Operations Research and Fuzziology 运筹与模糊学, 2024, 14(2), 1-8
Published Online April 2024 in Hans. https://www.hanspub.org/journal/orf
https://doi.org/10.12677/orf.2024.142106

Hans汉斯

基于聚类算法的PBFT共识优化方案与验证

杨兴宇，周　骅

贵州大学大数据与信息工程学院，贵州　贵阳

收稿日期：2024年1月5日；录用日期：2024年1月25日；发布日期：2024年3月15日

摘　要

随着区块链技术在多个领域的广泛部署，其面临的共识机制的效率较低和计算资源的过度消耗的挑战。本研究通过对DBSCAN聚类算法进行改进，使之适应去中心化的区块链共识网络，进而对大规模的网络节点进行有效的聚类和层次化管理。通过引入了监督节点以增强共识模型的安全性。在模拟环境中进行实验，验证了通过适当的聚类可以显著优化共识过程中的耗时和通信复杂度。在拥有1,000个节点的网络环境中，相比于传统的PBFT算法，我们的方案能够将单次共识的耗时降低26.7%，并且在最佳情况下，共识通信的次数可以至少减少一个数量级。这一方案显著提高了参与共识节点的效率，使得区块链应用更加节能且高效。

关键词

聚类算法，群体共识，实用拜占庭容错，区块链

Optimization and Verification of PBFT Consensus Mechanism Using Clustering Algorithms

Xingyu Yang, Hua Zhou

College of Big Data and Information Engineering, Guizhou University, Guiyang Guizhou

Received: Jan. 5th, 2024; accepted: Jan. 25th, 2024; published: Mar. 15th, 2024

Abstract

As blockchain technology is extensively deployed across various sectors, it faces challenges associated with the inefficiency of consensus mechanisms and the excessive consumption of computational resources. This study improves the DBSCAN clustering algorithm to adapt it to the decentra-

文章引用: 杨兴宇, 周骅. 基于聚类算法的 PBFT 共识优化方案与验证[J]. 运筹与模糊学, 2024, 14(2): 1-8.
DOI: 10.12677/orf.2024.142106

lized consensus networks of blockchain, thereby enabling effective clustering and hierarchical management of large-scale network nodes. Supervisory nodes are introduced to enhance the security of the consensus model. Experiments conducted in a simulated environment have verified that appropriate clustering can significantly optimize the time consumption and communication complexity during the consensus process. In a network environment with 1,000 nodes, compared to the traditional PBFT algorithm, our scheme can reduce the time required for a single consensus by 26.7%, and in the best-case scenario, the number of consensus communications can be reduced by at least an order of magnitude. This approach significantly improves the efficiency of nodes participating in the consensus, making blockchain applications more energy-efficient and effective.

Keywords

Clustering Algorithm, Group Consensus, PBFT (Practical Byzantine Fault Tolerance), Blockchain

1. 引言

区块链自 2009 年随着比特币爆火[1]，走进人们的视野，作为基于分布式网络技术(Peer to Peer, P2P)的分布式存储系统，结合了密码学、共识算法和智能合约等，构建了一个具有去中心化、不可篡改的数据结构，因其分布式布局和不可篡改的特性，现在已经应用在很多关键领域。

在当前的区块链技术体系中，网络架构可按照参与权限的开放程度被分类为三种主要类型：公开的区块链(Public Blockchains) [2]，私有的区块链(Private Blockchains) [3]和以及由多个实体共同管理的联盟链(Consortium Blockchains) [4]。在中国“十四五”时期的国家信息化发展战略中，政策制定者强调了区块链技术应用及其产业生态的健康有序发展的必要性。该战略明确提出要加大对密码学、共识机制、智能合约等区块链核心技术的研究力度，并支持构建安全、可控、可持续发展的底层技术平台及其开源社区，该规划还包括建立完整的区块链标准规范体系，强化技术测试与评估，以及制订关键领域的行业应用标准。在应用层面，规划提倡在金融科技、供应链管理、政务服务以及商业科技等关键领域开展示范性应用试点[5]。旨在通过实践探索区块链技术的最佳实施路径和应用模式，以促进技术创新与实际应用的有机结合。

当前主流的区块链共识算法各自面临特定的挑战[6]。例如，工作量证明(Proof of Work, PoW)机制，尽管在保障网络去中心化和安全性方面有着显著优势，但其对能源的高消耗和计算资源的大量浪费已成为普遍关注的问题。此外，算力集中化的趋势，特别是由于矿池的形成，增加了网络对于 51%攻击的脆弱性，从而可能威胁到系统的安全性。权益证明(Proof of Stake, PoS)机制，虽然在能源消耗上相对较低，但它容易引起链的分叉，这可能会破坏区块链的一致性和稳定性。此外，PoS 机制在某些情况下可能会导致“富者更富”的问题，从而影响网络的分散性。对于经典的拜占庭容错(Practical Byzantine Fault Tolerance, PBFT)算法，它在处理少量高信任节点的环境中表现良好，但是随着参与共识的节点数量增加，其网络通信复杂度呈指数级增长，从而导致通信开销大幅上升并降低共识效率。

为了解决 PBFT 共识算法在可扩展性方面的局限性，黄冬艳等研究人员[7]借鉴了网络分片技术的理念，提出了一种适用于联盟链的分层共识机制，其中引入监督节点以增强共识过程的可监管性。这种两级共识结构显著提高了共识的效率，并增强了整体网络的可扩展性。李强等[8] [9]提出了结合 K-medoids 聚类算法和经监督节点优化的 Raft 算法的共识策略。此方法将网络节点有效地划分为若干簇，每个簇形

成一个分片，从而实现了分层次的多中心共识机制。这种创新的共识框架大幅减少了共识过程中的通信负担和时延，显著提高了共识效率和网络吞吐量。同时，它也展现了出色的可扩展性和对网络动态变化的适应能力。

本研究通过采用改进的聚类算法对 PBFT 节点进行有效分组，并在每个分组间引入 Raft 算法以建立组间的监督机制，构建了一个多中心的分层区块链共识机制模型。该模型通过分组策略显著降低了共识过程中的通信频次，并通过设立监督节点来加强共识机制的安全保障。使得共识模型极大地提高了区块链技术在多样化和广泛的应用场景中的适用性和可行性。

2. 准备知识

2.1. PBFT 算法

实用拜占庭容错(PBFT)算法由 Castro 和 Liskov 在 1999 年提出[10]，是一种旨在分布式计算环境中解决状态机副本一致性问题的协议。该算法能够在失效节点数量不超过集群节点总数的三分之一时，即 $f<(n-1)/3$ 的情况下，确保系统的可用性和一致性。PBFT 中的节点扮演两种角色：一是作为协调者的主节点(Leader)，另一些则作为参与者的从节点(Follower)。PBFT 算法的共识过程主要分为三个阶段，以确保全网的一致性和容错性，其输出传输流程如图 1 所示。

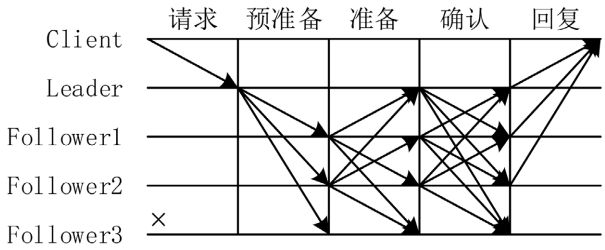

Figure 1. Consensus flow of PBFT algorithm
图 1. PBFT 算法的共识流程

预准备阶段(Pre-Prepare)：主节点在接收并验证客户端请求的合法性后，会创建一个预准备消息，并将该消息广播至所有从节点。此消息包含请求的详细信息和一个唯一的序列号，用于同步系统状态和维持请求顺序。

准备阶段(Prepare)：从节点在收到主节点发出的预准备消息后，会进行校验，以确保该消息在传输过程中未被篡改。验证无误后，从节点则生成一个准备消息，并将此消息广播至所有节点，包括主节点和其他从节点。这一步骤旨在确保网络中的所有节点对该请求的处理达成一致性协议。

确认阶段(Commit)：一个节点在收到至少 $2f+1$ (f表示可容忍的最大失效节点数)，如果这些消息都被验证为真实有效，该节点则进入确认阶段。随后，节点会生成一个确认消息并将其广播给所有节点。该节点同时也会继续接收其他节点的确认消息，并在收到至少 $2f+1$ 个有效的确认消息后，达到提交状态(Committed)。此时，即使只有一个节点达到了提交状态，也足以证明该请求已经得到了大多数节点的验证和批准。

通过这三个阶段的严格协调，PBFT 能够在存在拜占庭错误的不完全可靠网络中达成一致性，从而使得系统在面对各种失效情况时仍能保持正常运作。

2.2. 聚类算法

常见的聚类算法有 K-means [11]、K-medoids [12]、基于密度的空间聚类算法[13] (Density-Based Spatial

Clustering of Applications with Noise, DBSCAN)等。其中 K-means 和 K-medoids 算法类似，都是以待聚类集合中点在某些特征空间中的距离作为评判相似度的标准，来达到聚类的目的。K-means 算法属于均值聚类算法，其在聚类过程中选择的聚类中心是同一簇内一群真实节点的特征均值。K-medoids 算法放弃了 K-means 中的均值策略，而是采用实际的节点作为聚类中心，从而减小了对孤立节点的敏感性，提高了聚类的准确性。DBSCAN 算法基于密度的聚类算法，能够找出被低密度区域分隔的高密度区域，能够处理不同形状的集群，并识别噪声点，但是无法进行预先指定集群数量。

3. 基于 DBSCAN 算法的 PBFT 共识机制

3.1. 改进的 DBSCAN 算法

为了使 DBSCAN 算法更适应区块链的应用场景，本研究提出了一种改进方案。传统的 DBSCAN 算法不允许直接指定聚类的数量，这限制了其在组织区块链主节点共识网络中的应用。为了解决这个问题，采用了一种结合 K-means 聚类算法和人工指定策略的方法进行初始分组。首先，利用 K-means 算法进行粗略的聚类，以确定大致的簇心；然后，依据节点的职能和热力密度，由上层管理者选出一批中心节点。这样的预分组不仅显著减少了 DBSCAN 算法的计算量，而且提高了聚类执行的效率。在此基础上，DBSCAN 算法被用来对预分组的簇进行细化处理，以提升聚类的准确度。当遇到边界簇时，通过合并临近簇以实现指定簇数量的聚类。由于算法考虑了节点的热力密度，聚类中心节点会根据这一参数动态调整，这意味着中心节点的变化将局限在较小的范围内，而不是每次都进行随机选择，从而增强了网络的稳定性和可靠性。这种改进的 DBSCAN 算法能够提供一种更为稳定和高效的节点组织方式，尤其是在需要快速响应和高度协作的共识网络中。它不仅降低了计算成本，还提高了网络的适应性和鲁棒性。

该方法大致分为三个步骤，使用 K-means 聚类算法对全体节点 V 进行分组，对每一组中的数据执行 DBSCAN 算法产生最终分组簇，其详细流程如下：

步骤 1：组织中心节点集群。对全体节点 V 应用 K-means 聚类算法进行初步分组。在此过程中，结合节点的职能分布，精心选择 K 个初始簇中心，从而形成主节点集群。

步骤 2：在各组中使用 DBSCAN 算法。在 K-means 算法划分出的每个分组中独立运行 DBSCAN 算法。这一步骤进一步细化了数据分组，确保了聚类结果的精确性。

步骤 3：合并边界簇。对 DBSCAN 算法处理后的分组进行边界检查和合并。通过分析 K-means 算法中确定的节点间距离，对距离过长(超过 ε)的簇进行修剪和合并。同时，检查各簇之间是否存在潜在的内部聚类现象，并进行相应合并，同时排除那些不适合合并的簇。

3.2. 改进的 PBFT 共识机制

通过使用聚类算法，将原本无差别的全体共识节点进行聚类和分层，将聚类后的簇中心节点作为该簇的主节点，簇中其他节点作为从节点。每个簇中全部节点构成一个子共识集群，选取的 K 个初始簇中心，作为主节点组成主共识集群。改进后的共识机制模型如图 2 所示。为了防止子共识集群中，领导者作恶，破坏共识的一致性，根据分布不同，在每组中设立一个或多个匿名监督节点，监督节点负责对集群主节点进行监督，监督节点不定期比对主共识网络签名与日志信息，判断主节点是否为恶意节点。

主要流程阶段如图 3 所示。

步骤 1：初始化。依据聚类算法和职能分析对所有节点进行分类。服务运营商在 K 个初始簇中指定一个中心节点作为主节点，并在子共识网络中随机选定监督节点。

步骤 2：子共识网络共识。子共识网络中的节点会将请求发送至其对应的主节点。主节点负责将这些请求打包成区块，并引导子节点一起执行一轮 PBFT 共识算法。

步骤 3：骨干共识网络共识。子共识网络达成共识后，主节点在主共识网络中发起第二轮 PBFT 共识。骨干共识集群中的节点通过轮询方式逐一广播已经通过子共识验证的区块，以进行共识。当轮询到的节点会将子共识的结果打包成一个新区块，并提交至骨干共识网络中进行 PBFT 共识。如果子共识网络中没有新的请求，该节点则打包一个空区块提交以维持链的连续性。

步骤 4：提交阶段。共识达成后，骨干共识节点对区块进行数字签名，并收集其他主节点的签名，共同确认区块的真实性和有效性。将这些签名和区块本身打包成提交消息，并广播至所属的子共识集群中的所有从节点，表明该区块已获准加入区块链。

步骤 5：执行阶段。监督节点根据收到的主节点信息，对区块的签名和内容进行核验。如果核验失败，表明主节点可能存在恶意行为，监督节点将向运营商举报，以实现对节点的监管。若核验成功，则按照区块内容，更新区块链记录并执行上链操作。

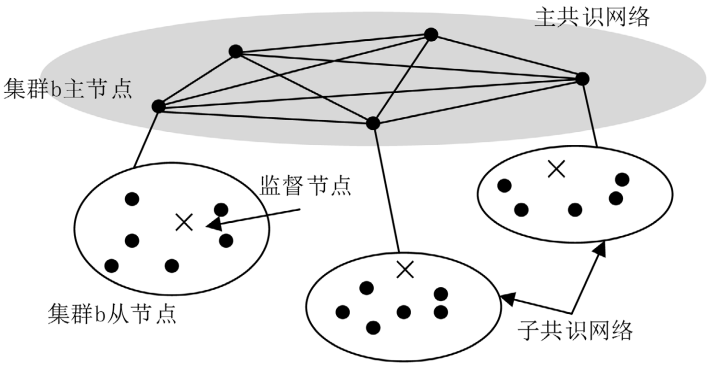

Figure 2. Improved PBFT consensus mechanism model
图 2. 改进后的 PBFT 共识机制模型

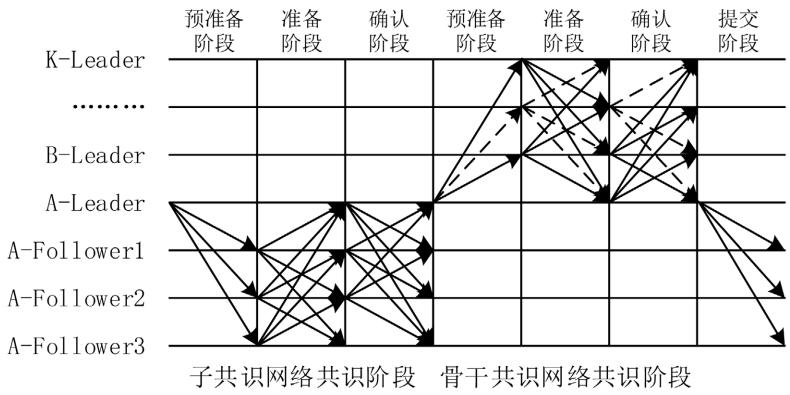

Figure 3. Improved PBFT consensus process
图 3. 改进后的 PBFT 共识过程

4. 仿真实验

在对比改进后的聚类 PBFT 共识算法与传统 PBFT 共识算法，可以从两个关键的性能指标进行分析：单次共识耗时与单次共识过程中的通信次数。下面是对这两个方面的概要分析。

4.1. 共识耗时

共识耗时实验中，通过产生 1000 个节点，产生 10 个子共识网络，通过为节点赋值一个欧氏坐标，通过坐标距离对子共识集群进行聚类，产生具体的共识网络。在建模过程中，为了能够较好地比较对节

点间网络延迟进行如下设定。

1) 同一子共识网络中节点延迟 $delay\left(v_{ia},v_{ib}\right)$，其中 i 表示子共识网络，$a,b,\cdots,n$ 表示子共识网络中的具体节点，设定为 10~30 ms。

2) 骨干共识网络节点间延迟 $delay\left(v_{im},v_{jm}\right)$，其中 i 表示子共识网络，m 表示主节点，根据阵营特性，设定为 100~150 ms。

3) 骨干共识网络主节点与子共识网络中节点间延迟 $delay\left(v_{mm},v_{ia}\right)$，其中 m 表示骨干共识网络，m 表示主节点，设定为 110~180 ms。

按照设定条件，生成的节点模拟产生节点之间的延迟，对共识模型的共识耗时进行仿真，结果如图 4 所示。

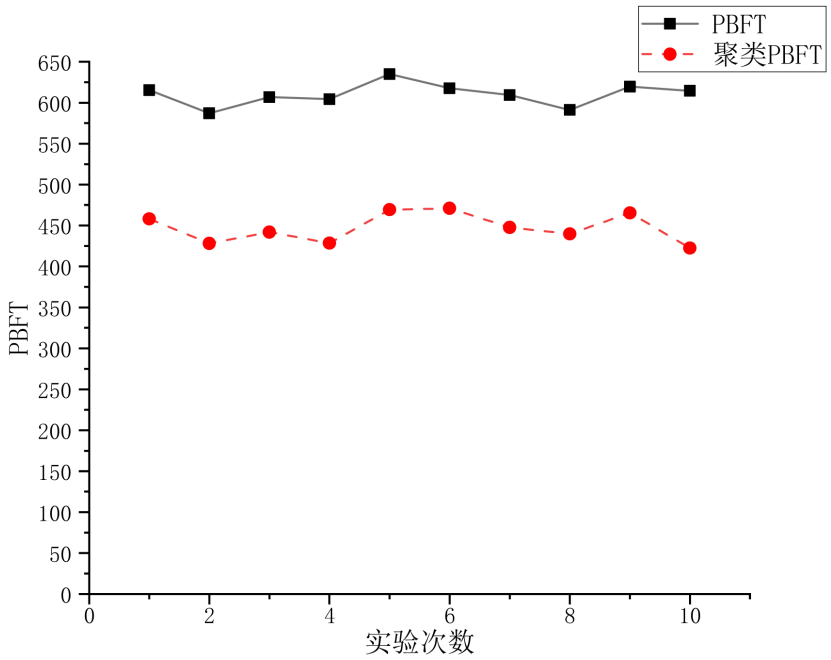

Figure 4. Improved PBFT consensus time consumption comparison
图 4. 改进后的 PBFT 共识耗时对比

通过 10 次实验对共识耗时进行对比，传统 PBFT 单次共识的平均时间为 610 ms，聚类 PBFT 单次共识的平均时间为 447 ms 相较与传统 PBFT 算法，单次共识耗时缩短了 26.7%。相对传统 PBFT 算法每次都需要进行全体节点参加共识过程，大量节点频繁广播共识消息造成网络堵塞，通过使用聚类算法，将节点进行聚类，对共识节点进行分组分层，大幅减少了参与共识的节点数量，从而在节点众多的网络环境中有效缩短了共识耗时，并提升了共识效率。同时由于分层共识网络，子共识网络和骨干共识网络可以并行的执行共识过程，进一步提高共识效率。

4.2. 共识通信消耗

本文对两种共识机制的通信复杂度进行对比。

1) 计算 PBFT 共识机制的通信次数

如表 1 所示，假设平台中共有 $n\left(n\geq 3\right)$ 个节点参与 PBFT 共识，完成一次 PBFT 共识过程所需的总通信次数为 $2n\left(n-1\right)$。

Table 1. PBFT communication complexity
表 1. PBFT 通信复杂度

共识阶段	预准备	准备	确认
通信次数	$(n-1)$	$(n-1)(n-1)$	$n(n-1)$

2) 计算聚类 PBFT 共识机制的通信次数

假设平台中共有 $n(n\geq 3)$ 个节点参与 PBFT 共识，设置聚类簇数为 K，为简化计算，假设聚类为平均分配，即每个子共识网络中节点个数为 n/K，骨干共识网络节点个数为 K。首先子共识节点分别进行分组共识，即通信次数为 $2n/K\cdot(n/K-1)$，有 K 个子共识网络即全体子共识网络中通信次数为 $2n\cdot(n/K-1)$。骨干共识节点进行一次共识的通信次数为 $2K(K-1)$，最终骨干共识居群中的节点将共识区块广播到所属共识网络中，完成本次共识，此环节的通信次数为 $n-K$。计算可得。

Table 2. Clustering PBFT communication complexity
表 2. 聚类 PBFT 通信复杂度

共识阶段	子共识	骨干共识	确认
通信次数	$2n\cdot(n/K-1)$	$2K(K-1)$	$n-K$

由表 2 可得，完成一次聚类 PBFT 共识过程的总通信次数为 $2n\cdot(n/K-1)+2K(K-1)+n-K$。两种算法的通信次数比如式所示

$$Z=\frac{2n(n-1)}{2n\cdot(n/K-1)+2K(K-1)+n-K} \tag{1}$$

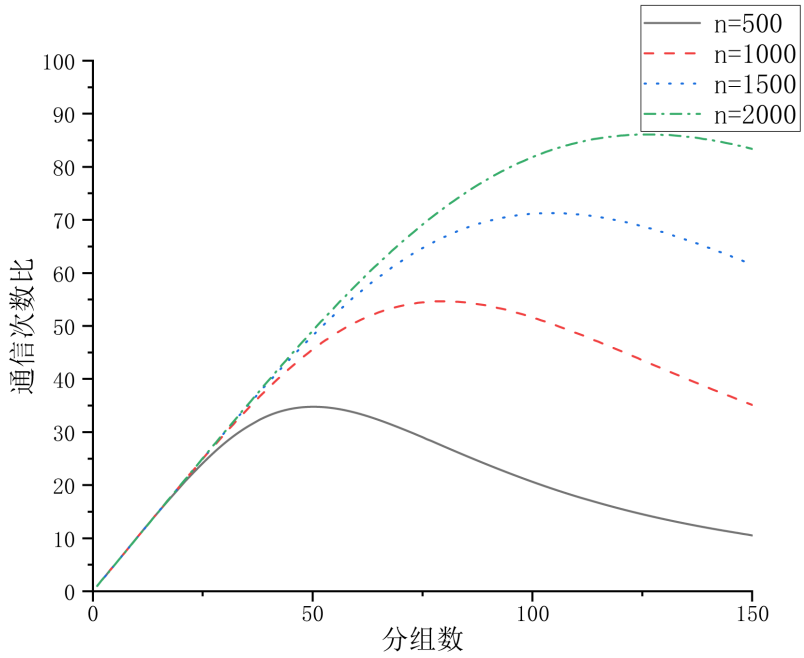

Figure 5. Algorithm communication count ratio
图 5. 算法通信次数比

分别绘制节点数量 $n=500,1000,1500,2000$ 时，Z 的曲线，如图 5 所示。可见，不同数量节点情况下，随着分组的增加，通信次数比大幅度提升，但随着分组数量的增加，过多的主共识节点会导致骨干共识网络过于臃肿，导致通信效率下降，由此根据通信节点数量选定分组数才能获取最好的优化效果。

5. 结语

本文提出了一种改进的 DBSCAN 算法，并将其应用于优化 PBFT 共识网络性能中。对体系结构进行了仿真描述，并对性能提升指标进行量化，通过仿真实验，验证本文提出的架构有效性。仿真结果表明，该设计在 1000 个节点分 10 组的情况下，单次共识耗时缩短了 26.7%，并对不同节点数量情况下，分析了不同分组对通信负责度的优化效果。

参考文献

[1] Nakamoto, S. and Bitcoin, A. (2008) A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

[2] Das, D. (2021) Toward Next Generation of Blockchain Using Improvized Bitcoin-NG. *IEEE Transactions on Computational Social Systems*, **8**, 512-521. https://doi.org/10.1109/TCSS.2021.3049477

[3] Jo, M., Hu, K., Yu, R., *et al*. (2020) Private Blockchain in Industrial IoT. *IEEE Network*, **34**, 76-77. https://doi.org/10.1109/MNET.2020.9199796

[4] Zhong, C., Zhang, Z., Lin, P., *et al*. (2021) Research on Photovoltaic Subsidy System Based on Alliance Chain. In: Liu, Q., Liu, X., Shen, T. and Qiu, X., eds., *Proceedings of the* 10*th International Conference on Computer Engineering and Networks*, Springer, Singapore, 1048-1055. https://doi.org/10.1007/978-981-15-8462-6_121

[5] 中央网络安全和信息化委员会印发《“十四五”国家信息化规划》[EB/OL]. http://www.cac.gov.cn/2021-12/27/c_1642205312337636.htm, 2022-11-28.

[6] 林知微, 张嵩川, 王成吉, 周亦炜. 区块链技术综述: 在下一代智能制造中的应用[J]. 智能科学与技术学报, 2023, 5(2): 200-211.

[7] 黄冬艳, 李浪, 陈斌, 王波. RBFT: 基于 Raft 集群的拜占庭容错共识机制[J]. 通信学报, 2021, 42(3): 209-219.

[8] 陈子豪, 李强. 基于 K-medoids 的改进 PBFT 共识机制[J]. 计算机科学, 2019, 46(12): 101-107.

[9] 王谨东, 李强. 基于 Raft 算法改进的实用拜占庭容错共识算法[J]. 计算机应用, 2023, 43(1): 122-129.

[10] Castro, M. and Liskov, B. (2002) Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, **20**, 398-461. https://doi.org/10.1145/571637.571640

[11] Ahmed, M., Seraj, R. and Islam, S.M.S. (2020) The k-Means Algorithm: A Comprehensive Survey and Performance Evaluation. Electronics, **9**, Article No. 1295. https://doi.org/10.3390/electronics9081295

[12] Park, H.S. and Jun, C.H. (2009) A Simple and Fast Algorithm for K-Medoids Clustering. *Expert Systems with Applications*, **36**, 3336-3341. https://doi.org/10.1016/j.eswa.2008.01.039

[13] Gunawan, A. (2013) A Faster Algorithm for DBSCAN. Master’s Thesis, Eindhoven University of Technology, Eindhoven.