic fixed points. In the case of $k = 0$ the model is a free rotor corresponding to regular motion. As the value of k becomes larger, the KAM tori decompose into cantori [40, 7](invariant Cantor sets) and these are partly passable. There exists a critical nonlinearity k_c with universal scaling properties [27, 54], where the

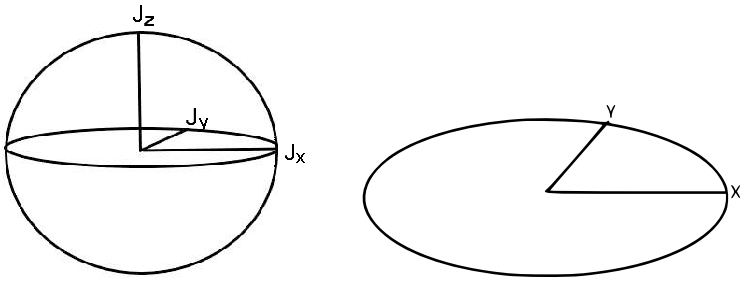


Figure 3: (left)The magnitude of the angular momentum J of the kicked top model is connserved quantity therefore it is displayed on a sphere. (right) The rotor limit is driven with rescaling $\alpha = k/j$, $\beta = j/I$, as $j \to \infty$. If we start in the equatorial waistband, the rescaling constrains the angular momentum to $X = \cos\Phi$, $Y = \sin\Phi$, P/j.

final KAM tori split the phase space into partition along the P axis and the system becomes globally chaotic. The classical KAM theory is extended to the quantum sytems [23].

Classical rotor-limit The time evolution of the rotor map can be risen from the kicked top model, if we bound the top to an equatorial waistband as plotted in Fig. (3). Then the precession frequency is decreased around the x-axis fulfilling the rescaling.

$$\alpha = k/j, \quad \beta = j/I, \tag{27}$$

where $j \to \infty$. This means the rotor-limit of the top kicked model.

We begin in the equatorial waistband, this rescaling recricts the angular momentum to Fig. (3)

$$X = \cos\Phi, \quad Y = \sin\Phi \quad Z = P/j. \tag{28}$$

If we replace the Eq. (27) and (28) in the kicked top map of Eq. (18), we get the kicked rotor map of Eq. (26) [29].

Quantum rotor-limit Here we introduce the next rescaled operators:

$$\hat{X} \equiv \hat{J}_x/j, \quad \hat{Y} \equiv \hat{J}_y, \quad \hat{P} \equiv \hat{J}_x \tag{29}$$

These operators fulfill the communication relations of Eq. (16). Consider $j \to \infty$ so that we may cat the $1/j^2$ terms:

$$[\hat{X}, \hat{Y}] = 0, \quad [\hat{Y}, \hat{P}] = i\hat{X}, \quad [\hat{P}, \hat{X}] = i\hat{Y}. \tag{30}$$

These expressions fulfill the communication relations:

$$\hat{X} = \cos\phi, \quad \hat{Y} = \sin\Phi, \quad \hat{P} = -i\frac{\partial}{\partial\Phi}. \tag{31}$$

These Equations (27) (29) and (30) are put in the top Hamiltonian of Eq. (15) than we obtain the rotor Hamiltonian of Eq. (22).

5 Numerical approximation

In this section we represent the statistical complexity of the kicked top and kicked rot model. This is a well signature of the chaotic features of these systems to show the mixed inner structure and the time dependent evolution in the top model associating the behavior of qbits.

Statistical complexity The statistical complexity is introduced on the probability distribution yielding a statistical estimation of the points in the phase space (section 2).

The dynamical behavior of the systems is discussed in the next form [21]. The notation of measured sequence is denoted by $y_1, \ldots, y_n$ time series, where y_i corresponds to measurement of the quantity y at the time $t_i = t_0 + iT$, ($T > 0 \in \mathbb{R}$). The trajectory of length $n \in \mathbb{R}^d$ i.e. time series of the measurement is written by $\underline{x}^{(n)}$. The point of the orbit of the length n is denoted by $x_k^{(n)}$, ($k = 1, \ldots, n$) and the set K contains the points of some trajectories $x_k^{(n)}$ ($k = 1, \ldots, n$). Let us consider a time sequent of length $N' >> n$. A given series $x_k^{(n)}$, ($k = 1, \ldots, n$) appears with probability $P(\underline{x}^{(n)})$ along the the sequences of length N', where the corresponding set of discrete probability distribution $P \equiv \{p_1, \ldots, p_{N'}\}$, $p_i = P(\underline{x}_i^n)$ ($\sum_{i=1}^{N'} p_i = 1$), and $p_i > 0 \ \forall i$.

The driven systems can be derived numerically in different methods. We may determine a single very long trajectory of the periodically excited system or compute an ensemble of orbits. If the kicked model is periodic, the single long trajectory can be simulated stroboscopically in the three dimensional phase space. We select the smooth initial values in the past from the domain of the map for the variables x, y and z at time $t = 0$. The periodicity is $T = 1$

and the length of the trajectory is chosen as $N = 10^4$. On this basis the value of the entropy, disequilibrium and statistical complexity are able to determine unambiguously.

The motion of the (a)kicked top and (b)kicked rotor map becomes on a surface in the three dimensional phase space. The trajectories of the kicked top map are found on the sphere (section 3) and the orbits of rotor map are located on cylinder (section 4). In the periodically driven model these systems depend on the parameter k, this means the strength of the excitation.

The chaotic behavior turns up at the critical parameter value k_c and the statistical complexity C becomes to zero. The quantity of the kicked top map is $k_c = 1.26$ and the value of the rotor map is $k_c = 2.64$ ((a)Fig. 4, (b)Fig. 5). Because the distance between the probability distribution P and the uniform distribution P_e tends to zero and the entropy approaches 1 at the equal probability points in the phase space.

The parameter k is extended to the neighbor of critical values (a)$k \in [0, 6]$ (b)$k \in [0, 2]$. Due to the increasing strong perturbation of the periodic driven systems, the statistical complexity C decreases to zero at the same time the value of the entropy H tends to 1 depending on the parameter k, therefore the values C and H changing between extreme states C_{max} at $H \sim 0$ and C_{min} at $H \sim 1$ with the transition intervals. The chaotic behavior corresponds to the range, where the values is $C = 0$ and $H = 1$ (Fig. (6)). For the calculation owing to the finite size of the simulation, the count accuracy becomes larger when the number of element N increases. At the small value k the periodically driven forcing does not have effect on the model i.e. the the motion of system is regular on the surface.

The spectrum of the statistical complexity is finite and limiting but not inevitably a unique function of H and there exists a range of values between a minimal value C_{min} and a maximal value C_{max} containing the inner structure (Fig. (6)).

Because the number of points on the phase space is finite, C as a function H shows scaling behavior, i.e. the bigger complexity associates with less entropy with a larger discrete probability distribution. Since the probability distribution of element in the phase space is discontinuous in the three-dimensional space, some complexity and disequilibrium values do not appear for certain entropy quantities.

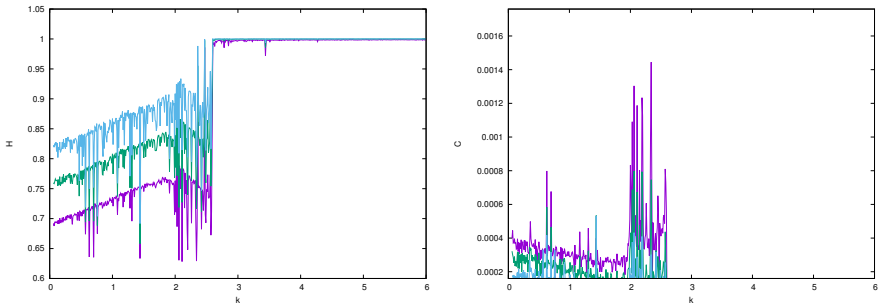


Figure 4: Kicked top model (left):Entropy H as a function strength of the driven k. (right): Statistical complexity C depends on the driven parameter k.

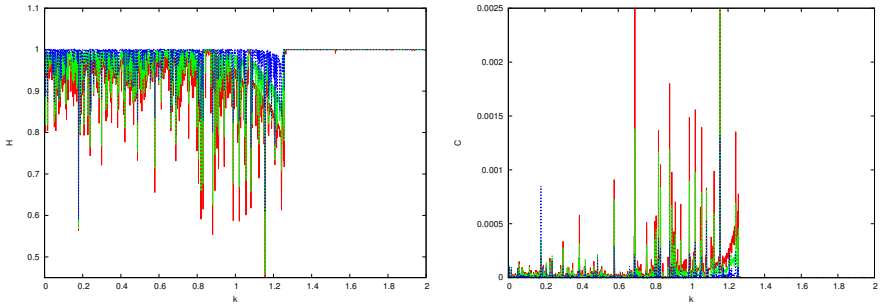


Figure 5: Kicked rotor model: (left): Entropy H as a function strength of the driven k. (right): Statistical complexity C depends on the driven parameter k.

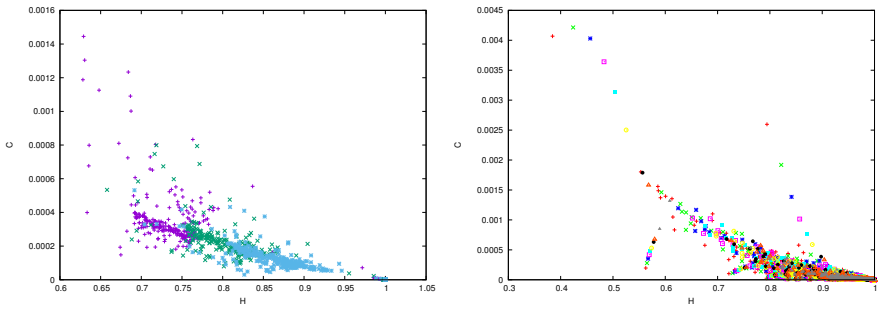


Figure 6: The statistical complexity C depends on the entropy H. left:kicked top model, right: kicked rotor system.

6 Conclusion

The model describing the qbits are kicked top and rotor model, which are studied by statistical complexity in a finite probability distribution in a three-dimensional space considering the measure of the entropy and disequilibrium using the scaling behavior of these quantities. At the range of chaoticity the statistical complexity approximates zero and entropy goes to one.

We extended the parameter values of k to the neighbor of the critical quantities studying the spectrum of the statistical complexity and the disequilibrium depending in the entropy. In the range of parameter k we reached the dependence of the quantities C and H, which is acts as a periodical driven force.

References

[1] C. Adami, N. T. Cerf, Physical complexity of symbolic sequences, *Physica D: Nonlinear Phenomena* **137** (2000) 62–69. doi:10.1016/S0167-2789(99)00179-7 ⇒ 284

[2] C. Anteneodo, A. R. Plastino, Some features of the López-Ruiz-Manchini-Calbet (LMC) statistical measure of complexity, *Physics Letters A* **223** (1996) 348–354. doi:10.1016/S0375-9601(96)00756-6 ⇒284

[3] J. N. Bandyopadhyay, A. Lakshminarayan, Entanglement production in coupled chaotic systems: Case of the kicked tops *Phys. Rev. E* **69** (2004) 016201. doi:10.1103/PhysRevE.69.016201 ⇒285, 291

[4] J. N. Bandyopadhyay, A. Lakshminarayan, Testing Statistical Bounds on Entanglement Using Quantum Chaos *Phys. Rev. Lett.***89** (2002) 060402. doi:10.1103/PhysRevLett.89.060402 ⇒284

[5] M. Basseville, *Information: Entropies, Divergences et Mayennes*, (IRISA) Publication Interne **1020** (1996) (Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France). ⇒287

[6] J. Bene, P. Szépfalusy, A. Fülöp Generic dynamical phase-transition in chaotic Hamiltonian-systems *Phys. Rev. A* **40** (1989) 6719–6722. doi:10.1103/physreva.40.6719 ⇒293

[7] D. Bensimon, L. P. Kadanoff, Extended chaos and disappearance of KAM trajectories *Physica D: Nonlinear Phenomena* **13** (1984) 82–89. doi:10.1016/0167-2789(84)90271-9 ⇒293

[8] P. M. Binder, N. Perry, Comment II on: Simple measure of complexity. *Phys. Rev. E* **62** (2000) 2998–2999. ⇒288

[9] U. T. Bhosale and M. S. Santhanam, Signatures of bifurcation on quantum correlations: Case of the quantum kicked top *Phys. Rev. E* **95** (2016) 012216. doi:10.1103/PhysRevE.95.012216 ⇒284, 285, 291

[10] U. T. Bhosale, M. S. Santhanam Periodicity of quantum correlations in the quantum kicked top, *Phys. Rev. E* **98** (2018) 052228. doi:10.1103/physreve.98.052228 ⇒285

[11] G. Boffetta, M. Cencini, M. Falcioni, A. Vulpiani, Predictability: a way to characterize complexity, *Phys. Reports* **356** (2002) 367–474. doi:10.1016/S0370-1573(01)00025-4 ⇒284

[12] J. Briet, P. Harremoes, Properties of classical and quantum Jensen-Shannon divergence. *Phys. Rev. A* **79** (2009) 052311. ⇒288

[13] X. Calbet, R. López-Ruiz, Tendency towards maximum complexity in a nonequlibrium isolated system, *Phys. Rev. E* **63** 066116. ⇒289

[14] S. Chaudhury, A. Smith, B. E. Anderson, S. Ghose, P. S. Jessen, Quantum signatures of chaos in a kicked top *Nature* **461** (2009) 768. ⇒285

[15] B. V. Chirikov A universal instability of many-dimensional oscillator systems *Phys. Rep.* **52** (1979) 265. ⇒293

[16] J. P. Crutchfield,D.P. Feldman,C.R. Shalizi Comment I on: simple measure of complexity. *Phys.Rev. E* **62** (2000) 2996–2997. ⇒288

[17] J. P. Crutchfield, K. Young, Inferring statistical complexity, *Phys. Rev. Lett.* **63** (1989) 105. ⇒283, 284

[18] D. P. Feldman, J. P. Crutchfield, Measures of statistical complexity: Why? *Phys. Lett. A* **238** (1998)244–252. ⇒284

[19] G. L. Ferri, F. Pennini, A. Plastino, LMC-complexity and various chaotic regime, *Physics Letters A* **373** (2009) 2210–2214. ⇒284

[20] H. Fujisaki, T. Miyadera, A. Tanaka, Dynamical aspects of quantum entanglement for weakly coupled kicked tops *Phys. Rev. E* **67**, (2003)066201. ⇒285

[21] Á. Fülöp, Estimation of the Kolmogorov entropy in the generalized number system, *Annales Univ. Sci. Budapest Sect. Comp.* **40** (2013) 245–256. ⇒295

[22] Á. Fülöp, Statistical complexity and generalized number system, Acta Univ. Sapientiae, Informatica **6** (2) (2014) 230–251. ⇒284

[23] T. Geisel, G. Radons, J. Rubner, Kolmogorov-Arnold-Moser Barriers in the Quantum Dynamics of Chaotic Systems *Phys Rew. Letters* **57** (1986) 2883. ⇒ 293, 294

[24] S. Ghose, R. Stock, P. Jessen, R. Lal, A. Silberfarb, Chaos, entanglement, and decoherence in the quantum kicked top *Phys. Rev. A* **78** (2008) 042318. ⇒284

[25] C. M. Gonzalez, H. A Larrondo, O. A. Rosso, Statistical complexity measure of pseudorandom bit generators, *Physica A* **354** (2005) 281. ⇒284

[26] P. Grassberger, Toward a Quantitative Theory of self-generated complexity, *Int. Journ. Theor. Phys.* **25** (1988) 907–938. ⇒283

[27] J. M. Greene A method for determining a stochastic transition *J. Math. Phys.* **20** (1979) 1183. ⇒293

[28] F. Haake, M. Kus, R. Scharf, Classical and quantum chaos for a kicked top *Z. Phys. B* **65** (1987) 381. ⇒291

[29] F. Haake, D. L. Shepelyansky, The kicked rotator as a limit of the kicked top, *EPL (Europhys. Lett.)* **5** (1988) 671. ⇒294

[30] A. N. Kolmogorov, Entropy per unit time as a metric invariant of automorphism,

Doklady of Russian Academy of Sciences, **124** (1959) 754–755. ⇒283
[31] A. M. Kowalski, M. T. Martin, A. Plastino, O. A. Rosso, M. Casas, Distances in probability space and the statistical complexity setup, *Entropy* **13** (2011) 1055–1075. ⇒286
[32] S. Kullback, R. A Leibler, On information and sufficiency *Ann. Math. Stat.* **22** (1951)79—86. ⇒287
[33] M. Kumari, S. Ghose Quantum-classical correspondence in the vicinity of periodic orbits *Phys. Rev. E* **97** (2018) 052209. ⇒284, 291
[34] A. Lakshminarayan, Entangling power of quantized chaotic systems *Phys. Rev. E* **64** 2001 036207. ⇒284
[35] P. W. Lamberti, M. T. Martin, A. Plastino, O. A. Rosso, Intensive entropic nontriviality measure, *Physica A* **334** (2004) 119—131. ⇒284, 286, 287
[36] A. Lempel, J. Ziv On the complexity of finite sequences, *IEEE Trans. Inform Theory* **22** (1976) 75–81. ⇒283, 284
[37] M. Lombardi, A. Matzkin, Entanglement and chaos in the kicked top *Phys. Rev. E* **83**, 2001 016207 (2011). ⇒284, 285
[38] R. López-Ruiz, H.L. Mancini, X. Calbet, A statistical measure of complexity, *Phys. Letters A* **209** (1995) 321–326. ⇒284, 286, 287, 289
[39] M. Lovallo, V. Lapenna, L. Telesca, Transitionmatrix analysis of earthquake magnitude sequences *Chaos, soliton and fractals* **24** (1) (2005) 33–43. ⇒284
[40] R. S. Mackay, J. D. Meiss, I. C. Shepelyanski Transport in Hamiltonian systems, *Physica* **13D** (1984) 55. ⇒293
[41] V. Madhok, V. Gupta, D. A. Trottier, S. Ghose, Signatures of chaos in the dynamics of quantum discord, *Phys. Rev. E* **91** (2015) 032906. ⇒285, 291
[42] V. Madhok, S. Dogra, A. Lakshminarayan, Quantum correlations as probes of chaos and ergodicity *Opt. Commun.***420**(2018) 189. ⇒284
[43] M. T. Martin, A. Plastino, O. A. Rosso, Statistical complexity and disequilibrium, *Physics Letters A* **311** (2003) 126–132. ⇒284, 286, 287
[44] M. T. Martin, A. Plastino, O. A. Rosso, Generalized statistical complexity measures: Geometrical and analytical properties, *Physica A* **369** (2006) 439–462. ⇒288
[45] P. A. Miller, S. Sarkar, Signatures of chaos in the entanglement of two coupled quantum kicked tops *Phys. Rev. E* **60** (1999) 1542. ⇒284
[46] H. Ming-Lian, X. Xiao-Qiang, Mixedness of the N-qubit states with exchange symmetry *Chinese Physics B* **17,** 10 (2008) 3559. doi:10.1088/1674-1056/17/10/006 ⇒290
[47] C. Neill, P. Roushan, M. Fang, Y. Chen, M. Kolodrubetz, Z. Chen, A. Megrant, R. Barends, B. Campbell, B. Chiaro et al., Ergodic dynamics and thermalization in an isolated quantum system *Nat. Phys.* **12** (2016) 1037–1041. doi:10.1038/nphys3830 ⇒285, 291
[48] A. Piga, M. Lewenstein, J. Q. Quach Quantum chaos and entanglement in ergodic and nonergodic systems, *Phys. Rev. E* **99** (2019) 032213. ⇒284
[49] A. R. Plastino, A. Plastino, Symmetries of the Fokker-Plank equation and Fisher-Frieden arrow of time, *Phys. Rev. E* **54** (1996) 4423–4326. ⇒289

[50] O. A. Rosso, H. A. Larrondo, M. T. Martin, A. Plastino, M. A. Fuentes, Distinguishing noise from chaos, *Phys. Rev. Lett.* **99** (2007) 154102. doi:10.1103/PhysRevLett.99.154102 ⇒289

[51] O. A. Rosso, L. De Micco, H. A. Larrondo, M. T. Martin, A. Plastino, Generalized statistical complexity measure, *Int. J. Bif. Chaos* **20** (2010) 775—785. doi:10.1142/S021812741002606X ⇒286, 289

[52] J. B. Ruebeck, J. Lin, and A. K. Pattanayak, Entanglement and its relationship to classical dynamics *Phys. Rev. E* **95** (2017)062222. ⇒284, 285

[53] C.E. Shannon, The Mathematical Theory of Communication, *Bell System Technical Journal*, **27** (1948) 379–423, 623–656. ⇒286

[54] S.J. Shenker, L.P. Kadanoff Critical behavior of a KAM surface: I. Empirical results *J. Stat. Phys.* **27** (1982) 631. ⇒293

[55] J.S. Shiner, M. Davison, P.T. Landsberg, Simple measure for complexity, *Phys. Rev. E* **59**(2)(1999)1459–1464. ⇒284, 288

[56] J.S. Shiner, M. Davison,P.T Landsberg, Replay to comments on: simple measure for complexity. *Phys. Rev. E* **62** (2000) 3000–3003. ⇒288

[57] G. Stamatiou and D. P. K. Ghikas, Quantum entanglement dependence on bifurcations and scars in non-autonomous systems. The case of quantum kicked top *Phys. Lett. A* **368** (2007) 206. ⇒284

[58] C. Tsallis, Possible generalization of Boltzmann-Gibbs statistics, *J. Stat. Phys.* **52** (1988) 479. ⇒284

[59] R. Wackerbauer, R.A. Witt, H. Atmanspacher, J. Kurths, H. Scheingraber, A comparative classification of complexity-measures. *Chaos Solitons Fractals* **4** (1994) 133–173. ⇒284

[60] X. Wang, S. Ghose, B. C. Sanders, and B. Hu Entanglement as a signature of quantum chaos *Phys. Rev. E* **70** (2004) 016217. ⇒284

[61] W.K. Wootters, Statistical distance and Hilbert space, *Phys. Rev. D* **23** (1981) 357. ⇒284, 287

[62] R. Zarum and S. Sarkar Quantum-classical correspondence of entropy contours in the transition to chaos *Phys. Rev. E* **57** (1998) 5467. ⇒285

Received: November 8, 2020 • Revised: November 20, 2020

Acta Univ. Sapientiae, Informatica **12,** 2 (2020) 302–324

DOI: 10.2478/ausi-2020-0018

A review on suppressed fuzzy c-means clustering models

László SZILÁGYI
Sapientia Hungarian University of Transylvania,
Cluj-Napoca, Romania
Dept. of Electrical Engineering, Târgu Mureş
Óbuda University, Budapest, Hungary
University Research, Innovation and Service Center
email: lalo@ms.sapientia.ro
szilagyi.laszlo@nik.uni-obuda.hu

László LEFKOVITS
Sapientia Hungarian University of Transylvania,
Cluj-Napoca, Romania
Dept. of Electrical Engineering, Târgu Mureş
email: lefkolaci@ms.sapientia.ro

David ICLANZAN
Sapientia Hungarian University of Transylvania,
Cluj-Napoca, Romania
Dept. of Mathematics-Informatics, Târgu Mureş
email: iclanzan@ms.sapientia.ro

Abstract. Suppressed fuzzy c-means clustering was proposed as an attempt to combine the better properties of hard and fuzzy c-means clustering, namely the quicker convergence of the former and the finer partition quality of the latter. In the meantime, it became much more than that. Its competitive behavior was revealed, based on which it received two generalization schemes. It was found a close relative of the so-called fuzzy

Computing Classification System 1998: I.5.1, I.5.3
Mathematics Subject Classification 2010: 62H30
Key words and phrases: fuzzy c-means algorithm, suppressed fuzzy c-means algorithm, image segmentation, data mining

c-means algorithm with generalized improved partition, which could improve its popularity due to the existence of an objective function it optimizes. Using certain suppression rules, it was found more accurate and efficient than the conventional fuzzy c-means in several, mostly image processing applications. This paper reviews the most relevant extensions and generalizations added to the theory of fuzzy c-means clustering models with suppressed partitions, and summarizes the practical advances these algorithms can offer.

1 Introduction

C-means clustering algorithms represent a subset of the objective function optimizer clustering methods, which group a set of object data into a set of predefined number of clusters. Chronologically, the first c-means clustering algorithm is hard c-means (HCM) having its origins in the works of Steinhaus [34] and McQueen [23], also known as k-means, which uses bivalent (crisp) logic to represent the created partition, namely it assigns each object to a single cluster. The fuzzy c-means (FCM) algorithm was introduced by Dunn [7], and generalized by Bezdek [2]. FCM uses a probabilistic partition to create the clusters: for any object, the sum of the fuzzy memberships with respect to all clusters is always one.

Both HCM and FCM have certain limitations. HCM converges quickly but it is very sensitive to initialization [1], and frequently gives mediocre partitions because it gets stuck in local minima of the objective function. On the other hand, FCM has a slower convergence, which becomes a problem when the input data is huge. Despite these limitations, HCM and FCM are very popular algorithms, having lots of applications in various research domains.

Several solutions have been proposed to reduce the runtime of the FCM algorithm, without damaging the quality of the provided partition. Early solutions generally turned to data approximation, e.g. Cannon [3] et al. and Kamel et al. [18] implemented FCM using only computations on integer values. Cheng et al. [6] deployed a random sampling of the input data, thus achieving a fast approximative FCM clustering. Later, data reduction schemes were introduced, aggregating similar input data before proceeding to clustering. Eschrich et al. [8] accelerated FCM this way by an order of magnitude. Data aggregation was also employed in image segmentation: clustering gray intensity levels instead of individual pixel intensities can speed up FCM by up to two orders of magnitude [29]. Szilágyi et al. [31] extended this pixel aggregation scheme to color images, thus achieving an efficient color reduction procedure. Alternately, Lázaro et

al. [21] proposed a parallel hardware implementation to FCM, and deployed it successfully in signal processing. Kolen and Hutcheson [20] proposed an FCM implementation that does not need to store the partition matrix, which is a relevant step toward clustering unloadable amounts of data. Further remarkable FCM solutions specialized for clustering huge data sets were introduced by Hathaway and Bezdek [12], and Havens et al. [13].

FCM has a main governing parameter called fuzzy exponent, usually denoted by m, and generally constrained by $m > 1$. The value of m has a strong impact on the fuzzyness of the created partition, and on the convergence speed as well. Large values of m reduce the ability of FCM to distinguish the input data: above a certain limit value all clusters merge together at the grand mean of the input data. However, this limit value is unknown, it strongly depends on the data. If m approaches its lower limit, the fuzzy partition tends toward the crisp one. In the limit case $m \to 1_+$, FCM becomes HCM.

Let us consider an FCM algorithm that uses fuzzy exponent m_0. If we do not like its convergence speed and the partition it makes, we may reduce the fuzzy exponent to m (where $1 < m < m_0$), which is a step towards the behavior of the HCM algorithm. The algorithm we obtain with this change is still FCM. The suppressed fuzzy c-means (s-FCM) algorithm, introduced by Fan et al. [9] in 2003, also makes a step towards HCM determined by the so-called suppression rate $\alpha \in [0, 1]$, but a different way, without staying in the bounds of the FCM algorithm. The s-FCM proved to converge in less iterations than FCM when used with the same fuzzy exponent m, and it provided fine partitions in all tested cases. However, the authors left several questions open, including (1) how to choose the value of the suppression rate α, or (2) is s-FCM an optimal algorithm? Since its introduction, the theory of the s-FCM algorithm evolved a lot, some of the open questions were answered and further open questions emerged. For example, Szilágyi et al. [28] explained the effect of the partition suppression in a comparative study with competitive learning [19], based on which later they introduced several generalized suppression rules for the fuzzy partition [30]. Several other works [15, 16, 17, 24, 27, 36] proposed minor modifications of the original s-FCM algorithm, providing parameter selection schemes for the suppression rate, and successfully applying s-FCM in various image processing tasks.

This paper proposes to provide an inventory of the theoretical advances regarding the s-FCM algorithm, and the successful applications that emerged since its introduction. The rest of this paper is structured as follows: Section 2 enumerates the foundations of the s-FCM algorithm, the c-means clustering models s-FCM relies on. Section 3 presents the details of the original s-FCM

algorithm, analyses its competitive behavior, and shows some of its generalization schemes. Section 4 explores the relation between suppressed c-means clustering models and the so-called FCM with generalized improved partition, giving some hints about the optimality of suppressed FCM clustering models. Section 5 relates on suppression parameter setting techniques found in the literature. Section 6 discusses the advantages and disadvantages of suppressed FCM clustering algorithms, while Section 7 concludes this study.

2 Background

2.1 The fuzzy and hard c-means algorithms

The conventional c-means clustering algorithms partition a set of object data into a predefined number of c clusters, through minimizing of a quadratic objective function. The objective function of FCM is:

$$J_{FCM} = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} \|\mathbf{x}_k - \mathbf{v}_i\|^2 = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^{m} d_{ik}^2 \ , \tag{1}$$

where

- $\mathbf{x}_k$ stands for the input data ($k = 1, 2, \ldots, n$),
- $\mathbf{v}_i$ represents the prototype (or centroid or representative element) of cluster i ($i = 1, 2, \ldots, c$),
- $u_{ik} \in [0, 1]$ is the fuzzy membership function describing the degree to which input vector $\mathbf{x}_k$ belongs to cluster i,
- $m > 1$ is the fuzzy exponent ($m = 2$ in the version of Dunn [7]),
- and d_{ik} represents the distance between vector $\mathbf{x}_k$ and cluster prototype $\mathbf{v}_i$.

FCM uses a probabilistic partition, meaning that for any input vector $\mathbf{x}_k$ we have

$$\sum_{i=1}^{c} u_{ik} = 1 \ . \tag{2}$$

The objective function J_{FCM} is minimized by alternately applying the optimization of J_{FCM} over $\{u_{ik}\}$ with $\mathbf{v}_i$ fixed, $i = 1, 2, \ldots, c$, and the optimization

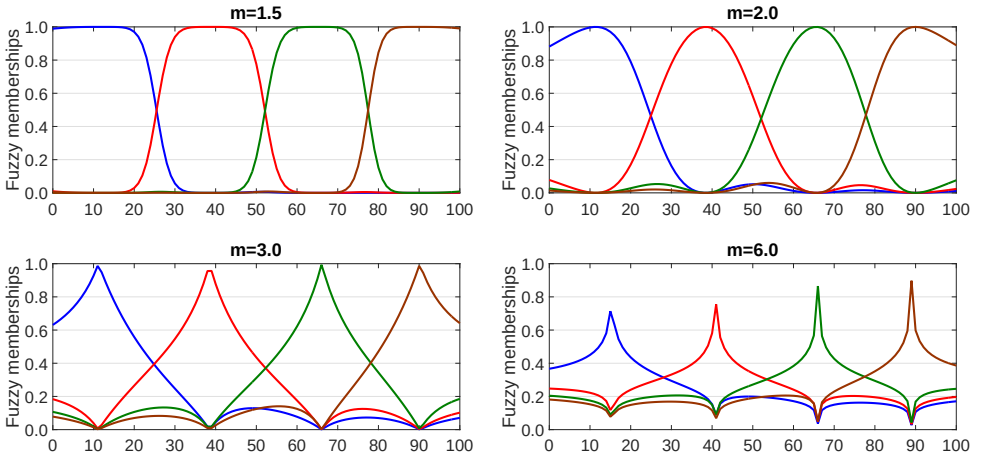


Figure 1: Fuzzy membership functions provided by FCM in a 1D problem, with random integer input values between 0 and 100. In case of $m = 6$, the peak of the membership functions also reach the maximum value 1, but not at integer valued inputs.

of J_{FCM} over $\{\mathbf{v}_i\}$ with u_{ik} fixed, $i = 1, 2, \ldots, c$; $k = 1, 2, \ldots, n$ [2]. The optimization formulas are deduced from the zero gradient conditions of J_{FCM} and Lagrange multipliers, and obtained as follows:

$$u_{ik}^{\star} = \frac{d_{ik}^{-2/(m-1)}}{\sum_{j=1}^{c} d_{jk}^{-2/(m-1)}} \qquad \begin{array}{l} \forall\, i = 1, 2, \ldots, c \\ \forall\, k = 1, 2, \ldots, n \end{array} , \tag{3}$$

$$\mathbf{v}_i^{\star} = \frac{\sum_{k=1}^{n} u_{ik}^{m} \mathbf{x}_k}{\sum_{k=1}^{n} u_{ik}^{m}} \qquad \forall\, i = 1, 2, \ldots, c \ . \tag{4}$$

According to the optimization scheme of the FCM, Eqs. (3) and (4) are alternately applied, until cluster prototypes converge.

HCM is a limit case of FCM, which uses $m \to 1_+$, and thus the memberships are obtained by the winner-takes-all rule:

$$u_{ik}^{\star} = \begin{cases} 1 & \text{if } i = \arg\min_j \{d_{jk}, j = 1, \ldots, c\} \\ 0 & \text{otherwise} \end{cases} . \tag{5}$$

The propotype of each cluster in HCM is the average of the input vectors assigned to it, according to Eq. (4) using $m = 1$.

2.2 FCM versions with improved partition

An undesired property of the fuzzy memberships provided by FCM is their multimodality. Figure 1 presents some membership functions obtained by FCM in a single dimension problem with $c = 4$ clusters at various values of the fuzzy exponent. Each of the four fuzzy membership functions has a maximum at the cluster prototype, having the value of 1. Because of the probabilistic constraint, all other fuzzy membership functions are zero at these points. But, for example the fuzzy membership function represented in blue can have elevated values for input data situated very far from the cluster prototype. This is the multimodality, which gets stronger as the value of the fuzzy exponent m grows. So it is obvious that the multimodality gets suppressed if we reduce the value of m. Alternately, intending to suppress the multimodality, Höppner and Klawonn [14] proposed the so-called FCM with improved partition (IFP-FCM), which adds a rewarding term to the objective function of FCM:

$$J_{\text{IFP–FCM}} = \sum_{i=1}^{c}\sum_{k=1}^{n} \mu_{ik}^{m} d_{ik}^{2} - \sum_{k=1}^{n} a_k \sum_{i=1}^{c} (\mu_{ik} - 1/2)^2 \ , \tag{6}$$

where parameters denoted by a_k are supposed to be positive numbers. The second term pushes the fuzzy membership values u_{ik}, $i = 1, 2, \ldots, c$; $k = 1, 2, \ldots, n$ towards the limits situated at 0 and 1, while preserving the probabilistic constraint. A generalized version of this algorithm (GIFP-FCM) was introduced by Zhu et al. [40], by replacing the rewarding term as follows:

$$J_{\text{GIFP–FCM}} = \sum_{i=1}^{c}\sum_{k=1}^{n} \mu_{ik}^{m} d_{ik}^{2} + \sum_{k=1}^{n} a_k \sum_{i=1}^{c} \mu_{ik}(1 - \mu_{ik}^{m-1}) \ . \tag{7}$$

The partition update formula derived by zero gradient conditions of the objective function is

$$\mu_{ik}^{\star} = \frac{(d_{ik}^{2} - a_k)^{-1/(m-1)}}{\sum_{j=1}^{c} (d_{jk}^{2} - a_k)^{-1/(m-1)}} \qquad \begin{matrix} \forall\, i = 1, 2, \ldots, c \\ \forall\, k = 1, 2, \ldots, n \end{matrix} \ . \tag{8}$$

Fuzzy membership functions obtained with Eq. (8) can be interpreted as FCM partition memberships obtained by applying virtually reduced distances between cluster prototypes and input data. Each distance d_{ik} is replaced by

$\delta_{ik} = \sqrt{d_{ik}^2 - \alpha_k}$. The authors also proposed a formula for α_k:

$$\alpha_k = \omega \min_i \{d_{ik}^2, i = 1, 2, \ldots, c\} \ , \tag{9}$$

where $\omega \in [0.9, 0.99]$. Setting $\omega = 1$ would reduce GIFP-FCM to HCM, while $\omega = 0$ to FCM.

It is important to remark that both of the above improved clustering models kept FCM's prototype update formula given in Eq. (4), but applied to fuzzy membership functions μ_{ik} instead of u_{ik}, as described in Eq. (11).

Lately, these improved FCM clustering models were involved in several applications [4, 5].

3 The suppressed FCM algorithm

3.1 The original suppressed FCM

The suppressed fuzzy c-means algorithm was introduced by Fan et al. [9], declaring the goal to propose an algorithm with better convergence speed and reduced execution time than FCM, but providing the same partition quality. The s-FCM algorithm manipulates the optimization scheme of FCM, by inserting an extra step in each iteration, between partition updating via Eq. (3) and prototype updating via Eq. (4). This new step deforms the partition given by FCM according to the following rule:

$$\mu_{ik} = \begin{cases} 1 - \alpha + \alpha u_{ik} & \text{if } i = \arg\max_j \{u_{jk}\} \\ \alpha u_{ik} & \text{otherwise} \end{cases} \ , \tag{10}$$

where μ_{ik} ($i = 1, 2, \ldots, c$; $k = 1, 2, \ldots, n$) represents the fuzzy memberships obtained after suppression. The cluster prototype update formula of s-FCM becomes:

$$\mathbf{v}_i^\star = \frac{\sum_{k=1}^{n} \mu_{ik}^m \mathbf{x}_k}{\sum_{k=1}^{n} \mu_{ik}^m} \qquad \forall\, i = 1, 2, \ldots, c \ . \tag{11}$$

Just like in case of competitive clustering [19], s-FCM sets up a competition for each input vector $\mathbf{x}_k$ in each iteration, which is won by the cluster whose prototype is situated at shortest distance from $\mathbf{x}_k$. Fuzzy memberships of $\mathbf{x}_k$ with respect to any non-winner cluster is proportionally suppressed ($\mu_{ik} = \alpha u_{ik}$), while all suppressed parts are given to the winner cluster to preserve

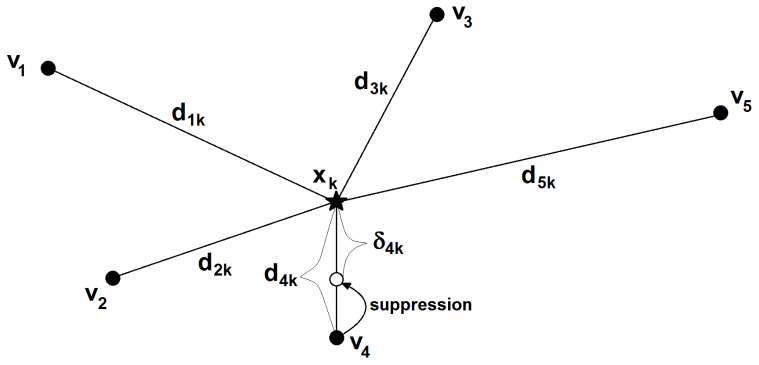


Figure 2: Shortened distance caused by suppression.

the probabilistic constraint: $\mu_{wk} = 1 - \alpha + \alpha u_{wk}$. Let w stand for the winner class index in the current competition for input vector $\mathbf{x}_k$. Actually it should be denoted by w_k, as it is specific to input vector $\mathbf{x}_k$, but for the sake of simplicity, the index of w is neglected in all formulas.

In the original article that introduced s-FCM, Fan et al. did not give any recipe or hint how to choose a suppression rate that would be optimal in any sense, or suitable for general or any specific purpose. They set the suppression rate to the middle of its range ($\alpha = 0.5$). Fan et al. validated s-FCM with some toy problems and found it insensitive to the fuzzy exponent m [9].

3.2 Suppression, regarded as a competition

Szilágyi et al. [28] showed that the proportional suppression of the FCM partition, the multiplication of all non-winner fuzzy memberships with a suppression rate $\alpha \in [0, 1]$, is mathematically equivalent with a reduction of the distance between the input vector and the closest cluster prototype. In any stage of the algorithm at partition updating, for any input vector $\mathbf{x}_k$ and its winner class with index w, there exists a virtually reduced distance $\delta_{wk} < d_{wk}$, which fulfils all partition update formulas of s-FCM, namely:

$$\mu_{wk} = \frac{\delta_{wk}^{\frac{-2}{m-1}}}{\delta_{wk}^{\frac{-2}{m-1}} + \sum_{j=1, j\neq w}^{c} d_{jk}^{\frac{-2}{m-1}}} = 1 - \alpha + \alpha \frac{d_{wk}^{\frac{-2}{m-1}}}{\sum_{j=1}^{c} d_{jk}^{\frac{-2}{m-1}}} \tag{12}$$

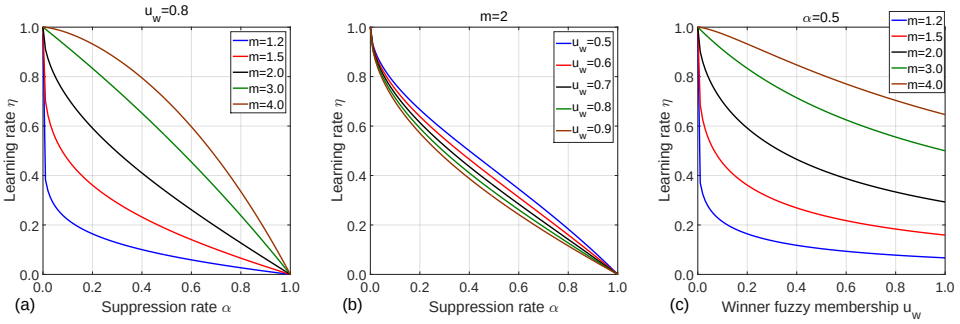


Figure 3: Some characteristics of the s-FCM algorithm: (a) learning rate η plotted against suppression rate α in case of $u_w = 0.8$, at various constant values of fuzzy exponent m; (b) learning rate η plotted against suppression rate α in case of $m = 2$, at various constant values of winner fuzzy membership u_w; (c) learning rate η plotted against winner fuzzy membership u_w in case of $\alpha = 0.5$, at various constant values of fuzzy exponent m.

and

$$\mu_{ik} = \frac{d_{ik}^{\frac{-2}{m-1}}}{\delta_{wk}^{\frac{-2}{m-1}} + \sum_{j=1, j\neq w}^{c} d_{jk}^{\frac{-2}{m-1}}} = \alpha \frac{d_{ik}^{\frac{-2}{m-1}}}{\sum_{j=1}^{c} d_{jk}^{\frac{-2}{m-1}}} \qquad \forall i \neq w \ . \tag{13}$$

This virtual distance reduction is exhibited in Fig. 2. For the sake of clarity we need to remark that although the competition among clusters seems to be the same as in case of conventional competitive algorithms, these algorithms radically differ in the sense that conventional competitive algorithms do not work with quadratic objective functions [26].

Szilágyi et al. [28] also defined a quasi learning rate η of the s-FCM algorithm, in a similar way to the learning rate of competitive algorithms, and deduced its formula:

$$\eta(m, \alpha, u_{wk}) \equiv 1 - \frac{\delta_{wk}}{d_{wk}} = 1 - \left(1 + \frac{1-\alpha}{\alpha u_{wk}}\right)^{\frac{1-m}{2}} \ , \tag{14}$$

where u_{wk} stands for the winner fuzzy membership value of vector $\mathbf{x}_k$, obtained without suppression. The learning rate η depends on both parameters of the s-FCM algorithm (fuzzy exponent m and suppression rate α), but also on the winner fuzzy membership function (u_{wk}) of the given input vector $\mathbf{x}_k$. This

means that η can hardly be a constant in any s-FCM scenario, unless defined as such.

3.3 Generalized suppressed FCM algorithm

The FCM algorithm with generalized suppression (gs-FCM) was introduced by Szilágyi et al. [30]. The s-FCM algorithm, as proposed by Fan et al. [9], uses a constant suppression rate α. There are three possible ways to give the suppression rate some variation:

1. *Time variant suppression* means to employ a suppression rate α_t that changes according to the iteration counter t. This sort of variation was used for example by Hung et al. [15].

2. *Context sensitive* or *data sensitive suppression* means to introduce time invariant rules of suppression, which provide dedicated suppression rate α_k for each input vector $\mathbf{x}_k$, depending on the current distances d_{ik}, $k = 1, 2, \ldots, n$.

3. *Time and context variant suppression* means to combine both previous variation versions in a unique suppression rule.

Szilágyi et al. [30] focused on context sensitive suppression rules only. Aiming at achieving quicker convergence they did not consider changing the suppression rule in every iteration. Their suppression rules were define according to two different schemes, presented in the following sections.

3.3.1 Learning rate defined as a function of the winner fuzzy membership

The first generalization scheme of the s-FCM algorithm uses a learning rate defined as a function of winner fuzzy membership u_w: $\eta = f(u_w)$, where $f : [0, 1] \to [0, 1]$ is a continuous function. Using Eq. (14), the context dependent suppression rate generally becomes

$$\alpha_k = \left[1 - u_w + u_w(1 - f(u_w))^{\frac{2}{1-m}}\right]^{-1} . \tag{15}$$

Some special cases using the above definition are:

- θ-type gs-FCM (gs_θ-FCM) that uses constant learning rate $\eta = f(u_w) = \theta$ with parameter $\theta \in [0, 1]$, which leads to suppression rate

$$\alpha_k = \left[1 - u_w + u_w(1 - \theta)^{\frac{2}{1-m}}\right]^{-1} , \tag{16}$$

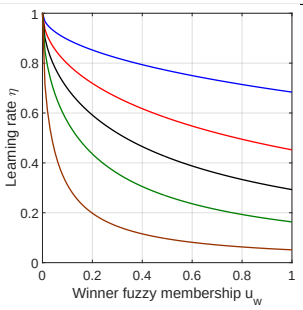
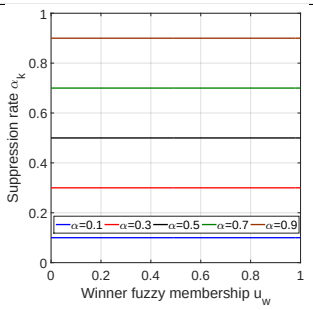
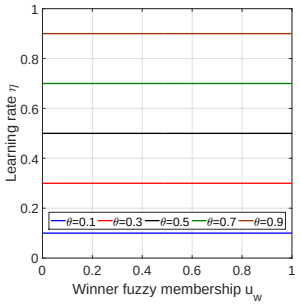
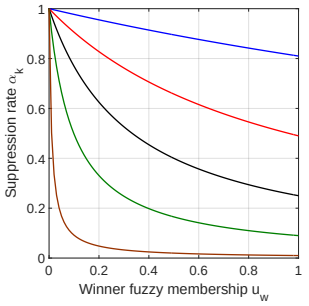
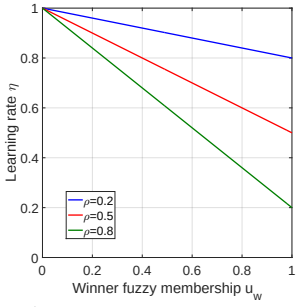
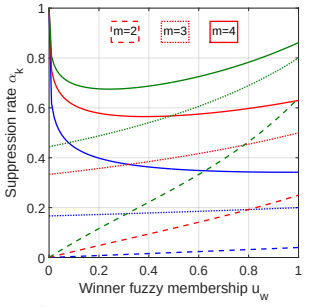
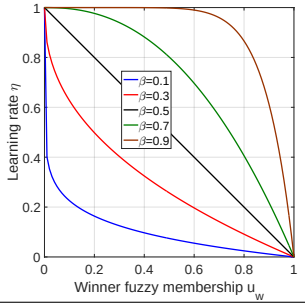
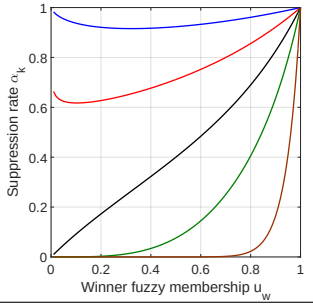
Definition	η vs. u_w	α_k vs. u_w
Original s − FCM $\alpha_k = \alpha$ $\alpha \in [0, 1]$		
gs_θ−FCM $\eta = \theta$ $\theta \in [0, 1]$		
gs_ρ−FCM $\eta = 1 - \rho u_w$ $\rho \in [0, 1]$		
gs_β−FCM $\eta = 1 - u_w^{\beta/(1-\beta)}$ $\beta \in [0, 1)$		

Table 1: Original s-FCM and generalized s-FCM algorithms defined with the first scheme ($\eta = f(u_w)$): definitions and context dependent suppression rates.

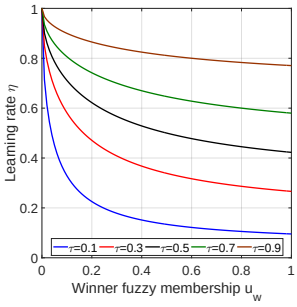
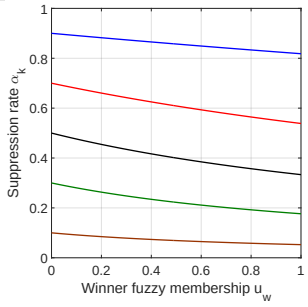
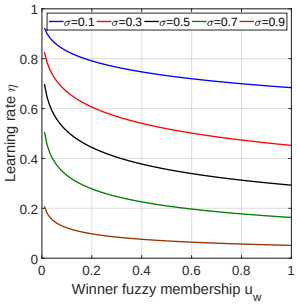
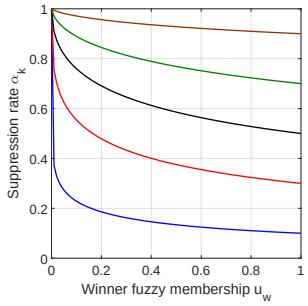
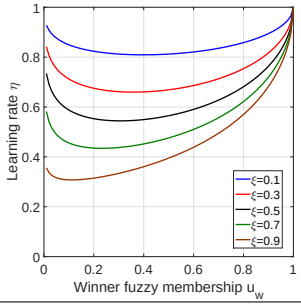
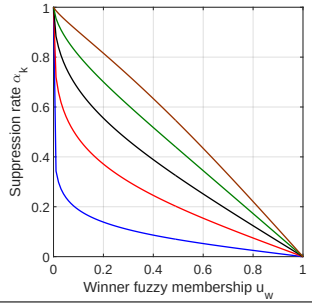
Definition	η vs. u_w	α_k vs. u_w
gs_τ–FCM $\mu_w = \frac{u_w+\tau}{1+u_w\tau}$ $\tau \in [0,1]$		
gs_σ–FCM $\mu_w = u_w^\sigma$ $\sigma \in [0,1]$		
gs_ξ–FCM $\mu_w = (\sin\frac{\pi u_w}{2})^\xi$ $\xi \in [0,1]$		

Table 2: Generalized s-FCM algorithms defined with the second scheme ($\mu_w = g(u_w)$): learning rates and context dependent suppression rates.

- ρ-type gs-FCM (gs_ρ-FCM) that uses learning rate linearly decreasing with the winner fuzzy membership $\eta = f(u_w) = 1 - \rho u_w$ with parameter $\rho \in [0,1]$, which leads to suppression rate

$$\alpha_k = \left[1 - u_w + \rho \frac{2}{1-m} u_w^{\frac{3-m}{1-m}}\right]^{-1}, \tag{17}$$

- β-type gs-FCM (gs_β-FCM) that uses learning rate decreasing with the winner fuzzy membership according to the exponential rule $\eta = f(u_w) = 1 - u_w^{\frac{\beta}{1-\beta}}$ with parameter $\beta \in [0, 1)$, which leads to suppression rate

$$\alpha_k = \left[1 + u_w \left(u_w^{\frac{2\beta}{(1-m)(1-\beta)}} - 1\right)\right]^{-1} \quad . \tag{18}$$

3.3.2 Direct formula between μ_w and u_w

The second generalization scheme is defined by a direct formula between the winner fuzzy membership values before and after suppression. In a general form, it is defined as $\mu_w = g(u_w)$ with $g : [0, 1] \to [0, 1]$ and $g(x) \geq x$ $\forall x \in [1/c, 1]$. Using Eq. (14), the context dependent suppression rate generally becomes

$$\alpha_k = \frac{1 - g(u_w)}{1 - u_w} \qquad \forall u_w \in [1/c, 1) \ . \tag{19}$$

For the special case when $u_w = 1$, the suppression rate is irrelevant, as non-winner memberships are zero valued, so there is nothing to suppress. Some special cases using the above definition are:

- τ-type gs-FCM (gs_τ-FCM) that is inspired by the relativistic speed addition formula $\mu_w = (u_w + \tau)/(1 + u_w\tau)$ with parameter $\tau \in [0, 1]$, which leads to suppression rate

$$\alpha_k = \frac{1 - \tau}{1 + u_w\tau} \qquad \forall u_w \in [1/c, 1] \ , \tag{20}$$

- σ-type gs-FCM (gs_σ-FCM) that uses the relation $\mu_w = u_w^\sigma$ with parameter $\sigma \in [0, 1]$, which leads to suppression rate

$$\alpha_k = \frac{1 - u_w^\sigma}{1 - u_w} \qquad \forall u_w \in [1/c, 1) \ , \tag{21}$$

- ξ-type gs-FCM (gs_ξ-FCM) that uses learning rate decreasing with the winner fuzzy membership according to the rule $\mu_w = (\sin \pi u_w/2)^\xi$ with parameter $\xi \in [0, 1]$, which leads to suppression rate

$$\alpha_k = \frac{1 - \left(\sin \frac{\pi u_w}{2}\right)^\xi}{1 - u_w} \qquad \forall u_w \in [1/c, 1) \ . \tag{22}$$

Algorithm 1: The gs-FCM algorithm

Data: Input data $\mathbf{x}_k$, $k = 1, 2, \ldots, n$
Data: Fuzzy exponent $m > 1$, suppression scheme and parameter, limit ε
Result: Cluster prototypes $\mathbf{v}_i$, $i = 1, 2, \ldots, c$
`Initialize cluster prototypes` $\mathbf{v}_i$, $i = 1, 2, \ldots, c$ `with random input vectors,` $\mathbf{v}_i \neq \mathbf{v}_j \ \forall i \neq j$
repeat
  for $k = 1, 2, \ldots, n$ **do**
    `Compute fuzzy memberships` u_{ik}, $i = 1, 2, \ldots, c$ `with Eq. (3).`
    `Compute suppression rate` α_k, `according to the chosen suppression scheme and parameter value with one of the Eqs. (16)-(22).`
    `Compute suppressed fuzzy memberships` μ_{ik}, $i = 1, 2, \ldots, c$ `with Eq. (10), using suppression rate` α_k.
  end
  for $i = 1, 2, \ldots, c$ **do**
    $\mathbf{v}_i^{(\mathrm{old})} \leftarrow \mathbf{v}_i$
    `Update cluster prototype` $\mathbf{v}_i$ `with Eq. (11).`
  end
until $\sum_{i=1}^{c} \|\mathbf{v}_i^{(\mathrm{old})} - \mathbf{v}_i\| < \varepsilon$;

3.3.3 The gs-FCM algorithm

The previous sections presented six generalized suppression rules, each regulated by a suppression parameter that can take an infinite number of different values. A limited number of these parameter values (up to two) reduce the generated algorithm to either FCM or HCM, while all other value define new clustering algorithms, different from HCM and FCM, or the original s-FCM. The suppression rules introduced above are summarized in Tables 1 and 2. Anyone can define further suppression rules by following the recipe given in Sections 3.3.1 and 3.3.2, by proposing a function $\eta = f(u_w)$ or $\mu_w = g(u_w)$, different from the ones exhibited in Tables 1 and 2. The gs-FCM algorithm is summarized in Algorithm 1.

4 The relation between s-FCM and GIFP-FCM clustering models

Now let us investigate the similarities and differences in the main loop of gs-FCM variants (let us call in this section the original s-FCM a variant of gs-FCM) and GIFP-FCM:

- All gs-FCM variants and GIFP-FCM use the same formula to update the cluster prototypes, given in Eq. (11).
- Compared to the original FCM, all gs-FCM variants and GIFP-FCM use modified d_{ik} distances. GIFP-FCM changes all distances by subtracting the same amount from the square of all d_{ik} values, $i = 1, 2, \ldots, c$, while gs-FCM variants reduce only the shortest distance d_{wk} to $\delta_{wk} = d_{wk}(1 - \eta_k)$, where η_k is the learning rate applied to vector x_k.
- GIFP-FCM makes two optimal steps in each loop, by executing its partition update and cluster prototype update formula. However, it changes the cost function in every loop by establishing the winner cluster for each vector x_k and adjusting the a_k values accordingly. Szilágyi [32] showed that gs-FCM variants can act the same way, the only difference is in the a_k terms, which are changed to s_{ik} and thus made dependent on cluster index i.

Szilágyi [32] introduced a unification theory for gs-FCM variants and GIFP-FCM. The new clustering model has the objective function very similar to the one of GIFP-FCM, but the so-called rewarding term, now denoted by s_{ik}, has a double indexing.

$$J_U = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m d_{ik}^2 + \sum_{k=1}^{n} s_{ik} \sum_{i=1}^{c} u_{ik}(1 - u_{ik}^{m-1}) \ . \tag{23}$$

Obviously, we can make this clustering model act like GIFP-FCM by setting $s_{ik} = a_k \ \forall i = 1, 2, \ldots, c$, where a_k is the rewarding term of GIFP-FCM defined in Eq. (9).

On the other hand, if we wish this new clustering model act like a certain gs-FCM variant, it is necessary to set:

$$s_{ik} = \begin{cases} d_{wk}^2(1 - \eta_k^2) & \text{if } i = w \equiv \arg\min_j \{d_{jk}, j = 1, \ldots, c\} \\ 0 & \text{otherwise} \end{cases} \ , \tag{24}$$

Algorithm	Parameter	Formula of s_{wk}
s-FCM	$\alpha \in [0,1]$	$d_{wk}^2 \left[1 - \left(\frac{\alpha u_w}{1-\alpha+\alpha u_w}\right)^{m-1}\right]$
gs_θ-FCM	$\theta \in [0,1]$	$d_{wk}^2 \theta(2-\theta)$
gs_ρ-FCM	$\rho \in [0,1]$	$d_{wk}^2 (1-\rho^2 u_w^2)$
gs_β-FCM	$\beta \in [0,1)$	$d_{wk}^2 \left(1 - u_w^{2\beta/(1-\beta)}\right)$
gs_τ-FCM	$\tau \in [0,1]$	$d_{wk}^2 \left[1 - \left(\frac{u_w(1-\tau)}{u_w+\tau}\right)^{m-1}\right]$
gs_σ-FCM	$\sigma \in [0,1]$	$\begin{cases} d_{wk}^2 \left[1 - \left(\frac{u_w - u_w^{\sigma+1}}{u_w^{\sigma} - u_w^{\sigma+1}}\right)^{m-1}\right] & \text{if } u_w < 1 \\ 0 & \text{if } u_w = 1 \end{cases}$
gs_ξ-FCM	$\xi \in [0,1]$	$\begin{cases} d_{wk}^2 \left[1 - \left(\frac{u_w}{1-u_w}((\sin\frac{\pi u_w}{2})^{-\xi} - 1)\right)^{m-1}\right] & \text{if } u_w < 1 \\ 0 & \text{if } u_w = 1 \end{cases}$

Table 3: The definition of s_{wk} rewarding term for gs-FCM algorithm variants, u_w stands for the highest fuzzy membership provided by FCM for the input vector $\mathbf{x}_k$.

where η_k represents the learning rate applied to the current vector $\mathbf{x}_k$ according to the chosen suppression scheme, suppression parameter, and d_{ik} distances with $i = 1, 2, \ldots, c$. Table 3 exhibits s_{wk} rewarding terms for various gs-FCM algorithms.

Consequently we can affirm that s-FCM and gs-FCM are optimal algorithms to the same extent as GIFP-FCM, as they all optimize J_U. The partition update formula of the unified clustering model is:

$$\mu_{ik}^{\star} = \frac{(d_{ik}^2 - s_{ik})^{-1/(m-1)}}{\sum_{j=1}^{c} (d_{jk}^2 - s_{jk})^{-1/(m-1)}} \qquad \begin{matrix} \forall\, i = 1, 2, \ldots, c \\ \forall\, k = 1, 2, \ldots, n \end{matrix}, \tag{25}$$

while the cluster prototype update formula is the one given in Eq. (11). The unified clustering algorithm that integrates all suppressed clustering models and the GIFP-FCM algorithm is exhibited in Algorithm 2. This version of the algorithm is not the recommended one to implement GIFP-FCM or gs-FCM model. It is only the proof of their similar structure. GIFP-FCM runs optimally as described in [40], while gs-FCM variants as described in Algorithm 1.

Algorithm 2: The unified algorithm

Data: Input data $\mathbf{x}_k$, $k = 1, 2, \ldots, n$
Data: Fuzzy exponent $m > 1$, limit ε, the chosen algorithm and its parameter (one of $\{\alpha, \theta, \rho, \beta, \tau, \sigma, \xi, \omega\}$)
Result: Cluster prototypes $\mathbf{v}_i$, $i = 1, 2, \ldots, c$
Initialize cluster prototypes $\mathbf{v}_i$, $i = 1, 2, \ldots, c$ with random input vectors, $\mathbf{v}_i \neq \mathbf{v}_j \ \forall i \neq j$
repeat
  for $k = 1, 2, \ldots, n$ **do**
    Find the index of the winner cluster $w = \arg\min_i \{d_{ik}, i = 1, 2, \ldots, c\}$.
    Compute fuzzy memberships u_{ik}, $i = 1, 2, \ldots, c$ with Eq. (3).
    if *algorithm is GIFP-FCM* **then**
      $s_{ik} \leftarrow \omega d_{wk}^2$, $i = 1, 2, \ldots, c$
    end
    else
      Set s_{ik} values according to Eq. (24) and Table 3.
    end
    Compute suppressed fuzzy memberships μ_{ik}, $i = 1, 2, \ldots, c$ with Eq. (25).
  end
  for $i = 1, 2, \ldots, c$ **do**
    $\mathbf{v}_i^{(\mathrm{old})} \leftarrow \mathbf{v}_i$
    Update cluster prototype $\mathbf{v}_i$ with Eq. (11).
  end
until $\sum_{i=1}^{c} \|\mathbf{v}_i^{(\mathrm{old})} - \mathbf{v}_i\| < \varepsilon$;

5 Parameter selection

Fan et al. [9] did not give any recipe how to choose the value of the suppression rate α. They set the suppression rate to the middle of its definition interval, to be well in between HCM and FCM. Several further works, including the gs-FCM clustering models of Szilágyi et al. [28, 30], set the parameter values experimentally and showed that there are various settings that make s-FCM and gs-FCM models work better than FCM or HCM in various applications.

However, there are some application papers [15, 16, 17, 24, 27, 36] in the literature that give recipes for the choice of the suppression parameter.

5.1 Constant suppression rate based on input data

Fan et al. [10] proposed a constant suppression rate based on the distribution of the input data, defined as

$$\alpha = \frac{\sum_{j=1}^{n} \sum_{r=1}^{n} \|\mathbf{x}_j - \mathbf{x}_r\|}{n \sum_{j=1}^{n} \|\mathbf{x}_j - \bar{\mathbf{x}}\|} - 1 \ , \tag{26}$$

where $\bar{\mathbf{x}}$ stands for the grand mean of the input vectors: $\bar{\mathbf{x}} = n^{-1} \sum_{i=1}^{n} \mathbf{x}_i$. The authors proved that the value of α defined by Eq. (26) is always in the interval $[0, 1]$. The value of α should be evaluated once as an initialization step of the algorithm, and applied as constant compression rate through all optimization loops. Obviously, this only works with the original s-FCM algorithm, not with gs-FCM models.

5.2 Time variant suppression rate based on partition entropy

Li et al. [22] proposed a time variant suppression rate based on the entropy of the partition provided by the FCM algorithm, defined with the formula

$$\alpha_{\mathrm{Li}} = \frac{1}{\log c} \left(-\frac{1}{n} \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik} \log(u_{ik}) \right) \ . \tag{27}$$

This formula is evaluated in every optimization loop, after having applied the partition update formula given in Eq. (3) and before starting the partition suppression using Eq. (10). The authors found their method successful in image segmentation problems, despite this entropy based suppression rate fully suppresses the crisp partition and applies no change in the completely ambiguous situation described by $u_{ik} = 1/c \ \forall i = 1, 2, \ldots, c$ and $\forall k = 1, 2, \ldots, n$. In our opinion, it would be more useful setting the suppression rate to $1 - \alpha_{\mathrm{Li}}$.

5.3 Time variant suppression rate based on current cluster prototypes

There is a set of works that apply time variant suppression rate, which changes from iteration to iteration according to the current cluster prototypes. As

prototypes converge, the suppression rate also stabilizes. The foundation of all these recipes is the formula of the Xie-Beni cluster validity index [38]:

$$XB = \frac{\sum_{i=1}^{c}\sum_{k=1}^{n} u_{ik}^2 \|\mathbf{x}_k - \mathbf{v}_i\|^2}{n\left(\min_{1\le i\ne j\le c}\|\mathbf{v}_i - \mathbf{v}_j\|^2\right)} \quad , \tag{28}$$

which indicates fine cluster quality at low values of XB. Knowing that well separable clusters are best partitioned by HCM, while FCM handles overlapping clusters better, it seems a good idea to apply stronger suppression when the minimum distance between cluster prototypes is higher. In this order, Hung et al. [15] proposed a time variant suppression rate defined as

$$\alpha = \exp\left(-\frac{1}{\beta}\min_{1\le i\ne j\le c}\|\mathbf{v}_i - \mathbf{v}_j\|^2\right) \quad , \tag{29}$$

where

$$\beta = \frac{1}{n}\sum_{j=1}^{n}\|\mathbf{x}_j - \bar{\mathbf{x}}\|^2 \quad , \tag{30}$$

and $\bar{\mathbf{x}} = n^{-1}\sum_{i=1}^{n}\mathbf{x}_i$ is the grand mean of input vectors. In the application of Hung et al. [15], α is evaluated at the beginning of each optimization loop, and applied to suppress the fuzzy memberships provided by the FCM partition update formula. This suppression was found successful in an ophthalmology image segmentation problem, similarly to the alternative one introduced by the same authors in [16]:

$$\alpha = \left(1 + \min_{1\le i\ne j\le c}\frac{\|\mathbf{v}_i - \mathbf{v}_j\|^2}{\beta}\right)^{-1} \quad , \tag{31}$$

that also uses the formula of β given in Eq. (30). Tsai et al. [36] also introduced a kernel-based suppressed FCM version, using the suppression formula derived from Eq. (29).

A very similar formulation of the suppression rate formula was given by Nyma et al. [24]:

$$\alpha = \exp\left(-\min_{1\le i\ne j\le c}\frac{\|\mathbf{v}_i - \mathbf{v}_j\|^2}{m}\right) \quad , \tag{32}$$

where m is the fuzzy exponent. In spite of being applied successfully in medical image segmentation by the authors, we find this an ill-posed formula, as it advises a different suppression rate if we replace all input vectors by a constant $\kappa \ne 1$.

6 Discussion

All fuzzy c-means algorithms with suppressed or improved partitions use an extra parameter compared to FCM, which regulates the alteration of the FCM partition. In case of the GIFP-FCM algorithm we cannot talk about suppression, but the effect is similar: although the parameter ω is recommended to be chosen from the interval $[0.9, 0.99]$, we need to remark that $\omega = 1$ reduces the GIFP-FCM to HCM, while $\omega = 0$ means fully FCM behavior for GIFP-FCM. Some of the suppression schemes, namely the original s-FCM and the gs-FCM algorithm of type σ act like HCM when the value of the parameter (α, σ) is 0, and as FCM at the other extremum 1. On the other hand, the gs-FCM algorithms of type θ, β, τ act like FCM when the value of the parameter (θ, β, τ) is 0, and as HCM at the other extremum 1. Any other value of the suppression parameters defines a clustering model that is different from HCM or FCM of any fuzzy exponent $m > 1$.

Suppressed FCM clustering algorithms have a moderate popularity, because they are not optimal, as they do not minimize the objective function of FCM or any other known objective function. The unification theory with the GIFP-FCM algorithm revealed that all suppressed clustering models can be considered optimal to the same extent as GIFP-FCM. Despite this disadvantage, several applications showed that s-FCM and gs-FCM clustering models can perform better than HCM or FCM, and run in less time than FCM. Although the quality of the clustering outcome should be characterized by cluster validity indexes (CVI), several image processing applications showed that suppressed FCM clustering models can capture better the underlying structure of the input data than FCM or HCM, leading to better segmentation quality. Szilágyi et al. [30] employed the Xie-Beni [38], the extended Xie-Beni [25], and the Fukuyama-Sugeno [11] CVIs to prove the ability of gs-FCM models to produce valid partitions. The authors also showed that suppressed FCM clustering is substantially less sensitive to imbalanced cluster sizes. Hung et al. [15, 16] deployed suppressed FCM in an ophthalmologic MRI image segmentation problem and found it more effective than the classical FCM. Zhao et al. [39] reported improved image processing accuracy achieved via suppressing the FCM partition in a general purpose image processing environment. Improvement in performance against the clock achieved via suppressing the FCM partition were reported by Szilágyi et al. [31, 33] in a color reduction application. In a very recent paper, Wu et al. [37] combined the suppressed FCM with the so-called picture fuzzy clustering method [35] that has type

II fuzzy background. The resulting clustering model performed better than previous methods in terms of both accuracy and efficiency.

7 Conclusion

This paper presents the short history of the suppressed fuzzy c-means algorithm, focusing on the most important theoretical advances and providing a short summary of practical achievements. Applying suppressed partitions in clustering models derived from fuzzy c-means currently have a moderate popularity, which may rise in the future due to the recent successful extensions and applications.

Acknowledgements

This study was supported in part by the Institute for Research Programs of Sapientia University. The work of L. Szilágyi was supported by the ÚNKP 20-5 New National Excellence Program of the Ministry of Human Capacities of Hungary, contract no. OE-RH, 1109/3, 2020. L. Szilágyi is Bolyai Research Fellow of the Hungarian Academy of Sciences.

References

[1] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA USA*, 2007, pp. 1027–1035. ⇒303

[2] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*, Plenum, New York (1981) ⇒303, 306

[3] R. L. Cannon, J. V. Dave, J. C. Bezdek, Efficient implementation of the fuzzy c-means clustering algorithms, *IEEE Trans. Pattern Anal. Mach. Intell.* **8** (1986) 248–255. ⇒303

[4] A. Celikyilmaz, I. B. Türkşen, Enhanced fuzzy system models with improved fuzzy clustering algorithm, *IEEE Trans. Fuzzy Syst.* **16** (2008) 779–794. ⇒308

[5] A. Celikyilmaz, I. B. Türkşen, R., Aktaş, M. M. Doganay, N. B. Ceylan, Increasing accuracy of two-class pattern recognition with enhanced fuzzy functions, *Expert Syst. Appl.* **36** (2009) 1337–1354. ⇒308

[6] T. W. Cheng, D. B. Goldgof, L. O. Hall, Fast fuzzy clustering, *Fuzzy Sets Syst.* **93** (1998) 49–56. ⇒303

[7] J. C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters, *J. Cybern.* **3** (1974) 32–57. ⇒303, 305

[8] S. Eschrich, J. Ke, L. O. Hall, D. B. Goldgof, Fast accurate fuzzy clustering through data reduction, *IEEE Trans. Fuzzy Syst.* **11** (2003) 262–270 ⇒303

[9] J. L. Fan, W. Z. Zhen, W. X. Xie, Suppressed fuzzy c-means clustering algorithm. *Patt. Recogn. Lett.* **24** (2003) 1607–1612. ⇒304, 308, 309, 311, 318

[10] J. L. Fan, J. Li, A fixed suppressed rate selection method for suppressed fuzzy c-means clustering algorithm, *Appl. Math.* **5** (2014) 1275–1283. ⇒319

[11] Y. Fukuyama, M. Sugeno, A new method of choosing the number of clusters for the fuzzy c-means method (in Japanese), *Proc. 5th Fuzzy Systems Symposium*, Japan, 1989, pp. 247–250. ⇒321

[12] R. J. Hathaway, J. C. Bezdek, Extending fuzzy and probabilistic clustering to very large data sets. *Comput. Stat. Data Anal.* **51** (2006) 215–234. ⇒304

[13] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, M. Palaniswami, Fuzzy c-means algorithms for very large data, *IEEE Trans. Fuzzy Syst.* **20** (2012) 1130–1146. ⇒304

[14] F. Höppner, F. Klawonn, Improved fuzzy partition for fuzzy regression models, *Int. J. Approx. Reason.* **5** (2003) 599–613. ⇒307

[15] W. L. Hung, M. S. Yang, D. H. Chen, Parameter selection for suppressed fuzzy c-means with an application to MRI segmentation, *Patt. Recogn. Lett.* **27** (2006) 424–438. ⇒304, 311, 319, 320, 321

[16] W. L. Hung, Y. C. Chang, A modified fuzzy c-means algorithm for differentiation in MRI of ophtalmology, *Int'l Conference on Modeling Decisions for Artificial Intelligence*, Tarragona, Spain, *LNCS* **3885** (2006) 340–350. ⇒304, 319, 320, 321

[17] W. L. Hung, D. H. Chen, M. S. Yang, Suppressed fuzzy-soft learning vector quantization for MRI segmentation, *Artif. Intell. Med.* **52** (2011) 33–43. ⇒304, 319

[18] M. S. Kamel, S. Z. Selim, New algorithms for solving the fuzzy clustering problem, *Patt. Recogn.* **27** (1994) 421–428. ⇒303

[19] T. Kohonen, The self-organizing map, *Proc. IEEE* **78** (1990) 1474–1480. ⇒304, 308

[20] J. F. Kolen, T. Hutcheson, Reducing the time complexity of the fuzzy c-means algorithm. *IEEE Trans. Fuzzy Syst.* **10** (2002) 263–267. ⇒304

[21] J. Lázaro, J. Arias, J. L. Martín, C. Cuadrado, A. Astarloa, Implementation of a modified fuzzy c-means clustering algorithm for real-time applications, *Microproc. & Microsyst.* **29** (2005) 375–380. ⇒304

[22] J. Li, J. L Fan, Parameter selection for suppressed fuzzy c-means clustering algorithm based on fuzzy partition entropy, *11th Int. Conf. on Fuzzy Systems and Knowledge Discovery*, Xiamen, China, 2014, pp. 82–87. ⇒319

[23] S. McQueen, Some methods for classification and analysis of multivariate observations, *Proc. 5th Berkeley Symp. Math. Stat. Probab.*, 1967, pp. 281–297. ⇒303

[24] A. Nyma, M. Kang, Y. K. Kwon, C. H. Kim, J. M. Kim, A hybrid technique for medical image segmentation, *J. Biomed. Biotechnol.* **2012** (2012) 830252. ⇒304, 319, 320

[25] N. R. Pal, J. C. Bezdek, On cluster validity for the fuzzy c-means model, *IEEE Trans. Fuzzy Syst.* **3** (1995) 370–379. ⇒321

[26] N. R. Pal, J. C. Bezdek, R. J. Hathaway, Sequential competitive learning and the fuzzy c-means clustering algorithms, *Neural Networks* **9** (1996) 787–796. ⇒ 310

[27] M. F. Saad, A. M. Alimi, Improved modified suppressed fuzzy c-meanss, *Proc. 2nd Int'l Conference on Image Processing Theory, Tools and Applications*, Paris, 2010, pp. 313–318. ⇒304, 319

[28] L. Szilágyi, S. M. Szilágyi, Z. Benyó, Analytical and numerical evaluation of the suppressed fuzzy c-means algorithm: a study on the competition in c-means clustering models, *Soft. Comput.* **14** (2010) 495–505. ⇒304, 309, 310, 318

[29] L. Szilágyi, S. M. Szilágyi, B. Benyó, Efficient inhomogeneity compensation using fuzzy c-means clustering models, *Comput. Meth. Prog. Biol.* **108** (2012) 80–89. ⇒303

[30] L. Szilágyi, S. M. Szilágyi, Generalization rules for the suppressed fuzzy c-means clustering algorithm, *Neurocomput.* **139** (2014) 298–309. ⇒304, 311, 318, 321

[31] L. Szilágyi, G. Dénesi, S. M. Szilágyi, Fast color reduction using approximative c-means clustering models, *Proc. IEEE Int. Conference on Fuzzy Systems*, Beijing, 2014, pp. 194–201. ⇒303, 321

[32] L. Szilágyi, A unified theory of fuzzy c-means clustering models with improved partition, *Proc. Int'l Conference on Modeling Decisions for Artificial Intelligence*, Skövde, Sweden, *LNCS* **9321** (2015) 129–140. ⇒316

[33] L. Szilágyi, G. Dénesi, C. Enăchescu, Fast color quantization via fuzzy clustering, *Proc. 23rd International Conference on Neural Information Processing*, Kyoto, 2016, *LNCS* 9950 (2016) 95–103. ⇒321

[34] H. Steinhaus, Sur la division des corps matériels en parties, *Bulletin de l'Académie Polonaise des Sciences* **C1** III.(IV) (1956) 801–804. ⇒303

[35] P. H. Thong, L. H. Son, Picture fuzzy clustering: a new computational intelligence method, *Soft. Comput.* **20** (2016) 3549–3562. ⇒321

[36] H. S. Tsai, W. L. Hung, M. S. Yang, A robust kernel-based fuzzy c-means algorithm by incorporating suppressed and magnified membership for MRI image segmentation, *Proc. Int'l Conference on Artificial Intelligence and Computational Intelligence*, Chengdu, China, *LNCS* 7530 (2012) 744–754. ⇒ 304, 319, 320

[37] C. M. Wu, N. Liu, Suppressed robust picture fuzzy clustering for image segmentation, *Soft. Comput.* doi: 10.1007/s00500-020-05403-8. ⇒321

[38] X. L. Xie, G. A. Beni, Validity measure for fuzzy clustering, *IEEE Trans. Pattern Anal. Mach. Intell.* **3** (1991) 841–846. ⇒320, 321

[39] F. Zhao, J. L. Fan, H. Q. Liu, Optimal-selection-based suppressed fuzzy c-means clustering algorithm with self-tuning non local spatial information for image segmentation, *Expert Syst. Appl.* **41** (2014) 4083–4093. ⇒321

[40] L. Zhu, F. L. Chung, S. Wang, Generalized fuzzy c-means clustering algorithm with improved fuzzy partitions, *IEEE Trans. Syst. Man Cybern. B.* **39** (2009) 578–591. ⇒307, 317

Received: 15 November 2020 • Revised: 21 November 2020

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae Informatica

Each volume contains two issues.

Sapientia University
Sciendo by De Gruyter
Scientia Publishing House

ISSN 1844-6086

http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae Informatica publishes peer-reviewed, high-quality original papers and surveys in English, in all areas of Computer Science and allied fields. The papers must be self-contained, including the purpose of the research, the basic definitions, the antecedents of the research, the presentation of the state-of-the-art and novel contributions. The articles must contain sufficient references to give the reader a broad view of the studied subject. Articles must follow the technical requirements described in the Instruction for Authors. Please note, that if an article fails to do so, it will be rejected without peer-review. The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities or institutions, if needed, for publication, the Editorial Board is disclaiming any responsibility.

The papers have to be prepared carefully, and must include the relevant keywords, and the appropriate ACM Computing Classification System (http://www.acm.org/about/class/1998) and AMS Mathematics Subject Classification codes (http://www.ams.org/msc/). References should be listed alphabetically based on the Instruction for Authors given at the address http://www.acta.sapientia.ro.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: http://www.acta.sapientia.ro.

Submission must be made by email (acta-inf@acta.sapientia.ro) only, using the LaTeX style and sample file at the address http://www.acta.sapientia.ro. Beside the LaTeX source a pdf format of the paper is necessary too.

Contact address and subscription:
Acta Universitatis Sapientiae Informatica
RO 400112 Cluj-Napoca, Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by F&F INTERNATIONAL
Director: Enikő Ambrus

Supported by:

ISSN 1844-6086
http://www.acta.sapientia.ro