Received: May 17, 2023. Revised: July 16, 2023. 

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

# A New Approach for Detecting Selfish-Mining Attacks in Blockchain Networks

**Amira Hamdi Shaaban[1*] Saleh Mesbah Elkaffa[2] Gamal Abd El-Nasser A. Said[3] Osama M Badawy[4]**

*[1]Computer Science Department, Faculty of Computing and IT, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt*
*[2]Information Systems Department, Faculty of Computing and IT, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt*
*[3]Computer, and Information Technology Department, Port Training Institute, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt*
*[4]Computing, and Information Technology Department, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt*
* Corresponding author's Emails: amira.gaber@student.aast.edu

**Abstract:** The identification of selfish mining (SM) activity is critical in blockchain mining. In exchange for successfully mining blocks, volunteers known as miners maintain the network (N/W). Assigning transactions to a block requires verifying each one, then adding the block to the blockchain and broadcasting it to the peers. This paper investigates the profitability of Bitcoin block-discarding attacks. To represent the blockchain's dynamics, a discrete event simulator is built. A simple N/W concept is employed by everyone as a miner. It is assumed that all miners are honest and that all block broadcasts are latencies, to begin with. In the case of blockchain, the latency produces splits, which are rapidly resolved. The proposed N/W model's simulation findings closely match the real-world Bitcoin N/W observations. According to the selfish-mine block discarding attack, a small group of cooperating miners can corrupt the Bitcoin system. Both immediate block transmission and latencies block are used to test this claim. This research suggested to use of metaheuristic algorithms such as ant colony optimization (ACO) and particle swarm optimization (PSO) to the mining process and applies them to previous solutions of SM detection to runtime efficiency and solution quality performance and contribute to reducing SM behavior all that by determine the optimal threshold α, optimal hash powering, and the optimal execution time to gain profits with the maximum relative revenue. Finally, our findings indicate that, when optimized, SM performs better than ordinary SM in terms of relative revenue and confirmed blocks. SM is profitable when the threshold is 0.6 and the results appear that the ACOSM optimization is better than PSOSM optimization.

**Keywords:** Blockchain, Selfish mining (SM), Bitcoin, Ant colony optimization (ACO), Particle swarm optimization (PSO), Latencies block, Runtime efficiency, Solution quality, Threshold, Hash powering, Relative revenue.

## 1. Introduction

This study assumes that the SM assault is unrealistic, as explained in [1, 2]. This paper seeks to investigate and verify these claims using a model that more closely depicts the Bitcoin N/W's operation. The primary objective of this study is to develop a system to do a solution for the honest miner through which can determine the optimal value of the threshold (α), through which the selfish can be determined all of that is built automated by using an optimization technique. The method developed by [5] was presented, and the assessment criteria, relative revenue, and mining efficiency were determined.

A simulation model with block propagation latencies was used to evaluate the outcomes of several simulation tests. The presented used the same assessment criteria to evaluate the outcomes of other tests that expressly anticipated instantaneous N/W block transmissions.

A basic model was used to test the efficacy of the Sybil attack to see if SM is lucrative. In conclusion, to show that the pool benefits from honest mining (HM), and to present the generalized SM algorithm based on the parameter vector $v = (ml; p2; p3; d2; d3; dsl; ra)$. Searched for the best possible configuration of $v$ for the pool, utilizing Selfish Mining Ant Colony Optimization (SMACO) and Selfish Mining Particle Swarm Optimization (SMPSO) [19, 20] algorithms to outperform ordinary SM. The generic SM technique performed as well as, but not better than, HM.

Blockchain is a developing database of connected documents, or blocks, that allows for rapid, direct transactions between individuals and organizations on a network. Blocks are maintained in chronological order and are updated with the most recent transactions. Blocks keep a decentralized, independent record of digital currency transactions [3]. A type of digital currency is called Bitcoin. It is a decentralized digital currency that may be sent directly between users on the peer-to-peer Bitcoin blockchain network. The foundation of the Bitcoin trust model is computing. In a process known as mining, transactions are collected into blocks that demand a significant amount of computing to verify [2].

The process of mining involves some specific nodes, referred to as miners, which include transactions in a block and produce a valid header for those transactions. Mining is responsible for updating the blockchain. To prove the validity of a new block, miners must first add unconfirmed transactions from this pool to the new block and attempt to solve an extremely challenging issue (called the "Proof-of-Work"). Each miner competes to solve the problem first in a sort of race for money (a block reward plus transaction fees).

SM's core tenet is that one adversary does not publish legitimate blocks, causing others to squander their mining efforts on the already cracked hash. Newly mined blocks are not instantly published by the self-centered miner with a hash rate of the threshold. Since honest miners are unaware of these blocks, they inadvertently use hash power to mine blocks that are likely to be replaced in the chain. This makes a selfish miner more likely to receive block rewards than an honest miner [9].

The contributions of this study:

- Determine whether the SM is viable when block propagation latency exists.
- Determine the profitability threshold α for SM.
- Determine how lucrative SM is when more than 1 pool employs the SM method, and how this affects the threshold α.
- Determine whether the SM technique can be improved to make it more profitable.
- Block propagation latencies have an impact on the profitability of the SM approach. Because we account for block propagation latencies, the simulation model is more accurate for the Bitcoin N/W. However, we discover that it makes no difference whether the N/W's Avg block transmission latency is $1 - sec$ or $10 - sec$.
- Also demonstrated that, in terms of relative income, any pool may profit from SM, if the pool achieves total N/W superiority.
- Also demonstrated that if the pool does not acquire N/W supremacy, the SM technique is only advantageous when $\alpha \geq 0.60$ if the mining difficulty is reduced.
- It makes no difference whether the selfish pool employs a distributed or centralized implementation of the SM technique. The distributed and centralized implementations behave nearly identically.
- We present and analyse a novel defense called freshness preferred (FP), in which the miners want to maximize their income (profit) from mining, HM is in their best interests since it maximizes their profit P (real revenue collected during a mining session) for raises the security bound to 60 %.
- All of that, we do automate by using the metaheuristics optimization techniques such as (SMACO & SMPSO) and determine the optimal value of the selfish threshold α, optimal execution time, optimal hash powering, and the optimal execution time to gain profits with the maximum relative revenue.

The remainder of this paper is organized as follows. Section 2 reviews the literature on mining and selfish mining in the bitcoin N/Ws. Section 3 introduces the proposed suggested architecture of (SMPSO and SMACO) Algorithms for Detect SM. Section 4 shows the experimental results and configuration. It also provides the two suggested algorithms with their performance and then compares them. Finally, section 5 concludes the paper and highlights future research.

## 2. Related work

Many relevant studies in the literature addressed selfish mining (SM) attacks in blockchain, as follows:

Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong. The two profitable SM were suggested, and it is believed that they use a minimum hash power of

about *21.48%*. The SM technique needs *51* rounds of difficulty adjustment to turn a profit when the hash rate of SMs is *22%* of the network's total. If the hash rate of the SMs drops below *33%* of the N/W total, the number of rounds lowers to *5*. The likelihood of honest miners mining after either selfish or honest chains wasn't initialized internally, hence the influence of N/W factors such as latency wasn't taken into account. An inadequate analysis was done of the impacts of block withholding assaults and simultaneous block withholding delays. [3].

M. Butucaru, and S. Solat. Proposed Zeroblock, where miners must release their blocks within a predetermined window of time. The peers in the N/W build their phony blocks and add them to their blockchain if the miners withhold their blocks for SM and fail to broadcast them in the anticipated amount of time. The N/W hash rate cannot be varied when the difficulty parameter is constant and Zeroblock is not viable. Due to hash rate fluctuations, the expected block times may vary widely and under Zeroblock, legitimate blocks may become invalid. [8].

E. Heilman. proposed the "Freshness Prefered" (FP) method, which favors newer blocks over those with older timestamps. He increases the minimum required mining power from 25% to 32% under all propagation benefits to profitably mine selfishly. Although the system's security relies on immutable timestamps, it is resilient to their manipulation [9].

I. Eyal, and E.Sirer. Published as a thorough and official analysis of the motivation behind the SM approach. First, the results are intriguing; if the SM cartel can win block races, it will only require a small amount of mining power under its control to receive more mining rewards than is justly due. Second, even if honest miners never mine on a selfish block during a block race or $\gamma = 0$, a selfish cartel will win more than its fair share of the mining rewards with ownership of *33%* of the mining power [1].

Although block withholding has been the subject of numerous papers [6, 7, 8, 9], none have looked at the impact of concurrent block withholding latency and block withholding attack on the N/W. In the field of cloud computing, a variety of metaheuristic algorithms have been proposed and used for work scheduling. There are two methods that effective metaheuristic algorithms use: The idea of a successful subversive method requiring less than *30%* of the Bitcoin N/W's total computing power is not new and dates to the early days of the system. One of the first dangers mentioned was double spending, which Nakamoto [10] addressed in the original Bitcoin paper. Since Bitcoin's remarkable rise in comparison to other cryptocurrencies, many disruptive strategies have been proposed and investigated. While [1] refers to it as SM, [2, 3] refers to it as st1, a subset of the wider *stk* family of block-discarding attacks with $k = 0, 1, 2, \ldots$ [2, 3] Concludes that the *st*1 assault is lucrative if the attacker's total processing power $p$ is more than or equal to [1]:

$$P > \frac{1 - ns}{3 - 2ns} \tag{1}$$

Where *ns* denotes what they refer to as N/W supremacy, and [1] finds that SM is profitable when the selfish pool's total computing power ($\alpha$) is constrained to be [3]:

$$\left(\frac{1 - \gamma}{3 - 2\gamma}\right) < \alpha < \frac{1}{2} \tag{2}$$

Optimizes SM using a $MDP$ technique and demonstrates that following the longest chain criterion is not necessarily the best solution. The inventors of [14] combine tactics akin to SM with N/W-level assaults such as an eclipse attack. Both [14, 15] believed that there are more profitable options available than the SM strategy presented in [1]. On the other hand, both papers [16, 13] concentrate on attacks directed toward Bitcoin N/W pools (selfish or non-selfish). While both [16, 13] make use of game-theoretical analysis, [13] focuses on $DDoS$ assaults, whereas [16] focuses on sabotage attempts aimed at reducing the revenue of the targeted pool. This paper aims to analyze the profitability of Bitcoin N/W block discarding attacks. The analysis begins with a model in which all miners are honest and adhere to Bitcoin's regulations, and all block transmissions are delayed.

Demonstrate that latency (s) produce blockchain splits, which are promptly resolved. The N/W model's simulation findings closely match those found in the real Bitcoin N/W. Relative revenue and confirmed blocks are two ways to look at this claim and use them to see if this is true for the case where there is instantaneous block transmission and for the case where there are latencies in the transmission of blocks. First, show that this claim is true. When it comes to the other case, though, SM is only profitable when the pool has more than *30%* of the total computing power on the N/W. Show that more relative income does not always mean more verified blocks and that SM hurts both honest and dishonest miners. Finally, show a generalized SM that can handle more block-discarding assaults, and utilize a genetic approach to find the best configuration for the pool. Attempt to maximize relative income or verified blockages. The results demonstrate that when properly set, generalized SM outperforms

ordinary SM in terms of relative revenue and verified blocks.

Every 2016 new block's difficulty is separately recalculated by each mining node using the following formula. [2]:

$$\text{New}_{\text{Diff}} = \text{Old}_{\text{Diff}} \times \frac{Time\ n\ Blocks}{(Time\ Target\ x\ n\ Blocks)} \quad (3)$$

Where $New_{Diff}$ represents the newly determined network difficulty and $Old_{Diff}$ represents the previous network difficulty.

This equation is used to get the technical execution time and the threshold α needed for a successful (SM) [3]:

$$\text{Threshold Needed} = \frac{1-(1-e_{600}^{-(1-\alpha)\times t})}{3-2\times(1-e_{600}^{-(1-\alpha)\times t})} \quad (4)$$

Also, these equations are used to calculate the Relative Revenue ($R_P$) for Selfish Nodes ($N_S$) and the Honest Nodes ($N_P$) [15]:

$$R_P = \frac{Ns}{(Ns+Np)} \quad (5)$$

And, The relative honest revenue ($R_O$) equation [15]:

$$R_O = 1 - R_P \quad (6)$$

## 3. Proposed architecture of (SMPSO and SMACO) algorithms for detect SM

The main chain is the branch of the blockchain that consists of a series of blocks (connected by references to the preceding block) with the greatest amount of work performed in expectation, starting with the Genesis block. To define the SM strategy, we suppose that miners are separated into two groups open groups [1] : those who adhere to Bitcoin's regulations and a minority of dishonest nodes who employ the SM method. The selfish pool's proportion of overall computing power is referred to as alpha ($\alpha$). When a dishonest node mines a block (SM Step1), the block is announced exclusively to other dishonest nodes in the selfish pool. At this point, honest nodes continue mining on the earlier public block (block with the **serial number** $(np)$), while dishonest nodes mine on the newer Hidden block : $ns = np + 1$ . The portion of the blockchain at dishonest nodes is referred to as the HiddenExtension since it contains the **secret blocks** $(ns)$.

If the selfish pool is unaware of any rival public block (block at the same depth of the blockchain as $ns$ ), it assumes a one-block lead and continues mining on the Hidden block (*ns)* in the hope of achieving a two or three-block advantage. When an honest node mines a block, it broadcasts it promptly to the rest of the N/W's nodes [12]. When the pool receives the block, it computes the lead and decides on a course of action depending on the lead's value. The lead refers to the length of the secret branch of the blockchain in comparison to the public branch. The Hidden branch is made up of the HiddenExtension and the public branch's parent (the chain of blocks beginning at the predecessor to the first Hidden block to the Genesis block). If the selfish pool's most recent public block has the serial number (*np)* and the most recent Hidden block has the serial number $ns$, then [15]:

$$lead = ns - np \quad (7)$$

If $lead < 0$, the hidden branch has slipped behind the public branch. The selfish pool might continue mining on the final block ns of the HiddenExtension, but the chances of catching up with most honest nodes are slim, therefore the selfish pool abandons the HiddenExtension and begins mining on the last known public block $(np)$. If $lead = 0$, it signifies that the hidden and public branches are the same lengths. The selfish pool promptly publishes (broadcasts a Hidden block to all honest nodes) the lone Hidden block in the extension, causing the blockchain to split (after a transmission latency) at all honest nodes and a race to determine which branch will become the main chain to commence. Depending on the N/W structure and propagation latency (*s)*, one of the two contending blocks will arrive in an honest node first [6].

Because honest nodes adhere to the Bitcoin protocol requirements, they will mine on the first block they get, whether it is a public or published block. As a result, some honest nodes will start mining on the just disclosed block, therefore expanding the previously hidden branch. The proportion of honest nodes that mine on the published block is referred to as, and it is a measure of how frequently a published block arrives first to an honest node when there is a race. Simultaneously, nodes in the selfish pool continue to mine on the published block. Eventually, a new block is mined, extending either the public or previously hidden branch. There are 3 possibilities. The selfish pool mines a block that is a successor to the publicized block *ns*.

In Fig. 1 (b) the pool wins the race and earns the payout for both blocks. On the winning block, both

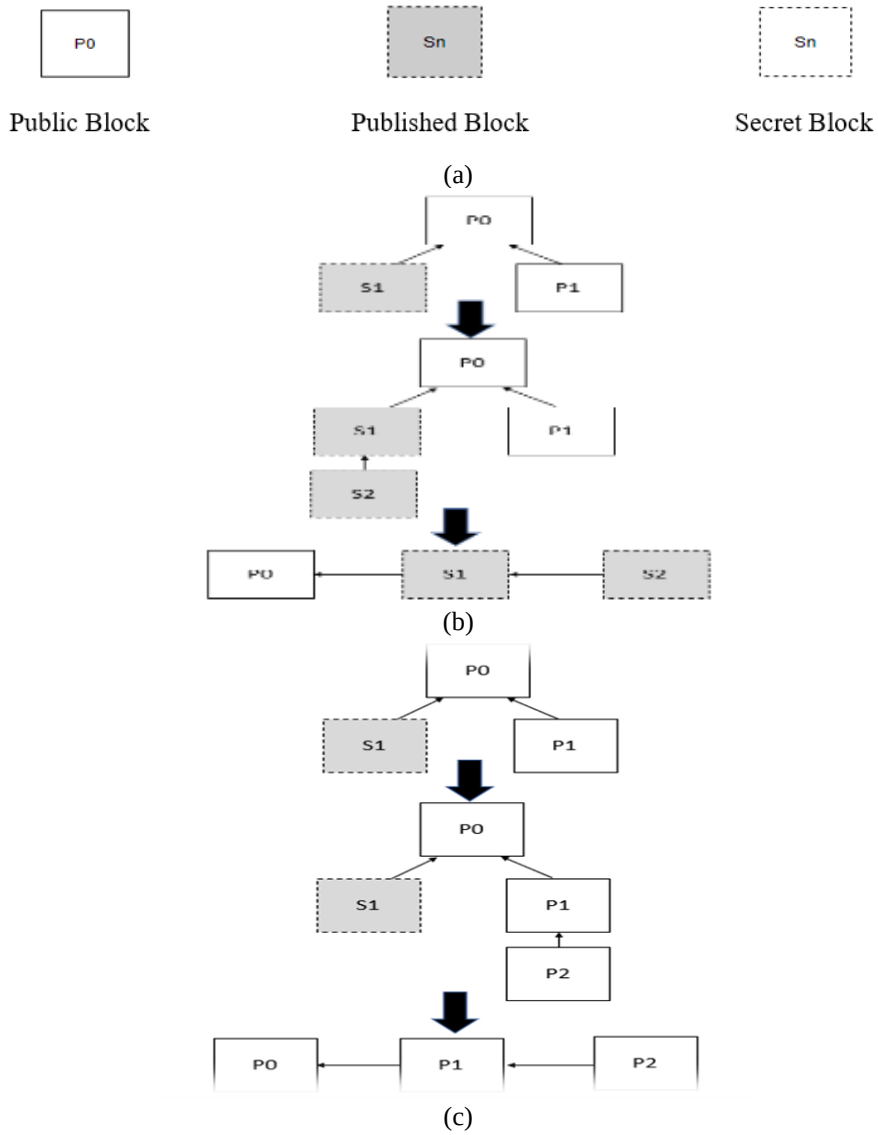


Figure. 1 The Effects of a race at a dishonest node: (a) Story, (b) Outcome 1: The SM wins the race, and (c) Outcome 2: An HM wins the race

honest and dishonest nodes begin mining. An honest node mines a block that is a successor to the publicized block ns. The poo still wins the race but gets only awarded one block, the block preceding the victory block. In Fig. 1 (c) on the winning block, both honest and dishonest nodes begin mining. An honest node mines a block that is a successor to the competing public block (*np)*. The pool loses the race and money for block (*ns)*, whilst the honest nodes receive credit for both public blocks. The pool drops the HiddenExtension, and both honest and dishonest nodes start mining on the winning block. If lead is equal to one, the hidden branch is one block longer than the public branch.

To guarantee that the Hidden blocks become part of the main chain, the pool publishes both Hidden blocks in the HiddenExtension, as seen in Fig. 2. Both blocks reach the honest nodes, and the honest nodes stop mining on the block on which they were working. If lead is more than one, the pool has a comfortable lead over the honest nodes. As seen in Fig. 3, the pool publishes the first unpublished.

### 3.1 SM in general

To apply ACO to the SM approach, we first define an artificial ant colony algorithm implementation [19, 20]. According to [2] SM is a subset of $stk$, $k = 0, 1, 2, 3, :::, \infty$, where $st0$ is honest mining, $st1$ is SM suggested in [1] and $stk$, $k > 1$ are versions of SM believed to be more lucrative than $st1$. $stk$ is less resistant to block discarding assaults than ordinary SM. The universal SM method [19, 20] is defined by the parameter vector $v$ $(ml, p2, p3, d2, d3, dsl, ra)$ shown below.

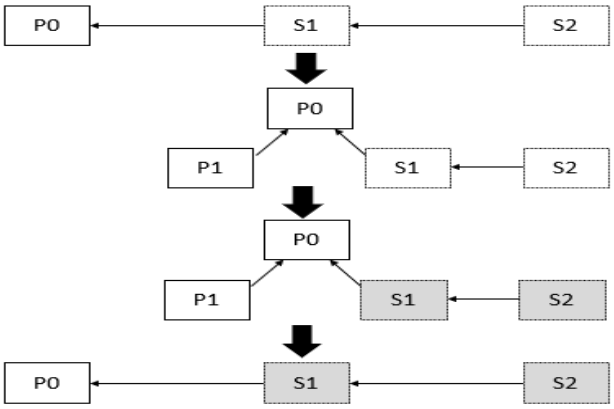


Figure. 2 A Dishonest node publishes both hidden blocks after a public block

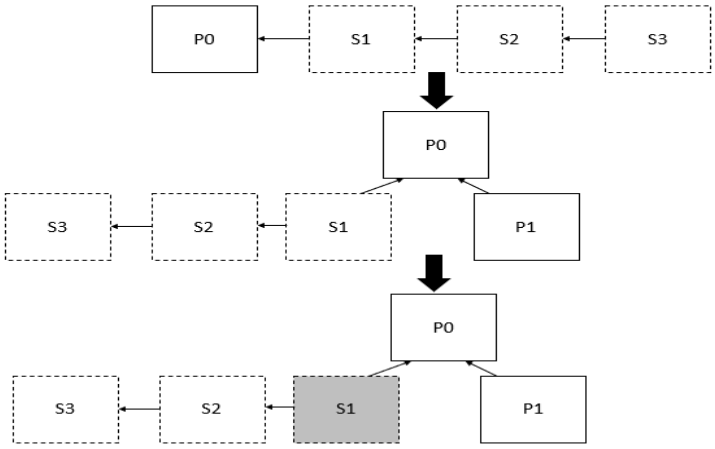


Figure 3. A public block precedes a hidden block for publishing

$ml$ is the minimum lead (length of the secret extension) required by the pool to publish one secret block after detecting another secret block. By default, $ml = 0$ indicates no blocks are published unless in response to pool learning. This is the case with the [1] SM technique.

$ml = 1$ Implies the pool publishes secret blocks instantly. This is ethical mining. When the secret extension is $ml$ long, the pool broadcasts one block. In big pools, this may be required to limit the hidden extension's length. $p2$ and $p3$ are ints. When $lead = 2\ or\ 3$, they represent the number of blocks the selfish pool discloses in response to the pool learning a new public block. To calculate the lead, divide the latest known secret and public block serial numbers by the lead.

In an N/W with communication latency, publishing more blocks than advised by [15] may be beneficial. Because public blocks may be published but not reach the pool owing to communication issues. Publishing many blocks increases the chances of the secret blocks being accepted by honest nodes. This compensates for unused public blocks [20]. $d2, d3,$ and $dsl$ are numbers between 0 and $1200 - sec$. A race or a lead of one reduces the pool's time in publishing a block. The secret block is then instantly published. When $lead$ is 2 or 3, the pool publishes a block $d2, d3\ sec$ later. The $dsl$ sec' latency is used for publishing once the pool finds a new secret block. It is only utilized when $ml > 0$. These latencies are intended to deny honest nodes the chance to mine on the main chain. This increases the pool's chances of receiving revenue for subsequent secret blocks.

$ra$ is an integer that indicates whether the pool will race when the $lead$ is $zero$. When the $lead$ is $zero$, the pool does not race. Instead, the pool instantly abandons the hidden extension and starts mining on the public chain block [5]. This may be important if the pool has a limited number of nodes and the pool's prospects of winning the race are slim, as the pool may spend computational effort mining on the previously hidden block.

$ra = 1$ Indicates that the pool enters a race when the lead is reduced to $zero$. This is true of the SM method presented by [3]. The broad SM method comprises HM, denoted by $v = (1,*,*,*,*,0,0)$, and the SM approach suggested by [3], denoted by $v =$

Table 1. SMPSO parameters

| No. | Parameters | Values |
|---|---|---|
| 1 | Number of particles | 100 |
| 2 | $C_1$ | 3 |
| 3 | $C_2$ | 3 |
| 4 | $R_1$ | [0-1] |
| 5 | $R_2$ | [0-1] |
| 6 | $W_{max}$ | 0.8 |
| 7 | $W_{min}$ | 0.3 |
| 8 | Number of iterations | 1000 |
| 9 | $W$ | 1 |

Table 2. SMACO parameters

| No. | Parameters | Values |
|---|---|---|
| 1 | $\alpha$ | 1 |
| 2 | $\beta$ | 2 |
| 3 | $p$ | 0.4 |
| 4 | $q$ | 1 |
| 5 | $m$ | 80 |
| 6 | $t_k$ | 1000 |
| 7 | $p_{ro}$ | 0.8 |

$(0, 1, 1, 0, 0, 0, 1)$, where 2 denotes an inapplicable parameter. With this information, we can now utilize a SMACO to determine an optimal configuration of the parameter vector $v$ that results in $near-maximal$ income for the selfish pool.

### 3.2 The algorithm for SM in general

***procedure*** Initialize
read $ml, p2, p3, d2, d3, dsl, ra$
secretExtension ← empty
race ← $F$
mine on the last block of the chain
***end procedure***
***procedure*** SM (block)
append the block to secretExtension:
$ns = ns + 1$
$lead \leftarrow ns - np$. The last block of secretExtension has serial $ns$
***if*** race = $T$ ***and*** $ra = 1$ ***then***
PUBLISH $(1, 0)$
race ← $F$
secretExtension ← $empty$
***else if*** $lead \geq ml > 0$ ***then***
PUBLISH $(1, dsl)$
***end if***
mine on block
***end procedure***
procedure HM (block) append the block to the blockchain
$np \leftarrow np + 1$. The last public block has serial $np$
$lead \leftarrow ns - np$. The last block of secretExtension has serial $ns$
***if*** $lead < 0$ or $(lead = 0$ ***and*** $ra = 0)$***then***
race ← $F$
secretExtension ← $empty$
mine on block $np$
***else if*** $lead = 0$ ***and*** $ra = 1$ ***then***
race ← $T$
PUBLISH $(1, 0)$
mine on block $ns$
***else if*** $lead = 1$ ***then***
PUBLISH $(2, 0)$
mine on block $ns$
***else if*** $lead = 2$ ***then***
PUBLISH $(p2, d2)$
mine on block $ns$
***else***
PUBLISH $(p3, d3)$
mine on block $ns$
***end if***
***end procedure***
***procedure*** Publish
$($***integer*** $n$, ***integer*** $latency)$
remove $n$ blocks from secretExtension
publish $n$ blocks after latency $seconds$
***end procedure***

The suggested system incorporates two different algorithms: SMPSO and SMACO [4, 19, 20]. These algorithms are performed on SM. The execution time, hash mining power, threshold, and relative revenue availability of this method are compared. The task scheduling in SM using the SMACO algorithm was found to be faster than that using the SMPSO algorithm. It spent a shorter time completing the different scheduling tasks. In addition, the SMACO provided better results for large-scale optimization problems, than the SMPSO. However, many techniques and strategies were developed to improve the SMACO for task scheduling. The combination of SMACO and SMPSO algorithms improves the performance, convergence speed, and resource utilization ratio, providing near-optimal solutions within a reasonable amount of execution time.

## 4. Results and simulations

Extended our simulator to use a java apache NetBeans IDE 16. These results consider the experimental that is calculated by equations. Modified the simulator to search for an ideal configuration of the parameter vector v that delivers superior performance for the selfish pool using a SMACO library. Used rank-based selection to search SMACO. In the $first$ set of tests, maximized $Rp$. In the second set of examinations, the maximized of $Ns$ (the total number of confirmed blocks mined by the pool that end up in the main chain). Contrary to popular belief, a higher relative revenue does not necessarily imply more confirmed blocks in the main chain [2, 1, 15]. The parameter vector (v) was also constrained to $1 \leq ml \leq 15, 1 \leq p2, p3 \leq 3, and\ 0 \leq d2, d3, dsl \leq 1200$. Other SMACO configuration details are as follows:

### 4.1 SMACO algorithmus:

***procedure*** SMACO $(ngen, popSize, pcross, pmut, nconvergence)$
$benchmarkScore \leftarrow$ score of standard SM
$gen \leftarrow 0$
converged $= F$
randomly generate $popSize$ genomes $G\ (gen)$
***while*** $gen < ngen$ and converged $= F$ ***do***
***for all*** $g \in G\ (gen)$***do***
$g\ (rank) \leftarrow OBJECTIVEFUNCTION\ (g)$
***end for***
select $pcross$% genomes in $G\ (gen)$ based on rank to mate.
replace $pcross$% genomes in $G\ (gen)$ with new offspring.
$gen \leftarrow gen + 1$
mutate $pmut$% of genomes in $G\ (gen)$
***if*** $nconvergence \geq gen$ ***then***
***if*** algorithm has converged ***then***
converged $\leftarrow T$
***end if***
***end if***
***end while***
***return*** best genome
***end procedure***
***procedure*** $getScore\ (ml, p2, p3, d2, d3, dsl, ra)$
run the general simulation with the given parameters.
***return*** $Rp\ or\ Ns$
***end procedure***
***procedure*** Objective Function $(g)$

$$val\ GETSCORE\begin{pmatrix} g(ml), g(p2), g(p3), g(d2), \\ g(d3), \\ g(dsl), g(ra) \end{pmatrix}$$

***return*** $val$ − benchmarkScore
***end procedure***

The first set of tests, which aimed to maximize $Rp$, is summarized in Tables 3 and 4. On the other hand, Tables 5 and 6 illustrate the SMPSO optimization outcomes while seeking to maximize $Ns$. Only list the $ones$ in the tables that the SMACO was randomly chosen from a group of equally ranked top persons after each search for a given. Column $H$ implements the pool when using an HM strategy, Column $S$ implements the pool when using the standard SM strategy, and column $G$ implements the pool when using the general SM strategy with the parameter vector $v$ configured as shown in the columns labeled configuration (general SM). We make the following observations based on the data in Tables 3–6.

- Increasing $Rp$ may not necessarily increase $Ns$.
- Standard SM always outperforms HM in terms of $Ns$, even though $Rp$ is more when the pool follows SM (Column$S$) (Column $H$).
- There exist configurations of the parameter vector $v$ that provide a greater $Rp$ than the usual SM [1] while increasing the value of $Ns$, such that $Ns\ (selfish) \geq Ns\ (honest) for\ all\ \alpha$.
- $Ns$ when the latency is $equal\ to\ zero$ is greater than $Ns$ when the latency is higher than $zero$ (latency as given by the equation latency $= kAN\ (\mu, 6)$, with $k$ equal to $one$).
- Even when the pool is tiny ( $\alpha \approx 0.06$ ), employing a well-configured general SM approach enables the pool to perform somewhat well than or on par with HM in terms of $Rp$ and $Ns$.

The selfish pool can increase income by utilizing the additional degrees of freedom provided by the parameters $(ml, p2, p3, d2, d3, dsl,$ and $ra)$. For instance, by opting out of races when $lead = 0$, the pool can avoid spending computing resources mining on the hidden branch. This can be advantageous in an N/W with block transmission latencies (see Tables 1 and 3) when the pool has a limited number of nodes, which reduces the likelihood of winning the race. Since the N/W has only $100 - nodes, \gamma < 0.6$ as seen in Fig. 4 even when $\alpha = 0.6$ (only $20 - nodes$ are dishonest).

In Fig. 5, the pool's mining efficiency $(Ps)$ is depicted for HM, the ordinary SM approach, and the broad SM method with $v = (5, 3, 3, 781, 19, 1158, 0)$. We observe that when the pool is highly connected ( $k = 0$, implying that $ns = 1$), the general SM strategy performs almost identically to HM, but consistently slightly better

Table 3. SMACO algorithm with instantaneous block transmission

| | $N_s$ | | | $R_p$ | | | Configuration SM | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha$ | H | S | G | H | S | G | $M_1$ | $p_2$ | $p_3$ | $d_2$ | $d_3$ | dsl | Ra |
| 0.05 | 255 | 46 | 253 | 0.05 | 0.01 | 0.05 | 2 | 3 | 1 | 645 | 527 | 179 | 0 |
| 0.10 | 505 | 165 | 495 | 0.10 | 0.04 | 0.10 | 2 | 3 | 1 | 645 | 527 | 179 | 0 |
| 0.15 | 763 | 344 | 763 | 0.15 | 0.08 | 0.16 | 13 | 2 | 1 | 805 | 85 | 1186 | 0 |
| 0.20 | 1005 | 547 | 1004 | 0.20 | 0.13 | 0.22 | 5 | 2 | 1 | 461 | 908 | 927 | 0 |
| 0.25 | 1255 | 799 | 1255 | 0.25 | 0.19 | 0.27 | 13 | 2 | 1 | 805 | 85 | 1186 | 0 |
| 0.30 | 1493 | 1061 | 1493 | 0.30 | 0.27 | 0.33 | 13 | 2 | 1 | 805 | 85 | 1186 | 0 |
| 0.35 | 1743 | 1357 | 1738 | 0.35 | 0.36 | 0.41 | 14 | 1 | 1 | 1176 | 94 | 584 | 0 |
| 0.40 | 1995 | 1665 | 1966 | 0.40 | 0.45 | 0.51 | 14 | 1 | 1 | 1176 | 94 | 584 | 0 |
| 0.45 | 2243 | 1987 | 2063 | 0.45 | 0.56 | 0.65 | 14 | 1 | 1 | 1176 | 94 | 584 | 0 |
| 0.50 | 2365 | 2140 | 2365 | 0.50 | 0.61 | 0.73 | 14 | 1 | 1 | 1176 | 94 | 584 | 0 |
| 0.55 | 2370 | 2144 | 2369 | 0.55 | 0.70 | 0.81 | 15 | 1 | 1 | 1179 | 98 | 588 | 0 |
| 0.60 | 2420 | 2177 | 2379 | 0.60 | 0.79 | 0.90 | 15 | 1 | 1 | 1186 | 98 | 588 | 0 |

Table 4. SMACO algorithm with block transmission latencies

| | $N_s$ | | | $R_p$ | | | Configuration SM | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha$ | H | S | G | H | S | G | $M_1$ | $p_2$ | $p_3$ | $d_2$ | $d_3$ | dsl | Ra |
| 0.05 | 252 | 79 | 252 | 0.05 | 0.02 | 0.05 | 2 | 3 | 1 | 645 | 527 | 179 | 0 |
| 0.10 | 496 | 223 | 494 | 0.10 | 0.05 | 0.10 | 2 | 3 | 1 | 645 | 527 | 169 | 0 |
| 0.15 | 753 | 442 | 761 | 0.15 | 0.10 | 0.16 | 13 | 2 | 1 | 805 | 85 | 1186 | 0 |
| 0.20 | 994 | 673 | 986 | 0.20 | 0.16 | 0.22 | 12 | 2 | 3 | 19 | 560 | 372 | 0 |
| 0.25 | 1242 | 957 | 1251 | 0.25 | 0.24 | 0.27 | 7 | 2 | 1 | 998 | 292 | 33 | 0 |
| 0.30 | 1476 | 1216 | 1477 | 0.30 | 0.31 | 0.33 | 5 | 2 | 1 | 461 | 908 | 927 | 0 |
| 0.35 | 1724 | 1493 | 1740 | 0.35 | 0.39 | 0.41 | 14 | 1 | 1 | 1176 | 94 | 584 | 0 |
| 0.40 | 1974 | 1795 | 1988 | 0.40 | 0.49 | 0.51 | 14 | 1 | 1 | 1176 | 94 | 584 | 0 |
| 0.45 | 2220 | 2085 | 2136 | 0.45 | 0.60 | 0.65 | 15 | 3 | 1 | 489 | 19 | 692 | 1 |
| 0.50 | 2342 | 2223 | 2293 | 0.50 | 0.70 | 0.73 | 15 | 3 | 1 | 489 | 19 | 692 | 1 |
| 0.55 | 2350 | 2354 | 2342 | 0.55 | 0.76 | 0.78 | 16 | 3 | 1 | 490 | 80 | 692 | 1 |
| 0.60 | 2355 | 2355 | 2344 | 0.60 | 0.77 | 0.79 | 17 | 3 | 1 | 499 | 80 | 699 | 1 |

Table 5. SMPSO algorithm with instantaneous block transmission

| | $N_s$ | | | $R_p$ | | | Configuration SM | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha$ | H | S | G | H | S | G | $M_1$ | $p_2$ | $p_3$ | $d_2$ | $d_3$ | dsl | Ra |
| 0.05 | 255 | 46 | 253 | 0.05 | 0.01 | 0.05 | 2 | 3 | 1 | 656 | 528 | 179 | 0 |
| 0.10 | 506 | 165 | 505 | 0.10 | 0.04 | 0.10 | 13 | 2 | 1 | 806 | 86 | 1187 | 0 |
| 0.15 | 763 | 344 | 763 | 0.15 | 0.08 | 0.16 | 13 | 2 | 1 | 806 | 86 | 1187 | 0 |
| 0.20 | 1005 | 557 | 1003 | 0.20 | 0.13 | 0.22 | 8 | 2 | 2 | 348 | 24 | 160 | 0 |
| 0.25 | 1255 | 799 | 1256 | 0.25 | 0.20 | 0.27 | 14 | 2 | 1 | 805 | 85 | 1186 | 0 |
| 0.30 | 1493 | 1061 | 1493 | 0.30 | 0.28 | 0.33 | 14 | 2 | 1 | 805 | 85 | 1186 | 0 |
| 0.35 | 1743 | 1357 | 1738 | 0.35 | 0.36 | 0.41 | 14 | 2 | 1 | 805 | 85 | 1186 | 0 |
| 0.40 | 1995 | 1665 | 1995 | 0.40 | 0.45 | 0.51 | 9 | 2 | 1 | 1087 | 292 | 109 | 0 |
| 0.45 | 2253 | 1987 | 2224 | 0.45 | 0.57 | 0.65 | 9 | 2 | 1 | 1087 | 292 | 109 | 0 |
| 0.50 | 2365 | 2150 | 2367 | 0.50 | 0.61 | 0.73 | 8 | 2 | 1 | 998 | 294 | 33 | 0 |
| 0.55 | 2369 | 2160 | 2367 | 0.55 | 0.66 | 0.75 | 8 | 2 | 1 | 1088 | 297 | 33 | 0 |
| 0.60 | 2380 | 2178 | 2370 | 0.60 | 0.69 | 0.78 | 8 | 2 | 1 | 1088 | 297 | 33 | 0 |

than HM or significantly better than the standard SM strategy in the remaining scenarios.

Fig. 6, shows the comparison between the SMACO strategy and the SMPSO strategy based on the optimal amount of data with the total number of execution times and shows the sum of latencies.

Fig. 7, shows the optimal threshold needed for the SMACO strategy and SMPSO strategy and a comparison between them and calculates which one is better.

Fig. 8, shows the optimal revenue with the threshold α and compared the results between both algorithms (SMACO and SMPSO) and selected which one is better.

Fig. 9, shows the optimal number of nodes with the better cost (reduced) for comparison between the SMACO algorithm and SMPSO algorithm.

TABLE 6: SMPSO algorithm with block transmission latencies

| $\alpha$ | $N_s$ | | | $R_p$ | | | Configuration SM | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | H | S | G | H | S | G | $M_1$ | $p_2$ | $p_3$ | $d_2$ | $d_3$ | Dsl | Ra |
| 0.05 | 252 | 79 | 253 | 0.05 | 0.03 | 0.05 | 3 | 3 | 1 | 656 | 528 | 179 | 0 |
| 0.10 | 497 | 233 | 504 | 0.10 | 0.06 | 0.10 | 8 | 2 | 3 | 979 | 335 | 53 | 0 |
| 0.15 | 755 | 452 | 762 | 0.15 | 0.10 | 0.16 | 14 | 2 | 1 | 806 | 86 | 1187 | 0 |
| 0.20 | 995 | 683 | 1002 | 0.20 | 0.16 | 0.22 | 14 | 2 | 1 | 806 | 86 | 1187 | 0 |
| 0.25 | 1252 | 967 | 1248 | 0.25 | 0.24 | 0.27 | 14 | 2 | 1 | 806 | 86 | 1187 | 0 |
| 0.30 | 1486 | 1226 | 1480 | 0.30 | 0.33 | 0.33 | 4 | 2 | 1 | 505 | 132 | 99 | 0 |
| 0.35 | 1744 | 1494 | 1733 | 0.35 | 0.39 | 0.41 | 6 | 2 | 3 | 1093 | 44 | 749 | 0 |
| 0.40 | 1976 | 1796 | 1980 | 0.40 | 0.49 | 0.51 | 6 | 3 | 3 | 792 | 19 | 1168 | 0 |
| 0.45 | 2221 | 2096 | 2226 | 0.45 | 0.50 | 0.65 | 7 | 1 | 3 | 989 | 35 | 64 | 0 |
| 0.50 | 2352 | 2234 | 2352 | 0.50 | 0.70 | 0.73 | 5 | 2 | 2 | 956 | 30 | 1182 | 0 |
| 0.55 | 2355 | 2240 | 2355 | 0.55 | 0.78 | 0.75 | 5 | 2 | 2 | 967 | 34 | 65 | 0 |
| 0.60 | 2358 | 2245 | 2358 | 0.60 | 0.80 | 0.80 | 7 | 2 | 2 | 987 | 37 | 1190 | 0 |

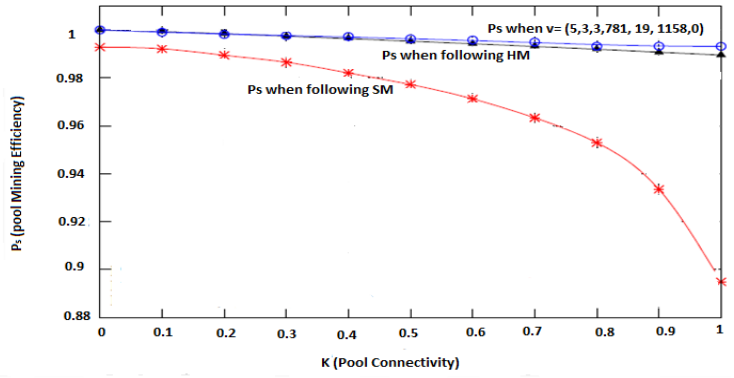


Figure. 4 SMACO calculating $\gamma$ When $\boldsymbol{\alpha = 0.6}$ N/W of varying sizes with an Avg communication latency of $\boldsymbol{10 - sec}$.

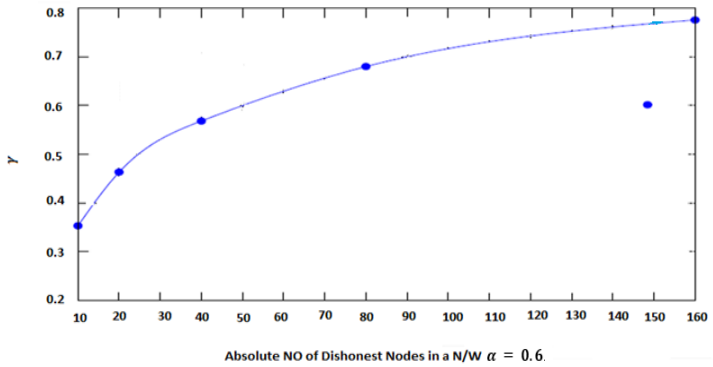


Figure. 5 SMACO performance of pool ($\boldsymbol{Ps}$) with varying $\boldsymbol{k}$ for $\boldsymbol{\alpha = 0.6}$

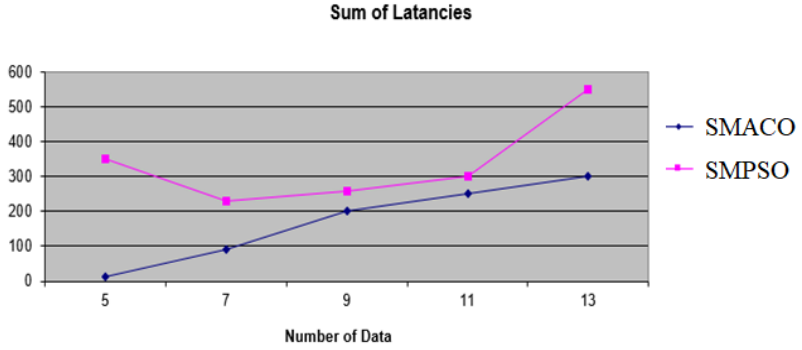


Figure. 6 SMACO and SMPSO execution time compared with the number of data

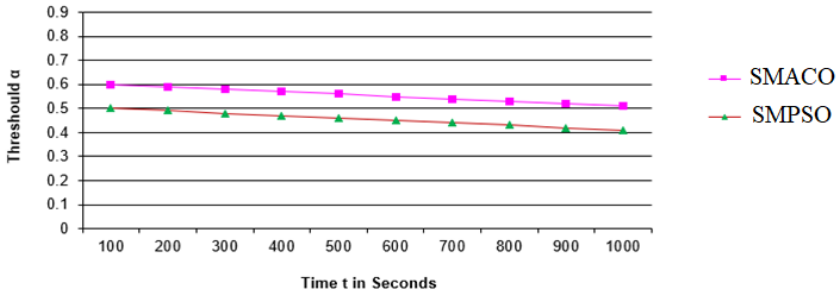


Fig. 7 SMACO and SMPSO the threshold α needed for detecting SM

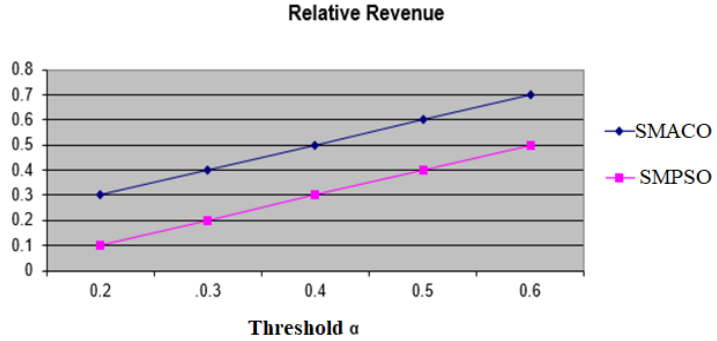


Figure. 8 SMACO and SMPSO relative revenue compared with threshold α

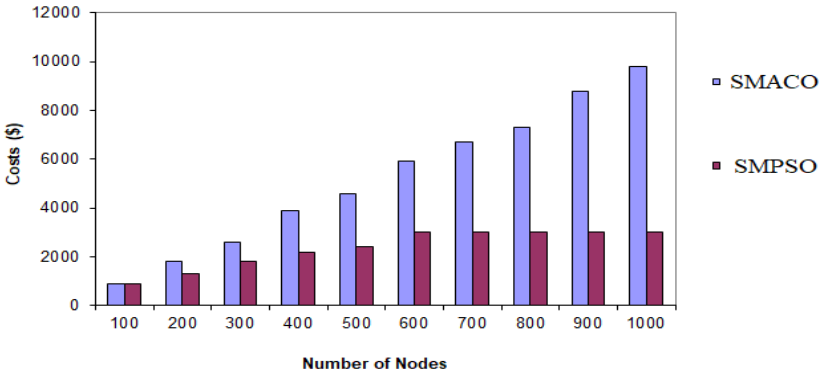


Figure. 9 SMACO and SMPSO number of nods and costs comparison

## 5. Conclusion

SM behavior is negatively affecting the cryptocurrency mining process by allowing a few groups of people SM to gain unfair which leads more HM to quit the mining process due to less profitability. Fewer miners on blockchain will harm the process and reduce the integrity of the blockchain concept and open more windows for different types of attacks. This research applies the two metaheuristics optimization algorithms (SMACO and SMPSO) to the mining process and applies it to a previous solution of SM detection to contribute and reduce SM behavior in Blockchain, analyzing their performance in terms of both runtime efficiency and solution quality, and allowing the Bitcoin community to respond and fixed the damage. The suggested system architecture was constructed and implemented using the bitcoin mining process. The simulation results showed that the proposed algorithms were more efficient and better than the compared algorithms. The metaheuristics optimization algorithms are leveraged to achieve high availability and optimize the performance of the threshold, and the relative revenue, raising the minimum share of hash mining power necessary to profitably SM In future work, the suggested system

architecture will be assessed using a real computing environment. Also, the SM will be improved to optimize costs, storage space, waiting times, data availability, performance, and access speed for mining. The preliminary work is to validate the freshness preferred (FP) strategy using bitcoin discrete event simulators. At this point, we are not convinced that existing discrete event simulators offer the necessary fidelity to provide additional validation of our arguments. We are working on a more accurate Bitcoin simulator to fill this role. The results show that SMACO has better solution quality while SMPSO has a faster execution time and gains unfair profits in comparison with other algorithms. We presented the generalized SM method, which is based on the parameter vector $v = (ml, p2, p3, d2, d3, dsl, ra)$. Using the SMACO algorithm, searching for the ideal configuration of $v$ that produces greater pool performance than ordinary SM. We demonstrated that the optimal configuration of the general SM strategy discovered by the SMACO outperformed the standard SM strategy under all conditions, both in terms of total blocks mined by the pool that ends up in the main chain at the end of the simulation and relative revenue of the pool. Using the same criteria, demonstrated that the broad SM technique performed as well as, but not better than, HM. Finally the conclusion of this reserch the SM is profitable when the threshold is 0.6 and the results appear that the ACOSM optimization is better than PSOSM optimization. It would be intriguing to find out whether this might affect our results in any way, providing a chance for potential future research. Insightful analysis of potential and previously recommended countermeasures to these block discarding attacks should also be conducted. The ACO optimisation algorithm and other metaheuristic optimisation algorithms can be compared in the future to produce the best outcomes.

## Conflicts of interest

The authors declare no conflict of interest in this research.

## Author contributions

Conceptualization: Amira Hamdi, Osama Badawy, Saleh Mesbah, Gamal Abd Elnasser. Methodology: Amira Hamdi, Osama Badawy, Saleh Mesbah, Gamal Abd Elnasser. Software: Amira Hamdi, Gamal Abd Elnasser. Validation: Saleh Mesbah, Gamal Abd Elnasser. Formal Analysis: Amira Hamdi. Investigation: Amira Hamdi. Resources: Amira Hamdi, Saleh Mesbah. Data Curation: Amira Hamdi, Saleh Mesbah. Writing - Original Draft: Amira Hamdi. Writing - Review & Editing: Amira Hamdi, Osama Badawy, Saleh Mesbah, Gamal Abd Elnasser. Visualization: Amira Hamdi, Osama Badawy, Saleh Mesbah, Gamal Abd Elnasser. Supervision: Osama Badawy, Saleh Mesbah, Gamal Abd Elnasser.

## References

[1] I. Eyal and E. Sirer, “Majority is Not Enough: Bitcoin Mining is Vulnerable”, In: *Proc. of International Conf. on Financial Cryptography and Data Security, Berlin*, pp. 436–454, 2014, doi: 10.1007/978-3-662-45472-5_28.

[2] L. Bahack, “Theoretical Bitcoin Attacks with Less Than Half of The Computational Power (Draft)”, In: *Proc. of International Conf. on Cryptography and Security*, Israel, pp.1–18, 2013, doi: 10.14419/ijet.v7i2.3.9957.

[3] Q. Bai, X. Zhou, X. Wang, Y. Xu, X. Wang, and Q. Kong, “A Deep Dive Into Blockchain Selfish Mining”, In: *Proc. of International Conf. IEEE on Communications*, China, pp. 1–6, 2019, doi: 10.1109/ICC.2019.8761240.

[4] N. Bharanidharan and H. Rajaguru, “Performance Enhancement of Swarm Intelligence Techniques in Dementia Classification Using Dragonfly-Based Hybrid Algorithms”, *International Journal of Image System Technology*, Vol. 30, No. 1, pp. 57–74, Mar. 2020.

[5] J. Göbel, P. Keeler, A. Krzesinski, and P. Taylor, “Bitcoin Blockchain Dynamics: The Selfish-Mine Strategy in The Presence of Propagation Latency”, *International Journal of a Peer-Reviewed*, South Africa, pp. 1–14, 2015.

[6] J. Gobel, H. Keeler, P. Taylor, and A. Krzesinski, “Bitcoin Blockchain Dynamics: The Selfish-Mine Strategy in The Presence of Propagation Latency”, *International Journal of Advances in Applied Probability*, Vol. 104, pp. 23–41, 2016, doi: 10.1016/j.peva.2016.07.001.

[7] T. Bradford and W. Keeton, “New Person-to-Person Payment Methods: Have Checks et Their Match”, In: *Proc. of International Conf. Federal Reserve Bank of Kansas City Economic Review Third Quarter*, Vol. 97, No. 3, pp.1–38, 2012.

[8] S. Solat and M. Butucaru, “ZeroBlock: Timestamp-Free Prevention of Block-Withholding Attack in Bitcoin”, In: *Proc. of Cryptography and Security International Conference*, France, Vol. 1, pp. 1-11, 2017.

[9] E. Heilman, “One Weird Trick to Stop Selfish Miners: Fresh Bitcoins, A Solution for the Honest Miner (Poster Abstract)”, In: *Proc. of*

*Financial Cryptography Conf. on Data Security*, pp. 161-162, 2014, doi: 10.1007/978-3-662-44774-1_12.

[10] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", *International Journal of Modern Economy*, pp. 1-9, 2009, Available at: https://bitcoin.org/bitcoin.pdf.

[11] O. Hegazy, O. Soliman, and M. A. Salam, "Comparative Study between FPA, BA, MCS, ABC, and PSO Algorithms in Training and Optimizing of LS-SVM for Stock Market Prediction", *International Journal of Advanced Computer Research*, Vol. 5, No. 18, pp. 35-45, 2015.

[12] N. Courtois, "On the Longest Chain Rule and Programmed Self-Destruction of Crypto Currencies", *International Journal of Politically Research – Applied Cryptography Blog*, London, pp.1–90, 2014.

[13] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore, "Game-Theoretic Analysis of DDOS Attacks Against Bitcoin Mining Pools", In: *Proc. of International Conf. on Financial Cryptography and Data Security*, pp. 1–15, 2014, doi: 10.1007/978-3-662-44774-1_6.

[14] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn Mining: Generalizing Selfish Mining and Combining with an Eclipse Attack", In: *Proc. of International Conf. IEEE European Symposium on Security and Privacy*, Germany, pp. 1–16, 2016, doi: 10.1109/EuroSP.2016.32.

[15] M. Mwale, "Modelling the Dynamics of the Bitcoin Blockchain", *Thesis: MSc in Computer Science*, Suid Afrika, pp. 1–101, 2016.

[16] I. Eyal, "The Miner's Dilemma", In: *Proc. of International Conf. on IEEE Symposium on Security and Privacy (SP)*, pp. 89–103, 2015, doi: 10.1109/SP.2015.13.

[17] Y. Moon, H. Yu, J. Gil, and J. Lim, "A Slave Ants Based Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing Environments", *International Journal of Human-Centric Computing and Information Sciences*, Vol. 7, No. 28, pp. 1-10, 2017.

[18] N. Navimipour and B. Milani, "Replica Selection in The Cloud Environments Using an Ant Colony Algorithm", In: *Proc. of International Conf. on Control Engineering and Communication Technology*, Moscow, Russia, pp. 1-9, 2016.

[19] A. Awad, R. Salem, H. Abdelkader, M. A. Salam, "A Novel Intelligent Approach for Dynamic Data Replication in Cloud Environment", In: *Proc. of International Conf. On IEEE Communications*, Vol. 9, pp. 40240–40254, 2021, doi: 10.1109/ACCESS.2021.3064917.

[20] A. Awad, R. Salem, H. Abdelkader, M. A. Salam, "A Swarm Intelligence-based Approach for Dynamic Data Replication in a Cloud Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 2, pp. 271–286, 2021, doi: 10.22266/ijies2021.0430.24.