Analisis Efisiensi Algoritma *Alpha Beta Pruning* dan MTD(f) pada Connect4

Lukas Tommy
Program Studi Teknik Informatika
STMIK Atma Luhur
Jl. Jend. Sudirman Selindung – Pangkalpinang
lukastommy@atmaluhur.ac.id

Eza Budi Perkasa
Program Studi Teknik Informatika
STMIK Atma Luhur
Jl. Jend. Sudirman Selindung – Pangkalpinang
ezabudiperkasa@atmaluhur.ac.id

***Abstrak*—Komputer membutuhkan kecerdasan buatan/artificial intelligence agar dapat bermain selayaknya manusia pada Connect Four/Connect4. Terdapat beberapa algoritma yang dapat diterapkan pada Connect4, namun tidak diketahui mana yang cocok. Algoritma yang cocok berarti optimal dalam memilih langkah sekaligus waktu eksekusinya tidak lambat pada kedalaman pencarian/depth yang cukup dalam. Pada penelitian ini, akan dilakukan analisis dan perbandingan antara alpha beta (AB) Pruning dan MTD(f) pada prototipe Connect4, dalam hal keoptimalan (persentase kemenangan) dan kecepatan (waktu eksekusi dan jumlah simpul daun). Pengujian dilakukan dengan menjalankan mode komputer melawan komputer dengan kondisi berbeda. Persentase yang diraih MTD(f) berdasarkan pengujian adalah menang 41,67%, kalah 41,67% dan seri 16,66%. Pada pengujian dengan depth 8, waktu eksekusi MTD(f) 35,19% lebih cepat dan mengevaluasi simpul daun 66,2% lebih sedikit dibandingkan AB Pruning. Hasil dari penelitian ini adalah MTD(f) sama optimalnya dengan AB Pruning pada prototipe Connect4, namun MTD(f) secara rata-rata lebih cepat dan mengevaluasi simpul daun lebih sedikit dibandingkan AB Pruning. Waktu eksekusi MTD(f) tidak lambat dan jauh lebih cepat dibandingkan AB Pruning pada depth yang cukup dalam.**

Kata kunci—Efisiensi Algoritma, Connect4,* Artificial Intelligence*, Alpha Beta* Pruning*, MTD(f)

I. PENDAHULUAN

Dalam implementasinya, biasanya permainan papan seperti Connect4, dimainkan oleh manusia melawan manusia lainnya ataupun komputer. Bila melawan komputer, dibutuhkan suatu kecerdasan buatan/*artificial intelligence* (AI) agar komputer dapat bermain selayaknya manusia.

Terdapat beberapa algoritma AI yang dapat diterapkan pada permainan papan seperti Connect4, yaitu antara lain *random*, *brute force*, *greedy* dan *minimax* serta variannya seperti *negamax*, *alpha beta* (AB) *Pruning* dan *negamax* AB *Pruning*.

Penelitian mengenai penerapan dan perbandingan algoritma AI pada Connect4 dan permainan papan lainnya telah pernah dilakukan sebelumnya. Penelitian [1] menerapkan algoritma *minimax* dan AB *Pruning*, sedangkan penelitian [2] menerapkan algoritma *random* dan *greedy* pada Connect4. Penelitian [3] menerapkan algoritma *negamax* dan AB *Pruning*, sedangkan penelitian [4] menerapkan algoritma *brute force* dan *negamax* AB *Pruning* pada Othello.

Terdapat permasalahan pada algoritma AI yang ada, algoritma *random*, *brute force* dan *greedy* waktu eksekusinya cepat, namun mengembalikan langkah yang kurang optimal bila dibandingkan varian *minimax* [2,4]. Varian *minimax* mengembalikan langkah yang optimal, namun waktu eksekusinya lama jika kedalaman pencarian/*depth* (d) cukup dalam [4].

Untuk mengatasi permasalahan di atas, perlu dilakukan analisis dan perbandingan algoritma dalam menentukan yang cocok untuk diimplementasikan pada Connect4. Algoritma yang cocok berarti tidak hanya optimal, tapi juga waktu eksekusinya tidak lambat pada *depth* yang cukup dalam. Algoritma yang dipilih adalah AB *Pruning* dan MTD(*f*) yang kemudian dianalisis dan dibandingkan dalam hal persentase kemenangan, waktu eksekusi dan jumlah simpul daun/ *leaf node* (LN) yang dievaluasi. AB *Pruning* dipilih karena optimal layaknya *minimax* namun lebih cepat[1,3]. *Negamax* AB *Pruning* tidak dipilih karena mengevaluasi LN yang sama dengan AB *Pruning* meskipun implementasinya lebih sederhana [5]. MTD(*f*) dipilih karena diklaim lebih sederhana, lebih efisien, dan secara rata-rata lebih baik dibandingkan dengan algoritma terdahulu [6,7,8].

Penelitian mengenai analisis dan perbandingan algoritma AB *Pruning* dan MTD(*f*) ini memiliki beberapa keterbatasan. AB *Pruning* yang digunakan adalah AB *Pruning* standar. Papan Connect4 yang digunakan berukuran standar, yaitu 7 kolom dan 6 baris. Parameter yang diuji adalah nilai evaluasi setiap langkah, persentase kemenangan, jumlah LN yang dievaluasi dan waktu eksekusi dari algoritma. *Depth* yang digunakan dalam pengujian adalah 5 sampai dengan 10. Penelitian hanya mencakup *mode* komputer melawan komputer, dimana kedua komputer menggunakan *depth* yang sama.

Dengan adanya analisis dan perbandingan kedua algoritma, diharapkan dapat menentukan algoritma yang paling baik untuk diterapkan pada Connect4 dengan *depth* cukup dalam.

II. METODOLOGI PENELITIAN

Metode studi pustaka merupakan langkah awal dalam penelitian ini, dimana landasan teori terkait analisis algoritma AB *Pruning* dan MTD(*f*) pada Connect4 pada beberapa literatur dan referensi lainnya dipelajari.

Langkah kedua dalam penelitian ini adalah metode observasi. Metode observasi dilakukan dengan memainkan permainan Connect4 yang tersedia secara *online* maupun *offline*, kemudian menganalisa algoritma yang digunakan.

Analisis dan perbandingan algoritma *AB pruning* dan MTD(*f*) akan dilakukan pada prototipe Connect4. Kedua algoritma digunakan oleh komputer dalam memilih lubang. Parameter yang dianalisis dan dibandingkan adalah keoptimalan (nilai evaluasi) dari lubang yang dipilih dan jumlah kemenangan, serta waktu eksekusi dan jumlah simpul daun yang dievaluasi.

Environtment yang digunakan dalam pengujian adalah laptop dengan prosesor intel *dual-core* @2.4GHz, memori 4GB dan sistem operasi Windows 7 64-bit.

Pengujian dilakukan dengan menjalankan prototipe Connect4 dengan *mode* komputer X (AB *pruning*) melawan komputer O (MTD(*f*)) dengan *depth* yang sama untuk kedua algoritma. *Depth* yang diuji adalah 5 sampai 10 dan pada setiap *depth* dilakukan pengujian dengan giliran pertama komputer X dan giliran pertama komputer O.

Setiap kondisi diuji sebanyak 10 kali dan waktu eksekusi pada setiap langkah yang dipilih untuk dibandingkan adalah rata-rata dari 10 kali pengujian tersebut. Berdasarkan pengujian yang telah dilakukan, didapatkanlah informasi berupa waktu eksekusi rata-rata, jumlah LN yang dievaluasi dan persentase kemenangan pada setiap kondisi. Informasi ini kemudian dianalisa untuk menentukan mana yang paling baik di antara kedua algoritma.

Hasil dari penelitian ini adalah dapat ditentukannya algoritma mana yang paling cepat, paling optimal serta mengevaluasi simpul daun paling sedikit pada permainan Connect4. Algoritma yang cocok juga dapat ditentukan untuk diterapkan pada Connect4 dengan *depth* cukup dalam.

A. *Rancangan Algoritma*

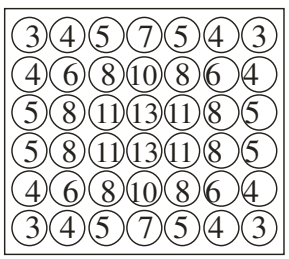
Pseudocode dari algoritma yang dirancang untuk *mode* komputer melawan komputer adalah sebagai berikut.

```
Pilih siapa giliran pertama
Pilih kedalaman pencarian
giliran_ke ← 1
LN_X ← 0; LN_O ← 0; Pass_O ← 0;
Urutan ←[3,4,2,5,1,6,0]
Tampilkan papan Connect4 kosong

WHILE gameover=False DO
  IF giliran="X" THEN
    Hitung waktu mulai
    Lakukan fungsi AB Pruning kedalaman n
    Perbarui LN X
    Pilih lubang dan cetak nilai evaluasi
    waktu_eksekusi← waktu sekarang-waktu
mulai
    giliran←"O"
  ELSE
    Hitung waktu mulai
    Lakukan fungsi MTD(f) kedalaman n
    Perbarui LN_O; Perbarui Pass_O
    Pilih lubang dan cetak nilai evaluasi
    waktu_eksekusi← waktu sekarang-waktu
mulai
    giliran←"X"
  END IF

  Perbarui papan Connect4
  Cetak waktu_eksekusi
  Periksa segmen H,V,D pada papan
    IF koin X terhubung >=4 THEN
      pemenang←"X"
      gameover←True
    ELSE IF koin O terhubung >=4 THEN
      pemenang←"O"
      gameover←True
    END IF

  giliran_ke+=1
  IF giliran_ke > 42 THEN
    pemenang←"Seri"
    gameover←True
  END IF
END WHILE

Cetak giliran_ke
Cetak pemenang, LN X, LN O, Pass O;
```

B. *Rancangan Fungsi Evaluasi*

Algoritma AI membutuhkan fungsi evaluasi sehingga komputer dapat memberikan penilaian terhadap keadaan papan permainan saat itu. Hal ini dilakukan agar komputer dapat memilih lubang-lubang terbaik dari lubang yang ada untuk memenangkan permainan.

Pada papan Connect4 standar, terdapat 69 segmen kemenangan pada papan, yaitu 24 segmen horizontal (H), 21 segmen vertikal (V) dan 24 segmen diagonal (D). Setiap segmen terdiri atas 4 lubang dan setiap lubang diberi bobot berdasarkan banyaknya segmen yang dapat dibentuk dari lubang tersebut [9], seperti ditunjukkan pada Gambar 1. Komputer akan mempertimbangkan untuk menelusuri lubang dengan bobot terbesar terlebih dahulu.



Gambar 1. Pemberian Bobot Lubang Connect4

Terdapat beberapa aturan dalam mengevaluasi segmen [10]:

1) 0 untuk lubang kosong atau terdapat koin berbeda;
2) 1 untuk satu koin sejenis;
3) 10 untuk dua koin sejenis;
4) 100 untuk tiga koin sejenis; dan
5) *Infinity* (∞) untuk empat koin sejenis (menang).

Nilai dari segmen akan menjadi positif jika segmen yang dievaluasi adalah koin pemain MAX (X) atau negatif jika segmen yang dievaluasi adalah pemain MIN (O). Nilai yang dikembalikan dari fungsi evaluasi yang dirancang adalah jumlah nilai evaluasi semua segmen pemain dikurangi jumlah nilai evaluasi semua segmen lawan.

C. Rancangan Pengujian

Pengujian dilakukan dengan menjalankan *mode* komputer melawan komputer pada prototipe Connect4 dengan berbagai kondisi yang berbeda. Kondisi permainan yang akan diuji berjumlah 12 dengan rincian sebagai berikut.

1) Giliran pertama adalah X (Gil1X) dengan *depth* (d) 5 sampai dengan 10.
2) Giliran pertama adalah O (Gil1O) dengan *depth* (d) 5 sampai dengan 10.

Setiap kondisi permainan akan diuji sebanyak 10 kali untuk mencari waktu eksekusi rata-rata pada setiap langkah. Pada pengujian yang akan dilakukan, informasi-informasi yang diperoleh selama pengujian dicatat. Informasi tersebut berupa lubang yang dipilih setiap giliran dan nilai evaluasinya, waktu eksekusi setiap giliran, waktu eksekusi rata-rata, jumlah LN dan total waktu eksekusi.

Total waktu eksekusi (T) adalah jumlah waktu eksekusi seluruh langkah, dapat dicari dengan rumus:

$$T = \text{waktu eksekusi}_1 + \text{waktu eksekusi}_2 + \cdots + \text{waktu eksekusi}_n \quad (1)$$

Waktu rata-rata (t) merupakan waktu eksekusi rata-rata setiap langkah, dapat dicari dengan rumus:

$$t = \frac{T}{\text{banyak langkah}} \quad (2)$$

Informasi dari pengujian ini kemudian dirangkum dan dikelompokkan berdasarkan siapa giliran pertama dan berdasarkan kedalaman. Informasi yang dirangkum berupa kedalaman, waktu eksekusi rata-rata, jumlah LN, persentase kecepatan O dibanding X, persentase LN O banding X dan hasil akhir permainan. Informasi-informasi ini kemudian dianalisis dan dibandingkan untuk menentukan manakah yang paling cepat dan paling optimal di antara kedua algoritma, untuk diterapkan pada permainan Connect4.

Persentase (%) kecepatan O:X menunjukkan persentase perbandingan waktu eksekusi rata-rata antara komputer O dengan X. Persentase akan diawali dan diakhiri dengan tanda kurung jika O lebih lambat. Persentase kecepatan dapat dicari dengan rumus:

$$\%\text{Kecepatan O:X} = \frac{t_X - t_O}{t_X} \times 100\% \quad (3)$$

Persentase (%) LN O:X menunjukkan persentase perbandingan jumlah simpul daun yang dievaluasi antara komputer O dengan X. Persentase akan diawali dan diakhiri dengan tanda kurung jika LN O lebih banyak. Persentase LN dapat dicari dengan rumus:

$$\%\text{LN O:X} = \frac{LN_X - LN_O}{LN_X} \times 100\% \quad (4)$$

Persentase (%) kemenangan menunjukkan besarnya persentase hasil akhir permainan yaitu X menang, O menang atau seri dari jumlah pengujian yang dilakukan. Persentase kemenangan dapat dicari dengan rumus:

$$\%\text{Kemenangan} = \frac{\text{Jumlah Kemenangan}}{\text{Jumlah Pengujian}} \times 100\% \quad (5)$$

III. Hasil dan Pembahasan

A. Proses Pengujian

Pengujian algoritma AB *Pruning* dan MTD(*f*) pada prototipe Connect4 dilaksanakan dengan 12 kondisi berbeda. Kondisi pengujian berupa komputer dengan algoritma mana yang mulai pertama/ giliran satu (Gil1) dan *depth* (d) yang beragam. Algoritma AB *Pruning* diimplementasikan pada komputer X, sedangkan algoritma MTD(*f*) diimplementasikan pada komputer O. Pada setiap pengujian, kedalaman yang sama diterapkan pada kedua algoritma. Parameter yang diuji adalah waktu eksekusi, jumlah simpul daun, nilai evaluasi lubang yang dipilih dan jumlah kemenangan yang diraih. Setiap kondisi diuji masing-masing sebanyak 10 kali dan waktu eksekusi yang dipilih adalah rata-ratanya. Dari hasil pengujian yang dilakukan, dapat ditentukan mana yang paling cepat dan paling optimal dari kedua algoritma yang diuji untuk diimplementasikan pada permainan Connect4 dengan *depth* cukup dalam.

B. Hasil Pengujian

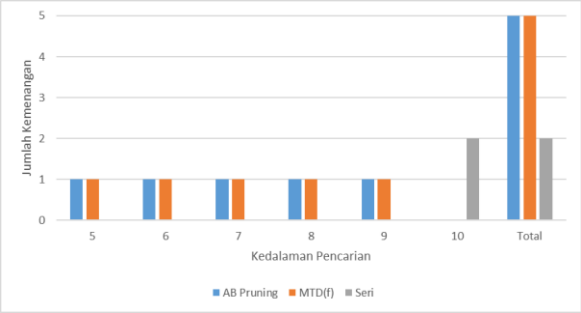
Rangkuman hasil pengujian setiap kedalaman/ *depth* ditunjukkan pada Tabel I. Pada *depth* 8 O(MTD(*f*)) begitu juga X(AB *Pruning*) menang sekali. Waktu eksekusi rata-rata setiap langkah X 2,3185 detik dan O 1,5027 detik sedangkan Jumlah *leaf node* (LN) X sebanyak 218.777 dan O sebanyak 73.957. Waktu eksekusi O 35,19% lebih cepat dan mengevaluasi *leaf node* (LN) 66,2% lebih sedikit dibandingkan X. Tanda kurung akan diberikan pada persentase kecepatan jika O lebih lambat dari X. Jumlah *pass*/ pemanggilan fungsi ABMemori oleh O sebanyak 715 kali. Demikian juga halnya dengan *depth* lainnya.

Perbandingan jumlah kemenangan setiap kedalaman dirangkum dalam bagan yang ditunjukkan pada Gambar 2, dimana X menang 5 kali, O menang 5 kali dan seri 2 kali

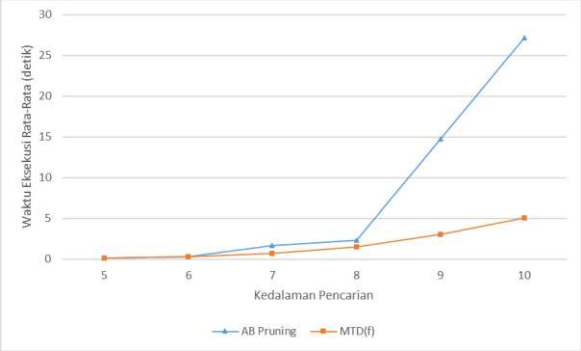
Perbandingan waktu eksekusi rata-rata setiap kedalaman dirangkum dalam grafik yang ditunjukkan pada Gambar 3. Perbandingan jumlah LN yang dievaluasi kedua algoritma pada setiap kedalaman dirangkum dalam grafik yang ditunjukkan pada Gambar 4. Perbedaan waktu eksekusi dan jumlah LN kedua algoritma meningkat dengan semakin dalamnya *depth* dan terlihat jauh berbeda pada *depth* 9 dan 10.

TABEL I. RANGKUMAN HASIL PENGUJIAN SETIAP *DEPTH*

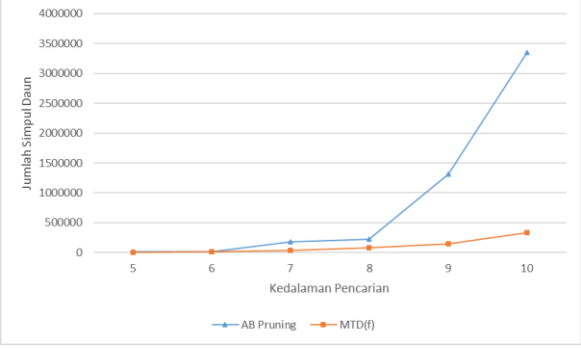
Depth	Waktu(detik)		% Kecepatan O:X	Jumlah LN		% LN O:X	Pass MTD(*f*)	Hasil Akhir
	X	O		X	O			
5	0,1317	0,1089	17,28%	12242	5451	55,47%	201	X(1), O(1)
6	0,3083	0,3089	(0,19%)	17885	8705	51,33%	255	X(1), O(1)
7	1,6981	0,6767	60,15%	175759	36906	79,00%	456	X(1), O(1)
8	2,3185	1,5027	35,19%	218777	73957	66,20%	715	X(1), O(1)
9	14,7149	3,0543	79,24%	1311365	143557	89,05%	478	X(1), O(1)
10	27,1850	5,0731	81,34%	3352837	327609	90,23%	590	Seri(2)
Jumlah Hasil Akhir				**X menang 5 kali, O menang 5 kali, Seri 2 kali**				



Gambar 2. Perbandingan Jumlah Kemenangan Setiap *Depth*



Gambar 3. Perbandingan Waktu Eksekusi Setiap *Depth*



Gambar 4. Perbandingan Jumlah LN Setiap *Depth*

C. *Analisa dan Pembahasan*

MTD(*f*) memilih lubang-lubang yang sama dengan AB *Pruning* karena kedua algoritma sama-sama varian dari pencarian *minimax* dan menggunakan fungsi evaluasi yang sama.

Pada *depth* 6, waktu eksekusi algoritma MTD(*f*) sedikit lebih lambat dari AB *Pruning* meskipun MTD(*f*) mengevaluasi LN lebih sedikit. Hal ini disebabkan kompleksitas kode algoritma MTD(*f*) lebih besar dibandingkan kode AB *Pruning* standar. Waktu eksekusi dari MTD(*f*) akibatnya melambat sehingga diperlukan perbedaan jumlah LN yang lebih banyak lagi.

Persentase perbandingan LN antara O dengan X pada *depth* 7 lebih besar dibandingkan pada *depth* 8. Anomali ini terjadi karena jumlah *pass* MTD(*f*) pada *depth* ganjil secara rata-rata lebih sedikit dibandingkan pada *depth* genap. Jumlah LN yang dievaluasi akan berkurang dengan semakin sedikitnya jumlah *pass*. Anomali ini juga terjadi akibat pengurangan jumlah faktor percabangan AB *Pruning* sebanyak akar kuadrat pada kedalaman genap. Hal ini menyebabkan *pruning* lebih sering terjadi sehingga jumlah LN yang dievaluasi oleh AB *Pruning* berkurang.

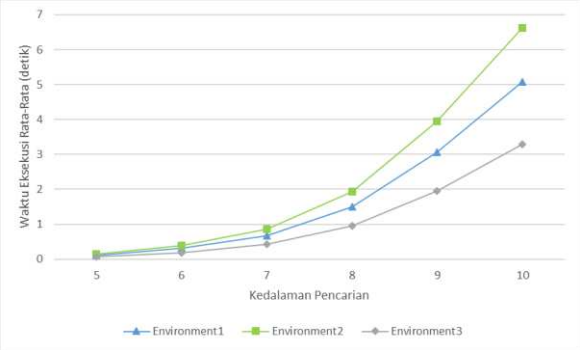
Pemberian bobot lubang seperti Gambar 1 membuat komputer menelusuri lubang dengan urutan bobot terbesar ke bobot terkecil. Jumlah LN menjadi berkurang karena nilai *minimax* dapat ditemukan lebih awal yang menyebabkan *pruning* terjadi semakin awal .

Pada Connect4, hasil akhir permainan selain dipengaruhi oleh keoptimalan algoritma juga dipengaruhi oleh faktor siapa yang mulai pertama. Komputer dengan giliran pertama memiliki kesempatan lebih besar untuk memenangkan permainan. Waktu eksekusi dan jumlah LN juga dipengaruhi oleh faktor siapa yang mulai pertama.

MTD(*f*) secara rata-rata lebih cepat dan mengevaluasi lebih sedikit LN dibandingkan AB *Pruning*. Hal ini karena MTD(*f*) menggunakan pencarian jendela nol dan tabel tranposisi. Pencarian jendela nol mengurangi ruang pencarian dengan *pruning* yang lebih awal dibandingkan pencarian

jendela lebar AB *Pruning*. Tabel transposisi digunakan untuk menyimpan dan mengambil nilai evaluasi dari keadaan papan sehingga keadaan papan yang sama tidak perlu dievaluasi ulang. Penambahan fungsi pencerminan pada tabel transposisi menyebabkan komputer tidak perlu mengevaluasi keadaan papan yang sama dengan pencerminan dari keadaan papan sebelumnya. Pada *depth* cukup dalam (9 dan 10) waktu eksekusi MTD(*f*) tidak lambat dan jauh lebih cepat dari AB *Pruning*.

Pengujian juga dilakukan pada dua *environment* berbeda. *Environtment* kedua adalah laptop dengan prosesor *dual-core* @2.0GHz, memori 3GB dan sistem operasi Windows 7 32-bit. *Environtment* ketiga adalah PC (*personal computer*) dengan prosesor AMD *octa-core* @4.4GHz, memori 8GB dan sistem operasi Windows 10 64-bit. Perbandingan waktu eksekusi MTD(*f*) setiap kedalaman dengan *environment* berbeda dirangkum dalam grafik yang ditunjukkan pada Gambar 5. Berdasarkan Gambar 5, diketahui bahwa spesifikasi *hardware* yang digunakan dalam pengujian juga mempengaruhi waktu eksekusi algoritma.



Gambar 5. Waktu Eksekusi MTD(f) Setiap *Depth* Dengan *Environment* Berbeda

Berdasarkan pengujian dengan 12 kondisi berbeda yang telah dilakukan, persentase X menang sebesar 41,67%, persentase O menang sebesar 41,67%, sedangkan persentase permainan berakhir seri sebesar 16,66%.

Algoritma yang cocok untuk Connect4 dari dua algoritma yang diuji adalah MTD(*f*). Hal ini dikarenakan MTD(*f*) optimal seperti AB *Pruning*, namun waktu eksekusinya tidak lambat dan jauh lebih cepat jika *depth* cukup dalam.

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil analisis yang dilakukan terhadap permasalahan, rumusan masalah, rancangan penelitian serta pengujian dalam penelitian ini didapatkan kesimpulan berikut.

1) Algoritma MTD(*f*) sama optimalnya dengan AB *Pruning* pada permainan Connect4. Berdasarkan pengujian dengan 12 kondisi berbeda, persentase yang diraih MTD(*f*) adalah menang 41,67%, kalah 41,67% dan seri 16,66%.
2) Waktu eksekusi algoritma MTD(*f*) secara rata-rata lebih cepat dibandingkan AB *Pruning* pada permainan Connect4. Pada *depth* 6, waktu eksekusi MTD(*f*) lebih lambat 0,19% dibandingkan AB *Pruning*. Pada *depth* 5, 7, 8, 9 dan 10, waktu eksekusi MTD(*f*) lebih cepat berturut-turut sebesar 17,28%, 60,15%, 35,19%, 79,24% dan 81,34% dibandingkan AB *Pruning*.
3) Algoritma MTD(*f*) mengevaluasi LN lebih sedikit dibandingkan AB *Pruning* pada permainan Connect4. Pada *depth* 5, 6, 7, 8, 9 dan 10, komputer MTD(*f*) mengevaluasi simpul daun berturut-turut 55,47%, 51,33%, 79%, 66,2%, 89,05% dan 90,23% lebih sedikit dibandingkan komputer AB *Pruning*.
4) Algoritma MTD(*f*) cocok untuk diterapkan pada permainan Connect4. MTD(*f*) sama optimalnya dengan AB *Pruning*, namun waktu eksekusinya tidak lambat dan jauh lebih cepat jika *depth* cukup dalam.

B. Saran

Berikut adalah saran yang dapat diberikan sebagai bahan pertimbangan penelitian selanjutnya.

1) Membandingkan algoritma MTD(*f*) dengan algoritma yang lebih baru dan lebih baik dari AB *Pruning* standar.
2) Perbaikan fungsi evaluasi Connect4 yang telah ada, sehingga AI dapat lebih akurat dan optimal dalam memilih lubang.
3) Optimasi algoritma pencarian untuk mempercepat waktu eksekusi dengan menggunakan *database*. Nilai evaluasi tiap segmen pada papan dihitung terlebih dahulu dan disimpan dalam bentuk tabel *hash* pada *database*.
4) Menggunakan *environment* dengan spesifikasi *hardware* yang tinggi untuk mempercepat waktu eksekusi algoritma.

DAFTAR PUSTAKA

[1] Lestari, J. dan Winata, A. 2012, Implementasi Algoritma Minimax dengan Optimasi Alpha-Beta Pruning pada Aplikasi Permainan Connect Four, *Prosiding Seminar Nasional Multidisiplin Ilmu (SeNMI)*, Jakarta, 8 Desember.

[2] Sarhan, A.M., Shaout, A. dan Shock, M. 2009, Real - Time Connect 4 Game Using Artificial Intelligence, *Journal of Computer Science*, vol. 5, no. 4, hal. 283–289.

[3] Handayani, M.S., Arisandi, D. dan Sitompul, O.S. 2012, Rancangan Permainan Othello Berbasis Android Menggunakan Algoritma Depth - First Search, *Jurnal Dunia Teknologi Informasi*, vol. 1, no. 1, hal. 28–34.

[4] Siswanto, Laurentsia, dan Anif, M. 2014, Pengembangan Aplikasi Permainan Othello dengan Negamax Alpha Beta Pruning dan Brute Force, *Seminar Nasional Sistem Informasi Indonesia (SESINDO)*, Surabaya, 22 September.

[5] Millington, I. dan Funge, J. 2009, *Artificial Intelligence for Games*, Ed. 2. Morgan Kaufmann, Burlington.

[6] Gunawan, Kristian, Y. dan Andika, H. 2009, Game Playing untuk Othello dengan Menggunakan Algoritma Negascout dan MTDF, *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)*, Yogyakarta, 20 Juni.

[7] Plaat, A. 1997, Aske Plaat MTD(f) - a new chess algorithm, http://people.csail.mit.edu/plaat/mtdf.html, diakses tgl 29 Februari 2016.

[8] Shibahara, K., Inui, N. and Kotani, Y. 2005, Adaptive Strategies of MTD-f for Actual Games, *Proceedings of the 2005 Symposium on Computational Intelligence and Games (CIG05), Essex, April 4-6.*

[9] Stenmark, M. 2005, Synthesizing Board Evaluation Functions for Connect4 using Machine Learning Techniques, *thesis*, M.C.S. Department Computer Science, Østfold University College, Halden.

[10] Rivest, R. L. 1988, Game Tree Searching by Min / Max Approximation, *Artificial Intelligence*, vol. 34, hal. 77–96.