SOLUTION OF SELECTED PROBLEMS USING IBM QX

Ján KOLLÁR, Vojtech FLORKO

Abstract: This paper presents a set of works that were chosen to be decomposed to individual algorithms and solved using a quantum computer, which is available publicly through the IBM QX platform. The objective was to demonstrate a fundamental change in the way of thinking about problems when working with a quantum computer as opposed to solving a computational problem on a classical computer. Although a novelty for most people today, the way of approach that is explored in this paper might become a valuable skill one day, when quantum computers become more wide-spread.

Keywords: Quantum Computer; Qubit; IBM QX; Quantum Circuit; Quantum Gate.

1 INTRODUCTION

Quantum computer is surprisingly old as a concept, more specifically it comes from 1959, when the underlying idea was presented as a concept of computational machine for physicists.

Recent developments and headlines have launched it as a popular subject in the realm of information technologies, but in reality it is principles and fundamental ways of operating it are mostly unknown to the mainstream software development community.

As of today, there is a number of public application programming interfaces available on the internet, which offer access to a quantum computer, so at the moment there is a huge opportunity to broaden the horizons of what can be done using this technology in the future, once it becomes more ubiquitous. There are a few well known algorithms available today which demonstrate the advantages of quantum computers. A lot of these algorithms show promise of quantum computers based on certain criteria, such as performance gains we receive during the program execution, but the advancement of development of these machines is based very much on how the quality of the software catches up with the limited hardware we have available today.

To this end, we have prepared a set of algorithms that show a potential way of choosing certain types of problems and developing their solutions on quantum computers. To amend the broad lack of simple practical examples, we have chosen to experiment on the most advanced way of accessing a quantum computer at the moment, the IBM QX platform. This platform offers a set of public quantum computers, where anyone with an internet connection is capable of running their own quantum programs [1].

All of the algorithms were executed on a 5 qubit ibmqx4 quantum computer. These were developed and tested on a web graphical interface that allows composition of quantum circuits and executed as a part of desktop programs with a quantum component.

2 MOTIVATION

As with all emerging technologies, to empower it is use, there have to exist more applications that can be incorporated in a practical manner into our daily world. Since the types of algorithms and the way quantum computers work are notably different from a classical computer, there is a need to invent ways of how to leap from a well established way of algorithmic thinking to this new mindset.

This paper above all else aims to present a select few results that show various techniques of thinking when approaching a development of new algorithms for a quantum computer, as opposed to algorithms on a classical von neumann computer.

3 RELEVANT WORKS

The most important notion that needs to be considered when exploring the potential for a new quantum algorithm is whether the quantum computer offers a significant advantage over abundantly more cost effective classical computer implementation.

There are several of well known and established algorithms that leverage the quantum computers in varous ways. One such algorithm for example is Grover's algorithm, which Coles et al. describe in their paper about quantum algorithms implementation. As a matter of fact, Grover's search algorithm, which enables to find a spefic item in a randomly ordered database, provides a notable speedup over a classical computer. The quantum implementation would require $O(\sqrt{N})$ operations, while the classical computer would do this in $O(N)$ operations [2][3]. The speedup provided in factorization incorporated in the famous Shor's algorithm is also a similar type of application [4].

As Valiron mentions in his work, the real advantage and gain we strive for with quantum algorithms is for example when it leverages the quantum phenomena, namely entanglement [5], which Horodecki et al. describe in how important role it has played in development of quantum computing, including measurement based schemes [6] as well as superposition, with Sarma noting that these phenomena can be used to process data [7]. When implementing a mapping for a boolean truth table on a quantum circuit, he mentions two ways one can go about implementing a tangible logical state on a set of quantum bits, the naïve one using a direct mapping, where we map subparts of the object correspondingly to the number of quantum bits we have,

or a compositional implementation, where the efficiency is taken into account. There are of course, some limitations as well in the current IBM QX hardware offering, which makes the creation of new algorithms more complicated. In the works of Zulehner and DiVincenzo we can see how the two-qubit CNOT gates cannot be arbitrarily placed in the architecture but are restricted to dedicated pairs of qubits only [8] [9]. In addition, Wooton et al. describe in their paper how there is a need not only for devices with more quantum bits, but also the ability to achieve fault-tolerance as well. To this end, there is a need for algorithms that have been specifically designed to work on small and noisy devices [10]. The way of assembling an algorithm to be executed on IBM QX was mentioned to be a quantum circuit, which is a timeline of quantum gates (general types of quantum operations) by Nielsen et al. [11].

The path to invention of new practical quantum algorithms is definitely rooted in experimenting and this can be seen in a work of Pathak, where he describes the development of their implementation of a quantum random number generator and what it means for cryptographic purposes [12]. Like Pathak, we have been tinkering with random number generators independently ourselves. Laforest provides evidence of this noting existing stable quantum cryptography systems which are already to some extent used by government and banks [13]. Biamonte et al. describe how the search for quantum machine learning algorithms strive to gain a speedup compared to their classical couterparts [14]. In the paper of Avaliani, another topic related to true randomness is simulations, which are important for developing applications for chemistry and biology [15]. This further confirms what Valiron mentions, that the search for quantum optimizations over classical algorithms is a vibrant area [4].

4 RESULTS

Quantum computers are not oriented to evaluate and present the results to an end user in a user friendly fashion at this moment. Therefore, when striving to create quantum programs as practical applications, the quantum algorithm should form a component of a larger program running and evaluating the results on a classical computer.

As previously mentioned, the big challenge with using quantum computers is choosing an appropriate problem or a part of a problem that fits the way quantum calculations work. Rather than focusing on well documented types of problems, we have chosen at first to attempt to assemble a set of simple and very common operations. One such type are operations with strings of characters. Attempting to perform these on a quantum computer shows the type of problems a quantum computer is not good at.

The first operation has a task of reverting a string at the input. Each letter is encoded by a numerical index value before being sent to a quantum circuit and thus is reverted purely numerically. A sample four letter word is encoded by a sequence of indices 0123. Since quantum computers return individual result values within an interval of <0; 1>, each number must be modified by a factor of string length:

$$factor = \frac{lengt}{10}, \quad index^* = \frac{index}{factor}.$$

If we have a string with a length of 5 characters and we invert a letter under index 3, we have a factor of *0.5* and index* of *6*. We then execute a quantum circuit with the following results:

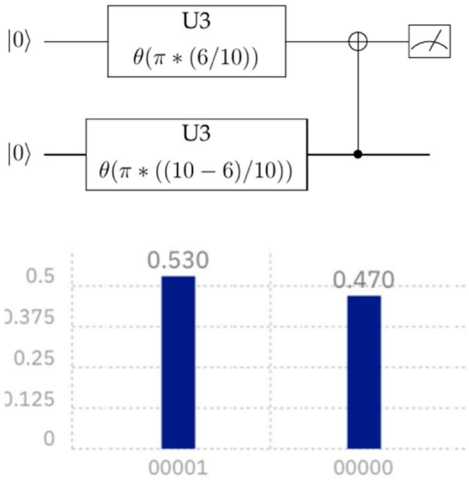

Fig. 1 Quantum circuit performing a revertion process using a modified index
Source: authors.

If our value index* is higher than 5, we are interested in the lower result of the two and vice versa. Since we have a bit of imprecision going on, we round the values by ceiling the value which is higher than *0.5* and flooring the lower value. We then multiply the result with our factor and get the reverted index, in our case it is *2*. We do one quantum operation per index and assemble the word on a classical computer.

The second experiment we have decided to perform is also a common string operation, which is getting a character from a string using a specific index or a list of indices at the input. In this case, the task was to encode each letter on one quantum bit. Since we receive the results as individual numerical values per quantum bit, we need to assign a unique number to each letter of the word.

Here, an interesting experimental phenomena occurred when choosing numbers for the characters. When we set a quantum bit by rotating it is theta angle, we expect that exact particular number at the results, which is how we identify the chosen character. In the higher range of numbers, e.g. higher than 0.8, we receive relatively precise results with

minimal variation. The lower we go, the higher the spread of possible values we get, which is not what we are looking to leverage for this experiment.

Again, we are using the 5 qubit quantum computer, and bearing in mind the aforementioned manifestation, we encode the sample four character string ABCD using the following values:

A = 0.95, B = 0.9, C = 0.85, D = 0.8. We map the values by encoding the first four quantum bits by a theta rotation using the U3 quantum gate available at the IBM QX. Each quantum bit performs a CNOT operation with the controlling quantum bit, which uses the X gate to determine the chosen characters at the output. While the mapping at the first four quantum bits is the same for any four character string, the distribution of X gates in between the CNOT operations determines the input of the chatacter indices. We then rotate each theta backwards tonullify the values of uncaptured quantum states.

Fig. 2 shows us the circuit in place when choosing the first three letters at the output. Had we set the second X gate one CNOT operation backwards, we would have received only the first two letters at the output. If we had set it at the very end of the quantum bit (or not at all), we would have received the result comprising of all four letters. There is a possibility to add multiple X gates as well, if we had set the first two gates around the first CNOT operation and the second pair of X gates around the last operation, we would have received only the first and the last character codes at the output.

When we run the circuit illustrated in Fig. 2, we receive the in Fig. 3.

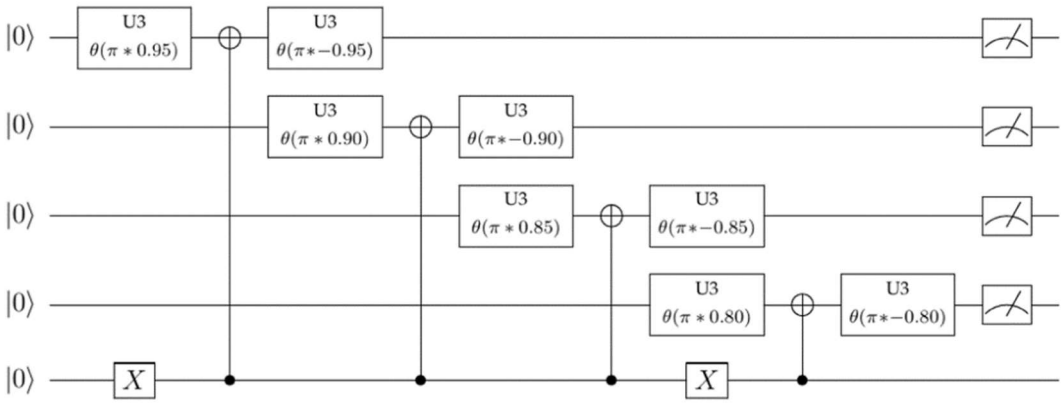

Fig. 2 Quantum circuit performing a mapping of a four character string, while choosing the first three characters at the output. Note: the phi and lambda rotation angles are set to 0, only the relevant first parameter theta is shown to make the circuit more concise
Source: authors.

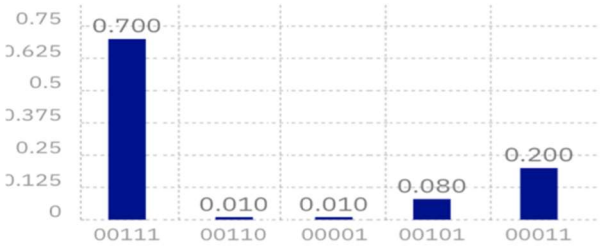

Fig. 3 Results from running a quantum circuit
Source: authors.

Right away we can see that the fourth quantum bit is not contained in the results, which is exactly how this circuit was supposed to run. We then proceed with adding the results up for each represented quantum bit:

$$q[0] = 0.7 + 0.01 + 0.08 + 0.2 = 0.99$$
$$q[1] = 0.7 + 0.01 + 0.2 = 0.91$$
$$q[2] = 0.7 + 0.01 + 0.08 = 0.79$$

The results come from a quantum simulation, where we can see that while the value 0.95 was received as 0.99 and the value 0.95 was received as 0.91, the 0.85 expected at the output turned into 0.79. Running the circuit on a quantum computer would provide more precise results, but at the moment there is not an available quantum chip at IBM QX, where four quantum bits can perform a CNOT to a particular designated controlling bit. Performing this operation with two quantum bits, ergo two character string, shows that this algorithm works properly.

Another type of algorithm that has shown potential for using a quantum computer, is optimizing a route through a graph/map, for example, looking for the shortest route. Operations of this kind have a number of real world applications, from logistics to biology. If we successfully map a particular graph onto the circuit, we can leverage the significant parallelism quantum computers provide. A classical computer algorithm would recursively search all the different variations of possible subroutes between

two chosen points. The challenge that presents itself is how do we represent the different subroutes that can occur in the search by encoding them on the quantum bits.

The natural choice might of course be the simplest one, by attempting to map the graph in a naïve manner, in other words, imprinting the whole graph onto a circuit. It of course depends how we can encode the individual routes. The mapping in Fig. 2 is one of the ways we have tried, because each quantum bit rotation is essentially a graph node, where the CNOT operations are the subroutes. The values encoded by the rotations are lengths of the route coming from the node. This mapping is of course very ineffective and sluggish, because we waste quantum bits with each node and the method does not capture any branching situations from nodes.

A better solution would be to map the routes directly in a way that the circuit would encompass crossroads on the way. Let us consider the following simple graph:

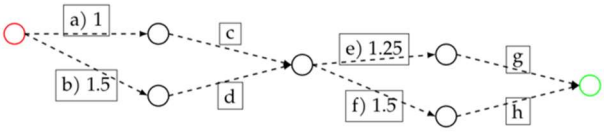

Fig. 4 A simple graph with two crossroads, routes c), d), g) and h) are of same lengths
Source: authors.

Naturally, this particular graph is easy to solve on a classical computer, but we are trying to construct a method on how we can solve any graph with a reasonable size on quantum computer. The way of how we resolve individual crossroads is more effective compared to the previous attempt, because we solve one crossroad on one quantum bit, provided the crossroad has two outbound pathways. The way this is done is we chain two theta rotations to effectively perform a comparison of individual values, which represent the lengths of the compared routes.

The first rotation is by $\pi * (2 - route1)$ and the second rotation by $\pi * route2$, where route1 and route2 are the respective lengths of the outbound paths. If the first value is higher, the rotation shifts into the third quadrant so much, that a lower second value cannot rotate it backwards to revert to the second quadrant and vice versa. After the rotations, we need to apply Hadamard's gate to shift the perspective from the X-axis to the Z-axis, to be able to read the results. For the first crossroad, the solution is as follows:

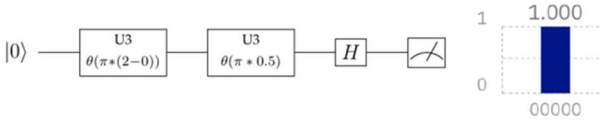

Fig. 5 Results from the circuit solving the first crossroad from the graph Fig. 4
Source: authors.

We have subtracted 1 from both values to prevent unnecessary rotation. The first value is zero, thus we receive 0 at the result. We can then solve both crossroads in a parallel way, where we can see that the second solution is readable from the overall combined result:

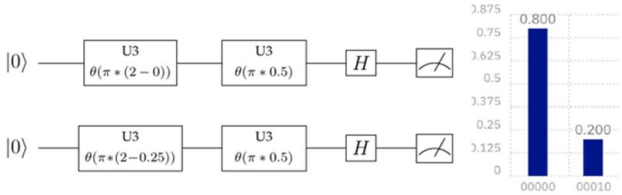

Fig. 6 Results from the circuit solving both crossroads from the graph Fig. 4
Source: authors.

This approach is fine for solving the routes on a per crossroad basis, but since we are not leveraging the remaining quantum bits, we can try a more complete approach where we choose the shortest route with one calculation. Since have four overall routes in this particular graph, we can compare individual subroutes in a total of 6 operations. It is important to note that when assembling this circuit, the lengths of the routes might seem to be are already known. In a practical implementation, the circuit would be assembled dynamically, meaning that the classical computer would only have to assemble all of the combinations of the available subroutes, which would be then fed as a calculation to the interface of the IBM QX.

After evaluation of the results, this circuit identifies the shortest route of all possible routes in the graph. With the currently limited selection and number of quantum bits in quantum computers, an application utilizing this operation can be created in a way that a large graph is broken down to a set of key subgraphs which need to be resolved as a chain in order to determine the optimal route.

To create applications with a quantum component, there is a need for a library that interacts with the IBM QX application programming interface. The official solution is called Qiskit, which provides access to applications developed in the Python programming language. Since we wanted to create a mobile implementation of quantum applications, we have created a port of the library for the iOS platform called QuantumSwift for the Swift programming language.

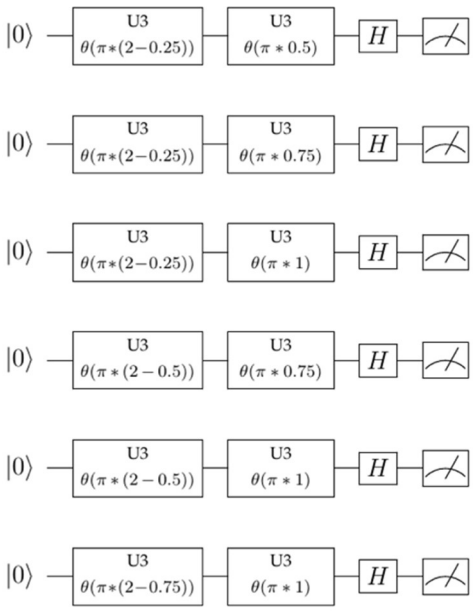

Fig. 7 Circuit which compares individual routes of the graph Fig. 4
Source: authors.

The resulting program is an application of the experiments that is used to determine the shortest path between a choice of two cities in Slovakia on the map (Fig. 8).

5 EVALUATION OF RESULTS

This paper has provided a set of algorithms executed on public IBM QX quantum computers. We have demonstrated that quantum computer can be used in some manner for ordinary operations, such as working with strings. Using quantum computers for string operations is obviously an overkill in terms of cost-benefits perspective, however it is a good practice in terms of evolving one's thinking when developing quantum circuits.

The graph representative algorithms further expanded the ways of approaching a quantum mapping problem from different perspectives. The naïve approach was described but rejected because of it is inefficiency and the granular crossroads and route circuits showed how to extract the key points from a particular problem with relation to encoding them on quantum bits and reading the results.

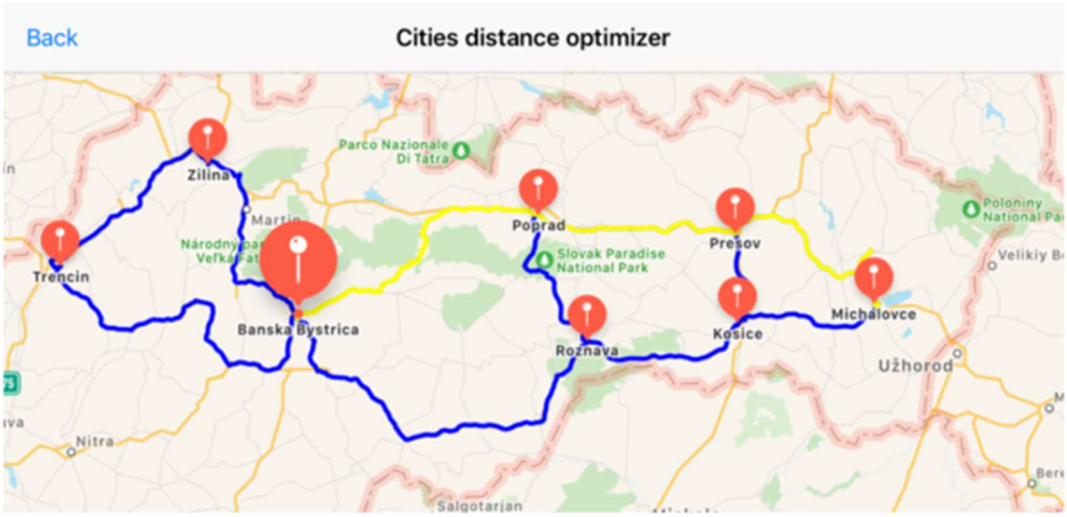

Fig. 8 Mobile application of the optimal route algorithm on a set of major cities in Slovakia
Source: authors.

6 CONCLUSION

The algorithms provided in this paper are by no means aimed for production grade software, however they might be expanded upon in the future to provide real quantum applications of some value, such as some evolved version of the example application in Fig. 8.

Although the search for viable quantum algorithms has been ongoing for decades, the important works are yet to come thanks to unprecedented availability of quantum computers such as IBM QX platform and the popularity of the subject which might potentially provide solutions to important problems using the advantages the quantum computers might provide.

References

[1] FERRARI, D., AMORETTI, M.: Demonstration of Envariance and Parity Learning on the IBM 16 Qubit Processor. In : *CoRR abs/1801.02363,* 2018.

[2] COLES, P. et al.: Quantum Algorithm Implementations for Beginners. (apr 2018)

[3] GROVER, L. K.: A fast quantum mechanical algorithm for database search. Proceedings, 28th ACM Symposium on Theory of Computation, 1996.

[4] SHOR, P.: Algorithms for quantum computation: discrete logarithms and factoring. Proceedings, 35th Annual Symposium on Fundamentals of Computer Science, 1994.

[5] VALIRON, B.: Quantum Computation: a Tutorial. In: *New Generation Comput,* 2012.

[6] HORODECKI, R., HORODECKI, P., HORODECKI, M. and HORODECKI, K.: Quantum entanglement. In: *Reviews of Modern Physics,* Vol. 81, No 2, 2009.

[7] SARMA, K. J.: Understanding Quantum Computing. In: *International Journal of Scientific Engineering and Applied Science (IJSEAS),* Vol. 1, Issue 6, Sept. 2015.

[8] ZULEHNER, A., PALER, A. and WILLE, R.: Ecient Mapping of Quantum Circuits to the IBM QX Architectures. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (dec. 2017)*. 10.1109/TCAD. 2018.2846658.

[9] DiVINCENZO, D. P.: Quantum gates and circuits. *Proceedings: Mathematical, Physical and Engineering Sciences*, 454(1969):261–276, 1998.

[10] WOOTTON, James, R. a LOSS, D. Repetition code of 15 qubits. In: *Phys. Rev. A 97* (5 máj 2018), s. 052313. : 10.1103/ PhysRevA. 97.052313.: Available at: < https://link.aps.org/ doi/10.1103/PhysRevA.97.052313>.

[11] NIELSEN, M. A., CHUANG, I. L.: *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[12] PATHAK, A.: Experimental quantum mechanics in the class room: Testing basic ideas of quantum mechanics and quantum computing using IBM quantum computer. In: *New Generation Comput. 2018.* Available at: https://arxiv.org/pdf/1805.06275.pdf

[13] LAFOREST, M.: *The mathematics of quantum mechanics.* University of Waterloo, 2015.

[14] BIAMONTE, J., WITTEK, P., PANCOTTI, N., REBENTROST, P., WIEBE, N., LLOYD, S.: Quantum machine learning. In: *Nature Insight,* Vol. 549, No. 7671, pp. 195–202 (2017).

[15] AVALIANI, A.: Quantum Computers. In: CoRRcs.AI/0405004 In:(2004) Available at: <http://arxiv.org/abs/cs.AI/0405004>.

Prof. Dipl. Eng. Ján **KOLLÁR**, PhD.
Department of Computers and Informatics
Technical University of Košice
Letná 9
042 00 Košice
Slovak Republic
E-mail: jan.kollar@tuke.sk

Bc. Vojtech **FLORKO**
IBM Slovensko spol. s. r. o.
Krasovského 14
851 01 Bratislava
Slovak Republic
E-mail: Vojtech.Florko1@ibm.com

Ján Kollár – He is Full Professor of Informatics at Department of Computers and Informatics, Technical university of Košice, Slovakia. He received his M.Sc. summa cum laude in 1978 and his Ph.D. in Computer Science in 1991. In 1978-1981 he was with the Institute of Electrical Machines in Košice. In 1982-1991 he was with Institute of Computer Science at the P.J. Šafárik University in Košice. Since 1992 he is with the Department of Computer and Informatics at the Technical University of Košice. In 1985 he spent 3 months in the Joint Institute of Nuclear Research in Dubna, USSR. In 1990 he spent 2 months at the Department of Computer Science at Reading University, UK. He was involved in research projects dealing with realtime systems, the design of microprogramming languages, image processing and remote sensing, dataflow systems, implementation of programming languages, and high performance computing. His research area covers formal languages and automata, functional programming, implementation of programming languages, grammatical language evolution, natural language analysis, adaptive software and language evolution and reflection of languages in quantum systems.

Vojtech Florko – He received his Bachelor‘s degree in Informatics at Department of Computers and Informatics, Technical university of Košice, Slovakia in 2017. He is a student of Master‘s degree in Informatics in the same university. His thesis is based on exploring ways on how to construct algorithms for quantum computers and how to integrate them into tangible computer applications. He is also working as application developer for the IBM company in Košice, Slovakia, being responsible for mobile development practice, frontend and backend technologies as well as cloud and DevOps integration.