Jurnal Bakti Nusa, P. ISSN : 2721-2181
Vol. 1, No. 2. 37 – 48 Oktober 2020
Doi: ----------

IMPLEMENTASI *MULTITHREADING* PADA ALGORITMA MINIMAX

IMPLEMENTATION OF MULTITHREADING ON MINIMAX ALGORITHM

Lalu Imam Adiguna Rinjani[1], Lalu A. Syamsul Irfan Akbar[2], A. Sjamsjiar Rachman[3]
[1]Jurusan Teknik Elektro Fakultas Teknik Universitas Mataram, Nusa Tenggara Barat, Indonesia
Email :imamrinjani2@gmail.com[1],irfan@unram.ac.id[2],asrachman@unram.ac.id[3]

ABSTRAK

Algoritma minimax merupakan salah satu dari algoritma pencarian yang mencari cabang dengan nilai tertinggi pada suatu pohon pencarian, dimana pohon pencarian adalah pohon yang terbuat dari kemungkinan – kemungkinan yang dapat terjadi. Kekurangan dari algoritma ini adalah semakin besarnya pohon pencarian maka waktu eksekusi yang dibutuhkan dalam melakukan pencarianpun meningkat. Teknik *multithreading* merupakan suatu teknik yang membagi sebuah tugas besar menjadi beberapa tugas kecil dan dikerjakan secara bersamaan. Teknik *multithreading* akan diimplementasikan pada algoritma minimax untuk mengurangi waktu eksekusi pencarian cabang dengan nilai tertinggi. Penelitian ini menggunakan bahasa pemrograman java dengan NetBeans IDE 8.0.2. dan komputer personal. Hasil penelitian menunjukkan algoritma minimax dengan implementasi *multithreading* lebih efisien dibandingkan dengan algoritma minimax dengan *Speed Up* yang dihasilkan pada *multithreading* dengan 2 *thread* adalah 1.31 pada kedalaman maksimal 4, 1.41 pada kedalaman maksimal 5, dan 1.52 pada kedalaman maksimal 6. Sedangkan *Speed Up* yang dihasilkan pada *multithreading* dengan 3 *thread* adalah 1.35 pada kedalaman maksimal 4, 1.42 pada kedalaman maksimal 5, dan 1.55 pada kedalaman maksimal 6. *Gradien Speed Up* yang dihasilkan dari 2 *thread* pada kedalaman 4 dan 5 adalah 0.1 dan pada kedalaman 5 dan 6 adalah 0.11. Sedangkan *Gradien Speed Up* yang dihasilkan dari 3 *thread* pada kedalaman 4 dan 5 adalah 0.07 dan pada kedalaman 5 dan 6 adalah 0.13

Kata Kunci : Algoritma Minimax, Multithreading, Speed Up

ABSTRACT

Minimax algorithm is one of the searching algorithm that search a branch with the highest value on a searching tree, a searching tree is a tree that made of any possibility that might happen. The lack of this algorithm is that the bigger the tree than the time it consume to search is bigger too. Multithreading technique is a technique that divide a big job into a few small jobs, the few small job will be done at the same time. Multithreading technique will be implemented on minimax algorithm to reduce the execution time to search the branch with the highest value. This research use java programming language with NetBeans IDE 8.0.2. and personal computer. The result of this research is that the minimax algorithm with multithreading implementation is more efficient than minimax algorithm with the resulting Speed Up in multithreading with 2 thread are 1.31 with 4 as the maximum depth, 1.41 with 5 as the maximum depth, and 1.52 with 6 as the maximum depth. Gradien Speed Up for 3 thread toward 2 thread with the maximum depth of 4 and 5 is 0.7, meanwhile with the maximum depth of 5 and 6 is 1.2. On the depth of 4 and 5 the rise of the Speed Up on 2 thread is bigger than 3 thread, meanwhile on the depth of 5 and 6 the rise of the Speed Up on 3 thread is bigger than 2 thread. This thing can be seen by looking at the resulting Gradien Speed Up from 2 thread on the depth of 4 and 5 which is 0.1 and on the depth of 5 and 6 which is 0.11. Meanwhile the resulting Gradien Speed Up from 3 thread on the depth of 4 and 5 is 0.07 and on the depth of 5 and 6 is 0.13.

Keywords: Minimax Algorithm, Multithreading, Speed Up

PENDAHULUAN

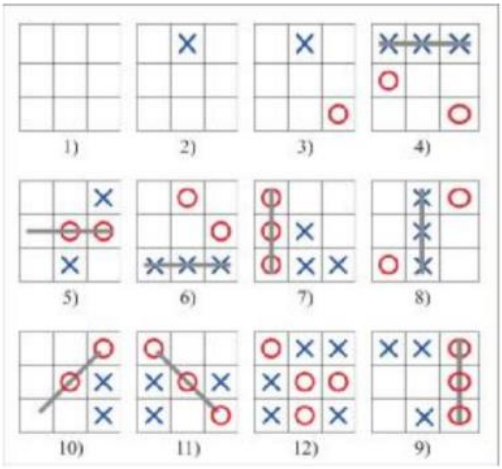
Perkembangan teknologi yang sangat pesat menyebabkan kebutuhan manusia akan teknologipun berkembang. Salah satu teknologi yang saat ini menjadi kebutuhan manusia adalah perangkat lunak. Dalam perangkat lunak umumnya dibutuhkan adanya algoritma pencarian. Salah satu contohnya adalah perangkat lunak untuk sistem informasi, untuk mempermudah pencarian informasi dalam sistem informasi dibutuhkan fungsi pencarian. Perangkat lunak, menggunakan beberapa macam metode pencarian, seperti metode pencarian vertikal (*depth first search*) dan metode pencarian horizontal (*breadth first search*).

Salah satu algoritma pencarian yang menggunakan metode *depth first search* adalah algoritma minimax, dan algoritma ini dapat diterapkan pada permainan Tic Tac Toe. Minimax meminimalisir kerugian terbesar atau mengoptimalkan keuntungan terkecil, dengan cara memprediksi gerakan lawan sebelum melangkah (Nuzulla dan Solichin, 2012). Algortiam minimax memiliki kelemahan ketika algoritma digunakan untuk memroses data dengan masukan yang besar, karena proses untuk membangun pohon pencarian dengan algoritma ini memiliki kompleksitas algoritma eksponensial untuk itu memerlukan waktu yang lebih lama untuk menemukan solusi yang terbaik (Kosasi, 2014).

Salah satu teknik untuk mengurangi waktu eksekusi algoritma minimax tersebut adalah dengan membagi bagian-bagian tugas yang ada dan memprosesnya secara bersamaan, dan teknik tersebut disebut dengan teknik *multithreading* (Millington dan Funge, 2012).

Dalam tugas akhir ini akan dikembangkan suatu permainan Tic Tac Toe 5x5 dengan algoritma pencarian menggunakan algoritma minimax yang sudah dimodifikasi dengan teknik *multithreading*. Keluaran dari perangkat lunak ini berupa waktu pencarian cabang maksimal.

Tic Tac Toe. Tic Tac Toe adalah permainan dimana dua pemain secara bergantian menjalankan gilirannya untuk membentuk sebaris, sekolom, atau segaris diagonal dengan 3 tanda ‘X’ atau ‘O’ dalam papan permainan yang terdiri dari 3 x 3 kotak (Nuzulla dan Solichin, 2012).

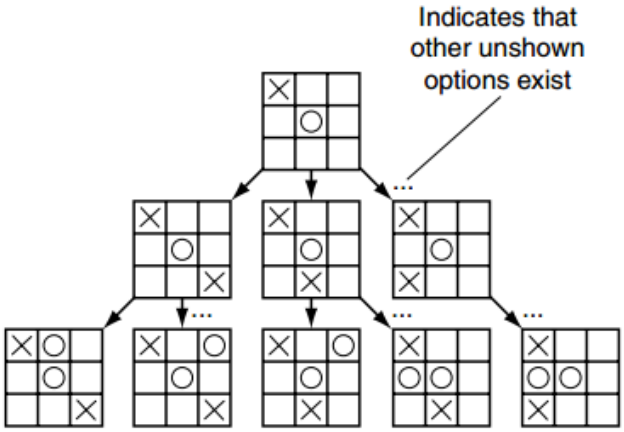


Gambar 1 Kondisi Kemenangan Tic Tac Toe 3 x 3 (Nuzulla dan Solichin, 2012)

Tata cara permainan Tic Tac Toe adalah sebagai berikut:

a) Pada umunya pemain yang mendapatkan giliran pertama adalah pemain dengan tanda ‘X’, tetapi pemain dengan giliran pertama juga dapat memilih antara tanda ‘X’ atau ‘O’ sebagai tandanya.
b) Permainan Tic Tac Toe dilakukan pada sebuah papan yang memiliki 9 kotak terdiri dari 3 baris dan 3 kolom, pemain akan secara bergiliran mengisi tanda mereka di dalam kotak yang masih kosong.
c) Tujuan dari setiap pemain adalah mengisi 3 kotak secara berurutan dengan tanda mereka baik secara vertikal, horizontal atau diagonal.
d) Permainan akan berakhir jika salah satu pemain berhasil mencapai tujuan mereka atau jika seluruh kotak sudah terisi semua dengan tanda, jika hal ini terjadi dan tidak ada pemenangnya , maka permainan dianggap seri.

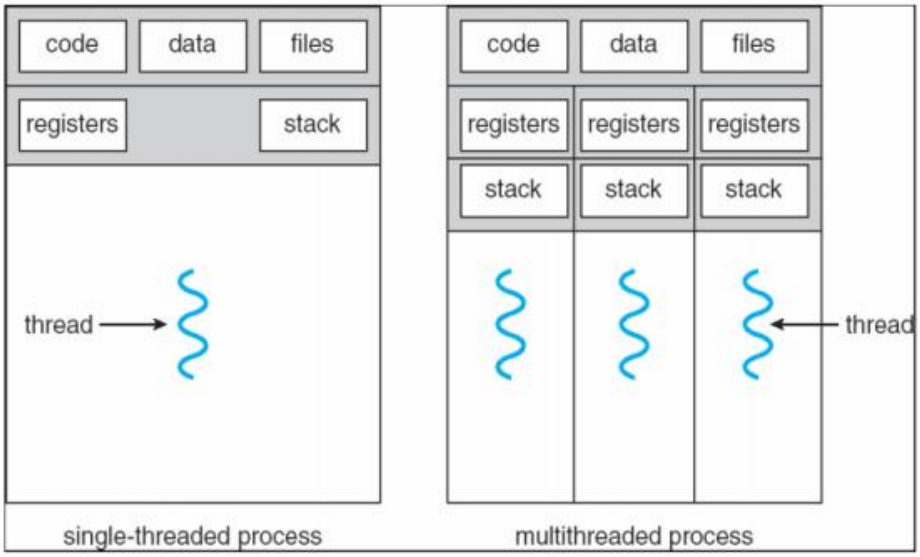
Algoritma Minimax. Algoritma minimax (*Minimize the Maximum Possible Loss*) merupakan salah satu algoritma yang diterapkan pada permainan 2 (dua) orang pemain, seperti : tic-tac-toe, checkers, chess, dan go. Algoritma minimax juga merupakan suatu algoritma berulang (*recursive algorithm*), dalam setiap perulangan algoritma ini akan menberikan *score* berdasarkan gerakan yang dilakukan. Hasil akhir algoritma ini adalah sebuah gerakan yang memiliki *score* tertinggi (Millington dan Funge, 2012).



Gambar 2 Algoritma Minimax pada Permainan Tic Tac Toe (Millington dan Funge, 2012)

Pada gambar 2 di atas dapat dilihat bahwa algoritma minimax pada permainan Tic Tac Toe akan melakukan simulasi permainan dari awal sampai kedalaman yang telah ditentukan atau kondisi yang diinginkan telah terpenuhi. Hal ini dilakukan agar algoritma mendapatkan gerakan yang memiliki keuntungan terbesar dan kerugian terkecil.

Multithreading. *Multithreading* adalah suatu teknik untuk meningkatkan proses perangkat lunak yaitu dengan cara membagi bagian-bagian tugas yang ada dan memprosesnya secara bersamaan (Microsystem, 1994).

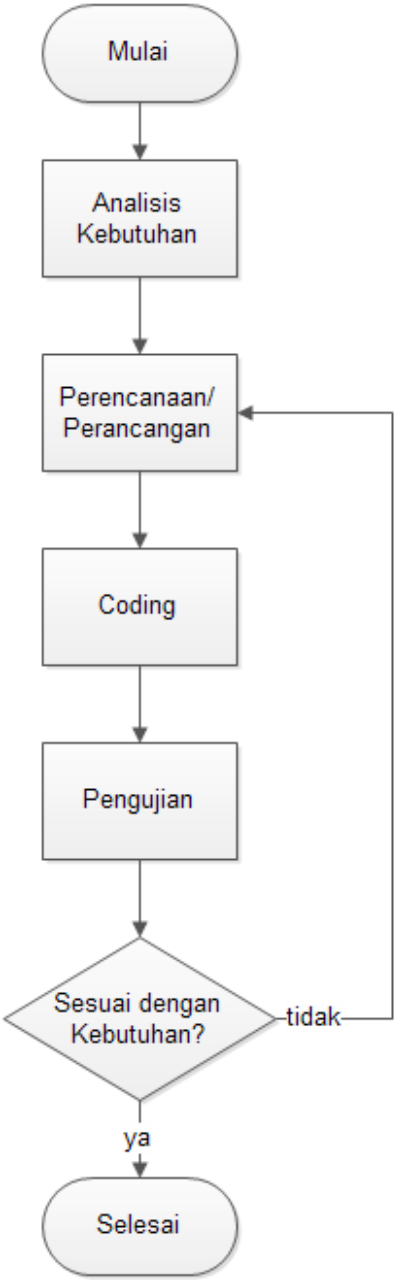


Gambar 3 Perbedaan *Single-Threading* dengan *Multi-Threading* (Silberschatz, 2005)

Pada Gambar 3 dapat dilihat perbedaan antara *single-thread* dengan *multi-thread. Single-thread* merupakan pengerjaan tugas yang dilakukan oleh sebuah *thread,* sedangkan *multi-thread* merupakan pengerjaan tugas, dimana tugas tersebut dibagi menjadi beberapa bagian dan dikerjakan oleh beberapa *thread.*

METODELOGI PENELITIAN

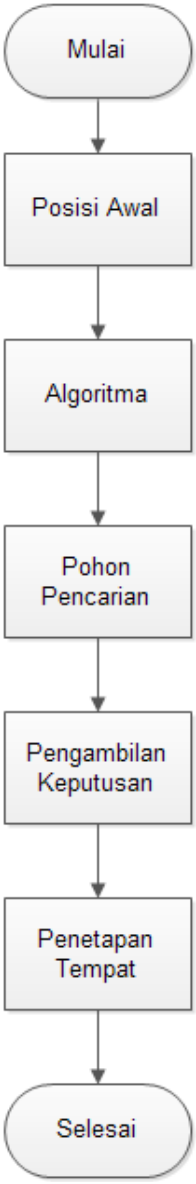
Diagram Alir Penelitian. Diagram dari metode penelitian untuk megaplikasikan multithreading pada algoritma minimax dapat dilihat pada Gambar 4.



Gambar 4 Diagram Alir Penelitian

Pada Gambar 4 dapat dilihat bahwa dalam penelitian ini tahap pertama yang dilakukan adalah Analisis kebutuhan. Yaitu tahap dimana mengumpulkan data yang nantinya diperlukan sebagai masukan dalam penelitian ini. Tahap kedua yaitu Perancangan atau Design. Dalam tahap ini terdapat pemodelan proses dan data. Tahap ketiga yaitu coding. Tahap ini mengimplementasikan rancangan sistem sehingga program yang dihasilkan sesuai dengan rancangan. Tahap keempat yaitu tahap pengujian sistem. Pengujian sistem dilakukan untuk menguji apakah sistem tersebut sesuai dengan kebutuhan. Apabila sistem yang dihasilkan telah sesuai ketentuan dan kebutuhan pengguna maka akan berlanjut ketahap selanjutnya. Tetapi jika sistem tidak sesuai, maka akan kembali ke tahap analisis kebutuhan.

Diagram Alir Pengambilan Keputusan. Gambar 5 menggambarkan bagaimana pengambilan keputusan dalam perangkat lunak yang akan dibuat dalam penelitian ini.

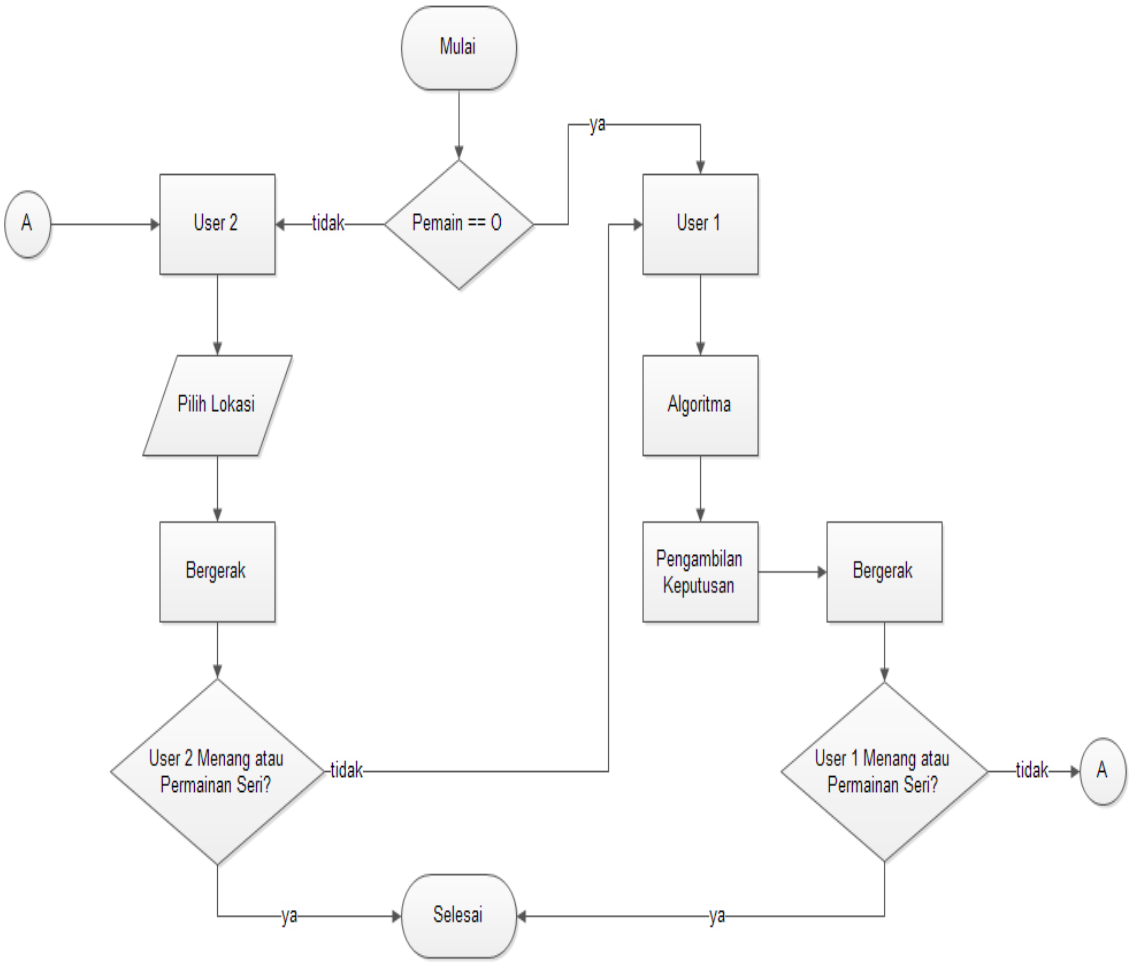


Gambar 5 Diagram Alir Pengambilan Keputusan

Pada Gambar 5 dapat dilihat bahwa dalam pengambilan keputusan dibagi menjadi beberapa bagian yaitu :

1. Posisis Awal : Tahap pertama yang dilakukan adalah pengambilan data status permainan, dimana data ini melingkupi letak token komputer dan letak pion pemain.
2. Algoritma : Tahap kedua adalah pemanggilan algoritma minimax.
3. Pohon Pencarian : Tahap ketiga adalah tahap pembuatan pohon pencarian menggunakan algoritma minimax dengan masukan berupa data yang telah diambil sebelumnya.
4. Pengambilan Keputusan : Tahap keempat adalah pengambilan keputusan berdasarkan pohon pencarian yang telah dibuat sebelumnya.
5. Penetapan Tempat : Tahap kelima atau tahap terakhir adalah penetapan tempat atau lokasi token dengan nilai maksimal yang didapat dari tahap pengambilan keputusan.

Diagram Alir Permainan. Diagram Alir bagaimana permainan akan berjalan pada perangkat lunak yang digunakan dalam penelitian ini dapat dilihat pada Gambar 6.

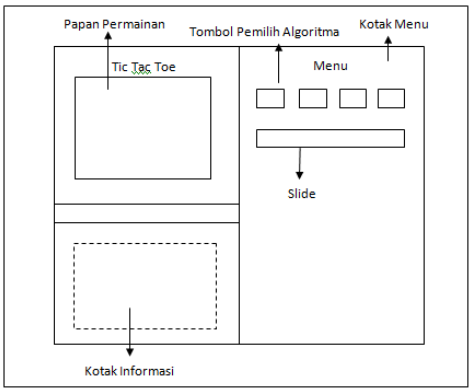


Gambar 6 Diagram Alir Permainan

Dari Gambar 6 dapat diketahui bahwa dalam permainan Tic Tac Toe 5 x 5 dengan algoritma minimax, memiliki tahap permainan sebagai berikut:

1. Tahap pertama adalah pemeriksaan token pemain pertama untuk menentukan apakah *User* 1 (komputer) atau *User* 2 (manusia) yang akan mendapatkan giliran pertama.
2. Jika giliran pertama adalah giliran *User* 2 maka permainan akan menunggu sampai *User* 2 memilih suatu lokasi untuk tokennya.
3. Tahap selanjutnya adalah permainan akan menandakan lokasi yang telah dipilih oleh *User* 2 dengan token *User* 2.
4. Setelah itu, permaianan akan mengecek apakah *User* 2 menang atau permainan berakhir seri, jika tidak maka akan beralih ke giliran *User* 1.
5. Pada giliran *User* 1, untuk penempatan tokennya, akan dilakukan dengan pemanggilan algoritma.
6. Hasil keluaran dari algoritma tersebut akan berupa lokasi penempatan token *User* 1.
7. Tahap selanjutnya adalah permainan akan menandakan lokasi yang telah dihasilkan oleh algoritma sebelumnya dengan token milik *User* 1.
8. Setelah itu, adalah pengecekan apakah *User* 1 menang atau permainan berakhir seri, jika tidak maka permainan akan berlanjut dengan giliran selanjutnya adalah giliran *User* 2.
9. Permainan akan berakhir jika *User* 1 atau *User* 2 berhasil mendapatkan nilai benar pada saat pengecekan kemenangan atau pada saat pengecekan permainan berakhir seri.

Rancangan *Interface* Permainan Tic Tac Toe. Gambar 7 adalah rancangan *Interface* dari papan permainan Tic Tac Toe yang akan digunakan dalam penelitian ini. Rancangan tersebut terdiri dari 2 bagian yaitu kotak menu dan papan permainan.



Gambar 7 Rancangan *Interface* Permainan Tic Tac Toe

Dari Gambar 7 dapat dilihat bahwa perangkat lunak yang digunakan akan dibagi menjadi 2 bagian yaitu papan permainan dimana tempat permainan Tic Tac Toe akan dilakukan, dibawah papan permainan terdapat kotak informasi yang akan berisi informasi seputar permainan yang dilakuka. Selain papan permainan dan kotak informasi akan terdapat Kotak Menu. Pada kotak menu terdapat 3 tombol untuk memilih algoritma yang akan digunakan, sebuah tombol *Toggle* untuk menandakan apakah pemain manusia akan mendapatkan giliran pertama atau tidak dan sebuah *slider* untuk mengendalikan batas kedalaman tempat dilakukannya pencarian pada pohon pencarian oleh algoritma minimax.

HASIL DAN PEMBAHASAN

Pengujian Menggunakan Blackbox. Pengujian *Blackbox* dilakukan untuk menguji kelayakan program sesuai dengan target yang diinginkan atau tidak. Pengujian ini berisikan kumpulan kuisoner atas serangkaian tes untuk memenuhi kriteria.

Tabel 1 Pengujian *Blackbox*

Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
Pemain memilih algoritma minimax	Permaian diatur ulang dengan algoritma minimax sebagai algoritma yang digunakan dalam permainan	Sesuai	Valid
Pemain memilih algoritma minimax yang telah diimplementasikan teknik *multithreading* dengan *thread* sebanyak 2	Permaian di atur ulang dengan algoritma minimax yang telah diimplementasikan teknik *multithreading* dengan *thread* sebanyak 2 sebagai algoritma yang digunakan	Sesuai	Valid
Pemain memilih algoritma minimax yang telah diimplementasikan teknik *multithreading* dengan *thread* sebanyak 3	Permaian di atur ulang dengan algoritma minimax yang telah diimplementasikan teknik *multithreading* dengan *thread* sebanyak 3 sebagai algoritma yang digunakan	Sesuai	Valid
Pemain memilih bermain sebagai pemain pertama	Ketika permainan di atur ulang giliran pertama adalah pemain	Sesuai	Valid
Pemain memilih bermain tidak sebagai pemain pertama	Ketika permainan di atur ulang giliran kedua adalah pemain	Sesuai	Valid
Pemain mengatur kedalaman pencarian menjadi 4	Batas maksimal pencarian dalam algoritma pencarian menjadi 4	Sesuai	Valid
Pemain mengatur kedalaman pencarian menjadi 5	Batas maksimal pencarian dalam algoritma pencarian menjadi 5	Sesuai	Valid
Pemain mengatur kedalaman pencarian menjadi 6	Batas maksimal pencarian dalam algoritma pencarian menjadi 6	Sesuai	Valid
Pemain selesai memasukkan gerakan yang diinginkan	Keadaan papan dikirim ke algoritma, dan algoritma mencari langkah terbaik berdasarkan keadaan papan, lalu mengirim langkah terbaik kembali ke papan permainan	Sesuai	Valid

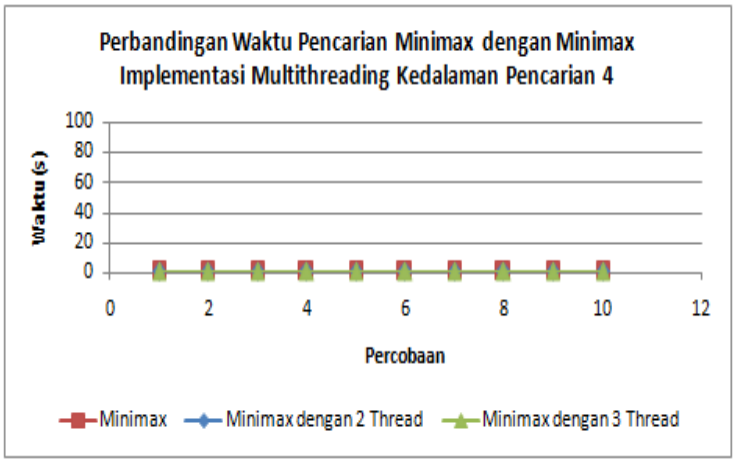
Pada Tabel 1 dapat dilihat hasil pengujian dari pengujian *Blackbox* yang telah dilakukan pada *User Interface* yang digunakan pada penelitian ini. Dapat dilihat bahwa semua fungsi yang diharapkan tercapai.

Uji Kasus Kedalaman Pencarian Maksimal 4. Pada kasus ini kedalaman pencarian maksimal pada algoritma minimax dan algoritma minimax implementasi *multithreading* akan bernilai 4.

Tabel 2 Hasil Waktu Pencarian Dengan Kedalaman Maksimal 4

Percobaan	Minimax(ms)	Minimax 2 Thread(ms)	Minimax 3 Thread(ms)
1	265	187	187
2	249	187	202
3	250	218	187
4	250	187	202
5	281	203	188
6	265	187	187
7	249	188	187
8	265	202	187
9	249	203	188
10	249	202	188

Pada Tabel 2 dapat dilihat bahwa waktu tercepat untuk mencari gerakan terbaik menggunakan algoritma minimax pada 10 kali percobaan adalah 249 ms, menggunakan minimax implementasi *multithreading* dengan 2 *thread* adalah 187 ms, dan menggunakan minimax implementasi *multithreading* dengan 3 *thread* adalah 187 ms. Dari ketiga waktu tersebut dapat diketahui pada kedalaman maksimal 4, kedua algoritma minimax implementasi *multithreading* lebih efisien dibandingkan algoritma minimax.



Gambar 8 Grafik Perbandingan Waktu Pencarian Minimax dengan Minimax Implementasi *Multithreading* dengan Kedalaman Pencarian Maksimal 4

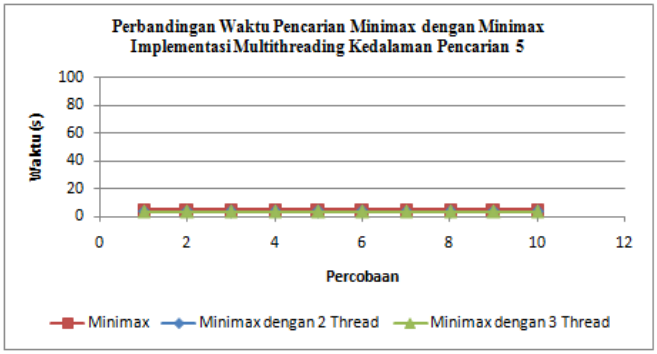
Dari Gambar 8 dapat dilihat bahwa kedua algoritma minimax dengan implementasi *multithreading* lebih cepat dibandingkan dengan algoritma minimax. Pemotongan waktu pencarian pada algoritma minimax implementasi *multithreading* dengan 2 *thread* tidak mencapai setengahnya walaupun pada algoritma minimax dengan implementasi *multithreading* ini menggunakan 2 buah *thread.* Hal ini dikarenakan *thread* tersebut membutuhkan waktu untuk berkomunikasi karena terpisah pada 2 *core* yang berbeda tetapi keduanya berbagi seksi data dan kode. Jumlah *core* prosesor juga mempengaruhi algoritma minimax implementasi *multithreading* dengan 3 *thread*, dikarenakan percobaan dilakukan pada laptop yang prosesornya hanya memiliki 2 *core* maka *thread* yang ketiga harus menunggu salah satu *di antara dua thread yang dieksekusi selesai terlebih dahulu agar dapat dieksekusi*.

Uji Kasus Kedalaman Pencarian Maksimal 5. Pada kasus ini kedalaman pencarian maksimal pada algoritma minimax dan algoritma minimax implementasi *multithreading* akan bernilai 5.

Tabel 3 Hasil Waktu Pencarian Dengan Kedalaman Maksimal 5

Percobaan	Minimax(ms)	Minimax 2 Thread(ms)	Minimax 3 Thread(ms)
1	4961	3588	3463
2	4945	3557	3479
3	4945	3510	3495
4	4960	3541	3479
5	4945	3494	3495
6	4961	3479	3463
7	4961	3572	3510
8	4929	3432	3541
9	4945	3557	3494
10	4946	3448	3510

Pada Tabel 3 dapat dilihat bahwa waktu tercepat untuk mencari gerakan terbaik menggunakan algoritma minimax pada 10 kali percobaan adalah 4929 ms, menggunakan minimax implementasi *multithreading* dengan 2 *thread* adalah 3432 ms, dan menggunakan minimax implementasi *multithreading* dengan 3 *thread* adalah 3463 ms. Dari ketiga waktu tersebut dapat diketahui pada kedalaman maksimal 5, kedua algoritma minimax implementasi *multithreading* lebih efisien dibandingkan algoritma minimax. Hal ini dibuktikan dari kedua waktu yang didapatkan dari penggunaan *thread* lebih dari satu.



Gambar 9 Grafik Perbandingan Waktu Pencarian Minimax dengan Minimax Implementasi *Multithreading* dengan Kedalaman Pencarian Maksimal 5

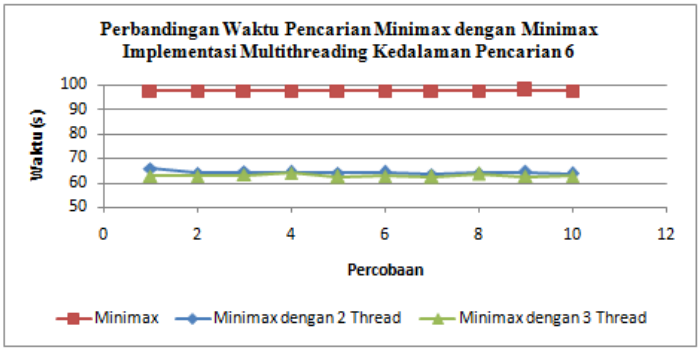
Dari Gambar 9 dapat dilihat bahwa kedua algoritma minimax dengan implementasi *multithreading* lebih cepat dibandingkan dengan algoritma minimax pada kedalaman pencarian maksimal 5, tetapi sama seperti percobaan sebelumnya dibutuhkannya waktu komunikasi diantara *thread*. Waktu komunikasi yang dibutuhkanpun semakin meningkat. Hal ini dikarenakan semakin meningkat jumlah tugas yang diberikan maka waktu komunikasi yang dibutuhkan diantara *thread* juga akan semakin meningkat.

Uji Kasus Kedalaman Pencarian Maksimal 6. Pada kasus ini kedalaman pencarian maksimal pada algoritma minimax dan algoritma minimax implementasi *multithreading* akan bernilai 6.

Tabel 4 Hasil Waktu Pencarian Dengan Kedalaman Maksimal 6

Percobaan	Minimax(ms)	Minimax 2 Thread(ms)	Minimax 3 Thread(ms)
1	97500	65832	62712
2	97485	64054	62744
3	97656	64397	63165
4	97516	64116	63913
5	97625	63929	62478
6	97547	64319	62712
7	97672	63398	62416
8	97657	64038	63632
9	97796	64209	62571
10	97547	63617	62665

Pada Tabel 4 dapat dilihat bahwa dari ketiga waktu tersebut dapat diketahui pada kedalaman maksimal 6, kedua algoritma minimax implementasi *multithreading* lebih efisien dibandingkan algoritma minimax.



Gambar 10 Grafik Perbandingan Waktu Pencarian Minimax dengan Minimax Implementasi *Multithreading* dengan Kedalaman Pencarian Maksimal 6

Dari Gambar 10 dapat dilihat bahwa kedua algoritma minimax dengan implementasi *multithreading* lebih cepat dibandingkan dengan algoritma minimax pada kedalaman pencarian maksimal 6, dan sama seperti percobaan sebelumnya waktu komunikasi mengalami peningkatan. Hal ini membuktikan bahwa waktu komunikasi akan meningkat sesuai dengan peningkatan pada jumlah tugas. Oleh karena itu maka percobaan dengan kedalaman pencarian 6 ini juga sama seperti percobaan sebelumnya, yaitu terpengaruhi oleh waktu komunikasi.

Speed Up. Dalam penelitian ini akan dilakukan perbandingan antara algoritma yang yang menggunakan sebuah *thread* dengan algoritma yang menggunakan lebih dari sebuah *thread.*

Tabel 5 Hasil *Speed Up* Minimax Implementasi *Multithreading* dengan 2 *Thread* terhadap Minimax

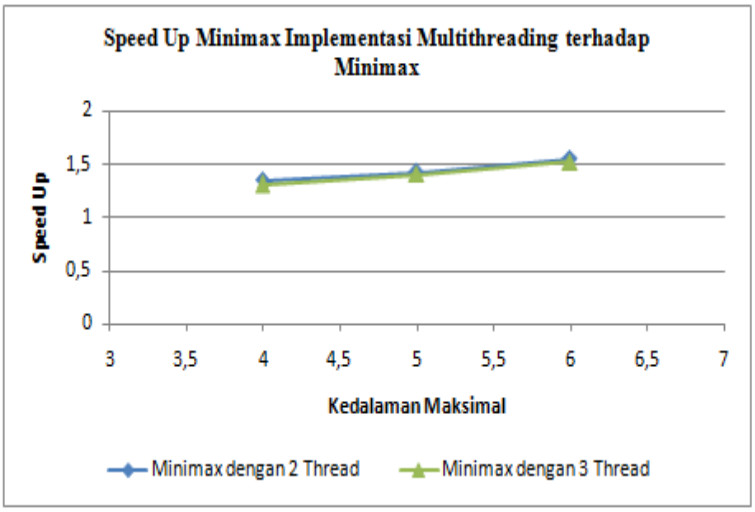
Kedalaman	Rerata Minimax(ms)	Rerata Minimax 2 Thread(ms)	Speed Up
4	257.2	196.4	1.31
5	4949.8	3517.8	1.41
6	97600.1	64190.9	1.52

Pada Tabel 5 dapat dapat dilihat bahwa setelah waktu eksekusi 10 percobaan pada kedalaman 4, 5, dan 6 dirata-ratakan, disimpulkan bahwa algoritma minimax implementasi multithreading dengan 2 thread lebih cepat dibandingkan algoritma minimax, dengan Speed Up sebesar 1.31 pada kedalaman maksimal 4, 1.41 pada kedalaman maksimal 5, dan 1.52 pada kedalaman maksimal 6.

Tabel 6 Hasil *Speed Up* Minimax Implementasi *Multithreading* dengan 3 *Thread* terhadap Minimax

Kedalaman	Rerata Minimax(ms)	Rerata Minimax 3 Thread(ms)	Speed Up
4	257.2	190.3	1.35
5	4949.8	3492.9	1.42
6	97600.1	62900.8	1.55

Pada Tabel 6 dapat dilihat bahwa setelah waktu eksekusi 10 percobaan pada kedalaman 4, 5, dan 6 dirata-ratakan, disimpulkan bahwa algoritma minimax implementasi multithreading dengan 3 thread lebih cepat dibandingkan algoritma minimax, dengan Speed Up sebesar 1.35 pada kedalaman maksimal 4, 1.42 pada kedalaman maksimal 5, dan 1.55 pada kedalaman maksimal 6.



Gambar 11 Grafik *Speed Up* Minimax Implementasi *Multithreading* terhadap Minimax

Dari Gambar 11 dapat dilihat bahwa semakin dalam batas pencarian yang dilakukan maka *Speed Up* yang didapatkan akan semakin besar. Hal ini dikarenakan semakin dalam batas maksimal pencarian maka tugas yang dibagikan semakin banyak sehingga keuntungan yang didapatkan dari pembagian tugas tersebut akan semakin besar, keuntungan tersebut akan menutupi kerugian waktu komunikasi yang terjadi. Selain itu dapat diketahui bahwa *Gradien Speed Up* yang dihasilkan dari 2 *thread* pada kedalaman 4 dan 5 adalah 0.1 dan pada kedalaman 5 dan 6 adalah 0.11. Sedangkan *Gradien Speed Up* yang dihasilkan dari 3 *thread* pada kedalaman 4 dan 5 adalah 0.07 dan pada kedalaman 5 dan 6 adalah 0.13. Sehingga dapat disimpulkan bahwa pada kedalaman 4 dan 5 peningkatan *Speed Up* pada 2 *thread* lebih besar daripada 3 *thread*, sedangkan pada kedalaman 5 dan 6 peningkatan *Speed Up* pada 3 *thread* lebih besar daripada 2 *thread.*

KESIMPULAN

Kesimpulan yang dapat diambil dari pembahasan dan penelitian mengenai implementasi teknik *multithreading* pada algoritma minimax adalah algoritma minimax dengan implementasi teknik *multithreading* lebih efisien dibandingkan algoritma minimax tanpa implementasi teknik *multithreading*. Hal ini dibuktikan dengan *Speed Up* yang didapatkan dari perata-rataan 10 percobaan algortima minimax implementasi *multithreading* dengan 2 *thread*, yaitu 1.31 pada kedalaman maksimal 4, 1.41 pada kedalaman maksimal 5, dan 1.52 pada kedalaman maksimal 6. Sedangkan *Speed Up* yang didapatkan dari perata-rataan 10 percobaan algortima minimax implementasi *multithreading* dengan 3 *thread*, yaitu 1.35 pada kedalaman maksimal 4, 1.42 pada kedalaman maksimal 5, dan 1.55 pada kedalaman maksimal 6. Pada kedalaman 4 dan 5 peningkatan *Speed Up* pada 2 *thread* lebih besar daripada 3 *thread*, sedangkan pada kedalaman 5 dan 6 peningkatan *Speed Up* pada 3 *thread* lebih besar daripada 2 *thread.* Hal ini dapat dilihat dari *Gradien Speed Up* yang dihasilkan dari 2 *thread* pada kedalaman 4 dan 5 adalah 0.1 dan pada kedalaman 5 dan 6 adalah 0.11. Sedangkan *Gradien Speed Up* yang dihasilkan dari 3 *thread* pada kedalaman 4 dan 5 adalah 0.07 dan pada kedalaman 5 dan 6 adalah 0.13.

SARAN

Penelitian implementasi teknik *multithreading* pada algoritma minimax ini masih jauh dari kata sempurna dan perlu di sempurnakan. Kekurangan pada penelitian ini adalah keterbatasan *core* yang digunakan sehingga lebih efisien ketika menggunakan 2 buah *thread* saja, selain itu algoritma yang diimplementasikan hanya algoritma minimax biasa. Kedepannya diharapkan dilakukan penelitian lebih lanjut menggunakan komputer dengan *core* yang lebih banyak untuk melihat tingkat efisiensinya dan menggunakan algoritma minimax yang telah dikembangkan seperti algoritma minimax alpha beta pruning.

DAFTAR PUSTAKA

Kosasi, S., 2014, *Permainan Papan Strategi Menggunakan Algoritma Minimax*, SNITIKI 6, ISSN : 2085 – 9902.

Microsystem, S., 1994, Multithreaded Programming Guide, Sun Microsystem, USA : 2550 Garcia Avenue Mountain View, CA 94043.

Millington, I., dan Funge, J., 2009, *Artificial Intelligence For Games*, Elsevier, Kanada, USA.

Nuzulla, B., dan Solichin, A., 2012, *Implementasi Algoritma Steepest Ascend Hill Climbing dengan Optimasi Minimax pada Permainan Tic Tac Toe Berbasis Android*, Jurusan Teknik Informatika, ISSN : 2339-210X, Fakultas Teknologi Informasi, Universitas Budi Luhur.

Sarcar, A., 2009, *Performance Analysis of BFS & DFS Algorithms for Various Applications*, Dept of Computer Science, University Avenue, El Paso.

Silberschatz, A., 2005, *Operating System Concepts*, John Wiley and Sons Inc, New York.