Perbandingan Performa Algoritma *Minimax* dan *Breadth First Search* Pada Permainan *Tic-Tac-Toe*

http://dx.doi.org/10.28932/jutisi.v4i1.757

Jerry Setiawan[#1], Farhan Agung Famerdi[#2], Daniel Udjulawa[#3], Yohannes[#4]

[#]*Teknik Informatika, STMIK Global Informatika MDP*
Jalan Rajawali No. 14 Palembang

[1]jerrysetiawan@mhs.mdp.ac.id
[2]farhanagung@mhs.mdp.ac.id
[3]daniel@mdp.ac.id
[4]yohannesmasterous@mdp.ac.id

Abstract — **Tic-Tac-Toe is one of the board games that can hone the motor skills of the brain. This game uses 2 pawns, there are X and O. The game started with X's pawn as the player who first turns, the game got win condition if the player or the enemy put the 3 pawns in a diagonal, vertical or horizontal line. While the game got draw if there is no player or enemy who put 3 pawns in a diagonal, vertical or horizontal line. The game's problems are the player should think about the next best step to win and defend with put pawn to block enemy's steps to win. To solve the problems, the game needs some algorithms, there are Minimax algorithm and Breadth First Search algorithm. Minimax algorithm explores node from deepest level and evaluates the scores using minimum or maximum value. Breadth First Search algorithm is an algorithm which explores node widely and compares evaluation scores to the deepest level. In this research, each algorithm is tested to response time and number of nodes needed on a game board with 3×3, 5×5, 7×7, and 9×9 size as much as 16 scenarios. Based on the test results, Breadth First Search algorithm is superior to Minimax on 3×3 board size in terms of response time and the number of nodes required. While the Minimax algorithm is superior to Breadth-First Search on 5×5 and 9×9 board size in terms of response time and the number of nodes required. In the first turn, the algorithm will trace the number of nodes larger than the next step so that the placement of the algorithm for the first turn affects the final result of the node number parameter.**

Keywords— **Minimax, Breadth First Search, Tic-Tac-Toe**

I. PENDAHULUAN

Banyak *game* yang memiliki sisi yang positif, salah satunya yaitu untuk mengasah otak. *Tic-Tac-Toe* adalah salah satu *board game* yang dapat mengasah kemampuan motorik otak. Permainan ini telah populer sejak zaman Romawi kuno sekitar abad pertama sebelum masehi. Permainan ini menggunakan dua bidak yaitu X dan O. Permainan dimulai dengan bidak X sebagai pemulai langkah, permainan dapat dikatakan menang jika pemain atau lawan menyusun 3 bidak secara diagonal, vertikal atau horizontal, sedangkan permainan dikatakan seri jika pemain dan lawan tidak menyusun bidak secara diagonal, vertikal atau horizontal. Permasalahan pada permainan ini adalah pemain harus memikirkan langkah terbaik untuk menang dan bertahan dengan menempatkan bidak untuk menghalangi langkah lawan untuk menang

Permainan tradisional seperti congklak, catur, othello telah menggunakan *artificial intellegence* sebagai lawan main yang membuat permainan lebih menantang. Algoritma *Minimax* merupakan algoritma menentukan langkah terbaik. Algoritma ini menggunakan dasar pencarian *Depth First Search* dengan mengeksekusi *depth* terdalam, tujuannya agar mendapat kemungkinan terbesar untuk menang. Pada algoritma *Minimax* terdapat dua jenis *node*, yaitu *node max* dan *node min*. *Node max* akan memilih nilai tertinggi dan *node min* akan memilih nilai terendah. Dalam kasus permainan catur dimana hanya terdapat kemungkinan untuk bergerak bagi 8 pion dan 2 kuda, akan terdapat sebanyak 20 kemungkinan. Hal ini berarti pada simpul awal akan didapatkan 20 sub-simpul sebagai langkah penyelesaian pada *level* pertama. Jika komputer melakukan hal yang sama untuk menganalisis langkah selanjutnya maka akan dilakukan 20 pembangkitan dimana pada setiap pembangkitan akan dilakukan puluhan pembangkitan lain untuk *level* kedua. Demikian seterusnya hingga algoritma mencapai *level* terakhir permainan. Maka dari itu algoritma *Minimax* ini mempunyai kelemahan, yaitu ketika jumlah *input* yang dimasukkan menjadi banyak (Ilham, 2008). Algoritma *Breadth First Search* merupakan salah satu algoritma traversal dalam graf. *Breadth First Search* adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara *preorder*, yaitu mengunjungi suatu simpul, kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih

dahulu. Dari sudut pandang algoritma, semua simpul anak didapatkan dengan memperluas simpul yang ditambahkan pada antrian FIFO [1]. Algoritma *Breadth First Search* banyak sekali diterapkan dalam berbagai permainan seperti othello dan catur untuk mencari solusi terbaik sehingga algoritma *Breadth First Search* menjadi rekomendasi yang cukup baik dalam penerapan di bidang *game*.

Pada penelitian ini akan digunakan dua algoritma, yaitu *Minimax* dan *Breadth First Search*. Algoritma *Minimax* dan *Breadth First Search* telah diterapkan pada permainan Othello [2]. Dari penelitian [2] diperoleh hasil yang telah berjalan sesuai alur dan dapat mengambil langkah timbal balik dengan mengambil poin maksimum bagi komputer. Algoritma *Minimax* juga telah diterapkan pada permainan Pingdoll [3]. Pada penelitian [3] diperoleh bahwa algoritma *Alpha-Beta Pruning* sedikit lebih cepat daripada *Minimax* dan jumlah *node* yang ditelusuri pada algoritma *Alpha-Beta Pruning* jauh lebih sedikit dibandingkan *Minimax*.

Algoritma *Breadth First Search* juga telah diterapkan pada bidang *edugame* [4]. Dari penelitian [4], algoritma *Breadth First Search* dapat memberikan solusi dalam pencarian *icon*, ketika pemain mengalami kesulitan dalam menemukan *icon* yang berpasangan. Dari hasil penelitian sebelumnya, algoritma *Minimax* dan *Breadth First Search* dapat digunakan secara maksimal tetapi dalam perubahan variasi jumlah kemungkinan solusi akan membuat perubahan yang signifikan meskipun variasinya kecil. Sedangkan pada permainan *Tic-Tac-Toe* variasi lebih sedikit dibandingkan dengan permainan Othello sehingga penggunaan algoritma *Minimax* dan *Breadth First Search* pada permainan *Tic-Tac-Toe* memungkinkan terjadinya perbedaan hasil yang signifikan.

Berdasarkan hal tersebut, maka akan dilakukan perbandingan algoritma *Minimax* dan *Breadth First Search* untuk menentukan waktu respon dan mendapatkan algoritma terbaik pada permainan *Tic-Tac-Toe*.

II. STUDI LITERATUR

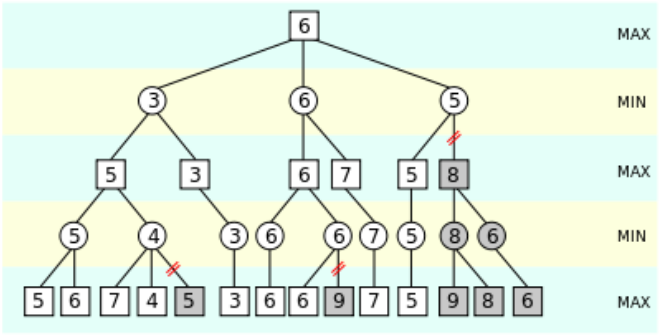
Pada bagian ini berisi teori mengenai permainan *Tic-Tac-Toe*, *Minimax*, *Breadth First Search*, dan penelitian-penelitian sebelumnya.

A. *Tic-Tac-Toe*

Secara umum *Tic-Tac-Toe* dimainkan oleh dua orang. Biasanya permainan ini menggunakan 2 bidak dengan simbol X dan O, yang dimainkan pada sarana papan persegi berukuran 3×3. Cara bermain *Tic-Tac-Toe* cukup sederhana dimana pemain harus menjalankan bidak yang dipilih secara bergantian. Pemain akan dinyatakan menang jika salah satu bidak dapat menyusun 3 baris bidak secara horizontal, vertikal maupun diagonal. *Tic-Tac-Toe* mempunyai beberapa nama lain, seperti catur jawa, nought and crosses, Xs and Os [5].

B. *Minimax*

Algoritma *Minimax* merupakan algoritma yang sering digunakan dalam permainan berbasis *Artificial Intelligence* (AI). Pada algoritma *Minimax*, penelusuran setiap perpindahan posisi dalam permainan akan dilakukan menurut aturan yang ada sehingga didapatkan seluruh kemungkinan yang ada sampai akhir permainan dilakukan [6]. Penelusuran setiap langkah akan menghasilkan pohon permainan yang berisi semua kemungkinan dari permainan tersebut. Setiap penelusuran akan membutuhkan *resource* yang besar dalam melakukan pencarian pohon solusi. Pohon solusi yang dihasilkan sesuai dengan jumlah kombinasi dari setiap kemungkinan langkah dalam sebuah permainan. Permainan *Tic-Tac-Toe* mempunyai aturan dan basis pengetahuan yang lebih sedikit dibandingkan dengan permainan catur. Dengan aturan yang lebih sedikit, proses komputasi untuk penelusuran setiap langkah berdasarkan kondisi. Algoritma *Minimax* bekerja dengan mencari langkah sesuai dengan fungsi heuristik. Fungsi heuristik merepresentasikan hasil permainan dalam setiap pemilihan langkah. Pada umumnya permainan *Tic-Tac-Toe* menggunakan nilai 1, 0, dan -1 untuk mewakilkan hasil akhir permainan berupa menang, seri, dan kalah. Selanjutnya penentuan simpul bergantung pada pohon permainan yang akan dipilih. Simpul yang akan dipilih adalah simpul dengan nilai heuristik yang membuat hasil permainan lebih optimal [6]. Penelusuran simpul pohon algoritma *Minimax* dapat dilihat pada Gambar 1.



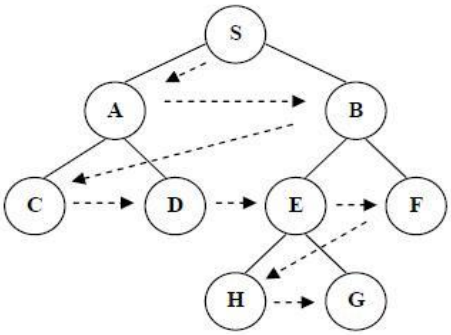
Gambar 1 Ilustrasi *Minimax*

C. *Breadth First Search*

Breadth First Search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara *preorder*, yaitu mengunjungi suatu simpul lalu mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu kemudian dilanjutkan dengan mengunjungi simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang sebelumnya dikunjungi, demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum simpul-simpul pada aras $d + 1$.

Algoritma *Breadth First Search* memerlukan sebuah antrian q untuk menyimpan simpul yang telah dikunjungi. Simpul-simpul ini diperlukan sebagai acuan untuk

mengunjungi simpul-simpul yang bertetanggaan dengannya. Tiap simpul yang telah dikunjungi masuk ke dalam antrian hanya satu kali. Algoritma ini juga membutuhkan tabel *boolean* untuk menyimpan simpul yang telah dikunjungi sehingga tidak ada simpul yang dikunjungi lebih dari satu kali [2]. Penelusuran simpul pohon algoritma *Breadth First Search* dapat dilihat pada Gambar 2.



Gambar 2 Ilustrasi *Breadth First Search*

D. Penelitian Terdahulu

Pada penelitian [7], algoritma *Minimax* membutuhkan waktu yang cukup lama dibandingkan dengan algoritma *Aplha Beta Pruning* dalam menentukan langkah dengan asumsi langkah yang dilakukan merupakan langkah terbaik di setiap posisi pada permainan catur. Algoritma *Minimax* sering digunakan pada penelitian di bidang permainan. Pada penelitian [8] melakukan perbandingan algoritma *Minimax* dan *Negascout* dalam permainan *checkers*. Berdasarkan penelitian [8], algoritma *Negascout* menghasilkan jumlah *node* lebih sedikit dan waktu pencarian lebih singkat jika dibandingkan dengan algoritma *Minimax*, yakni pada *depth* 3, 5 dan 7.

Algoritma *Minimax* digunakan juga sebagai pengambil keputusan dalam permainan *Tic-Tac-Toe* pada penelitian [6]. Pada penelitian [5] mengemukakan bahwa dengan menggunakan fungsi seleksi yang tepat, algoritma *Greedy* dapat memberikan hasil yang maksimal dengan waktu yang lebih cepat daripada algoritma *Minimax* pada permainan *Tic-Tac-Toe*. Selain *Minimax*, *Breadth First Search* juga digunakan pada penelitian [4] untuk memberikan solusi dalam proses pencarian icon, ketika pemain mengalami kesulitan dalam menemukan icon yang berpasangan. Selain itu *Minimax* juga digunakan dalam permainan dengan optimasi MTD(f) [9], [10].

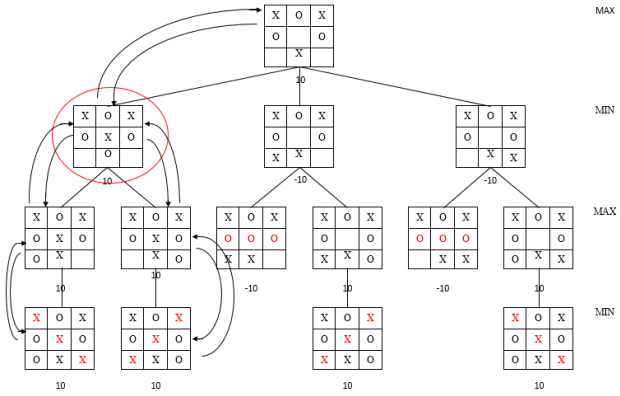
Pada penelitian [1] dan [11] algoritma *Depth First Search* tidak akan selalu lebih baik daripada algoritma *Breadth First Search* apabila solusi berada di simpul kanan dan solusi berada tidak terlalu dalam, maka algoritma *Depth First Search* akan membutuhkan lebih banyak penelusuran daripada algoritma *Breadth First Search*. Di sisi lain *Breadth First Search* dan *Depth First Search* juga digunakan untuk pencarian kata dalam sebuah permainan *puzzle* [12].

III. METODOLOGI

Secara normal permainan *Tic-Tac-Toe* memiliki papan dengan ukuran 3×3, dan memiliki dua bidak, yaitu bidak X dan bidak O. Namun pada penelitian ini, permainan *Tic-Tac-Toe* dirancang dengan ukuran papan yang memiliki sejumlah variasi, yaitu: 3×3, 5×5, 7×7 dan 9×9. Algoritma yang diterapkan pada permainan ini sebagai AI adalah *Minimax* dan *Breadth First Search* untuk menentukan langkah terbaik. Parameter pengujian pada penelitian ini adalah waktu respon dan jumlah *node*.

A. Rancangan Algoritma Minimax Pada Permainan Tic-Tac-Toe

Pada algoritma *Minimax*, penelusuran akan dilakukan sesuai *rule* yang ada. Penelusuran tersebut dapat menghasilkan pohon permainan yang berisi semua kemungkinan dari permainan tersebut. Berikut ilustrasi pohon permainan untuk algoritma *Minimax* pada Gambar 3.

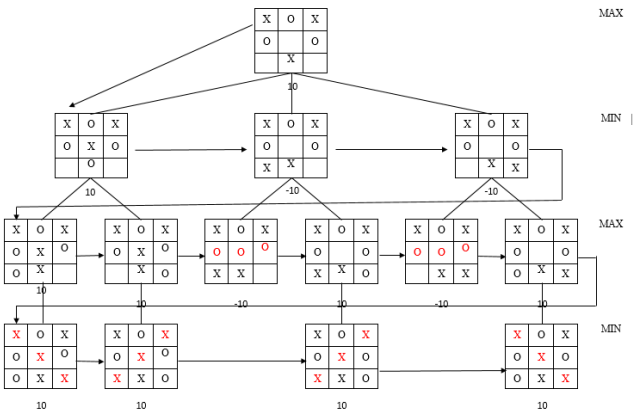


Gambar 3 Pohon Algoritma *Minimax* Pada Permainan *Tic-Tac-Toe*

B. Rancangan Algoritma Breadth First Search Pada Permainan Tic-Tac-Toe

Algoritma *Breadth First Search* menelusuri pohon secara melebar. Algoritma ini akan menyeleksi simpul awal kemudian mengecek apakah simpul tersebut merupakan solusi atau tidak, jika tidak maka semua simpul tetangga akan dimasukkan ke dalam antrian, jika ya maka nilai simpul akan dikembalikan. Berikut ilustrasi pohon permainan untuk algoritma *Breadth First Search* pada Gambar 4.

e-ISSN : 2443-2229

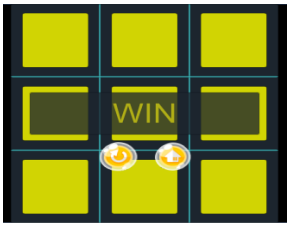


Gambar 4 Pohon Algoritma *Breadth First Search* Pada Permainan *Tic-Tac-Toe*

C. *Rancangan Level Permainan*

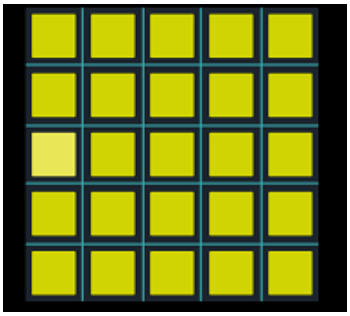
Tingkat kesulitan (*level*) permainan dirancang menjadi 4 bagian, yaitu mudah, sedang, sulit, dan sangat sulit dengan jumlah kotak yang berbeda-beda.

1) Level Mudah: Level mudah memiliki 9 kotak papan. Pengujian ini akan dilakukan AI melawan AI dengan hasil uji berupa perbandingan jumlah *node* yang dilalui dan waktu respon. Rancangan papan permainan *Tic-Tac-Toe* pada *level* mudah dapat dilihat pada Gambar 5.



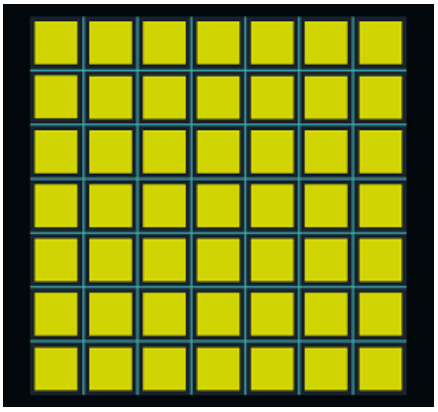
Gambar 5 Rancangan Papan Permainan *Tic-Tac-Toe* pada *Level* Mudah

2) Level Sedang: Level sedang memiliki 25 kotak papan. Pengujian ini akan dilakukan AI melawan AI dengan hasil uji berupa perbandingan jumlah *node* yang dilalui dan waktu respon. Rancangan papan permainan *Tic-Tac-Toe* pada *level* sedang dapat dilihat pada Gambar 6.



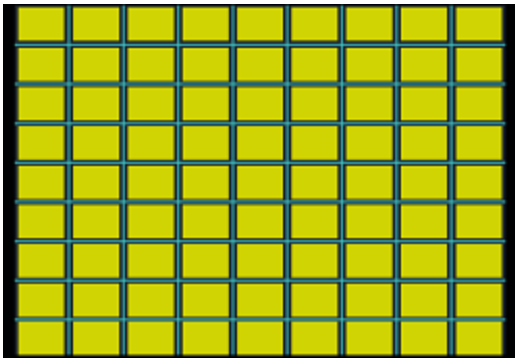
Gambar 6 Rancangan Papan Permainan *Tic-Tac-Toe* pada *Level* Sedang

3) Level Sulit: Level sulit memiliki 49 kotak papan. Pengujian ini akan dilakukan AI melawan AI dengan hasil uji berupa perbandingan jumlah *node* yang dilalui dan waktu respon. Rancangan papan permainan *Tic-Tac-Toe* pada *level* sulit dapat dilihat pada Gambar 7.



Gambar 7 Rancangan Papan Permainan *Tic-Tac-Toe* pada *Level* Sulit

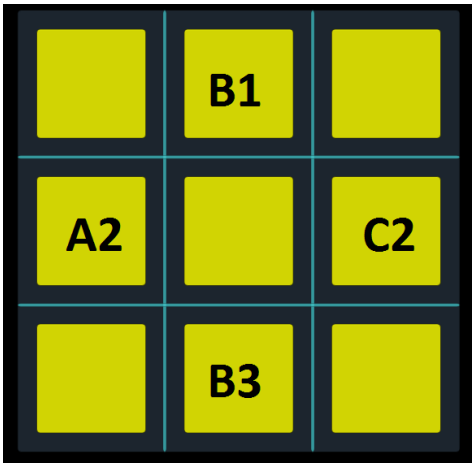
4) Level Sangat Sulit: Level sangat sulit memiliki 81 kotak papan. Pengujian ini akan dilakukan AI melawan AI dengan hasil uji berupa perbandingan jumlah *node* yang dilalui dan waktu respon. Rancangan papan permainan *Tic-Tac-Toe* pada *level* sangat sulit dapat dilihat pada Gambar 8.



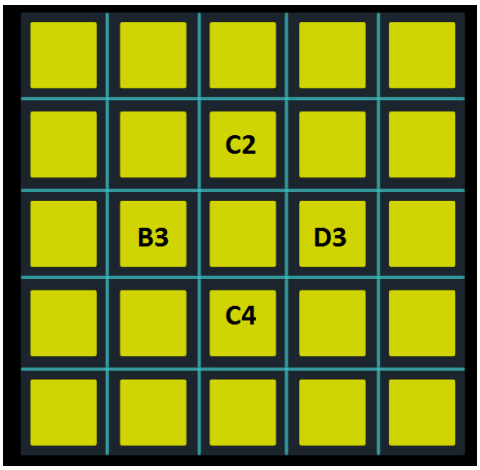
Gambar 8 Rancangan Papan Permainan *Tic-Tac-Toe* pada *Level* Sangat Sulit

D. *Rancangan Titik Pengujian Dalam Permainan*

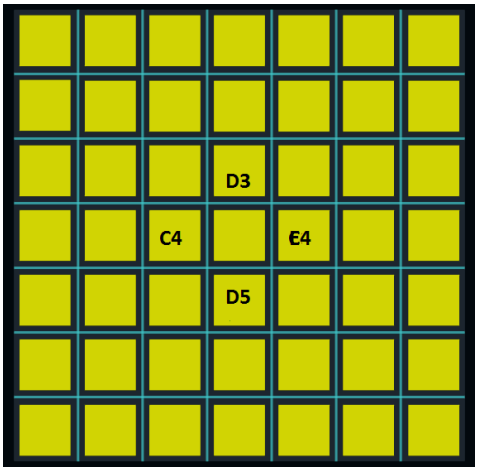
Pada dasarnya, langkah pertama yang akan dilakukan dalam permainan *Tic-Tac-Toe* merupakan posisi dimana memiliki potensi untuk bisa menciptakan pola berurutan di sekitarnya dengan jumlah kemungkinan paling banyak. Pada papan permainan dengan ukuran 3×3 (*level* mudah), terdapat 4 (empat) titik yang memiliki potensi sebagai posisi terbaik untuk langkah pertama dalam permainan *Tic-Tac-Toe*. Keempat titik tersebut akan digunakan sebagai titik pengujian dalam permainan pada papan berukuran 3×3 (*level* mudah). Dengan 4 (empat) titik yang sama juga digunakan untuk menguji papan permainan dengan ukuran 5×5 (*level* sedang), 7×7 (*level* sulit), dan 9×9 (*level* sangat sulit). Dengan demikian, posisi pengujian akan menguji 4 (empat) titik pada tiap *level* permainan. Titik pengujian dapat dilihat pada Gambar 9, Gambar 10, Gambar 11 dan Gambar 12.



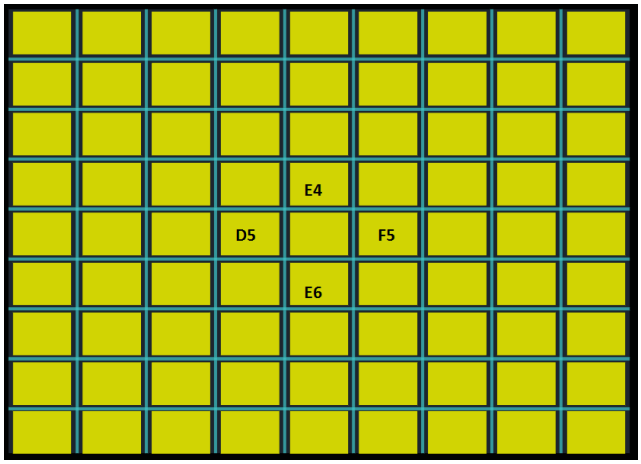
Gambar 9 Titik Pengujian Permainan *Tic-Tac-Toe* pada *Level* Mudah



Gambar 10 Titik Pengujian Permainan *Tic-Tac-Toe* pada *Level* Sedang



Gambar 11 Titik Pengujian Permainan *Tic-Tac-Toe* pada *Level* Sulit



Gambar 12 Titik Pengujian Permainan *Tic-Tac-Toe* pada *Level* Sangat Sulit

E. Skenario Rancangan Pengujian

Pengujian penelitian ini mempunyai paramater uji, yaitu waktu respon AI dan jumlah *node* yang dihasilkan. Pengujian pada tiap *level* akan menguji algoritma *Minimax* melawan *Breadth First Search*.

Level mudah akan menggunakan papan permainan 3×3. Permainan dapat dinyatakan menang jika AI dapat membentuk 3 bidak yang sama secara horizontal, vertikal atau diagonal. Berikut ini adalah skenario pengujian pada *level* mudah.

1) Skenario Pertama: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah B1.

2) Skenario Kedua: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah A2.

3) Skenario Ketiga: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah C2.

4) Skenario Keempat: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah B3.

Level sedang akan menggunakan papan permainan 5×5. Permainan dapat dinyatakan menang jika AI dapat membentuk 5 bidak yang sama secara horizontal, vertikal atau diagonal. Berikut ini adalah skenario pengujian pada *level* sedang.

1) Skenario Kelima: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah C2.

2) Skenario Keenam: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah B3.

3) Skenario Ketujuh: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah D3.

4) Skenario Kedelapan: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah C4.

Level sulit akan menggunakan papan permainan 7×7. Permainan dapat dinyatakan menang jika AI dapat

membentuk 7 bidak yang sama secara horizontal, vertikal atau diagonal. Berikut ini adalah skenario pengujian pada level sulit.

1) Skenario Kesembilan: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah D3.

2) Skenario Kesepuluh: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah C4.

3) Skenario Kesebelas: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah E4.

4) Skenario Keduabelas: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah D5.

Level sangat sulit akan menggunakan papan permainan 9×9. Permainan dapat dinyatakan menang jika AI dapat membentuk 9 bidak yang sama secara horizontal, vertikal atau diagonal. Berikut ini adalah skenario pengujian pada *level* sangat sulit.

1) Skenario Ketigabelas: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah E4.

2) Skenario Keempatbelas: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah D5.

3) Skenario Kelimabelas: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah F5.

4) Skenario Keenambelas: Minimax melawan *Breadth First Search* dengan titik awal permainan adalah E6.

IV. HASIL DAN PEMBAHASAN

Pengujian terhadap permainan *Tic-Tac-Toe* dilakukan pada 4 *level* yaitu mudah, sedang, sulit, dan sangat sulit. Setiap *level* memiliki 4 titik pengujian. Pengujian akan menentukan performa yang lebih baik antara algoritma *Minimax* dan *Breadth First Search* dengan membandingkan parameter waktu respon dan jumlah *node*.

TABEL I
HASIL PENGUJIAN SKENARIO PERTAMA PADA TITIK B1

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	65	4.7673792
3	211	8.8183019	4	35	2.8849392
5	93	7.1046885	6	16	0.6359504
7	17	1.6019423	8	3	0.2291108
9	1	0.0557809			
Total	322	4.395178		119	2.129345

Tabel I merupakan hasil pengujian skenario pertama dengan 9 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Breadth First Search* lebih unggul pada waktu respon dan jumlah *node* daripada algoritma *Minimax*.

TABEL III
HASIL PENGUJIAN SKENARIO KEDUA PADA TITIK A2

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	64	4.6753088
3	199	8.0970911	4	36	2.5390584
5	69	5.3673168	6	18	1.276144
7	17	1.1240621	8	4	0.3041696
9	1	0.0746293			
Total	286	3.665775		222	2.19867

Tabel II merupakan hasil pengujian skenario kedua dengan 9 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Breadth First Search* lebih unggul pada waktu respon dan jumlah *node* daripada algoritma *Minimax*.

TABEL IIIII
HASIL PENGUJIAN SKENARIO KETIGA PADA TITIK C2

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	66	5.4173696
3	223	13.0119608	4	38	2.26647
5	117	7.6699467	6	19	1.31784
7	25	1.6777725	8	7	0.59088
9	1	0.0401725			
Total	366	5.599963		130	2.39814

Tabel III merupakan hasil pengujian skenario ketiga dengan 9 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Breadth First Search* lebih unggul pada waktu respon dan jumlah *node* daripada algoritma *Minimax*.

TABEL IV
HASIL PENGUJIAN SKENARIO KEEMPAT PADA TITIK B3

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	67	5.36704
3	235	21.297627	4	37	3.2082732
5	141	10.8983694	6	19	1.519
7	41	3.2418085	8	5	0.2866258
9	1	0.0656364			
Total	418	8.87586		128	2.595235

Tabel IV merupakan hasil pengujian skenario keempat dengan 9 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Breadth First Search* lebih unggul pada waktu respon dan jumlah *node* daripada algoritma *Minimax*.

TABEL V
HASIL PENGUJIAN SKENARIO KELIMA PADA TITIK C2

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	288	25.3622592
3	264	22.4489614	4	242	20.3284356
5	220	18.284742	6	200	16.72812
7	180	14.9198773	8	162	13.4245188
9	144	16.1444359	10	128	10.4480512
11	112	9.25146	12	98	7.9732604
13	84	6.9765735	14	72	5.8438224
15	60	5.0391544	16	50	4.10121
17	40	3.6659061	18	32	2.6704512
19	24	3.0187822	20	18	2.5926336
21	12	1.03082	22	8	0.6746752
23	4	0.3888162	24	2	0.1685836
25	1	0.0407882			
Total	1145	8.434193		1300	9.193002

Tabel V merupakan hasil pengujian skenario kelima dengan 25 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada waktu respon dan jumlah *node* daripada algoritma *Breadth First Search*.

TABEL VI
HASIL PENGUJIAN SKENARIO KEENAM PADA TITIK B3

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	288	27.7944768
3	264	24.7018538	4	242	19.1345528
5	220	18.3719369	6	200	19.88276
7	180	15.0135207	8	162	13.8514536
9	144	11.8809345	10	128	10.6846976
11	112	9.6907275	12	98	7.9581488
13	84	7.1215417	14	72	5.9217924
15	60	5.1132664	16	50	4.10953
17	40	3.5645832	18	32	2.6979136
19	24	2.0265028	20	31	1.4509332
Total	1128	10.83165		1303	11.34863

Tabel VI merupakan hasil pengujian skenario keenam dengan 20 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah yang *node* dan waktu respon daripada algoritma *Breadth First Search*.

TABEL VII
HASIL PENGUJIAN SKENARIO KETUJUH PADA TITIK D3

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	281	17.5302016
3	263	12.5170331	4	251	13.1810832
5	221	15.4781367	6	206	11.87716
7	182	9.1120859	8	166	10.7123284
9	128	9.5833069	10	124	9.0928896
11	120	7.822945	12	93	10.4921348
13	85	5.3327208	14	81	9.4787856
15	62	8.2418303	16	60	4.47312
17	39	4.823469	18	34	3.9314368
19	21	3.0467122	20	18	2.7947044
21	8	1.1644475	22	6	0.9511072
Total	1129	7.712269		1320	8.592268

Tabel VII merupakan hasil pengujian skenario ketujuh dengan 22 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada waktu respon dan jumlah *node* daripada algoritma *Breadth First Search*.

TABEL VIII
HASIL PENGUJIAN SKENARIO KEDELAPAN PADA TITIK C4

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	266	17.0239168
3	263	19.983407	4	241	22.0014004
5	226	12.0224607	6	203	15.44272
7	181	14.4486279	8	161	19.3033692
9	152	16.2978371	10	146	13.4903296
11	131	12.3345225	12	127	11.291099
13	112	9.7408016	14	108	7.2361744
15	83	8.0975499	16	76	5.11644
17	61	6.9670287	18	56	3.3943296
19	40	4.0395012	20	38	2.421234
21	23	1.1590775	22	16	1.6568016
23	8	0.3777354	24	4	0.2231296
25	1	0.0711267			
Total	1281	8.434193		1442	9.193002

Tabel VIII merupakan hasil pengujian skenario kedelapan dengan 25 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada waktu respon dan jumlah *node* daripada algoritma *Breadth First Search*.

TABEL IX
HASIL PENGUJIAN SKENARIO KESEMBILAN PADA TITIK D3

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	768	64.1223128
3	746	64.3032536	4	705	30.7820162
5	695	47.9243925	6	635	27.3156048
7	622	47.2945232	8	588	40.0387428
9	570	23.7879430	10	533	32.989067
11	517	39.8618103	12	481	33.4162973
13	446	33.7082025	14	432	17.751312
15	418	30.53092	16	385	20.9815926
17	363	28.0875606	18	341	13.7832789
19	331	17.7890693	20	300	19.91778
21	289	19.8615606	22	261	17.8426378
23	243	11.2750785	24	225	12.341732
25	218	11.2089583	26	192	10.244064
27	182	12.0021459	28	156	6.5548604
29	147	7.7707875	30	133	6.57913
31	120	7.439066	32	108	4.3153344
33	96	4.8669623	34	85	5.6684032
35	75	3.5875425	36	65	2.57065106

37	56	3.00944496	38	48	2.4989904
39	40	2.03787796	40	33	2.05714
41	27	1.4234724	42	21	1.032442
43	16	1.3697427	44	12	0.564546
45	10	0.451273	46	8	0.39328693
47	6	0.1585308	48	4	0.06423564
49	1	0.0312545			
Total	6234	17.49089		6519	15.57606

Tabel IX merupakan hasil pengujian skenario kesembilan dengan 49 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah *node* dan algoritma *Breadth First Search* lebih unggul pada waktu respon.

TABEL X
HASIL PENGUJIAN SKENARIO KESEPULUH PADA TITIK C4

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	803	68.2738095
3	742	60.31852828	4	723	52.3184188
5	676	46.25281961	6	635	29.838529
7	603	412.684939	8	588	42.583948
9	561	23.325226	10	534	35.236566
11	516	40.24527	12	489	34.536373
13	448	33.168438	14	430	16.352325
15	412	31.425422	16	395	20.646748
17	361	28.6447437	18	327	15.324532
19	316	18.3342224	20	303	20.214361
21	289	18.239428	22	273	16.439939
23	262	14.452422	24	223	11.452245
25	205	10.3115252	26	179	9.121425
27	166	7.353636	28	152	7.00134253
29	140	6.9343464	30	135	6.264564
31	115	5.23536364	32	101	4.1363634
33	86	4.65695309	34	80	3.639693
35	79	3.49360	36	75	2.62096090
37	61	3.035920	38	56	2.5325262
39	48	1.9235263	40	43	1.73946934
41	25	1.44069920	42	23	1.033284
43	19	1.30190931	44	16	0.8163633
Total	6130	36.799		6150	18.27835

Tabel X merupakan hasil pengujian skenario kesepuluh dengan 44 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah *node* dan algoritma *Breadth First Search* lebih unggul pada waktu respon.

TABEL XI
HASIL PENGUJIAN SKENARIO KESEBELAS PADA TITIK E4

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	742	62.643634
3	732	53.64322	4	705	32.346373
5	674	52.3663	6	637	31.26442
7	611	42.364363	8	573	40.325235
9	541	36.436346	10	526	32.42622
11	511	39.433473	12	477	32.525252
13	453	33.352352	14	434	18.345226
15	411	30.636765	16	385	15.63636
17	364	27.534534	18	323	14.35232
19	321	16.232367	20	295	19.574743
21	286	13.043633	22	273	17.447744
23	252	11.463633	24	232	12.54366
25	211	10.143288	26	187	9.148838
27	164	8.1248881	28	159	6.834789
29	145	7.4354389	30	130	6.394892
31	117	6.437373	32	112	4.264627
33	94	4.645245	34	84	5.876756
35	77	4.245525	36	62	2.345222
37	55	3.5425234	38	45	2.252524
39	43	1.884892	40	37	1.7312312
41	25	1.4323223	42	22	0.9424924
43	14	0.6354353	44	11	0.532423
45	10	0.4499359	46	8	0.34953995

TABEL XI
HASIL PENGUJIAN SKENARIO KESEBELAS PADA TITIK E4

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
47	6	0.1425435	48	4	0.09124124
49	1	0.049529			
Total	6118	16.90316		6463	15.42482

Tabel XI merupakan hasil pengujian skenario kesebelas dengan 49 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah *node* dan algoritma *Breadth First Search* lebih unggul pada waktu respon.

TABEL XII
HASIL PENGUJIAN SKENARIO KEDUABELAS PADA TITIK D5

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	738	62.99459
3	730	61.939299	4	725	53.4359959
5	684	48.6443925	6	655	28.36765048
7	606	47.299005	8	582	40.0384567
9	567	22.657765	10	542	31.569067
11	502	38.495949	12	482	34.499359
13	488	32.295929	14	474	19.255929
15	412	29.494995	16	386	20.989267
17	363	27.596949	18	378	18.29599
19	325	17.393949	20	294	15.293499
21	254	19.29599	22	251	17.3932
23	243	16.949342	24	232	13.942499
25	212	13.58353	26	187	10.347599
27	160	11.002138	28	155	7.3498249
29	143	6.938498	30	138	6.3249852
31	121	5.394829	32	115	6.0290923
33	92	4.283189	34	83	5.242385
35	72	3.825927	36	69	2.892489
37	55	3.854838	38	45	2.928929
39	46	2.9894482	40	38	2.029831
41	31	1.9213891	42	23	1.923422
43	21	0.5697427	44	15	0.264546
Total	6127	19.82986		6110	18.24585

Tabel XII merupakan hasil pengujian skenario keduabelas dengan 44 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Breadth First Search* lebih unggul pada jumlah *node* dan waktu respon daripada algoritma *Minimax*.

TABEL XIII
HASIL PENGUJIAN SKENARIO KETIGABELAS PADA TITIK E4

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	1066	63.1969067
3	1040	59.49721283	4	1014	63.2529144
5	988	56.854902	6	962	55.65409
7	937	63.5943809	8	912	59.64922
9	888	63.2590820	10	864	45.483274
11	840	60.5490880	12	816	48.19401
13	793	43.6800225	14	770	39.32642
15	748	38.6549082	16	726	49.983298
17	704	42.7490823	18	682	48.960345
19	661	40.3689086	20	640	32.45684
21	620	37.804256	22	600	29.4364537
23	580	28.6908224	24	560	27.3459875
25	541	26.720943	26	522	38.368643
27	504	32.420985	28	486	28.56936
29	468	33.638963	30	450	28.747843
31	433	23.386089	32	416	35.346437
33	400	19.168632	34	384	18.426786
35	368	27.863083	36	352	22.862572
37	337	19.630808	38	322	22.74267
39	308	16.806448	40	294	20.49024
41	280	12.278453	42	266	12.794027
43	253	33.860528	44	240	15.3079232
45	228	10.65892	46	216	10.8697
47	204	9.368032	48	192	8.93573
49	181	16.480289	50	170	7.37573
51	160	9.746892	52	150	10.809337
53	140	8.443593	54	130	8.435757
55	121	7.53955	56	112	6.83735
57	104	4.573034	58	96	4.3693757
59	88	5.873581	60	80	4.8947347
61	73	7.33583	62	66	4.63478
63	60	4.728398	64	54	3.57478
65	48	3.2786302	66	42	2.3473
67	37	1.6446309	68	32	1.409474
69	28	1.684589	70	24	1.03683
71	20	0.8638902	72	16	0.78537
73	19	0.669480	74	15	0.4377357
75	14	0.323482	76	12	0.2463235
77	8	0.23259	78	6	0.176325
79	4	0.12289	80	2	0.092578
81	1	0.072313			
Total	14229	21.87796		14759	22.09664

Tabel XIII merupakan hasil pengujian skenario ketigabelas dengan 81 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah *node* dan waktu respon daripada algoritma *Breadth First Search*.

TABEL XIV
HASIL PENGUJIAN SKENARIO KEEMPATBELAS PADA TITIK D5

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	1183	64.27531
3	1097	59.53879	4	1042	60.278686
5	978	51.42575	6	948	58.157309
7	931	44.82795	8	916	46.42786
9	893	63.246796	10	874	45.37824
11	854	63.279795	12	836	49.7349892
13	782	42.267868	14	766	43.7482797
15	749	38.43897	16	736	40.924788
17	714	42.42368	18	681	42.729889
19	659	40.428799	20	640	38.342423
21	626	38.79236	22	606	35.34576
23	591	29.24794	24	574	32.43734
25	541	25.6563	26	523	30.3489
27	504	32.4623	28	492	29.45387
29	473	31.5262	30	457	29.34566
31	429	29.74272	32	414	35.263424
33	395	25.62462	34	381	24.4977942
35	362	20.435325	36	352	18.53276
37	331	19.47246	38	327	22.423786
39	314	15.23526	40	286	20.4372889
41	278	11.366287	42	258	16.134789
43	248	18.36463	44	240	14.41387
45	223	15.246426	46	216	11.6164
47	208	10.63434	48	189	8.943278
49	172	15.257899	50	171	7.432789
51	161	9.528778	52	145	10.4287979
53	137	8.42798	54	129	12.0024178
55	119	7.12478	56	107	9.024787
57	98	5.43252	58	90	7.427998
59	83	5.4279789	60	78	6.2478979
61	71	7.336324	62	68	5.8959287
63	64	4.6489723	64	60	3.748924
65	57	3.27929	66	53	2.32658
67	47	1.43297	68	45	1.927985
69	40	1.3457978	70	35	0.82847687
71	33	`0.6787297	72	28	0.64287624
73	25	0.478726	74	21	0.4016873
75	18	0.3725885	76	13	0.292578678
77	10	0.2382756	78	7	0.17267236
79	4	0.125786	80	2	0.092857676
81	1	0.0623978			
Total	14320	21.54376		14989	22.2022

Tabel XIV merupakan hasil pengujian skenario keempatbelas dengan 81 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah *node* dan waktu respon daripada algoritma *Breadth First Search*.

TABEL XV
HASIL PENGUJIAN SKENARIO KELIMABELAS PADA TITIK F5

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	1205	68.1969067
3	1124	67.9721283	4	1031	66.2529144
5	990	63.289267	6	964	54.86789362
7	942	60.278005	8	914	51.35876
9	894	57.390884	10	876	46.85383
11	904	56.68081	12	886	44.358835
13	844	56.4303	14	795	40.683753
15	781	54.6390	16	776	42.987474
17	763	54.954264	18	732	50.5879343
19	711	51.682575	20	686	42.358578
21	674	49.257962	22	646	44.37648
23	598	47.4395792	24	573	47.689456
25	554	42.2479427	26	542	45.8468986
27	501	41.24897294	28	484	46.68464

TABEL XV
HASIL PENGUJIAN SKENARIO KELIMABELAS PADA TITIK F5

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
29	467	39.4298529	30	424	44.683858
31	413	35.65792	32	402	46.87637
33	384	33.16834	34	365	37.8769
35	356	30.354792	36	343	32.37573
37	352	29.54375	38	314	30.537573
39	305	26.649729	40	296	27.57347
41	291	22.6487982	42	287	25.624768
43	263	25.479793	44	234	20.29602
45	224	21.53479	46	216	19.35792
47	204	18.683069	48	196	18.937958
49	193	16.4353262	50	188	15.7956668
51	180	12.786735	52	175	16.80958
53	171	11.4454643	54	166	13.435753
55	160	10.573876	56	146	10.88358
57	125	9.57234624	58	113	9.36938
59	97	8.876386	60	93	7.89358
61	82	7.376853	62	74	6.34673
63	66	5.723926	64	58	3.5478
65	51	3.272935	66	49	2.5647
67	43	1.646439	68	40	1.547352
69	38	1.686439	70	35	1.8345784
71	29	0.8436939	72	25	1.476847
73	21	0.66430934	74	18	1.0865383
75	16	0.3649339	76	12	0.275486278
77	8	0.23649	78	6	0.126473
79	4	0.132580	80	2	0.1232489
81	1	0.076538			
Total	14824	26.95944		15387	27.25906

Tabel XV merupakan hasil pengujian skenario kelimabelas dengan 81 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah *node* dan waktu respon daripada algoritma *Breadth First Search*.

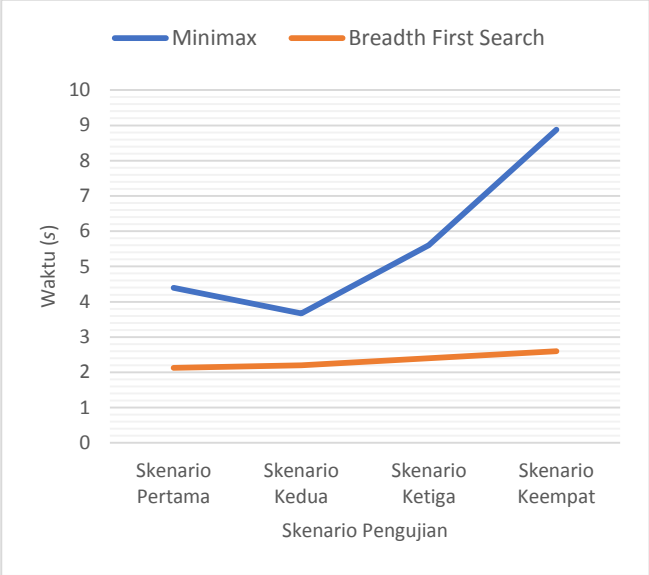
TABEL XVI
HASIL PENGUJIAN SKENARIO KEENAMBELAS PADA TITIK E6

Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
1	0	0	2	1006	59.5328779
3	984	55.2792597	4	932	52.2857878
5	924	56.289267	6	911	55.54378
7	902	60.528991	8	881	58.4278979
9	861	60.4326878	10	854	56.42678
11	841	59.5287777	12	804	52.278289
13	791	41.27532	14	762	37.7432979
15	753	33.74992	16	736	49.243789
17	714	40.6372	18	690	48.2347988
19	683	43.6383	20	664	45.53479
21	642	37.63877	22	626	39.346782
23	593	28.23578	24	551	26.453897
25	537	23.525541	26	520	38.26424
27	514	33.5235323	28	493	25.63462
29	478	33.45742	30	451	28.46124
31	431	23.463763	32	406	35.943797
33	397	19.23758	34	380	18.479927
35	371	26.25627	36	357	16.763573

TABEL XVI
HASIL PENGUJIAN SKENARIO KEENAMBELAS PADA TITIK E6

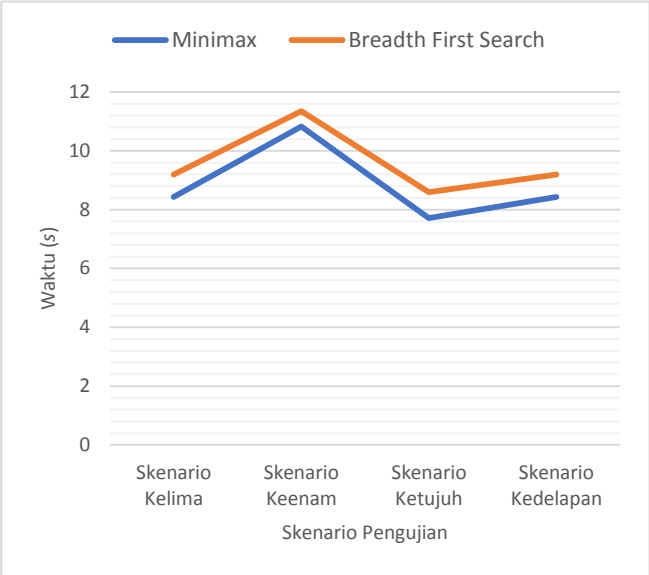
Langkah	*Minimax* (X)		Langkah	*Breadth First Search* (O)	
	Jumlah *Node*	Waktu Respon (s)		Jumlah *Node*	Waktu Respon (s)
37	325	19.4573	38	306	15.65426
39	296	16.2326	40	294	18.647265
41	277	12.63981	42	253	16.64425
43	247	14.57357	44	235	15.56262
45	225	12.3262	46	214	10.36536
47	201	11.946262	48	188	14.237528
49	180	10.637247	50	173	12.4364372
51	152	9..7634376	52	144	10.765363
53	139	9.357482	54	131	8.564337
55	128	7.945782	56	116	7.2294867
57	102	5.476363	58	88	4.257979
59	81	5.1840438	60	76	4.89278529
61	70	7.323476	62	66	4.24874627
63	60	4.724786	64	66	3,.5478368
65	63	3.2747799	66	59	2.4398369
67	54	1.64752987	68	49	1.62642
69	45	1.4328798	70	39	1.32384778
71	35	1.15183787	72	31	1.0012743
73	27	0.93247974	74	22	0.8434873
75	19	0.78237497	76	14	0.5328497
77	11	0.3543987	78	8	0.212879
79	4	0.1025987	80	2	0.0963488
81	1	0.0653093			
Total	14158	21.13506		14598	22.2080325

Tabel XVI merupakan hasil pengujian skenario keenambelas dengan 81 langkah. Hasil perbandingan parameter menunjukkan bahwa algoritma *Minimax* lebih unggul pada jumlah *node* dan waktu respon daripada algoritma *Breadth First Search*.



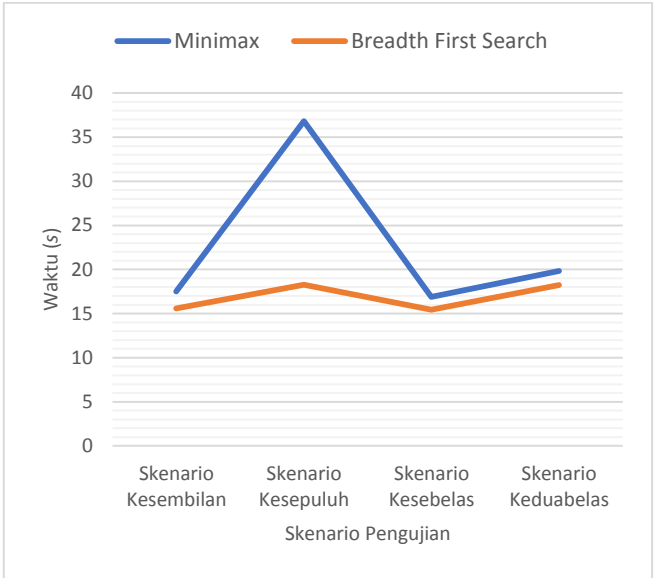
Gambar 13 Grafik Waktu Respon *Level* Mudah

Gambar 13 menunjukkan grafik waktu respon pengujian skenario pertama sampai skenario keempat. Skenario pertama sampai skenario keempat menunjukkan bahwa algoritma *Breadth First Search* mendapatkan hasil waktu respon yang lebih baik dibandingkan algoritma *Minimax*.



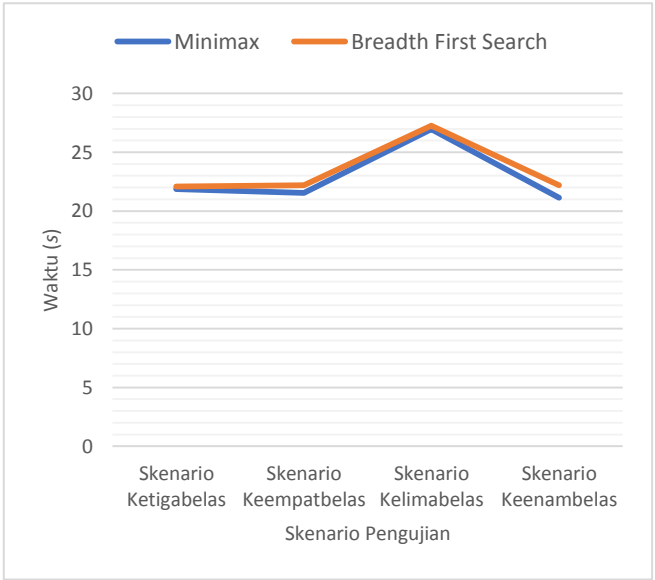
Gambar 14 Grafik Waktu Respon *Level* Sedang

Gambar 14 menunjukkan grafik waktu respon pengujian skenario kelima sampai skenario kedelapan. Skenario kelima sampai skenario kedelapan menunjukkan bahwa algoritma *Minimax* mendapatkan hasil waktu respon yang lebih baik dibandingkan algoritma *Breadth First Search*.



Gambar 15 Grafik Waktu Respon *Level* Sulit

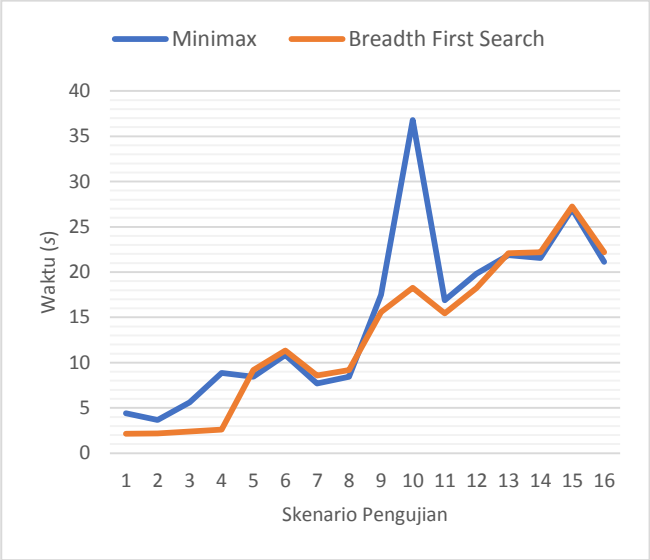
Gambar 15 menunjukkan grafik waktu respon pengujian skenario kesembilan sampai skenario keduabelas. Skenario kesembilan sampai skenario keduabelas menunjukkan bahwa algoritma *Breadth First Search* mendapatkan hasil waktu respon yang lebih baik dibandingkan algoritma *Minimax*.



Gambar 16 Grafik Waktu Respon *Level* Sangat Sulit

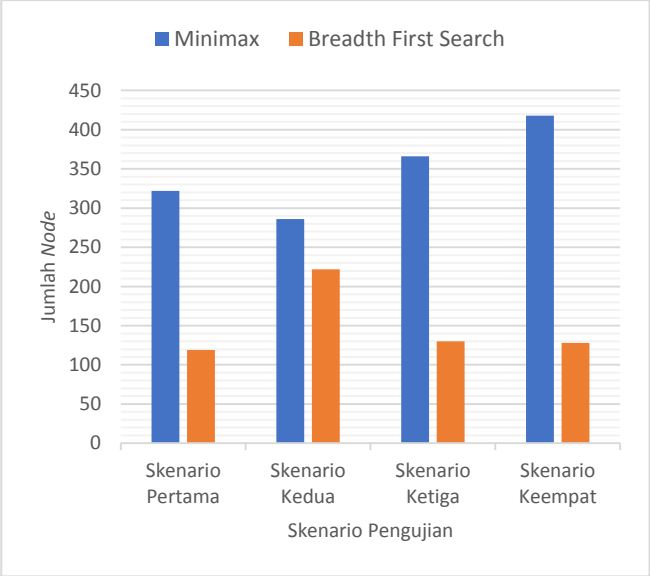
Gambar 16 menunjukkan grafik waktu respon pengujian skenario ketigabelas sampai skenario keenambelas. Skenario ketigabelas sampai skenario keenambelas menunjukkan bahwa algoritma *Minimax* mendapatkan hasil waktu respon yang lebih baik dibandingkan algoritma *Breadth First Search*.

e-ISSN : 2443-2229



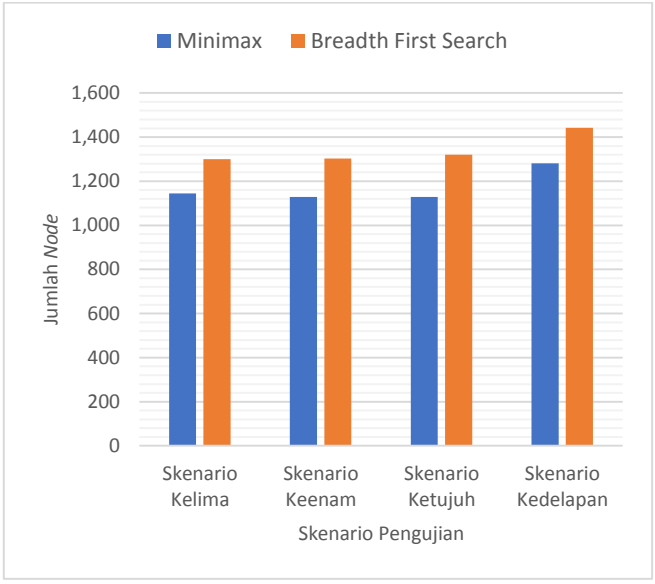
Gambar 17 Grafik Rata-Rata Waktu Respon Berdasarkan Skenario

Gambar 17 menunjukkan grafik hasil rata-rata waktu respon pengujian skenario pertama sampai skenario keenambelas. Skenario pertama sampai keenambelas mengalami perubahan yang berbeda pada tiap *level*, berdasarkan hasil pengujian keseluruhan bahwa algoritma *Breadth First Search* lebih unggul sebanyak 1,9718 *s* daripada algoritma *Minimax*.



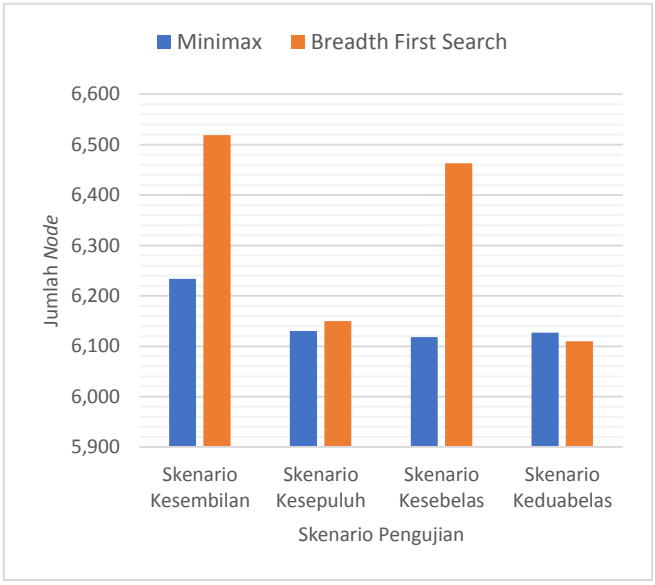
Gambar 18 Grafik Jumlah *Node* Pada *Level* Mudah

Gambar 18 menunjukkan grafik jumlah *node* pengujian skenario pertama sampai skenario keempat. Skenario pertama sampai skenario keempat menunjukkan bahwa algoritma *Breadth First Search* mendapatkan hasil jumlah *node* yang lebih baik dibandingkan algoritma *Minimax*.



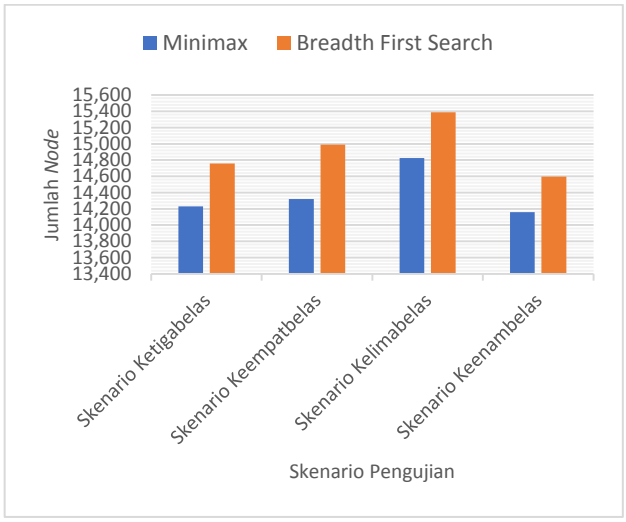
Gambar 19 Grafik Jumlah *Node* Pada *Level* Sedang

Gambar 19 menunjukkan grafik jumlah *node* pengujian skenario kelima sampai skenario kedelapan. Skenario kelima sampai skenario kedelapan menunjukkan bahwa algoritma *Minimax* mendapatkan hasil jumlah *node* yang lebih baik dibandingkan algoritma *Breadth First Search*.



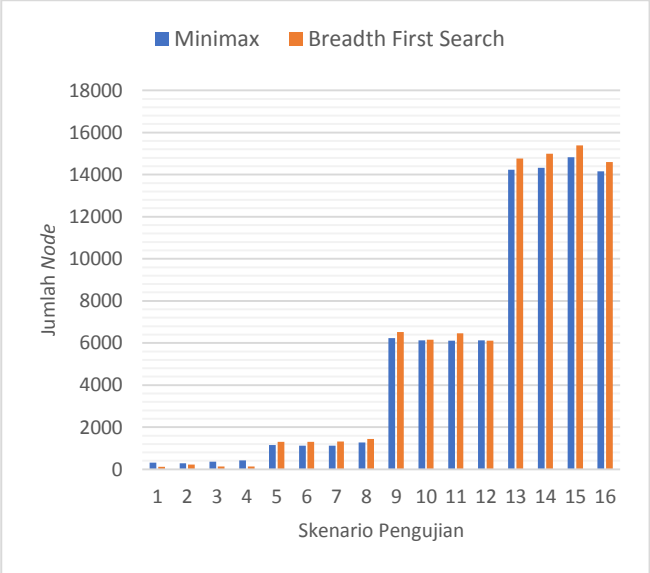
Gambar 20 Grafik Jumlah *Node* Pada *Level* Sulit

Gambar 20 menunjukkan grafik jumlah *node* pengujian skenario kesembilan sampai skenario keduabelas. Skenario kesembilan, skenario kesepuluh dan skenario kesebelas menunjukkan bahwa algoritma *Minimax* mendapatkan hasil jumlah *node* yang lebih baik dibandingkan algoritma *Breadth First Search*. Sedangkan, pada pengujian skenario keduabelas algoritma *Breadth First Search* mendapatkan jumlah *node* lebih baik dibandingkan algoritma *Minimax*.



Gambar 21 Grafik Jumlah *Node* Pada *Level* Sangat Sulit

Gambar 21 menunjukkan grafik jumlah *node* pengujian skenario ketigabelas sampai skenario keenambelas. Skenario ketigabelas sampai skenario keenambelas menunjukkan bahwa algoritma *Minimax* mendapatkan hasil jumlah *node* yang lebih baik dibandingkan algoritma *Breadth First Search.*



Gambar 22 Grafik Rata-Rata Jumlah *Node* Berdasarkan Skenario

Gambar 22 menunjukkan grafik hasil rata-rata jumlah *node* pengujian skenario pertama sampai skenario keenambelas. Skenario pertama sampai keenambelas mengalami perubahan yang berbeda pada tiap *level.* Berdasarkan hasil pengujian keseluruhan bahwa algoritma *Minimax* lebih unggul dari *Breadth First Search* dengan selisih sebanyak 2.724 *node.*

V. KESIMPULAN

Berdasarkan hasil pengujian performa algoritma *Minimax* dan *Breadth First Search* pada permainan *Tic-Tac-Toe*, maka didapatkan bahwa pada *level* mudah dengan ukuran papan permainan 3×3, waktu respon dan jumlah *node* yang dibutuhkan algoritma *Breadth First Search* lebih sedikit dibandingkan dengan *Minimax*. Sedangkan pada *level* sedang dengan ukuran papan permainan 5×5, waktu respon dan jumlah *node* yang dibutuhkan algoritma *Breadth First Search* lebih banyak dibandingkan dengan *Minimax*.

Pada *level* sulit dengan ukuran papan permainan 7×7, waktu respon yang dibutuhkan algoritma *Breadth First Search* lebih sedikit dibandingkan dengan *Minimax* tetapi jumlah *node* yang dibutuhkan algoritma *Breadth First Search* lebih banyak dibandingkan dengan *Minimax*. Sedangkan pada *level* sangat sulit dengan ukuran papan permainan 9×9, waktu respon dan jumlah *node* yang dibutuhkan algoritma *Breadth First Search* lebih banyak dibandingkan dengan *Minimax*.

Berdasarkan hasil tersebut, algoritma *Breadth First Search* lebih unggul dari *Minimax* pada ukuran papan permainan 3×3 dari sisi waktu respon dan jumlah *node* yang dibutuhkan. Sedangkan algoritma *Minimax* lebih unggul dari *Breadth First Search* pada ukuran papan permainan 5×5 dan 9×9 dari sisi waktu respon dan jumlah *node* yang dibutuhkan. Pada giliran pertama dalam penelitian ini, algoritma akan menelusuri jumlah *node* yang lebih besar daripada langkah selanjutnya sehingga penempatan algoritma untuk giliran pertama mempengaruhi hasil akhir parameter jumlah *node*.

DAFTAR PUSTAKA

[1] H. Eridaputra, "Penerapan DFS dan BFS dalam Pencarian Solusi Game 'Japanese River IQ Test,'" *Inst. Teknol. Bandung*, 2012.
[2] Y. Chandra, S. Sabloak, and R. Angreni, "Penerapan Algoritma Minimax dan Breadth First Search pada Permainan Othello Menggunakan Framework Phonegap," *STMIK GI MDP*, 2015.
[3] S. Nugroho and A. T. Wibowo, "Analisis dan Implementasi Algorima Minimax dengan Alpha-Beta Pruning pada Permainan Pingdoll," *Univ. Telkom*, 2010.
[4] Y. Prasetiyo, S. W. Dewa, and D. Udjulawa, "Penerapan Algoritma Breadth First Search untuk Rancang Bangun Edugame Icon Berbasis Android," *STMIK GI MDP*, 2015.
[5] R. Matthew, "Perbandingan Algoritma Greedy dan Algoritma Minimax pada Permainan Tic Tac Toe," *Inst. Teknol. Bandung*, 2016.
[6] M. Kurniawan, A. Pamungkas, and S. Hadi, "Algoritma Minimax Sebagai Pemgambil Keputusan dalam Game Tic-Tac-Toe," *Semin. Nas. Teknol. Inf. dan Multimed.*, 2016.
[7] T. A. S. P., J. Prestiliano, and S. Raymond, "Perbandingan Penerapan Algoritma Minimax Dengan Alpha-Beta Pruning pada Permainan Othello," *J. Ilm. Teknol. Inf. dan Multimed.*, 2012.
[8] Ardiansa, Susanto, A. Rahman, and Yohannes, "Perbandingan Performa Algoritma Minimax dan Negascout pada Permainan Checkers Berbasis Android," *STMIK GI MDP*, 2017.
[9] A. Ilman, "Penerapan Algoritma Minimax dengan Optimasi MTD(f) pada Permainan Catur," *Inst. Teknol. Bandung*, 2008.
[10] E. Sanjaya, T. Wonggo, and R. Angreni, "Penerapan Algoritma Minimax dan Memory-Enhanced Test Driver With Value f pada Permainan Checkers," *STMIK GI MDP*, 2015.

[11] B. Prasetiyo and M. R. Hidayah, "Penggunaan Metode Depth First Search (DFS) dan Breadth First Search (BFS) pada Strategi Game Kamen Rider Decade Versi 0.3," *Sci. J. Informatics*, vol. 1, 2014.

[12] S. Lauren, "Penerapan Algoritma DFS dan BFS untuk Permainan Wordsearch Puzzle," *Inst. Teknol. Bandung*, 2012.