ALGORITMA *MINIMAX* DALAM PENGAMBILAN KEPUTUSAN PADA PERMAINAN *TIC-TAC-TOE*

Khoirush Sholih Ridhwaana Akbar
(13505120)

Program Studi Teknik Informatika
STEI ITB
Alamat : Jl. Pak Gatot IV no.68 , KPAD Bandung
e-mail: if15120@students.if.itb.ac.id

ABSTRAK

Algoritma *minimax* merupakan metode yang sangat terkenal dalam pengambilan keputusan untuk meminimalisasi maksimumnya peluang kalah atau rugi. [3]
Algoritma *minimax* merupakan algoritma yang cukup terkenal dalam bidang kecerdasan buatan. Dimana dengan algoritma tersebut komputer dapat mengambil keputusan terbaik untuk menyelesaikan masalah.
Masalah yang akan diangkat disini adalah *AI* untuk permainan *tic-tac-toe* , dimana *AI* tersebut tidak akan pernah kalah.
Dengan algoritma *minimax* ini, pohon solusi akan dibuat dari awal permainan sampai akhir permainan dimana semua kemungkinan kondisi dijadikan simpul dari pohon solusi, sehingga *AI* tinggal memilih langkah yang akan menuntunnya ke hasil akhir berupa kemenangan atau setidaknya seri.

Kata kunci: *Artificial Inteligence (AI)*, *tic-tac-toe* , *minimax , heuristic, resource, state.*

1. PENDAHULUAN

Bermain *game* merupakan salah satu aktifitas yang sangat disukai oleh sebagian besar masyarakat didunia ini. Alasan mereka bermain *game* tentunya berbeda-beda, ada yang untuk melepas lelah, ada juga yang memang suka atau hobi bermain *game.*

Dengan berkembangnya teknologi sekarang ini, *game-game* ini tidak hanya dapat kita jumpai pada kehidupan nyata, tapi juga dapat kita jumpai d idalam dunia maya. Jenis nya pun semakin banyak dan bervariasi. Salah satu yang cukup menarik perhatian adalah permainan komputer.

Permainan-permainan berbasis komputer ini juga bermacam-macam. Salah satu kelebihannya adalah kita tidak harus mencari orang untuk menjadi lawan tanding jika ingin bermain karena permainan berbasis komputer ini sudah mendukung *single-player mode* dimana kita dapat bermain sendiri melawan komputer yang dirancang untuk dapat berlaku seperti pemain manusia atau yang sering dikenal dengan *Artificial Inteligince (AI)*. Contoh-contoh permainan yang menggunakan *AI* adalah permainan catur, go, othello, *checkers, bridge, tic-tac-toe* dan lain sebagainya.

Untuk membuat pemain merasa seperti melawan pemain manusia lainnya, diperlukan suatu algoritma yang dapat membuat *AI* ini mampu mengambil keputusan yang terbaik agar dapat mengalahkan pemain atau setidaknya menghalau pemain menang.

Algoritma *minimax* ini merupakan algoritma yang sangat sering dipakai untuk permasalah tersebut. Dan permainan *tic-tac-toe* merupakan salah satu contoh yang baik dan cukup sederhana untuk kita mengerti bagaimana cara kerja dan efeknya.

1.1 Linkup Masalah

Linkup permasalahan yang akan dibahas pada makalah ini adalah penggunaan algoritma *minimax* untuk membuat *AI* yang dapat mencari dan menentukan keputusan terbaik dalam permainan *tic-tac-toe* sedemikian sehingga *AI* tersebut tidak akan pernah kalah.

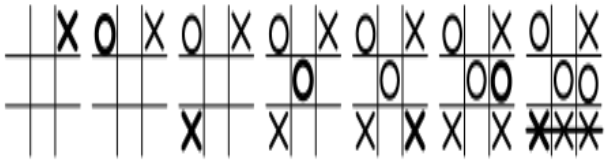
2. DASAR TEORI

2.1 Permainan *Tic-Tac-Toe*

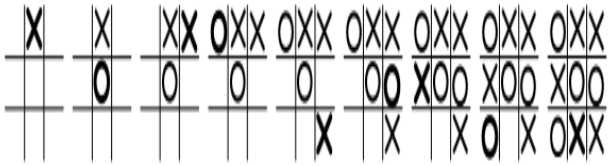
Permainana *tic-tac-toe* merupakan permainan berjenis *board-game* berukuran 3x3. Pemain harus mengisi sel-sel tersebut sedemikian sehingga karakter yang dimasukkan pemain tersebut dapat membentuk suatu garis lurus horizontal, vertikal, ataupun juga diagonal. Permainan ini biasanya dimainkan oleh 2 orang pemain, tapi pada versi permainan komputer, pemain lawan dapat digantikan oleh komputer. Dalam permainan ini hasilnya dapat berupa menang,kalah, ataupun seri.

Disini dengan adanya *AI* yang mampu meminimalisir kemungkinan untuk pemain menang, permainan ini akan menjadi sangat sulit untuk dimenangkan oleh pemain. Bahkan kemungkinan terbaik untuk pemain hanyalah seri. Dengan kata lain dengan menggunakan algoritma *minimax* ini, komputer tidak akah pernah kalah.

Berikut contoh permainan *tic-tac-toe* :



Gambar 1. Contoh kondisi menang permainan *tic-tac-toe.*



Gambar 2. Contoh kondisi seri permainan *tic-tac-toe.*

2.2 Algoritma *Minimax*

Algoritma minimax merupakan basis dari semua permainan berbasis *AI* seperti permainan catur misalnya. *AI* permainan catur tentunya sudah sangat terkenal dimana *AI* tersebut bahkan dapat mengalahkan juara dunia sekalipun. Pada algoritma minimax, pengecekan akan seluruh kemungkinan yang ada sampai akhir permainan dilakukan. Pengecekan tersebut akan menghasilkan pohon permainan yang berisi semua kemungkinan tersebut. Tentunya dibutuhkan *resource* yang berskala besar untuk menangani komputasi pencarian pohon solusi tersebut berhubung kombinasi kemungkinan untuk sebuah permainan catur pada setiap geraknya sangat banyak sekali.

Berbeda dengan permainan catur, permainan *tic-tac-toe* ini mempunyai lebih sedikit kemungkinan solusi, sehingga kita akan mempunyai cukup komputasi untuk memainkan setiap kombinasi langkah dari setiap posisi dan kondisi. Namun hal ini dapat dihindari dengan membatasi sejauh mana komputer akan menganalisis hasil dari langkah-langkah yang mungkin (menentukan kedalaman pohon). Tetapi dengan hal ini, kita harus menambah kedalaman pohon tersebut setiap langkahnya agar kedalaman pohon pada *state* tersebut sama dengan *state* sebelumnya.

Algoritma minimax ini bekerja secara rekursif dengan mencari langkah yang akan membuat lawan mengalami kerugian minimum. Semua strategi lawan akan dihitung dengan algoritma yang sama dan seterusnya. Ini berarti, pada langkah pertama komputer akan menganalisis seluruh pohon permainan. Dan untuk setiap langkahnya, komputer akan memilih langkah yang paling membuat lawan mendapatkan keuntungan minimum, dan yang paling membuat komputer itu sendiri mendapatkan keuntungan maksimum.

Dalam penentuan keputusan tersebut dibutuhkan suatu nilai yang merepresentasikan kerugian atau keuntungan yang akan diperoleh jika langkah tersebut dipilih. Untuk itulah disini digunakan sebuah fungsi *heurisitic* untuk mengevaluasi nilai sebagai nilai yang merepresentasikan hasil permainan yang akan terjadi jika langkah tersebut dipilih. Biasanya pada permainan *tic-tac-toe* ini digunakan nilai 1,0,-1 untuk mewakilkan hasil akhir permainan berupa menang, seri, dan kalah. Dari nilai-nilai *heuristic* inilah komputer akan menentukan simpul mana dari pohon permainan yang akan dipilih, tentunya simpul yang akan dipilih tersebut adalah simpul dengan nilai *heuristic* yang akan menuntun permainan ke hasil akhir yang menguntungkan bagi komputer.

Berikut garis besar algoritma minimax secara umum :

> Cari langkah yang dengan nilai maksimum
> IF langkah tersebut merupakan langkah kemenangan THEN pilih lagkah tersebut.
> ELSE
> FOR EACH kemungkinan langkah yang ada
> Cari langkah lawan yang bernilai minimum.
> RETURN nilai dari langkah tersebut.
> Pilih langkah yang bernilai maksimum dari langkah-langkah tersebut. [2]

Pemakaian algoritma umum diatas untuk permainan *tic-tac-toe* adalah sebagai berikut :

> IF ada langkah kemenangan THEN pilih langkah tersebut.
> ELSE IF lawan mempunyai 2 spot terisi dalam satu garis dengan spot ketiga masih kosong THEN tutup langkah tersebut (isi spot kosong ketiga tersebut).
> ELSE melangkah ke *state* yang mempunyai kemungkinan menang tertinggi (berdasarkan nilai *heuristic* yang dibangkitkan). [2]

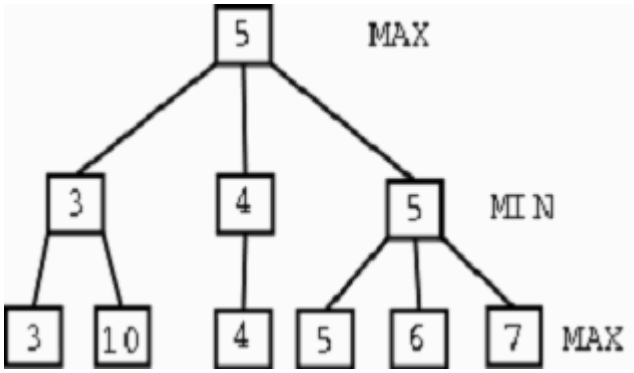
Dilihat dari algoritma diatas, nilai *heuristic* yang dibangkitkan akan sangat mempengaruhi analisis dari *AI*

tersebut. Oleh karena itu, semakin bagus fungsi *heuristic* maka semakin bagus pula analisis *AI* tersebut, dan semakin sulit pula *AI* untuk dikalahkan.

3. PENERAPAN ALGORITMA *MINIMAX*

Anggap lah ada 2 pemain A dan B. Jika pemain A bisa menang dalam 1 langkah, maka langkah tersebut adalah langkah kemenangannya. Jika pemain B mengetahui bahwa langkah tersebut akan mengarahkan ke hasil akhir dimana pemain A akan menang, dan di lain kondisi ada langkah lain yang akan mengarahkan ke hasil akhir seri, maka langkah terbaik untuk pemain B adalah langkah yang akan mengarahkan hasil akhir permainan ke hasil seri. Di setiap tahap algoritma ini mengasumsikan bahwa pemain A mencoba untuk memaksimalisasi peluang menang. Di lain pihak, pada giliran berikutnya pemain B akan mencoba meminimalisir peluang menang untuk pemain A. Oleh karena itu, A disebut juga *maximizing player* (MAX) dan B disebut juga *minimizing player* (MIN).

Pembentukan pohon pencarian solusi digunakan dengan menggunakan konsep *depth-first*, dimulai dari awal permainan sampai akhir permainan. Setelah itu, posisi akhir permainan dievaluasi melalui sudut pandang MAX seperti gambar dibawah ini :



Gambar 2. Representasi pohon pencarian pada algoritma *minimax.*

Setelah itu nilai dari setiap simpul diisi dari bawah ke atas dengan nilai yang sudah dievaluasi oleh fungsi *heuristic*. Simpul milik pemain A (MAX) menerima nilai maksimum dari simpul-simpul anaknya. Simpul milik pemain B (MIN) akan memilih nilai minimum dari simpul anak-anaknya. [5]

Berikut *pseudocode* dari algoritma yang digunakan :

```
MinMax (GamePosition game) {
  return MaxMove (game);
}
```

```
MaxMove (GamePosition game) {
  if (GameEnded(game)) {
    return EvalGameState(game);
  }
  else {
   best_move < - {};
   moves <- GenerateMoves(game);
   ForEach moves {
     move <- MinMove(ApplyMove(game));
     if (Value(move) > Value(best_move)){
        best_move < - move;
     }
   }
   return best_move;
  }
}

MinMove (GamePosition game) {
  best_move <- {};
  moves <- GenerateMoves(game);
  ForEach moves {
     move <- MaxMove(ApplyMove(game));
     if (Value(move) > Value(best_move)){
        best_move < - move;
     }
  }

  return best_move;
}
```

Values disini merepresentasikan sebagaimana bagusnya suatu langkah dalam permainan. Jadi, pemain A (MAX) akan memilih langkah dengan nilai paling tinggi diakhir. Di sisi lain, pemain B (MIN) akan melakukan serangan balik dengan memilih langkah yang terbaik untuknya, yakni meminimalisir hasil dari langkah yang dipilih pemain A. [5]

4. HASIL ANALISIS

AI akan selalu memilih langkah yang dapat meminimalisir kemungkinan pemain (manusia) untuk menang dan memblok semua langkah kemenangan pemain. Dengan demikian permainan akan selalu seri apabila pemain cukup teliti dalam menentukan langkah. Namun jika pemain melakukan langkah yang salah, maka *AI* akan langsung menggunakan kesempatan tersebut

untuk mengambil langkah yang akan mengarahkannya ke hasil akhir berupa kemenangan atau setidaknya seri.

5. KESIMPULAN

1. Algoritma *minimax* merupakan algoritma yang sangat bagus dan cocok untuk pengambilan keputusan oleh *AI*, terutama dalam permainan *n-player* (n>=2).
2. Algoritma *minimax* menggunakan konsep DFS dalam pembentukan pohon solusi.
3. Pohon solusi dibentuk dari awal permainan sampai akhir permainan.
4. Untuk permainan yang terbilang cukup kompleks seperti permainan catur, pembentukan pohon solusi dari awal permainan sampai akhir permainan akan sulit direalisasikan berhubung kemungkinan yang ada sangat besar. Oleh karena itu, kita dapat membatasi dalamnya pohon solusi pada suatu tahap untuk mempercepat kinerja pengambilan keputusan.
5. Semakin akurat fungsi *heuristic* yang digunakan, semakin baik pula pengambilan keputusan yang dilakukan oleh *AI*.
6. Dengan menggunakan algoritma *minimax* untuk *AI* dalam permainan *tic-tac-toe*, pemain (manusia) tidak akan pernah menang melawan *AI* tersebut.

REFERENSI

[1] Munir, Rinaldi.2006. "Strategi Algoritmik". Program Studi Informatika, Institut Teknologi Bandung.

[2] Kevin McGee, "*Advance Game Programming* : *AI*", Desember 9, 2005.

[3] Wikimedia Foundation, Inc. *"A * Search Algorithm"*. **http://en.wikipedia.org/wiki/Astar_search_algorithm**. Diakses tanggal 17 Mei 2006 pukul 10.45

[4] **www.stancford.edu~msirotasocominimax.html** Diakses tanggal 15 Mei pukul 15.00

[5] **http://ai-depot.com/articles/minimaxa-explained/1/** Diakses tanggal 15 Mei pukul 15.00