Richard Rivaldo
13519185 – K04

Minimax Algorithm Implementation on TicTacToe

1. Jelaskan secara umum algoritma minimax!

   Algoritma Minimax adalah salah satu algoritma yang digunakan dalam penentuan keputusan atau *decision making*. Algoritma ini secara umum termasuk ke dalam algoritma dalam algoritma Intelejensia Buatan. Ketika melihat beberapa penjelasan cara kerja algoritma ini, yang terlintas di benak Saya adalah Algoritma *Backtracking* yang baru saja diajarkan pada mata kuliah Strategi Algoritma.
   Setelah melakukan penelusuran literatur, ternyata hal tersebut memang benar, dan biasanya, algoritma Minimax ini digunakan dalam permainan *turn-based* antara dua pemain. Setiap gerakan yang dilakukan oleh kedua pemain akan memiliki dampak terhadap hasil akhir permainan. Oleh karena itu, algoritma ini digunakan untuk memilih langkah yang memaksimalkan keuntungan untuk suatu pihak dan dengan demikian pula meminimumkan keuntungan pihak lawan. Dalam kata lain, persoalan yang ingin dipecahkan oleh algoritma ini adalah persoalan optimasi.
   Implementasi algoritma ini bisa dilakukan dengan menggunakan struktur pohon maupun dengan menggunakan *local search* secara rekursif. Dalam algoritma ini, ada dua buah aktor yang ditentukan, yaitu *Maximizer* dan *Minimizer*. Jika dikaitkan dengan permainan, maka *bot* yang dibangun akan berperan sebagai *Maximizer* dengan memilih langkah yang memaksimalkan keuntungannya, dan sebaliknya untuk pemain sebagai *Minimizer*. Pembangkitan simpul pada pohon ini akan dilakukan berselang-seling antara keduanya, sampai mencapai *terminal state*.
   Terminal state dapat dikatakan sebagai status pada saat hasil akhir dari pengambilan keputusan sudah ditemukan. Dalam permainan, maka status ini dapat diibaratkan ketika permainan telah berakhir. Oleh karena itu, representasi status ini pada pohon digambarkan sebagai simpul daun pada setiap pembangkitan *depth-first*. Sesuai dengan *nature* dari algoritma *backtracking*, maka ketika simpul daun sudah diekspan, akan dilakukan *backtrack* ke simpul-simpul yang telah berperan sebagai *Minimizer* dan *Maximizer* tersebut.
   Pada saat *backtracking*, setiap simpul dalam akan diisi nilainya sampai kembali ke akar pohon. Dengan demikian, dibutuhkan sebuah fungsi heuristik yang bisa memberikan penilaian terhadap status permainan berdasarkan peran simpul di dalam pohon tersebut. Fungsi ini disebut dengan fungsi evaluasi, dan setiap Algoritma Minimax bebas mengimplementasikan fungsi heuristik apapun, tetapi tentunya tetap akan memengaruhi performansi dari agen yang dibangun.
   Biasanya, fungsi evaluasi yang dibangun akan memberikan nilai positif yang menandakan posisi menguntungkan bagi agen, 0 jika posisi mengakibatkan hasil imbang, dan nilai negatif untuk posisi yang sangat merugikan. Ketika sudah mencapai aras terendah, maka rekursif dihentikan dan dalam kebanyakan kasus, pilihan yang memaksimumkan keuntungan suatu agen akan dipilih.
   Karena tentunya banyak sekali kemungkinan yang ada dalam sebuah permainan, maka jika semua simpul yang ada di dalam pohon yang dibangun diekspan satu persatu, algoritma ini menjadi tidak mangkus. Oleh karena itu, ekspansi simpul bisa dipangkas dengan menggunakan metode *Alpha-Beta Pruning* yang mirip dengan prinsip *bounding function*.
   Secara sederhana, *Alpha* adalah sebuah nilai tertinggi yang bisa dijamin oleh *Maximizer* pada aras tersebut atau aras yang lebih rendah (lebih dekat dengan akar). Sebaliknya, *Beta* adalah nilai terendah yang bisa dijamin oleh *Minimizer*. Biasanya, *Alpha* diinisiasi dengan nilai yang sangat kecil ($-\infty$) dan *Beta* dengan nilai yang sangat besar ($+\infty$).

Nantinya, kedua nilai ini akan digunakan sebagai parameter dalam penentuan nilai simpul *Maximizer* dan *Minimizer* sebagai *threshold* untuk pemotongan cabang tertentu yang sudah pasti akan menjamin nilai terbaik untuk masing-masing simpul. Pada dasarnya, saat berada pada simpul dengan aktor *Maximizer*, maka nilai *Alpha* yang akan dibandingkan dan diganti, dan sebaliknya dengan aktor *Minimizer* yang akan membandingkan dan mengganti nilai *Beta*.

2. Jelaskan bagaimana algoritma minimax mengambil langkah terbaik dalam permainan TicTacToe yang kalian buat!

Dalam permainan TicTacToe yang telah dibuat, telah ditentukan sejak awal bahwa pemain *Human* akan memiliki simbol "O" dan *Bothicc*, *bot* AI yang dirancang untuk permainan TicTacToe ini memiliki simbol "X". Pada permainan yang dirancang, pemain yang memulai giliran terlebih dahulu ditentukan secara acak. Ketika pemain *Human* mendapat giliran, maka pengguna akan dimintai masukan indeks baris dan kolom yang akan diisikan dengan "O". Adapun ketika gerakan valid dan sudah diaplikasikan ke papan permainan, maka giliran akan diberikan ke *Bothicc* dan *bot* ini akan menggunakan Algoritma Minimax untuk menentukan pilihannya.
Dalam perancangannya, algoritma ini dipisahkan seperti yang telah dijelaskan sebelumnya, menjadi *Maximizer* dan *Minimizer*. Dalam hal ini, *Maximizer* diperankan oleh *bot* tersebut, sedangkan *Minimizer* dimainkan oleh *Human* sebagai bentuk prediksi langkah oleh *bot* berdasarkan suatu langkah tertentu. Karena direncanakan akan menggunakan *Alpha-Beta Pruning* sebagai bentuk optimasi, maka kedua fungsi ini akan menerima parameter tambahan berupa *Alpha* dan *Beta*.
Untuk kedua fungsi tersebut, diinisialisasi terlebih dahulu nilai maksimum dan minimum yang sesuai untuk kedua fungsi dengan -∞ dan ∞. Setelah itu, akan diperiksa status permainan pada saat itu dengan menggunakan sebuah fungsi evaluasi. Fungsi evaluasi ini mengecek status terminal dari permainan, yaitu ketika permainan telah berakhir. Dalam hal ini, apabila hasil akhir permainan dimenangkan oleh *bot*, maka akan diberikan skor +10, 0 jika imbang, dan -10 jika permainan dimenangkan oleh *Human*. Penilaian ini nantinya akan digunakan sebagai basis rekursif untuk algoritma tersebut.
Kemudian, untuk suatu kondisi papan tertentu, akan dicari semua gerakan yang mungkin untuk dilakukan. Gerakan ini didefinisikan sebagai gerakan yang tidak keluar indeks dan dilakukan pada blok yang belum diisi. Untuk setiap gerakan yang mungkin, maka akan dilakukan sebuah percobaan gerakan *dummy* pada papan sesuai dengan properti pemain. Setelah dilakukan percobaan tersebut, akan ditinjau pengaruh gerakan tersebut terhadap keuntungan lawan. Tentunya, hal ini dilakukan dengan menggunakan *Minimizer* sebagai tolak ukurnya. Jika dilihat pada *Minimizer*, kedua fungsi ini saling memanggil satu sama lain, karena memang jika direpresentasikan sebagai pohon, kedua aktor berperan secara berganti-gantian.
Nilai minimum yang didapat dari *Minimizer* akan kemudian dibandingkan dengan nilai maksimum yang dimiliki pada saat itu. Jika nilai minimum ternyata lebih besar dibandingkan nilai maksimum, maka nilai maksimum ini akan diganti dengan nilai minimum tersebut. Selain itu, gerakan yang menghasilkan nilai maksimum tersebut juga akan dicatat. Perlu diperhatikan bahwa karena tadi dilakukan gerakan *dummy*, maka papan harus dikembalikan lagi pada posisi sebelumnya.
Tahap berikutnya adalah tahap *pruning* dengan menggunakan parameter *Alpha-Beta*. Secara *default*, kedua parameter ini masing-masing akan ditetapkan dengan nilai -∞ dan ∞. Pada *Maximizer*, nilai *Alpha* akan diisi dengan nilai maksimum antara *Alpha* dan nilai

maksimum sebelumnya. Sebaliknya, pada *Minimizer*, nilai *Beta* diisi dengan nilai minimum dari *Beta* dan nilai minimum sebelumnya.
Dengan demikian, perbandingan selanjutnya dilakukan terhadap *Alpha* dan *Beta*. Syarat dari *Alpha-Beta Pruning* adalah nilai *Alpha* lebih besar atau sama dengan nilai *Beta*. Jika pengecekan pada suatu langkah (`node`) tertentu memenuhi syarat tersebut, maka `*child node*` selanjutnya akan di-*pruning* sehingga tidak perlu dilakukan traversi lanjutan untuk langkah tersebut, dan bisa dilanjutkan dengan langkah *backtracking* untuk mengoptimasi algoritma ini. Kedua fungsi ini akan mengembalikan nilai maksimum atau nilai minimum dari fungsi evaluasi tersebut, serta mengembalikan posisi gerakan yang menghasilkan nilai tersebut.