CS 535 Term Project

Furkan Reha Tutaş
Master Student
Yenisehir District
Pendik/Istanbul
+905318479707
furkanreha@sabanciuniv.edu

PAPER INTODUCTION

The paper [1] tries to explore game theory to model the situation for wireless network with malicious nodes and solve the problem of secure wireless communications by using a game-based approach; several efficient game-based algorithms that support reliable and secure wireless communications against the attacks of malicious nodes in the wireless sensor networks. Additionally, the paper evaluates provided algorithm with a simulation experiment and analyzes the simulation results in detail (Mao, Zhu, & Wei, p.1, 2013).

ABSTRACT

For the term project of Wireless Network Security Course (coded as CS 535) at Sabanci University, this report tries to; understand the paper, replicate the provided method, replicate the provided simulation for the method and compare replicated and provided simulation results.

1.NETWORK MODEL

In the paper [1], WSN is assumed to be a simple-undirected-unweighted graph that consists of k nodes with static locations and having at least one edge.

In the WSN, there can be two different types of nodes: normal and malicious. Each node in the WSN has 4 different types of actions: Forward, Receive, Detect /Jam (depends on node type), and Sleep.

An assumption about WSN is that the number of normal nodes is much larger than the number of malicious nodes. Another assumption is that each node in WSN knows only the type of itself (Mao, Zhu, & Wei, p.2, 2013).

2.GAME THEORETIC FORMULATION

In the paper [1], to calculate payoffs for each node in WSN by game theory, pairwise simultaneous games are constructed between the node and its neighbors.

Pairwise simultaneous games' payoffs are calculated for each node. After that, according to the rules of spatial structured game, payoffs are recalculated (Mao, Zhu, & Wei, p.2, 2013).

2.2 Pairwise Simultaneous Game

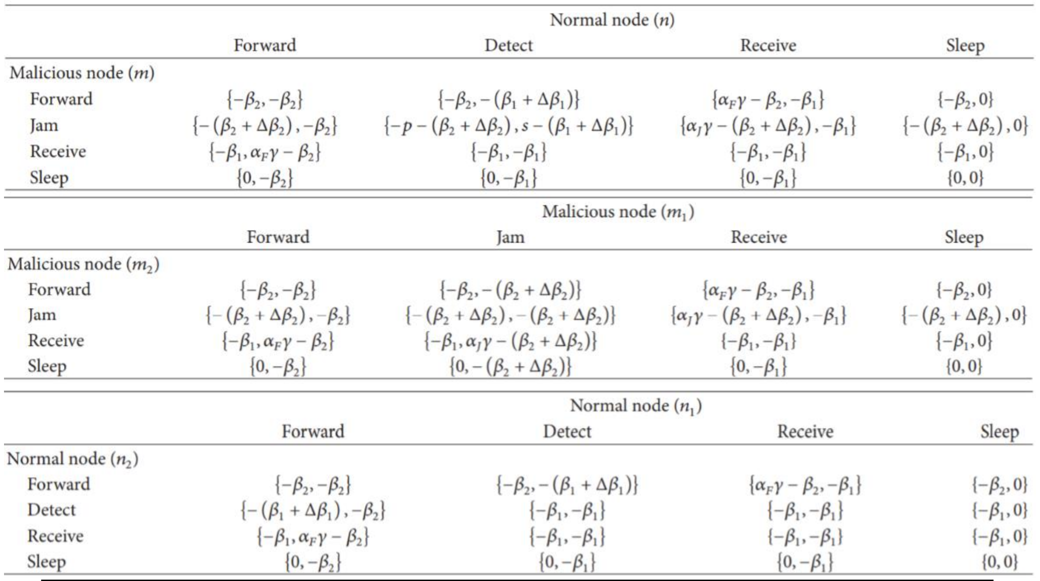
The paper [1] states that between any two neighbors, there is a simultaneous game where each node has four action types represented with the first letter of the actions [F, R, D/J, S].

[Table 1] shows some of the defined variables for the game-theoretic formulation.

Denotation	Meaning
Y	Direct income for a node to forward successfully.
A_F	Probability that a node forwards successfully.
A_J	Probability that a node jams successfully.
P	Punishment to a detected jamming
S	Stimulus to a successful jamming detection of the normal node
B1	Cost to receive a packet or detect jamming
ΔB1	Additional cost to detect jamming
B2	Cost to forward a packet or jam
ΔB2	Additional cost to jam

[Table 1: Defined variables for the game theoretic formulation (Mao, Zhu, & Wei, p.2, 2013)]

The paper [1] specifies payoffs for three different scenarios for two-nodes games. Normal-Normal, Malicious-Normal (vice versa) and Malicious-Malicious. These payoffs can be seen in [Figure 1].



	Normal node (n)			
	Forward	Detect	Receive	Sleep
Malicious node (m)				
Forward	$\{-\beta_2, -\beta_2\}$	$\{-\beta_2, -(\beta_1+\Delta\beta_1)\}$	$\{\alpha_F\gamma-\beta_2, -\beta_1\}$	$\{-\beta_2, 0\}$
Jam	$\{-(\beta_2+\Delta\beta_2), -\beta_2\}$	$\{-p-(\beta_2+\Delta\beta_2), s-(\beta_1+\Delta\beta_1)\}$	$\{\alpha_J\gamma-(\beta_2+\Delta\beta_2), -\beta_1\}$	$\{-(\beta_2+\Delta\beta_2), 0\}$
Receive	$\{-\beta_1, \alpha_F\gamma-\beta_2\}$	$\{-\beta_1, -\beta_1\}$	$\{-\beta_1, -\beta_1\}$	$\{-\beta_1, 0\}$
Sleep	$\{0, -\beta_2\}$	$\{0, -\beta_1\}$	$\{0, -\beta_1\}$	$\{0, 0\}$

	Malicious node (m_1)			
	Forward	Jam	Receive	Sleep
Malicious node (m_2)				
Forward	$\{-\beta_2, -\beta_2\}$	$\{-\beta_2, -(\beta_2+\Delta\beta_2)\}$	$\{\alpha_F\gamma-\beta_2, -\beta_1\}$	$\{-\beta_2, 0\}$
Jam	$\{-(\beta_2+\Delta\beta_2), -\beta_2\}$	$\{-(\beta_2+\Delta\beta_2), -(\beta_2+\Delta\beta_2)\}$	$\{\alpha_J\gamma-(\beta_2+\Delta\beta_2), -\beta_1\}$	$\{-(\beta_2+\Delta\beta_2), 0\}$
Receive	$\{-\beta_1, \alpha_F\gamma-\beta_2\}$	$\{-\beta_1, \alpha_J\gamma-(\beta_2+\Delta\beta_2)\}$	$\{-\beta_1, -\beta_1\}$	$\{-\beta_1, 0\}$
Sleep	$\{0, -\beta_2\}$	$\{0, -(\beta_2+\Delta\beta_2)\}$	$\{0, -\beta_1\}$	$\{0, 0\}$

	Normal node (n_1)			
	Forward	Detect	Receive	Sleep
Normal node (n_2)				
Forward	$\{-\beta_2, -\beta_2\}$	$\{-\beta_2, -(\beta_1+\Delta\beta_1)\}$	$\{\alpha_F\gamma-\beta_2, -\beta_1\}$	$\{-\beta_2, 0\}$
Detect	$\{-(\beta_1+\Delta\beta_1), -\beta_2\}$	$\{-\beta_1, -\beta_1\}$	$\{-\beta_1, -\beta_1\}$	$\{-\beta_1, 0\}$
Receive	$\{-\beta_1, \alpha_F\gamma-\beta_2\}$	$\{-\beta_1, -\beta_1\}$	$\{-\beta_1, -\beta_1\}$	$\{-\beta_1, 0\}$
Sleep	$\{0, -\beta_2\}$	$\{0, -\beta_1\}$	$\{0, -\beta_1\}$	$\{0, 0\}$

[Figure 1: Specified payoffs for each possible scenario between any two types of nodes (Mao, Zhu, & Wei, p.3-4, 2013).]

[Figure 1] can be simplified with the following [pseudo code 1].

```
def calculatePayOff(action):
    if action.type == F:
        if action.success = True:
            return $A_F$ * Y – B2
        else:
            return – B2
    if action.type == D:
        if action.success = True:
            return S– (B1 + ΔB1)
        else:
            return – (B1 + ΔB1)
    if action.type == J:
        if action.success = True:
            return $A_J$ * Y – (B2 + ΔB2)
        elif action.detected = True:
            return -P – (B2 + ΔB2)
        else:
            return – (B2 + ΔB2)
    if action.type == R:
        return -B1
    if action.type == S:
        return 0
```

[Pseudo Code 1: Simplified version of Figure 1]

2.3 Spatial Structured Game

For each node in WSN, a pairwise simultaneous game for each neighbor pair can be constructed. Since another neighbor may affect payoffs for a neighbor pair, additional rules are introduced. These rules are deducted by WSN network topology considering with 3 nodes (denoted as n1, n2, n3 where n2 and n3 are neighbors of n1).[Figure 2] shows network topology in detail which is used to generate the rules.

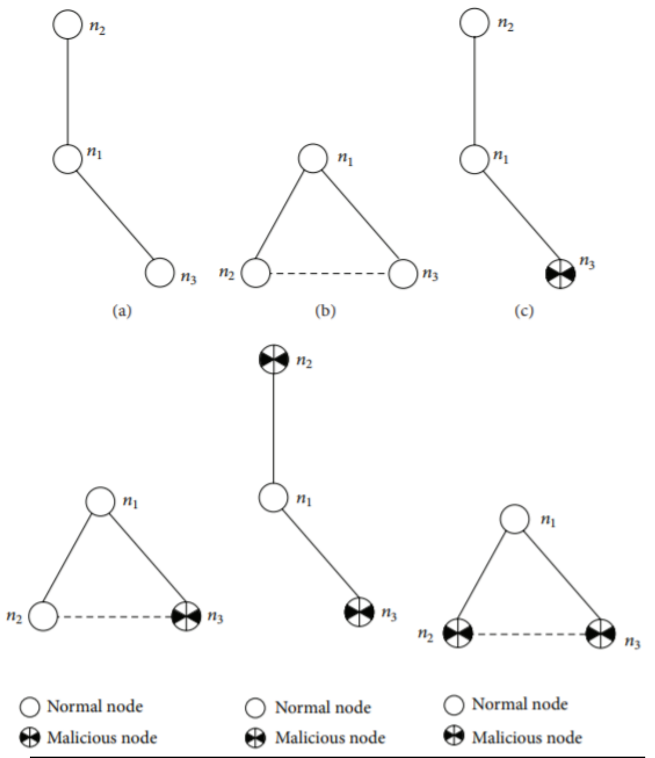
The following [Table 2] shows denotations for the spatial structed game.

Denotation	Meaning
*	Any action
-X	Any action except X
$U_{ni,nj}$	Payoff function for n_i in the subgame between n_i and n_j

[Table 2: Denotations for the spatial structed game (Mao, Zhu, & Wei, p.3, 2013)]

As it can be observed from [Figure 2], there are subgames between n1-n2 and n1-n3 at the same time. Therefore, some of the costs shouldn't be calculated more than once.

[Figure 3.a-b-c] shows exception rules for network topologies



[Figure 2: WSN network topology (Mao, Zhu, & Wei, p.3, 2013)]]

(a, b)-(c,d)-(e,f). For (a, b) both neighbors of n1 are normal, for (c, d) one of neighbors of n1 are malicious, for (a, b) both neighbors of n1 are malicious.

If $u_{n_1,n_2}(F,R) = \alpha_F\gamma - \beta_2$, then $u_{n_1,n_3}(F,*) = 0$, and vice versa (Rule 1).

If $u_{n_1,n_2}(F,*) = -\beta_2$, then $u_{n_1,n_3}(F,-R) = 0$ and $u_{n_1,n_3}(F,R) = \alpha_F\gamma$, and vice versa (Rule 2).

If $u_{n_1,n_2}(D,*) = -(\beta_1 + \Delta\beta_1)$ or $u_{n_1,n_2}(D,*) = -\beta_1$, then $u_{n_1,n_3}(D,*) = 0$, and vice versa (Rule 3).

If $u_{n_1,n_2}(R,*) = -\beta_1$, then $u_{n_1,n_3}(R,-F) = 0$, and vice versa (Rule 4).

[Figure 3.a: Rules for network topology a and b (Mao, Zhu, & Wei, p.4, 2013)]

If $u_{n_1,n_2}(D,*) = -(\beta_1 + \Delta\beta_1)$ or $u_{n_1,n_2}(D,*) = -\beta_1$, then $u_{n_1,n_3}(D,*) = 0$, and vice versa (Rule 5).

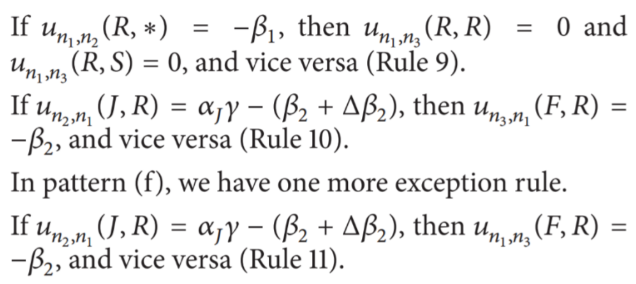
If $u_{n_1,n_2}(R,*) = -\beta_1$, then $u_{n_1,n_3}(R,-F) = 0$, and vice versa (Rule 6).

If $u_{n_3,n_1}(J,R) = \alpha_J\gamma - (\beta_2 + \Delta\beta_2)$, then $u_{n_2,n_1}(F,R) = -\beta_2$ (Rule 7).

In pattern (d), we have one more exception rule.

If $u_{n_3,n_1}(J,R) = \alpha_J\gamma - (\beta_2 + \Delta\beta_2)$, then $u_{n_1,n_2}(F,R) = -\beta_2$ (Rule 8).

[Figure 3.b: Rules for network topology c and d (Mao, Zhu, & Wei, p.4, 2013)]


If $u_{n_1,n_2}(R,*) = -\beta_1$, then $u_{n_1,n_3}(R,R) = 0$ and $u_{n_1,n_3}(R,S) = 0$, and vice versa (Rule 9).

If $u_{n_2,n_1}(J,R) = \alpha_J\gamma - (\beta_2 + \Delta\beta_2)$, then $u_{n_3,n_1}(F,R) = -\beta_2$, and vice versa (Rule 10).

In pattern (f), we have one more exception rule.

If $u_{n_2,n_1}(J,R) = \alpha_J\gamma - (\beta_2 + \Delta\beta_2)$, then $u_{n_1,n_3}(F,R) = -\beta_2$, and vice versa (Rule 11).

[Figure 3.c: Rules for network topology e and f (Mao, Zhu, & Wei, p.4, 2013)]

The first two rules stated in [Figure 3.a] are also included in the other two network topologies stated in [Figure 3.a-b].

As it can be observed from [Figure 3.a-b-c], since all nodes take an action simultaneously in each turn, no node should pay more than once for an action. Payoff values for nodes and their corresponding neighbors are recalculated according to the defined rules.

3.IMPLEMENTATION

In this section, implementation details are explained. Additionally, there are several additional assumptions/differences in the implementation compared to the original work. These are also explained in this section.

```
//WSN is a simple-undirected-unweighted graph

// Node is any vertex on WSN

// PayOffs is a dictionary that keeps track of pay-offs for each neighbor pair on WSN. This is the same thing denoted as $U_{ni,nj}$ at [Table 2].

def pairwiseGamesForSingleNode(WSN, node, payOffs):
    for neighbor in node.neighbors:
        pairwiseGameForTwoNodes(node, neighbor, payOffs)
        // Update payOffs dictionary according to [Figure 1]
    for neigbor1, neigbor2 in node.neighbors
        if neigbour1 != neighbor2:
            applyRules(node, neigbour1, neighbor2, payOffs)
            // Update payOffs dictionary according to [Figure 3]
    return payOffForNode(node, payOffs)
    // Calculate total pay-off for node equals to $\sum_{n_2}^{R} u_{n,n2}$ where R are neighbors of n.
```

[Pseudo Code 2:Game theoretical algorithm for a single node (Mao, Zhu, & Wei, p.5, 2013)]

[Pseudo Code 2] provides a game-theoretical algorithm to combine pairwise simultaneous games in [Figure 1] and exception rules in [Figure 3].

[Pseudo Code 2] only calculates the payoff for a single node. Therefore, to calculate all payoffs [Pseudo Code 3] is used. To calculate payoffs for all nodes in WSN, it is preferred to traverse WSN with Breadth-First Search (BFS) and call the function in [Pseudo Code 2] for each traversed node.

```
// Output is dictionary of total payoffs for each node, defined as P

def getAllPayOffs(WSN, node, payOffs):
    assignInitialActions(WSN)
    // Randomly assigns an initial action to each node in WSN
    for each n in BFS over WSN:
        P[node] = pairwiseGamesForSingleNode(WSN, n, payOffs)
    return P
```

[Pseudo Code 3:Game theoretical algorithm for WSN (Mao, Zhu, & Wei, p.6, 2013)]

[Pseudo Code 3] only iterates once for a WSN. It is impossible to get equilibrium by examining only one iteration. Therefore, multiple iterations are needed. [Pseudo Code 4] iterates the algorithm in [Pseudo Code 3] k times, and averages payoffs for each action-node pair. After averaging these actions for each node, calculate the equilibrium action for each node.

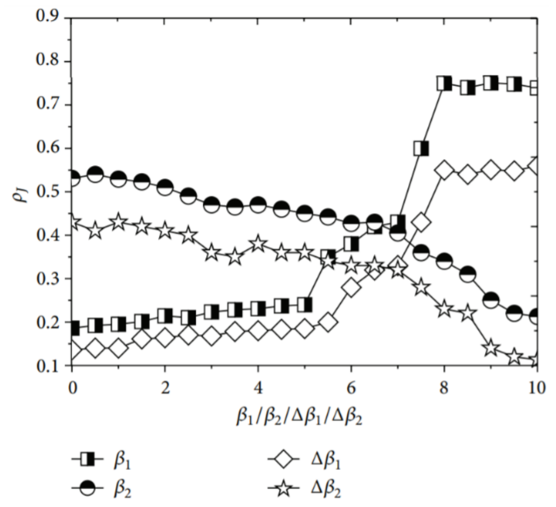
In [Figure 1] each node's payoff is -B1 when the action is Receive, on the other hand, each node's payoff is 0 when the action is Sleep. For this reason, an intelligent node would choose Sleep over Receive since it always gets a higher payoff. This is not specified in the paper, however, in this report, it is assumed that with %25 probability each node selects Sleep action to save battery. Equilibrium is calculated only considering the actions; Receive, Forward, Detect/Jam by looking at which action has a higher average payoff after k iterations. Therefore, the equilibrium state for a node is calculated with %75 probability.

```
// Output is dictionary of total payoffs for each node, defined as P

// Current payoff dictionary for WSN for the i$^{th}$ iteration, defined as payoff

// Average payoff dictionary for each node including i$^{th}$ iteration, defined as averagePayOffs

// A dictionary that keeps calculated equilibrium actions for each node, defined as equilibriumActions

def getEquilibriumActions(WSN, k):
    for i in [1,…..,k]:
        payoff = pairwiseGamesForSingleNode(WSN, n, payOffs)
        updateAveragePayOffs(averagePayOffs, payoff)
    equilibriumActions = getEquilibriumActions(averagePayOffs)
    return equilibriumActions
```

[Pseudo Code 4:Iterative Algorithm over Pseudo Code 3]

4.SIMULATION

There are significant differences in the simulation of [Pseudo Code 4] compared to provided simulation in the paper due to computational/time constraints. It results in some differences between replicated-provided simulations' results which are explained in [section 5].



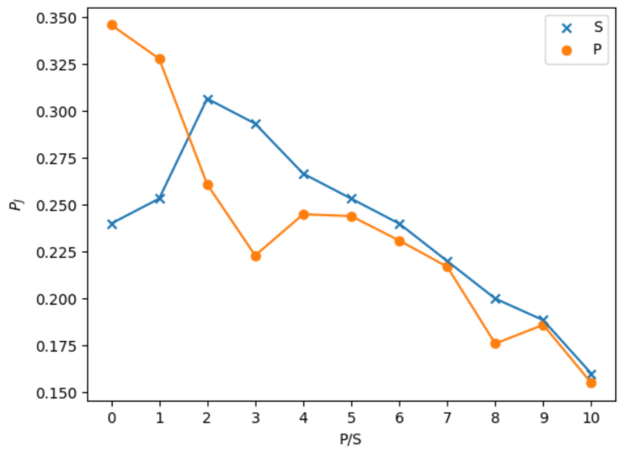
[Figure 4.a: Simulation results from the paper (Mao, Zhu, & Wei, p.6, 2013)]

In the simulation, the effect of B1, B2, ⊿B1, ⊿B2, P, S on P_J (Fraction of malicious nodes that select jamming action) are examined by changing the value of one variable while keeping others' values the same.

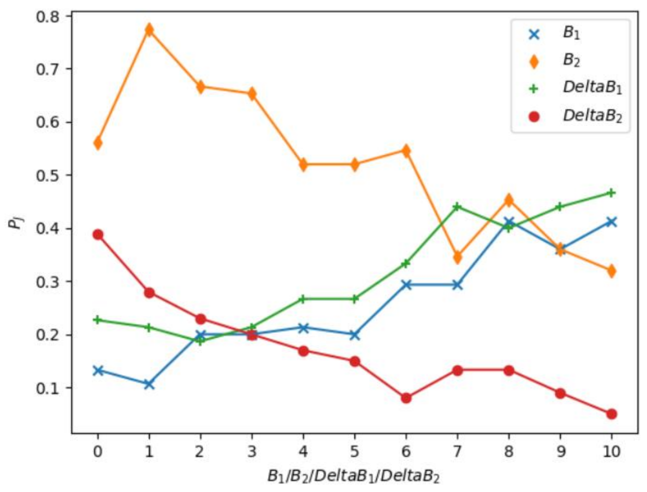
Detailed variables' value-related information is explained in [Table 3].

Dynamic variables can take values of [0,1,….,10]. Their default value is 1.
Static Variables; Y=1, A_F = 0.75 and A_J = 1
Average Number of Neighbors: 8
Number of Nodes in a WSN: 1000
Initial Probability for 4 Actions: %25
Number of Different WSNs used: 1000
Number of Malicious Nodes in a WSN: 100
P_J = Average ((# of nodes decide to jam) / (# of malicious nodes) for each WSN in the simulation)
*Number of Iterations (k): 250 (In this report)
* Originally 3000 generations after 10.000 transient generations *Due to time/computation power constraints simulation results obtained only after 250 iterations for each dynamic variable. *Designed algorithm for this project is open further running-time improvements. * This report does not include any joint affect analysis of dynamic variables due to the reasons specified above.

[Table 3: Detailed information about simulation]



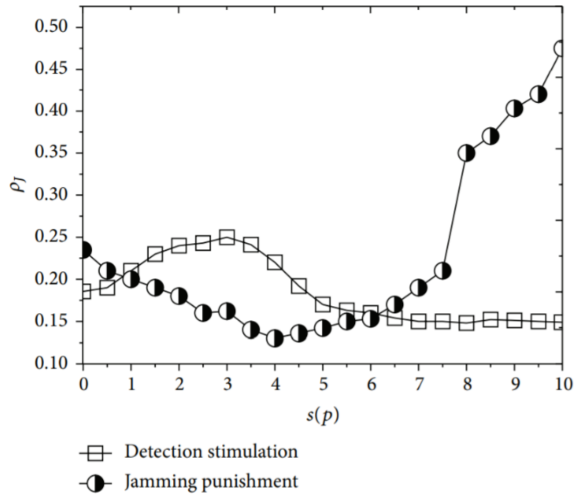
[Figure 5.b: Simulation results for replicated algorithm]



[Figure 4.b: Simulation results for replicated algorithm]

5.SIMULATION RESULTS

In [Figure 4], both replicated and provided simulation results for dynamic variables B1, B2, ⊿B1 and ⊿B2 by the paper [1] are provided. Also, in [Figure 5], simulation results are provided similarly for dynamic variables P and S.



[Figure 5.a: Simulation results from the paper (Mao, Zhu, & Wei, p.7, 2013)]

5.1 [Figure 4] Comparisons

In the following table, similarities and differences in the simulation results in [Figure 4] are compared. Examined variables in this section are B1, B2, ΔB1 and ΔB2.

[Figure 4.a]	[Figure 4.b]
All the scatter plots nearly are *bitonic sequences	Too many fluctuations in the scatter plots
When B1 increases, P_J tends to increase. (especially from 5 to 8)	Similarly, When B1 increases, P_J tends to increase. (There are also some decreases, not the case in figure-a)
When B2 increases, P_J tends to decrease. (After 7, decrease is fastened)	Similarly, when B2 increases, P_J tends to decrease. (There are also some increases, not the case in figure-a)
When ΔB1 increases, P_J tends to increase. (especially from 6 to 8)	Similarly, when ΔB1 increases, P_J tends to increase.
When ΔB2 increases, P_J tends to decrease. (After 7, decrease is fastened)	Similarly, when B2 increases, P_J tends to decrease.

*sequence of numbers which is first strictly increasing then after a point strictly decreasing vice versa.

[Table 4: Simulation interpretations for [Figure 4]]

Although replicated algorithm produces fluctuating simulation results (probably due to the small number of iterations (k)), both results suggest that when B1(Cost to receive a packet/detect jamming) and ΔB1(Additional cost to detect jamming) increases P_J (Fraction of malicious nodes that select jamming action) tends to increase and vice versa.

On the other hand, both results suggest that when B2 (Cost to forward a packet/jam) and ΔB2(Additional cost to jam) increases P_J (Fraction of malicious nodes that select jamming action) tends to decrease and vice versa.

5.2 [Figure 5] Comparisons

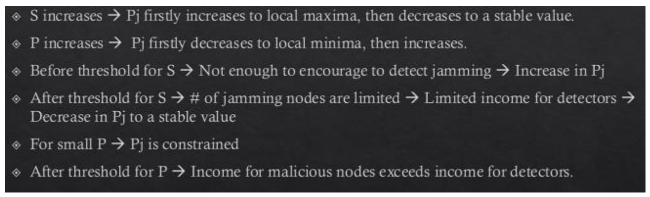
In the following four tables, similarities and differences in the simulation results in [Figure 5] are compared. Examined variables in this section are P and S.

[Figure 5.a]	[Figure 5.b]
All the scatter plots nearly are bitonic sequences	Only the scatter plot for S is nearly a bitonic sequence, whereas the scatter plot for S is flactuating
When S increases from 0 to 3, P_J tends to increase. After 3, P_J tends to decrease.	When S increases from 0 to 2, P_J tends to increase. After 2, P_J tends to decrease.
When P increases from 0 to 4, P_J tends to decrease. After 4, P_J tends to increase.	Differently, when P increases, P_J always tends to decrease. (Although there are some fluctuations, the pattern is to decrease)

[Table 5: Simulation interpretations for [Figure 5]]

Similarly, both results suggest that when S (Stimulus to a successful jamming detection of the normal node) increases, P_J tends to increase, and after some point, P_J tends to decrease.

Differently, the paper [1] suggests that when P (Punishment to detected jamming) increases, P_J tends to decrease and after some point, P_J tends to increase. On the other hand, the simulation result of replicated algorithm suggests that when P increases, P_J always tends to decrease (probably due to the small number of iterations (k)).



- S increases → Pj firstly increases to local maxima, then decreases to a stable value.
- P increases → Pj firstly decreases to local minima, then increases.
- Before threshold for S → Not enough to encourage to detect jamming → Increase in Pj
- After threshold for S → # of jamming nodes are limited → Limited income for detectors → Decrease in Pj to a stable value
- For small P → Pj is constrained
- After threshold for P → Income for malicious nodes exceeds income for detectors.

[Table 6: Further simulation interpretations for [Figure 5.a]]

6.SUMMARY

Although there are differences between the two simulation results (especially in S and some fluctuations in scatter plots), replicated algorithm mostly captures what the paper [1] suggests in its simulation results.

REFERENCES

[1] Mao, Y., Zhu, P., &amp; Wei, G. (2013). A Game Theoretic Model for Wireless Sensor Networks with Hidden-Action Attacks. International Journal of Distributed Sensor Networks, 9(8), 836056. doi:10.1155/2013/836056