Decision System Modelling for "Saboteur" using Heuristics and Means-Ends Analysis

Dion Krisnadi, Samuel Lukas, Laurentius Dominick Logan, Nadya Felim Bachtiar, Livia Andriana Lohanda
Informatics department, Faculty of Computer Science, Universitas Pelita Harapan
UPH Tower, Lippo Karawaci, Tangerang, Indonesia, 15811
{Dion.krisnadi, Samuel.lukas, Livia.Lohanda}@uph.edu, {LL9920, NB9602}@student.uph.edu

ABSTRACT
Saboteur is a card game played by several players. Players are divided into goldminers and saboteurs who prevents goldminers from finding gold. Each player knows his/her role, but he/she does not know who has the same role. Gold searching process starts by making a path of seven cards horizontally from a single point of reference to the gold card's position. Saboteur is classified as an incomplete information card game with mechanisms that requires complex decision making. This paper proposes an approach to model a decision system for Saboteur using Heuristics and Means-End Analysis. Implementation was purely done in Java language. To test it, we performed benchmarking for the proposed system against previous intelligent agent models. The result is much better.

CCS Concepts
• Computing methodologies → Artificial intelligence → Search methodologies → Game tree search

Keywords
Saboteur, game modelling, artificial intelligence, heuristics, means-ends analysis, intelligent agent.

1. INTRODUCTION
Saboteur is a card game designed by Frédéric Moyersoen, and it was published in 2004. The game was created for three to ten players. BoardGameGeek has classified this game as a Bluffing game with notable mechanisms, such as Hand Management, Partnerships, and Route/Network Building [1]. Further information regarding the game components, set up, and rules can be seen at [2].

Building an Artificial Intelligence for this game has proven to be quite a challenge. Saboteur is an incomplete information game where the player has no knowledge regarding other players' hand and role. This game requires cooperation even without knowing others' role, which makes the players resort to bluffing. Different approaches such as Means-Ends Analysis [3], Monte Carlo Tree Search [4] and Kripke model [5] have been used to solve the aforementioned issues. In this paper, we propose a new approach to the matter by using Heuristics and Means-Ends Analysis.

2. GAME MODELLING

2.1 Card
The game cards consist of four main types, path cards (π), map cards (μ), rockfall cards (ρ), and action cards ($\emptyset$). There are 67 playable cards in total. In addition, there are unplayable cards, which consists of the start card and three goal cards as shown in Fig 1. The deck of each playable card is $\chi = \{\langle\pi, 40\rangle, \langle\mu, 6\rangle, \langle\rho, 3\rangle, \langle\phi, 18\rangle\}$.

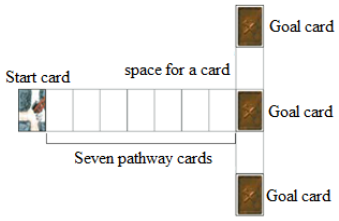

Figure 1. The arena for a game from Start to Goal card

The set π consists of 31 pathway cards and 9 dead-end cards. Each card has four sides. A tuple $\pi = \langle top, right, bottom, left\rangle$ where each side is an element of $\{path, deadend, wall\}$. Fig 2 shows three path cards and their representation in graph. A path card π can be rotated with the function:

$$\mathbf{rotate}(\pi) = \langle bottom, left, top, right\rangle \#(1)$$

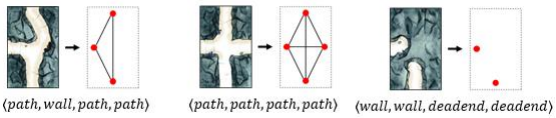

Figure 2. Path cards labeling and their graph

Map card (μ) is used by a player to see one of the goal cards personally and identify whether that goal card is a gold or rock card. There is only one type of Map card. Like map cards, there is only one type of rockfall cards (ρ). It is used to destroy a pathway or dead-end card on the board. Saboteur player typically uses this card to disconnect the path from the start card to gold card. On the other hand, goldminer uses the card to destroy a dead-end card so that a connected path might be made.

Action card (ϕ) has two actions, block or repair, and is denoted as

$$\phi = \{\langle block, a\rangle, \langle repair, b\rangle | a, b \in \{empty, c, l, p, cl, cp, lp\}\}$$

with c for cart, l for lantern, and p for pickaxe. There are three cards for each c,l and p block action cards, while only two for each type of the repair action card and one for each repair action card with cl, cp and lp. Figure 3 shows map, rockfall and action cards. Start and goal cards are unplayable cards and are placed when the game first starts. There is only one start card in the whole game labelled as $\zeta = \{\langle start, 1\rangle\}$, but there are two goal cards in the game. Goal cards are represented as $\gamma = \{\langle gold, 1\rangle, \langle rock, 2\rangle\}$, indicating there is only one gold goal and two rock goals.

ICAAI 2019, October 26-28, 2019, Istanbul, Turkey.
 ISBN 978-1-4503-7253-4
DOI: 10.1145/3369114.3369162

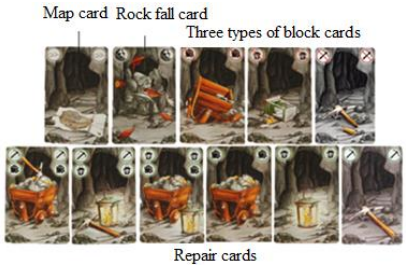

Figure 3. map, rockfall and action cards

2.2 Cell and Board

A cell is a single space in the game board, and is labeled as tuple $\beta_{x,y}(\alpha) = \langle top, right, bottom, left, \alpha \rangle$, where:

- $\beta_{x,y}(\alpha)$ is the cell at (x, y) filled with card α, $\alpha \in \{\pi, \varsigma, \gamma\}$.
- $top, right, bottom,$ and $left$ are side value of the cell. If the cell is empty, then all sides are empty, otherwise all sides are equal to each corresponding side of α.
- $x = \{0,1, \dots, 8\}$ and $y = \{0,1, \dots, 4\}$.
- $\beta_{0,0}$ is the cell at the top-left corner of the board and $\beta_{8,4}$ is the one at the bottom-right corner of the board
- Two adjacent cells β and β' are considered valid iff their touching sides s and s' fulfill (2):

$$\textbf{valid}(s, s') \coloneqq \begin{array}{l} (s = s') \\ \vee\, empty \in \{s, s'\} \\ \vee\, (\{deadend, path\} = \{s, s'\}). \end{array} \quad \#(2)$$

A function **reachable**(x, y) is defined for the board B. It tests if $B_{x,y}$ is reachable from the starting position B_{02}. For this function, the board is first interpreted as a graph. Fig 4 illustrates how each cell and the board are represented. Each cell is represented with a subgraph of one to four vertices with each vertex represents its side value. When at least one of two adjacent cells contains a card, their touching sides relate to an edge.

Reachable(x, y) is true iff $\beta_{x,y}$ is on the board, and that there is at least a path connecting any vertex from $\beta_{x,y}$ to $\beta_{0,2}$. For the case in Fig 4, **reachable**(x, y) is true for (0,2), (0,1), (0,3), (1,2), (1,1), (2,2), (2,1), (2,3), (2,4), (3,3), (4,3), (4,4), (3,4), (5,4), (6,4), (7,4), (6,3), (7,3), (8,3), (8,2), (7,2), and (8,4).

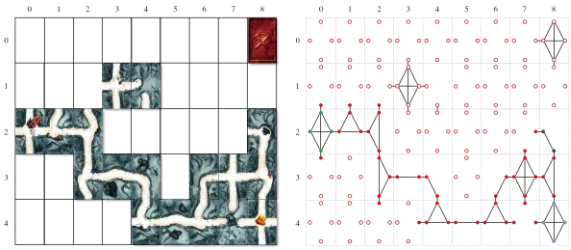

Figure. 4 Graph representation of a game

2.3 Player

A player's role $R = \{miner, saboteur\}$, can be either a goldminer or a saboteur. A player p is defined as $p = \langle r_p, \phi_p, \gamma_p, \chi_p \rangle$:

- r_p is the role of the player, $r_p \in R$.
- ϕ_p is p's blocked action card given by other players.
- γ_p represents goals known to player p. γ_p is written as a set of tuples $\beta_{x,y}(a)$, $a \in \{gold, rock\}$.
- χ_p is the current hand of player p, $\chi_p \subset \chi$.

2.4 Game Start

The game starts by determining each player's role, then randomly placing the goal cards in the designated area and distributing playable cards. The number of saboteurs is defined in (3), while the number of cards for each player is in (4).

$$|r = saboteur| = \begin{cases} 1, & n = 4, \\ 2, & 4 < n \leq 6, \\ 3, & 6 < n \leq 10. \end{cases} \quad (3)$$

$$\forall_{p_n} \left(|\chi_p| = \begin{cases} 6, & n \leq 5, \\ 5, & 5 < n \leq 7, \\ 4, & n > 7. \end{cases} \right). \quad (4)$$

Initial conditions for the board and players are

- $B = \{\beta_{0,2}(start), \beta_{8,0}(\gamma_{8,0}), \beta_{8,2}(\gamma_{8,2}), \beta_{8,4}(\gamma_{8,4})\}$,
- $p_n = \langle r_p, \langle block, empty \rangle, \langle \beta_{8,0}(), \beta_{8,2}(), \beta_{8,4}() \rangle, \chi_p \rangle$,
- $\chi = \chi - \cup_{p=1}^{n} \chi_p$.

2.5 Moves

Six types of move can be done by players. They can form a path by $play(\pi, x, y)$, open a goal card by $play(\mu, x, y)$, destroy a path by $play(\rho, x, y)$, block or repair a player by $play(\phi, p')$, and discard a card in the hand by $play(x)$. The preconditions for each move are described in Table 1.

2.6 Game End

The game ends with either goldminers or saboteurs as the winner. Goldminers win if one out of these three conditions hold, otherwise saboteurs win:

- reachable$(8,0) \wedge \beta_{8,0} = \langle gold \rangle$,
- reachable$(8,2) \wedge \beta_{8,2} = \langle gold \rangle$,
- reachable$(8,4) \wedge \beta_{8,4} = \langle gold \rangle$.

3. DECISION SYSTEM

3.1 Movement Decision

When the player is prompted to move, it calculates the heuristic value for each card on its hand, χ_p. It is represented as $Move = \{\langle c, heuristic, (x, y), p' \rangle \mid c \in \chi_p\}$. For every invalid move $c \in \pi_p$, its heuristic is set to zero. The best move is determined in Algo. where

- d_δ is the minimum heuristic value for a non-discard move,
- c is the played card,
- $heuristic$ is the heuristic value of the card,
- (x, y) is the position of the card in board, and
- p' is the player associated with the played card.

$max \leftarrow \arg\max_{c \in Move} Move_{heuristic\ of\ card\ c}$
if
$max > d_\delta \wedge precondition\ is\ true\ for\ playing\ card\ c$
  play$(\langle c, heuristic, (x, y), p' \rangle)$
else:
  $discard\ card$

Algo. 1 General decision making for player movement.

When a map card is in possession, it is better to open either the top or bottom goal card. This way, the formed path can more directed, and split paths can be avoided. If top goal is a rock card, a goldminer might want to direct the path away from the top. On the other hand, goldminer might direct the path to the top. Therefore, playing this card can change the player's knowledge on gold's position. The best move for a map card a is determined in Algo 2, where d_μ is the constant heuristic value for a map card.

Table 1. Preconditions for each move

ove Operator	Precondition	Result
$play(x_p, x, y), x_p \in \pi_p$ Player p plays path card x_p from his hand and draws a new card x_n from the deck (χ) randomly	$\lvert\emptyset_p\rvert = 0$	The player has not been blocked
	$\beta_{x,y}(empty)$	$\beta_{x,y}(x_p), \quad \chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$
	reachable(x, y) is true	$\beta_{x,y}(x_p), \quad \chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$
	$if\ y \geq 1 \rightarrow$ **valid**$(\boldsymbol{\beta_{x,y}}, \boldsymbol{\beta_{x,y-1}})$	$\beta_{x,y}(x_p), \quad \chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$
	$y < 3 \rightarrow$ **valid**$(\boldsymbol{\beta_{x,y}}, \boldsymbol{\beta_{x,y+1}})$	$\beta_{x,y}(x_p), \quad \chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$
	$if\ x < 7 \rightarrow$ **valid**$(\boldsymbol{\beta_{x,y}}, \boldsymbol{\beta_{x+1,y}})$	$\beta_{x,y}(x_p), \quad \chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$
	$x \geq 1 \rightarrow$ **valid**$(\boldsymbol{\beta_{x,y}}, \boldsymbol{\beta_{x-1,y}})$	$\beta_{x,y}(x_p), \quad \chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$
$play(x_p, x, y), x_p \in \mu_p$ Player p plays map card x_p from his hand to goal card at (x, y) and draws a new card x_n from the deck (χ) randomly		$\chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$ $p_n^k = \langle r, \phi_p, \gamma_p = \gamma_p \cup \gamma_{x,y}, \chi_p \rangle$
$play(x_p, x, y), x_p \in \rho_p$ Player p plays rockfall card x_p from his hand at (x, y) and draws a new card x_n from the deck (χ) randomly	$\lvert\emptyset_p\rvert = 0$ $\beta_{x,y}(empty)$ is false	$\chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$ $\beta_{x,y}(empty)$
$play(x_p, p'), x_p \in \rho_p$ Player p plays action block card x_p from his hand to player p' and draws a new card x_n from the deck (χ) randomly	$p \neq p'$ $\phi_{p'} \cap \{x_p\} = \emptyset$	The player cannot block himself $\chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$ $\phi_{p'} = \phi_{p'} \cup \{x_p\}$
$play(x_p, p'), x_p \in \rho_p$ Player p plays action repair card x_p from his hand to player p' and draws a new card x_n from the deck (χ) randomly	$\phi_{p'} \cap \{x_p\} = \{x_p\}$	$\chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$ $\phi_{p'} = \phi_{p'} - \{x_p\}$
$play(x_p), x_p \in \chi_p$ Player p discards card x_p from his hand and draws a new card x_n from the deck card (χ) randomly		$\chi_p = (\chi_p - \{x_p\}) \cup \{x_n\}, x_n \in \chi$

function generate_best_map(a):
  if $\beta_{8,0} = \emptyset$
    $Pos \leftarrow (8,0)$
  else
    if $\beta_{8,2} = \emptyset$
      $Pos \leftarrow (8,2)$
    else
      $bestPos \leftarrow (8,4)$
  $heu = d_\mu$
    $Return\ \langle a, heu, (Pos_x, Pos_y), "" \rangle$

Algo. 2 General decision making for a map card.

For path and rockfall cards, firstly determine the value of each cell on the board based on the player's role and knowledge. This value is denoted as $v(x, y)$, and is defined as equation (5)

$$v(x, y) = v(x) \times v(y), \#(5)$$

where $v(x)$ is the value of the x, and $v(y)$ is the value of the y.

The value of $v(x)$ is defined in (6), whereas $v(y)$ is in (7) for miner and in (8) for saboteur. Prior to calculate $v(y)$, the player's knowledge about goal cards is scored as in Table 2. It is denoted as $score(\gamma_p)$, where γ_p is the player's knowledge of the goal cards.

$$v(x) = \begin{cases} 0.1(1 + x), & r_p = miner \\ 0.1(9 - x), & r_p = saboteur \end{cases} \#(6)$$

Table 2. Scoring player's knowledge of the goal cards

Condition (γ_p)	$score(\gamma_p)$
$\gamma_p = \langle \beta_{8,0}(), \beta_{8,2}(), \beta_{8,4}() \rangle$	-1
$\gamma_p = \langle \beta_{8,0}(rock), \beta_{8,2}(), \beta_{8,4}() \rangle$	0
$\gamma_p = \langle \beta_{8,1}(gold), \beta_{8,2}(rock), \beta_{8,4}(rock) \rangle$	1
$\gamma_p = \langle \beta_{8,1}(rock), \beta_{8,2}(gold), \beta_{8,4}(rock) \rangle$	2
$\gamma_p = \langle \beta_{8,1}(rock), \beta_{8,2}(rock), \beta_{8,4}(gold) \rangle$	3

$$v(y) = \begin{cases} 1, & score(\gamma_p) = -1 \\ \min(0.5y, 1), & score(\gamma_p) = 0 \\ 1 - 0.25y, & score(\gamma_p) = 1 \\ 1 - |1 - 0.5y|, & score(\gamma_p) = 2 \\ 0.25y, & score(\gamma_p) = 3 \end{cases} \#(7)$$

Fig 5 illustrates the value of each cell if the player is a miner with $score(\gamma_p) = 3$. The closer the cell to the gold goal the bigger is the value of $v(y)$.

$$v(y) = \begin{cases} 1, & score(\gamma_p) = -1 \\ \max(1 - 0.5y, 0), & score(\gamma_p) = 0 \\ \frac{1}{16}y^2, & score(\gamma_p) = 1 \\ |1 - 0.5y|, & score(\gamma_p) = 2 \\ \frac{1}{16}(y - 4)^2, & score(\gamma_p) = 3 \end{cases} \#(8)$$

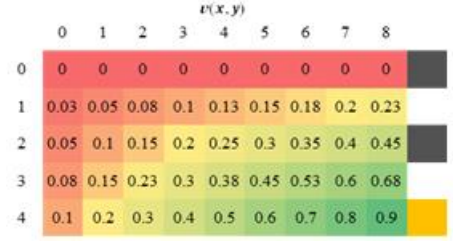

Figure. 5 Value of $v(x, y)$ for a miner with $score(\gamma_p) = 3$

Secondly, let φ be defined in (9) as a collection of reachable empty cell, while $v(B)$ in (10) as the value of the board.

$$\varphi = \{(x, y) | \textbf{reachable}(x, y) \wedge \beta_{xy}()\} \quad (9)$$

$$v(B) = \max_{(x,y) \in \varphi} v(x, y) \quad (10)$$

Thirdly, collect all possible player's path cards in C where $C = \{\langle a, (a_x, a_y)\rangle | a \in \pi_p, (a_x, a_y) \in reachable$ and $\langle a, (a_x, a_y)\rangle \in C$ if $\beta_b(a) \vee \beta_b(rotate(a))$ is true. Finally, finding the best move for a path card is done using Algo 3, where d_π is the heuristic multiplier for a path card. The algorithm for placing a rockfall card is the same with placing a patch card, with φ is defined as φ' in (11) and d_π as d_ρ.

$$\varphi' = \{(x, y) | \mathbf{reachable}(x, y) \wedge \beta_{xy} \neq \emptyset\} \quad (11)$$

function generate_best_path$(a, (a_x, a_y))$:
  $heu \leftarrow -\infty$
  $Pos \leftarrow (-1, -1)$
  $h \leftarrow 0$
  foreach $\langle a, (a_x, a_y)\rangle \in C$
    $B' \leftarrow$ simulated placement of $\langle a, b\rangle$ on B
    $h \leftarrow v(B') - v(B)$
    if $h > heu$
      $heu \leftarrow h$
      $Pos \leftarrow (a_x, a_y)$
  return $\langle a, heu * d_\pi, (Pos_x, Pos_y), ""\rangle$

Algo.3 Best move for a path and rockfall card.

For doing action card, then a player can decide whether to repair or block other players based on their affiliation. If the targeted player is considered as an enemy, then the target is more likely to be blocked. Otherwise, the target is more likely to be repaired.

A block move will be prioritized to block unblocked enemies. Let P_a be the set of allies according to player p, including p itself, while P_{-a} be the set of p 's enemies. The steps to determine the best move for a block card $\phi_i \epsilon \phi_p$ is defined in 错误!未找到引用源。, where d_ϕ is the multiplier for a block move.

A repair move will be prioritized to an ally with the most blocked tools and to repair the current player (repairing itself). The steps to determine the best move for a repair card is defined in A 错误!未找到引用源。.

Discard card is played when all moves are considered bad. That card is the worst card that has $heuristic = 0$. This has been explained in Algo. . Discarded card is selected randomly from pathway card for saboteur and dead-end card for miner.

3.2 Trust Decision

Each move played by the other players will affect how the current player trusts them as an ally. This is modelled using an array of trust values $T_p = \{\theta_i | i = 1,2, \dots, n, i \neq p\}$. The trust value of player p towards another player q can be denoted as θ_q, and is initialized as a constant d_0. For every move, a player should update the value of θ_q.

function generate_best_block(ϕ_i, P_{-a}):
  $P'_{-a} \leftarrow \{p' \mid \phi_{p'} \cap \{\phi_i\} = \emptyset, p \neq p', p' \in P_{-a}\}$
  $heu \leftarrow -\infty$
  $Player \leftarrow \emptyset$
  foreach p' in P'_{-a} :
    $h \leftarrow \frac{1}{|\phi_{p'}|}$
    if $h > heu$
      $heu \leftarrow h$
      $Player \leftarrow p'$
  return $\langle \phi_i, heu * d_\phi, (-1, -1), Player\rangle$

Algo.4 Best move for a block card.

function generate_best_repair(ϕ_i, P_a):
  $P'_a \leftarrow \{p' \mid \phi_{p'} \cap \{\phi_i\} \neq \emptyset, p' \in P_a\}$
  $heu \leftarrow -\infty$
  $Player \leftarrow \emptyset$
  foreach p' in P'_a:
    $h \leftarrow \frac{3}{2} d_\phi \cdot |\phi_{p'}| - 1$
    if $p' = p$:
      $h \leftarrow 2h$
    if $h > heu$
      $heu \leftarrow h$
      $player \leftarrow p'$
  return $\langle \phi_i, heu, (-1, -1), Player\rangle$

Algo.5 Best move for a repair card

Let P_a be the set of current player's allies and P_{-a} be the set of current player's enemies. P_{-a} can simply be determined by k lowest values of θ_q, where k equals to the number of saboteurs in the game excluding itself. The next four subsections discuss how each type of move affects current player's trust values

If another player places or destroys a path so that it helps the current player achieve its goal (a miner's goal is to open a gold card, while a saboteur wants to stay away from the gold card), then he/she is more likely to be an ally. Otherwise, he/she is more likely to be an enemy. Let q be the player who played a path or a rockfall card, such that they modified the board from B into B'. The steps to modify the trust value is in 错误!未找到引用源。

function modify_trust_board(B', B, θ_q):
  $\Delta trust \leftarrow v(B') - v(B)$
  $\theta_q \leftarrow \theta_q + \Delta trust$

Algo.6 Trust modifier for another player's path/rockfall move

Trust modification for block moves is adopted from [3]. If another player blocks an ally, then he/she is more likely to be an enemy. On the other hand, if another player blocks an enemy, then he/she is more likely to be an ally. Let q be the player who played the block card ϕ_q to q'. The details are explained in Algo 7., where d_b is the base trust modifier for a block move.

function modify_trust_block$(\phi_q, P_a, \theta_q, q')$:
  if $q' \in P_a$
    $\theta_q \leftarrow \theta_q - d_b$
  Else
    $\theta_q \leftarrow \theta_q + d_b$

Algo.7 Trust modifier from another player's block move.

Trust modification for repair moves is also adopted from [3]. If another player repairs an ally, then that player is more likely to be an ally. Otherwise, if an enemy is repaired, then the player is more likely to be also an enemy. The details are explained in Algo 8.

function modify_trust_repair$(\phi_q, P_a, \theta_q, q')$:
  if $q' \in P_{-a}$:
    $\theta_q \leftarrow \theta_q + 2d_b$
  else
  $\theta_q \leftarrow \theta_q - 2d_b$

Algo.8 Trust modifier from another player's repair move.

Discard moves may indicate two possible scenarios. It can be that the player has no good card to play, or the player is a saboteur, and is trying to waste cards. Either way, this decision system does not consider discard moves as a factor to the trust model.

4. IMPLEMENTATION AND RESULT

Implementation is done in Java for both the game model and the AI representing the decision system. For this implementation, the chosen constants for the proposed decision system are

$$\langle d_\delta, d_\mu, d_\pi, d_\rho, d_\phi, d_0, d_b \rangle = \langle -0.5,3,50,50,0.4,2.5,2.6 \rangle.$$

The constants are not from thin air, but an adoption of the ones proposed in [3]. To measure the performance of the decision system, the game will be run in several tests. First, to compare the proposed system with [3], three experiments are conducted. The results are depicted in Table 3 and Table 4. Secondly, another test using only the proposed decision system will be done 3 × 1000 times repeatedly for different opposing teams. The result can be seen in Table 5.

Table 3. Saboteur player with [3] and miner with the proposed decision system

Test case	Saboteurs win rate	Miners win rate	Win ratio
1 saboteur vs 3 miners	11.0%	89.0%	1 : 8.09
	11.2%	88.8%	1 : 7.93
	9.2%	90.8%	1 : 9.87
1 saboteur vs 4 miners	5.9%	94.1%	1 : 15.95
	5.9%	94.1%	1 : 15.95
	7.0%	93.0%	1 : 13.29
2 saboteurs vs 3 miners	35.4%	64.6%	1 : 1.82
	34.5%	65.5%	1 : 1.90
	34.9%	65.1%	1 : 1.87
2 saboteurs vs 4 miners	25.0%	75.0%	1 : 3.00
	23.6%	76.4%	1 : 3.24
	26.2%	73.8%	1 : 2.82
2 saboteurs vs 5 miners	15.0%	85.0%	1 : 5.67
	14.5%	85.5%	1 : 5.90
	14.9%	85.1%	1 : 5.71
2 saboteurs vs 6 miners	14.6%	85.4%	1 : 5.85
	16.8%	83.2%	1 : 4.95
	15.1%	84.9%	1 : 5.62

Single miner win rate is better than that of saboteur players. Even as a team, miner does better than saboteur players.

Table 4. Saboteur with the proposed decision system and miner with [3]

Test case	Saboteurs win rate	Miners win rate	Win ratio
1 saboteur vs 3 miners	82.4%	17.6%	4.68 : 1
	82.6%	17.4%	4.75 : 1
	82.7%	17.3%	4.78 : 1
1 saboteur vs 4 miners	74.5%	25.5%	2.92 : 1
	73.0%	27.0%	2.70 : 1
	74.1%	25.9%	2.86 : 1
2 saboteurs vs 3 miners	98.2%	1.8%	54.56 : 1
	97.9%	2.1%	46.62 : 1
	98.0%	2.0%	49.00 : 1
2 saboteurs vs 4 miners	95.5%	4.5%	21.22 : 1
	94.1%	5.9%	15.95 : 1
	93.4%	6.6%	14.15 : 1
2 saboteurs vs 5 miners	90.7%	9.3%	9.75 : 1
	91.7%	8.3%	11.05 : 1
	90.3%	9.7%	9.31 : 1
2 saboteurs vs 6 miners	86.4%	13.6%	6.35 : 1
	87.1%	12.9%	6.75 : 1
	89.1%	10.9%	8.17 : 1

The results show an outstanding improvement for both miners and saboteurs. As the number of miners increases, the win rate also increases with the maximum win rate of 94.1%. When playing as a saboteur, the miners are clearly outplayed, with the maximum saboteur win rate of 98.2%. Even when there are far more miners than saboteurs (4:1), the win rate of the saboteurs is at least 73%. Finally, the last benchmark is to put the proposed decision system against itself.

With only one saboteur, the miners clearly outplay the opposing role, with an average win rate of 81.17% and 88.57%. However, when there are two saboteurs in the game, the miners are challenged, especially when there is nearly the same number of saboteurs and miners—the saboteurs are three times more likely to win when there are three saboteurs against two miners. With two saboteurs against four miners, the playing field is evened out, with an average win ratio of 1.19:1 for saboteurs. Adding more miners will result in a playing field that once again is not favorable for the saboteurs.

Table 5. Both saboteur and miner with the proposed system

Test case	Saboteurs win rate	Miners win rate	Win ratio
1 saboteur vs 3 miners	17.6%	82.4%	1 : 4.68
	20.6%	79.4%	1 : 3.85
	18.3%	81.7%	1 : 4.46
1 saboteur vs 4 miners	11.7%	88.3%	1 : 7.55
	11.7%	88.3%	1 : 7.55
	10.9%	89.1%	1 : 8.17
2 saboteurs vs 3 miners	75.6%	24.4%	3.10 : 1
	76.1%	23.9%	3.18 : 1
	77.4%	22.6%	3.42 : 1
2 saboteurs vs 4 miners	52.5%	47.5%	1.11 : 1
	55.0%	45.0%	1.22 : 1
	55.8%	44.2%	1.26 : 1
2 saboteurs vs 5 miners	37.6%	62.4%	1 : 1.66
	38.1%	61.9%	1 : 1.62
	38.3%	61.7%	1 : 1.61
2 saboteurs vs 6 miners	30.0%	70.0%	1 : 2.33
	33.0%	67.0%	1 : 2.03
	31.6%	68.4%	1 : 2.16

5. CONCLUSION

This paper has proposed a way to model Saboteur game and a decision system along with it. Results show that this paper's decision system performs well when playing as either a miner or a saboteur. Constants $d_\delta, d_\mu, d_\pi, d_\rho, d_\phi, d_0, d_b$ heavily influence how the decision system works and may affect how well it performs. Further investigation regarding the given constants should be done for better optimization. This paper also has not discussed how to implement bluffing for the saboteurs, as they are very reckless. Implementing bluffing for saboteurs might be of interest as it may increase their winning chance against miners. Other alternatives for intelligent agent modelling may also be explored to model Saboteur game, such as Monte Carlo tree search or even deep learning algorithms.

6. REFERENCES

[1] BoardGameGeek, "Saboteur," [Online]. Available: https://boardgamegeek.com/boardgame/9220/saboteur. [Accessed 1 May 2019].

[2] UltraBoardGames, "Saboteur Game Rules," [Online]. Available: https://www.ultraboardgames.com/saboteur/game-rules.php. [Accessed 1 May 2019].

[3] Krisnadi and S. Lukas, "Saboteur game modelling using means-ends analysis," in *2016 International Conference on Informatics and Computing (ICIC)*, Mataram, 2016.

[4] Whitehouse, "Monte Carlo Tree Search for games with Hidden Information and Uncertainty," 2014.

[5] M. Gołuński, R. Tomanek and P. Wasiewicz, "Modeling intelligent agent beliefs in a card game scenario," in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2012*, Wilga, 2012.