Exploring Arbitrary Real-life Problems in Proof-of-Useful-Work: Myth busting?

1st Tatjana Davidović
Mathematical Institute SASA
Belgrade, Serbia
tanjad@mi.sanu.ac.rs

2nd Milan Todorović
Mathematical Institute SASA
Belgrade, Serbia
mtodorovic@mi.sanu.ac.rs

3th Bharat Sharma
Penn State Great Valley The Pennsylvania State University
Malvern, PA, USA
bqs5791@psu.edu

4rd Dušan Ramljak
Penn State Great Valley The Pennsylvania State University
Malvern, PA, USA
dusan@psu.edu

***Abstract*—Proof-of-Useful-Work (PoUW) consensus protocols (CPs) are commonly used to improve the Blockchain (BC) efficiency and security. Numerous papers propose various PoUW concepts, majority of them being based on solving the hard real-life optimization problems. A recently proposed Combinatorial Optimization Consensus Protocol (COCP), tried to address exploration of an arbitrary real-life combinatorial optimization problem. Although the ideas are quite straightforward, the actual implementation of such a BC system is still challenging. In this paper we reflect upon the main components of PoUW-based BC system with an emphasis on implementation challenges that arise from the necessity to define and solve an arbitrary real-life CO problem with the appropriate CO algorithm.**

***Index Terms*—Blockchain systems, energy savings, optimization problems, solution methods, rewarding schemes**

I. Introduction

Consensus protocols (CPs) are used within unsupervised maintenance of BlockChain (BC) systems [30]. They are often considered as a very high computation-intensive part of BC systems as they require considerable amount of energy to complete all the tasks needed for this autonomous maintenance. Proof-of-Useful-Work (PoUW) CPs [1], [2], [4], [7], [8], [11]–[15], [17], [20], [24]–[27] were developed to ensure the efficient exploration of available resources, including energy. As in any other CP, the main issues of PoUW-based CPs are to provide security, consistency, reliability, fairness, and immutability of the considered BC system. Those issues are inherently resolved within Proof-of-Work (PoW) based CPs. PoUW-based CPs should be built in such a way to preserve good characteristics of PoW-based CPs and improve upon its downfalls.

There are other CPs that aim to provide solutions for PoW downfalls [9], [28]. The most successful one is Proof-of-Stake [10] (PoS), which was adopted by Ethereum in 2022 [6]. However, for successful adoption of PoS, to become a stakeholder, user needs to have a valuable stake and an established trustful relationship with other users. It is important to note that even then, malicious activities might occur. For example, the highest stake in the context of the considered BC system may constitute only a negligible part of stakeholders' actual wealth. Thus, those stakeholders might use their invested stake to impose their malicious agenda, and effectively centralize the BC system [18].

We focus on PoUW for which we highly value its advantages. Main advantage of PoUW is that, instead of cryptographic puzzles, miners perform tasks useful to wider community. Useful work considered in the majority of PoUW-related papers can involve Machine Learning (ML) or Artificial Intelligence (AI) model training [1], [4], [11], [12], [14], [20] or solving the instances of some real-life Combinatorial Optimization (CO) problems, such as Traveling Salesman Problem (TSP) [13], [15], [25] and maritime and hinterland supply chain related transportation [8], [17]. The latter two papers actually propose the self-sustained BC systems, i.e., the BC systems are intended for transportation purposes and their CPs consider CO problems from the same domain. This holds also for the [21], where useful-work is dedicated to assist the BC management system in identifying malicious BC users. Theoretical model for parameter configuration to minimize the probability of fork occurrences and to increase the throughput of the whole BC system is studied in [29]. However, this study does not consider the details related to maintenance of useful-work tasks within CP.

Some attempts to widen a spectrum of CO problems available as the useful work in PoUW-based BC systems could be found in [2], [7], [24], [26], [27]. However, to the best of our knowledge there is no work that addresses implementation challenges related to solving arbitrary real-life CO problem. In particular, challenges like who will define CO problem, what needs to be transformed, calculated or provided, how to find a corresponding solver, how to verify a solution for an arbitrary real-life problem, etc.

Our main focus is to fill that void. PoUW-based Combinatorial Optimization Consensus Protocol (COCP) [5], [26] started considering the generality of usefulness. Useful work in COCP consists of solving instances of any real-life CO problem as long as instances are well defined and the adequate solution method is available within BC system. For numerous CO problem instances efficient solution methods have already been developed, most of them being available to the wider communities. However, as the number of possible problems and their variants are countless, a new CO problem or a

Serbian Ministry of Science, Technological Development, and Innovations, Agreement No. 451-03-47/2023-01/200029; Science Fund of Republic of Serbia, project AI4TrustBC; Penn State Great Valley Big Data Lab.

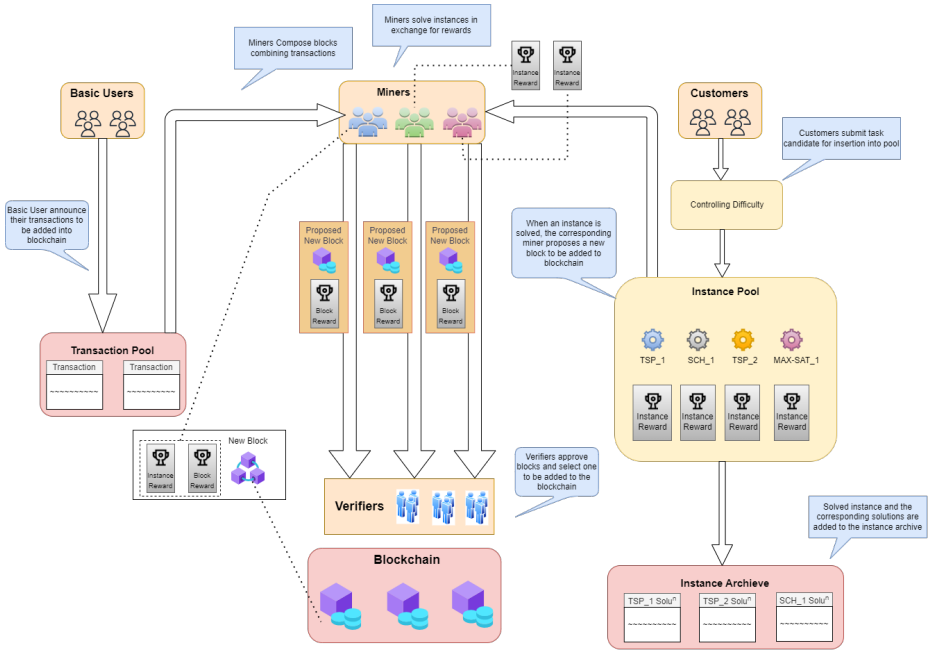

Fig. 1. Structure of PoUW-Based Consensus Protocol

new variant of some existing one can appear with very high probability. Additionally, a framework to group instances into tasks of similar difficulty is developed in [16]. Its role is to ensure the fairness for miners in any PoUW-based CP. These works represent a good basis for achieving the objectives that we set up front and we build our framework and considerations upon them.

Our main contributions include considering how to extend the PoUW-based BC system in such a way to enable the inclusion of arbitrary new CO problems and corresponding solution methods. The main challenges that we identify are to insure that instances are well defined, appropriate optimization algorithms are available, and fairness is achieved.

The paper is organized as follows. After introducing the problem and motivating the solution framework, we recall the main components of PoUW-based CP, emphasizing COCP that allows dealing with different CO problems in Section II. Section III contains structural changes and additions that are needed in PoUW-based CP components to address any real-life CO problem. Main implementation challenges related to the inclusion of a new problem and a new solution method are identified in Section IV. Finally, Section V is devoted to the concluding remarks and directions for future work.

II. The Structure of PoUW-Based Consensus Protocol

Three main components of standard CPs in BC systems: Transaction submission, block mining, and block verification, along with additional components needed for PoUW-based CP are shown in Figure 1 and we explain them in the remainder of this section.

Transaction submission module is usually related to either data exchange or smart contracts. Each transaction is characterized mainly by its origin (sender), destination (receiver), signature of the sender, value to be sent and/or code realizing smart contract, and transaction fee that represents miner's reward. All submitted transactions enter the *transaction pool* waiting to be included in one of the forthcoming blocks.

At the beginning of block mining process, each miner selects a set of transactions from the transaction pool and composes a block to be possibly included into the BC. To increase their reward, miners usually select transactions in a greedy manner. In the next step of block mining, the miner executes CP tasks, which have to testify about the effort invested in the mining process. Usually, a huge amount of computational power or some other resource is required to complete tasks constituting the CP.

For each announced block, a verification has to be performed. The validity of all transactions, as well as the correctness of the CP procedure, are examined by the verifiers. Upon verification, the block is either approved or prohibited for insertion. To be inserted in the BC, block needs to be approved by some predefined number of verifiers and this is referred to as the agreement (consensus). If the block is approved, the corresponding miner is rewarded accordingly. The blocks that are prohibited or that are not verified by the required number of verifiers are discarded and all transactions (if not included in the approved block) are to be returned to the transaction pool.

These three modules are performed in an asynchronous and concurrent way by the BC participants when they take the corresponding roles. They are sufficient to ensure autonomous maintenance of the classical BC systems. However, the main

issue in these BC systems is the enormous usage of resources, especially the electrical energy, for tasks that have no value other than providing autonomous BC maintenance.

When PoUW concept is applied, it is necessary to add two more modules: module for submission of problem instances to be solved as useful work during the mining process and module for retrieval of the corresponding solutions. It is also necessary to add a new type of users, customers, who provide instances of problems they are interested in being solved. In addition, a set of algorithms for solving the submitted problems has to be provided. We briefly describe the two PoUW modules here.

The module for submission of problem instances into the *instance pool* takes as input a description of instances that includes all relevant data, as well as the appropriate solution algorithm. Each instance should be described by the following features: Identification of the user who submits it; a valid address for input data; the solution threshold; the deadline for finding the solution; and the reward offered for providing a valid solution. A valid solution is a feasible one that meets a given threshold. During the mining process, an instance is selected from the instance pool and it is solved by the miner as a part of the CP execution. As various instances may require significantly different execution time for finding a valid solution, grouping of instances into approximately same sized packages is proposed in [16] to balance the miners' work. The size of packages should correspond to the Block Insertion Time (BIT). Some other approaches to balancing miners' work are considered in [2], [7], [24]. The instance (or a package of instances) must be connected to the composed block in an unique way (to mimic the correspondence between the block and the nonce value in PoW). One possible way to realize this correspondence is described in [26]. The obtained valid solutions (if any) are stored in the *solution pool* for verifiers to validate them and nominate one of the miners for a reward.

To get the required valid solutions of their instances, customers should invoke Solution retrieval module [26]. It searches through the solution pool for all valid solutions of a given instance, finds the best one, and provides it to the customer. After that, the reward is granted to the miner who provided the selected best valid solution. It may happen that the Solution retrieval module cannot find valid solution for a number of reasons (deadline is missed, threshold is unreachable, there is no adequate solution method, etc.). It may be possible to overcome the first two reasons, if the corresponding input parameters are flexible and the instance can remain active in the mining process. Otherwise, these issues are problem dependent and cannot be resolved by the means of BC framework. There can be found some attempts to prevail the lack of solution methods, by requiring customers to provide the optimization algorithm together with the problem instance [24], [27], by transforming the optimization problem into an equivalent one for which the solution method exists [2], or by applying some general solution method [7]. However, these approaches may not be always feasible. It is not realistic to assume that customers from industry are familiar with Operation Research field and, even if they are, they might not know how to provide an appropriate solver. In addition, asking customers to submit solution method could deffer them to other resource providers, such as grids, clouds, etc. On the other hand, problem transformation requires additional effort either by customer or by miner (actually, BC management software) and the instance of resulting problem may not be of same size and/or difficulty as the original instance. Moreover, general purpose solvers are usually not as efficient as dedicated solution methods, i.e., solution algorithms developed taking into account an *a priory* knowledge about the considered problem [19].

In contrast to the previous definition of CO problem instance [26] that assumed input data file, threshold, and deadline, we propose to use the term instance for a 5-tuple: input data file, threshold, deadline, corresponding solver, and the estimated solver execution time. To ensure the possibility of solving any CO problem, we need to consider inclusion of additional BC components that we explain in the following sections.

III. The Structure of Changes in PoUW-Based Consensus Protocol

Additional BC components that are needed to support solving an arbitrary CO problem are related to CO problem identification, solver identification, verification of identified solver, how to develop a solver if one does not exist, and how to verify the correctness of a newly developed solver for a given problem. In Figure 2 additional BC components are represented and we describe them in the remainder of this section.

When a customer sends a transaction, or a smart contract containing request to solve their instance is established, the underlying CO problem might be unknown. It is not expected that customers know how to define a CO problem in a way that our framework will be able to match the instance with the appropriate solver. Therefore, we need an additional BC component (*Problem identification*) that will identify a CO problem class and its variant, based on the text files customers provide. We propose to make a module based on modern generative AI technologies (Google AI Bard, ChatGPT, etc.). It is not possible to just rely on inquiries as they might result in multiple or wrong CO problem definitions, and even if they are correct they need to be verified.

As matching between the instance and a CO problem definition is established, the next step is to provide an appropriate solver. The additional BC component needs to automatically find the solver. Two scenarios can be identified for this search: first, the solver is already incorporated into BC framework *Solver matching*, second, it is necessary to search for solver outside the BC framework *Finding solver*. Both scenarios might be considered as a part of useful work, i.e., a search can be performed by miners during the mining phase. In the first case, the complexity of the search is fully defined by the complexity of BC system, i.e., the number of CO problem solvers already included. For example, we could establish a

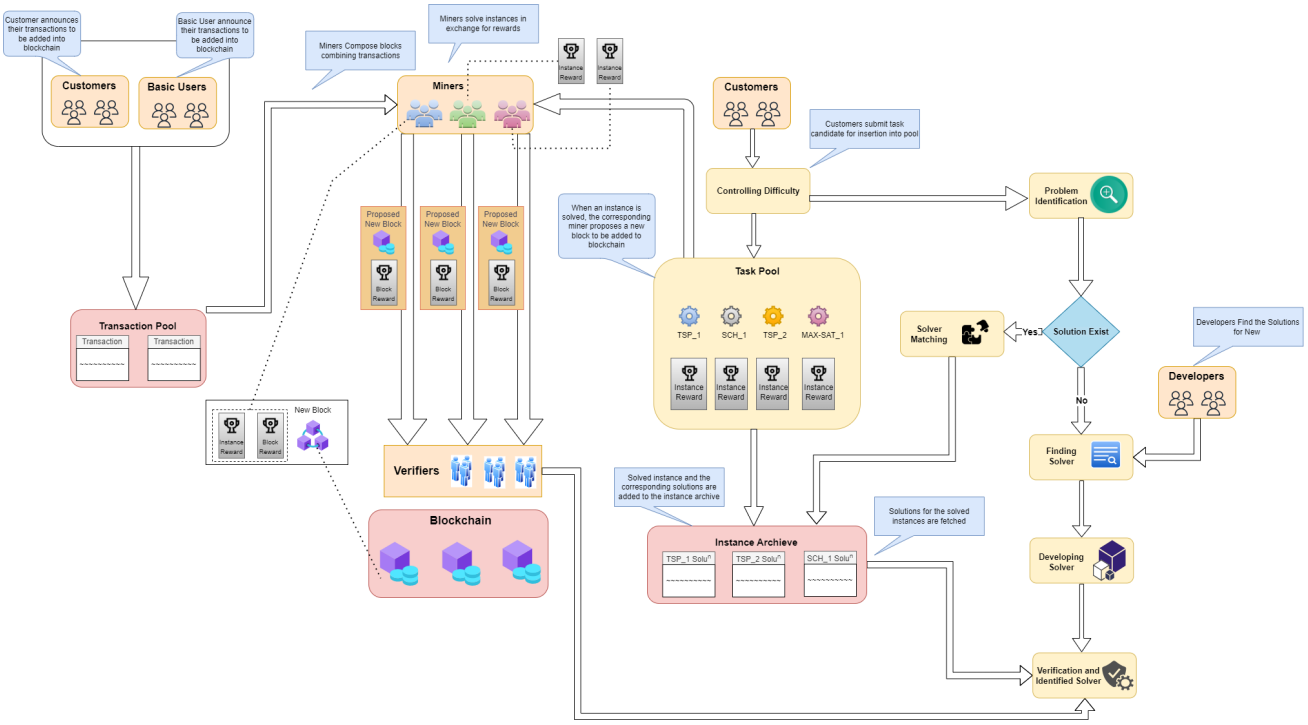

Fig. 2. The new COCP solution framework

smart contract that contains a list of all already incorporated CO problem solvers. In the second case, the search should be expanded on publicly available repositories outside of BC system and thus its complexity needs to be estimated.

Verification of identified solver is needed to check whether the match is correct. As the search for solver is performed as a part of useful work, verification should be performed inside the BC system. However, we still need to add this component that would be used by the verifiers.

In case that a solver does not exist, an additional BC component *Developing solver* is needed to develop an appropriate algorithm. This could also be considered as a part of useful work of CP, if an automatic program generator or AI is used. However, we do not expect an algorithm to be developed within the block mining time defined by BIT. Thus, there might be a need to include a new type of BC users that we will call *developers*. For developers' autonomous work we need to plan for adequate incentives and reward scheme. The developers' contributions should be considered as useful work for the whole BC system, outside of CP.

Similarly to the customers who join the BC system to benefit from getting their CO problem instances solved, the developers can be new participants with good programming skills that wish to make a profit by being rewarded for providing solution algorithms for some CO problems. On the other hand, any other BC user can decide to change their role, and become a developer. For example, miners who did not manage to publish a block for some time may decide to increase the reward by developing new optimization algorithms.

Whether the solver is developed as a part of useful work or by developers, *Verification of the developed solver* needs to be envisioned as an additional BC component.

IV. Implementation Challenges

Implementing a successful PoUW-based consensus protocol requires addressing the many implementation challenges that we identify in the remainder of this section. We grouped these challenges in such a way to correspond to the suggested additional BC components.

Performance evaluation of the envisioned framework is not possible before all the challenges are resolved. As we rely on previously established COCP [5], [26] and difficulty estimation framework [16], stability, scalability, and fairness should not change when we extend the variety of the considered CO problems.

A. CO Problem Identification

As we already mentioned, identification of CO problem through modern generative AI technologies need to be verified. Verification of CO problem definition might be impossible without customer interaction. A possible solution would be to make an user interface which would allow a customer to reflect upon correctness of problem identification.

B. Solver Identification

The goal of solver identification is to pair up a CO problem and the appropriate solver. In such a way the CO problem instance is fully defined and can be sent to the controlling difficulty module [16] for estimating the solver execution time. If the result of solver identification is `multiple solvers`,

the difficulty must be estimated for all input data-solver pairs. After the controlling difficulty is completed, the solver with the shortest estimated execution time should be combined with input data to make an instance which will be added. As the solver execution time is already estimated, it should be set to the appropriate value. When the result of solver matching is `not found`, that means that we do not have an appropriate solver and finding solver needs to be triggered. However, when finding solver results in `not found`, that might indicate that we didn't allocate enough time for this task. We can increase the time for search up to the BIT and, if the result is still `not found`, it is time to give up searching and delegate the problem to the developers.

Announcing the need for a new solver could be done through a smart contract or transaction, the same way that we use to submit new instances. That announcement should be a result of both the estimated demand for CO problem instances without a solver and the estimated return of the investment in development. In such a way we prevent malicious behavior of users who have already developed solvers for their own use.

C. Verification of Identified Solver

Automatic verification whether the solver identification was successful may be a very difficult challenge. There is no known evaluation measure that will assess the correctness of an arbitrary CO problem solver.

There exist evaluation measures for some known CO problems and their solvers. This knowledge is the basis for building an automatic verifier for each new problem solver that we want to incorporate into the BC system.

D. Developing Solver

Any user may choose to become a developer at any time, so the solver developing process should be considered concurrent. Consequently, it is possible to end up with several solvers for the given CO problem. Therefore, suitable incentives and reward scheme should be proposed to both stimulate the developers and discourage any malicious behaviour of other users.

To supply the incentives, the customers could be asked to increase the reward provided for solving their instances. Miners' reward would be one part of the provided reward, while the other part would be stored for rewarding developers.

Malicious behavior of developers should be prevented even before the stating the announcement for the new solver and it is thus addressed in solver identification.

E. Verification of Developed Solver

With respect to verification of identified solver, automatic verification of developed solver is even more complex challenge. However, resolution of this challenge could be the same automatic verifier we introduced.

F. Additional Implementation Challenges

Additional challenges include how to incorporate the request to solve new problems and how to estimate a difficulty for that new problem. Regarding the first mentioned challenge, one possible solution would be to delay a few initial requests for solving some new CO problem, but to track requests similarly to caching in storage systems [22]. If there is an identified need to consider this particular CO problem, only then we initialize the procedure to include it in the BC system.

To estimate a difficulty of CO problem instances, we need information about solvers and their complexity. This information should be available to the difficulty estimation module [16]. Storage of solvers and solved instances might become a problem for an established network. On an opposite end, the problem is how to estimate difficulty without any a priori information about instances. One possible solution, inspired by virtual cache framework for storage systems [3], [22], [23], is to limit the solution time to BIT, track it, and learn the difficulty during the execution of solution method over instances in the instance pool.

Finally, our future work needs to include resolution of a challenge that is related to miners not being able to solve all instances they were given. Problems include the ways how to verify their effort. As main goals of the PoUW-based CPs are to provide more incentives to miners for their effort and at the same time provide customers with valid solutions (thus encourage them to submit their instances to be solved), it would make sense to reward miners for the work they completed. Thus, it seems to be a good idea to set a threshold of the percentage of instances that need to be solved to publish a block. This should be considered carefully as the fairness established through difficulty estimation framework may be in jeopardy. However, establishing this threshold contributes to efficient usage of the BC system resources.

V. Conclusions

We presented the main components of the Proof-of-Useful-Work (PoUW) based Consensus Protocol (CP), with an aim to address exploration of an arbitrary real-life combinatorial optimization (CO) problems. The main advantages of the envisioned consensus protocol besides efficient utilization of computing resources and various sources of rewards for participants are that it allows for wide community adoption by treating useful work that is of value for more customers. We discussed how the structure of the envisioned changes would look like and what are the challenges that need to be solved for the system to work. The new type of BC users, developers are introduced to supply CP with solutions for new CO problems not included in the Blockchain (BC) system.

Possible directions for future work may include detailed implementation of all components of envisioned CP. It is also important to identify and resolve all security challenges that come with solving real-life CO problem instances, introducing new type of participants, and providing various incentives for them. A very important part of future work that might have further implications is resolving the identified challenges with the automatic verification of developed solvers.

ACKNOWLEDGMENT

Authors thank students Anja Vujačić University of Belgrade, School of Electrical Engineering, Ognjen Nešković, and Pavle Sekešan, University of Belgrade, Faculty of Mathematics for valuable discussions. In addition, authors express their gratitude to the anonymous reviewers for their valuable comments that resulted in the improvements of this paper.

REFERENCES

[1] A. Baldominos and Y. Saez, "Coin. ai: A proof-of-useful-work scheme for blockchain-based distributed deep learning," *Entropy*, vol. 21, no. 8, p. 723, 2019.

[2] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, "Proofs of useful work," IACR Cryptology ePrint Archive (https://eprint.iacr.org/2017/203.pdf), 2017, last update 2021.

[3] S. Bhattacharya, K. Gopinath, and D. Voigt, *Resource Proportional Software Design for Emerging Systems*. CRC Press, 2020.

[4] C. Chenli, B. Li, Y. Shi, and T. Jung, "Energy-recycling blockchain with proof-of-deep-learning," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 19–23.

[5] T. Davidović, M. Todorović, D. Ramljak, T. Jakšić-Krüger, L. Matijević, Đ. Jovanović, and D. Urošević, "COCP: blockchain proof-of-useful-work leveraging real-life applications," in *International Conference on Blockchain Computing and Applications (BCCA 2022)*. San Antonio, Texas, USA: IEEE, 2022, pp. 107–110.

[6] Ethereum.org, "Proof-of-stake (pos)," https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/, 2023, accessed July 2023.

[7] M. Fitzi, A. Kiayias, G. Panagiotakos, and A. Russell, "Ofelimos: Combinatorial optimization via proof-of-useful-work\a provably secure blockchain protocol," IACR Cryptology ePrint Archive (https://eprint.iacr.org/2021/1379.pdf), 2021.

[8] M. Haouari, M. Mhiri, M. El-Masri, and K. Al-Yafi, "A novel proof of useful work for a blockchain storing transportation transactions," *Information Processing & Management*, vol. 59, no. 1, p. 102749, 2022.

[9] L. Ismail and H. Materwala, "A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions," *Symmetry*, vol. 11, no. 10, p. 1198, 2019.

[10] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, no. 1, 2012.

[11] B. Li, C. Chenli, X. Xu, T. Jung, and Y. Shi, "Exploiting computation power of blockchain for biomedical image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 2802–2811.

[12] B. Li, C. Chenli, X. Xu, Y. Shi, and T. Jung, "Dlbc: A deep learning-based consensus in blockchains for deep learning services," *arXiv preprint arXiv:1904.07349v2*, 2020.

[13] W. Li, "Adapting blockchain technology for scientific computing," *arXiv preprint arXiv:1804.08230*, 2018.

[14] A. Lihu, J. Du, I. Barjaktarevic, P. Gerzanics, and M. Harvilla, "A proof of useful work for artificial intelligence on the blockchain," *arXiv preprint arXiv:2001.09244*, 2020.

[15] A. F. Loe and E. A. Quaglia, "Conquering generals: an NP-hard proof of useful work," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*. ACM, New York, NY, 2018, pp. 54–59.

[16] U. Maleš, D. Ramljak, T. Jakšić-Krüger, T. Davidović, D. Ostojić, and A. Haridas, "Controlling the difficulty of combinatorial optimization problems for fair proof-of-useful-work-based blockchain consensus protocol," *Symmetry, Special Issue "Advances in Multidisciplinary Exploration for Symmetric Key Cryptography and Blockchain Technology"*, vol. 15, no. 1, pp. 140:1–32, 2023.

[17] M. Mhiri, K. Al-Yafi, B. Legros, O. Jouini, and M. Haouari, "A blockchain-based framework to optimize shipping container flows in the hinterland," *International Transactions in Operational Research*, 2023.

[18] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85 727–85 745, 2019.

[19] D. Ostojić, A. Urošević, T. Davidović, T. Jakšić Krüger, and D. Ramljak, "Decomposition-based efficient heuristic for scheduling," in *(submitted)*, 2023.

[20] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud-edge-end in iot: A blockchain-assisted collective q-learning approach," *IEEE Internet of Things Journal*, 2020.

[21] D. Ramljak, T. Davidović, D. Urošević, T. Jakšić Kruger, L. Matijević, M. Todorović, and Đ. Jovanović, "Combinatorial optimization for self contained blockchain: An example of useful synergy," in *Proc. XLVIII Symposium on Operational Research, SYMOPIS 2021*, Banja Koviljača, Serbia, 2021, pp. 285–290.

[22] D. Ramljak, *Data Driven High Performance Data Access*. Temple University, 2018.

[23] D. Ramljak, D. A. Tom, D. Voigt, and K. Kant, "Modular framework for data prefetching and replacement at the edge," in *Edge Computing–EDGE 2018: Second International Conference, Held as Part of the Services Conference Federation, SCF 2018, Seattle, WA, USA, June 25-30, 2018, Proceedings 2*. Springer, 2018, pp. 18–33.

[24] N. Shibata, "Proof-of-search: combining blockchain consensus formation with solving optimization problems," *IEEE Access*, vol. 7, pp. 172 994–173 006, 2019.

[25] W. A. Syafruddin, S. Dadkhah, and M. Köppen, "Blockchain scheme based on evolutionary proof of work," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 771–776.

[26] M. Todorović, L. Matijević, D. Ramljak, T. Davidović, D. Urošević, T. Jakšić-Krüger, and Đ. Jovanović, "Proof-of-useful-work: Blockchain mining by solving real-life optimization problems," *Symmetry, Special Issue "Advances in Multidisciplinary Exploration for Symmetric Key Cryptography and Blockchain Technology"*, vol. 14, no. 9, pp. 1831:1–47, 2022.

[27] A. Toulemonde, L. Besson, L. Goubin, and J. Patarin, "Useful work: a new protocol to ensure usefulness of pow-based consensus for blockchain," in *Proceedings of the 2022 ACM Conference on Information Technology for Social Good*, 2022, pp. 308–314.

[28] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *ACM Computing Surveys*, 2023.

[29] Q. Zhao, X. Tai, J. Yuan, J. Xu, L. Feng, and Z. Ma, "Performance analysis of pouw consensus mechanism: Fork probability and throughput," *Peer-to-Peer Networking and Applications*, vol. 15, no. 2, pp. 1126–1138, 2022.

[30] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *IEEE international congress on big data (BigData congress)*. IEEE, 2017, pp. 557–564.