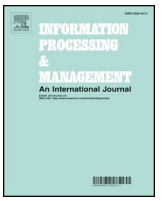
Information Processing and Management 58 (2021) 102393

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman



PF-BTS: A Privacy-Aware Fog-enhanced Blockchain-assisted task scheduling

Hamza Baniata[*,a], Ahmad Anaqreh[b], Attila Kertesz[a]

[a] University of Szeged, Department of Software Engineering, Szeged 6720, Hungary
[b] University of Szeged, Department of Computational Optimization, Szeged 6720, Hungary

ARTICLE INFO

Keywords:
Cloud computing
Fog computing
Internet of Things
Blockchain
Task scheduling
Ant Colony Optimization

ABSTRACT

In recent years, the deployment of Cloud Computing (CC) has become more popular both in research and industry applications, arising form various fields including e-health, manufacturing, logistics and social networking. This is due to the easiness of service deployment and data management, and the unlimited provision of virtual resources (VR). In simple scenarios, users/applications send computational or storage tasks to be executed in the cloud, by manually assigning those tasks to the available computational resources. In complex scenarios, such as a smart city applications, where there is a large number of tasks, VRs, or both, task scheduling is exposed as an NP-Hard problem. Consequently, it is preferred and more efficient in terms of time and effort, to use a task scheduling automation technique. As there are many automated scheduling solutions proposed, new possibilities arise with the advent of Fog Computing (FC) and Blockchain (BC) technologies. Accordingly, such automation techniques may help the quick, secure and efficient assignment of tasks to the available VRs. In this paper, we propose an Ant Colony Optimization (ACO) algorithm in a Fog-enabled Blockchain-assisted scheduling model, namely PF-BTS. The protocol and algorithms of PF-BTS exploit BC miners for generating efficient assignment of tasks to be performed in the cloud's VRs using ACO, and award miner nodes for their contribution in generating the best schedule. In our proposal, PF-BTS further allows the fog to process, manage, and perform the tasks to enhance latency measures. While this processing and managing is taking place, the fog is enforced to respect the privacy of system components, and assure that data, location, identity, and usage information are not exposed. We evaluate and compare PF-BTS performance, with a recently proposed Blockchain-based task scheduling protocol, in a simulated environment. Our evaluation and experiments show high privacy awareness of PF-BTS, along with noticeable enhancement in execution time and network load.

1. Introduction

Cloud Computing (CC) is the paradigm that has been used for years now, performing tasks for end-users in a reliable, and efficient manner. CC paradigm provides different types of services, such as Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS), and Platform-as-a-Service (PaaS). It also offers different types of tasks that can be performed in order to meet storage, computation and communication needs of end-users. Cloud services mainly rely on the use of virtual machines (VMs) concept, which logically divides the resources in the cloud, into separate machines, in a way to make it easier for users to get services in a Pay-as-you-Go manner. This means that the less virtual machines/resources are used, the less payment for the services to be made. Consequently, in

* Corresponding author.
E-mail address: baniatah@inf.u-szeged.hu (H. Baniata).

https://doi.org/10.1016/j.ipm.2020.102393
Received 14 April 2020; Received in revised form 18 September 2020; Accepted 21 September 2020
Available online 30 September 2020

big applications that require a large number of Virtual Resources (VRs, which we use for the generalization of VMs), optimizing the deployment of VRs to be used for the least time may save the application users a great deal of money. Furthermore, the decreased usage of VRs leads to enhanced levels of energy utilization and efficiency Gai, Qiu, and Zhao (2018).

Public clouds provide different options of services that can be adopted in complex smart Internet of Things (IoT) applications, such as Smart City Tortonesi, Wrona, and Suri (2019) and industrial IoT Yang, Hou, Ma, and He (2019), to enhance mostly all Quality of Service (QoS) measures of those applications. As those services include platforms, software, and infrastructure levels, they facilitate data outsourcing/storage and information analysis. Constant streams of data and information are, thus, expected to rise as an issue in such complex smart IoT-Cloud scenarios. Further, those streams may carry highly sensitive and private data that need to be secured during and after its processing and storage in the cloud. As an extension of the cloud at the edge of the network, Fog Computing (FC) is envisioned to provide cloud services closer to end-users. FC was majorly proposed, in 2013, as a middle layer between end-users and the cloud, performing tasks and services while reducing latency, network load, and energy consumption. Applications of the IoT paradigm was the motivation of the FC introduction, as most applications require real-time services, and short response time. This is due the fact that Things are resource-limited (i.e. energy, storage, and computation power). Additionally, the fog layer is expected to dynamically control the streams of data and information among all parties of the network. To allow the integration of CC, FC, and IoT, other services were recently made available, such as Backend-as-a-Service, Process-as-a-Service, and Security-as-a-Service Gai, Guo, Zhu, and Yu (2020).

Nowadays social media and networking services also produce a vast amount of information. The management of such big data would not be possible without the use of virtualized, cloud services. Furthermore, an effective information processing and management, and traffic engineering of such complex systems require advanced techniques from additional (even encoupled) research fields, such as IoT, FC, mobile technologies and security Bellavista et al. (2020); Karthikeyan and Thangavel (2018). In order to accomplish this required effectiveness of data management in an IoT-Fog-Cloud environment, the deployment of highly efficient task scheduling schemes is essential.

In computing, task scheduling is the method by which tasks are assigned to resources that perform the computations Panda, Gupta, and Jana (2019). This assignment problem has been shown to be NP-Hard problem Kalra and Singh (2015); Ullman (1975), because the more tasks assigned, the higher the number of probable assignment schedules T'kindt and Billaut (2006). To find the best assignment of requested tasks to the available resources, we need to generate $n!$ assignment schedules, where n is the number of tasks, and find the optimum assignment out of them. For example, a bakery production line that processes 40 products on 26 production stages, leads to 8.2×10^{47} different possible schedules, which is not solvable in reasonable time using exact methods Hecker, Stanke, Becker, and Hitzmann (2014). In such cases, optimization algorithms are used to find the most optimal solution. The assignment problem had been discussed in many books and papers due to the wide variety of problem versions got when considering different criteria in different execution environments. For deep and detailed explanation of those versions, criteria and systems, we direct the reader to seek information in Bokhari (2012) Pentico (2007), and Bacsó, Kis, Visegrádi, Kertész, and Németh (2016).

Ant Colony Optimization (ACO) Dorigo, Birattari, and Stutzle (2006) was introduced as a Computational Intelligence (CI) meta-heuristic technique for optimizing a wide set of different problems, such as the assignment problem we are presenting. ACO takes inspiration from the social behaviour of some ant species, who deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. In ACO, a number of artificial ants build solutions to the considered optimization problem at hand and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants.

Blockchain (BC) is the base technology of many available cryptocurrency systems. BC was firstly introduced in Nakamoto (2019), as an underlying technology that guarantees a trusted, fully-distributed, digital money system. Since then, BC had been proposed for enhancing trust and security in various applications, by being deployed for managing data, payment, identity, reputation, or other purposes Baniata and Kertesz (2020). Further, the ability of providing decentralized decisions by a BC network has excited many researchers to deploy it in the decentralized, distributed Cloud servers at the edge of the network. Subsequently, Blockchain-as-a-Service (BaaS) was proposed and analyzed in the literature, as a service model similar to the Process-as-a-Service schemes Samaniego, Jamsrandorj, and Deters (2016). Accordingly, different BaaS models are strongly expected to support CC during the life cycle of information processing and management in complex scenarios Garcia-Font (2020). That is, BC providing computational power and immutable storage abilities should lighten the burden on clouds.

In this paper, we propose a task scheduling optimization protocol, called Privacy-aware Fog-enhanced Blockchain-assisted Task Scheduling (PF-BTS). This protocol deploys BC miners for performing a number of ACO iterations, in order to provide a secure and efficient assignment of cloud VRs to perform the tasks requested by applications (e.g. IoT), hence achieve a minimal task computing time on the cloud's VRs. Meanwhile, we propose fog components to control and handle the communication between the IoT layer, the BC network, and the Cloud layer. PF-BTS assures the anonymity of end-user devices and performed tasks, while it allows the cloud to award BC miners for their contributions through smart contracts. Thus, PF-BTS facilitates the provision of a privacy-aware BaaS to further enhance data and information processing in the fog and the cloud layers.

The rest of this paper is presented as follows: Section 2 presents some of the related work regarding our proposal. Section 3 presents our proposed protocol in the FC architecture. Section 4 presents our evaluation in terms of privacy, execution time, and network load. Finally, Section 5 concludes our work.

2. Related work

Ala'a Al-Shaikh, Sharieh, and Sleit (2016) investigated the resources utilization problem in cloud computing, and proved that it is an NP-Complete problem. Accordingly, they implemented and analyzed a greedy algorithm to address such issue. Panda et al. (2019) proposed three allocation-aware task scheduling algorithms for a multi-cloud environment. In their work, different cloud service providers

were assigned to different tasks. The FCFS approach was used in different algorithms, in which one was shown to be better than the others.

The origin of the ACO algorithm goes back to the Ant algorithm proposed in the early nineties by Colorni, Dorigo, Maniezzo et al. (1992). Various variations have been proposed since then, such as Ant Colony System (ACS), Elitist Ant System (EAS), Max-Min Ant System (MMAS), and Rank-based Ant System (ASrank). Tuba, Jovanovic, and SERBIA (2009), provided deep analysis and comparison between different variations of the Ant algorithms. The comparison of different variations of Ant Algorithms is out of the scope of this paper. However, for more information on this topic, we direct the reader to investigate them in Dorigo and Blum (2005).

Wilczyński and Kołodziej (2020) developed a Proof-of-Schedule (PoSch) algorithm in their proposed Blockchain-based Scheduling, namely (BS) approach, for solving the same problem we are discussing. In their proposed approach, task schedulers in the cloud layer are treated as BC miners. Moreover, the BC miners in their approach were categorized into four different groups, each of these groups find their optimal solution using a different algorithm. The algorithms used in the four groups are the FCFS, Shortest Job First (SJF), Round-Robin (RR), and Hybrid Heuristic based on Genetic Algorithm (HSGA). The cloud then chooses the least time/energy consuming assignment. Zhou et al. (2019) proposed MGAS; a modified genetic algorithm (GA) combined with greedy strategy, which was evaluated in terms of average response time and and maximum QoS in addition to the total execution time of VRs. MGAS was compared to the classical GA, FCFS, and Min-Min algorithms, and was shown to outperform them in most cases. Umarani Srikanth, Maheswari, Shanthi, and Siromoney (2012) used ACO algorithm to achieve a feasible assignment of tasks to heterogeneous processors. In their research, it was also experimentally proven that optimizing task scheduling using ACO guarantees better task assignment than the results of FCFS approach.

Merkle and Middendorf (2003) used the ACO global pheromone variable in addition to using only local pheromone information. Before that, they used a combination of two pheromone evaluation methods by the ants to find new solutions in Merkle, Middendorf, and Schmeck (2002), and analyzed a case where an ant forgets the best-found solution. This influenced the heuristic on the decisions made by the ants during the run of the algorithm.

Li and Wu (2019) studied the load imbalance problem in System Wide Information Management (SWIM) task scheduling. To achieve an optimal scheduling, they deployed the ACO with some specific measurements to update the pheromone value, leading to a new variation of the ACO algorithm that performs even better. In Jiao, Wang, Niyato, and Suankaewmanee (2019) an auction mechanism for offloading computations, such as puzzle solving tasks in PoW, from resource-poor BC miners, such as mobile devices, to the fog or the cloud, was proposed. This allocation of computing resources to miners was shown to be computationally efficient.

Mirtaheri and Shirzad (2019) deployed ACO to find the shortest route for a packet to go from a node in an FC environment, visiting all fog nodes, and then coming back to the starting point. Their problem was a type of Traveling Salesman Problem (TSP). Hussein and Mousa (2020) proposed two improved variants of ACO and Particle Swarm Optimization (PSO) for scheduling tasks of end users to be performed in the fog and cloud layers. Xu, Hao, Zhang, and Sun (2019) deployed the laxity and ant colony system algorithm (LBP-ACS) in their proposed cloud-fog task scheduling. The system includes end-users sending their tasks to the nearest fog node, the fog node then forwards the tasks to the fog broker, which performs the assignment of the tasks into the fog layer and the cloud. When the results come back to the broker from both the fog and the cloud, the broker combines the results and sends them back to the fog node. Finally, the fog node sends the results back to end-users.

Kavitha and Sharma (2019) concluded, in their ACO performance analysis, that most of the research works proposed in the literature focused on reducing the cost of VM allocation, while some others focused on reducing energy. Yet the response time and network load measurement had not been given much attention, which is an important factor for delay-sensitive applications. For example, a comparison conducted by Dam, Mandal, Dasgupta, and Dutta (2014) showed that ACO outperforms FCFS and GA in terms of response time, which implies that ACO is better for delay-sensitive applications.

The Proof-of-Work (PoW) algorithm used in most famous BC systems, exposed high consumption of energy for the system to maintain its chain Becker et al. (2013). This motivated the proposal of other proof-based variants for BC applications Nguyen and Kim (2018), to mention some: Proof-of-Stack (PoS) King and Nadal (2014), Proof-of-Contribution (PoCot) Xue et al. (2018), Proof-of-Luck Milutinovic, He, Wu, and Kanwal (2016), Proof-of-Concept (PoC) Ak and Canberk (2019), and Proof-of-Schedule (PoSch) Wilczyński and Kołodziej (2020). All of those, and others, require miner nodes to solve/contribute a challenge, yet the challenge is way less energy consuming compared to PoW.

The trust model provided by BC proof-based systems, along with the availability of different variants of proof-based algorithms, encouraged the embodiment of BC for different applications that require high level of reliability and immutability, such as data management systems as in Li, Zhu, and Lin (2018), reputation management systems as in Debe, Salah, Rehman, and Svetinovic (2019), e-voting as in Kshetri and Voas (2018), and access control systems as in Zhu and Badr (2018).

The high trust the PoW algorithm provides, increased the expectations of BC to be the future of financial systems, as discussed in Singh and Singh (2016), and IoT services, as discussed in Samaniego et al. (2016). However, few proposed the integration of BC and FC, some of which proposed deploying BC in the end user's layer as in Tang, Zhao, Rafique, and Dou (2018), others in the fog layer as in Samaniego et al. (2016), and most of them in the cloud layer as in Xiong et al. (2018).

3. Modeling the proposed PF-BTS system

3.1. Blockchain modeling

Blockchain (BC) is a distributed ledger technology that enables saving data, whose validity must be agreed on by BC miners, in a fully distributed manner Swan (2015). Through the last few years, BC miner nodes had boosted in terms of computational power, due to high benefits rewarded for mining a new block, and the fast advancement of technological solutions of Graphical Processing Units

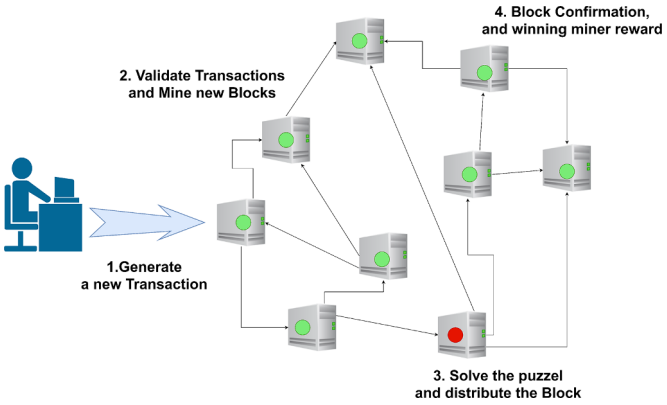


Fig. 1. Steps of generating a new block in a BC system.

(GPUs). BC has been proven to provide secure and trusted transaction (TX) validation and confirmation without the interference of a Trusted Third Party (TTP).

A BC miner is, in most cases, a rather fast computer that is able to relatively conduct huge number of computational tasks per second Crosby, Pattanayak, Verma, Kalyanaraman et al. (2016). When Bitcoin was firstly introduced in 2008 Nakamoto (2019), BC was the foundation of a peer-to-peer network, where peers are challenged to solve a mathematical problem using a Brute-Force approach. As demonstrated in Fig. 1, once a BC node reaches a solution for the problem, also named puzzle, the solution is broadcast among other peers for confirmation Zheng, Xie, Dai, Chen, and Wang (2018). Reaching a consensus of 51% or more on the validity of the solution, as well as on other information related to the puzzle and the performed TXs by end users, results in confirming the block of TXs and adding it to a publicly available ledger Vukolić (2015).

The miner node who solves the puzzle is then awarded by the network and by the end users whose TXs were confirmed. The difficulty of the puzzle is conditionally determined Kraft (2016). For example, Bitcoin requires one block to be confirmed every 10 minutes, while Ethereum average block time is 15 seconds Ethereum (2020). Hence, if a BC miner is able to solve the puzzle in less time, the algorithm increases the difficulty so that the average time of puzzle solution, and hence the block generation, remains Garay, Kiayias, and Leonardos (2017). This mentioned approach for reaching a consensus is what PoW actually is; the winning miner must prove its work on the puzzle by providing the solution, in order to get rewarded.

Bitcoin is a cryptocurrency secured by a PoW-based BC, where money TXs, merchandise and exchange can be performed. Ethereum, on the other hand, is a PoW-based BC that can handle wider range of applications, including money transfer. Specifically, Ethereum provides a platform where on-line Smart Contracts (SC) can be generated and agreed-on by different users. Those contracts are conditionally automated providing higher trust measures for sensitive applications. Further, the smartness of these contracts is highly flexible as users write and agree on the terms of each contract, hence each contract is different than other contracts and users are not forced to play by any body's rules. The need of SCs does not appear in Bitcoin, or similar cryptocurrency systems, as the rules and terms are unified for all TXs. Although there is no SCs in Bitcoin, we can still project it, as a concept, on its workflow. That is, a user generates a TX to another user, hence she is generating a SC in which a certain amount of digital coins are unconditionally transferred to another user. Miner nodes pick this TX according to some criteria, such as the preferred TX fees waived by the first user, and accumulate many TXs in one block. Once a block reaches a completeness status (configured by the system) such as specific number of TXs per block, the block is mined (i.e. hashed and the nonce is found in a Brute-Force manner). Consequently, the first miner node to find the nonce broadcasts the new block with its nonce and all network peers validate the correctness of the solution and add the block to the locally saved chain. Once the block is added to the chain, all TXs within are confirmed and hence the amount of the sender's digital coins is decreased while the amount of the receiver's digital coins is increased.

In Ethereum, a SC does not have to include transferring digital coins to another user, although it can. That is, all the above mentioned steps apply in Ethereum, except that users actually write a code, which is the SC, that is run on one Ethereum node. The SC can include any type of implementation, and the user who generates this SC provides TX fees that is given for the node who runs the SC code. Depending on how power consuming the SC code is, and fees decided by the system per each computational cycle, the TX fee is decided. This is why the TX fees in Ethereum are called GAS, as the node that runs the code consumes fees as much as needed to compensate its power consumption for running the code. However, GAS coupled with each SC is also flexible as it is decided by the user, not the system. If GAS was sufficient to perform the SC, it is completely and successfully run and the unused GAS is returned to the generator. Thus, the SC is run as long as the GAS is available. If the GAS was consumed before the SC is finished, then the SC is terminated, and the TX is not conducted, yet the provided GAS is waived to the node who ran the code.

In our present research, we propose exploiting BC miners (e.g. peers of Ethereum Blockchain), who usually perform computational puzzle solving in order to receive digital coins from the BC network. By generating a SC to the platform, we aim to exploit BC nodes to assign end users' task into the cloud VRs. We deploy the fog nodes to control the communication among end users, BC network, and the cloud. The fog node in our proposed system is responsible for choosing the best schedule according to specific criteria, which reduces the latency compared to the case of BC network generating the best schedule by consensus. That is, a BC node picks the SC generated by the fog node, and sends its proposed schedule back to the fog node. The fog node then chooses the best

schedule among the received ones and sends it to the cloud. The cloud accordingly performs the tasks using the assigned VRs and returns the results to the fog node, who forwards them to end users. The direct incentive for BC miners who run the SCs is provided in the form of GAS, while incentives for generating new blocks are given by the BC network itself. Notably, the results are saved on the chain by BC peers consensus (e.g. PoW in the case of Ethereum) for future reference and analysis. However, the time needed for reaching a consensus in order to save the data on the chain is not included in the time required to generate the optimal schedule. Thus, the type of consensus algorithm deployed in the BC network does not affect the latency. If some related information are needed, by the system administrators, the cloud, or the fog, it can be easily retrieved as the result of each SC is immutably saved on the chain.

To evaluate our proposed system, we adopt two different approaches of parallel computing Stauffer, Hehl, Ito, Winkelmann, and Zabolitzky (2012). The first is the Single Instruction Multiple Data (SIMD), used to enforce BC miners who pick the generated-by-fog SCs to run the required code and computations. The second is the Multiple Instruction Multiple Data (MIMD), which we use to simulate the Blockchain Scheduling (BS) system proposed in Wilczyński and Kołodziej (2020), for evaluating our results by comparison. To clarify, SIMD enforces simulated miner nodes to run the similar SCs that hold the same code (i.e. ACO in our proposed system), yet each of them provides different output. The TPC then recognizes the best schedule proposed from the SIMD-enabled miners. Finally, each MIMD-enabled miner runs different SC with different code, resulting in different outputs. The outputs of the two groups are compared and the best result is sent to the cloud. Nevertheless, all outputs are saved on the chain for future reference. Originally, miner nodes in BS were grouped into four categories, each group ran a different algorithm with the same input. They proposed the Proof-of-Schedule (PoSch) algorithm using Game Theory, in order to obtain the best assignment from the four groups. Some miners in BS provide the same results however, so we believe there was no need to recruit more than one miner in that group. For example, the SJF algorithm was run on four miner nodes, yet they all shall, by definition, output the same exact result. This also applies for the FCFS and RR algorithms as, initially, all miner nodes receive the same input, all tasks have the same arrival time, and all miner nodes use the same quantum value. Following these mentioned insights, we performed our experimentation on eight virtual machines, five of them ran the ACO assignment optimization presented in the following subsection, while the other three performed SJF, FCFS, and RR assignment. We excluded the fourth group studied in Wilczyński and Kołodziej (2020) since it presented the worst performance in most cases that were originally evaluated in the BS proposal.

3.2. Ant colony optimization (ACO) and problem statement

Algorithms of Meta-heuristics are the state of the art solution for complex optimization problems. Most of these algorithms are so-called ænature-inspired algorithmsg Saka and Dogan (2012). Specifically, Ant Colony Optimization (ACO) Dorigo et al. (2006) is an optimization algorithm that is motivated by the behaviour of some type of ants in real-life. Ants of the colony communicate by depositing chemical pheromones along the paths they take from home colony to food sources. Such behaviour provokes other ants of the colony to follow the paths that are most recommended by the majority of ants, indicated by high levels of pheromones deposited. Implementing this nature-inspired methodology was shown to provide high ability of solving different versions of the assignment problem Chen and Cheng (2005); Srikanth, Maheswari, Palaniswami, and Siromoney (2016). There are several proposed versions of ACO depending on the problem to be solved. Generally, ACO works over the following phases:

1. Construct Ant Solutions: each ant stochastically constructs its solution from a finite set of available solutions.
2. Update Pheromone: In this phase the pheromone values are modified to indicate promising/bad solutions. Moreover, all pheromone values are equally decreased every predefined number of iterations due to the pheromone evaporation concept.
3. The final solution is biased by the pheromone values deposited by all ants.

We adopted a Multi-Objective ACO with Global Pheromone Evaluation, improved by a greedy optimal assignment approach to get even better solutions. That is, once an ant constructed a solution, it improves the proposed solution through local greedy search, which results in the Best Individual Assignment (BIA). Consequently, BIA is updated through the run, and the assignment obtained depending on the pheromone matrix is compared to the BIA. The best of the two is then proposed as the final solution. Following, we will describe the version we adopted, its equations, our proposed improvement, and final solution algorithm. It is worth noting that what we present in this subsection is how each BC miner performs in PF-BTS. On the higher level of PF-BTS, however, our optimization model is inspired by the Multi-Colony Parallel ACO described in Pedemonte, Nesmachnow, and Cancela (2011). The problem that needs to be solved by our task scheduling protocol, is the assignment of n tasks, denoted by T_i where $i = 1...n$, to m machines denoted by M_j where $j = 1. ..m$, such that each task is assigned to one machine. Our objective function is to find a schedule where the time consumption, required by the cloud VRs to perform the computational tasks, is minimal. The utilization of the machines, denoted by u_{ij}, relates to two parameters. First, the computing power of M_j, measured by MFLOPS (Million FLoating point Operations Performed in one Second). Second, the length of T_i, which is the number of operations or instructions needed to execute the task T_i expressed in MFLO (milion floating point operations). The utilization values of a machine performing any task is calculated using Eq. 1.

$$u_{ij} = \frac{T_i(MFLO)}{M_j(MFLOPS)} \tag{1}$$

Based on these values the algorithm constructs the utilization matrix U whose size is $n \times m$. The number of rows equals the number of tasks, while the number of columns equals the number of machines. The element of utilization matrix is real numbers. A sample of the utilization matrix is presented in Table 1.

Table 1
Utilization matrix for 3 tasks and 3 machines.

	M1	M2	M3
T1	u_{11}	u_{12}	u_{13}
T2	u_{21}	u_{22}	u_{23}
T3	u_{31}	u_{32}	u_{33}

The scheduling can be presented using a $n \times m$ binary matrix called schedule matrix S whose entries are denoted by s_{ij}. That is, $s_{ij} = 1$ if task i is assigned to machine j, and equals zero otherwise. Our algorithm makes sure that no more than one '1' is in the same row to make sure that each task is assigned for exactly one machine. However, there can be several tasks performed by one machine, depending on the best assignment criteria. This means that there are no restrictions on the number of 1's in any given column. The reason behind implementing our algorithm as for each task is assigned to exactly one machine, is that assigning a task to multiple machines complicates the problem even more, which is highly inefficient in terms of time and energy. That is, the state space under our mentioned condition is decreased from $n!$ to m^n, hence the optimum assignment is more likely to appear during the search (for huge number of iterations and ants). Further, the Single-Task-Multi-VR approach is doable only if tasks were partition-able, which is a condition that may not apply in all cases. However, if for some reason this was necessary, we believe our model and its algorithm can be easily developed to handle it using a multi-threading scenario.

The global pheromone matrix is also represented using a $n \times m$ matrix. The elements of the pheromone matrix called pheromone values, denoted by τ_{ij}. The pheromone values indicate the preference of assigning specific task on a specific machine. τ_{ij} in the pheromone matrix is conditionally affected by a probability of $p(i, j)$. That is, higher probability of performing task i on machine j is obtained for assignments that were recommended by more ants. Initially, all pheromone values are equal. During the iterations of the algorithm, the pheromone matrix is updated by each ant. The assignment of T_i to M_j, by an ant who just finished its iterations, increases the values of τ_{ij} according to Eqs. 2 and 3. Also these values are affected, during run time, by an evaporation factor ρ after each ant finishes its iterations. That is, the pheromone is decreased relatively to ρ to simulate the pheromone evaporation over time. For example, if $\rho = 0.6$ is the evaporation factor, then the evaporation rate is $1 - \rho$ which equals to 0.4. Accordingly, all elements in the pheromone matrix is decreased using Eq. 4.

$$p(i, j) = \frac{\tau(i, j)}{\sum_{j=0}^{m} \tau(i, j)} \tag{2}$$

$$\tau_{ij} = 1 + \tau_{ij} + \frac{p(i, j)}{NumberOfAnts} \tag{3}$$

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} \tag{4}$$

In our version of ACO, the first ant creates a schedule, so–called LOCAL, where each task is randomly assigned to a machine. After that the ant iterates a number of modifications, at which each task is assigned to a different machine, while other tasks remain assigned to their original machines as in LOCAL. Each new modified LOCAL is compared to the original random assignment. If an iteration resulted in a better schedule, in terms of machines maximum execution time, LOCAL is replaced with the better schedule. The evaluation is done by comparing the values of schedule usage, which is the total time consumption of VRs to perform the tasks. To get the Time Consumption R, the utilization matrix is multiplied by the schedule matrix (element-wise multiplication), which outputs a $n \times m$ matrix. After each ant finishes its iterations, the ant's best assignment is compared with the best assignment obtained from all previous ants. This comparison is beneficial later for the comparison between the best individual assignment (BIA) with the assignment obtained depending on the pheromone matrix.

Usually, after all ants finish their iterations, the pheromone matrix becomes ready to extract an assignment that the majority of the ants agreed on. That is, the maximum value of each row 'i' indicates that most ants recommended assigning task T_i to the correlated M_j. However, our aim is obtaining the best possible assignment of the tasks. Hence, we improved the ACO algorithm a bit, using a simple Greedy approach, to compare the BIA proposed by an individual ant with the assignment proposed by the majority of ants. The algorithm then chooses the best of the two and considers it the output of the algorithm. Eq. 5 presents how the algorithm computes the time consumption of tasks in a proposed assignment. Eqs. 6 and 7 clarify how the algorithm extracts the schedule's time consumption, and the estimate cost, respectively. Algorithm 1 illustrates the workflow of our version of ACO.

$$R = U \times S \tag{5}$$

$$Usage = \sum_{i=0}^{n} \sum_{j=0}^{m} R[i, j] \tag{6}$$

$$Cost = \frac{Usage \times one\_min\_cost\_of\_machine}{60} \tag{7}$$

Result: Best Assignment of *n* Tasks onto *m* VRs
Initialization: Construct U using Equation 1
for *i in range(NumberOfAnts)* **do**
  Randomly generate LOCAL matrix;
  for *iter in range(NumberOfIterations)* **do**
    Slightly modify LOCAL;
    if *Usage(modified LOCAL) < Usage(LOCAL)* **then**
      LOCAL = modified LOCAL;
    end
  end
  Apply Evaporation using Equation 4
  if *Usage(LOCAL) < Usage(BIA)* **then**
    BIA = LOCAL
  end
  for *t in range(n)* **do**
    for *p in range(m)* **do**
      if *Local[t,p] == 1* **then**
        Update τ_{tp} using equation 3
      end
    end
  end
end
for *y in range(n)* **do**
  for *z in range(m)* **do**
    if *Pheromone Matrix[y,z] is the max in row 'y'* **then**
      S _Pheromone[y,z]=1
    else
      S _Pheromone[y,z]=0
    end
  end
end
R_Pheromone = U × S _Pheromone ;
R_BIA = U × BIA;
if *Usage(R_Pheromone) < Usage(R_BIA) AND Cost(R_Pheromone) < Cost(R_BIA)* **then**
  Best Assignment = S_Pheromone
else
  Best Assignment = S_BIA
end
Return Best Assignment

Algorithm 1. ACO algorithm.

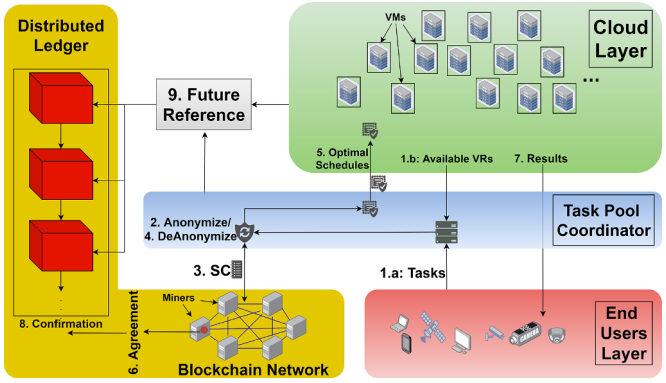


Fig. 2. The Fog-Cloud architecture in which the fog and BC are deployed to perform the proposed PF-BTS protocol.

3.3. System characteristics and protocol

Next, we present our proposed deployment of ACO algorithm in BC miners, for the benefit of a fog-enabled environment. The ACO algorithm, implemented in a form of a SC, optimizes the assignment of tasks to the available cloud VRs, while fog nodes control the process and guarantees the anonymity of end-users and tasks' information. Fig. 2 clarifies the framework of our proposed protocol in a FC architecture, while Algorithm 2 clarifies how the fog node enforces the needed measures to preserve the privacy of end-users, and controls the communications between end-users, BC miners, and the cloud.

The key steps of the proposed protocol are as follows:

1. End-users send their computation tasks to the fog node they are authenticated with.
2. Fog nodes are the Task Pool Coordinator (TPC) component in our protocol. Fog nodes receive the tasks from end-users, and receive information about the capacities of available VRs from the cloud.
3. Unless the TPC is capable of performing the computational tasks, it randomly assigns each task, or group of tasks, a unique ID (anonymization). The abilities of the TPC to perform the tasks is determined by different criteria, such as the CPU computational power, the length of each task, the network load balancing, etc.
4. After that, the TPC generates a number of SCs into the BC network which include: the optimization algorithm (i.e. ACO), a list of tasks IDs each coupled with its task length, a list of VRs capacities, the public address of the fog node, and some appropriate amount of GAS.
5. As the SCs are successfully generated, each SC is picked and run by a BC miner. According to a defined criteria (e.g. the least time consumption/ the least cost) each miner proposes a different optimal schedule that is immediately sent back to the TPC. All miners are expected to finish their iterations and send the results at approximately the same time because the SIMD principle applies.
6. Each time the TPC receives a better schedule than a previously proposed one, the best assignment is updated.
7. TPC waits for a pre-determined time to check if it receives a better schedule. If the time passes and no other better schedule appears, the tasks are deanonymized and sent to the cloud. Otherwise, TPC replaces the schedule with the better one and waits again. However, if the number of received schedules equals the number of generated SCs, the best is chosen and sent to the cloud regardless of the waiting time.
8. After the cloud receives the optimal schedule from TPC, the cloud performs the tasks and sends the results back to the TPC, which forwards the results back to end-users.
9. Later, the results of the SCs are saved On-Chain using any consensus algorithm (e.g. PoW in the case of Ethereum) for future reference. Those saved results are immutable and private. That is, all information are anonymized and can not be used/analyzed from any party other than the trusted ones (i.e. the cloud and TPC).

4. Evaluation

In this section we evaluate our proposed protocol in terms of Privacy, Total Execution Time, and Network Load.

4.1. Privacy

Privacy issues are considered a challenge when it comes to BC-IoT integration Tseng, Wong, Otoum, Aloqaily, and Othman (2020). Both the layer at which BC is deployed, and the purpose of the deployment affect the privacy measures differently in different cases. Consequently, every integration case needs to be evaluated individually, especially that BC technology itself is not yet standardized, nor privacy measures are unified for a BC-Fog integration Baniata and Kertész (2020). In PF-BTS, SCs generated into the BC network by the TPC expose only the expected computational power needed for each task/group of tasks (i.e. task length). The identities of end-user devices, and the nature of the tasks/applications performed by end-users are anonymous. Further, usage privacy

Result: Assign computational tasks (*Ts*) to the available Virtual Resources (*VRs*)
Initialization: *P*=Waiting time;
A=Maximum *T* to be handled by TPC;
Receive *Ts* from end users;
Receive the *VRs'* computational power from the cloud;
if *T* <= *A* **then**
    Randomly assign each T a unique ID;
    Generate a SC that assigns anonymous Ts to available VRs using ACO optimization;
else
    Perform T;
    Send the results back to end-users;
end
if *Best Assignment 'R' is received from BC* **then**
    while *i* <*p* **do**
        if *a new Best Assignment 'N' is received AND N's execution time is less than R's* **then**
            R = N;
            i = 0
        end
    end
    De-Anonymize *Ts*;
    Send R to the cloud;
end

Algorithm 2. TPC Workflow.

and location privacy are also guaranteed by the protocol, since no information included in the SC can be related to the identities or location of end-users. Similar approaches were proven effective, in terms of privacy, in Yi, Qin, and Li (2015) and Ni, Zhang, Lin, and Shen (2017).

To systematically validate PF-BTS in terms of privacy, we model the probable attacks that can be launched against End-Users using the Attack Trees modelling Schneier (2015). We assume that fog components and cloud components are trusted but curious components. That is, fog and cloud components can be trusted with data such as timestamps and end-user identities/applications. However, the discussion of the trust in the privacy measures provided by the fog and the cloud are out of the scope of this paper, hence, we recommend referring to Chen, Takabi, and Le-Khac (2019) and Khalid et al. (2019) for details. It is worth noting, though, that malicious BC nodes may misuse the public addresses of fog nodes. Yet,this cannot expose any location or usage data of end users. Simply because such data can only be obtained if the fog node was hackable, which contradicts our initial assumption about fog nodes being trusted and secure entities.

Consequently, our goal here is to prove that no private information about End-Users can leak to/be inferred by BC miners. It is worth adding here that end users are not connected to the blockchain (BC) nor to BC miners. End users are only connected and authenticated with the fog nodes. Further, fog nodes are not authenticated with the BC nor with the miners as well. That is, fog nodes in PF-BTS are simply *users* of the BC system. To clarify, in a typical public and permission-less BC system (such as Ethereum), users of the BC system (i.e. the fog nodes in PF-BTS) are not required to be authenticated with any entity of the BC network. All users (fog nodes) need to do is to have a BC-readable wallet that is defined using a public key, and a private key, and contains some amount of digital coins so that the user can pay for the smart contracts (SC) Gas. PF-BTS proposes that end users send their computational tasks to the fog nodes they are -classically- authenticated with (such authentication is beyond the scope of our proposal). The fog nodes generate SCs to the BC network as if the fog nodes were users of the BC (using the automation mechanisms clarified in algorithm 2). After that, the BC miners run the SCs (perform the computational tasks), send the results back to the generator of the SC (which is the fog node), and add the result of the SC to the public ledger. The results being sent to the generator of the SC uses a simple encrypted messaging service over the IP. This messaging service is available in most applicable BCs (including Ethereum). Further, Ethereum provides an even more exclusive approach to explore data fields in SCs (i.e. using the etherscan.io platform). Consequently, only the public address of the fog node can be read by a BC miner (for the messaging), and the public wallet address of the fog nodes (for the Gas).

Privacy is usually studied w.r.t. four main concerns, namely Data, Location, Identity, and Usage Patterns. By referring to the architecture presented in Fig. 2 and the protocol description discussed in subsection 3.3, we can build the privacy Attack Trees.

An Attack Tree is a rooted directed acyclic bundle graph, where the grand root represents the attack and its children represents sub attacks/components that need to be conducted to have a successful breach into the grand root attack. A bundle is the connections from the grand root to a multi-set of nodes. Meanwhile, a child may be either an AND-Node or an OR-Node. For a successful breach, AND-Nodes need to be sequentially/simultaneously conducted, while one OR-Node is sufficient. The math behind complex Attack Trees is somewhat complicated so we skip such mathematical representation as our case is not considered complex. More details, however, can be found in Mauw and Oostdijk (2005).

In our case, let C denote a set of attack components where an attack is a finite non-empty multi-set of C and an attack suite is a finite set of attacks. Following these definitions, we can formulate four sets of attack components, each for one of the aforementioned probable privacy threats. That is, C_{Data}, $C_{Location}$, $C_{Identity}$, and C_{Usage}, denotes attack components required to breach Data, Location, Identity, and Usage Patterns Privacy, respectively.

4.1.1. Data privacy

Data that may be considered private to end-users is data that can be related to certain users/organizations, such that exposing it would cause legal/personal issues to the users. Examples of data that end-users need not expose to any non-trusted entities are the used applications, the nature of the problem to be solved, the type of end-user devices/protocols used for communications and analysis, etc.

As the fog component, namely TPC, is middling between end-users and miners, BC entities can not directly read any information provided to the TPC by end-users unless they have been sent it to the BC network. However, an indirect approach is expected from malicious BC miners in trials to infer end-users' data from the SCs. For example, if the BC miner knows that a specific end-user asks for a named service/application each day at 11am, then some operational conclusions can be drawn about that end-user. However, multiple pieces of information need to be traced for a successful data privacy breach.

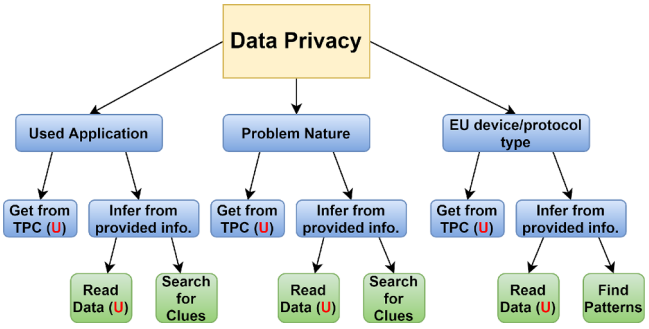


Fig. 3. Attack Tree for Data Privacy probable breaches. Blue = OR-Nodes, Green = AND-Nodes, U = Unavailable.

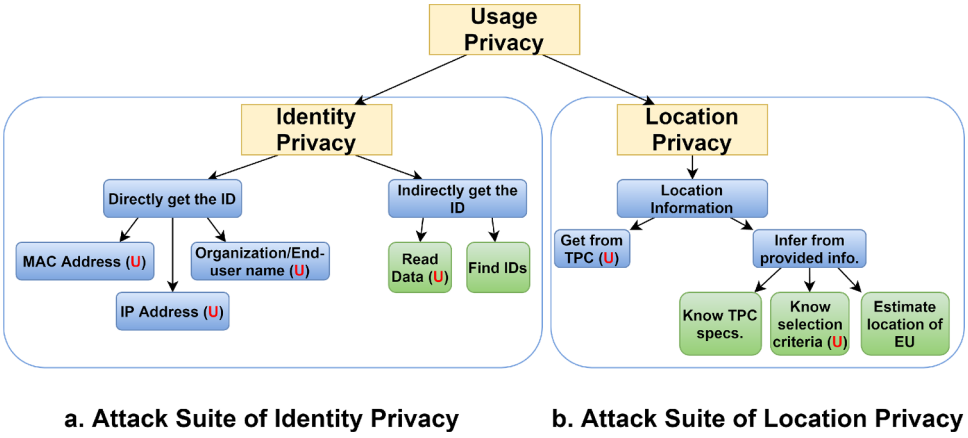


Fig. 4. Attack Tree for Usage Privacy (including the Attack Suites of Location and Identity Privacy) probable breaches. Blue = OR-Nodes, Green = AND-Nodes, U = Unavailable.

Specifically, we need to assure that used applications, problem nature, and end-user device/communication types are not exposed. As represented in Fig. 3, these three end-user sensitive data can be exposed either directly or indirectly. By recursively *Walking* on this attack tree, in all Bottom-Up possible paths, we notice that BC entities can not breach data privacy of end users. For example, a malicious BC miner needs to read the task AND search for clues about the application in which the task is being used. As the first component of the "Infer from provided info." sub-attack is UNAVAILABLE, this sub-attack can not be launched. Hence, the "Used Application" sensitive data can not be inferred, leading to proven Data Privacy from this path.

Similarly, we can see that end-user's Data Privacy can not be breached by walking on any other path. Thus, Data Privacy is guaranteed in PF-BTS.

4.1.2. Identity privacy

This type of privacy considers any information that may be related to any type of identity. Such identities may include the MAC address, IP Address, End-User/Organization name, etc. The identity anonymity is highly important as the NIST computer security handbook Guttman and Roback (1995) defines the Privacy as "a confidentiality term which assures that individuals control the influence of what information *related to them* may be collected and stored, and by whom and to whom that information may be disclosed". Practically, many types of data are useless unless coupled with the identity of its generator/owner. For example, sensitive military information are useless without recognizing what military is generating them. Consequently, we need to validate PF-BTS in terms of Identity privacy. That is, any direct or indirect information about the identity of end-users, who are requesting PF-BTS services, must be hidden from BC miners. By referring to the attack suite "*a*" illustrated in Fig. 4, we can notice by walking paths similar to those described in subsection 4.1.1. that no Identity Privacy breaches can be successfully conducted by BC miners against end-users. Thus, PF-BTS is proven to preserve Identity Privacy.

4.1.3. Location privacy

Generally, a discussion related to Location Privacy does not indicate location data that are voluntarily shared by end-users. That is, privacy preserving of such data is discussed under the umbrella of Data Privacy. Rather, Location Privacy is the term used to indicate that no end-users' location information can be inferred using their behaviour or involuntarily provided location by device's sensors. For example, even when end-users turn off all gyroscopes or location sensors in their devices, their location can still be inferred by noticing the selection criteria of communicated Access Points (AP)/fog nodes. To estimate the location of an end-user, if we know that their device connects to the closest AP/TPC, while we also know the specifications of the AP/TPC (i.e. signal strength), then we can infer that the end-user is located within a circle around the AP/TPC whose radius is known. Further, we can track the path that an end-user walks in the spacial domain by using the same above information Butt, Iqbal, Salah, Aloqaily, and Jararweh (2019). However, unless end-users voluntarily share their location data, such privacy breaches require multiple components to be successful. Accordingly, some approaches to defend against such probable Location Privacy breaches were proposed in the literature. Examples include using different selection criteria of APs/Fog nodes Ni et al. (2017), using obfuscation algorithms Yi et al. (2015), and deploying fake APs/Fog components Dong, Ota, and Liu (2015).

For the present case of PF-BTS, we need to assure that, First, the location data provided by end-users to TPC is not exposed to the BC network. Second, TPC selection criteria by end-users is not known to BC network. Accordingly, we present the Location Privacy Attack Tree illustrated in the attack suite "*b*" of Fig. 4. Similar to the paths described in subsection 4.1.1, we can notice that a malicious BC entity can not breach the location privacy of end users. Accordingly, PF-BTS is proven to preserve Location Privacy of end-users.

4.1.4. Usage privacy

This term refers to keeping data about the frequency of data being sent to the APs/TPC, or data about the patterns of using a certain service or application private Baniata, Almobaideen, and Kertesz (2020). Such data may be used by an adversary to infer

sensitive information about end-users. Examples of applications whose usage patterns are not advised to be exposed include smart homes, where the usage patterns, during specified times/periods, of fog/cloud applications may reveal the actual times of home/application functionality. Such information may be valuable for malicious eavesdropping during down times of the facility, leading to different effective security threats. Usage patterns can be inferred by using counters to compute the times and periods of usage, and co-relate them with the intensity of computations, services, or electricity consumption. However, such information are useless without the end-users identities or locations. Since we have already proven the trustworthiness of the PF-BTS in preserving the Location and Identity Privacy measures, it is trivial to conclude that PF-BTS can be trusted to provide accountable Usage Privacy measures. This is illustrated in Fig. 4.

4.2. Total execution time of virtual resources

It is proven that the BS scheduling technique, proposed in Wilczyński and Kołodziej (2020), is able to provide more optimal solutions than FCFS, SJF, and RR approaches to problems similar to our problem. Consequently, for 't' tasks to be performed on 'v' resources, BC miners providing assignment schedules of t into v, provide better assignment solution than using FCFS, SJF, or RR alone. Moreover, ACO algorithm performs better than a random assignment on a single computer as shown in Reimann and Leal (2013); Rugwiro, Gu, and Ding (2019), and performs better than FCFS and RR as shown in Kavitha and Sharma (2019); Tawfeek, El-Sisi, Keshk, and Torkey (2013). However, the results presented in Gupta, Kumar, and Sidhu (2017) and Nosheen, Bibi, and Khan (2013) indicated an equivalency of performance of ACO compared to SJF in most cases. Depending on these results, along with the transitivity relation of the two homogeneous problems, we can logically draw some conclusions of an expected superiority of PF-BTS protocol over BS. This is because BS takes the best proposed assignment output from different miner nodes that perform FCFS, SJF, HSGA, and RR. The validity we will be seeking later is that PF-BTS outperforms BS in providing a more optimal schedule with respect to the computing cycles performed by the same tested tasks and VRs, or in providing equivalently optimal schedule using less resources. We logically present our argument as follows:

Definition:

An Optimal Schedule ($OS_{t,\ v}$) is the best assignment of requested computational tasks 't' to be performed on a set of available VRs 'v', in terms of minimal execution time of 'v' to perform 't'.

Hypothesis:

$OS_{PF-BTS(t,v)} < OS_{BS(t,v)}$

proof:

$OS_{ACO(t,\ v)} < OS_{Random(t,\ v)}$

$OS_{ACO(t,v)} <= Min(OS_{FCFS,SJF,RR(t,v)})$

$OS_{BS(t,v)} = Min(OS_{FCFS,SJF,RR(t,v)})$

$OS_{PF-BTS(t,v)} < OS_{ACO(t,v)}$

$\therefore OS_{PF-BTS(t,v)} < OS_{BS(t,v)}$

To experimentally prove our expectations about PF-BTS outperforming the BS model, we implemented a simulation environment using Python 3.8 programming language[1]. We ran our code on Google Cloud Platform (GCP) using a C2-standard-8 (8 vCPUs, 3.8 GHz, 32 GB memory). In our experiment, we dedicated the first core for the FCFS computations, the second for the SJF computations, the third for the RR computations, and the remaining five cores were dedicated to the ACO computations. As mentioned in Subsection 3.1, there is no need to recruit more than one miner handling each of the scheduling algorithms of BS (i.e. FCFS, SJF, and RR). A note needs to be dropped here, that SCs can be implemented using any language specified for the used BC platform. In Ethereum, for instance, SCs are implemented using the Solidity programming language (which is very similar to Python and JavaScript), while in a Hyperledger Fabric based BC platforms, SCs are implemented using GO, node.js, or Java. However, SCs in PF-BTS are represented by the modules named (ant_alg.py, first_come_first.py, round_robin.py, and short_job_first.py) implemented using Python and can be found in the publicly available Github repository. Only because the whole simulation code was implemented using python, the SCs in our simulation used the same language so that complexity is avoided. That is, the TPC receives the computational tasks from end users, anonymize them, adds them to the templates provided in the mentioned modules (now we have SCs automatically implemented by the TPC), and finally generates them into the BC. Accordingly, the implementation of the SCs is already available in Python. If PF-BTS needs to be implemented using other language, same principles apply as in our version. For PF-BTS suggests that the Ant Colony Optimization be used, the general algorithm of the ant_alg.py is provided in algorithm 1.

As it is clarified in Fig. 5, our simulation initialized the constants, variables, lists, and matrices. Then, if the TPC was not capable of performing the tasks according to certain conditions, the tasks are anonymized and a number of SCs are generated into the BC network. The network, which in our simulation consists of 8 miners, is grouped into four different groups categorized according to the code contained in the SC they picked. Groups A, B, C, and D are miners who picked SCs with FCFS, SJF, RR, and ACO codes, respectively. Each miner runs the SC and sends its result to TPC, where the best assignment among received results from Groups A, B and C is considered the BS result. The best assignment of received results from Group D is considered the PF-BTS result. The results of

[1] https://github.com/HamzaBaniata/PF-BTS

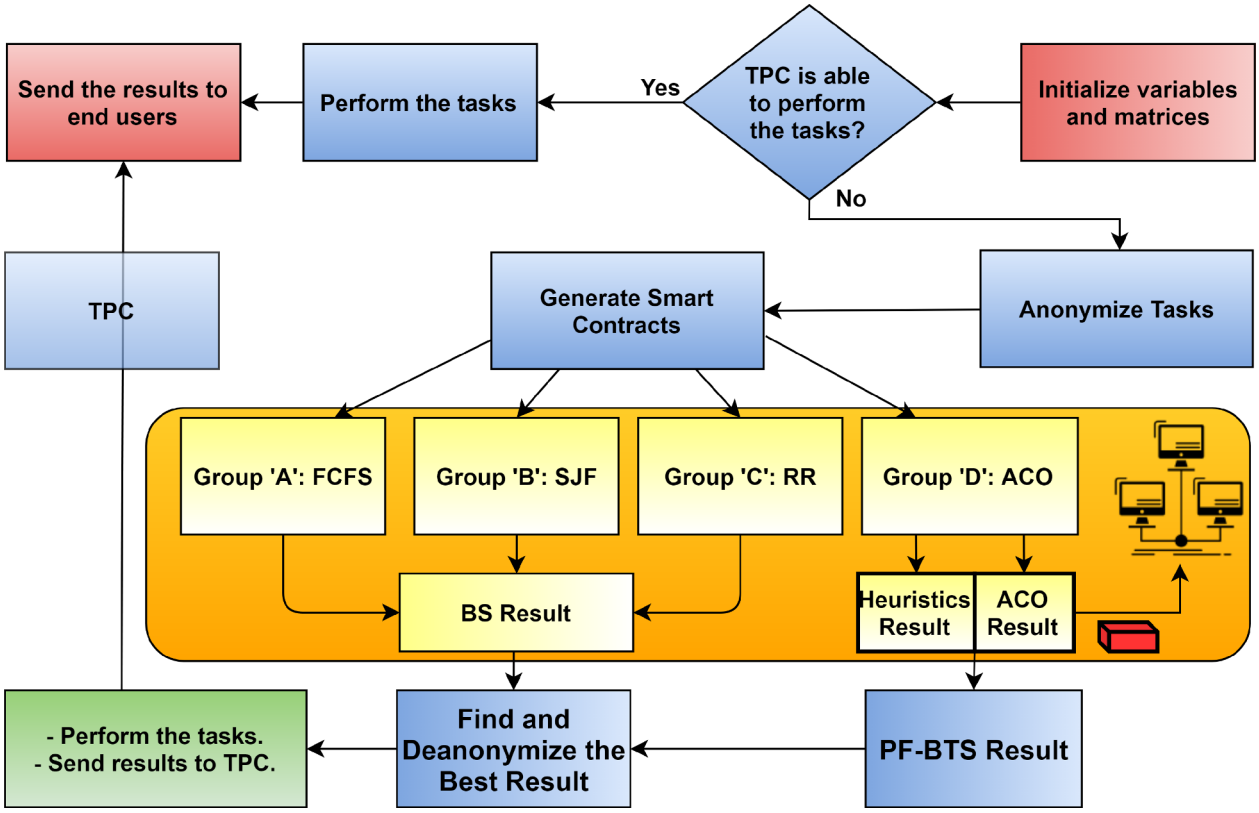


Fig. 5. The simulation workflow of our experimentation.

the SCs of all groups, however, are pooled to be saved on the distributed, publicly available ledger for future references. The TPC compares BS result with PF-BTS result, re-acknowledges the tasks and sends the best assignment to the cloud. Finally, the cloud VMs perform the tasks to the received OS and sends the computation results back to the TPC, who forwards them back to end users. To ease the mapping of Fig. 5 into the architecture presented in Fig. 2, we color-coded the nodes in those figures such that pink coloured, blue coloured, yellow coloured, and green coloured represents the processes performed in the end users layer, fog layer, BC network, and cloud layer, respectively. The results of comparing BS with PF-BTS are presented in Figs. 6, 7 and 8.

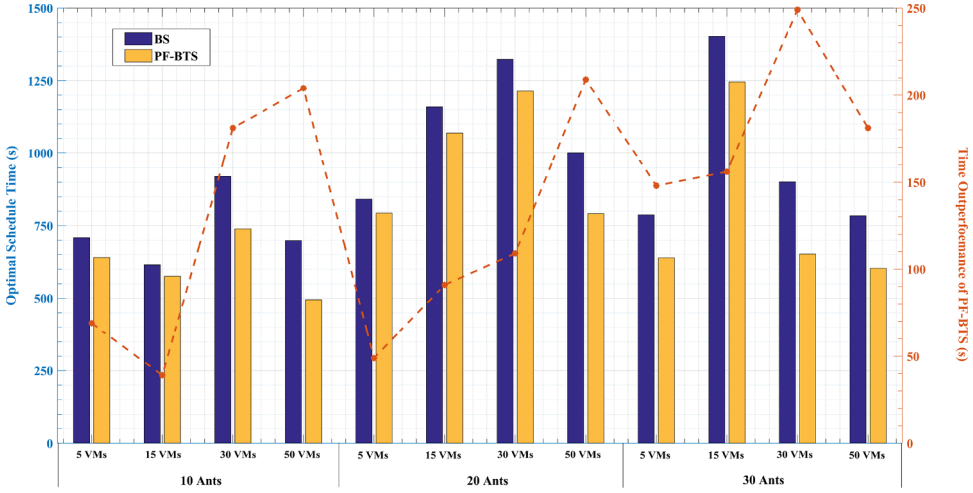


Fig. 6. Time consumption of schedules generated by BS and PF-BTS (with configurations of 10, 20, and 30 Ants) for assigning 5, 15, 30, and 50 VRs to perform 30 tasks. Time consumption of BS and PF-BTS is represented by blue and yellow bars, respectively, correlated with the primary y-axis on the left. Out-performance of PF-BTS over BS, represented by dotted red line, correlated with the secondary y-axis on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

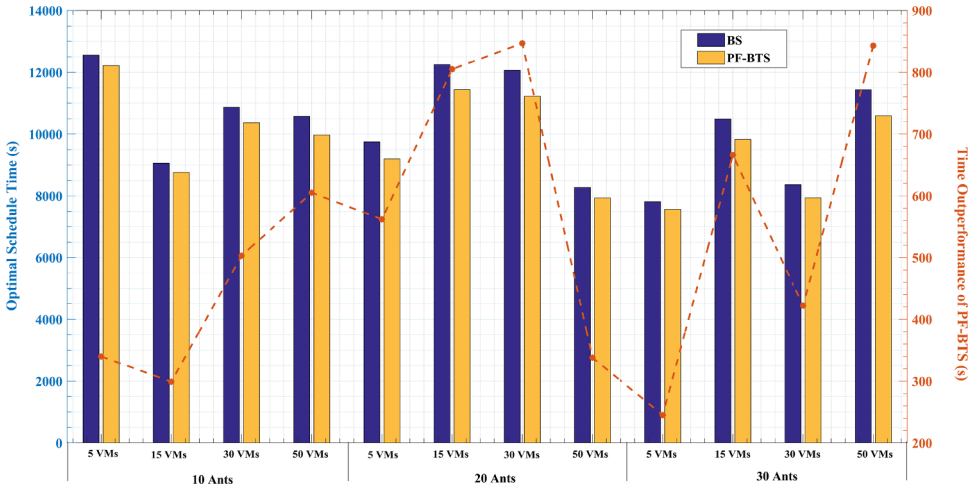


Fig. 7. Time consumption of schedules generated by BS and PF-BTS (with configurations of 10, 20, and 30 Ants) for assigning 5, 15, 30, and 50 VRs to perform 300 tasks. Time consumption of BS and PF-BTS is represented by blue and yellow bars, respectively, correlated with the primary y-axis on the left. Out-performance of PF-BTS over BS, represented by dotted red line, correlated with the secondary y-axis on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

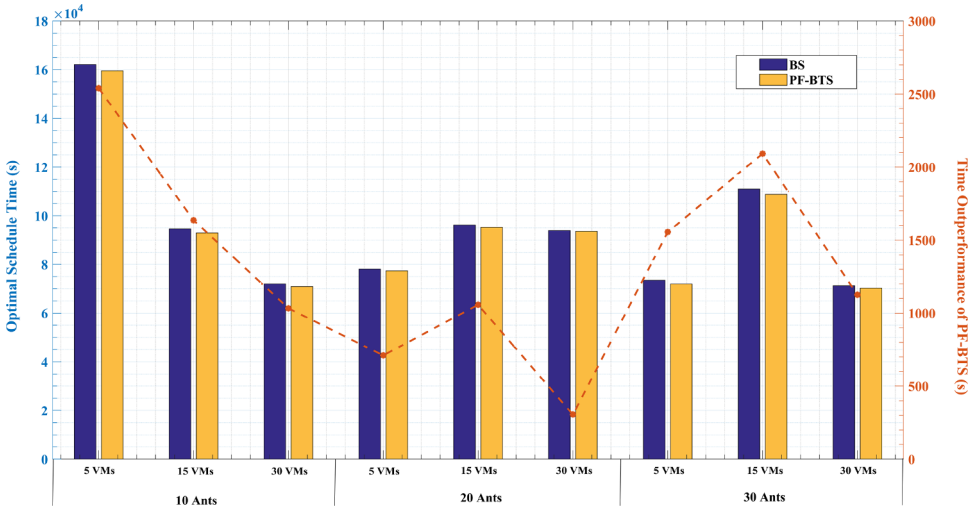


Fig. 8. Time consumption of schedules generated by BS and PF-BTS (with configurations of 10, 20, and 30 Ants) for assigning 5, 15, and 30 VRs to perform 3000 tasks. Time consumption of BS and PF-BTS is represented by blue and yellow bars, respectively, correlated with the primary y-axis on the left. Out-performance of PF-BTS over BS, represented by dotted red line, correlated with the secondary y-axis on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Simulation Parameters

- We conducted 33 simulation runs in groups of 12, 12, and 9 runs, for simulating 30, 300, and 3000 tasks, respectively, that need to be computed by 5, 15, 30, and 50 VRs. Sub-consequently, we configured the number of ants differently in each group. That is, we compared the results of BS with configurations of 10, 20, and 30 ants of PF-BTS.
- In each simulation run, same tasks' length and VRs' computing capacities were input into all generated SCs.
- Each VR is assigned a random computing capacity that ranges from 4 to 48 MFLOPS.
- Each task is assigned a random length that ranges from 100 to 1000 MFLO.
- The TPC was assigned a computing capacity of 20 MFLOPS. Accordingly, a maximum total execution time of generated tasks in TPC was configured to be 80 seconds. If this condition was met, the tasks were performed in the fog layer (i.e. the BC miners, the SCs, and the Cloud are not exploited).
- Evaporation factor is stated as 0.3
- One minute cost of VM is stated as 0.1 cent/minute.

4.3. Network load

In this subsection, we study a case where computational tasks are sent to the cloud by end user devices. Consequently, we compare PF-BTS with BS w.r.t. two criteria, namely the number of messages exchanged M from the moment of generating the tasks until the results are forwarded back to end users, and the time consumed by the BC network to generate its suggested schedule. To provide a comprehensive evaluation, we considered the two states of the TPC: able to perform the tasks, and unable to perform the tasks.

4.3.1. Exchanged messages

The proposed steps in PF-BTS imply that the following messages shall be exchanged. End-user devices send the tasks to the fog node, the messages used for this step are denoted by M_t. The cloud sends the available VRs to the fog node, the messages used for this step are denoted by M_v. The TPC then generates SCs containing the IDs of the tasks, the length of each task/group of tasks, and the available resources. The messages used for this step are denoted by M_i. The BC miners then start the process to produce the optimal schedule. We denote the messages exchanged between miners until reaching a consensus by M_c. In PF-BTS, M_c equals exactly the number of SCs, since each SC sends one message to the TPC. However, M_c in BS increases due to the PoSch consensus algorithm. After the optimal schedule is sent to the cloud by the TPC, which is denoted by M_p, the cloud sends the computation results back to end users with M_r. In contrast, we expect PF-BTS to require a number of exchanged messages that equals $\approx M_{BS} - M_c + 2q$, where q is the number of generated SCs. Consequently, we can conclude the following:

$M_{PF-BTS} <\ ?\ M_{BS}$

$M_{PF-BTS} <\ ?\ M_{PF-BTS} - 2q + M_c$

$\Rightarrow 2q < M_c$

$$\Rightarrow M_{PF-BTS}: \begin{cases} < M_{BS} & \text{if} \quad 2q < M_c \\ = M_{BS} & \text{if} \quad 2q = M_c \\ > M_{BS} & \text{if} \quad 2q > M_c \end{cases}$$

Nevertheless, if the computational tasks were performed in the fog node, a property that is available in PF-BTS, then the required number of exchanged messages will be equal to $\approx M_t + M_r$. This is a significantly less number of exchanged messages, than in BS.

4.3.2. Time needed by BC network to generate the optimal schedule

The time required to confirm the optimal schedule might also be critical for some delay-sensitive applications. In BS, the average time consumption to confirm a randomly generated schedule was 0.07 seconds. However, the presented time measurements aimed to highlight the time needed for the PoSch algorithm to reach a consensus on a proposed random schedule, where only four miner nodes were deployed to assign an average of 3 tasks to an average of 3 VRs. Bearing in mind that more BC miners deployment shall increase the confirmation time, due to the 50% of miners agreement rule of PoSch, the scalability and delay-tolerance properties of BS can be deduced differently for different cases.

Supposing that each miner node in BS has a computing capacity as randomized as the VRs of the cloud, a maximum computing capacity of a miner equals 12 MFLOPS. However, the average computing capacity of public BC miner nodes, as reported in Betelgeusian (2018), ranges from 336 MFLOPS to 18,000,000 MFLOPS. This is due to the deployment of the Graphical Processing Unit (GPU) for performing BC computations. Accordingly, the slowest miner node in PF-BTS performs 28 times faster than the fastest miner node in BS. Further, PF-BTS instantly compares all received schedules in the fog node, hence no consensus time consumption is expected.

To highlight the time, in a PF-BTS system, needed from sending the tasks and VRs capacities to the BC network until getting the schedules back from them, we conducted a different experiment. First, we conducted a simulation run where eight tasks need to be assigned into four VRs, the assignment job was distributed to 16 BC miner nodes (16 SCs were generated) with each running 10 ants. The 16 miners sent their assignment suggestions to the TPC within 0.03 seconds. Obviously, this is a better result than the one gained by BS, despite that there were 16 miner nodes involved instead of 4, and the schedules provided are optimal rather than random. Second, we conducted six other simulation runs where five VRs are expected to execute fifty tasks. We ran the simulation for a number of miner nodes ranging from 5 to 16, each miner ran 10 ants. The results we obtained are represented in Fig. 9.

Following the mentioned insights, PF-BTS is expected to outperform BS in terms of schedule generation/confirmation time. A note to be mentioned, ACO algorithm consumes much time to iterate when its input tasks counts in thousands, hence a suggestion of the PF-BTS algorithm is to group each thousand tasks into one block and give the block one ID, as if it was one task that has a computational length of the summation of all tasks lengths.

5. Conclusion and future work

In this paper, we proposed a Privacy-aware Fog-enhanced Blockchain-assisted Task Scheduling protocol, called PF-BTS. Our proposed protocol deploys the ACO algorithm to find the best assignment schedule of tasks into the cloud's virtual resources in a secure manner. We discussed, how PF-BTS preserves the privacy of end users by keeping their identities, location, and tasks anonymous for BC miners. The cloud utilizing PF-BTS receives the best assignment of tasks into its VRs, in terms of total execution time of the virtual resources. We have shown how our proposed protocol outperforms several previously proposed approaches for solving similar problems, in terms of privacy, efficient scheduling of the virtual resources, and minimal exchanged messages. Moreover, PF-BTS allows the fog node, as an extension of the cloud, to perform the computational tasks at the edge of the network. Hence, we expect that PF-BTS will compensate any increased network load by decreasing the required computational cycles performed by the VRs.

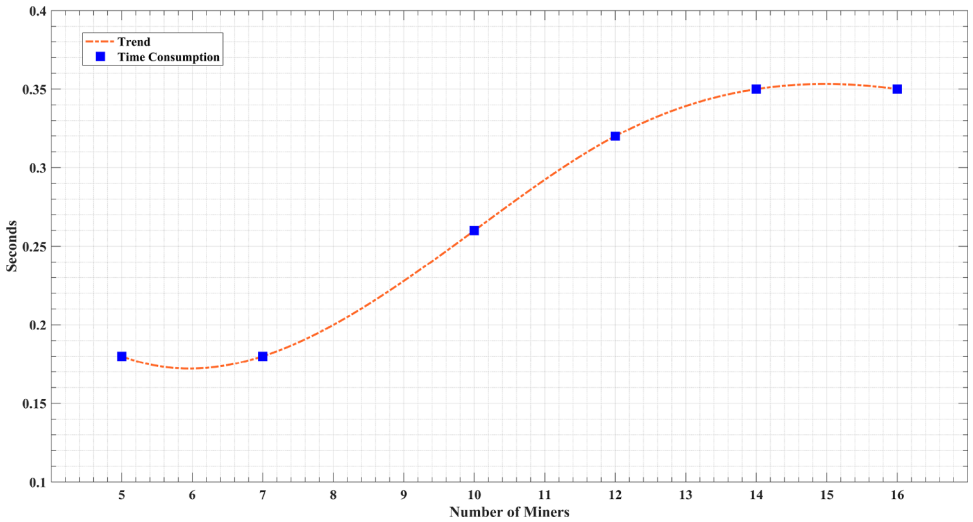


Fig. 9. Time consumption of PF-BTS relatively to the number of miner nodes.

Our future work will be directed towards experimenting other optimization approaches that are competitive with ACO, such as the ACTS-LB Li and Wu (2019), MO-ACO Guo (2017), or the Black Hole Hatamlou (2013) algorithms. We will also investigate the output of BC nodes in terms of assignment solutions quality and time consumption, by enforcing multiprocessing and multi-threading techniques in each miner node. Further, we will investigate the deployment possibilities of SC results into side-chains, and then accumulating all the blocks into one block to be saved on the root chain (similarly to Plasma BCs Ziegler, Gromann, & Krieger (2019)). Such approach may enhance the scalability measures of PF-BTS, and facilitate easy analysis of on-chain data.

Acknowlgedgments

This research was supported by the Hungarian Scientific Research Fund under the grant number OTKA FK 131793, and by the Hungarian Government and the European Regional Development Fund under the grant number GINOP-2.3.2-15-2016-00037 (Internet of Living Things), by the Hungarian Government under the grant number EFOP-3.6.1-16-2016-00008, and by the University of Szeged Open Access Fund under the grant number 4844.

References

Ak, E., & Canberk, B. (2019). Bcdn: A proof of concept model for blockchain-aided cdn orchestration and routing. *Computer Networks, 161*, 162–171.
Ala'a Al-Shaikh, H. K., Sharieh, A., & Sleit, A. (2016). Resource utilization in cloud computing as an optimization problem. *Resource, 7*(6).
Bacsó, G., Kis, T., Visegrádi, Á., Kertész, A., & Németh, Z. (2016). A set of successive job allocation models in distributed computing infrastructures. *Journal of Grid Computing, 14*(2), 347–358.
Baniata, H., Almobaideen, W., & Kertesz, A. (2020). *A privacy preserving model for fog-enabled mcc systems using 5g connection. The fifth international conference on fog and mobile edge computing (fmec-2020)*. IEEE.
Baniata, H., & Kertész, A. (2020). *Pf-bvm: A privacy-aware fog-enhanced blockchain validation mechanism. 10th international conference on cloud computing and services science (closer-2020)*430–439.
Baniata, H., & Kertesz, A. (2020). A survey on blockchain-fog integration approaches. *IEEE Access, 8*, 102657–102668.
Becker, J., Breuker, D., Heide, T., Holler, J., Rauer, H. P., & Böhme, R. (2013). *Can we afford integrity by proof-of-work? scenarios inspired by the bitcoin currency. The economics of information security and privacy*. Springer135–156.
Bellavista, P., Giannelli, C., Montenero, D. D. P., Poltronieri, F., Stefanelli, C., & Tortonesi, M. (2020). Holistic processing and networking (hornet): An integrated solution for iot-based fog computing services. *IEEE Access, 8*, 66707–66721.
Betelgeusian (2018). Mining hardware comparison. [Online; accessed 6-April-2020] https://en.bitcoin.it/wiki/Mining_hardware_comparison.
Bokhari, S. H. (2012). *Assignment problems in parallel and distributed computing. 32*. Springer Science & Business Media.
Butt, T. A., Iqbal, R., Salah, K., Aloqaily, M., & Jararweh, Y. (2019). Privacy management in social internet of vehicles: review, challenges and blockchain based solutions. *IEEE Access, 7*, 79694–79713.
Chen, H., & Cheng, A. M. (2005). Applying ant colony optimization to the partitioned scheduling problem for heterogeneous multiprocessors. *ACM SIGBED Review, 2*(2), 11–14.
Chen, L., Takabi, H., & Le-Khac, N.-A. (2019). *Security, privacy, and digital forensics in the cloud.* John Wiley & Sons.
Colorni, A., Dorigo, M., Maniezzo, V., et al. (1992). *An investigation of some properties of an" ant algorithm". Ppsn92*.
Crosby, M., Pattanayak, P., Verma, S., Kalyanaraman, V., et al. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation, 2*(6–10), 71.
Dam, S., Mandal, G., Dasgupta, K., & Dutta, P. (2014). *An ant colony based load balancing strategy in cloud computing. Advanced computing, networking and informatics-volume 2*. Springer403–413.
Debe, M., Salah, K., Rehman, M. H. U., & Svetinovic, D. (2019). Iot public fog nodes reputation system: A decentralized solution using ethereum blockchain. *IEEE Access, 7*, 178082–178093.
Dong, M., Ota, K., & Liu, A. (2015). *Preserving source-location privacy through redundant fog loop for wireless sensor networks. 2015 ieee international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing.* IEEE1835–1842.
Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine, 1*(4), 28–39.
Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical computer science, 344*(2–3), 243–278.

Ethereum, (2020). Ethereum average block time chart, [Online; accessed 25-June-2020] https://etherscan.io/chart/blocktime.
Gai, K., Guo, J., Zhu, L., & Yu, S. (2020). *Blockchain meets cloud computing: A survey*.
Gai, K., Qiu, M., & Zhao, H. (2018). Energy-aware task assignment for mobile cyber-enabled applications in heterogeneous cloud computing. *Journal of parallel and distributed computing, 111*, 126–135.
Garay, J., Kiayias, A., & Leonardos, N. (2017). *The bitcoin backbone protocol with chains of variable difficulty. Annual international cryptology conference*. Springer291–323.
Garcia-Font, V. (2020). Socialblock: An architecture for decentralized user-centric data management applications for communications in smart cities. *Journal of parallel and distributed computing*.
Guo, Q. (2017). *Task scheduling based on ant colony optimization in cloud environment. Aip conference proceedings1834. Aip conference proceedings* AIP Publishing LLC040039.
Gupta, D., Kumar, G. S. P., & Sidhu, H. J. S. (2017). *Comparative analysis of task scheduling algorithms in cloud environment*.
Guttman, B., & Roback, E. A. (1995). *An introduction to computer security: The NIST handbook*. Diane Publishing.
Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information sciences, 222*, 175–184.
Hecker, F. T., Stanke, M., Becker, T., & Hitzmann, B. (2014). Application of a modified ga, aco and a random search procedure to solve the production scheduling of a case study bakery. *Expert systems with applications, 41*(13), 5882–5891.
Hussein, M. K., & Mousa, M. H. (2020). Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access, 8*, 37191–37201.
Jiao, Y., Wang, P., Niyato, D., & Suankaewmanee, K. (2019). Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks. *IEEE Transactions on Parallel and Distributed Systems*.
Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing. *Egyptian informatics journal, 16*(3), 275–295.
Karthikeyan, P., & Thangavel, M. (2018). *Applications of security, mobile, analytic, and cloud (SMAC) technologies for effective information processing and management*. Engineering Science Reference.
Kavitha, K., & Sharma, S. (2019). Performance analysis of aco-based improved virtual machine allocation in cloud for iot-enabled healthcare. *Concurrency and Computation: Practice and Experience,* e5613.
Khalid, T., Abbasi, M. A. K., Zuraiz, M., Khan, A. N., Ali, M., Ahmad, R. W., ... Aslam, M. (2019). A survey on privacy and access control schemes in fog computing. *International Journal of Communication Systems,* e4181.
King, S., & Nadal, S. (2014). Ppcoin: Peer-to-peer crypto-currency with proof-of-stack, august 19, 2012.
Kraft, D. (2016). Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Networking and Applications, 9*(2), 397–413.
Kshetri, N., & Voas, J. (2018). Blockchain-enabled e-voting. *IEEE Software, 35*(4), 95–99.
Li, G., & Wu, Z. (2019). Ant colony optimization task scheduling algorithm for swim based on load balancing. *Future Internet, 11*(4), 90.
Li, M., Zhu, L., & Lin, X. (2018). Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet of Things Journal, 6*(3), 4573–4584.
Mauw, S., & Oostdijk, M. (2005). *Foundations of attack trees. International conference on information security and cryptology*. Springer186–198.
Merkle, D., & Middendorf, M. (2003). Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence, 18*(1), 105–111.
Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation, 6*(4), 333–346.
Milutinovic, M., He, W., Wu, H., & Kanwal, M. (2016). *Proof of luck: An efficient blockchain consensus protocol. proceedings of the 1st workshop on system software for trusted execution*1–6.
Mirtaheri, S. L., & Shirzad, H. R. (2019). Optimized distributed resource management in fog computing by using ant-olony optimization c. *Future Trends of HPC in a Disruptive Scenario, 34*, 206.
Nakamoto, S. (2019). *Bitcoin: A peer-to-peer electronic cash systemTechnical Report*. Manubot.
Nguyen, G.-T., & Kim, K. (2018). A survey about consensus algorithms used in blockchain. *Journal of Information Processing Systems, 14*(1).
Ni, J., Zhang, A., Lin, X., & Shen, X. S. (2017). Security, privacy, and fairness in fog-based vehicular crowdsensing. *IEEE Communications Magazine, 55*(6), 146–152.
Nosheen, F., Bibi, S., & Khan, S. (2013). *Ant colony optimization based scheduling algorithm. 2013 international conference on open source systems and technologies*. IEEE18–22.
Panda, S. K., Gupta, I., & Jana, P. K. (2019). Task scheduling algorithms for multi-cloud systems: allocation-aware approach. *Information Systems Frontiers, 21*(2), 241–259.
Pedemonte, M., Nesmachnow, S., & Cancela, H. (2011). A survey on parallel ant colony optimization. *Applied soft computing, 11*(8), 5181–5197.
Pentico, D. W. (2007). Assignment problems: A golden anniversary survey. *European journal of operational research, 176*(2), 774–793.
Reimann, M., & Leal, J. E. (2013). *Single line train scheduling with aco. European conference on evolutionary computation in combinatorial optimization*. Springer226–237.
Rugwiro, U., Gu, C., & Ding, W. (2019). Task scheduling and resource allocation based on ant-colony optimization and deep reinforcement learning. *Journal of Internet Technology, 20*(5), 1463–1475.
Saka, M., & Dogan, E. (2012). Recent developments in metaheuristic algorithms: Areview. *Comput Technol Rev, 5*(4), 31–78.
Samaniego, M., Jamsrandorj, U., & Deters, R. (2016). *Blockchain as a service for iot. 2016 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)*. IEEE433–436.
Schneier, B. (2015). Attack trees. *Secrets and Lies: Digital Security in a Networked World,* 318–333.
Singh, S., & Singh, N. (2016). *Blockchain: Future of financial and cyber security. 2016 2nd international conference on contemporary computing and informatics (ic3i)*. IEEE463–467.
Srikanth, U., Maheswari, U., Palaniswami, S., & Siromoney, A. (2016). Task scheduling using probabilistic ant colony heuristics. *International Arab Journal of Information Technology (IAJIT), 13*(4).
Stauffer, D., Hehl, F. W., Ito, N., Winkelmann, V., & Zabolitzky, J. G. (2012). *Computer simulation and computer algebra: Lectures for beginners.* Springer Science & Business Media.
Swan, M. (2015). *Blockchain: Blueprint for a new economy. "* O'Reilly Media, Inc.".
Tang, W., Zhao, X., Rafique, W., & Dou, W. (2018). *A blockchain-based offloading approach in fog computing environment. 2018 ieee intl conf on parallel & distributed processing with applications, ubiquitous computing & communications, big data & cloud computing, social computing & networking, sustainable computing & communications (ispa/iucc/bdcloud/socialcom/sustaincom)*. IEEE308–315.
Tawfeek, M. A., El-Sisi, A., Keshk, A. E., & Torkey, F. A. (2013). *Cloud task scheduling based on ant colony optimization. 2013 8th international conference on computer engineering & systems (icces)*. IEEE64–69.
T'kindt, V., & Billaut, J.-C. (2006). *Multicriteria scheduling: Theory, models and algorithms.* Springer Science & Business Media.
Tortonesi, M., Wrona, K., & Suri, N. (2019). Secured distributed processing and dissemination of information in smart city environments. *IEEE Internet of Things Magazine, 2*(2), 38–43.
Tseng, L., Wong, L., Otoum, S., Aloqaily, M., & Othman, J. B. (2020). Blockchain for managing heterogeneous internet of things: A perspective architecture. *IEEE network, 34*(1), 16–23.
Tuba, M., Jovanovic, R., & SERBIA, S. (2009). An analysis of different variations of ant colony optimization to the minimum weight vertex cover problem. *WSEAS Transactions on Information Science and Applications, 6*(6), 936–945.
Ullman, J. D. (1975). Np-complete scheduling problems. *Journal of computer and system sciences, 10*(3), 384–393.
Umarani Srikanth, G., Maheswari, V. U., Shanthi, P., & Siromoney, A. (2012). *Tasks scheduling using ant colony optimization*.
Vukolić, M. (2015). *The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. International workshop on open problems in network security*. Springer112–125.
Wilczyński, A., & Kołodziej, J. (2020). Modelling and simulation of security-aware task scheduling in cloud computing based on blockchain technology. *Simulation Modelling Practice and Theory, 99*, 102038.

Xiong, Z., Feng, S., Wang, W., Niyato, D., Wang, P., & Han, Z. (2018). Cloud/fog computing resource management and pricing for blockchain networks. *IEEE Internet of Things Journal, 6*(3), 4585–4600.

Xu, J., Hao, Z., Zhang, R., & Sun, X. (2019). A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. *IEEE Access, 7*, 116218–116226.

Xue, T., Yuan, Y., Ahmed, Z., Moniz, K., Cao, G., & Wang, C. (2018). *Proof of contribution: A modification of proof of work to increase mining efficiency. 2018 ieee 42nd annual computer software and applications conference (compsac)1. 2018 ieee 42nd annual computer software and applications conference (compsac)* IEEE636–644.

Yang, X., Hou, Y., Ma, J., & He, H. (2019). Cdsp: A solution for privacy and security of multimedia information processing in industrial big data and internet of things. *Sensors, 19*(3), 556.

Yi, S., Qin, Z., & Li, Q. (2015). *Security and privacy issues of fog computing: A survey. International conference on wireless algorithms, systems, and applications.* Springer685–695.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., & Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services, 14*(4), 352–375.

Zhou, Z., Li, F., Zhu, H., Xie, H., Abawajy, J. H., & Chowdhury, M. U. (2019). An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Computing and Applications,* 1–11.

Zhu, X., & Badr, Y. (2018). *Fog computing security architecture for the internet of things using blockchain-based social networks. 2018 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata).* IEEE1361–1366.

Ziegler, M. H., Gromann, M., & Krieger, U. R. (2019). *Integration of fog computing and blockchain technology using the plasma framework. 2019 ieee international conference on blockchain and cryptocurrency (icbc).* IEEE120–123.