IMPLEMENTASI ALGORITMA MINIMAX DAN ALPHA BETA PRUNING DALAM PERMAINAN TRADISIONAL NTIL

SKRIPSI

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh :

MARTASO'ID FITRIZA

NIM. 105090606111002

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2015

LEMBAR PERSETUJUAN

IMPLEMENTASI ALGORITMA MINIMAX DAN ALPHA BETA PRUNING DALAM PERMAINAN TRADISIONAL NTIL

SKRIPSI

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun Oleh:

MARTASO'ID FITRIZA

NIM. 105090606111002

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Wibisono Sukmo Wardhono, S.T., M.T	**Eriq M. Adams J, S.T., M.Kom.**
NI K. 820404 06 1 1 0091	**NIP. 19850410 201212 1 001**

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA MINIMAX DAN ALPHA BETA PRUNING DALAM PERMAINAN TRADISIONAL NTIL

Disusun Oleh:

MARTASO'ID FITRIZA

NIM. 105090606111002

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 29 Januari 2015

Penguji I	Penguji II
Issa Arwani, S.Kom., M.Sc.	**Fajar Pradana, S.ST, M.Eng**
NIP. 19830922 201212 1 003	**NIK. 871121 16110371**

Penguji III

Imam Cholissodin, S.Si, M.Kom

NIK. 850719 16 1 1 0422

Mengetahui

Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.

NIP. 19670801 199203 1 001

PERNYATAAN

ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 4 Februari 2015

Mahasiswa

Martaso'id Fitriza

NIM. 105090606111002

KATA PENGANTAR

Alhamdulillahi rabbil ‘alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayah-Nya, penulis dapat menyelesaikan tugas akhir ini. Dalam pelaksanaan dan penulisan tugas akhir ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moral maupun material. Dalam kesempatan ini penulis ingin mengucapkan terimakasih yang sebesar – besarnya kepada :

1. Kedua orang tua, yang terus mendo’akan dan menyemangati penulis untuk menyelesaikan skripsi secepat mungkin, meskipun jauh dari tepat waktu.
2. Wibisono Sukmo Wardhono,ST,.MT dan Eriq Muhammad Adams J.,S.T., selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
3. Semua pihak lain yang telah membantu terselesaikannnya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik membangun dari para pembaca senantiasa penulis harapkan guna perbaikan bagi tugas akhir selanjutnya. Semoga tugas akhir ini dapat memberikan manfaat bagi semua pihak, Amin.

Malang, Januari 2015

Penulis

repository.ub.ac.id

ABSTRAK

Martaso'id Fitriza. 2014 : Implementasi Algoritma *Minimax* dan *Alpha-Beta Pruning* dalam Permainan Tradisional Ntil. Skripsi Program Studi Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Wibisono Sukmo Wardhono, ST., MT. dan Eriq M. Adams J, S.T., M.Kom

Pada permainan Ntil, jumlah seluruh kombinasi langkah bidak yang mungkin terjadi sangat banyak. Dari keseluruhan langkah tersebut, terdapat langkah yang akan merugikan pemain, dan terdapat langkah yang akan menguntungkan pemain. Berdasarkan hal ini, maka dibutuhkan sebuah algoritma untuk menentukan langkah yang akan menguntungkan pemain atau agen dari kumpulan langkah yang ada.

Skripsi ini membahas mengenai implementasi algoritma *Minimax*, dan Algoritma Alpha-beta Pruning untuk pencarian langkah terbaik yang akan diambil oleh agen cerdas dalam permainan Ntil. Algoritma *Minimax* akan memaksimalkan langkah agen dan meminimalkan langkah lawan dari agen. Algoritma *Alpha-beta Pruning* merupakan optimasi dari algoritma *Minimax*, yang akan mengurangi jumlah simpul yang dievaluasi dalam pohon pencarian. Pohon pencarian dibangkitkan dengan mengambil semua kemungkinan langkah bidak pada permainan Ntil. Algoritma *Minimax* dan *Alpha-beta Pruning* dapat diimplementasikan untuk pencarian langkah terbaik yang akan diambil oleh agen dalam permainan Ntil. Pencarian langkah terbaik dengan menggunakan algoritma *Minimax* menghasilkan langkah yang sama dengan algoritma *Alpha-beta* Pruning. Algoritma *Alpha-beta Pruning* memiliki kinerja yang lebih baik dibandingkan algoritma *Minimax* dengan rata-rata perbedaan *running time* sebesar 1354.367 ms terhadap nilai *depth*, dan 65.9776 ms terhadap ukuran papan permainan.

Kata kunci : Permainan tradisional, permainan Ntil, *game board*, algoritma *Minimax*, algoritma *Alpha-beta Pruning*.

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Martaso'id Fitriza. 2014 : Implementasi Algoritma *Minimax* dan *Alpha-Beta Pruning* dalam Permainan Tradisional Ntil. Skripsi Program Studi Ilmu Komputer, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Wibisono Sukmo Wardhono, ST., MT. dan Eriq M. Adams J, S.T., M.Kom

In Ntil game, there are a lot of possible combinations of pawn movements. Some of them will make the player have a disadvantage in their position, and some of them will make the player have an advantage in their position. Based on this, an algorithm that determines the best pawn movements that will make the player have an advantage in their position is needed.

This paper discusses the implementation of Minimax algorithm, and Alpha-beta pruning algorithm to determine the best movements for the intelligent agent in Ntil Game. Minimax algorithm will maximize the agent's moves and minimize the opponent's moves. Alpha-beta Pruning algorithm is an optimization of Minimax algorithm, which will reduce the number of nodes evaluated in the search tree. A search tree is generated by taking all the possibilities of pawn's movements in the Ntil game. Minimax and Alpha-beta Pruning algorithm can be implemented as an intelligent agent for the Ntil game. The movements of pawns that are generated by Minimax algorithm have the same result with Alpha-beta Pruning algorithm. The performance of Alpha-beta Pruning algorithm is better than the Minimax algorithm with avarage different of 1354.367ms on thier running time based on depth's value, and with avarage different of 65.9776ms on thier running time based on board's size.

Keywords : *Traditional game, Ntil game, game board, Minimax algorithm, Alpha-beta Pruning algorithm.*

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

KATA PENGANTAR i
ABSTRAK ii
ABSTRACT iii
DAFTAR ISI iv
DAFTAR TABEL vii
DAFTAR GAMBAR viii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Sistematika Penulisan 3
BAB II DASAR TEORI 5
2.1 Kajian Pustaka 5
2.2 Permainan Ntil 6
2.2.1 Bidak dan Papan Permainan Ntil 6
2.2.2 Aturan Permainan Ntil 7
2.3 Algoritma *Minimax* 9
2.4 Algoritma *Alpha-Beta Pruning* 11
BAB III METODOLOGI DAN PERANCANGAN 12
3.1 Studi Literatur 13
3.2 Perancangan AI 14
3.2.1 Perancangan Dataset 14
1. Perancangan Bidak Permainan Ntil 15
2. Perancangan Papan Permainan Ntil 15
3.2.2 Perancangan Agen Cerdas 19
1. Perancangan Algoritma *Minimax* 20
2. Perancangan Algoritma *Alpha-Beta Pruning* 21

3. Perancangan Fungsi *evaluate* 21
4. Perancangan Fungsi *bestMove* 21
3.2.3 Perhitungan Manual 26
3.3 Implementasi 29
3.4 Pengujian dan Analisis 30
3.4.1 Pengujian 30
3.4.2 Analisis 30
BAB IV IMPLEMENTASI 32
4.1 Lingkungan Implementasi 32
4.1.1 Lingkungan Implementasi Perangkat Keras 32
4.1.2 Lingkungan Implementasi Perangkat Lunak 32
4.2 Batasan Implementasi 33
4.3 Implementasi Dataset 33
4.3.1 Implementasi Bidak Permainan Ntil 33
4.3.2 Implementasi Papan Permainan Ntil 34
4.4 Implementasi Agen Cerdas 39
4.4.1 Implementasi Algoritma *Minimax* 39
4.4.2 Implementasi Algoritma *Alpha-beta Pruning* 41
4.4.3 Implementasi Fungsi *evaluate* 43
4.4.4 Implementasi Fungsi *bestMove* 45
BAB V PENGUJIAN DAN ANALISIS 47
5.1 Lingkungan Pengujian 47
5.1.1 Lingkungan Pengujian Perangkat Keras 47
5.1.2 Lingkungan Pengujian Perangkat Lunak 47
5.2 Skenario Pengujian 48
5.2.1 Skenario I 48
5.2.2 Skenario II 48
5.2.3 Skenario III 49
5.2.4 Skenario IV 49
5.3 Hasil Pengujian 49
5.3.1 Skenario I 50
5.3.2 Skenario II 50

5.3.3 Skenario III 52
5.3.4 Skenario IV 53
5.3 Analisis 54
5.4.1 Skenario I 54
5.4.2 Skenario II 54
5.4.3 Skenario III 55
5.4.4 Skenario IV 56
BAB VI PENUTUP 57
6.1 Kesimpulan 57
6.2 Saran 57
DAFTAR PUSTAKA 58

DAFTAR TABEL

Tabel 2.1 *Pseudocode* Algoritma *Minimax* 10
Tabel 2.2 *Pseudocode* Algoritma *Alpha-beta Pruning* 12
Tabel 3.1 Perancangan Bidak Permainan Ntil 15
Tabel 3.2 Perancangan Bobot Nilai Titik pada Papan Permainan Ntil 15
Tabel 4.1 Spesifikasi Perangkat Keras Implementasi 32
Tabel 4.2 Spesifikasi Perangkat Lunak Implementasi 32
Tabel 4.3 Implementasi Bidak Permainan Ntil 33
Tabel 4.4 Template Papan Permainan Ntil 34
Tabel 4.5 *Pseudocode* Fungsi *minimax* 39
Tabel 4.6 *Pseudocode* Fungsi *alphabeta* 41
Tabel 4.7 *Pseudocode* Fungsi *evaluate* 43
Tabel 4.8 *Pseudocode* Fungsi *calculateValue* 44
Tabel 4.9 *Pseudocode* Fungsi *bestMove* 45
Tabel 5.1 Spesifikasi Perangkat Keras Pengujian 47
Tabel 5.2 Spesifikasi Perangkat Lunak Pengujian 47
Tabel 5.3 Hasil Pengujian Skenario I 50
Tabel 5.4 Hasil Pengujian Skenario II 50
Tabel 5.5 Hasil Pengujian Skenario III 52
Tabel 5.6 Hasil Pengujian Skenario IV 53

DAFTAR GAMBAR

Gambar 2.1 Papan Permainan Ntil dan Posisi Awal Bidak 7
Gambar 2.2 Pergerakan Bidak Biasa dan Bidak Raja Pada Permainan Ntil..... 8
Gambar 2.3 Pemain A Memakan Bidak Pemain B dalam Permainan Ntil..... 8
Gambar 2.4 Pohon Permainan Algoritma *Minimax*..... 10
Gambar 2.5 Pohon Permainan Algoritma *Alpha-beta Pruning* 11
Gambar 3.1 Diagram Alir Metode Penelitian 13
Gambar 3.2 Diagram Alir Perancangan 14
Gambar 3.3 Diagram Alir Perancangan Dataset 14
Gambar 3.4 Perancangan Papan Permainan Ntil Berukuran 5x5 16
Gambar 3.5 Perancangan Papan Permainan Ntil Berukuran 5x9 17
Gambar 3.6 Perancangan Papan Permainan Ntil Berukuran 7x9 17
Gambar 3.7 Perancangan Papan Permainan Ntil Berukuran 9x9 18
Gambar 3.8 Perancangan Papan Permainan Ntil Berukuran 7x11 18
Gambar 3.9 Perancangan Papan Permainan Ntil Berukuran 9x11 19
Gambar 3.10 Diagram Alir Perancangan Agen Cerdas 20
Gambar 3.11 Diagram Alir Perancangan Algoritma *Minimax* 22
Gambar 3.12 Diagram Alir Perancangan Algoritma *Alpha-beta Pruning*..... 23
Gambar 3.13 Diagram Alir Perancangan Fungsi *evaluate*..... 24
Gambar 3.14 Diagram Alir Perancangan Fungsi *bestMove*..... 25
Gambar 3.15 Pencarian Langkah Terbaik oleh Agen B dengan *depth=2* 26
Gambar 3.16 Pohon Perminan algoritma *Minimax*..... 27
Gambar 3.17 Evaluasi Papan Permainan pada Simpul A11 28
Gambar 3.18 Pohon Perminan algoritma *Alpha-beta Pruning* 29
Gambar 3.19 Diagram Alir Tahapan Implementasi..... 30
Gambar 3.20 Diagram Alir Tahapan Pengujian dan Analisis..... 30
Gambar 4.1 *Screenshot* Model 3D Bidak Biasa Permainan Ntil 34
Gambar 4.2 *Screenshot* Model 3D Bidak Raja Permainan Ntil..... 34
Gambar 4.3 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 5x5..... 37

Gambar 4.4 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 5x9 37
Gambar 4.5 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 7x9 38
Gambar 4.6 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 9x9 38
Gambar 4.7 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 7x11 38
Gambar 4.8 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 9x11 39
Gambar 5.1 Grafik Hasil Pengujian Skenario II antara *depth* dengan *Running Time* 51
Gambar 5.2 Grafik Hasil Pengujian Skenario II antara *depth* dengan *Expanded Node* 51
Gambar 5.3 Grafik Hasil Pengujian Skenario III antara Ukuran Papan dengan *Running Time* 52
Gambar 5.4 Grafik Hasil Pengujian Skenario III antara Ukuran Papan dengan *Expanded Node* 53

BAB I
PENDAHULUAN

1.1 Latar Belakang

Permainan atau sering disebut dengan *game* merupakan suatu aksi yang didalamnya terdapat pemain, aturan-aturan tertentu, dan suatu kondisi menang [ROG-10]. Permainan dapat menjadi suatu sarana hiburan yang diminati dan dimainkan oleh banyak orang baik dari kalangan anak-anak, remaja maupun orang dewasa.

Permainan tradisional sering disebut juga permainan rakyat, merupakan permainan yang tumbuh dan berkembang pada masa lalu terutama tumbuh di masyarakat pedesaan. Permainan tradisional tumbuh dan berkembang berdasar kebutuhan masyarakat setempat [YUN-81]. Adapun jenis dari permainan tradisional ini yaitu : Grobak sodor, petal umpet, bentengan, ntil, dan lainnya. Permainan modern merupakan permainan yang disajikan pada suatu piranti atau perangkat teknologi dan dimainkan secara virtual. Media teknologi yang digunakan, yaitu *console*, komputer, *handphone* dan elektronik lainnya. Seiring perkembangan teknologi, permainan tradisional ini sudah jarang dimainkan dan kalah bersaing dari permainan modern.

Permainan Ntil merupakan permainan tradisional yang berasal dari Lombok. Permainan ini berjenis *board game* yang dimainkan oleh dua pemain secara bergiliran. Tujuan permainan ini adalah untuk menghabiskan bidak lawan dengan aturan permainan yang telah ditentukan. Pemain harus berpikir dalam menentukan langkah bidaknya, sehingga pemain dapat menghabiskan bidak lawan.

Pada permainan Ntil, jumlah seluruh kombinasi langkah bidak yang mungkin terjadi sangat banyak. Dari keseluruhan langkah tersebut, tidak semua langkah merupakan langkah yang baik. Terdapat langkah yang akan merugikan pemain, dan terdapat langkah yang akan menguntungkan pemain. Jika pemain mengambil langkah bidak secara acak, tanpa mempertimbangkan langkah bidak lain yang optimal, maka pemain tersebut memiliki kemungkinan kalah yang lebih

besar. Berdasarkan hal ini, maka dibutuhkan sebuah algoritma untuk menentukan langkah optimal tersebut dari sekumpulan langkah yang ada.

Agen cerdas adalah kreasi program komputer yang memiliki aksi dan pemikiran seperti manusia. Ada beberapa algoritma pencarian langkah, yang dapat digunakan untuk membangun agen cerdas, pada permainan yang dimainkan oleh dua pemain secara bergiliran. Salah satunya dengan algortima *Minimax,* dan *Alpha-Beta Pruning*. Penelitian Agus (2013) menunjukkan bahwa algoritma *Minimax* dapat digunakan untuk melakukan pencarian langkah terbaik yang diambil oleh agen cerdas pada permainan damdaman berbasis *mobile* [PRA-13]. Penelitian Lestari (2013) menunjukkan bahwa algoritma *Alpha-Beta Pruning* dapat diterapkan dalam menentukan pencarian langkah terbaik pada aplikasi permainan Bantumi berbasis *mobile* [LES-13].

Berdasarkan uraian diatas, penulis akan menerapkan algoritma *Minimax* dan *Alpha-beta Pruning* sebagai agen cerdas dalam permainan Ntil. Dengan pengembangan ini permainan ntil dapat dimainkan pada perangkat keras komputer, yang memiliki agen cerdas sebagai lawan main. Hal ini akan memudahkan pemain untuk bermain permainan tradisional ntil dengan lawan main yang sulit dikalahkan.

1.2 Rumusan Masalah

Berdasarkan paparan latar belakang tersebut, maka rumusan masalah adalah :

1. Mengimplementasikan algoritma *Minimax*, dan *Alpha-beta Pruning* sebagai agen cerdas dalam permainan Ntil.
2. Menguji kinerja algoritma *Minimax*, dan *Alpha-beta Pruning* dalam permainan Ntil.

1.3 Batasan Masalah

Batasan masalah dalam skripsi ini adalah :

1. Agen cerdas yang dibangun menerapkan algoritma *Minimax,* dan *Alpha-beta Pruning* dalam pencarian langkah bidak permainan Ntil.
2. Permainan dibangun dalam bentuk 3D untuk perangkat keras komputer.

3. Unity 4.5.4 sebagai lingkungan pengembangan.
4. Batas maksimum penelusuran kedalaman pohon permainan adalah 6.

1.4 Tujuan

Tujuan dari skripsi ini adalah mengimplementasikan algoritma *Minimax*, dan *Alpha-beta Pruning* untuk menghasilkan agen cerdas pada permainan ntil.

1.5 Manfaat

Manfaat dari hasil skripsi ini adalah:

1. Bagi Penulis :
   a. Mengaplikasikan ilmu yang didapat selama mengikuti perkuliahan di Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
   b. Menambah pengetahuan dan wawasan bagi penulis mengenai algoritma *Minimax* dan *Alpha-beta Pruning*.
2. Bagi Akademis :

   Dapat dijadikan sebagai pembanding atau literatur dalam penulisan skripsi yang menerapkan algoritma *Minimax* dan *Alpha-beta Pruning* dimasa yang akan datang.

1.6 Sistematika Penulisan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam tugas akhir ini adalah sebagai berikut:

Bab I Pendahuluan

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

Bab II Dasar Teori

Membahas teori dasar dan teori penunjang yang berkaitan dengan permainan tradisional ntil, algoritma *Minimax* dan *Alpha-beta Pruning*.

Bab III Metodologi dan Perancangan

Membahas tentang metode yang digunakan dalam penulisan yang terdiri dari studi literatur, perancangan, implementasi, pengujian dan analisis.

Serta membahas tentang perancangan permainan, algoritma minmax dan *Alpha-beta Pruning* yang akan diimplementasikan kedalam permainan ntil.

Bab IV Implementasi

Membahas tentang implementasi algoritma *Minimax* dan *Alpha-beta Pruning* untuk pembuatan agen cerdas pada permainan ntil.

Bab V Pengujian dan Analisis

Memuat tentang hasil pengujian terhadap metode yang telah diimplementasikan, serta menganalisa hasil pengujian.

Bab VI Penutup

Memuat kesimpulan dan saran yang diperoleh dari implementasi dan pengujian untuk pengembangan lebih lanjut.

BAB II
DASAR TEORI

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini adalah membahas 2 penelitian sebelumnya yang berjudul "Implementasi Kecerdasan Buatan Pada Permainan Damdaman Menggunakan Algoritma *Minimax*", dan "Implementasi Algoritma Alpha-Beta Pruning pada Permainan Bantumi dengan Berbasis *Mobile* Android".

Tujuan dari penelitian pertama adalah menerapkan algoritma minimax pada permainan Damdaman untuk menghasilkan kecerdasan buatan. Tujuan dari penelitian kedua adalah menerapkan algoritma alpha-beta pruning sebagai agen cerdas pada permainan Bantumi.

Perbedaan penelitian ini dengan penelitian pertama adalah algortima yang digunakan. Pada penelitian pertama menerapkan algoritma Minimax untuk menentukan langkah yang akan diambil oleh agen pada permainan damdaman. Pada penelitian ini menggunakan algoritma Minimax, dan alpha-beta pruning untuk menentukan langkah yang akan diambil oleh agen pada permainan Ntil. Permainan Ntil yang digunakan pada penelitian ini memiliki aturan main yang hampir sama dengan permainan Damdaman pada penelitian pertama. Permainan ini dikenal dengan nama Ntil di daerah Lombok, sedangkan di Jawabarat dikenal dengan nama Damdaman.

Perbedaan penelitian ini dengan penelitian kedua adalah permainan yang digunakan. Pada penelitian pertama menerapkan algoritma Alpha-Beta pruning untuk menentukan langkah yang akan diambil oleh agen pada permainan Bantumi. Pada penelitian ini menggunakan algoritma Minimax, dan Alpha-Beta pruning untuk menentukan langkah yang akan diambil oleh agen pada permainan Ntil. Permainan Ntil dan permainan Bantumi memiliki aturan main yang berbeda. Namun, kedua permainan tersebut dimainkan oleh dua pemain secara bergiliran.

Pada penelitian pertama, permainan yang digunakan adalah permainan Damdaman. Algoritma yang diterapkan untuk pembuatan agen cerdas adalah algoritma Minimax. Berdasarkan hasil penelitian pertama dapat disimpulkan

bahwa agen cerdas yang menerapkan algoritma *Minimax* pada permainan Damdaman menunjukkan hasil yang cukup baik [PRA-13].

Pada penelitian kedua, permainan yang digunakan adalah permainan Bantumi. Algoritma yang diterapkan untuk pembuatan agen cerdas adalah algoritma Alpha-beta Pruning. Berdasarkan hasil penelitian kedua dapat disimpulkan bahwa algoritma *Alpha-beta Pruning* dapat diimplementasikan dalam menentukan pencarian langkah terbaik pada permainan Bantumi[LES-13].

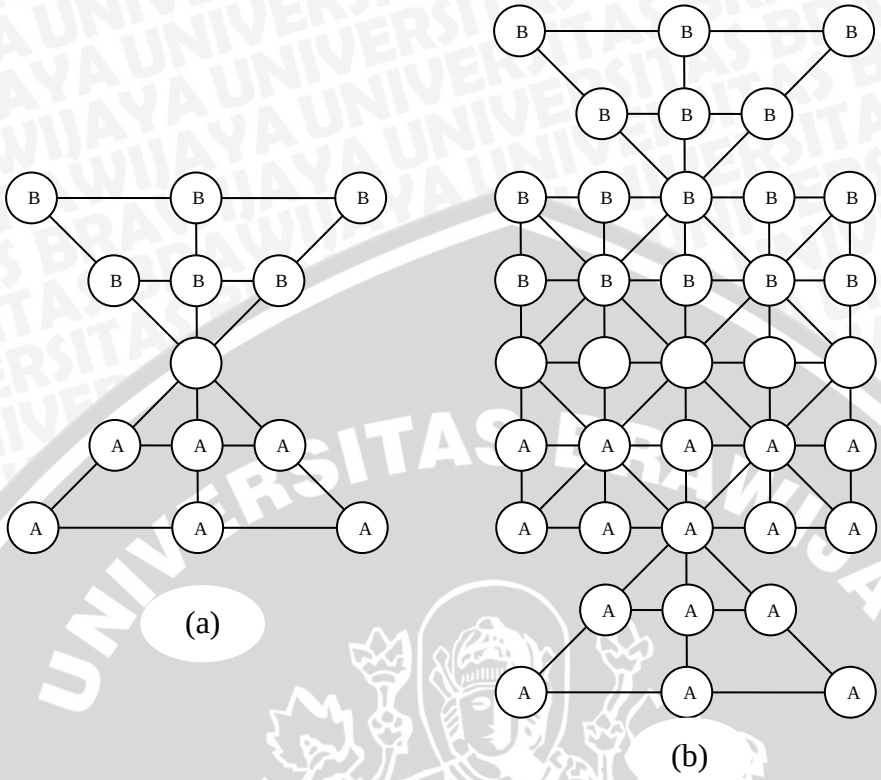
2.2 Permainan Ntil

Permainan Ntil merupakan permainan tradisional yang berasal dari daerah Lombok. Permainan ini mampu meningkatkan kecerdasan logika dan daya ingat pemain. Permainan ini juga terdapat di daerah lain, dengan nama yang berbeda. Masyarakat Yogya mengenal permainan ini dengan nama macanan, kemudian di masyarakat Betawi permainan ini dikenal dengan nama damdas, dan di jawabarat permaian ini dikenal dengan Damdaman. Dengan perbedaan nama disetiap daerah, maka dalam aturan yang digunakan sedikit berbeda, karena tidak adanya aturan yang jelas tertulis.

Permainan Ntil dimainkan oleh dua orang pemain, pada sebuah papan permainan, dengan jumlah bidak tertentu. Permainan ini dimainkan secara bergiliran, dengan menggerakkan bidak ke titik yang kosong pada papan permainan. Permainan ini akan selesai ketika salah satu bidak pemain habis dimakan oleh pemain lainnya, atau kedua pemain sama-sama memiliki satu bidak.

2.1.1 Bidak dan Papan Permainan Ntil

Permainan Ntil biasanya dimainkan pada dua jenis papan permainan, yaitu: Papan permainan yang berukuran 5x5 yang memiliki 14 titik, dengan masing-masing bidak pemain berjumlah 6. Dan papan permainan yang berukuran 5x9 yang memiliki 37 titik, dengan masing-masing bidak pemain berjumlah 16. Gambar 2.1a merupakan gambar papan permainan Ntil berukuran 5x5. Gambar 2.1b merupakan gambar papan permainan Ntil berukuran 5x9.



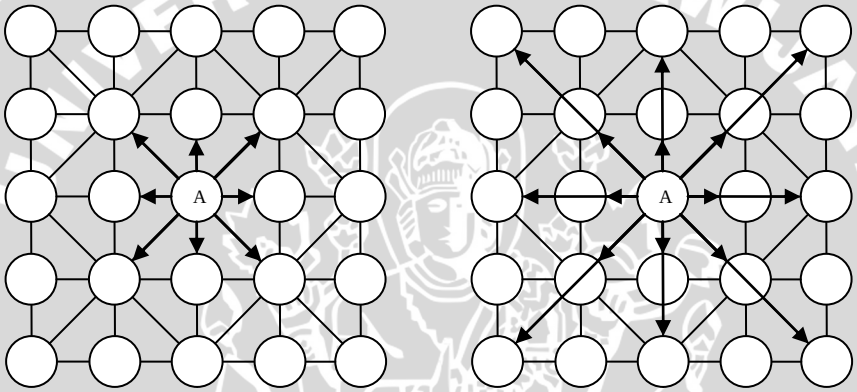
Gambar 2.1 Papan Permainan Ntil dan Posisi Awal Bidak

2.1.2 Aturan Permainan Ntil

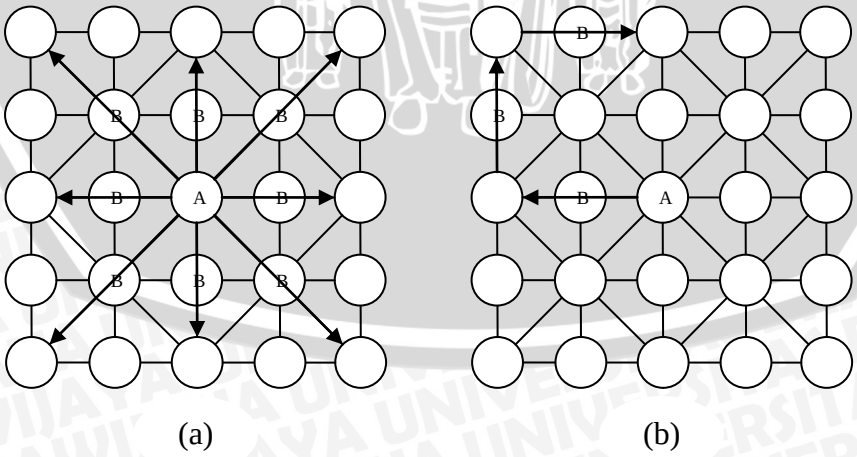
Adapun aturan dari permainan Ntil, diantara lain :

a. Permainan dimainkan oleh dua pemain yang dijalankan secara bergiliran.
b. Permainan ini memiliki dua jenis bidak, yaitu : Bidak biasa dan bidak raja.
c. Pada awal permainan, semua bidak adalah bidak biasa.
d. Bidak biasa akan menjadi bidak raja, jika bidak biasa berada di titik ujung tengah daerah lawan.
e. Bidak biasa bergerak satu titik setiap gerakan ke titik yang kosong yang terhubung oleh garis, dan tidak bisa bergerak mundur. Gambar 2.2a menunjukkan gerakan bidak biasa.
f. Bidak raja dapat bergerak lebih ke satu titik yang kosong selama titik-titik tersebut terhubung oleh garis lurus, dan bisa bergerak mundur. Gambar 2.2b menunjukkan gerakan bidak raja.

g. Pemain memakan bidak lawan dengan melompati bidak tersebut ke titik yang kosong yang terhubung oleh garis. Gambar 2.3a menunjukkan pemain memakan salah satu bidak lawan.
h. Pemain bisa memakan lebih dari satu bidak lawan dalam satu giliran, selama terdapat bidak yang bisa dimakan. Gambar 2.3b menunjukkan pemain memakan lebih dari satu bidak lawan.
i. Ntil merupakan peristiwa yang akan terjadi ketika pemain tidak memakan bidak lawan main. Ketika terjadi ntil maka lawan main akan mengambil tiga bidak pemain.



Gambar 2.2 Pergerakan Bidak Biasa dan Bidak Raja dalam Permainan Ntil



Gambar 2.3 Pemain A Memakan Bidak Pemain B dalam Permainan Ntil

2.3 Algoritma *Minimax*

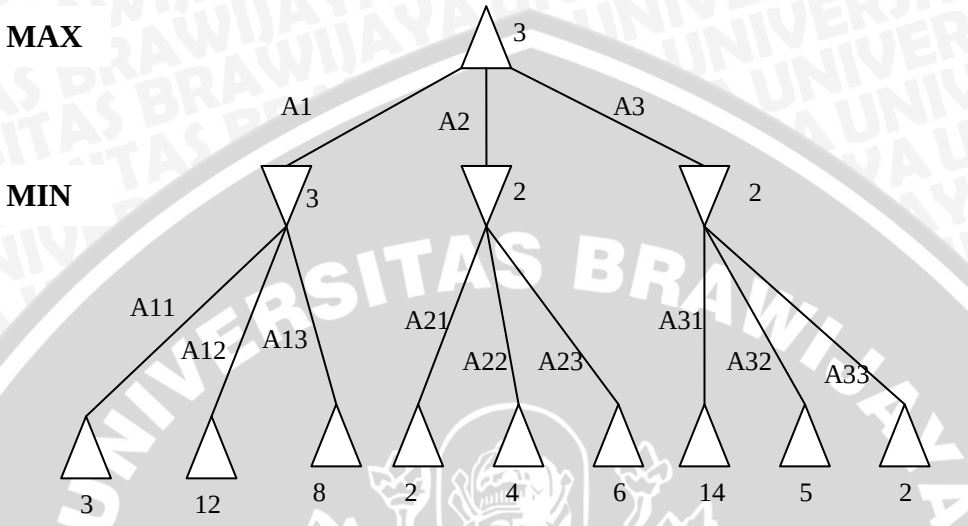
Algoritma *Minimax* merupakan satu metode yang sangat bermanfaat untuk tipe permainan bergiliran dengan dua pemain. Metode ini memilih langkah terbaik yang diberikan, dimana kedua pemain berusaha mengatur sebuah strategi untuk memenangkan permainan. Sehingga semua langkah selanjutnya akan dapat diketahui [JON-08].

Pada permainan yang dimainkan oleh dua pemain, MAX merepresentasikan pemain, dan MIN merepresentasikan lawan main. MAX akan malangkah terlebih dahulu, dilanjutkan dengan langkah MIN yang dilakukan secara bergiliran sampai permainan selesai. Algoritma *Minimax* digunakan untuk menentukan strategi yang paling optimal yang akan diambil oleh MAX untuk menetukan langkah pertama yang terbaik [RUS-95].

Algoritma *Minimax* terdiri dari 5 tahap [RUS-95], di antara lain:

a. Membangkitkan pohon permainan, dari kondisi awal permainan sampai kondisi akhir permainan.
b. Melakukan evaluasi kondisi permainan pada setiap simpul terakhir dalam pohon permainan yang telah dibangkitkan untuk mendapatkan nilainya.
c. Menggunakan nilai yang ada pada simpul terakhir, untuk menentukan nilai pada simpul yang berada satu level lebih tinggi dalam pohon pencarian. Jika simpul yang lebih tinggi satu level merupakan MAX, maka akan diambil langkah dengan nilai tertinggi. Sedangkah jika MIN, maka akan diambil langkah dengan nilai terendah. Pada gambar 2.4 simpul paling kiri, MIN memiliki pilihan langkah, dan langkah terbaik untuk MIN adalah A11, karena A11 akan menghasilkan nilai yang paling minimum, yaitu 3. Dengan cara yang sama, dilakukan pengambilan langkah MIN untuk simpul yang lain, sehingga didapatkan nilai 2.
d. Melakukan hal yang sama dari simpul daun sampi ke simpul akar, satu tingkat pada satu waktu.
e. Pada akhirnya, nilai tersebut sampai pada simpul teratas, MAX memilih langkah yang menghasilkan nilai terbesar. Pada gambar 2.4, MAX memiliki 3 pilihan langkah dengan nilai 3, 2, dan 2. Sehingga langkah terbaik yang diambil oleh MAX adalah A1 dengan nilai 3. Inilah yang

disebut dengan *Minimax Decision*, karena langkah yang diambil merupakan langkah terbaik untuk meminimalkan langkah lawan, dengan asumsi lawan main bermain secara sempurna.



Gambar 2.4 Pohon Permainan Algoritma *Minimax*
Sumber : [RUS-95]

Tabel 2.1 *Pseudocode* Algoritma *Minimax*

```
1  function MINIMAX-DECISION(game) returns an operator
2    for each op in OPERATORS[game] do
3      VALUE[op] ←MINIMAX-VALUE(APPLY(op, game), game)
4    end
5  return the op with the highest VALUE[op]
6
7  function MINIMAX-VALUE(state, game) returns a utility value
8    if TERMINAL-TEST[game](state) then
9      return UTILITY[game](state)
10   else if MAX is to move in state then
11     return the highest MINIMAX-VALUE of SUCCESSORS(state)
12   else
13     return the lowest MINIMAX-VALUE of SUCCESSORS(state)
```

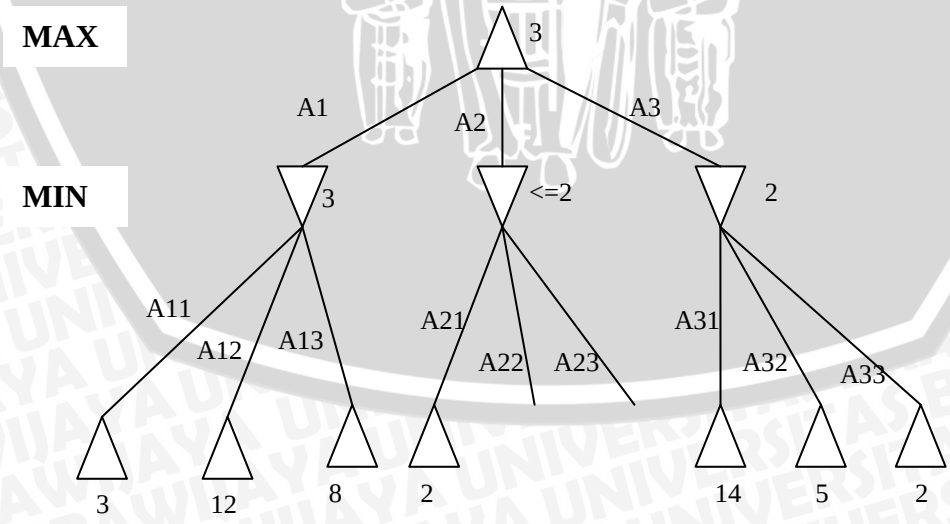
Sumber : [RUS-95]

2.4 Algoritma *Alpha-beta Pruning*

Algoritma *Alpha-beta pruning* merupakan optimasi dari algoritma *Minimax*. Algoritma ini akan menghasilkan langkah yang sama dengan langkah yang dihasilkan algoritma *Minimax,* dengan melakukan pemotongan cabang yang tidak akan mempengaruhi keputusan akhir yang diambil pada pohon *Minimax* [RUS-95].

Pada gambar 2.5 menunjukkan proses pemotongan cabang pada pohon permainan. Proses pencarian sama dengan proses sebelumnya, mulai dari A1, kemudian A11, A12, A13, sehingga didapatkan simpul dibawah A1 menghasilkan nilai *Minimax* 3. Kemudian dilanjutkan dengan A2, dan A21, yang memiliki nilai 2. Hal ini berarti nilai A2 mungkin bernilai 2, atau akan bernilai sesuai dengan nilai pada A22, dan A23. Karena A1 memiliki nilai 3 yang lebih besar dari nilai A2 saat ini yaitu 2, dapat dilakukan pemotongan cabang pada A22, dan A23 [RUS-95].

Secara umum, terdapat simpul *n* pada pohon permainan, dimana pemain mungkin melakukan langkah ke simpul tersebut. Jika pemain memiliki pilihan *m* yang lebih baik dari *n,* setelah dilakukan evaluasi pada beberapa keturunan simpul *n,* maka dapat dilakukan pemotongan cabang pada simpul *n* [RUS-95].



Gambar 2.5 Pohon Permainan Algoritma *Alpha-beta Pruning*
Sumber : [RUS-95]

Dalam algoritma *Alpha-beta Pruning*, terdapat dua nilai yaitu α, dan β. α merupakan nilai terbaik (maksimum) saat ini yang didapatkan untuk MAX, β merupakan nilai terbaik (minimum) saat ini yang didapatkan untuk MIN. Nilai α, dan β akan berubah seiring dengan pencarian *Alpha-beta,* dan akan dilakukan pemotongan cabang pada pohon permainan jika diketahui memiliki nilai yang lebih kecil dari nilai α atau β saat ini.

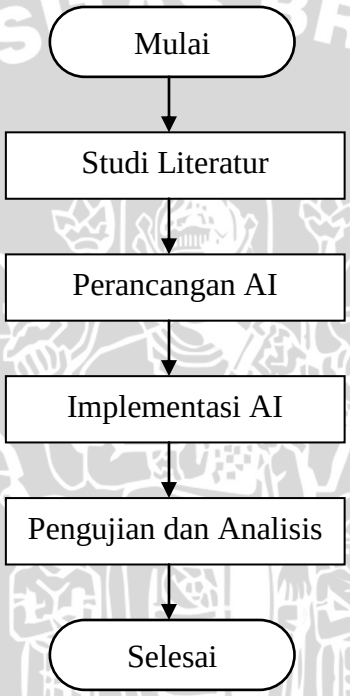
Tabel 2.2 *Pseudocode* Algoritma *Alpha-beta Pruning*

```
1  function MAX-VALUE(state, game, α, β) returns the minimax value of state
2    inputs: state, current state in game
3            game, game description
4            α, the best score for MAX along the path to state
5            β, the best score for MIN along the path to state
6    if CUTOFF-TEST(state) then return EVAL(state)
7    for each s in SUCCESSOR(state) do
8      α ← MAX(α, MIN-VALUE(s, game, α, β))
9      if α>β then return β
10   end
11   return α
12
13 function MIN-VALUE(state, game, α, β) returns the minimax value of state
14   if CUTOFF-TEST(state) then return EVAL(state)
15   for each s in SUCCESSORS(state) do
16     β ← MIN(β, MAX-VALUE(s, game, α, β))
17     if β < a then return a
18   end
19   return β
```

Sumber : [RUS-95]

BAB III
METODOLOGI DAN PERANCANGAN

Bab ini menjelaskan langkah-langkah yang ditempuh dalam penelitian. Penelitian ini akan dilakukan dalam beberapa tahap meliputi studi literatur, perancangan, implementasi, dan pengujian. Gambar 3.1 merupakan diagram alir metode yang digunakan dalam penelitian ini.



Gambar 3.1 Diagram Alir Metode Penelitian

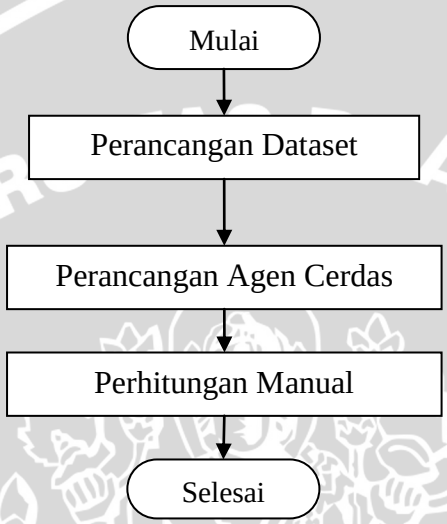
3.1 Studi Literatur

Metode ini digunakan untuk mendapatkan dasar teori sebagai sumber acuan untuk penulisan skripsi. Teori dan pustaka yang berkaitan dengan tugas akhir ini meliputi :

1. Permainan Ntil
    a. Bidak dan Papan Permainan Ntil
    b. Aturan Permainan Ntil
2. Algoritma *Minimax*
3. Algoritma *Alpha-Beta Pruning*

3.2 Perancangan AI

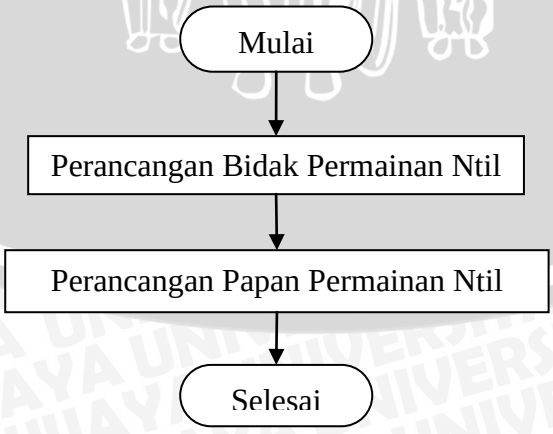
Tahap ini dilakukan untuk merancang hal-hal yang dibutuhkan untuk membangun sebuah AI dalam permainan ntil. Perancangan AI dibagi menjadi dua, yaitu perancangan dataset dan perancangan agen cerdas. Gambar 3.2 merupakan diagram alir perancangan.



Gambar 3.2 Diagram Alir Perancangan AI

3.2.1 Perancangan Dataset

Perancangan dataset meliputi perancangan bidak, papan permainan, Gambar 3.3 merupakan diagram alir perancangan dataset.



Gambar 3.3 Diagram Alir Perancangan Dataset

1. Perancangan Bidak Permainan Ntil

Tahap ini dilakukan untuk merancang bidak dalam permainan ntil. Hal ini perlu dilakukan untuk membedakan bidak biasa pemain dengan lawan main, dan bidak raja pemain dengan lawan main. Tabel 3.1 merupakan tabel atribut bidak permainan Ntil.

Tabel 3.1 Perancangan Atribut Bidak Permainan Ntil

Atribut	Keterangan
Id	Id digunakan untuk membedakan antara bidak biasa, dan bidak raja. Serta untuk membedakan bidak pemain A, dan bidak pemain B.
Nilai	Nilai bidak dibedakan antara bidak biasa, dan bidak raja. Untuk bidak biasa, masing-masing bernilai 3,5, dan 7 jika bidak tersebut berada pada daerah pemain itu sendiri, daerah tengah papan, dan daerah lawan. Sedangkan nilai bidak raja tetap yaitu 12.

2. Perancangan Papan Permainan Ntil

Papan permainan ntil akan dibentuk dalam sebuah *graph*. Setiap tititk pada papan merupakan *vertex*, dan garis penghubung setiap titik merupakan *edge*. Titik-titik tersebut akan menyimpan dua nilai yaitu : posisi bidak dan bobot dari titik tersebut. Setiap titik memiliki bobot nilai yang berbeda, tergantung dari posisi titik. Tabel 3.2 merupakan tabel bobot nilai titik pada papan permainan ntil.

Tabel 3.2 Perancangan Bobot Nilai Titik pada Papan Permainan Ntil

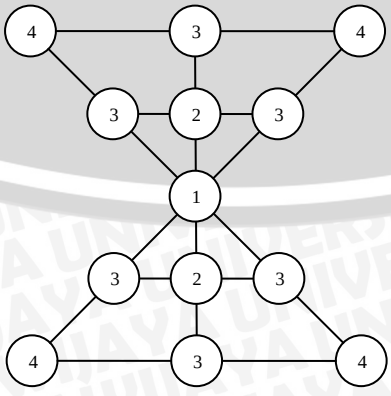
Bobot nilai	Keterangan
1	Merupakan titik dimana bidak bisa dimakan dari 6 atau 8 arah
2	Merupakan titik dimana bidak bisa dimakan dari 4 arah
3	Merupakan titik dimana bidak bisa dimakan dari 2 arah
4	Merupakan titik dimana bidak tidak bisa dimakan

Bobot titik pada papan permainan ntil ditentukan berdasarkan posisi titik pada papan permainan. Pada tabel 3.2 menunjukkan bobot titik pada papan

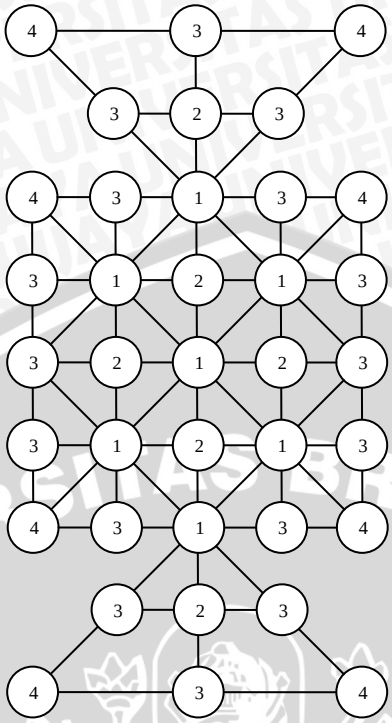
permainan bernilai semakin besar, jika bidak berada pada titik tersebut memiliki kemungkinan yang lebih kecil untuk dimakan oleh lawan. Hal ini bertujuan agar agen memilih langkah ketitik yang sulit dimakan oleh lawan, sehingga agen mampu mepertahankan jumlah bidaknya.

Papan permainan ntil akan dibuat dalam beberapa ukuran, diantaranya :

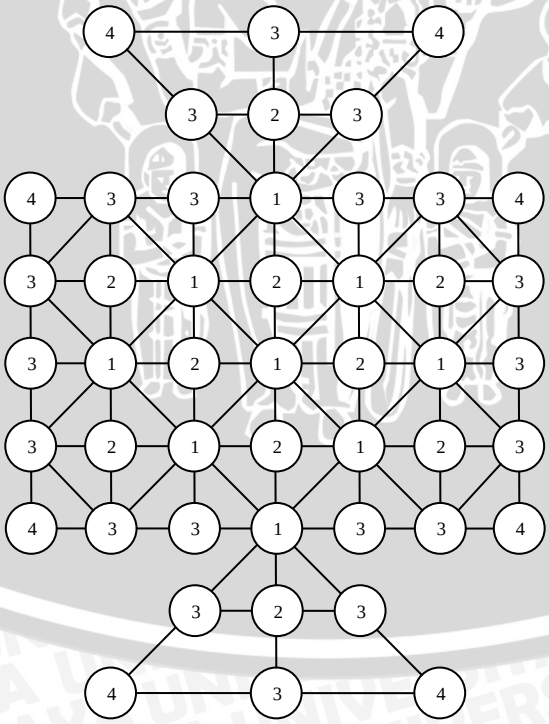
1. Papan permainan yang berukuran 5x5 memiliki 13 titik, dan masing masing bidak pemain berjumlah 6. Gambar 3.4 menunjukkan papan permainan Ntil berukuran 5x5.
2. Papan permainan yang berukuran 5x9 memiliki 37 titik, dan masing masing bidak pemain berjumlah 16. Gambar 3.5 menunjukkan papan permainan Ntil berukuran 5x9.
3. Papan permainan yang berukuran 7x9 memiliki 47 titik, dan masing masing bidak pemain berjumlah 20. Gambar 3.6 menunjukkan papan permainan Ntil berukuran 5x9.
4. Papan permainan yang berukuran 9x9 memiliki 57 titik, dan masing masing bidak pemain berjumlah 24. Gambar 3.7 menunjukkan papan permainan Ntil berukuran 5x9.
5. Papan permainan yang berukuran 7x11 memiliki 61 titik, dan masing masing bidak pemain berjumlah 27. Gambar 3.8 menunjukkan papan permainan Ntil berukuran 5x9.
6. Papan permainan yang berukuran 9x11 memiliki 75 titik, dan masing masing bidak pemain berjumlah 33. Gambar 3.9 menunjukkan papan permainan Ntil berukuran 5x9.



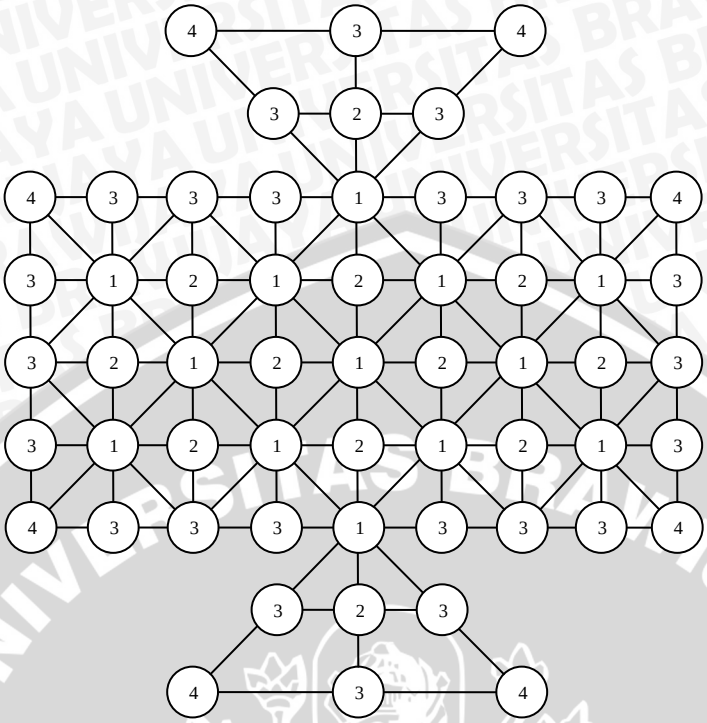
Gambar 3.4 Perancangan Papan Permainan Ntil Berukuran 5x5



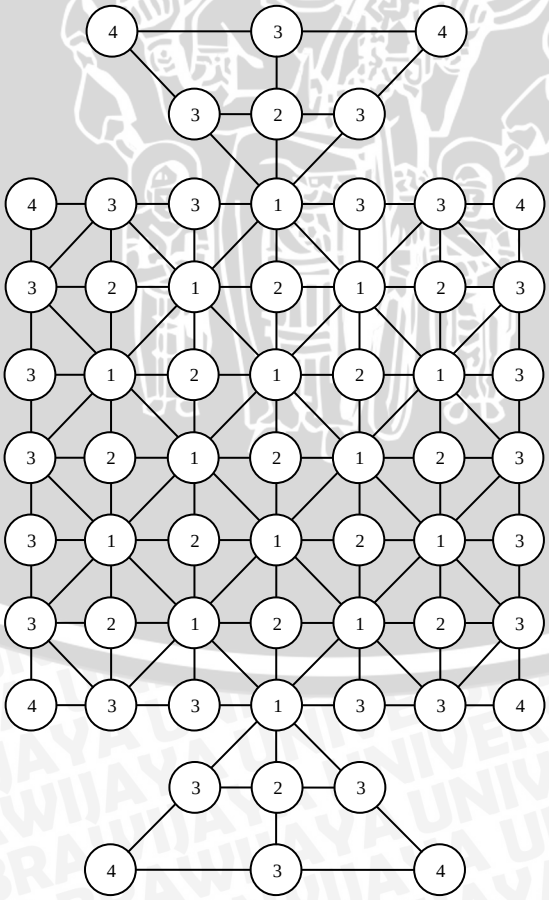
Gambar 3.5 Perancangan Papan Permainan Ntil Berukuran 5x9



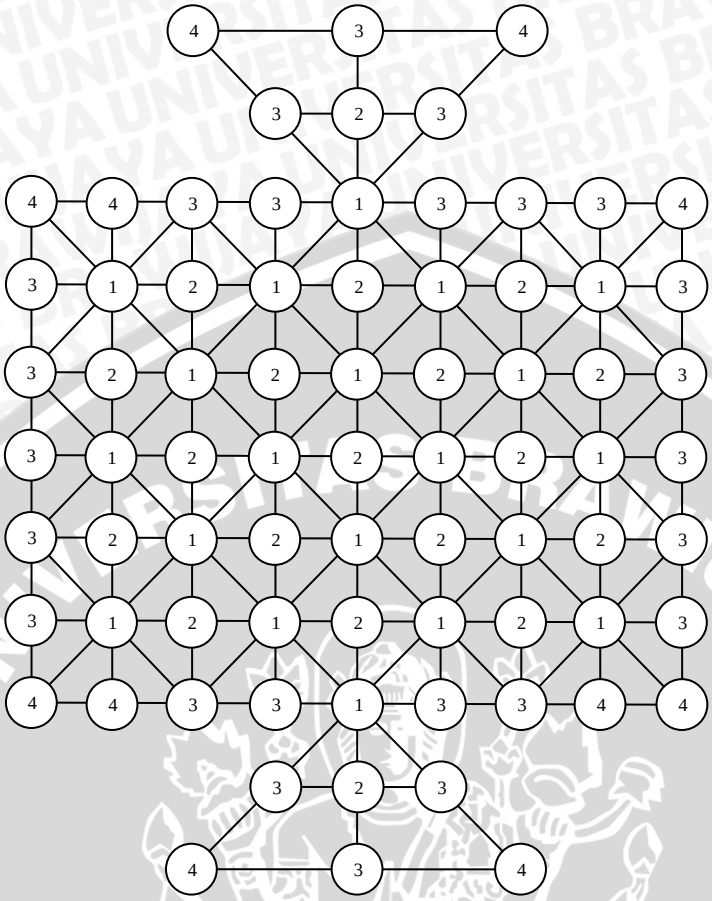
Gambar 3.6 Perancangan Papan Permainan Ntil Berukuran 7x9



Gambar 3.7 Perancangan Papan Permainan Ntil Berukuran 9x9



Gambar 3.8 Perancangan Papan Permainan Ntil Berukuran 7x11

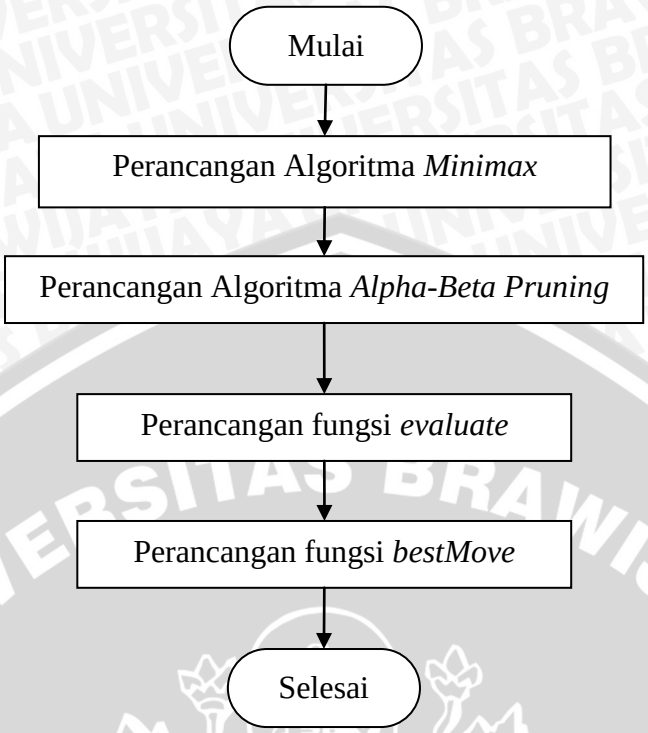


Gambar 3.9 Perancangan Papan Permainan Ntil Berukuran 9x11

Pada gambar 3.4 sampai gambar 3.9 merupakan gambar perancangan papan permainan Ntil dengan ukuran yang berbeda-beda, beserta dengan bobot nilai titik pada papan permainan. Pada setiap papan permainan memiliki persamaan yaitu papan permainan dibentuk oleh 2 segitiga yang memiliki 6 titik, dan persegi atau persegi panjang, yang kemudian dihubungkan dengan garis vertikal, horizontal, diagonal kanan, dan diagonal kiri. Sedangkan, untuk papan berukuran 5x5 hanya dibentuk dengan 2 segitiga.

3.2.2 Perancangan Agen Cerdas

Perancangan agen cerdas dilakukan dalam empat tahap, yaitu : Perancangan algoritma *Minimax*, algoritma *Alpha-beta Pruning*, fungsi *evaluate*, dan fungsi *bestMove*. Gambar 3.10 merupakan diagram alir perancangan agen cerdas.



Gambar 3.10 Diagram Alir Perancangan Agen Cerdas

1. Perancangan Algoritma *Minimax*

Proses pencarian langkah pada algoritma ini, dimulai dengan pembuatan pohon permainan. Simpul pada pohon permainan, menyimpan kondisi papan permainan (*board*). Simpul akar akan dibangkitkan terlebih dahulu, kemudian algoritma ini akan membangkitkan simpul pada tingkat kedua, yang akan dilanjutkan pada tingkat ketiga, dan seterusnya, sampai simpul terdalam (*depth==maxDepth*).

Setiap simpul terdalam akan dievaluasi oleh fungsi *evaluation* untuk mendapatkan nilainya. Nilai pada simpul yang berada satu level lebih tinggi pada pohon pencarian, didapatkan dengan membandingkan nilai dari anak simpul tersebut. Jika simpul yang lebih tinggi merupakan kemungkinan langkah yang diambil oleh agen (*maximazingPlayer==true*), maka nilai dari simpul tersebut adalah nilai maksimum dari hasil perbandingan nilai anak simpul tersebut. Jika simpul yang lebih tinggi merupakan kemungkinan langkah yang diambil oleh lawan main dari agen (*maximazingPlayer==false*), maka nilai dari simpul tersebut adalah nilai minimum dari hasil perbandingan nilai anak simpul tersebut.

Hal ini dilakukan secara rekursif sampai didapatkan nilai pada simpul akar. Gambar 3.11 merupakan diagram alir perancangan algoritma *Minimax*.

2. Perancangan Algoritma *Alpha-Beta Pruning*

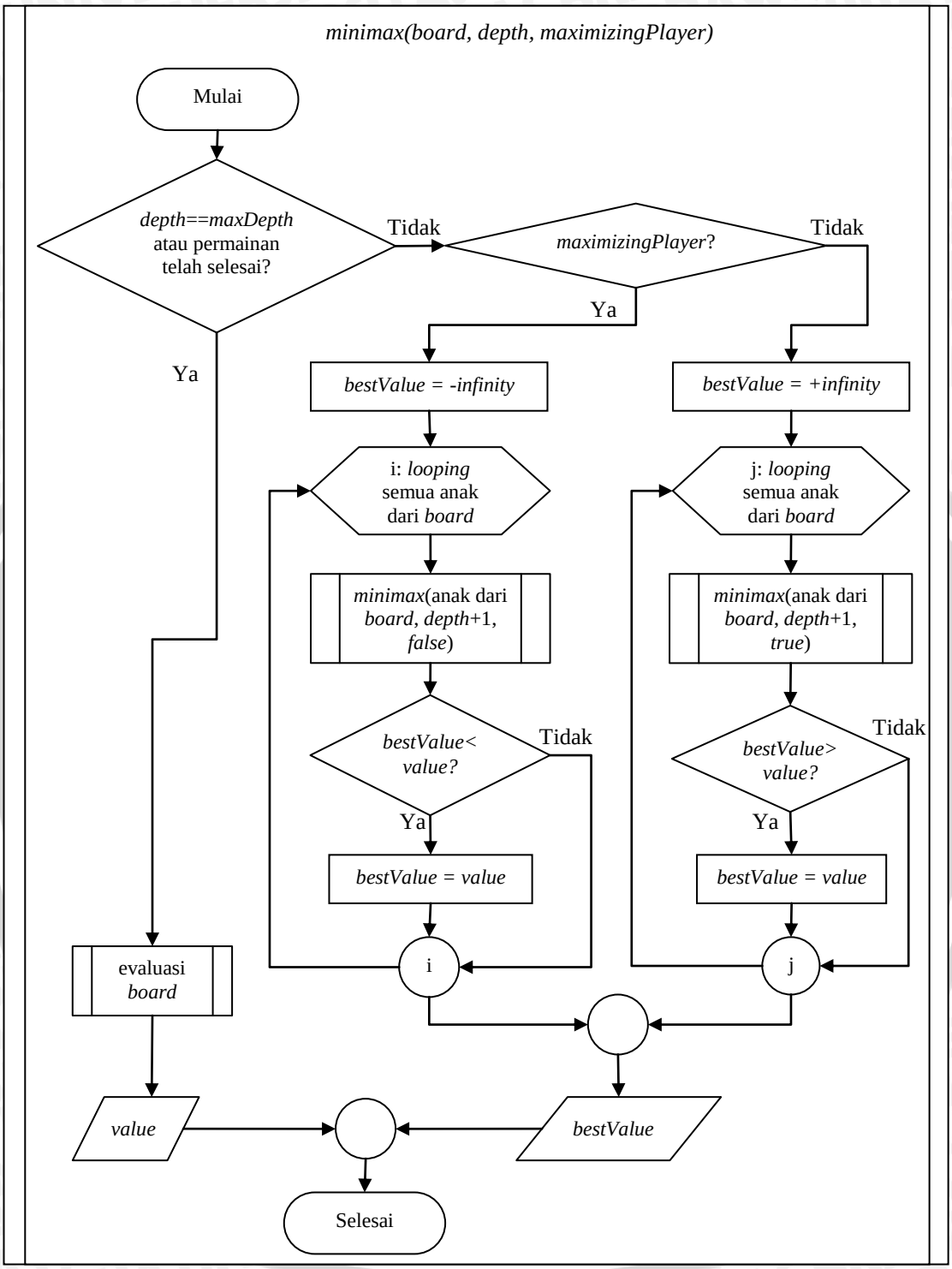
Secara umum, proses pencarian langkah pada *Alpha-Beta Pruning* sama dengan *Minimax*. Namun pada algoritma *Alpha-beta Pruning*, terdapat pemotongan cabang pada pohon permainan. Hal ini dilakukan jika nilai *alpha* (nilai maksimum untuk agen saat ini) lebih besar atau sama dengan nilai *beta* (nilai minimum untuk lawan main saat ini). Gambar 3.12 merupakan diagram alir perancangan algoritma *Alpha-beta Pruning*.

3. Perancangan Fungsi *evaluate*

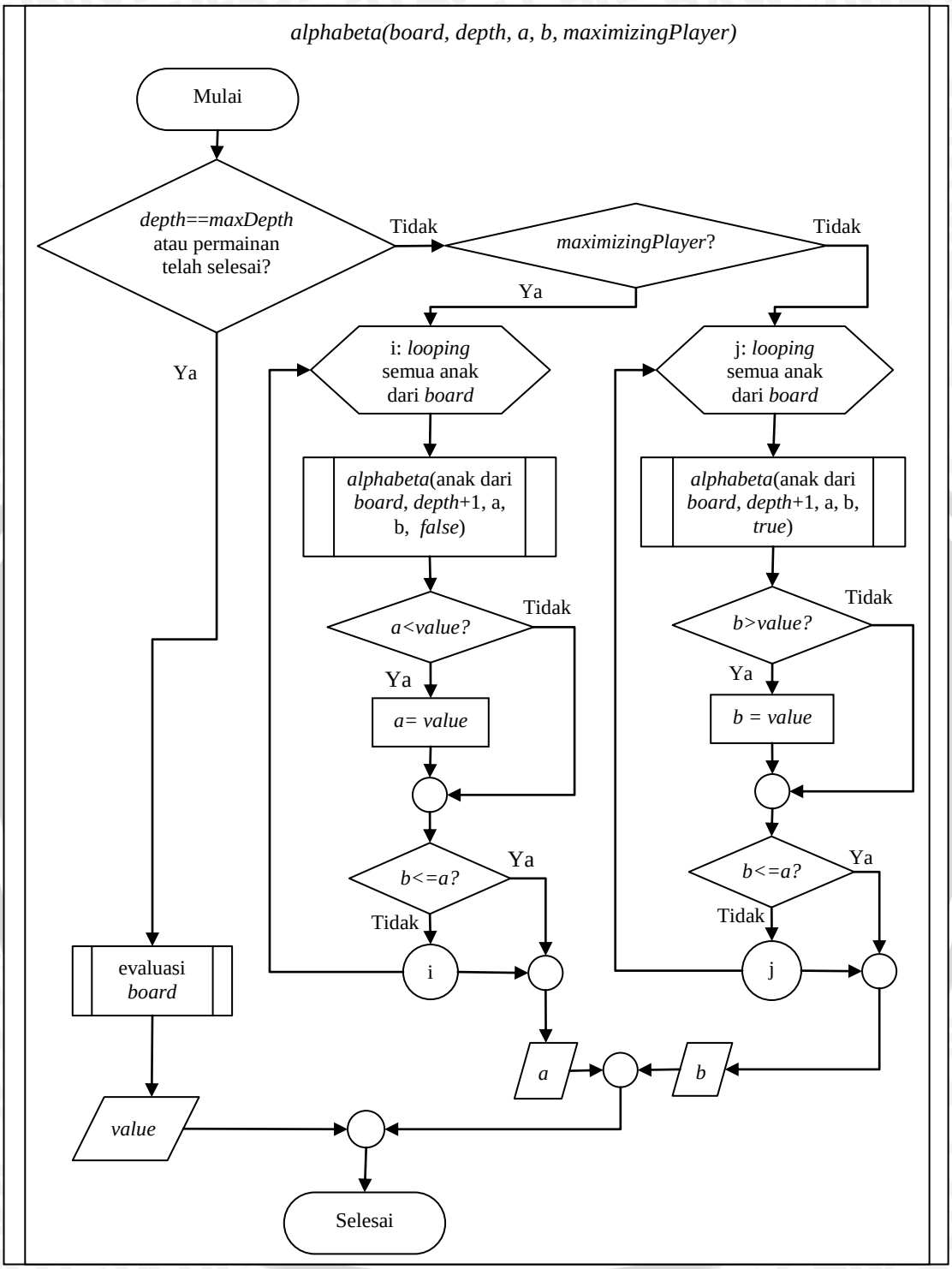
Fungsi *evaluate* digunakan untuk mengevaluasi kondisi papan permainan Ntil. Papan permainan dievaluasi berdasarkan nilai pemain dikurangi nilai lawan main (*playerValue-enemyValue*). Nilai pemain merupakan jumlah dari setiap bidak pemain pada titik ke-i dikali bobot nilai titik ke-i. Sedangkan nilai lawan main merupakan jumlah dari setiap bidak lawan pada titik ke-i dikali bobot nilai titik ke-i. Gambar 3.13 merupakan diagram alir perancangan fungsi *evaluate*.

4. Perancangan Fungsi *bestMove*

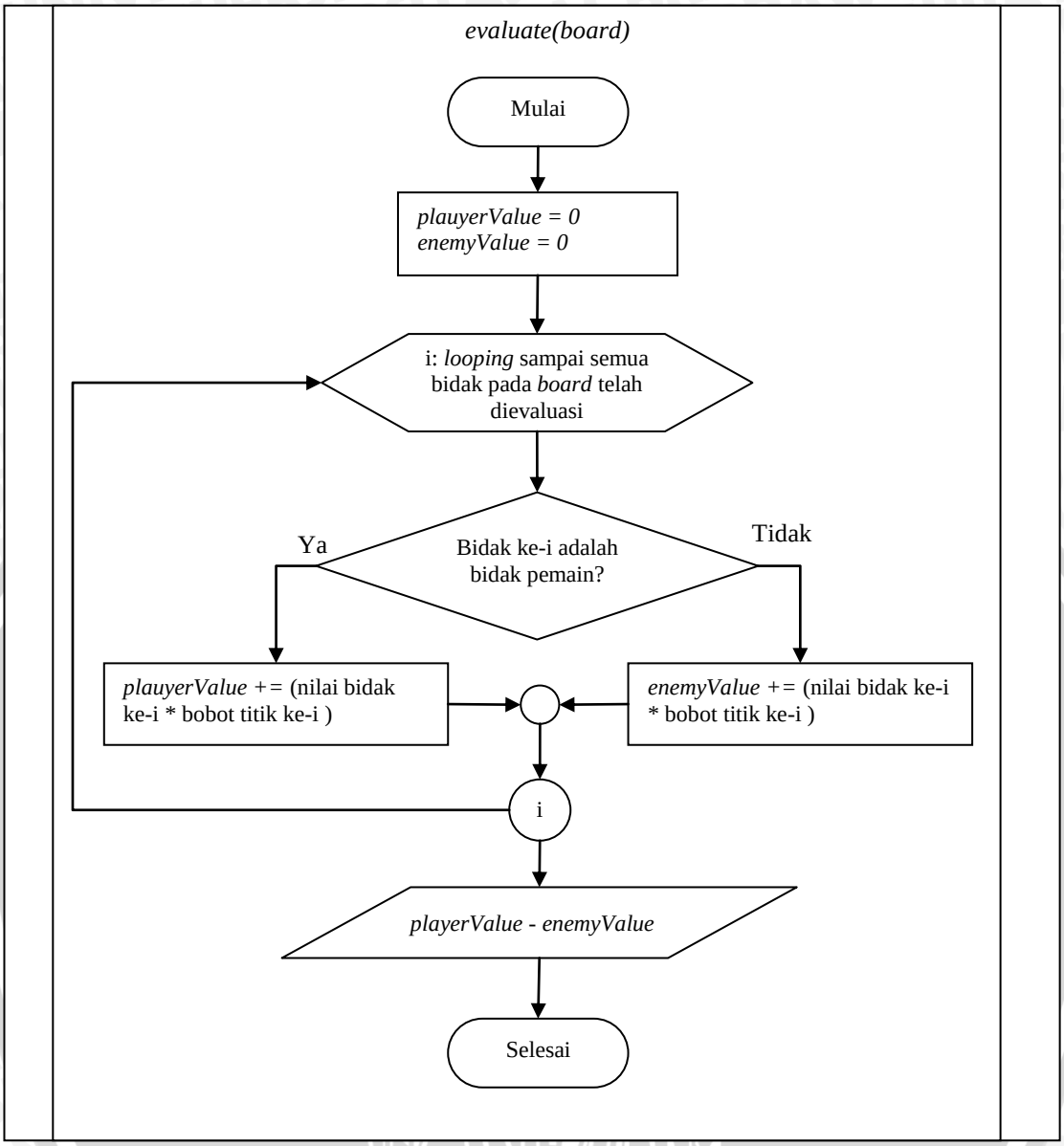
Fungsi ini diawali dengan pencarian semua kemungkinan langkah bidak agen, yang nantinya akan dijadikan simpul akar pada pohon permainan yang akan dibangkitkan oleh *Minimax* (*state*==1), maupun *Alpha-beta Pruning* (*state!*=1). Setelah dievaluasi oleh algoritma tersebut, akan didapatkan nilai pada setiap simpul. Simpul dengan nilai tertinggi merupakan langkah terbaik yang akan diambil oleh agen. Gambar 3.14 merupakan diagram alir perancangan fungsi *bestMove*.



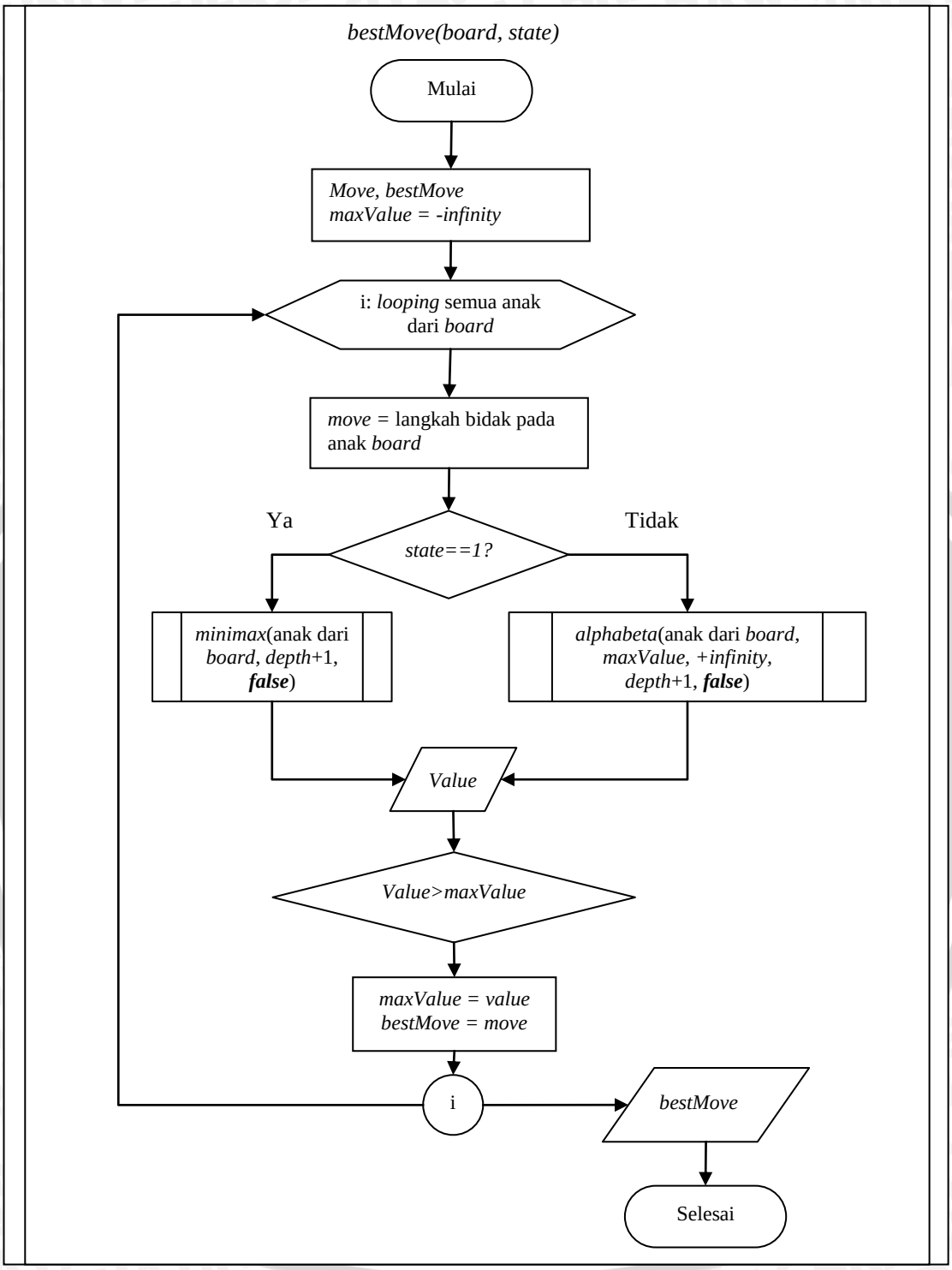
Gambar 3.11 Diagram Alir Perancangan Algoritma *Minimax*



Gambar 3.12 Diagram Alir Perancangan Algoritma *Alpha-beta Pruning*



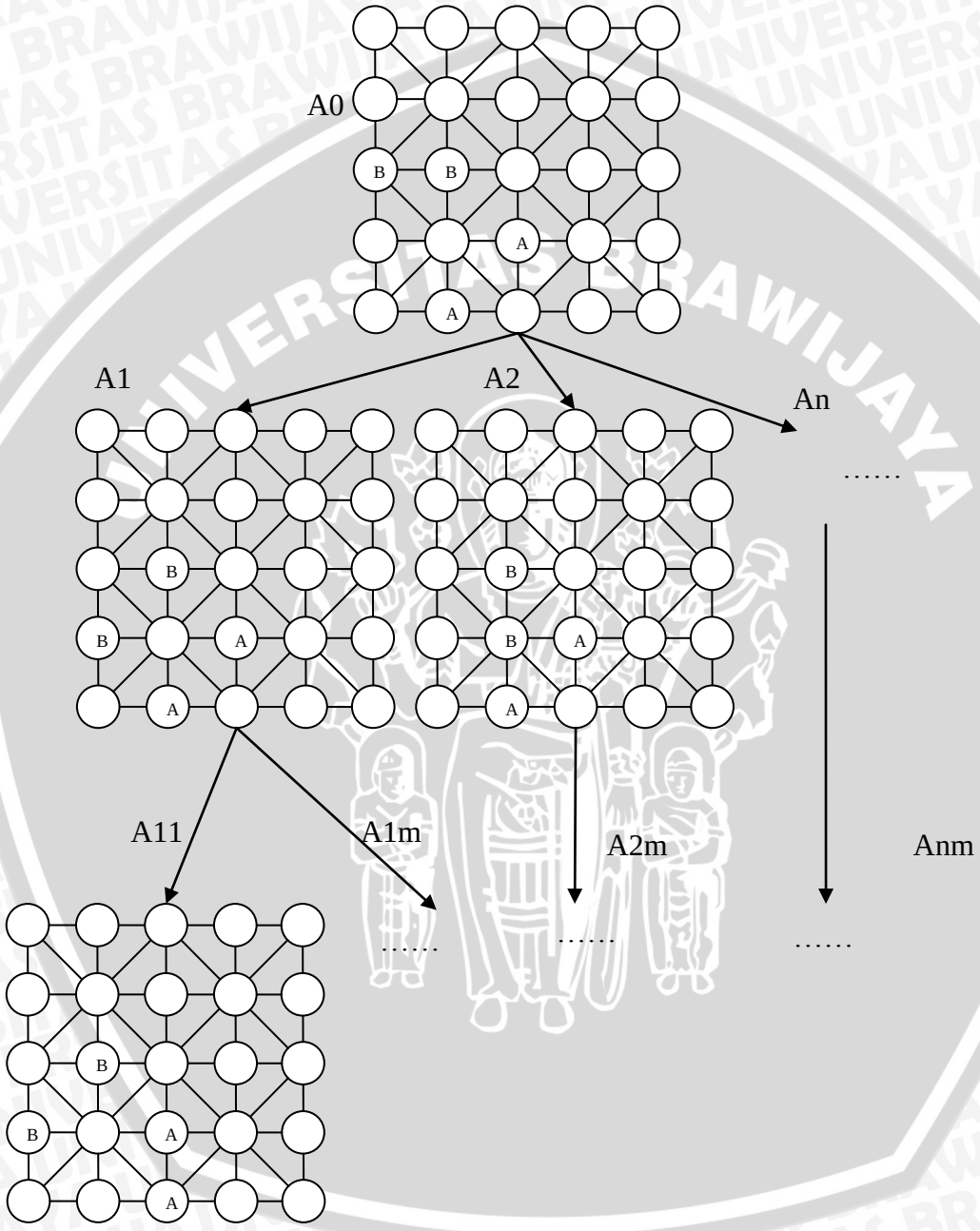
Gambar 3.13 Diagram Alir Perancangan Fungsi *evaluate*



Gambar 3.14 Diagram Alir Perancangan Fungsi *bestMove*

3.2.3 Perhitungan Manual

Tahap ini dilakukan untuk melakukan perhitungan manual, pada simpul terdalam pohon permainan yang dihasilkan.

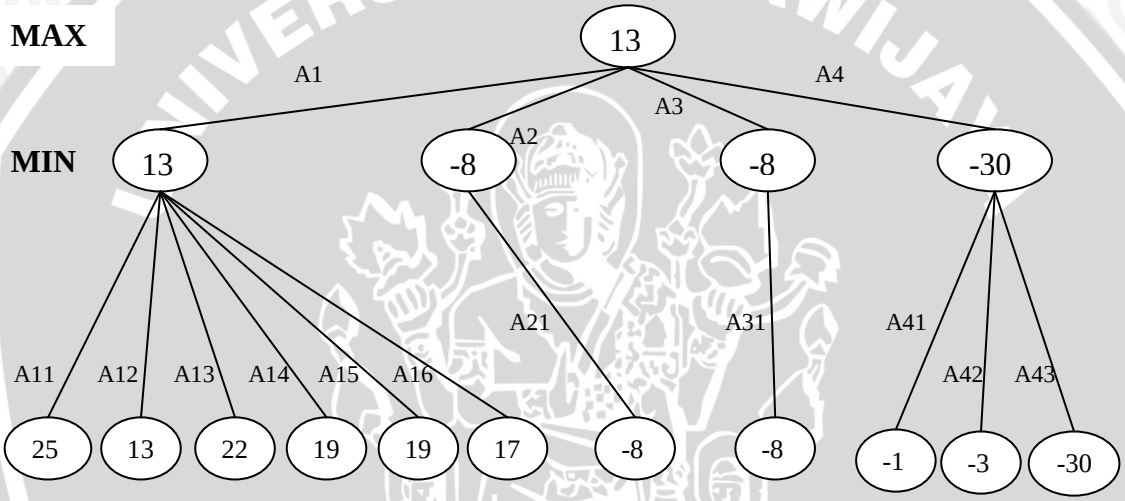


Gambar 3.15 Pencarian Langkah Terbaik oleh Agen B dengan *depth=2*

Pada gambar 3.15, A0 merupakan kondisi awal papan permainan. A1, A2, dan An merupakan semua kemungkinan langkah bidak yang akan diambil oleh agen B. Setelah didapatkan kemungkinan langkah dari agen B, maka dilakukan

pencarian kemungkinan langkah yang akan diambil oleh lawan dari B, yaitu A. A11, dan A1m merupakan kemungkinan langkah yang akan diambil oleh A, jika agen B mengambil langkah A1. A2m merupakan kemungkinan langkah yang akan diambil oleh A jika agen B mengambil langkah A2. Anm merupakan kemungkinan langkah yang akan diambil oleh A jika agen B mengambil langkah An.

Setelah didapatkan kemungkinan langkah yang akan diambil oleh agen B, maka dilakukan evaluasi pohon permainan dengan algoritma *Minimax*, maupun dengan algoritma *Alpha-beta Pruning*.

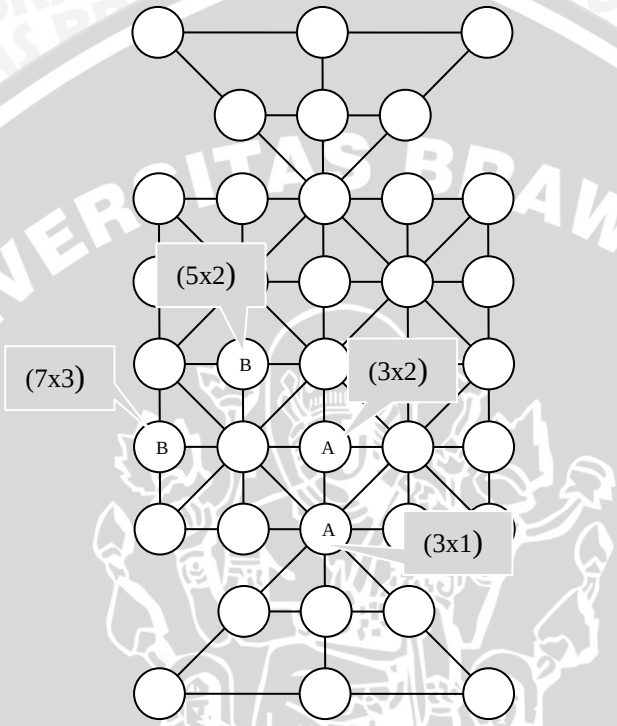


Gambar 3.16 Pohon permainan algoritma *Minimax*

Pada gambar 3.16 menunjukkan pohon permainan yang dievaluasi menggunakan algoritma *Minimax*. Algoritma ini akan mengevaluasi simpul terdalam pada pohon permainan terlebih dahulu. Evaluasi dilakukan berdasarkan nilai pemain dikurangai nilai lawan main, seperti yang dijelaskan pada fungsi *evaluate* sebelumnya.

Pada gambar 3.17 A11 merupakan simpul terdalam yang akan dievaluasi terlebih dahulu. B merupakan bidak pemain (agen), dan A merupakan bidak lawan main dari agen. Agen memiliki 2 bidak biasa, yang terletak pada bagian tengah papan permainan (nilai bidak = 5) dengan bobot titik bernilai 2, dan terletak pada daerah lawan (nilai bidak = 7) dengan bobot titik bernilai 3. Sehingga didapatkan nilai pemain (agen), yaitu (5x2) + (7x3) sama dengan 31. Lawan dari agen

memiliki 2 bidak biasa, yang terletak pada daerahnya sendiri (nilai bidak = 3) dengan bobot titik bernilai 1, dan 2. Sehingga didapatkan nilai untuk lawan main, yaitu (3x1) + (2x3) sama dengan 9. Hasil evaluasi pada simpul A11 yaitu nilai pemain dikurangi nilai lawan main (31 - 9), sehingga didapatkan nilai 25. Dengan cara yang sama, dilakukan evaluasi pada simpul terdalam lainnya.

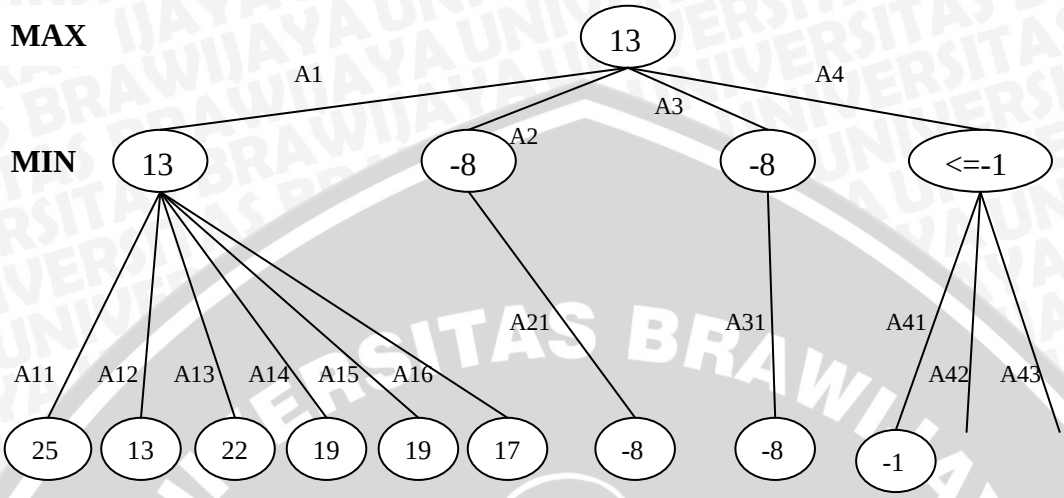


Gambar 3.17 Evaluasi Papan Permainan pada Simpul A11

Jika simpul yang lebih tinggi satu level merupakan MAX, maka akan diambil langkah dengan nilai tertinggi. Sedangkah jika MIN, maka akan diambil langkah dengan nilai terendah. MAX adalah agen yang langkahnya dimaksimalkan, sedangkan MIN adalah lawan dari agen yang langkahnya akan diminimalkan [RUS-95].

Pada gambar 3.16 simpul paling kiri, diambil langkah dengan hasil evaluasi paling minimal yaitu 13, sehingga langkah terbaik yang akan diambil oleh MIN adalah A12. Dengan cara yang sama, dilakukan pengambilan langkah MIN untuk simpul yang lain, sehingga didapatkan nilai -8, -8, dan -30. Setelah didapatkan nilai untuk MIN, maka pencarain diteruskan ke MAX yang akan mengambil nilai maksimal dari hasil evaluasi MIN. MAX memiliki 4 pilihan

langkah, dengan nilai 13, -8, -8, dan -30. Sehingga langkah terbaik yang diambil oleh MAX atau agen adalah A1 dengan nilai 13.



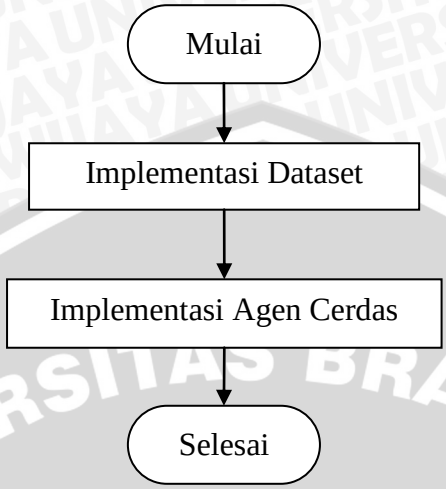
Gambar 3.18 Pohon permainan algoritma *Alpha-beta Pruning*

Pada gambar 3.18 menunjukkan pohon permainan yang dievaluasi menggunakan algoritma *Alpha-beta Pruning*. Pada gambar tersebut terlihat dilakukan pemotongan cabang pada pohon permainan. Proses pencarian sama dengan proses sebelumnya, mulai dari A1, kemudian A11-A16 sehingga didapatkan nilai untuk MIN yaitu 13. Kemudian dilanjutkan dengan A2, dan A3 didapatkan nilai yang sama yaitu -8, dimana A2 dan A3 hanya memiliki satu kemngukinan langkah. Kemudian dilanjutkan dengan A4, dan A41, yang memiliki nilai -1. Hal ini berarti nilai A4 mungkin bernilai -1, atau akan bernilai sesuai dengan nilai pada A42, dan A43. Karena A1 memiliki nilai 13 yang lebih besar dari nilai A4 saat ini yaitu -1, sehingga dapat dilakukan pemotongan cabang pada A42, dan A43. Sehingga didapatkan langkah terbaik yang akan diambil oleh MAX atau agen yaitu A1 dengan nilai 13.

3.3 Implementasi

Implementasi dilakukan sesuai dengan perancangan yang telah dibuat. Objek bidak dan papan permainan akan dibuat dalam model 3D. Sedangkan untuk implementasi permainan ntil, algoritma *Minimax* dan *Alpha-beta Pruning* menggunakan *game engine* Unity 4.5.4 dengan bahasa pemrograman C# dengan

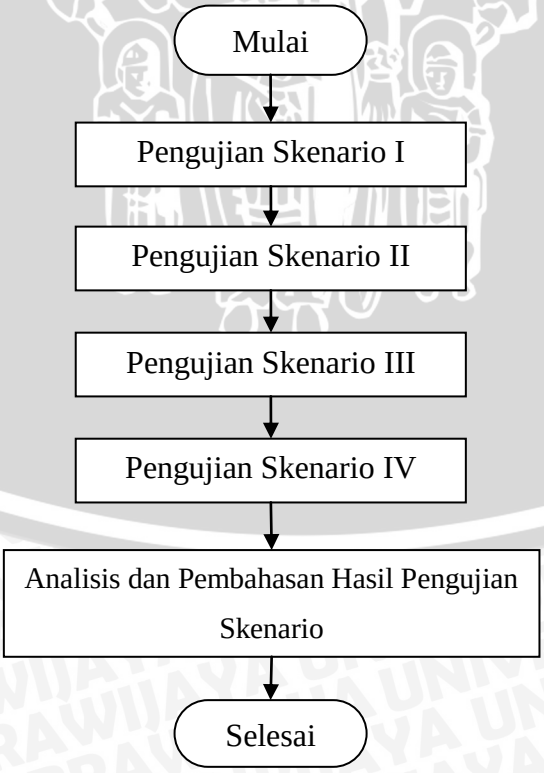
IDE MonoDevelop. Gambar 3.19 merupakan diagram alir tahapan yang akan dilakukan dalam proses implementasi.



Gambar 3.19 Diagram Alir Tahapan Implementasi

3.4 Pengujian dan Analisis

Pengujian yang dilakukan berdasarkan implementasi permainan yang telah dibuat. Gambar 3.20 merupakan diagram alir tahapan yang akan dilakukan dalam proses pengujian dan analisis.



Gambar 3.20 Diagram Alir Tahapan Pengujian dan Analisis

3.4.1 Pengujian

Pengujian akan dilakukan dengan pembuatan beberapa skenario. Skenario yang akan dibuat ditentukan dari jumlah kedalaman pencarian, tipe papan permainan, dan algoritma yang akan digunakan AI. Pada setiap skenario akan dicatat waktu (ms) dan jumlah simpul yang diekspansi ketika AI melakukan pencarian langkah terbaik.

3.4.2. Analisis

Hasil pengujian tiap uji coba per skenario akan dibedakan berdasarkan jumlah kedalaman pencarian, tipe papan permainan, dan algoritma yang akan digunakan AI. Hasil rata-rata waktu pencarian dengan algoritma *Minimax,* dan *Alpha-beta Pruning* akan dibandingkan untuk mengetahui perbedaan kinerja dari ke dua algoritma tersebut.

BAB IV
IMPLEMENTASI

4.1 Lingkungan Implementasi

Bagian ini berisi penjelasan tentang lingkungan implementasi berupa peralatan yang digunakan dalam pembuatan permainan. Dalam penulisan tugas akhir ini, lingkungan implementasi dibedakan menjadi 2 jenis, yaitu: Lingkungan perangkat keras, dan lingkungan implementasi perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam pembuatan permainan ini dijelaskan dalam Tabel 4.1.

Tabel 4.1 Spesifikasi Perangkat Keras Implementasi

Perangkat Keras	Spesifikasi
Prosesor	Intel(R) Core(TM) i3
Memori (RAM)	4 GB
Kartu Grafis	Intel(R) HD Graphics

4.1.2 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pembuatan permainan ini dijelaskan dalam Tabel 4.2.

Tabel 4.2 Spesifikasi Perangkat Lunak Implementasi

Perangkat Lunak	Spesifikasi
Operating System	*Microsoft windows 8.1 Pro 3- bit*
Game engine	Unity 4.5.5.f1
Bahasa pemrograman	C#
Integrated Development Environmet	MonoDevelop 4.0.1
3D *modeler*	Unity 4.5.5.f1

4.2 Batasan Implementasi

Agen cerdas permainan Ntil yang dibuat memiliki batasan-batasan implementasi sebagai berikut :

a. Agen cerdas menerapkan algoritma *Minimax*, dan *Alpha-beta Pruning* untuk melakukan pencarian langkah bidak.
b. Tidak akan terjadi Ntil pada agen cerdas permainan Ntil.

4.3 Implementasi Dataset

Tahap ini mengimplementasikan rancangan dataset yang telah dibuat berupa nilai bidak, model 3D bidak, template papan permainan yang menyimpan bobot nilai titik, ukuran papan permainan, posisi awal bidak, dan model 3D papan permainan yang dibangkitkan dari template papan permainan.

4.3.1 Implementasi Bidak Permainan Ntil

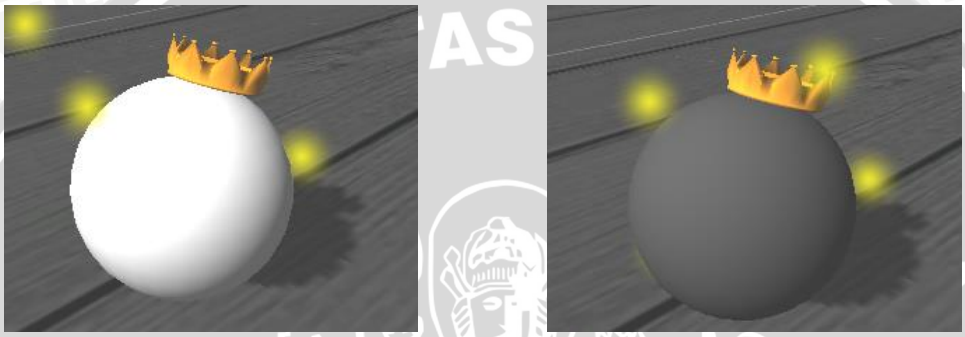
Tabel 4.3 merupakan tabel implementasi bidak permainan Ntil. Gambar 4.1, dan 4.2 merupakan *screenshot* model 3D bidak permainan Ntil.

Tabel 4.3 Implementasi Bidak Permainan Ntil

Id	Bidak	Nilai
1	Bidak biasa putih	- Bernilai 3 jika bidak berada pada daerah pemain itu sendiri. - Bernilai 5 jika bidak berada pada daerah tengah papan permainan. - Bernilai 7 jika bidak berada pada daerah lawan.
-1	Bidak biasa hitam	
2	Bidak raja putih	Bernilai 12
-2	Bidak raja hitam	

(a) (b)

Gambar 4.1 *Screenshot* Model 3D Bidak Biasa Permainan Ntil



(a) (b)

Gambar 4.2 *Screenshot* Model 3D Bidak Raja Permainan Ntil

4.3.2 Implementasi Papan Permainan Ntil

Papan permainan ntil diimplementasikan dalam sebuat template papan permainan yang disimpan dalam bentuk **.txt*. Template-template tersebut akan dibangkitkan kedalam model 3D. Tabel 4.4 merupakan tabel template papan permainan. Gambar 4.2 sampai 4.7 menunjukkan model 3D papan permainan.

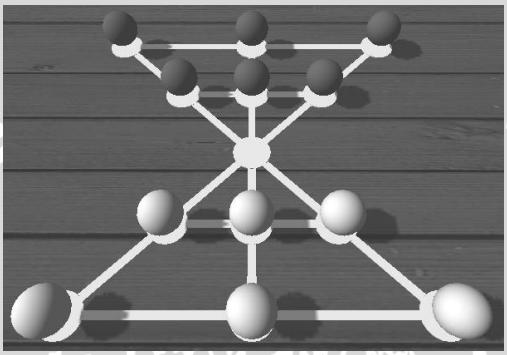
Tabel 4.4 Template Papan Permainan Ntil

No	Nama file	Implementasi template
1	5-5.txt	4b-x-5bk-x-4b \+ |+ /+ x3b-2b-3bx \ | /+ + xx1exx /+|+\+ x3w-2w-3wx /++ |++ \ 4w-x-5wK-x-4w

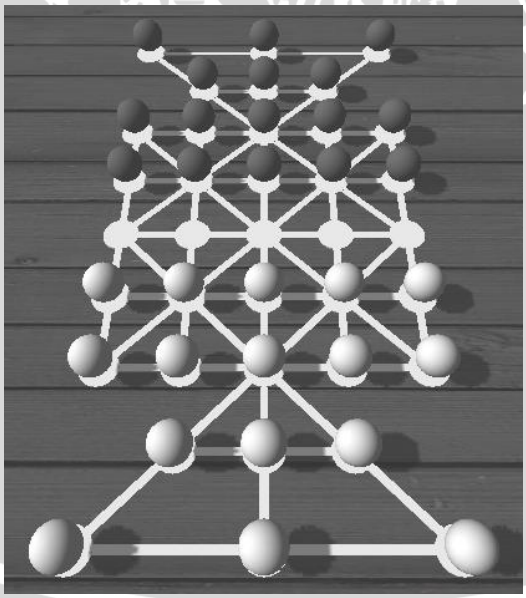
2	5-9.txt	4b-x-5bk-x-4b \+ |+ /+ x3b-2b-3bx \ | /+ + 4b-3b-1b-3b-4b | + |/+|+|/+| 3b-1b-2b-1b-3b |/+|+|/+|+| 3e-2e-1e-2e-3e |+|/+|+|/+| 3w-1w-2w-1w-3w |/+|+|/+|+| 4w-3w-1w-3w-4w /+|+\+ x3w-2w-3wx /++ |++ \ 4w-x-5wK-x-4w
3	7-9.txt	x4b-x-3bk-x-4bx \+ |+ /++ xx3b-2b-3bxx \ | /+++ 4b-3b-3b-1b-3b-3b-4b |/+|+|/+|+|/+|+| 3b-2b-1b-2b-1b-2b-3b |+|/+|+|/+|+|/+| 3e-1e-2e-1e-2e-1e-3e |/+|+|/+|+|/+|+| 3w-2w-1w-2w-1w-2w-3w |+|/+|+|/+|+|/+| 4w-3w-3w-1w-3w-3w-4w /+|+\++ xx3w-2w-3wxx /++ |++ \+ x4w-x-3wK-x-4wx
4	9-9.txt	xx4b-x-3bk-x-4bxx \+ |+ /+++ xxx3b-2b-3bxxx \ | /++++ 4b-3b-3b-3b-1b-3b-3b-3b-4b |+|/+|+|/+|+|/+|+|/+| 3b-1b-2b-1b-2b-1b-2b-1b-3b |/+|+|/+|+|/+|+|/+|+|

		3e-2e-1e-2e-1e-2e-1e-2e-3e |+|/+|+|/+|+|/+|+|/+| 3w-1w-2w-1w-2w-1w-2w-1w-3w |/+|+|/+|+|/+|+|/+|+| 4w-3w-3w-3w-1w-3w-3w-3w-4w /+|+\+++ xxx3w-2w-3wxxx /++ |++ \++ xx4w-x-3wK-x-4wxx
5	7-11.txt	x4b-x-3bk-x-4bx \+ |+ /++ xx3b-2b-3bxx \ | /+++ 4b-3b-3b-1b-3b-3b-4b |/+|+|/+|+|/+|+| 3b-2b-1b-2b-1b-2b-3b |+|/+|+|/+|+|/+| 3b-1b-2b-1b-2b-1b-3b |/+|+|/+|+|/+|+| 3e-2e-1e-2e-1e-2e-3e |+|/+|+|/+|+|/+| 3w-1w-2w-1w-2w-1w-3w |/+|+|/+|+|/+|+| 3w-2w-1w-2w-1w-2w-3w |+|/+|+|/+|+|/+| 4w-3w-3w-1w-3w-3w-4w /+|+\++ xx3w-2w-3wxx /++ |++ \+ x4w-x-3wK-x-4wx
6	9-11.txt	xx4b-x-3b-x-4bxx \+ |+ / + + + xxx3b- 2b-3bxxx \ | / + + + + 4b-3b-3b-3b-1b-3b-3b-3b-4b |+\ |/+|+|/+|+|/+|+|/+| 3b-1b-2b-1b-2b-1b-2b-1b-3b | /+|+|/+|+|/+|+|/+|+| 3b-2b-1b-2b-1b-2b-1b-2b-3b |+\ |/+|+|/+|+|/+|+|/+| 3e-1e-2e-1e-2e-1e-2e-1e-3e |/+|+\ |/+|+|/+|+|/+|+| 3w-2w-1w-2w-1w-2w-1w-2w-3w

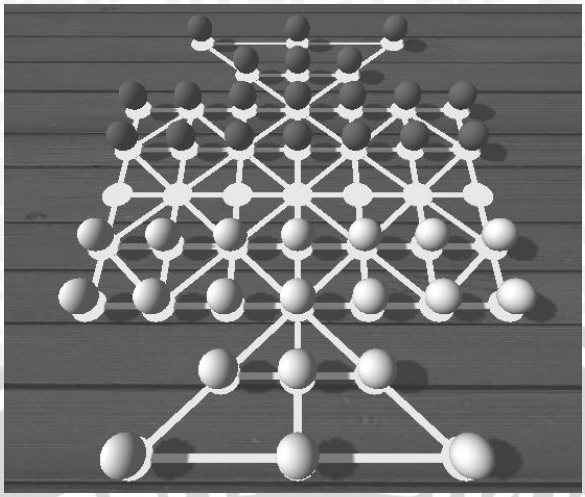
		|+\ |/+|+\\|/+|+\\|/+| 3w-1w-2w-1w-2w-1w-2w-1w-3w |/+|+\ |/+|+\\|/+|+\\|/+|+\\| 4w-3w-3w-3w-1w-3w-3w-3w-4w /+|+\+++ xxx3w-2w-3wxxx /+ +|+ +\+ + xx4w-x-3w-x-4wxx



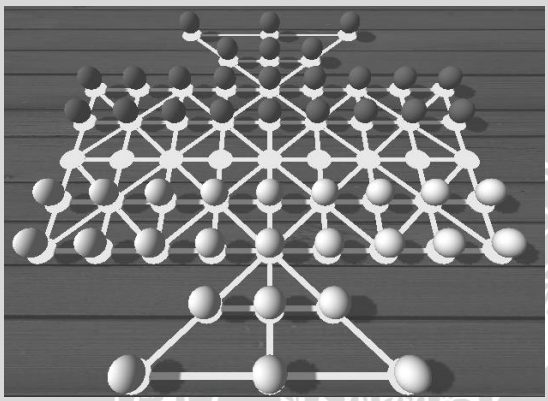
Gambar 4.3 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 5x5



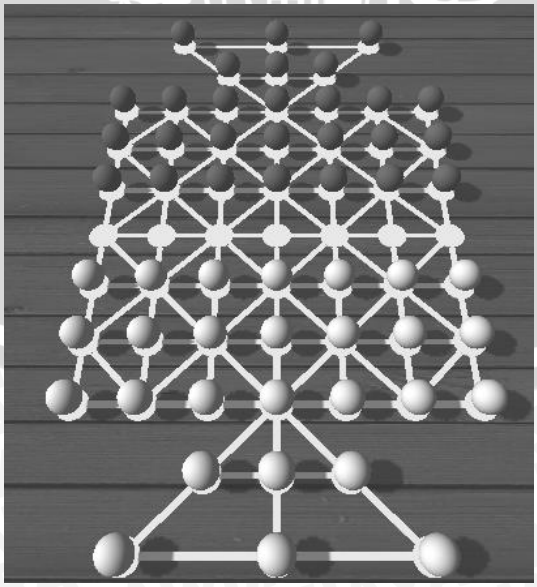
Gambar 4.4 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 5x9



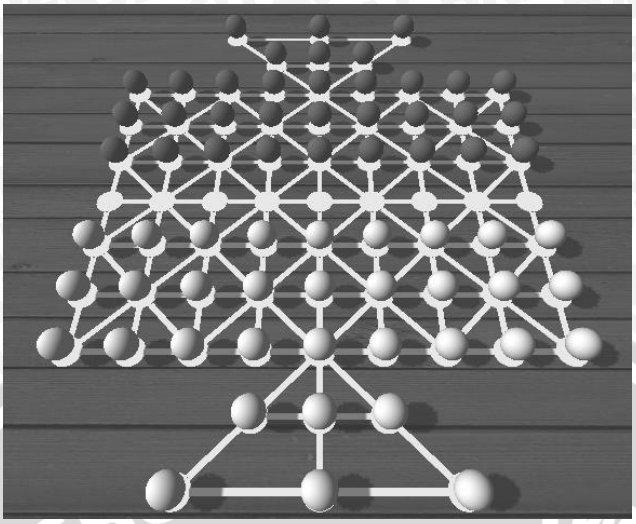
Gambar 4.5 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 7x9



Gambar 4.6 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 9x9



Gambar 4.7 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 7x11



Gambar 4.8 *Screenshot* Model 3D Papan Permainan Ntil Berukuran 9x11

4.4 Implementasi Agen Cerdas

Tahap ini mengimplementasikan algoritma *minimax*, dan *alpha-beta pruning* untk mencari langkah terbaik yang akan diambil oleh AI.

4.4.1 Implementasi Algoritma *Minimax*

Algoritma *Minimax* diimplementasikan dalam fungsi *minimax*, Tabel 4.5 merupakan tabel *pseudocode* fungsu *minimax* yang telah diimplementasikan.

Tabel 4.5 *Pseudocode* Fungsi *minimax*

```
1   function minimax(board, depth, maximizingPlayer)
2
3   //mengecek apakah simpul merupakan simpul terdalam
4   if depth==maxDepth or isGameOver() then
5      //jika benar, simpul dievaluasi dan hasil evaluasi dikembalikan
6   return evaluate(board)
7   end if
8
9   //variabel untuk menyimpan anak dari simpul board
10  nextBoard
11
12  //variabel untuk menyimpan hasil evaluasi simpul
13  value
14
15  //mencari semua langkah bidak yang nantinya akan digunakan sebagai
16  anak simpul
17  successors = board.findAllMove()
```

```
18
19  //jika pemain saat ini merupakan AI yang langkahnya akan
20  dimaksimalkan
21  if maximizingPlayer then
22
23   //inisialisasi awal nilai bestValue = nilai terendah dari integer
24     bestValue = Int32.MinValue
25
26   //melakukan perulangan selama simpul memiliki anak
27     while !sucessors.isEmpty() do
28
29  //melakukan clone board agar posisi bidak pada papan permainan yang
30  ada pada simpul akar tidak berubah,
31         nextBoard=board.cloneNtilBoard()
32
33  //menggerakkan bidak pada papan permainan yang ada pada simpul anak
34         move = sucessors.popFront()
35        pieceType = nextBoard.getCurrentPlayer()
36        nextBoard.move(pieceType,move)
37
38  //melakukan rekursif untuk mendapatkan simpul terdalam, dan hasil
39  evaluasi pada simpul terdalam disimpan pada variable value, dan
40  dilakukan pergantian pemain
41         value = minimax(nextBoard, depth+1, false)
42
43  //mengubah nilai bestValue menjadi nilai yang lebih besar dari hasil
44  perbandingan antara bestValue dengan value
45  //hal ini dilakukan untuk memaksimalkan langkah bidak pemain
46        bestValue = max(bestValue,value)
47     end while
48     return bestValue
49
50  //Jika pemain saat ini merupakan lawan main yang langkahnya akan
51  diminimalkan
52  else
53
54  //inisialisasi awal nilai bestValue = nilai terendah dari integer
55     bestValue = Int32.MinValue
56
57   //melakukan perulangan selama simpul memiliki anak
58     while !sucessors.isEmpty() do
59
60  //melakukan clone board agar posisi bidak pada papan permainan yang
61  ada pada simpul akar tidak berubah,
62         nextBoard=board.cloneNtilBoard()
63
64  //menggerakkan bidak pada papan permainan yang ada pada simpul anak
```

```
65          move = sucessors.popFront()
66          pieceType = nextBoard.getCurrentPlayer()
67          nextBoard.move(pieceType,move)
68
69      //melakukan rekursif untuk mendapatkan simpul terdalam, dan hasil
70      evaluasi pada simpul terdalam disimpan pada variable value, dan
71      dilakukan pergantian pemain
72          value = minimax(nextBoard, depth+1, true)
73
74      //mengubah nilai bestValue menjadi nilai yang lebih kecil dari hasil
75      perbandingan antara bestValue dengan value
76      //hal ini dilakukan untuk meminimalkan langkah bidak lawan
77          bestValue = min(bestValue,value);
78
79      end while
80      return bestValue
81
82  end if
83
84  end function
```

4.4.2 Implementasi Algoritma *Alpha-Beta Pruning*

Algoritma *Alpha-beta Pruning* diimplementasikan dalam fungsi *alphabeta*, Tabel 4.6 merupakan tabel *pseudocode* fungsi alphabeta yang telah diimplementasikan.

Tabel 4.6 *Pseudocode* Fungsi *alphabeta*

```
1   function alphabeta(board, depth, alpha, beta, maximizingPlayer)
2
3   //mengecek apakah simpul merupakan simpul terdalam
4   if depth==maxDepth or isGameOver() then
5       //jika benar, simpul dievaluasi dan hasil evaluasi dikembalikan
6   return evaluate(board)
7   end if
8
9   //variabel untuk menyimpan anak dari simpul board
10  nextBoard
11
12  //variabel untuk menyimpan hasil evaluasi simpul
13  value
14
15  //mencari semua langkah bidak yang nantinya akan digunakan sebagai
16  anak simpul
```

```
17  successors = board.findAllMove()
18
19  //jika pemain saat ini merupakan AI yang langkahnya akan
20  dimaksimalkan
21  if maximizingPlayer then
22
23   //melakukan perulangan selama simpul memiliki anak
24     while !sucessors.isEmpty() do
25
26  //melakukan clone board agar posisi bidak pada papan permainan yang
27  ada pada simpul akar tidak berubah,
28        nextBoard=board.cloneNtilBoard()
29
30  //menggerakkan bidak pada papan permainan yang ada pada simpul anak
31        move = sucessors.popFront()
32        pieceType = nextBoard.getCurrentPlayer()
33        nextBoard.move(pieceType,move)
34
35  //melakukan rekursif untuk mendapatkan simpul terdalam, dan hasil
36  evaluasi pada simpul terdalam disimpan pada variable value, dan
37  dilakukan pergantian pemain
38        value = alphabeta(nextBoard, depth+1, alpha, beta, false)
39
40  //mengubah nilai alpha menjadi nilai yang lebih besar dari hasil
41  perbandingan antara alpha dengan value
42  //hal ini dilakukan untuk memaksimalkan langkah bidak pemain
43        alpha = max (alpha, value)
44
45  //melakukan pemotongan pencarian jika nilai beta kurang dari sama
46  dengan alpha
47        if (beta<=alpha)
48           break
49     end while
50   return alpha
51
52  //Jika pemain saat ini merupakan lawan main yang langkahnya akan
53  diminimalkan
54  else
55
56   //melakukan perulangan selama simpul memiliki anak
57     while !sucessors.isEmpty() do
58
59  //melakukan clone board agar posisi bidak pada papan permainan yang
60  ada pada simpul akar tidak berubah,
61        nextBoard=board.cloneNtilBoard()
62
63  //menggerakkan bidak pada papan permainan yang ada pada simpul anak
```

```
64          move = sucessors.popFront()
65          pieceType = nextBoard.getCurrentPlayer()
66          nextBoard.move(pieceType,move)
67
68      //melakukan rekursif untuk mendapatkan simpul terdalam, dan hasil
69      evaluasi pada simpul terdalam disimpan pada variable value, dan
70      dilakukan pergantian pemain
71          value = alphabeta(nextBoard, depth+1, alpha, beta, true)
72
73      //mengubah nilai beta menjadi nilai yang lebih kecil dari hasil
74      perbandingan antara beta dengan value
75      //hal ini dilakukan untuk meminimalkan langkah bidak lawan
76          beta = min (beta, value)
77
78      //melakukan pemotongan pencarian jika nilai beta kurang dari sama
79      dengan alpha
80          if (beta<=alpha)
81              break
82
83      end while
84      return bestValue
85
86  end if
87  end function
```

4.4.3 Implementasi Fungsi *evaluate*

Fungsi *evaluate* akan memanggil fungsi *calculateValue* untunk mendapatkan nilai bidak pada titik tertentu. Tabel 4.7 merupakan tabel *pseudocode* fungsi *evaluate*, dan Tabel 4.8 merupakan tabel *pseudocode* fungsi *calculateValue*.

Tabel 4.7 *Pseudocode* Fungsi *evaluate*

```
1   function evaluate(board)
2
3   playerValue=0
4   enemyValue=0
5   index = 0
6   piece
7
8   //melakukan perulangan sampai semua bidak terevaluasi
9   while index!=board.points.Length do
10
11      //mendapatkan bidak pada posisi ke-index
```

```
12      piece = board.points[index]
13      if piece==board.forbidden or piece==board.empty then
14              index++
15
16      else
17              //perbaharui nilai playerValue jika bidak tersebut merupakan
18  bidak pemain
19              if piece==_Ntilboard.getCurrentPlayer()*1 or
20                piece==_Ntilboard.getCurrentPlayer()*2 then
21                      playerValue += calculateValue (piece, index)
22
23              //perbaharui nilai enemyValue jika bidak tersebut merupakan
24  bidak lawan main
25              else
26                      enemyValue += calculateValue (piece, index)
27
28              end if
29
30          index++
31
32      end if
33
34  end while
35
36  return playerValue-enemyValue;
37
38  end function
```

Tabel 4.8 *Pseudocode* Fungsi *calculateValue*

```
1   function calculateValue(piece, pos)
2
3   value=0
4
5   //mendapatkan posisi tengan pada papan permainan
6   middle = (_Ntilboard.points.Length-_Ntilboard.boardWidth)/2
7
8   //mendapatkan nilai bidak biasa putih
9    if piece==_Ntilboard.whitePawn then
10
11      //jika bidak berada di daerah pemain itu sendiri
12       if pos<middle then
13              value=3
14      //jika bidak berada di daerah tengan papan permainan
15       else if pos>=middle and pos<middle+_Ntilboard.boardWidth then
16              value=5
17      //jika bidak berada di daerah lawan
```

```
18          else
19              value=7
20          end if
21
22      //mendapatkan nilai bidak biasa hitam
23      else if piece==_Ntilboard.blackPawn then
24
25          //jika bidak berada di daerah pemain itu sendiri
26          if pos>=middle+_Ntilboard.boardWidth then
27              value=3
28          //jika bidak berada di daerah tengan papan permainan
29          else if pos>=middle and pos<middle+_Ntilboard.boardWidth then
30              value=5
31          //jika bidak berada di daerah lawan
32          else
33              value=7
34          end if
35
36      //mendapatkan nilai bidak raja
37      else
38          value = 12
39
40      end if
41
42      //kembalikan nilai bidak*bobot titik pada posisi bidak itu
43      return value * _Ntilboard.getBoardWeight(pos)
44
45      end function
```

4.4.4 Implementasi Fungsi *bestMove*

Tabel 4.9 merupakan tabel *pseudocode* fungsi *bestMove.*

Tabel 4.9 *Pseudocode* Fungsi *bestMove*

```
1       function bestMove(board, state)
2
3       //variabel untuk menyimpan anak dari simpul board
4       nextBoard
5
6       //variabel untuk menyimpan hasil evaluasi simpul
7       value = Int32.MinValue
8
9       //variable untuk menyimpan nilai terbaik dari simpul
10      maxValue = Int32.MinValue
11
```

```
12  //variable untuk menyimpan langkah bidak
13  move = new Move()
14
15  //variable untuk menyimpan langkah bidak terbaik
16  bestMove = new Move()
17
18  //mencari semua langkah bidak yang nantinya akan digunakan sebagai
19  anak simpul
20  successors = board.findAllMove()
21
22  //melakukan perulangan selama simpul memiliki anak
23  while !sucessors.isEmpty() do
24
25  //melakukan clone board agar posisi bidak pada papan permainan yang
26  ada pada simpul akar tidak berubah,
27        nextBoard=board.cloneNtilBoard()
28
29  //menggerakkan bidak pada papan permainan yang ada pada simpul anak
30        move = sucessors.popFront()
31        pieceType = nextBoard.getCurrentPlayer()
32        nextBoard.move(pieceType,move)
33
34  //agen melakukan pencarian langkah dengan fungsi minimax jika
35  state==1, jika tidak maka pencarian langkah dengan fungsi alphabeta
36        if state==1 then
37              value = minimax(nextBoard, depth+1, false)
38        else
39              value = alphabeta(nextBoard, depth+1, maxValue,
40  Int32.MaxValue, false)
41        end if
42
43  //jika hasil evaluasi simpul (value) lebih besar dari nilai simpul saat ini
44  (maxValue)
45        if value>maxValue then
46
47              //perbaharui nilai maxValue sama deangan value
48               maxValue=value
49
50              //perbaharui nilai bestMove sama dengan move
51               bestMove = move
52
53        end if
54
55   end while
56
57  end function
```

BAB V
PENGUJIAN DAN ANALISIS

5.1 Lingkungan Pengujian

Bagian ini berisi penjelasan tentang lingkungan pengujian berupa peralatan yang digunakan dalam menguji permainan. Dalam penulisan tugas akhir ini, lingkungan pengujian dibedakan menjadi 2 jenis, yaitu: Lingkungan perangkat keras dan lingkungan perangkat lunak.

5.1.1 Lingkungan Pengujian Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam pengujian permainan ini dijelaskan dalam Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Pengujian

Perangkat Keras	Spesifikasi
Prosesor	Intel(R) Core(TM) i3
Memori (RAM)	4 GB
Kartu Grafis	Intel(R) HD Graphics

5.1.2 Lingkungan Pengujian Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengujian permainan ini dijelaskan dalam Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak Pengujian

Perangkat Lunak	Spesifikasi
Operating System	*Microsoft windows 8.1 Pro 3- bit*
Game engine	Unity 4.5.5.f1
Bahasa pemrograman	C#
Integrated Development Environmet	MonoDevelop 4.0.1
3D *modeler*	Unity 4.5.5.f1

5.2 Skenario Pengujian

Skenario dalam pengujian ini dibagi menjadi tiga skenario. Skenario I dilakukan untuk menguji apakah langkah yang diambil oleh agen merupakan langkah yang terbaik. Skenario II dilakukan untuk menguji kinerja dari algoritma *Minimax* dan *Alpha-beta Pruning* terhadap nilai *depth*. Skenario III dilakukan untuk menguji kinerja dari algoritma *Minimax* dan *Alpha-beta Pruning* terhadap ukuran papan permainan

5.2.1 Skenario I

Skenario dalam pengujian ini adalah dengan mencatat skor akhir dari permainan ntil. Pada skenario ini, permainan ntil akan dimainkan pada papan berukuran 5x9, antara AI-1 melawan AI-2. AI-1 adalah AI yang akan bergerak terlebih dahulu pada awal permainan, yang menerapkan algoritma *Alpha-beta Pruning* dengan nilai *depth*=2. AI-2 akan menerapkan algoritma *Minimax,* dan *Alpha-beta Pruning*, dengan nilai *depth* dari 1 sampai 6. Permainan dilakukan sampai permainan selesai, atau kedua AI tersebut melakukan gerakan yang berulang-ulang.

Hasil dari skenario ini diharapkan dapat menentukan apakah algoritma *minimax,* dan *alpha-beta pruning* dapat menghasilkan langkah yang sama, pada lawan yang sama.

5.2.2 Skenario II

Pada skenario ini akan dilakukan permainan ntil antara AI-1 melawan AI-2 pada papan berukuran 5x9. AI-1 adalah AI yang akan bergerak terlebih dahulu pada awal permainan, yang menerapkan algoritma *Alpha-beta Pruning* dengan nilai *depth*=2. AI-2 akan menerapkan algoritma *Minimax*, dan *Alpha-beta Pruning*, dengan nilai *depth* dari 1 sampai 6. Skenario dalam pengujian ini adalah mengukur rata-rata *running time* dan rata-rata jumlah simpul yang diekspansi (*expanded node)* dari 10 langkah yang dilakukan oleh AI-2.

Hasil dari skenario ini diharapkan dapat menentukan kinerja dari algoritma *Minimax*, dan A*lpha-beta Pruning* terhadap nilai *depth* pada permainan ntil.

5.2.3 Skenario III

Pada skenario ini akan dilakukan permainan ntil antara AI-1 melawan AI-2 pada papan berukuran 5x5, 5x9, 7x9, 9x9, 7x11, dan 9x11. AI-1 adalah AI yang akan bergerak terlebih dahulu pada awal permainan, yang menerapkan algoritma *Alpha-beta Pruning* dengan nilai *depth*=2. AI-2 akan menerapkan algoritma *Minimax,* dan *Alpha-beta Pruning*, dengan nilai *depth=3*. Skenario dalam pengujian ini adalah mengukur rata-rata *running time* dan rata-rata jumlah simpul yang diekspansi (*expanded node)* dari 10 langkah yang dilakukan oleh AI-2.

Hasil dari skenario ini diharapkan dapat menentukan kinerja dari algoritma *Minimax,* dan A*lpha-beta Pruning* terhadap ukuran papan permainan pada permainan Ntil.

5.2.4 Skenario IV

Pada skenario ini akan dilakukan permainan ntil antara *human* melawan agen, pada papan permainan berukuran 5x9, 7x9, dan 7x11. agen akan menerapkan algoritma *Alpha-beta Pruning*, dengan nilai *depth* 3, 4, dan 5. Skenario dalam pengujian ini mengukur banyaknya menang, seri, atau kalah untuk *human* maupun agen.

Hasil dari skenario ini diharapkan dapat menentukan tingkat kesulitan agen terhadap nilai *depth* dalam permainan Ntil.

5.3 Hasil Pengujian

Tahap ini menunjukan hasil pengujian dari skenario yang telah dibuat. Skenario I menguji langkah yang diambil oleh agen berdasarkan nilai *depth*, dan algoritma yang digunakan. Skenario II dan III, menguji kinerja dari kedua algoritma tersebut berdasarkan nilai *depth*, dan ukuran papan permainan. Skenario IV, menguji tingkat kesulitan agen berdasarkan nilai *depth*.

5.3.1 Skenario I

Tabel 5.3 menunjukkan hasil pengujian skenario I.

Tabel 5.3 Hasil Pengujian Skenario I

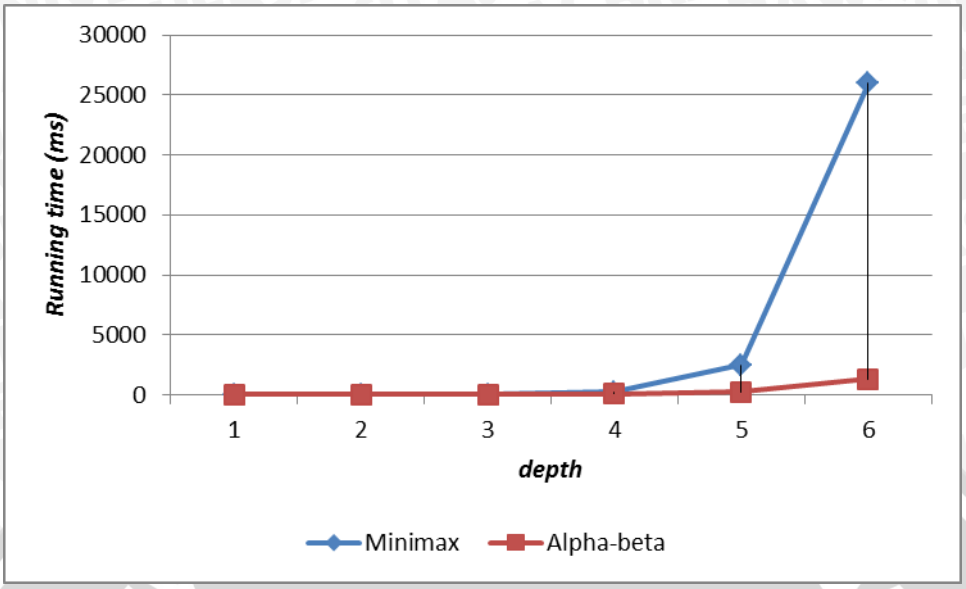
depth AI-1	*depth* AI-2	AI-2 dengan *Minimax*		AI-2 dengan *Alpha-beta*	
		Jumlah bidak AI-1	Jumlah bidak AI-2	Jumlah bidak AI-1	Jumlah bidak AI-2
2	1	8	5	8	5
2	2	7	7	7	7
2	3	10	11	10	11
2	4	3	6	3	6
2	5	0	4	0	4
2	6	1	5	1	5

5.3.2 Skenario II

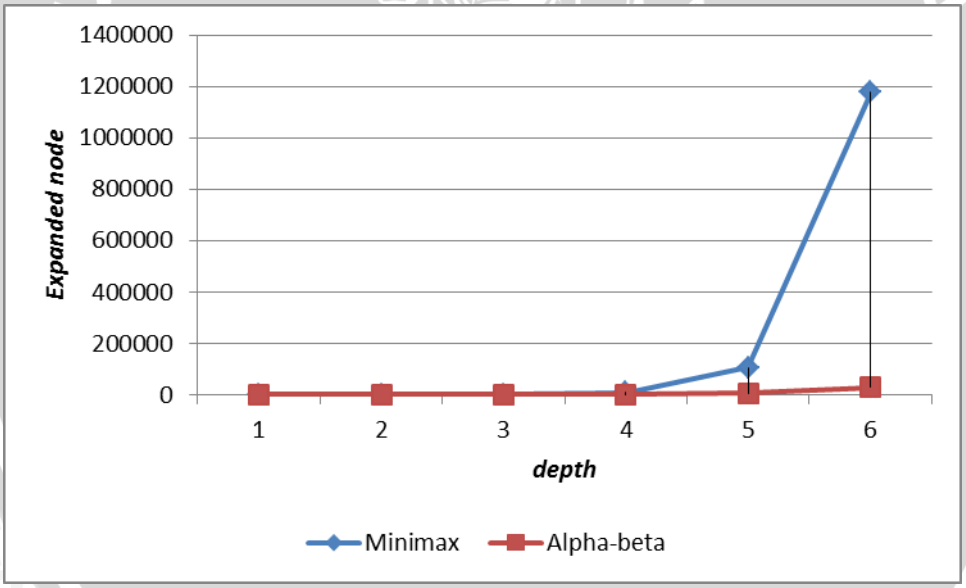
Tabel 5.4, Gambar 5.1, dan Gambar 5.2 menunjukkan hasil pengujian skenario II.

Tabel 5.4 Hasil Pengujian Skenario II

depth	*Minimax*		*Alpha-beta*	
	Rata-rata running time (ms)	Rata-rata expanded node	Rata-rata running time (ms)	Rata-rata expanded node
1	0.177573	5	0.149048	5
2	2.629388	59	1.685046	31
3	27.4571	1016	14.14933	275
4	224.9483	9729	50.1251	878
5	2490.953	105039	235.2254	5365
6	25969.5844	1176562	1300.446	28778



Gambar 5.1 Grafik Hasil Pengujian Skenario II antara *depth* dengan *Running Time* (ms)



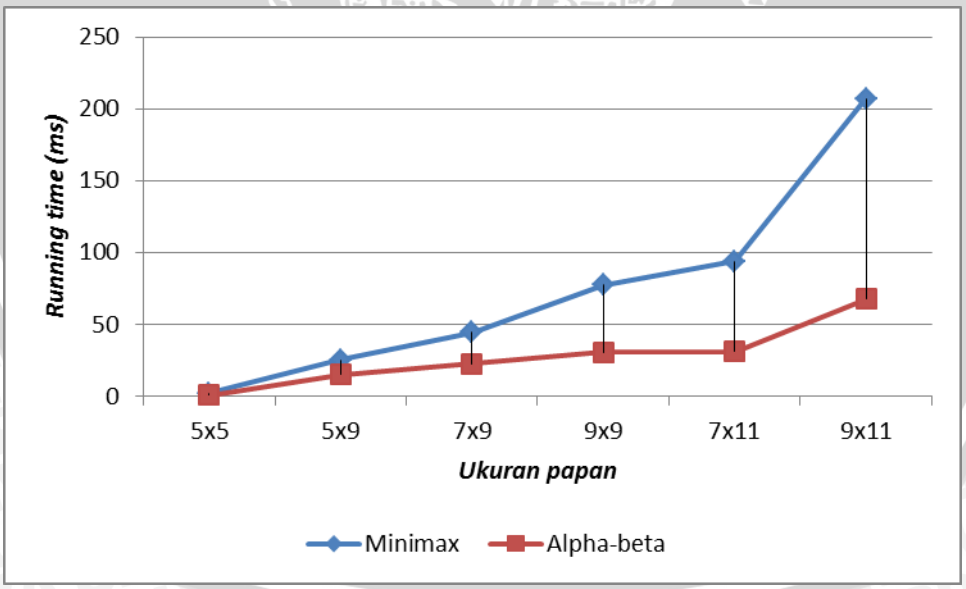
Gambar 5.2 Grafik Hasil Pengujian Skenario II antara *depth* dengan *Expanded Node*

5.3.3 Skenario III

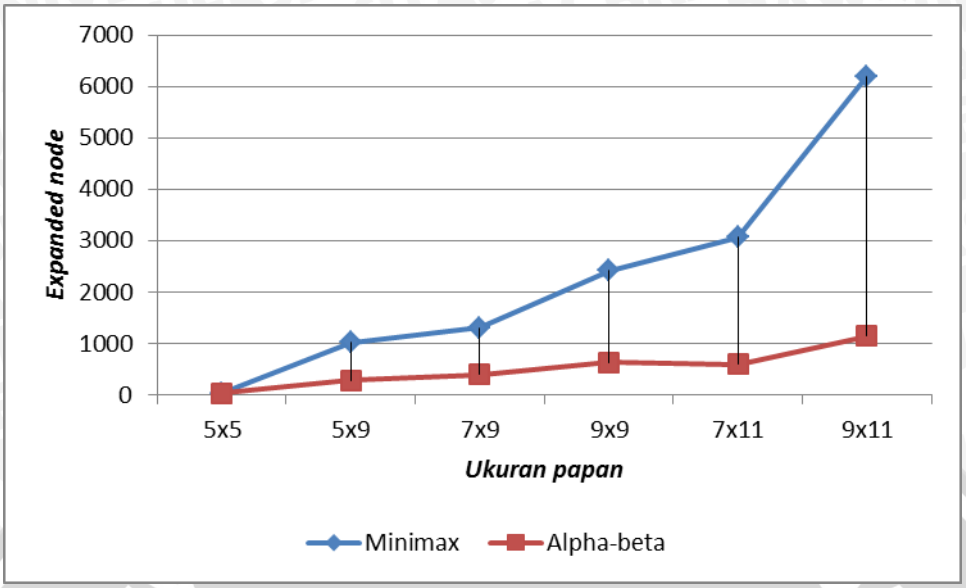
Tabel 5.5, Gambar 5.3, dan Gambar 5.4 menunjukkan hasil pengujian skenario III.

Tabel 5.5 Hasil Pengujian Skenario III

Ukuran papan	*Minimax*		*Alpha-beta*	
	Rata-rata running time (ms)	Rata-rata expanded node	Rata-rata running time (ms)	Rata-rata expanded node
5x5	2.092682	34	0.723981	26
5x9	25.27824	1016	15.09105	275
7x9	44.49535	1308	22.67479	389
9x9	77.25417	2419	30.38924	624
7x11	93.87739	3068	30.73445	588
9x11	206.9475	6196	67.53644	1139



Gambar 5.3 Grafik Hasil Pengujian Skenario III antara Ukuran Papan dengan *Running Time* (ms)



Gambar 5.4 Grafik Hasil Pengujian Skenario III antara Ukuran Papan dengan *Expanded Node*

5.3.4 Skenario IV

Tabel 5.6 menunjukkan hasil pengujian skenario IV.

Tabel 5.6 Hasil Pengujian Skenario IV

Ukuran Papan	*depth*	Menang		Seri	Kalah		Rata-rata *Runninng Time* (ms)
		Human	Agen		*Human*	Agen	
5x9	3	2	0	1	0	2	49.56439
	4	2	3	0	3	0	86.12891
	5	0	4	1	4	0	241.0342
7x9	3	1	4	0	4	1	111.5277
	4	0	3	2	3	0	148.9106
	5	0	5	0	5	0	413.0136
7x11	3	2	3	0	3	2	141.613
	4	0	4	1	4	0	312.1113
	5	0	4	1	4	0	1495.804

5.4 Analisis

Tahap analisis bertujuan untuk mendapatkan kesimpulan dari tiga skenario yang telah dibuat dalam menguji implementasi algoritma *Minimax,* dan *Alpha-beta Pruning* dalam permainan tradisional ntil. Analisis mengacu pada hasil pengujian yang diperolah dari tabel dan grafik pada skenario I, II, III, dan IV.

5.4.1 Skenario I

Skenario I bertujuan untuk mengetahui apakah langkah yang diambil oleh AI yang menerapkan algoritma *minimax* dan *alpha-beta pruning* menghasilkan langkah yang sama, dan merupakan langkah yang terbaik. Analisis dilakukan dengan melihat skor akhir dari permainan antara AI-1 melawan AI-2, dan *depth* dari setiap AI.

Dari Tabel 5.3 dapat dilihat bahwa skor akhir permainan antara AI yang menerapkan algoritma *Minimax* dengan AI yang menerapkan algoritma *Alpha-beta Pruning* adalah sama, pada lawan main yang sama. Hal ini menunjukkan kedua algoritma tersebut menghasilkan pergerakan bidak yang sama selama permainan berlangsung. Karena secara umum, algoritma *Alpha-beta Pruning* melakukan pencarian langkah dengan proses yang sama dengan *Minimax*.

5.4.2 Skenario II

Skenario II bertujuan untuk mengetahui kinerja algoritma *minimax* dan *alpha-beta pruning* terhadap nilai *depth*. Analisis dilakukan dengan melihat *running time* dan *expanded node* pada setiap algoritma yang diterapkan, dengan *depth* yang telah ditentukan.

Dari Gambar 5.1 dapat dilihat grafik hubungan antara jumlah *depth* dengan *running time* yang dihasilkan. Semakin besar *depth* maka semaking lama *running time* yang dihasilkan. Dari Gambar 5.2 dapat dilihat grafik hubungan antara jumlah *depth* dengan *expanded node* yang dihasilkan. Semakin besar *depth* maka semakin banyak jumlah *expanded node*. Dari kedua grafik tersebut menunjukkan bahwa, semakin besar *depth* maka semakin banyak jumlah simpul yang diekspansi (*expanded node*), sehingga membutuhkan waktu pencarian

(*running time)* yang lebih lama untuk mendapatkan langkah bidak yang terbaik, dimana kinerja akan semakin buruk.

Dari grafik hasil pengujian skenario II menunjukkan algoritma *Alpha-beta Pruning* memiliki kinerja yang lebih baik dibandinkang algortima Minimax dalam permainan Ntil. Hal ini menunjukan algoritma *Alpha-beta pruning* mampu mengatasi jumlah *depth* yang lebih besar, dengan nilai *running time* yang lebih cepat, dibandingkan dengan algoritma *Minimax*. Namun pada *depth*=1, kedua algoritma tersebut menghasilkan *running time* yang hampir sama, dengan jumlah *expanded node* yang sama. Karena pada *depth*=1, simpul yang dihasilkan merupakan langkah-langkah yang akan dimabil oleh agen itu sendiri, tanpa memperkirakan langkah lawan. Sehingga fungsi m*inimax*, maupun fungsi *alphabeta* tidak dijalankan.

5.4.3 Skenario III

Skenario III bertujuan untuk mengetahui kinerja algoritma *minimax* dan *alpha-beta pruning* terhadap ukuran papan permainan. Analisis dilakukan dengan melihat *running time* dan *expanded node* pada setiap algoritma yang diterapkan, dengan ukuran papan yang telah ditentukan.

Dari Gambar 5.3 dapat dilihat grafik hubungan antara ukuran papan dengan *running time* yang dihasilkan. Semakin besar ukuran papan maka semaking lama *running time* yang dihasilkan. Dari Gambar 5.4 dapat dilihat grafik hubungan antara ukuran papan dengan *expanded node* yang dihasilkan. Semakin besar ukuran papan maka semakin banyak jumlah *expanded node*. Dari kedua grafik tersebut menunjukkan bahwa, semakin besar ukuran papan (jumlah titik pada papan), maka semakin banyak jumlah simpul yang diekspansi (*expanded node*), sehingga membutuhkan waktu pencarian (*running time)* yang lebih lama untuk mendapatkan langkah bidak yang terbaik, dimana kinerja akan semakin buruk.

Dari grafik hasil pengujian skenario III menunjukkan algoritma *Alpha-beta Pruning* memiliki kinerja yang lebih baik dibandinkang algortima Minimax dalam permainan Ntil. Hal ini menunjukan algoritma *Alpha-beta pruning* mampu

mengatasi ukuran papan permainan yang lebih besar, dengan nilai *running time* yang lebih cepat, dibandingkan dengan algoritma *Minimax*.

5.4.4 Skenario IV

Skenario IV bertujuan untuk mengetahui nilai *depth* yang optimal ukuran papan permainan. Analisis dilakukan dengan melihat jumlah menang, seri, kalah dan *running time* pada setiap *depth* dan ukuran papan yang telah ditentukan.

Dari Tabel 5.6, dapat dilihat bahwa semaking besar nilai *depth* pada setiap papan permainan, jumlah menang *human* semakin sedikit. Hal ini menunjukkan tingkat kesulitan agen dapat ditentukan dengan mengatur nilai depth, dimana semakin besar nilai *depth*, maka agen semakin sulit dikalahkan. Namun, semakin besar nilai *depth*, agen membutuhkan waktu pencarian langkah semakin lama. Hal ini mengakibatkan kinerja dari agen semakin buruk.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang dilakukan, maka dapat diambil kesimpulan sebagai berikut :

1. Algoritma *Minimax* dan *Alpha-beta Pruning* dapat diimplementasikan untuk pencarian langkah bidak pada permainan Ntil. Dengan melakukan evaluasi pada pohon permainan yang dibangkitkan dari setiap kemungkinan langkah bidak dalam permainan Ntil.
2. Hasil pengujian langkah yang diambil oleh agen, menunjukkan algoritma *Minimax* dan *Alpha-beta Pruning* menghasilkan langkah yang sama, pada nilai *depth* dan lawan main yang sama.
3. Hasil pengujian kinerja terhadap nilai *depth* menunjukkan bahwa algoritma *Alpha-beta Pruning* memiliki kinerja yang lebih baik dibandingkan algoritma *Minimax* dengan rata-rata perbedaan *running time* sebesar 1354.367 ms.
4. Hasil pengujian kinerja terhadap ukuran papan menunjukkan bahwa algoritma *Alpha-beta Pruning* memiliki kinerja yang lebih baik dibandingkan algoritma *Minimax* dengan rata-rata perbedaan *running time* sebesar 65.9776 ms.
5. Hasil pengujian *depth* terhadap skor akhir permainan antara *human* melawan agen, menunjukkan bahwa semakin besar nilai *depth* maka agen semakin sulit untuk dikalahakan.

6.2 Saran

1. Pencarian langkah terbaik dalam permainan Ntil dapat dikembangkan lebih lanjut dengan algoritma pencarian yang lain, seperti algoritma *negamax*, *negascout*, dan algoritma lainnya.
2. Fungsi evaluasi dalam permainan Ntil dapat dikembangkan lebih lanjut untuk mendapatkan hasil pencarian langkah yang lebih baik.

DAFTAR PUSTAKA

.

[JON-08] Jones, M., Dkk. 2008, "Artificial Intelligence : A System Approach", Infinity Science Press LLC:Massachusetts.

[LES-13] Lestari, Jati. & Siti Annisa Amalia. 2013, "Implementasi Algoritma Alpha-Beta Pruning pada Permainan Bantumi dengan Berbasi Mobile Android", BIT Vol 10 No 1 : Universitas Budi Luhur.

[PRA-13] Prasetyo, Agung. 2013, "Implementasi Kecerdasan Buatan Pada Permainan Damdaman Menggunakan Algoritma Minimax", Universitas Pendidikan Indonesia.

[RUS-95] Russell, Stuart J. & Peter Norvig. 1995, "Artificial Intelligence : A Modern Approach", New Jersey :Alan Apt.

[ROG-10] Rogers, Scott. 2010, "Level Up : The Guide to Great Video Game Design", United Kingdom : Jonh Wiley & Sons, Ltd.

[YUN-81] Yunus, Ahmad. 1981, "Permainan Rakyat Daerah Istimewa Yogyakarta", Departemen Pendidikan dan Kebudayaan Proyek Inventarisasi dan Dokumentasi Kebudayaaan Daerah, Yogyakarta.