DOI: https://doi.org/10.46354/i3m.2019.sesde.005

OPTIMIZING ENERGY EFFICIENCY IN DISTRIBUTED MHEALTH NETWORKS

Philipp Skowron[(a)], Michael Aleithe[(b)], Bogdan Franczyk[(c)]

[(a),(b)] Leipzig University, Grimmaische Straße 12, 04109 Leipzig, Germany
[(c)] Wroclaw University of Economics, Wroclaw, Poland

[(a)]skowron@wifa.uni-leipzig.de, [(b)]aleithe@wifa.uni-leipzig.de,
[(c)]bogdan.franczyk@ue.wroc.pl

ABSTRACT
Energy efficiency in mobile health applications is a relevant problem for long-term monitoring and user acceptance. Various parameters influence the runtime of the system to some degree. One of the parameters is the sampling rate of the individual distributed sensors. Increasing the sampling rate can lead to an increase in energy consumption within the system. By contrast, a reduction can lead to a loss of the data quality, which reduces the informative value of the results of algorithms that use this data. Using optimization methods from reinforcement learning and deep learning to adaptive adjust the sampling rates during runtime, energy efficiency could be improved in only 40 training runs without losing data quality during sampling.

Keywords: mHealth, energy efficiency, therapy systems, reinforcement learning

1. INTRODUCTION

Monitoring and evaluation of daily activities, measurement and control of nutritional habits, monitoring of glucose levels, long-term ECGs for the detection of ischemia signs up to the mobile early detection of emotional break-ins of patients suffering from chronic depression. This is just a small overview of mHealth solutions in the end-customer market, which according to Reuters is expected to grow by more than 35% over the next three years and is already worth 23 billion dollars (Orbis Research 2017). This can be seen especially in the trend that end customers want to live more consciously and healthily and want to have full control over their physical health. In the same time, this trend makes it possible to expand medical care, even where there is no doctor or hospital nearby.

Many solutions use the already existing infrastructures at the end customer's side to perform their services. These include personal smartphone and wearables such as fitness trackers or smart watches. These are either extended with external sensors or the existing sensors in the devices are used. In order to be able to collect data and draw conclusions from this, the sensors in the devices must be used in different ways, which varies depending on the particular use case, i. e. so-called sampling rates are defined for each sensor, which collects data in different time intervals ranging from a few milliseconds to hours. The more fine-grained the sampling rate time intervals are, the higher the power consumption of the individual sensors. This also reduces the battery life of the entire system, which can have a negative impact on the acceptance and long-term use of the system for the end users (both patients and medical staff, depending on the application). Thus, in this way, user acceptance is also a significant aspect of the dissemination of mHealth solutions.

In particular, algorithms such as deep neural networks, which depend on large amounts of data in order to deliver good results, require high sampling rates in order to obtain the necessary data in order to be trained effectively. In order to achieve a long runtime of the overall system, the energy efficiency of the application can be increased by adjusting the sampling rates. This can be described as an optimization problem, whereby the objective function is the runtime of the overall system. The dependent variables of the optimization problem are the sampling rates on the one hand and a data quality measure of the used algorithms that process the measured data and depend on them on the other. Such a data quality measure should avoid that the meaningfulness of the algorithms is impaired during optimization.

The work is organized as follows. Section 2 discusses the existing approaches. Section 3 deals with the optimization problem between total runtime and sampling rates. In section 4 the concept is introduced to be able to do the optimization. Section 5 describes the use case to which the concept was applied. The results of the simulation are presented and discussed in section 6 and section 7 presents the future considerations in detail.

2. RELATED WORKS

In previous work, several approaches to saving energy resources have already been developed. (Nguyen et al. 2008) describes a simulation system with which strategic decisions for the selection of monitoring components, like sensors etc., can be determined to reduce the power consumption of remote monitoring systems. Here, the focus of the work is on the technical description of the simulation system as well as the architectural design of

the simulation system. The actual optimization, however, is carried out manually and not algorithmically.

(Zois, Levorato and Mitra 2013) presents an advanced approach that enables and disables sensors of a distributed monitoring system based on their power consumption and profile. This means that algorithmically, based on quality characteristics, it is determined when which sensor must be used. Like this paper, at (Zois, Levorato and Mitra 2013) the sensors are switched off completely, but no configuration elements of the sensors are included.

Other approaches such as that of (Chai et al. 2014) do not consider the sensors and their power consumption per se, but rather the data transmission and their frequency. It is algorithmically determined when and how many data are to be transmitted from which sensor. Thus, the focus of the paper is on optimizing the transmission intervals between the base station and the sensors without specifically addressing the underlying algorithms and their optimization.

As the approach of (Zois, Levorato and Mitra 2013), (Chattopadhyay and Mitra 2017) looks at the stochastic selection of transmitting sensors in a distributed sensor network based on a modified Gibbs algorithm to determine the incoming data under uncertainty. However, as with (Chai et al. 2014), the focus here is again on optimizing the transmission times between base station and sensors, without going into the optimal selection of sensors at a certain point in time, based on the measured data.

Like (Chai et al. 2014), (Zhang, Zhang and Zhang 2017) uses the transmission rate as a variable for optimizing energy consumption. They maximize the lifetime of the system through a heuristic approach that allows transmit power through multi-hop transmission of data from sensor to sensor. Although the system is working, it can only be used when a multi-hop application is possible.

(Aleithe et al. 2018) developed a simulation framework for troubleshooting mHealth systems. They investigated the runtime on the bases of the energy consumption of the sensors used. Although the approach identifies precise problem sources on the basis of the FMEA, the adjustment is still manual and cannot be changed dynamically.

(Wu et al. 2019) consider the energy consumption when transferring data to a personal device. The sensors are all equipped with an energy recovery module, which allows them to generate their own power. During transmission, no more energy should be consumed than is produced, without causing excessive delays in transmission. Even if the energy consumption is taken into consideration, due to the energy production of the sensors, this is not the main focus for the optimization.

3. OPTIMIZATION PROBLEM

An interesting parameter for improving distributed body area networks is the overall system runtime. Longer system runtime, e. g. due to reduced power consumption, reduces charging times, battery change, etc. This can be achieved by reducing the power consumption of distributed sensors, for example, by influencing certain parameters of the sensors. For example, sensors can be switched on and off or their scanning frequency can be changed. The disadvantage of switching the sensors on and off is that not all sensors can be completely disconnected from the power supply and reconnected later. In addition, there may be other functionalities of the systems that require certain sensors independent of the system under consideration. The scanning frequency of the sensors is thus an effective instrument to reduce power consumption in the overall system without compromising other functionalities.

The sampling interval for sensors determines how often data are collected from the sensors and transferred to the system, where each sensor has its own sampling interval. The sampling interval is defined with the interval $0 < T < +\infty$ and valid under the condition that $T \in \mathbb{R}$.Depending on the nature of the sensor, this sampling interval can take any positive value. The higher the value of the sampling interval, the fewer data from the sensors are collected in some time frame. The overall description of T_n is shown in Equation (1), where t_n is a moment in a time series which is generated by the sensor.

$$T_n = t_{n+1} - t_n | \mathrm{n} \in \mathbb{N}\{0 \leq \mathrm{n} < +\infty\}, \quad \mathrm{t} \in \mathbb{R}\{-\infty < \mathrm{t} < +\infty\} \tag{1}$$

If the sampling interval of a sensor is set to four, for example, the initial value is read in time zero and every four time steps new values are read in each subsequent time (Yang 2014). This fact can be illustrated in Figure 1 more clearly.

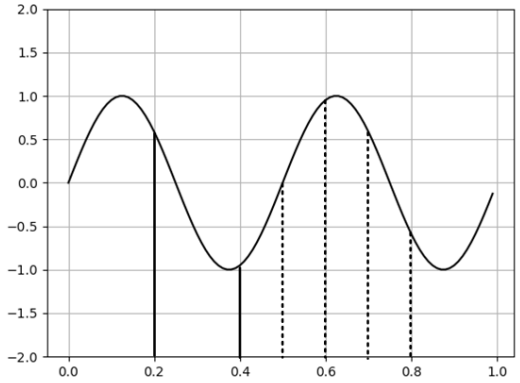

Figure 1: With a sampling interval of 0.2 (black line) only rough fragments of the actual signal are read and processed. If the sampling interval is decreased to 0.1 (dotted line), considerably more characteristics of the signal can be read and processed at the same time.

Now, T is defined as sampling interval, which can increase or decrease. This is an important fact for the optimization problem and the depending variables of T. Therefore, it must be defined how T behaves when it increases or vice versa.

$$T_{dec} := t_n(T) > t_n(T_{dec}) \tag{2}$$

Equation (2) defines, on the one hand, the decreasing of T (T_{dec}). This means that more data are collected in a shorter time than before.

$$T_{inc} := t_n(T) < t_n(T_{inc}) \quad (3)$$

Equation (3), on the other hand, shows that if T increases (T_{inc}), it collects less data at the same time as before, because it takes more time.

In a continuous space, as with physical data such as heart rates etc., the sampling rate determines how accurately the real world can be approximated. Different sampling rates with different sensors, therefore, map the signals and thus the values from the real world with different precision.

The selection of the optimal sampling rates for a maximum total runtime of the system can be described as an optimization problem. The runtime *dur* of the system represents the variable to be optimized and the sampling rates T of any number of sensors represent the parameters. In empirical experiments, it was proved, that the duration of a system is dependent on the sampling rates of all sensors in the system. Therefore Equation (4) defines the duration of the system.

$$dur(T) := d \in \mathbb{N} | 0 < \mathrm{d} < +\infty \quad (4)$$

The problem with this definition of the optimization problem is that no clearly defined optimum exists. In this way, the optimization would converge against the highest possible sampling rates, which would cause almost a stagnation of the data collection. One possibility to define an optimum is the total runtime of the system without sensor reading. However, functionalities based on the collected data would be affected. A better definition of the optimum is, therefore a quality measure, which reflects the quality of functionality or an algorithm output and at the same time is used as an optimization parameter. For predictive algorithms, for example, such a quality measure could be the accuracy of the algorithm, cf. Equation (5), with the measured data based on the sampling rates of the sensors.

$$acc(T) := a \in \mathbb{R} | 0 \leq \mathrm{a} \leq 1 \quad (5)$$

However, these quality measures must be defined individually for each algorithm and each function in order to measure the best possible representation of the quality of the algorithm's output. These quality measures limit the optimization of the sampling rates to the degree that the sampling rates can only take those values which are within a certain tolerance range of the quality measure.

$$maximize(acc, dur) \quad (6)$$

Both, the accuracy of the quality measure and the system duration, should be maximized after the optimization taking into account the dependencies between both variables.

To describe the dependency between *acc* and *dur* Equations (7) and (8) are provided.

$$acc(T) = \begin{cases} acc'(T) \geq 0, if\ T_{dec} \\ acc'(T) \leq 0, if\ T_{inc} \end{cases} \quad (7)$$

$$dur(T) = \begin{cases} dur'(T) \geq 0, if\ T_{inc} \\ dur'(T) \leq 0, if\ T_{dec} \end{cases} \quad (8)$$

Both equations show, that if T increases or vice versa, *acc* and *dur* are reverse related. This is because *acc* rises if more data are available to distinguish between multiple data samples. The connection between *acc* and *dur* will be further described in section 4 and 5.

Thus, the optimization in this case consists of the optimization algorithm, mapped as a reinforcement learning (RL) problem, the total runtime of the system, the sampling rates of the sensors as optimization parameters and the quality measures of the functions and algorithms, which should be included as further optimization parameters and ensure the quality of the system.

4. MODEL CONCEPT

As described in the previous chapter, the model concept presented here consists of several components. These components include the simulated environment with the associated sensors, the restrictive algorithms with their quality measures, which are formulated as boundary conditions for optimization, and the optimization algorithm itself. Figure 2 illustrates the exchange of information between the individual components.

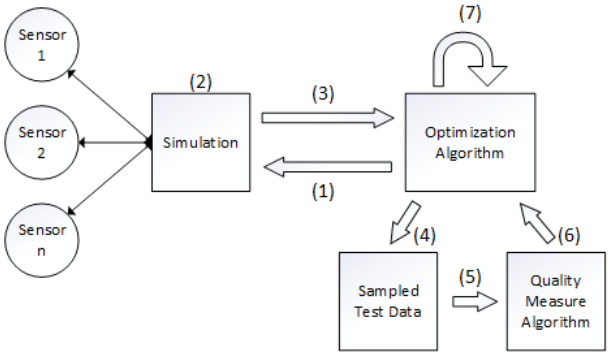

Figure 2: The optimization algorithm gives the sampling rates in the simulation (1). The simulation propagates the sampling rates to the sensors (2) and simulates the total runtime, which is passed back to the optimization algorithm (3). Based on the sampling rates, new test data are then generated (4) which are used in the quality measurement algorithm (5) to return the accuracy based on the current sampling rates to the optimization algorithm (6). The optimization algorithm then updates its network (7) and generates new sampling rates.

As it is shown in Figure 2, the environment simulation is responsible for generating the duration of the total system runtime under the given sampling rates. The restrictive algorithms are also re-tested according to the sampling rates to calculate the change in the quality measure. The

optimization algorithm then calculates new sampling rates and the loop starts again. The individual components of the model architecture are explained in more detail below.

The model of the environment maps the simulated data and the associated sensor network. The data generated for the sensors are based on different distributions to ensure variance between the individual sensors. This data is based on a continuous space, like most of the values measured in the real world. In a concrete example, these values could be the pulse rate or position determined via GPS. Due to the continuous nature of the measured data, the choice of sampling rate also determines the accuracy of the approximation of the environmental values. This, in turn, has an influence on pattern recognition of different algorithms whose data basis is subjected to this approximation.

The quality measurement algorithms in the model consist of both the algorithm itself and the quality measure. Depending on the algorithm, the measurement of quality is defined differently. In this process, the quality measures flow indirectly into the calculation of the optimum to varying degrees, depending on the weighting in the reward function of the optimization algorithm. There is no restriction on the number of quality measures to be used, but there is an increase in the complexity of the environment and the difficulty of finding a global or near-global optimum when using multiple quality measures at once.

As mentioned above, a reinforcement learning (RL) algorithm is defined as an optimization algorithm. Reinforcement learning is a semi-supervised learning method in the field of machine learning. This method consists of the continuous exchange between an action of the agent as a reaction to a state of the environment and the subsequent new state of the environment and a reward signal. Through the continuous exchange of status, reward and action, the agent develops a policy that attempts to optimize a variable in interaction with the environment.

These RL algorithms have the property of reducing the state and action space by learning a function where other approaches such as brute force algorithms fail due to the complexity of the environment and the resulting runtime. In addition, RL algorithms using neural networks enable the learning of non-linear contexts. Since the state space, i. e. the measured data from the sensors, is subject to continuous distribution, policy-gradient functions that can deal with continuous environments are suitable for the selection of appropriate RL algorithms (Sutton and Barto 2018). Recurrent Neural Networks can be used to map the temporal dependency between runtime and selected sampling rates (Hochreiter and Schmidhuber 1997). The output layer is a linear output consisting of three output nodes (total number of sensors in this use case). In contrast to classical classification methods, no classification is carried out here, but all values are returned directly. This has the advantage that the algorithm can adjust several sampling rates simultaneously during a one-time step.

Now that the general structure of the model concept has been described, the next chapter will explain the use case and simulation structure.

5. USE CASE SIMULATION

In the last chapter, the components of the presented model were introduced and explained in general. This chapter applies this model to a concrete use case from a current research project. The use case uses wearables and the participants' smartphones to identify mood patterns. Measured values can be obtained via the sensors from the smartphone as well as via the sensors of the wearables, which can be used to determine firmly classified mood levels. For this purpose, different sampling rates need to be set for different sensors. The problem here is the tradeoff between the long overall runtime of the system and the accuracy of the mood detection. If the sampling rate is maximized, the total runtime of the system drops to a few hours. If the sampling rates are increased simultaneously, the system can operate for up to one day. However, this reduces the accuracy of mood detection as predictive algorithms depend on the measured data.

Real data are not yet available for use in the simulation since they will be collected later in the research project, assumptions about the data must be made in order to prove the convergence and thus the functionality of the approach. The sensor data of the wearables were generated randomized and normally distributed. To be able to predict three different moods with the quality measurement algorithms, three slightly different distributions were added to the simulated data.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} * e^{\frac{-}{2\sigma^2}} \tag{9}$$

Equation (9) shows the used normal (Gaussian) distribution for generating the sample data. Depending on the mood, different mean values for each mood. Mood one got a mean of 2.0, mood 2 a mean of 0 and mood 3 a mean of -2.0. All have the same variance of 1.

The used quality measure algorithm is a fully-connected two hidden layer neural network with classification output. The quality algorithm was pre-trained with data collected from the sensors. All data were labeled with the corresponding class labels of the three moods in a one-hot-encoding. The simulation of the sensors and the associated continuous environment was implemented in Matlab Simulink® and generates the total runtime of the system. For the optimization algorithm, an actor-critic reinforcement learning model has been chosen, as this is a policy-gradient model that converges well with complex, continuous state-action spaces.

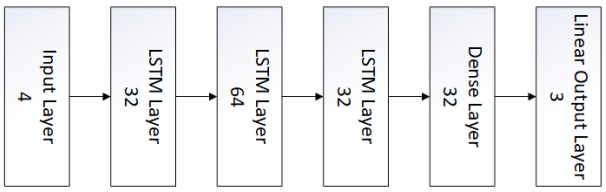

Figure 3: Illustrate the architecture of the deep neural network for the reinforcement learning algorithm

The neuronal network for the reinforcement learning algorithm, displayed in Figure 3, was built up empirically and consists of several layers of LSTM cells to map the temporal dependencies. The Reward hypothesis is formulated as follows:

$$E(r) = \delta * \Big((a_t - a_{t-1}) + \big(a_t - a_{t_0}\big)\Big) + (d_t - d_{t-1})) * y^{-1} \quad (10)$$

a describes the accuracy of the quality measurement and *d* the total runtime of the simulated system. δ and γ are hyperparameters for balancing the importance of the two parameters runtime and accuracy. In this use case, the total runtime was measured in seconds, resulting in a magnitude of 10^4. Variations in the accuracy of the quality measure were in the range from 10^{-3} to 10^{-1}, which required an adjustment of δ and γ to an appropriated level.
The loss function is a standard policy gradient update function similar to the publication in this paper (Mnih 2016). However, a synchronous variant like the one in (Wang et al. 2016) was used. This decision was made due to the limited resources on the one hand and also based on the considerations and arguments and test results of (Wu et al. 2017) on the other hand.

6. RESULTS & DISCUSSION

After a few episodes, the result of several training runs with different configurations was a convergence of the total runtime. As can be seen in Figure 4 the algorithm successfully converged after about ten episodes against a longer runtime than initially indicated.
In 40 episodes the sampling rates were adjusted 15 times in each episode. When averaging the episodes, Figure 4, a short slump within an episode of the total runtime can be considered. Afterward, the expected convergence to an optimal overall runtime happens. A view at the sampling rates of the sensors in Figure 4 over the episodes shows that they also oscillate to an optimum in the applied use case. However, a dynamic environment has not been included, in which the sampling rates do not seek a general optimum state but are adjusted again depending on the changes in the environment, e. g. through user interaction.
The presented results are a proof of concept which shows that the approach leads to an optimum in principle. The fast convergence and the high duration in the example are probably due to the simplicity of the environment, which consisted of only three sensors and a limiting algorithm. A more complex environment would require more effort from the RL algorithm.

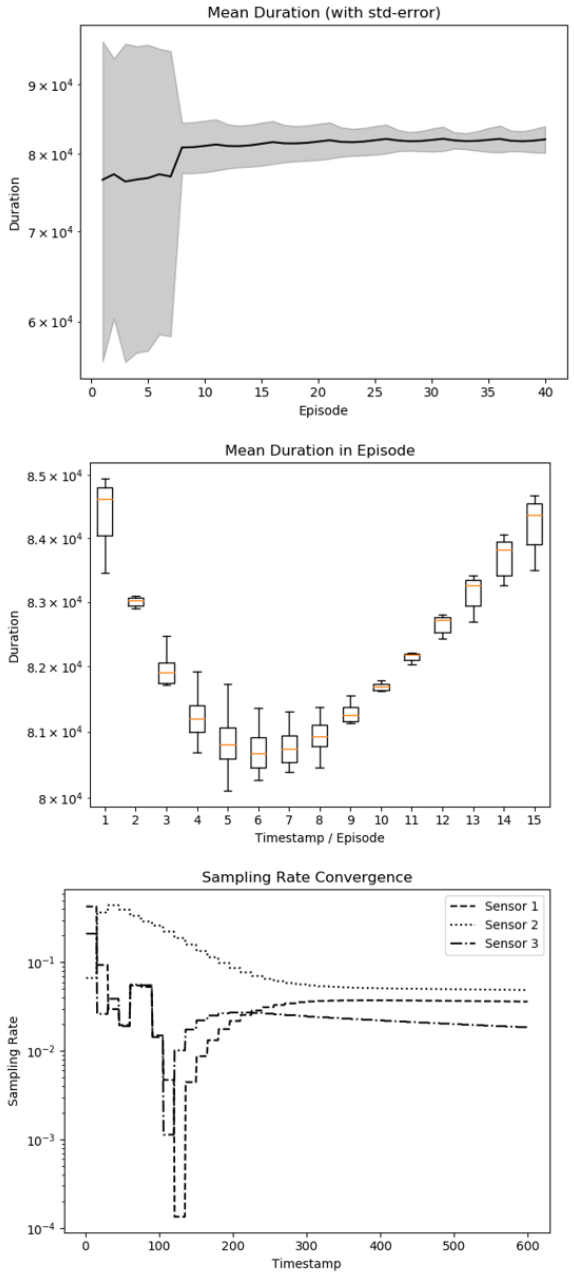

Figure 4: The first diagram shows the average total system runtime over the episodes with standard deviation as an error rate. The second diagram shows the development of the total runtime within an episode during the adjustment steps. The last illustration shows the sampling rates of the sensors and their development across the episodes.

Nevertheless, despite simulative data with fictitious distributions, these outcomes can be interpreted as representative results, since the concept will behave in the same way even with sufficient underlying real data. The same applies to determine the accuracy of the quality measurement algorithms. These fundamentally influence the sensitivity of the algorithm to changes in the dataset. A change of the dataset to real data would only shift the

optimum between total runtime and accuracy, but not the convergence of the RL algorithm, by adjusting the hyperparameter in the reward hypothesis. Thus, the use of the concept for optimizing the overall runtime in mHealth applications can be described as basically successful. The following section describes the next steps for further evaluation of the concept to test its stability under real conditions.

7. CONCLUSION & FUTURE WORK

This paper presented a concept to increase the overall runtime of a distributed sensor system by adjusting the sampling rates of the sensors. A reinforcement learning algorithm was used to optimize the ratio between sensor sampling rates and system runtime. In order to ensure that the quality of the collected data is respected, quality measurement has also been introduced. The result was a convergence of the overall system runtime to an optimal level without reducing the quality of the data and the resulting knowledge out of the data for further algorithms in the system. All calculations were performed on simulated data since the evaluation of real data can only be carried out after the proof of concept.

The aim is to replace the simulated data with real test data to validate the functionality of the system under real conditions. In addition, a dynamic environment will be tested in future observations. For this purpose, everyday user interactions should be considered when calculating the optimum solution. Also, it is planned to test the algorithm on a field test with patients of a depression study.

REFERENCES

Aleithe M., Skowron P., Carell A., Boettger D., Goblirsch T., Franczyk B., 2018. Simulation Framework for Mobile Patient Monitoring Systems. Proceedings of the International Workshop on Innovative Simulation for Health Care (IWISH), 17.09.2018, Budapest, Hungary.

Chai R., Wang P., Huang Z., Su C., 2014. Network lifetime maximization based joint resource optimization for Wireless Body Area Networks. 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC), 1088–1092. 02.09.2014, Washington DC, USA.

Chattopadhyay A., Mitra U., 2017. Optimal Dynamic Sensor Subset Selection for Tracking a Time-Varying Stochastic Process. arXiv preprint arXiv:1711.10610.

Hochreiter S., Schmidhuber J., 1997. Long Short-Term Memory. Neural Computation 8:1735–1780.

Mnih V., Badia A.P., Mirza M., Graves A., Lillicrap T.P., Harley T., Silver D., Kavukcuoglu K., 2016. Asynchronous Methods for Deep Reinforcement Learning. Proceedings of The 33rd International Conference on Machine Learning. 19.06.16, New York, NY, USA.

Nguyen K.D., Cutcutache I., Sinnadurai S., Liu S., Basol C., Sim E., Phan L.T.X., Tok T.B., Francis L.X., Tay E.H., Mitra T., Wong W.-F., 2008. Fast and accurate simulation of biomonitoring applications on a wireless body area network. 2008 5th International Summer School and Symposium on Medical Devices and Biosensors, 145–148. 01.06.2008, Hong Kong, China.

Orbis Research, 2017. mHealth Market Worth $23 Billion in 2017 and Estimated to Grow at a CAGR of more than 35% over the next three years. Reuters. Available from: https://www.reuters.com/brandfeatures/venture-capital/article?id=4640 [01.18].

Sutton R.S., Barto A.G., 2018. Reinforcement Learning. Cambridge:The MIT Press.

Wang J.X., Kurth-Nelson Z., Tirumala D., Soyer H., Leibo J.Z., Munos R., Blundell C., Kumaran D., Botvinick M., 2016. Learning to reinforcement learn. Proceedings of the 38th Annual Conference of the Cognitive Science Society. 10.08.16, Philadelphia, Pennsylvania, USA.

Wu Y., Mansimov E., Liao S., Grosse R., Ba J., 2017. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. Advances in neural information processing systems. 5279–5288. 04.12.2017, Long Beach, CA, USA.

Wu G., Chen Z., Zhang D., Jiaqi L., 2019. Resource allocation algorithm with worst case delay guarantees in energy harvesting body area networks in Peer-to-Peer Networking and Applications 12:74-87.

Yang S.-H., 2014. Wireless sensor networks. London:Springer.

Zhang Y.; Zhang B.; Zhang S., 2017. A Lifetime Maximization Relay Selection Scheme in Wireless Body Area Networks in Sensors, 17:1267-1287

Zois D.-S., Levorato M., Mitra U., 2013. Energy-Efficient, Heterogeneous Sensor Selection for Physical Activity Detection in Wireless Body Area Networks. IEEE Transactions on Signal Processing 7:1581–1594.

AUTHORS BIOGRAPHY

Philipp Skowron studied Business Information Systems at Harz University of Applied Sciences and afterwards he continued his studies at University of Leipzig in the same area of study. Currently, he is writing on his PHD about the simulation of cyber physical systems using artificial intelligence in the mHealth area, which is also his specialization and research interest. Supervisor is Prof. Dr.-Ing. Bogdan Franczyk.

Michael Aleithe studied Systemdesign at Ernst-Abbe University for Applied Sciences and specialized at Modeling and Simulating of Complex Technical Systems. In the past he was employee at Xceptance Software Technologies GmbH and works on agentbased imulations. His research-interest include interconnected data, simulation and mHealth. He is currently working on

a PhD thesis in the area of designing mHealth systems at Leipzig University, Germany supervised by Prof. Dr.-Ing. Bogdan Franczyk.

Bogdan Franczyk is professor at Leipzig University since 2002. In the past he was the research director of Intershop AG as well as professor in Swinburne, Australia and Wroclaw, Poland. His research-area includes especially information management systems.