Two Agents Search

3.1 Tujuan Instruksional

Mahasiswa dapat memformulasi pencarian untuk multi-agent, khususnya untuk dua agen.

Mahasiswa dapat membuat pohon pencarian untuk sebuah permainan

Mahasiswa dapat mengoperasikan algoritma minmax, dan dapat membuat kode programnya

Mahasiswa dapat membuat heuristik dan strategi untuk menghitung nilai heuristik

Mahasiswa dapat mengoperasikan algoritma alpha-beta, dan dapat mengevaluasi keunggulan komputasi dan optimasinya

3.2 Pencarian Adversarial (berlawanan)

Dalam bagian ini akan disusun sebuah kerangka pemikiran untuk memformulasikan permainan (lebih dari satu orang), sebgai sebuah problem pencarian. Permainan yang dipilih adalah yang memberikan alternatif bagi pemain untuk melakukan langkahnya secara bergiliran, dan berusaha untuk memaksimalkan langkahnya, dengan cara meminimalkan langkah lawan, dengan menggunakan sebuah fungsi utilitas (utility function). Permainan harus memenuhi persyaratan sebagai berikut:

Dimainkan oleh dua orang – tanpa adanya koalisi (berjalan bersamaan)

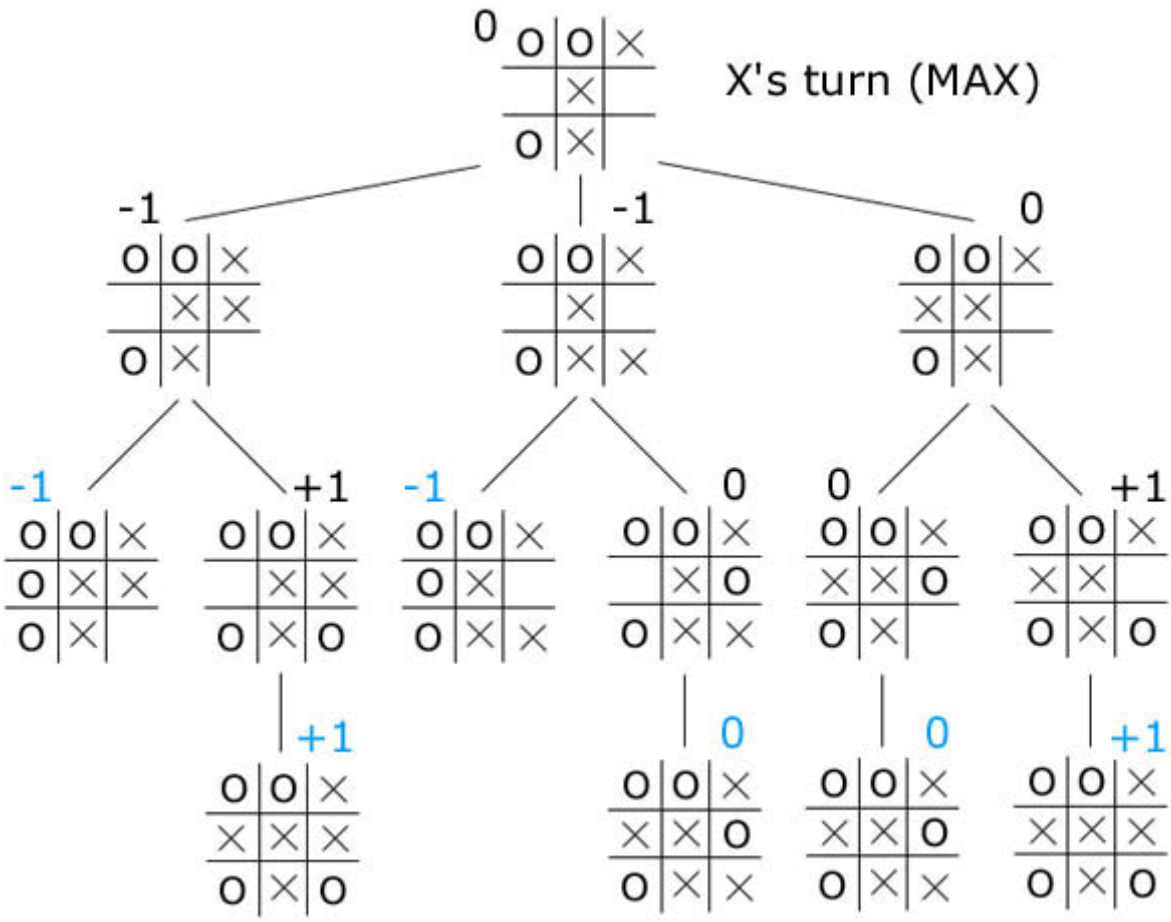
Hanya ada satu pemenang – yang lainnya kalah, mungkin bisa muncul nilai seri, namun yang dipilih adalah satu pemenang (*perfect information game*)

3.3 Pohon Permainan

Kategori permainan di atas, dapat direpresentasikan dengan sebuah pohon, yang memberikan sebuah node sebagai keadaan permainan saat ini, dan arcs (jalur) merepresentasikan langkah yang memungkinkan dari keadaan saat ini. Pohon permainan akan berisi semua langkah yang mungkin terjadi untuk pemain yang saat ini akan melangkah, dimulai dari root, dan semua langkah yang memungkinkan untuk pemain berikutnya pada anak-anak dari node, dst, sampai langkah akhir dalam permainan. Setiap langkah individu akan disebut sebagai “ply”. Daun (leaves) pada pohon akan merepresentasikan keadaan akhir (terminal), dengan hasil yang jelas (menang, kalah, seri atau mengulangi lagi). Setiap terminal memiliki skor akhir. Skor yang tinggi akan baik bagi seorang pemain (MAX). Pemain lainnya disebut sebagai MIN, selalu berusaha untuk meminimalkan skor. Sebagai contoh bisa saja nilai 1 melambangkan kemenangan untuk MAX, 0 untuk keadaan seri, dan -1 kekalahan bagi MAX.

Contoh permainan: TIC-TAC-TOE (TTT)

Perhatikan pohon permainan di bawah ini:



Contoh di atas adalah sepenggal permainan TTT. Setiap node merepresentasikan keadaan papan permainan, dan anak-anak dari node tersebut adalah langkah legal dari keadaan node parent. Untuk memberikan skor pada setiap node, akan diberikan nilai positif bagi pemain 1 (semakin positif berarti semakin menguntungkan). Demikian pula, nilai negatif akan menguntungkan bagi pemain 2.

Pada contoh, pemain 1 akan menjalankan X, pemain 2 memegang giliran O. Skor yang diberikan adalah +1 untuk kemenangan X, -1 untuk kemenangan O, dan nilai 0 jika seri. Perhatikan bahwa nilai yang berwarna biru pada gambar adalah nilai untuk satu keadaan saja (saat ini).

3.4 Algoritma MINMAX

Setelah diberikan pohon permainan, bagaimana sekarang caranya untuk mencari langkah yang optimal dari sebuah keadaan? Perlu diasumsikan bahwa lawan berpikir rasional (langkahnya logis), sesuai dengan aturan yang berlaku, dan memiliki keinginan untuk menang, sama halnya seperti yang sedang memiliki giliran melangkah. Sehingga terbentuk permainan yang sempurna (perfect game). Salah satu cara yang dapat dilakukan adalah dengan menggunakan algoritma MINMAX.

Setiap pemain akan berusaha mencapai skor yang diinginkan dari keadaan berikutnya yang terbaik, jadi pemain MAX akan mencari langkah positif, dan MIN mencari langkah negatif pada setiap gilirannya.

```
minimax(player,board)
    if(game over in current board position)
        return winner
    children = all legal moves for player from this board
    if(max's turn)
        return maximal score of calling minimax on all the children
    else (min's turn)
        return minimal score of calling minimax on all the children
```

Jika permainan terhenti pada posisi tertentu, maka tidak ada sesuatupun yang perlu dihitung, MINMAX akan mengembalikan nilai yang ada di atas papan permainan. Jika permainan belum berhenti / selesai, MINMAX akan mencari setiap posisi yang dimungkinkan dari node parent, dan mengevaluasi sesuai giliran pemain (MAX atau MIN), untuk mencari jalur yang paling menguntungkan.

Berapa lama waktu yang diperlukan algoritma ini?

Untuk permainan sederhana seperti TTT, tentu tidak akan terlalu lama, masih dimungkinkan untuk mengekspansi semua anak dari node parent. Tapi untuk permainan seperti Catur atau Othello, tentu waktu eksekusi akan menaik drastis. Yang perlu diperhatikan adalah bahwa dalam algoritma MINMAX, tidak perlu semua langkah dibuka sejak awal, algoritma hanya perlu mengantisipasi beberapa keadaan / langkah ke depan. Dan penilaian akan diberikan oleh fungsi evaluasi / utilitas tertentu. Fungsi utilitas ini terkadang menjadi kekuatan dari sebuah permainan komputer. Jika komputer menilai bahwa pemain berada dalam jalur yang tepat, maka akan ditemukan sebuah solusi untuk kemenangan, jika tidak maka pemain akan disarankan untuk mengubah jalur permainan, namun dalam permainan seperti ini, tidak ada backtracking, maka tugas pencarian berikutnya adalah hanya untuk mencoba memaksimalkan nilai langkah, meskipun bukan dalam jalur terbaik.