Rancang Bangun Aplikasi Catur Menggunakan Algoritma Minimax Dengan Optimasi Alpha-Beta Pruning

Panji Novantara, M.T*[1], Nana Suhana[*2]
[1]*Manajemen Informatika Universitas Kuningan*
[2]*Sistem Informasi Universitas Kuningan*
Jl Cut Nyak Dien No 36 A Cijoho Kabupaten Kuningan
[*1]***Panji@uniku.ac.id***, [2] ***Nana_Suhana@yahoo.com***

ABSTRACT

Chess is a strategy game which is played in turns. This game has been around since ancient times. In the game of chess, it requires strategy or a good way to win a game. It is one of sports that many people played. Almost everyone knows the game of chess, although some of them do not know the steps of the chess pieces. To determine a good movement in chess, we need an algorithm, one of them is the Alpha-Beta Pruning algorithm. Alpha-Beta Pruning is the development of MiniMax algorithm which is one of the searching algorithm to determine the maximum and minimum values by cutting unnecessary nodes. In this study, the writer makes an application chess game that anyone can play chess on an electronic device, especially PC, while the algorithm used is Alpha-Beta Pruning (specify the steps taken by the computer) with a software development method that is chosen is the Rational Unified Process (RUP). In the process of implementing, these algorithms needs a method called evaluation board, in which each chess piece has a respective value of 0 is king, queen 900, 300 elephants, 300 horses, castle 500 and pawns 100. The results of this paper is the use of an algorithm Alpha-Beta Pruning in this chess game that the computer can respond to the steps taken by the users quickly.

Keywords : *Chess, Minimax, Alpha-Beta Pruning, Node, RUP*

1. PENDAHULUAN

Perkembangan perangkat komputer selama beberapa tahun ini sangat cepat. Pada awal perkembangannya perangkat komputer hanya berupa alat yang mampu digunakan untuk melakukan komputasi yang berhubungan dengan aritmatika. Hingga beberapa tahun yang lalu, komputer sudah berkembang menjadi perangkat multifungsi. Beberapa kegunaan komputer sekarang antara lain untuk : mengetik, menggambar, mengedit video, main game dll. Pada abad 21 sekarang komputer digunakan untuk bekerja dan perangkat hiburan. Perangkat komputer sebagai hiburan, dikarenakan komputer dapat digunakan untuk bermain game.

Dengan pesatnya perkembangan teknologi perangkat lunak sekarang ini menyebabkan berkembangnya program aplikasi permainan untuk berbagai golongan usia. Mulai dari permainan yang dimanfaatkan untuk merangsang pertumbuhan otak anak sampai permainan yang dimanfaatkan untuk mengisi waktu luang. Jenis permainannya beraneka ragam seperti petualangan, pertarungan bahkan permainan yang membutuhkan strategi dalam penyelesaiannya. Seperti pada program permainan komputer dimana lawan seolah-olah mampu memberikan perlawanan karena komputer tersebut memiliki kecerdasan buatan atau yang sering disebut *Artificial Intelligence*. Salah satu contohnya adalah Permainan Catur.

Pada skripsi ini penulis akan mengimplementasikan suatu metode kecerdasan buatan ke dalam bentuk program permainan Catur, dimana kecerdasan buatan berfungsi untuk menetapkan langkah-langkah yang dapat diambil oleh komputer. Komputer dapat menganalisis berbagai kemungkinan langkah yang akan diambil lawan, sehingga komputer dapat memilih langkah yang paling menguntungkan. Adapun algoritma yang penulis ambil yaitu algoritma MiniMax dan algoritma Alpha-Beta Pruning.

Berdasarkan latar belakang di atas, maka penulis bermaksud membuat suatu aplikasi yang berjudul : "**RANCANG BANGUN APLIKASI CATUR MENGGUNAKAN ALGORITMA MINIMAX DENGAN OPTIMASI ALPHA-BETA PRUNING".**

2. LANDASAN TEORI

2.1. Sejarah Catur

Menurut H. J. R. Murray, penulis buku History of Chess (1913), catur berasal dari India dan mulai ada pada abad ke-6. Di sana catur dikenal dengan nama chaturanga, yang artinya empat unsur yang terpisah. Awalnya, buah catur memang hanya empat jenis. Menurut mistisisme India kuno, catur dianggap mewakili alam semesta ini, sehingga sering dihubungkan dengan empat unsur kehidupan, yaitu api, udara, tanah dan air karena dalam permainannya, catur menyimbolkan cara-cara hidup manusia.

2.2. Langkah Bidak Catur

1) ***King*** (Raja) dapat bergerak satu petak ke segala arah. Raja juga memiliki gerakan khusus yang disebut rokade yang turut melibatkan sebuah benteng.

2) ***Queen*** (Ratu) memiliki gerakan kombinasi dari Benteng dan Gajah.
3) **Rook** (Benteng) dapat bergerak sepanjang petak horizontal maupun vertikal, tetapi tidak dapat melompati bidak lain.
4) **Bishop** (Gajah) dapat bergerak sepanjang petak secara diagonal, tetapi tidak dapat melompati bidak lain.
5) **Knight** (Kuda) memiliki gerakan mirip huruf L, yaitu memanjang dua petak dan melebar satu petak. Kudalah satu-satunya bidak yang dapat melompati bidak-bidak lain.
6) **Pawn** (Pion) dapat bergerak maju (arah lawan) satu petak ke petak yang tidak ditempati. Pada gerakan awal, pion dapat bergerak maju dua petak. Pion juga dapat menangkap bidak lawan secara diagonal, apabila bidak lawan tersebut berada satu petak di diagonal depannya. Pion memiliki dua gerakah khusus, yaitu gerakan menangkap *en passant* dan promosi.

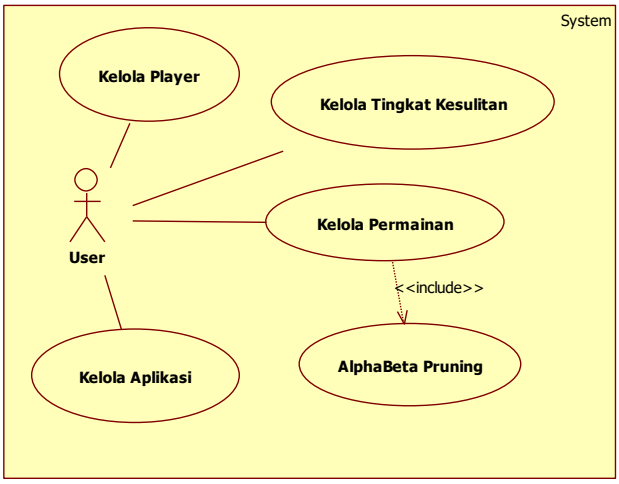
2.3. Tujuan dalam Catur

Tujuan utama dalam permainan catur adalah mencapai posisi skak mat. Hal ini bisa terjadi bila Raja terancam dan tidak bisa menyelamatkan diri ke petak lain. Tidak selalu permainan berakhir dengan kekalahan, karena bisa terjadi pula peristiwa seri atau remis di mana kedua belah pihak tidak mampu lagi meneruskan pertandingan karena tidak bisa mencapai skak mat. Peristiwa remis ini bisa terjadi berdasarkan kesepakatan maupun tidak.

3. METODE PENGEMBANGAN SISTEM

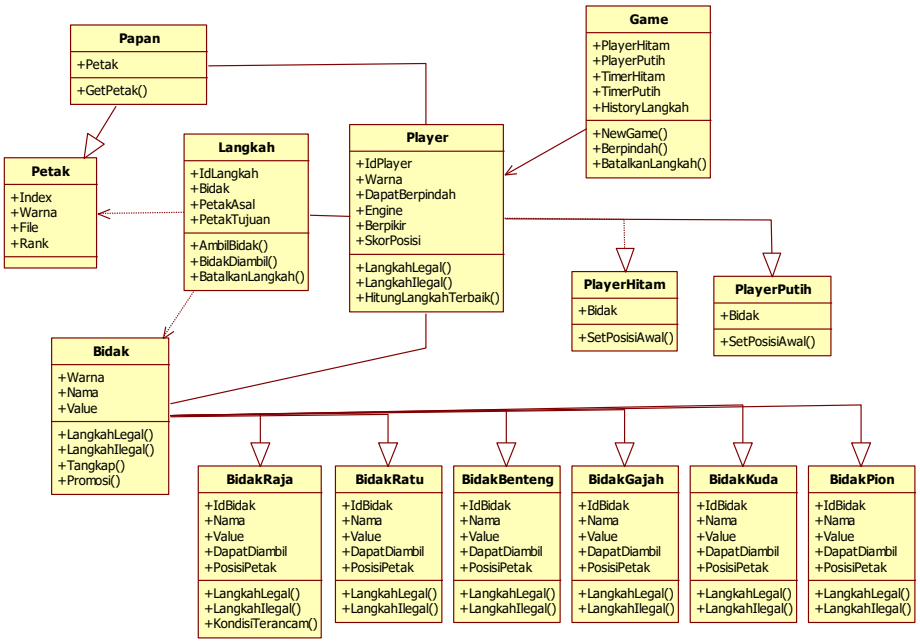
Sesuai dengan permasalahan diatas, penulis menggunakan pendekatan sistem berorientasi objek, yakni dengan membuat diagram UML untuk merancang aplikasi tersebut. Oleh karena itu akan dirancang diagram UML untuk membantu pembuatan proses aplikasi. Berikut use case diagram dan class diagram dari aplikasi yang dibuat:

3.1. Use Case Diagram



Gambar 3.19 : Use Case Diagram

3.2. Class Diagram



Gambar 3.20 : Class Diagram

4. IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Kode Program

Berikut ini dijelaskan implementasi terhadap kode program dari aplikasi yang dibuat berdasarkan perancangan user interface yang telah dijelaskan sebelumnya. Pembahasan sub-bab ini merujuk pada bab tiga yaitu proses login, proses inisialisasi data, proses atur jam kerja, proses pencarian data absensi karyawan dan proses absensi karyawan. Berikut pemaparan implementasi kode program:

1. Fungsionalitas Game Baru

Pada class ini dideklarasikan beberapa fungsi bawaan dimana ketika user membuka aplikasi (memulai permainan) ataupun user memilih menu reset maka secara otomatis class Game Baru tersebut akan menjalankan beberapa fungsi lainnya seperti berikut:

```
public Game(){
  resetTo(BoardFactory.valueOf(FASTEST, STARTING));
}
```

Fungsi diatas digunakan untuk mereset papan dan bidak catur pada permainan serta beberapa setingan lainnya seperti timer, tingkat kesulitan dll. Adapun fungsi dari resetTo tersebut diteruskan ke fungsi berikut ini.

```
public void resetTo(final MoveGenerator pMove){
  assert pMove != null;

  _moves.clear();
  _currentMove = _moves.size();
  _positions.clear();
```

```
    _positions.add(pMove);
    _currentPosition = _positions.size();
    if (_timer != null){
      _timer.cancel();
    }
    _blackTimer = GAME_DURATION;
    _whiteTimer = GAME_DURATION;
    _lastTimerTick = System.currentTimeMillis();
    _timer = new Timer();

_propertyChangeSupport.firePropertyChange("position",
null, null);
    _timer.scheduleAtFixedRate(new TimerTask(){
      @Override
      public void run(){
        if (getState() == State.IN_PROGRESS){
          final long time = System.currentTimeMillis();
          if (getBoard().isWhiteActive()){
            _whiteTimer -= time - _lastTimerTick;
          }
          else{
            _blackTimer -= time - _lastTimerTick;
          }
          _lastTimerTick = time;

_propertyChangeSupport.firePropertyChange("timer",  null,
null);
        }
      }
    }, 250, 1000);
  }
```

2. Fungsionalitas Pilih Warna

   Pada class ini dideklarasikan sebuah fungsi bernama getPlayer dengan parameter pColor bertipe boolean, dimana fungsi ini memiliki percabangan untuk menentukan apakah player merupakan bidak putih, jika tidak maka *return* nilai berupa bidak hitam.

```
  public Player getPlayer(final boolean pColor){
    if (pColor){
      return _whitePlayer;
    }
    return _blackPlayer;
  }
```

3. Fungsionalitas Pilih Player

   Pada class ini dideklarasikan sebuah fungsi bernama player yang memiliki dua buah attribut yaitu variable *_white* dan variable *_name*, dimana variable *_white* digunakan untuk memuat warna dan variable *_name* digunakan untuk memuat player apakah human atau computer. Pada class tersebut terdapat fungsi isWhite yang digunakan untuk memberikan nilai balik dari warna yang dipilih oleh player.

```
  public Player(final boolean pColor){
    _white = pColor;
    _name = "";
  }
  public boolean isWhite(){
    return _white;
  }
```

4. Fungsionalitas Pilih Level

Fungsi ini digunakan untuk menentukan level atau tingkat kesulitan dari computer berdasarkan dengan algoritma yang diterapkan, dalam hal ini adalah algoritma AlphaBeta Prunning. Untuk menentukan level dari computer itu digunakan pohon graf dengan kedalaman atau depth sebagai indikator utamanya, pada fungsi kedalaman tersebut digunakan fungsi sebagai berikut.

```
  if (pDepth == 0){
     return getHeuristic().evaluate(pMove, trait);
  }
```

Berdasarkan kedalaman tersebut, maka dilakukan fungsi evaluasi terhadap pohon graf dari algoritma AlphaBeta tersebut, adapun fungsi dari algoritma tersebut yaitu.

```
  final int note = -alphabeta(pMove.derive(mvt, true),
pDepth - 1, -pBeta, -alpha);
  if (note > res){
    res = note;
    if (res > alpha){
       alpha = res;
       if (alpha > pBeta){
          if (killer != null){
             killer.put(mvt);
          }
          return res;
       }
    }
  }
```

5. Fungsionalitas Batalkan Langkah

Fungsi untuk membatalkan langkah maupun meneruskan langkah terdapat pada class Game.java dimana class ini memuat fungsi untuk membatalkan langkah dengan nama goPrevious() maupun meneruskan langkah dengan nama goNext().

```
  public void goNext(){
    if (_currentMove < _moves.size()){
      _currentMove++;
      _currentPosition++;

_propertyChangeSupport.firePropertyChange("position",
null, null);
    }
  }
```

```
  public void goPrevious(){
    if (_currentMove > 0){
      _currentMove--;
      _currentPosition--;

_propertyChangeSupport.firePropertyChange("position",
null, null);
    }
  }
```

6. Fungsionalitas Tutup Aplikasi

Berikut ini dijelaskan fungsi dari class ExitAction.java, dimana pada class ini terdapat dua fungsi utama yaitu fungsi ExitAction dan fungsi actionPerformed. ExitAction digunakan untuk menangani input dari class Game.java dan untuk mendefinisikan icon tersebut.

```
  ExitAction(final UI pUI){
    super("exit", pUI);
    putValue(Action.SMALL_ICON, ICON16);
    putValue(Action.LARGE_ICON_KEY, ICON22);
  }
```

Sedangkan fungsi actionPerformed digunakan untuk menampilkan notifikasi setelah user menekan menu exit.

```
  public void actionPerformed(final ActionEvent pEvent){
    assert pEvent != null;
    if     (JOptionPane.showConfirmDialog(getMainFrame(),
getI18NString("dialog.exit.question"),
getI18NString("dialog.exit.title"),
JOptionPane.YES_NO_OPTION,  JOptionPane.QUESTION_MESSAGE,
ICON32) == JOptionPane.YES_OPTION){
      getUI().stop();
    }
  }
```

4.2. Implementasi Program

Tampilan utama aplikasi permainan catur ini dimulai dengan memunculkan papan catur lengkap dengan bidak-bidak nya baik untuk bidak hitam maupun bidak putih, secara default aplikasi tersebut memuat bidak putih yang dapat dimainkan oleh manusia (dalam hal ini pengguna) dan bidak hitam yang dimainkan oleh komputer. Dalam permainan nya pengguna dapat memilih untuk bermain melawan pengguna lainnya lagi (human vs human), pengguna melawan komputer (human vs computer), maupun komputer dengan komputer (computer vs computer). Berikut tampilan form utama dari aplikasi permainan catur yang dibuat:

Gambar 4.1 : Form utama aplikasi permainan catur

4.3. Pengujian Black Box

Pengujian Black-Box merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, penguji dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. Pengujian ini digunakan untuk melakukan test fungsionalitas serta untuk memeriksa validasi terhadap aplikasi yang dibangun. Berdasarkan pembahasan pada impementasi user interface sebelumnya akan diuji validasi sesuai dengan cara pengujian yang ada sehingga hasilnya sesuai dengan harapan.

Pada pengujian Black Box ini fungsionalitas dari aplikasi dibagi menjadi dua bagian yaitu Fungsionalitas Menu dan Fungsionalitas Permainan. Adapun fungsi yang diuji dari Fungsionalitas Menu ditujukan seperti dibawah ini:

Tabel 4.1 : Pengujian Black Box Fungsionalitas Menu

No	Fungsi	Skenario uji	Hasil pengujian	Ket
1	Pilih warna	User memilih warna bidak yang akan digunakan dalam permainan	Muncul form player pada button warna (bidak) yang di klik	Valid
2	Pilih *player*	User memilih jenis player apakah *human* atau *computer*	Tampilkan pilihan *combobox* pada bidak warna apakah memilih *human* atau *computer*	Valid
3	Pilih *level*	User memilih tingkat kesulitan jika memilih *computer* sebagai lawan	Tampilkan menu jSlider untuk memilih tingkat kesulitan *computer*	Valid
4	*Game* baru	User memilih sub-menu *Reset* pada menu *File*	Semua bidak pada papan catur dan timer kembali ke settingan awal	Valid
5	Batalkan langkah	User memilih button previous ataupun next	Posisi bidak berubah ke posisi sebelumnya, atau	Valid

		ketika permainan tengah berlangsung	sesudahnya (jika menekan button previous sebelumnya)	
6	Tutup aplikasi	*User* memilih *sub-menu* Exit untuk keluar dari aplikasi	Aplikasi berhasil ditutup dan dihapus dari memori	Valid

Pada pembahasan ini Fungsionalitas Permainan mengarah pada pola pergerakan dari bidak catur baik untuk bidak putih maupun bidak hitam, baik untuk *player human* maupun *player computer*. Adapun fungsi yang diuji dari fungsionalitas Permainan ditujukan seperti dibawah ini:

Tabel 4.2 : Pengujian Black Box Fungsionalitas Permainan

NO	Fungsi	Skenario uji	Hasil pengujian	Ket
1	Raja	Pergerakan normal	Raja berpindah satu petak baik secara horizontal, vertikal maupun diagonal	Valid
		Pergerakan makan	Raja dapat memakan bidak lawan yang bebas dengan jarak satu petak	Valid
		Rokade dekat	Raja berpindah dua petak dan Benteng berpindah dua petak	Valid
		Rokade jauh	Raja berpindah dua petak dan Benteng berpindah tiga petak	Valid
		Petak rokade terancam	Raja tidak dapat melakukan rokade pada petak terancam yang dilewati raja	Valid
		Remis	Hanya raja yang dapat berpindah namun *player* tidak melakukan skak terlebih dahulu dan semua petak dimana raja akan berpindah terancam	Valid
		Skak mat	Player melakukan ancaman (skak) pada raja dan raja tidak memiliki petak aman untuk berpindah	Valid
2	Ratu	Pergerakan normal	Ratu berpindah secara horizontal, vertikal maupun diagonal dan tidak dapat melangkahi bidak	Valid
		Pergerakan makan	Ratu dapat memakan bidak lawan pada jangkauan horizontal, vertikal maupun diagonal	Valid
3	Benteng	Pergerakan normal	Benteng berpindah secara horizontal maupun vertikal dan tidak dapat melangkahi bidak	Valid
		Pergerakan makan	Benteng dapat memakan bidak lawan pada jangkauan horizontal maupun vertikal	Valid
		Rokade dekat	Raja berpindah dua petak dan Benteng berpindah dua petak	Valid

		Rokade jauh	Raja berpindah dua petak dan Benteng berpindah tiga petak	Valid
4	Gajah	Pergerakan normal	Gajah berpindah secara diagonal dan tidak dapat melangkahi bidak	Valid
		Pergerakan makan	Gajah dapat memakan bidak lawan pada jangkauan diagonal	Valid
5	Kuda	Pergerakan normal	Kuda berpindah dua petak dan berbelok satu petak serta dapat melangkahi bidak	Valid
		Pergerakan makan	Kuda dapat memakan bidak lawan pada jangkauan dua petak dan berbelok satu petak	Valid
6	Anak	Pergerakan awal	Anak dapat berpindah dua petak kedepan pada awal pergerakan dan tidak dapat melangkahi bidak	Valid
		Pergerakan makan	Anak dapat memakan bidak lawan secara diagonal pada petak di depannya	Valid
		Pergerakan normal	Anak berpindah satu petak kedepan ketika permainan tengah berlangsung dan tidak dapat melangkahi bidak	Valid
		En-Peassant	Anak dapat memakan (secara diagonal) bidak pada posisi sejajar	Valid

4.4. Pengujian White Box

Pada pengujian *white box* ini fungsi yang diuji yaitu untuk melakukan absensi karyawan dengan menggunakan teknik basis path testing, basis path testing mengijinkan pendesain kasus uji untuk mendapatkan perkiraan *logic* yang kompleks dari desain prosedural dan menggunakan perkiraan ini untuk mendefenisikan aliran eksekusi. Berikut kode program untuk proses evaluasi dengan menggunakan algoritma AlphaBeta Prunning:

`private int alphabeta(final MoveGenerator pMove, final int pDepth, final int pAlpha, final int pBeta){`	

`assert pMove != null;`
`assert pDepth >= 0;`
`assert pAlpha <= pBeta;`
`final boolean trait = pMove.isWhiteActive();`
	1

Deklarasi beberapa variable pembantu	

`if (pDepth == 0){`
`  return getHeuristic().evaluate(pMove, trait);`
`}`
	2

Validasi apakah variable pDepth samadengan 0, jika ya maka return hasil evaluasi	
`final Move [] shots = pMove.getValidMoves(trait);` `final int l = shots.length;`	3
Deklarasi variable shots bertipe Move dan variable l bertipe integer	
`if (l == 0){` `  return getHeuristic().evaluate(pMove, trait);`	4

}	
Validasi apakah variable l samadengan 0, jika ya maka return hasil evaluasi	

int res = MATE_VALUE - 1;
final Comparator<Move> tri = getMoveSorter();
final KillerMoveSorter killer;
	5

Deklarasi beberapa variable pembantu	

if (tri instanceof KillerMoveSorter){
killer = (KillerMoveSorter) tri;
}
	6

Validasi apakah variable tri merupakan instance dari KillerMoveSorter	

else{
killer = null;
}
	7

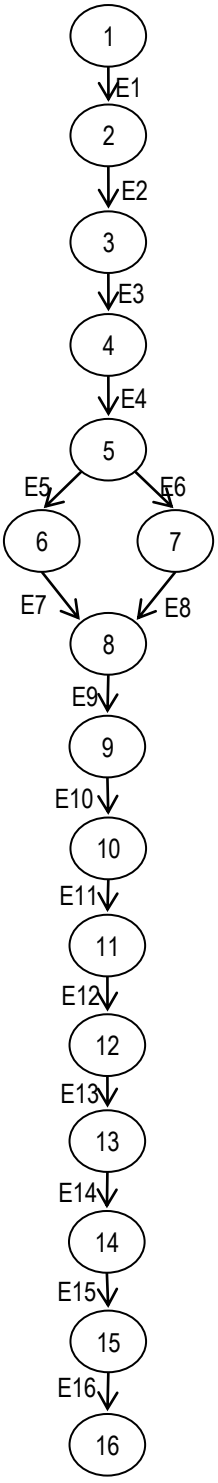
Selain itu maka isi variable killer dengan null	

Arrays.sort(shots, tri);
addHalfmove(l);
int alpha = pAlpha;
	8

Deklarasi beberapa variable pembantu	
for (final Move mvt : shots){ final int note = -alphabeta(pMove.derive(mvt, true), pDepth - 1, -pBeta, -alpha);	9
Untuk setiap langkah	
if (note > res){ res = note;	10
Jika nilai dari variable note lebih besar dari variable res, maka ubah nilai dari variable res menjadi nilai note	
if (res > alpha){ alpha = res;	11
Jika nilai dari variable res lebih besar dari variable alpha, maka ubah nilai dari variable alpha menjadi nilai res	
if (alpha > pBeta){	12
Jika nilai dari variable alpha lebih besar dari variable pBeta, maka...	

if (killer != null){
killer.put(mvt);
}
	13

Jika nilai dari variable killer tidak samadengan null, maka beri parameter killer dengan variable mvt	
return res; } } }	14
Return variable res sebagai nilai balik	
}	15
return res;	16
Return variable res sebagai nilai balik	
}	



Gambar 4.10 : Notasi Flow Graph

Keterangan Gambar :

Node (N) = 16
Edge (E) = 16
Predicate Node (P) = 1

a. V(G) = E – N + 2
   V(G) = 16 – 16 + 2
        = 2
b. V(G) = P + 1
   V(G) = 1 + 1
        = 2

Berdasarkan hasil perhitungan cyclomatic complexity terdapat dua independent path (jalur) yaitu:

Path 1 : 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 - 16
Path 2 : 1 – 2 – 3 – 4 – 5 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 - 16

Pada pengujian *white box* ini fungsi yang diuji yaitu untuk melakukan pengambilan langkah oleh bidak/kuda. Berikut kode program untuk menentukan/menetapkan langkah kuda:

Posisi (nilai) dasar kuda pada awal permainan

```
private static final int [] KNIGHT_POSITIONS = {
            -50, -30, -30, -30, -30, -30, -30, -50, // a1 ... h1
            -30, -20, -20, -20, -20, -20, -20, -30, // a2 ... h2
            -20, 0, 20, 20, 20, 20, 0, -20, // a3 ... h3
            -20, 0, 20, 20, 20, 20, 0, -20, // a4 ... h4
            -20, 0, 10, 20, 20, 10, 0, -20, // a5 ... h5
            -20, 0, 10, 10, 10, 10, 0, -20, // a6 ... h6
            -20, -10, 0, 0, 0, 0, -10, -20, // a7 ... h7
            -40, -20, -20, -20, -20, -20, -20, -40, // a8 ... h8
};
```

Pengambilan langkah kuda pada awal permainan

```
case KNIGHT :
    if (traitPiece)
    {
        pos = KNIGHT_POSITIONS[s.getIndex()];
    }
    else
    {
        pos = KNIGHT_POSITIONS[((RANK_COUNT - 1) - s.getRank()) * FILE_COUNT + s.getFile()];
     }
break;
```

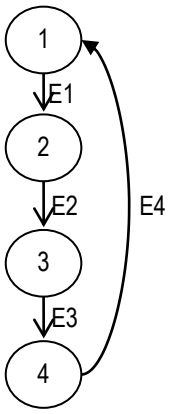
Posisi (nilai) dasar kuda pada pertengahan permainan

```
private static final int [] KNIGHT_POSITIONS = {
            -10, -5, -3, -1, -1, -3, -5, -10, // a1 ... h1
            -5, 0, 0, 3, 3, 0, 0, -5, // a2 ... h2
            -3, 0, 5, 5, 5, 5, 0, -3, // a3 ... h3
            -1, 1, 5, 10, 10, 5, 1, -1, // a4 ... h4
            -1, 1, 7, 12, 12, 7, 1, -1, // a5 ... h5
            -3, 0, 5, 7, 7, 5, 0, -3, // a6 ... h6
            -5, 0, 0, 3, 3, 0, 0, -5, // a7 ... h7
            -10, -5, -3, -1, -1, -3, -5, -10, // a8 ... h8
 };
static
{
```

```
        assert KNIGHT_POSITIONS.length == 64;
}
```

Pengambilan langkah kuda pada pertengahan permainan

```
case KNIGHT :
        if (traitPiece)
        {
                pos = KNIGHT_POSITIONS[s.getIndex()];
        }
        else
        {
                pos = KNIGHT_POSITIONS[((RANK_COUNT - 1) - s.getRank()) * FILE_COUNT +
        s.getFile()];
        }
        if (nbPieces >= MIDDLE_GAME)
        {
                mob = pMove.getKnightTargets(s, traitPiece).length * 4;
        }
        else
        {
                mob = 0;
        }
break;
```



Gambar 4.11: Notasi Flow Graph Kuda

Keterangan Gambar :

Node (N) = 4
Edge (E) = 4
Predicate Node (P) = 1

a. V(G) = E – N + 2
   V (G) = 4 – 4 + 2
   = 2
b. V(G) = P + 1
   V (G) = 1 + 1
   = 2

Berdasarkan hasil perhitungan cyclomatic complexity terdapat satu independent path (jalur) yaitu: Path 1 : 1 – 2 – 3 – 4

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Penelitian ini bertujuan untuk membangun aplikasi permainan catur dengan menggunakan algoritma *AlphaBetaPruning* (optimalisasi dari algoritma MiniMax). Berdasarkan hasil yang diperoleh dari perencanaan, pembuatan dan pengujian dapat disimpulkan beberapa hal, diantaranya:

1. Penggunaan algoritma AlphaBetaPruning untuk pengambilan langkah yang dilakukan oleh komputer tidak memakan waktu yang lama.
2. Tingkat kesulitan (level) komputer disesuaikan dengan kedalaman pohon pencarian yang direalisasikan dengan angka numerik dari tiga sampai lima.

5.2. Saran

Mengingat masih adanya ketidaksempurnaan pada aplikasi ini serta demi kepentingan pengembangan aplikasi ini sendiri, maka dapat diberikan beberapa saran yang mungkin dapat dipertimbangkan :

1. Diharapkan agar terdapat beberapa algoritma lainnya sebagai basis kecerdasan buatan komputer untuk pengambilan langkah yang dilakukan.
2. Diharapkan penelitian ini dapat dilanjutkan agar didapat hasil yang lebih optimal khususnya dalam konteks kecerdasan buatan dan permainan.
3. Diharapkan aplikasi ini dapat dikembangkan lagi lebih lanjut seperti dengan menambahkan beberapa fitur baru sehingga lebih kaya akan pengalaman.

6. DAFTAR PUSTAKA

Ayuningtyas, Nadhira.2008.Algoritma minimax dalam permainan checkers. Dalam Strategi Algoritmik 2008.Bandung, Indonesia: ITB.

Munir, Rinaldi.2007.Strategi Algoritmik.Teknik Informatika ITB: Bandung

Kusumadewi, Sri.2003.Artificial Intelligence (Teknik dan Aplikasinya). Yogyakarta: Graha Ilmu.

Dana Cremer.2007.“The Application of Artificial Intelligent to Solve a Physical Puzzle”.Departement of Komputer and Information Sciences. Indiana University South Bend.

Zainal A. Hasibuan.2007.“Metodologi Penelitian Pada Bidang Ilmu Komputer Dan Teknologi Informasi”.Fasilkom UI: Depok