AKADÉMIAI KIADÓ

Pollack Periodica • An International Journal for Engineering and Information Sciences

17 (2022) 1, 18–23

DOI: 10.1556/606.2021.00389

ORIGINAL RESEARCH PAPER

A graph-based data quality analysis in distributed telemedicine systems

Zoltán Richárd Jánki*

Department of Software Engineering, University of Szeged, Dugonics tér 13, H-6720 Szeged, Hungary

*Corresponding author. E-mail: jankiz@inf.u-szeged.hu

Received: January 4, 2021 • Revised manuscript received: May 28, 2021 • Accepted: June 8, 2021
Published online: November 1, 2021

ABSTRACT

Telemedicine is one of the most rapidly developing areas of healthcare and it plays an increasing role in modern medicine. As the amount of data and demand for features increase, the data paths are becoming ever-more complex. Owing to this, it is vital in telemedicine to find a proper balance between consistency and availability under any given circumstances. However, making a trade-off can significantly influence the quality of the data. This study seeks to get an in-depth view of the problem by considering a real-world telemedicine use-case and elaborating the formal system specification of the scenario. After evaluating the specification, the constructed state graph is examined using graph coloring and other graph algorithms.

KEYWORDS

telemedicine, distributed systems, state graph, data quality, graph analysis

1. TELEMEDICINE WEB SYSTEMS

Telemedicine is one of the fast developing areas of medicine. Today, more and more data items in a healthcare system are handled electronically, stored in the cloud and can be shared readily with other systems. Telemedicine is a comprehensive area of healthcare that concerns many specialties. Due to them, telemedicine systems have to be designed so as to be easily integrated with other system. There are various techniques available for having secure access to external systems, like standards, Application Programming Interfaces (API), portable server instances. There can also be self-developed parts, caches, Content Delivery Networks (CDN) in a system that can raise the level of availability. Server-side computational units are also frequent because they are responsible for unburdening the client-side by performing resource-intensive tasks. Sometimes the closeness of data is essential, so computation tasks are outsourced to edge devices [1]. Although telemedicine systems are usually viewed as simple client-server architecture-based systems, the reasons and solutions mentioned above can lead to very complex data paths. Figure 1 shows a schematic form of a real-world telemedicine system that shows how complicated the data paths can be in distributed telemedicine systems. However, the set of data portions and aggregations may produce a result table that is visible for a patient or a practitioner.

Since telemedicine applications are mostly web-based, a significant number of requests have to be performed simultaneously on the server-side. However, huge amounts of data and a lot of computational tasks are present. In order to serve so many requests, a distributed system is necessary. Besides the advantages of distributed systems, there are some disadvantages as well. Eric Brewer's theorem about Consistency, Availability and Partition tolerance (CAP) [2] states that there are no distributed systems that can guarantee at most two of the three desirable properties. The extension of the CAP Theorem states that in the case of network Partitioning (P) a trade-off has to be made between Availability (A) and Consistency (C), but Else (E), when the system is running normally in the absence of partitions, another trade-off has to be made between Latency (L) and Consistency (C). This extension is the so-

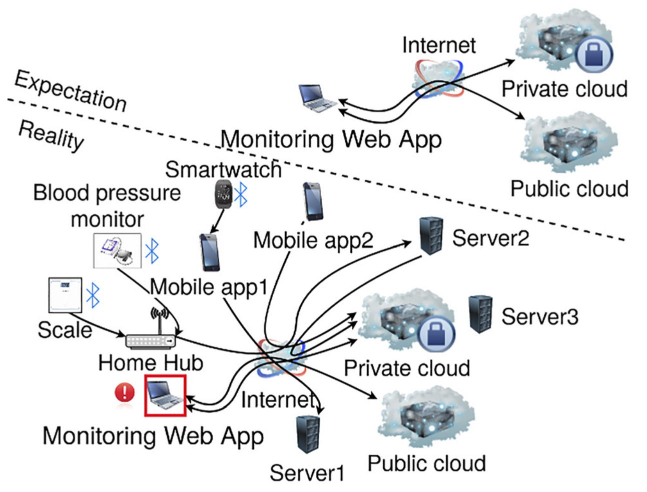

Fig. 1. Expected structure of telemedicine systems versus data paths in real telehealth systems

called PACELC theorem [3]. Both theorems assert that the availability and consistency cannot be guaranteed simultaneously at 100%.

Additionally, the data goes through many stages, so the round-trip and the production of the finally visible data takes some time. These latencies are external factors that are always present in these systems and play important roles. Latency not only strongly influences the availability, but also the consistency and data quality [4].

In this paper, the following results are explained in distributed telemedicine systems.

- The formal modeling of a concrete telemedicine system that operates today and the evaluation of the system specification via model checking: the system specification consists of a client, a distributed database system, computational units and a cache. The correctness of system model is verified with a model checker and the state graph is dumped into graph files;
- Visualizing the new metrics for reliability of data: the model checking results in a state graph file that contains the simulations under different circumstances grouped into graph components. The nodes of the components contain information concerning the Quality of Data (QoD) limited by caching strategies and latency values;
- Analyzing the state graph of the system model with graph theoretical algorithms: the structure of the state graph components contains graph theoretically relevant information, and they can be applied for clustering.

2. SYSTEM MODELING APPROACH

In telemedicine systems, availability and consistency are both important, and it is hard to make a trade-off between them, because different telemedicine use-cases need specific configurations.

It is found that measuring the consistency level in a distributed system is not trivial. Finding the proper metrics and measurement techniques are essential. The Probabilistically Bounded Staleness (PBS) is a promising method that was presented by Peter Bailis et al. [5]. It shows how much time has to elapse for eventual consistency in quorum-based distributed database systems. In their study, t-visibility and k-staleness metrics describe the trade-off between availability and consistency. Operation latency is described with 4 latency values, these are write request to replica, replica write acknowledgement, read request to replica and replica read response latencies. This is the so-called WARS model. Their results were obtained by Monte Carlo simulations, and good approximations can be achieved. However, this approach cannot be used for evaluating whole telemedicine systems.

Simple simulation is not satisfactory because some parts of the state space can be ignored due to randomization, so a modeling approach was chosen. Formal modeling is a widely spread technique for verifying the correctness of systems [6]. Amazon and Microsoft have already used the Temporal Logic of Actions (TLA) and its TLA+ formal language [7] for creating specification about their distributed systems [8, 9]. During the model checking, they found several serious bugs that had not come up before. After studying their approach and system specifications, several telemedicine systems were modeled [10], and it was shown that an easily tunable system can assist the design of information critical heterogeneous systems.

Making the trade-off between availability and consistency has notable effects on the QoD. Data quality can be measured in different ways. Its application greatly depends on the type of dataset and the context. In telemedicine systems, the most rapidly changing data portions are numeric data sets. QoD calculations are usually based on a distance function and an aggregation that describes inconsistencies between the real-world phenomena and the data obtained from resources. Hinrichs' formula stated in Eq. (1) describes what QoD means in the context of telemedicine, where x_{db} represents the data stored in a database and x_{real} stands for real-world data at a given t point [11],

$$QoD(x) = \begin{cases} \dfrac{1}{d(x_{db}, x_{real}) + 1}, & \text{if } x_{db} \neq x_{real}, \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

3. TELEMEDICINE USE-CASE

A former study [10] revealed the importance of availability and consistency in information critical heterogeneous systems. This paper presents a concrete, active telemedicine use-case maintained by Inclouded [12], through which the formal system modeling and a new graph-based evaluation technique are performed.

The selected project concerns patients that have been diagnosed with metabolic syndrome. They have high blood pressure along with high fasting glucose levels and abdominal obesity that can lead to cardiovascular disease [13]. Hence, different types of vital signs are measured, and often simultaneously. All the measurements go through a similar data path that includes a sensor and a mobile client that collect the raw data and send them to the cloud. These devices are usually in the patient's home. In the cloud, there is a distributed database and computational units that are responsible for persistence and performing resource-intensive tasks. There are also other computational tasks that depend on earlier aggregations and these can make the data path more complicated. The result is available at different places: it is stored in the database, but it may also exist in the cache. The request for the final result is performed by a Web client that is controlled by a doctor or a nurse.

Here, the patient's 24-h long electrocardiography measurement is taken from the project of metabolic syndrome. In this scenario, the raw data are sent to the database, and the computational unit calculates the Q-wave, R-wave, S-wave (QRS) interval [14], and it sends back the result to the database.

4. METHODOLOGY OF MEASUREMENT

4.1. Formal specification

Firstly, the fact that an approximation does not examine the whole state space is taken into account, so the chosen methodology for system verification is system modeling with formal tools. In TLA+, a complete system specification can be made with its own syntax. In the system spec, the main processes are defined; that is, client operations (Client Write (CW), Client Read (CR), Client Read from Cashe (CRC)), DataBase Write (DB_W) for persistence Data Base PROCessing (DB_PROC)) for aggregation. Both database persistence and aggregation are performed by distributed systems, so multiple server instances are initiated. In order to increase the availability of the system, the read operation of the client is separated into two parts, namely sending requests to the database and sending requests to the cache. Furthermore, to make the system easily tunable, the caches are configurable with the k-staleness parameter derived from the PBS method. Besides k-staleness, the latency is also taken into account. Code 1 shows the formal definition of the CW process. The original specification was written in PlusCal, but the TLA+ toolbox converts the spec using the TLA+ syntax.

Code 1. The CW process definition in TLA+

```
CW==/\pc[10]="CW"
/\IF (numOp<MaxNumOp)
THEN
/\ numOp'=numOp+1
/\ finalD'=finalD+1
/\ cRawD'=<<[d|->finalD',
op|->numOp']>> \o ecgRawD
/\ pc'=[pc EXCEPT ![10]="CW"]
ELSE
/\ pc'=[pc EXCEPT ![10]="Done"]
/\ UNCHANGED<<finalD,cRawD,numOp>>
/\UNCHANGED<<readD,dbL,
calcL,dbRawD,
dbProcD,lenCRawD,
lenDbRawD,latR,latW,latP
```

4.2. Simulation environment

The verification of the system spec can be performed by a model checker. The TLA+ toolbox has a built-in model checker, called TLA Checker (TLC). It constructs state graphs and evaluates them via graph traversals. The result is the diameter of the graph, the number distinct states and the total number of states found.

In order to terminate the model checking, it is necessary to set up a threshold that limits the size of the state graph. Here, the threshold is given by the maximum allowed number of write operations and it is set to 5. Also, the latency is restricted to the $[0\ldots5]$ interval because the state space is rapidly growing by increasing the interval by 1. 4 different latency types are taken into account, there being latency for client write, aggregation, client read from database and client read from cache. It is stated in [15] that there are huge differences among different computer actions. In the model checking only Central Processing Unit (CPU) and Random Access Memory (RAM) are used, and the RAM access takes the most of the time in the calculation of a new state in the graph. So, a new state of the graph can be generated within 100 ns. The significant amount of latency is caused by the network. It is also known that a network connection is almost 10,000,000 times slower than accessing the RAM [16], so increasing the latency by 1 means approximately 100 ms delay in our simulation environment. A delay between 0 and 500 ms can be valid for all the units in the system. The data used for simulation was obtained from the MIMIC-III Waveform Database [17–20] and transferred to integers in order to work with them in TLA+. The cache is configured with the k-staleness parameter that uses values from 0 to 3. If $k = 0$, the client tries to obtain the most up-to-date data. The higher the k-parameter, the more tolerance is added to the system for the staleness of the data.

4.3. Evaluation of state graph

TLC produced 4 variants of state graphs because of the 4 given k-staleness parameter values. Each graph file is 12 GB and dumped in dot format. Dot is the basic file extension for the Graphviz [21] library that is able to visualize and process graphs. On the one hand, the whole state space of the graphs consists of 335,409 nodes and 664,587 edges, and it is not understandable in one piece. On the other hand, this huge graph is hard to fit in memory and it would produce an image file with a similar size. A tile system [22] could solve the visualization problem, but Graphviz cannot make this conversion at this point.

In order to execute graph algorithms on these graphs, a smarter tool is needed. NetworkX [23] is a well-known Python package that was designed for studying structures and analyzing complex networks, but it is not compatible with dot graphs. Hence, the original dot file must be converted to another format. Graphviz supports conversion from dot to the Graph Modeling Language (GML) format. The result is another graph file that has a bigger size due to the syntax of GML. This giant graph file does not fit in memory, so some reductions have to be made.

The original dot files contain long labels describing the current state of the system. If one or two values are only investigated, a significant number of bytes can be dropped and the size of the files can be considerably reduced. This study focuses on the QoD in the project of metabolic syndrome, so only QoD values are kept. These values are the labels of the nodes that represent the possible states of the system. In order to make the graphs clearer, QoD values are converted to Red, Green, Blue (RGB) colors that are useful during visualization.

Hence, nodes have only an identifier and a fill color. Doing this, a file-size reduction of 75% was achieved and the original file size was cut to 3 GB. After reading GML files, NetworkX found the weakly connected components in the graph. Each weakly connected component represents a simulation performed by TLC under given circumstances. Lastly, every weakly connected component can be dumped into separate GML files to make further examination and visualization easier.

5. RESULTS

5.1. Evaluation of state graphs

TLC produced state spaces with more than 280,000 components. After checking the content of the variables in each state, it was found that the evaluation order of the time windows in QRS interval calculations is non-deterministic due to the multiple computational instances. This issue was identified and fixed in the system.

The graph components produced by TLC can have different sizes and various shapes depending on the number of states that can be reached from the initial ones. At first glance, these graphs appear to be acyclic, but after a quick graph analysis, it turns out that the original graph files contain cycles. These cycles are caused by self-loop edges at the leaves of the graphs. TLC adds edges to the graph depending on the next executed TLA+ process. The self-loop edges at leaves are added to indicate the termination. Since TLC did not find any deadlocks and errors, every execution of processes terminated and worked properly.

After removing terminating self-loop edges, it was found these graphs had a Directed Acyclic Graph (DAG) [24] structure. Since DAG is also feasible for describing data process networks [25], it is a suitable structure for characterizing distributed telemedicine systems as well. If the entire system is modeled, all the weakly connected components will have a DAG structure because the whole history of the system is kept, and it is impossible to have an edge to a node that has already been visited. The density of these DAGs is less than 0.3, so they are called sparse graphs.

5.2. Critical paths in information critical heterogeneous systems

The methodology and the elaborated system model show whether there are executions that lead to drastic reduction in QoD. This information can help limit the caches and latencies in order to get the required level of QoD.

DAGs carry lots of information within themselves. If DAGs are adapted to distributed telemedicine systems, topological ordering and the longest paths can present those paths and nodes that lead to critical operations [26]. Topological ordering returns an order of events in a system in which the system works properly or when the system does not work as expected.

Finding the longest paths in arbitrary graphs is a Non-deterministic Polynomial-time hard (NP-hard) problem, but it can be carried out in linear time if the graph is a DAG [27]. The longest path in a DAG can be used to find a critical path that can lead to inconsistency in the system or the system can go into unexpected states that may result in a lower QoD.

In Fig. 2, axis X stands for the length of the possible critical paths that group DAGs. QoD measurements were grouped by the longest paths found in DAGs. In this simulation environment, the possible longest path is 24 in the biggest components. However, there are some components that have only a 2-step-length longest path. Based on the topological ordering and finding the longest path algorithms, after examining the components, it is found that the data quality starts to straighten out after the point where the latencies start to have similar values (when the longest path is above 13). Since computational unit works as a trigger, with no further restrictions, consistent data can only be guaranteed if data arrival is slower than processing. Figure 2 shows this phenomenon in peaks. If the delay of persistence is increased while other processes left unchanged, a rapid

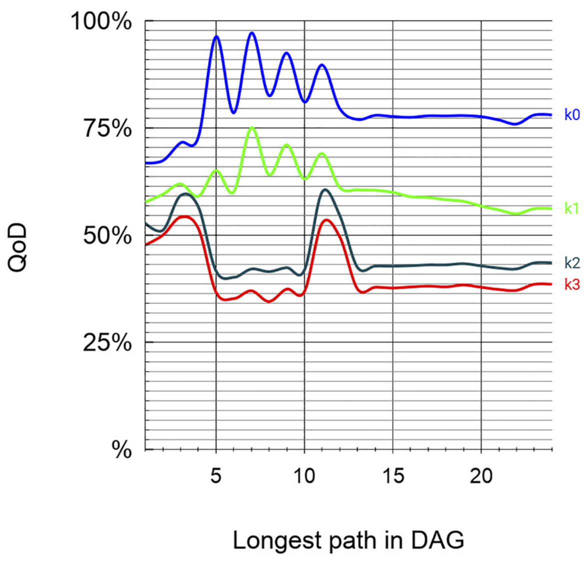

Fig. 2. The changes in QoD depending on the longest path in DAG

improvement can be seen in the QoD with given k-staleness parameter values.

5.3. QoD-based graph clustering

There are well-known and well-tried clustering techniques for finding similar objects and they can be applied to graph components as well [28]. After seeing how the QoD changes if the longest path in DAG is increased, graph components still cannot be clustered because of their cardinality. Several graph editor tools contain clustering methods and algorithms, but none of them found clear similarities among the components. It is found that if components had the same QoD values in leaves, they had the same structure. Therefore, after grouping the components of the whole state space by taking into account the QoD values in leaves, about 4,000 clusters were created. Some clusters contain 2 or 3 components, but others have thousands of graphs.

Figure 3 shows 3 different graphs that were obtained by this clustering method. These were created via the Graphia [29] visualization tool. Black nodes are root nodes of components that represent the 5 defined TLA+ processes. The white color means that the QoD value is 100% in the given state, but in the gray leaves, the QoD is reduced to 25%. The number in the upper-left corner stands for the identifier of the component in the state space. Table 1 lists the whole clustering results in numerical terms. It can be seen that many different executions of the system result in the same graph.

With this technique, not only the similar graph components can be grouped, but also the separate components can be dumped into graph files and visualization can be performed using low performance applications and computers. All in all, this clustering technique is the most helpful in the reduction of complexity in enormous graph spaces.

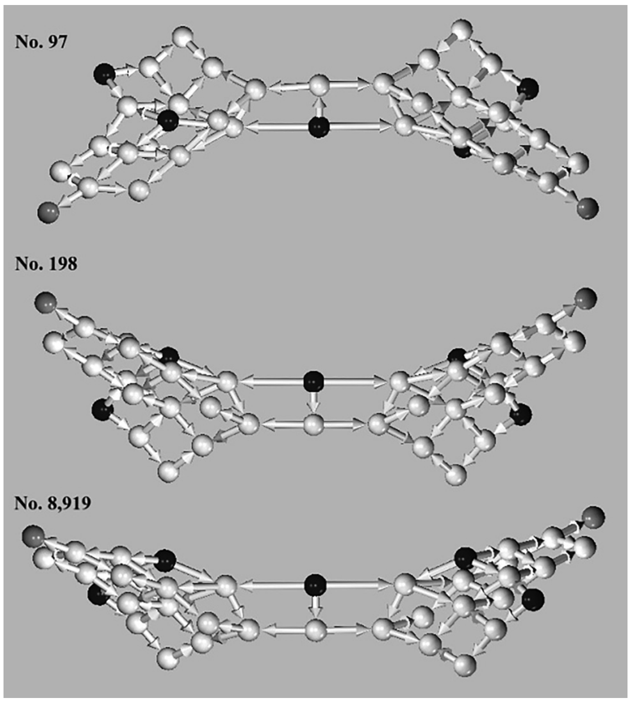

Fig. 3. Graph components obtained by QoD-based clustering

Table 1. Number of weakly connected components and generated clusters with different k-staleness parameter values

k-staleness parameter	Number of components constructed via model checking	Number of clusters obtained from QoD-based clustering method
k_0	276,862	5,383
k_1	288,988	4,735
k_2	282,104	4,112
k_3	282,032	4,139

6. CONCLUSIONS

In this study, it is found that formal modeling and model checking can achieve a complete simulation of a real-world telemedicine system. At this level of abstraction, the methodology pointed out phenomena that cannot be observed on the basis of knowing the system. The created state space can be described by a giant graph that has thousands of weakly connected components. Due to the size of the state space, the complexity of the graph must be reduced in order to obtain valuable graph analytical results. Grouping the components of this system graph by data quality measurements seems to be an appropriate clustering technique that makes visualization and analysis easier and clearer. All the weakly connected components in a telemedicine system graph have a DAG structure, and this composition makes many graph algorithms feasible in linear time. Based on the length of the longest paths in DAGs, critical paths can be found, and it was also shown in which component sizes the highest QoD is most likely to occur. In the future, it is planned to extend these graph analytical techniques and examine other telemedicine use-cases as well.

ACKNOWLEDGEMENTS

This research was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund. This research was also supported by the EU-funded Hungarian grant EFOP 3.6.1-16-2016-00008 and GINOP-2.2.1-15-2017-00073, as well as the 2018-1.1.1-MKI-2018-00249 project and the DeriCom Ltd.

REFERENCES

[1] F. A. Kraemer, A. Bräten, N. Tamkittikhun, and D. Palma. “Fog computing in healthcare a review and discussion,” *IEEE Access*, vol. 5, pp. 9206–9222, 2017.

[2] E. Brewer, “Cap twelve years later: How the “rules” have changed,” *IEEE Comput.*, vol. 45, no. 2, pp. 23–29, 2012.

[3] D. Abadi, “Consistency tradeoffs in modern distributed database system design: CAP is only part of the story,” *IEEE Comput.*, vol. 45, no. 2, pp. 37–42, 2012.

[4] P. Malindi, “QoS in Telemedicine,” in *Telemedicine Techniques and Applications*, G. Graschew Ed., InTech, Rijeka, 2011.

[5] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica. “Probabilistically bounded staleness for practical partial quorums,” *Endowment*, vol. 5, no. 8, pp, 776–787, 2012.

[6] G. Lukács and T. Bartha, “Construction of formal models and verifying property specifications through an example of railway interlocking systems,” *Pollack Period.*, vol. 14, no. 2, pp. 39–50, 2019.

[7] L. Lamport, J. Matthews, M. Tuttle, and Y. Yu, “Specifying and verifying systems with TLA+,” in *Proceedings of the 10th Workshop on ACM SIGOPS European Workshop*, Saint-Emilion, France, July 1, 2002, pp. 45–48.

[8] Consistency levels in Azure Cosmos DB, Microsoft, 2020. [Online]. Available: https://docs.microsoft.com/en-gb/azure/cosmos-db/consistency-levels. Accessed: Dec. 18, 2020.

[9] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff, “Use of formal methods at Amazon web services,” 2014. [Online]. Available: https://lamport.azurewebsites.net/tla/formal-methods-amazon.pdf. Accessed: Dec. 18, 2020.

[10] Z. R. Jánki and V. Bilicki, “Crosslayer cache for telemedicine,” in *The 12th Conference of PhD Students in Computer Science*, Szeged, Hungary, June 24–26, 2020, pp. 159–163.

[11] B. Heinrich, M. Kaiser, and M. Klier, “How to measure data quality? – A metric based approach,” in *Twenty Eighth International Conference on Information Systems*, Montreal, Quebec, Canada, Dec. 9–12, 2007, pp. 108–122.

[12] Inclouded helps teams developing telemedicine and smart city app succeed, Inclouded, 2017. [Online] Available: https://inclouded.sed.hu/. Accessed: Dec. 29, 2020.

[13] About metabolic syndrome, American Heart Association, 2016. [Online]. Available: https://www.heart.org/en/health-topics/metabolic-syndrome/about-metabolic-syndrome. Accessed: Dec. 29, 2020.

[14] A. Kashani and S. S. Barold, “Significance of QRS complex duration in patients with heart failure”, *J. Am. Coll. Cardiol.*, vol. 46, no. 12, pp. 2183–2192, 2005.

[15] J. Poli, Compute performance distance of data as a measure of latency, 2019. [Online] Available: https://formulusblack.com/blog/compute-performance-distance-of-data-as-a-measure-of-latency/. Accessed: May 24, 2021.

[16] T. F. Silva, The good, the bad and the ugly in software development, 2014. [Online] Available: https://tiagodev.wordpress.com/tag/event-loop/. Accessed: May 24, 2021.

[17] B. Moody, G. Moody, M. Villarroel, G. Clifford, and I. Silva, MIMIC-III waveform database (version 1.0),” *PhysioNet*, 2020. [Online]. Available: https://doi.org/10.13026/c2607m. Accessed: Dec. 28, 2020.

[18] A. Johnson, T. Pollard, and R. Mark, “MIMIC-III clinical database (version 1.4)”, *PhysioNet*, 2016. [Online]. Available: https://doi.org/10.13026/C2XW26. Accessed: Dec. 27, 2020.

[19] A. E. W. Johnson, T. J. Pollard, L. Shen, L. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, “MIMIC-III, a freely accessible critical care database,” *Scientific Data*, vol. 3, 2016, Paper no. 160035.

[20] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. C. Ivanov, R. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley, “PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals,” *Circulation,* vol. 101, no. 23, pp. e215–e220, 2000.

[21] Graphviz, Graph visualization software, Graphviz, 2020. [Online]. Available: https://graphviz.org. Accessed: Dec. 29, 2020.

[22] Image tiling, IDL Online Help, 2005. [Online]. Available: https://northstar-www.dartmouth.edu/doc/idl/html_6.2/Image_Tiling.html, Accessed: Dec. 28, 2020.

[23] Network analysis in Python, NetworkX, 2020. [Online]. Available: https://networkx.org. Accessed: Dec. 28, 2020.

[24] R. Tariq, F. Aadil, M. F. Malik, S. Ejaz, U. Khan, and M. F. Khan, “Directed acyclic graph based task scheduling algorithm for heterogeneous systems”, in *Intelligent Systems and Applications*, vol. 869, K. Arai, S. Kapoor, and R. Bhatia, Eds, 2019, pp. 936–947.

[25] A. V. Martinez, “Scheduling in heterogeneous distributed computing systems based on internal structure of parallel tasks graphs with meta-heuristics,” *Appl. Sci.*, vol. 10, 2020, Paper no. 6611.

[26] T. N. Takpé and F. Suter, “Critical path and area based scheduling of parallel task graphs on heterogeneous platforms,” in *Proceedings of the 12th International Conference on Parallel and Distributed Systems*, Minneapolis, MN, USA, July 12–15, 2006, pp. 3–10.

[27] P. M. Pardalos and A. Migdalas, “Note on the complexity of longest path problems related to graph color,” *Appl Maths. Lett.*, vol. 17, no. 1, pp. 13–15, 2004.

[28] D. Nagy, T. Mihálydeák, and L. Aszalós, “Graph approximation on similarity based rough sets,” *Pollack Period.*, vol. 15, no. 2, pp. 25–36, 2020.

[29] Graphia, Visualization Tool for the Creation and Analysis of Graphs, Graphia Technologies Ltd, 2020. [Online]. Available: https://graphia.app. Accessed: Dec. 28, 2020.