Theme Article:
Innovations Using Blockchain—Part 2

Research on Smart Contract Optimization Method on Blockchain

Wen Hu, Zhipeng Fan, and Ye Gao
Harbin University of Commerce

***Abstract*—The smart contract on the blockchain allows credible transactions without a third party. These transactions are traceable and irreversible. The deployment and implementation of smart contracts in Ethernet will consume some gas, which will directly affect the cost of smart contracts. In order to reduce the consumption of gas during the execution of smart contracts, this article proposes an optimization algorithm for generating business process smart contracts. First, business process modeling notation (BPMN) models are extended to Petri nets. Second, Petri nets are simplified to find nodes in BPMN models that can be considered fusion tasks. Using new mapping rules from the BPMN model to solidity language, BPMN model is generated into Ethereum Smart contract model. In the BPMN models with multilayer fusion task, experimental results show that the proposed algorithm can save 15% gas on average for business processes with multiple fusion tasks.**

Blockchain is an innovation that involves technologies such as peer-to-peer (p2p), Byzantine fault tolerance, smart contracts, and distributed consensus algorithms. Based on this integrated technology stack, blockchain implements a decentralized system that can achieve consensus and trust in digital space. The core idea is that under the decentralization characteristics of the p2p network, a consistent consensus is reached among nodes, and a data chain that links data blocks end to end in a chronological order is maintained. Each block on the data chain prevents data tampering and forgery by means of data verification such as the digital signature. As a result, blockchain implements a decentralized shared ledger. This new thinking can solve the problems of high cost, poor reliability, and low security that traditional centralized system generally relies on central authority, trust, and consensus.[1] It is a subversive change and challenge to traditional social organization and operation mode. The emergence of blockchain technology provides strong support for the redesign of collaborative business processes (such as supply chain and logistics processes).[2] With this

Digital Object Identifier 10.1109/MITP.2019.2923604
Date of current version 11 September 2019.

idea, Weber[3] proposed a solution to execute business processes on blockchain in the form of Smart contracts and verified the feasibility of this method through actual deployment tests. However, the gas cost was not considered in this method.

Ethereum is one of the most popular blockchain applications at present, which is an open source digital currency and blockchain application platform for developers to build and publish smart contracts.[4] The "blockchain" and "smart contract" referred to in this paper, respectively, refer to Ethereum blockchain and smart contract. Smart contracts can be developed using Solidity[5] programming language, and the Smart contracts written will eventually be compiled into bytecode and deployed to the Ethernet lane virtual machine for execution. Gas is a consumption unit of smart contract deployment and execution in Ethereum, which can be regarded as a currency of equal value, and its quantity is related to the bytecode size of smart contract.[6] Therefore, gas consumption needs to be considered when designing relevant business process contracts. The current research mainly considers reducing gas consumption from the aspects of contract deployment and invocation. In this article, an optimized business process smart contract template generation method is designed to solve the problem of gas consumption during contract deployment. This approach extends the business process business process modeling notation (BPMN) model to Petri nets. Then, find out which combinations of nodes in the BPMN model can be considered to be a fusion task. Finally, a mapping rule from the BPMN model to solidity code is proposed to translate the BPMN model into an optimized smart contract template. Experimental results show that the optimized smart contract template reduces gas consumption when deployed to Ethernet blockchain.

BPMN MODEL EXTENSION AND SIMPLIFICATION BASED ON PETRI NET

BPMN Model Extended to Petri Nets

BPMN[7] is the most popular visual business process modeling language. This approach enables business processes to be described graphically for business managers and software developers to understand. In general, the program code written is executed sequentially, including the correct execution of many logical branch structure auxiliary programs. According to business process modeling methods, we can set up two classes of BPMN business process models: 1) the order of business process BPMN model; 2) the choice to perform a BPMN model of the business process. No matter which kind of BPMN model is used for mapping smart contract code, all we need to ensure is the integrity of the business process. The model cannot simplify the step of the business process. So that we can avoid the results of the business process, smart contract cannot reach the desired effect. Sequential smart contracts are in the business process order execution, and the code content is fixed. So, we cannot change sequential smart contracts. Sequential smart contracts do not meet the optimized conditions. In this article, we do not discuss sequentially smart contracts. This article only includes selection branch of the structure of the smart contract business process optimization. Therefore, this article uses the following node types to model business processes BPMN: tasks, start and end events, XOR decision gateways, and XOR merge gateways. As shown in Figure 1(a), a basic BPMN model is shown, with each node annotated with a label for ease of reference.

BPMN models are extended to Petri nets according to mapping rules.[8] For example, a task or intermediate event is mapped to a transition with an input library and an output library, and the transition is marked with the name of the task or event to simulate the execution of the task or event. The start event is mapped to a static transition with an input library and an output library with initial internal conditions, and the input library has no input arc, which is only used to represent the beginning of Petri net. The end event is mapped to a static transition with an input library and an output library, and there is no output arc in the output library; it is only used to represent the end of the Petri net, as shown in Figure 1(b). Tasks and events in BPMN models are extended to mark-up transitions. The extension also introduces unmarked additional transitions. These transitions are static events that only show the start and end of Petri net models and the making of decisions. These transitions do not correspond to any specific work, so they can be implicitly modeled, as shown in Figure 1(c).

Petri Net Reduction

The Petri net, as a formal model to describe system behavior, is especially suitable for simulating system behavior according to control flow, object flow, or information flow. The Petri net has been widely used in the current research field. The Petri net modeling tool supports the automatic analysis and design of models and has the characteristics of accessibility, deadlock detection, and boundary analysis. Therefore, using Petri nets to define the behavior of business flows is a good new choice.

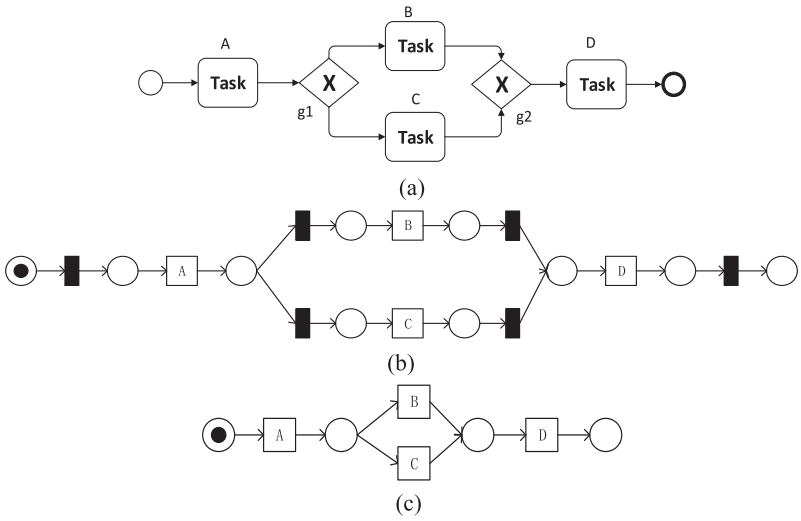

Figure 1. BPMN example model and Petri nets for BPMN model.

The Petri net expanded from the previous step is actually a working flow net. The workflow process defined by the Petri net has only one input library (start) and one output library (end), and each transition (task or event) or library (condition) must be between the input library and the output library. However, Petri nets conforming to this definition may still have some abnormal conditions, such as potential deadlock and failure to end. Therefore, a reasonable working flow network should meet the following three requirements:[9]

1) Tags in the input library will eventually appear in the output library.
2) When tags appear in the output library, there are no tags in other libraries.
3) For each transition (task or event), the ready state can be reached from the initial state.

Assume that Petri nets extended according to BPMN instance models satisfy the above properties. As we all know, such Petri nets can be simplified. The six simplification rules are proposed and verified by Murata.[10]

By using simplification rules, parallel transitions in extended Petri nets are fused to obtain Petri nets as shown in Figure 2. It can be seen that parallel transition B and C in the original Petri net are replaced by a new transition, which is named fusion transition in this paper.

BPMN TO INTELLIGENT SOLIDITY CONTRACT TEMPLATE

BPMN Node Combination

For simplified Petri nets, each change can be seen as one of the original BPMN model task node. Change A corresponds to the $Task_A$, change D corresponds to $Task_D$, and corresponding fusion changes this task; the task contains $Task_A$ and $Task_D$ path between nodes.

Through the above analysis, it can be found that when the BPMN model contains the following combinations of node, they can be seen as a task node:

1. The BPMN model contains XOR decision gateway and the corresponding XOR merge gateway.
2. There is only one task on the two paths between the XOR decision gateway and the XOR merge gateway.

The internal execution of the task node depends on the XOR decision gateway, which is equivalent to the judgment branch statement in the program code. Therefore, this article named this task node as the fusion task. The nodes in the example BPMN model are combined into a single fusion task. Similarly, we can deduce the node combination of two-layer and three-layer fusion task. This paper only discusses the business process BPMN model that contains a combination of these three nodes.

According to the BPMN to solidity mapping rule proposed by Weber, a function is generated for each task node.[11] If the business process BPMN

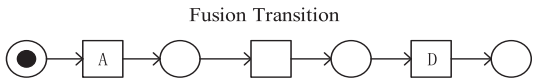

Figure 2. Simplified Petri net.

model contains the nodes that can be considered to be a fit task, it is considered to be a task node. Then, we just have to generate a function, but in order to ensure the integrity of the original business process, the original BPMN model needs to collect melting to task the gateway node containing conditions and tasks, and attach them to a function generated by the fusion task. Specific algorithm steps are as follows:

1. We go through the nodes in BPMN in order. If it is XOR decision gateway node step 2), or we continue until all the nodes have been traversed.
2. We judge whether the XOR decision gateway has an XOR merge gateway matching it. If it does, we execute step 3); otherwise, we continue to execute step 1).
3. We judge the nodes on the two paths between the XOR decision gateway and XOR merged gateway; if both are a task node, it can be considered a fusion task. We will record XOR decision conditions and the corresponding tasks; otherwise, continue to step 1).

Let us assume that the condition of the XOR decision gateway g1 in Figure 1(a) is P, and traverse the nodes in the figure in the following order: [Start, A, g1, B, g2...]. In accordance with step 1) to traverse the g1 node, we find that the g1 is XOR decision gateway; go to step 2). Step 2) looks for the XOR merged gateway that matches it, i.e., g2 in Figure 1(a), after the conditions are met; go to step 3). Step 3) will judge the node type on the two paths between g1 and g2. Because both B and C are task nodes, the judgment condition is satisfied. So, the node combination can be regarded as a fusion task. The condition P corresponding to g1 is a prerequisite for performing this task. Tasks B and C are the tasks that should be executed within the mission.

BPMN to Solidity Mapping Rules

For BPMN models, first generate a global Boolean variable StartEvent and EndEvent for the start and end events with the default set to false. The corresponding Start() and End() functions are then generated, and the StartEvent is set to true in the Start() function, which will be used to Start the entire business process. The EndEvent value is returned to the End() function. The EndEvent value will be updated after the last task execution to represent whether the entire business process has been completed.

After the normal Task node is determined, a global Boolean variable $Task_iComplete$ is defined for each normal Task node. The default value is false, which indicates whether the current Task has been completed. A function $Task_i()$ returns a Boolean value type for each normal Task node.

For a combination of nodes that can be considered a fused task, a global Boolean variable $FJTask_iComplete$ and a function $FJTask_i()$ are also generated, which act as normal task nodes. In addition, we also generate a function $XORgateway_i()$ that returns a Boolean value type for each XOR decision gateway contained in the node composition, which is used to determine the internal execution of the fusion task.

Because business processes execute sequentially, there are constraints between each task.[12] Only after a task has been fully executed can the next task be continued. Therefore, in each task function generated, the execution of its previous adjacent task node should be judged first. If the last task is completed, it can be executed; otherwise, the task cannot be executed. In addition, in order to prevent multiple invocations of the same task under one business execution, the last task needs to be deactivated after executing one task and wait for the start of the next business process. For the f fusion task, the judgment of the gateway condition should be added, because the internal execution of the fusion task function needs to be based on the value of the gateway condition.

EXPERIMENTAL RESULTS

To verify the effectiveness of the proposed approach, we modeled BPMN for existing executable business processes.[13] After the BPMN model is traversed according to the algorithm, the mapping rules proposed in this article are used to write the optimized Solidity Smart contract template. In order to check the correctness of smart contract template and facilitate the deployment test, this article uses the official development tool (Remix Solidity) of Ethereum Smart contract for deployment test.

In order to ensure that the optimized smart contract and the original smart contract operation results are consistent in the same Remix

Table 1. Gas consumption for smart contracts.

Classification	Original	Proposed	Gas saved	Rate
DecreaseApproval	279 758	248 034	31 724	11.34%
TransferFrom	359 847	358 489	11 358	3.16%
Buy	186 696	150 532	36 164	19.37%

test environment, the optimized business process smart contract method and the original method were, respectively, run and given the same test parameters. After several tests, the optimized business process smart contract has the same function as the original smart contract, and the execution result is consistent. The accuracy of the business process smart contract written according to the optimization method proposed in this article is verified.

Order processing is an indispensable business process in the process of commodity sales. It contains many logical branch structures, such as whether the goods are settled, whether the inventory is sufficient, whether the address filled in can be shipped, etc. According to the Solidity mapping rule proposed in this article, a new Solidity coding rule is defined for the node combination of the fusion task, which can write an optimized order business process Smart contract template. We tested the order business process in the simulator. According to the experiment, in the process of deployment to blockchain, compared with the use of the method of Weber, our method saved 230570 gas. This method saves gas proportion about 26.37%.

In order to further verify the effectiveness and correctness of the proposed algorithm, we selected three common types of smart contracts on Ethereum for testing. In the same test environment, we tested the gas consumption of the original program and the proposed algorithm, respectively. Furthermore, it also tests the proposed fusion task node combination separately. By establishing the basic BPMN model of the business process, three combinations of fusion task nodes are corresponding, respectively. The experimental results are shown in Table 1 and Figure 3.

Experimental results show that if the business process BPMN model contains node combinations that can be considered as a fusion task, the smart contract template written with the Solidity mapping rule proposed in this article will consume less gas when deployed to Ethereum. Due to a greater number of fusion tasks in the business processes of DecreaseApproval and Buy, the algorithm proposed in this article can save 15% gas on average, but in the business processes of TransferFrom, the proposed algorithm is not efficient due to fewer fusion tasks. In order to verify the correctness of the algorithm,

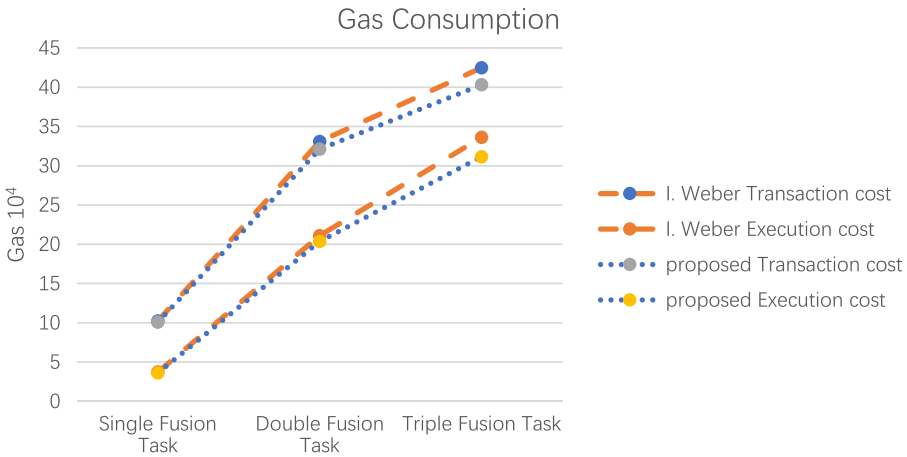

Figure 3. Comparison of gas consumption.

we use the same parameters to simulate the original business process and the optimized business process, and the results are consistent.

CONCLUSION

In order to reduce the Gas cost of business process intelligence contract deployment on the blockchain, this article proposes a mapping rule from the BPMN model to the Solidity code. First, the BPMN business process model is extended to the Petri net. It is then simplfied it according to the characteristics of Petri, the combination of nodes in the BPMN model that can be regarded as the fusion task are found, and these fusion nodes are simplified. This method can write the optimized Smart contract template while ensuring the integrity of the original business process. After the actual deployment test, the experimental results show that the proposed method can effectively reduce gas consumption of business process intelligence contracts on the blockchain.

ACKNOWLEDGMENTS

This work was supported by Team Research Project of Harbin University of Commerce (No. 2016TD001) and Research and Innovation Project of Harbin University of Commerce (No. 18XN022).

REFERENCES

1. Y. Yong and W. Fei-Yue, "Blockchain: The state of the art and future trends," *Acta Automatica Sinica*, vol. 42, no. 4, pp. 481–494, 2016.
2. L. García-Bañuelos *et al.*, "Optimized execution of business processes on blockchain," in *Proc. Bpm17 Int. Conf. Bus. Process Manage.*, 2017, pp. 130–146.
3. W. Ingo *et al.*, "Untrusted business process monitoring and execution using blockchain," in *Proc. Int. Conf. Bus. Process Manage.*, 2016, pp. 329–347.
4. A Next-Generation Smart contract and Decentralized Application Platform, White-Paper, Ethereum, Nov. 2015.
5. Solidity, 2017. [Online]. Available: http://solidity.readthedocs.io/en/v0.4.23/
6. T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reeng.*, 2017, pp. 442–446.
7. P. Bocciarelli, A. D'Ambrogio, A. Giglio, and E. Paglia, "A BPMN extension for modeling cyber-physical-production-systems in the context of Industry 4.0," in *Proc. IEEE Int. Conf. Netw.*, 2017, pp. 599–604.
8. L. Zonghua, Z. Xiaofeng, and W. Keli, "BPMN formalization based on extended Petri nets model," *Comput. Sci.*, vol. 43, no. 11, 2016, pp. 40–48.
9. Z. Han and G. B. Lee, "Reduction method for reachability analysis of Petri nets," *Tsinghua Sci. Technol.*, vol. 8, no. 2, pp. 231–235, 2012.
10. T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.
11. P. Rimba, A. B. Tran, I. Weber, M. Staples, A. Ponomarev, and X. Xu, "Comparing blockchain and cloud services for business process execution," in *Proc. IEEE Int. Conf. Softw. Archit.*, 2017, pp. 257–260
12. O. López-Pintado *et al.*, "Dynamic role binding in blockchain-based collaborative business processes," 2018. [Online]. Available: https://arxiv.org/abs/1812.02909
13. J. Mendling *et al.*, "Blockchains for business process management–Challenges and opportunities," *ACM Trans. Manage. Inf. Syst.*, vol. 9, no. 1, pp. 1–16, 2017.

Wen Hu is currently a Professor and Doctoral Supervisor with the School of Computer and Information Engineering, Harbin University of Commerce, Harbin, China. His research interests include intelligence information processing, blockchain, and electronic commerce. He received the M.S. degree from Harbin Engineering University. He is the Vice President of the Heilongjiang Computer Society. Contact him at huwen1957@126.com.

Zhipeng Fan is currently a Lecturer with the School of Computer and Information Engineering, Harbin University of Commerce, Harbin, China. His research interests include intelligence information processing, blockchain, and electronic commerce. He received the M.S. degree from the Harbin University of Commerce. He is a Member of the China Computer Federation. Contact him at fanzhipeng@hrbcu.edu.cn.

Ye Gao is currently working toward the Master's degree at the School of Computer and Information Engineering, Harbin University of Commerce, Harbin, China. His research interests include computer technology and blockchain. Contact him at 1815610194@qq.com.