Aplikasi Algoritma *MiniMax* pada Beberapa Permainan Papan

Gaudensius Dimas Prasetyo Suprapto - 13514059
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13514059@std.stei.itb.ac.id

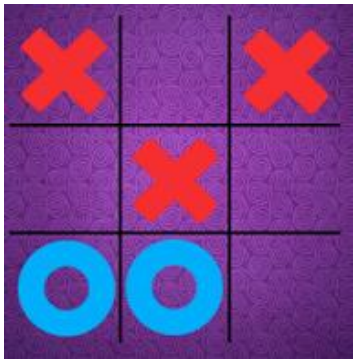
***Abstraksi*— Permainan papan (*board game*) adalah suatu permainan unik yang dibuat manusia yang tidak jarang membutuhkan strategi yang kompleks untuk memenangkannya. Permainan papan ini sudah banyak menjadi persoalan dalam pembuatan AI, seperti Catur, Reversi, dan lain-lain.**

***Kata Kunci*— Permainan papan, Algoritma *MiniMax*, *Greedy*.**

I. PENDAHULUAN

Permainan adalah suatu sarana yang manusia buat untuk menghibur dirinya. Sarana yang dimaksud di sini dapat berwujud macam-macam, bisa berupa nyata seperti bola sepak untuk bermain sepak bola, raket untuk bermain badminton, atau bahkan bisa tidak berbentuk nyata, misalnya peraturan untuk bermain petak umpet, bentengan, dan lain-lain. Yang akan kita bahas di sini adalah permainan dengan alat yang sudah biasa kita kenali, yaitu permainan papan.

Permainan papan sangatlah bermacam-macam jenisnya, ada yang modern, ada pula yang tradisional. Ukurannya pun bermacam-macam, dan tidak jarang ukuran menentukan kerumitan dalam memainkannya, sebagai contoh adalah permainan sederhana yang biasa kita mainkan saat di Sekolah Dasar dengan teman kita ketika guru menjelaskan, yaitu permainan XOX. Permainan XOX ini dapat dianggap sebuah permainan papan berukuran 3×3, di mana permainan ini sangatlah sederhana sehingga jika seseorang tahu "trik" untuk memainkannya, permainan XOX dengan orang tersebut akan berakhir antara seri atau orang tersebut menang.



Gambar 1. Permainan XOX
Source: http://media.androidappsgame.com/4/100941/

Permainan papan yang membutuhkan strategi yang rumit ini sudah menjadi banyak persoalan dalam pembuatan Inteligensi Buatan untuk berbagai permainan. Salah satu algoritma yang tidak jarang dipakai dalam pembuatan *AI* suatu permainan papan adalah Algoritma *MiniMax*, yang akan dijelaskan pada inti masalah dari makalah ini.

II. TEORI DASAR

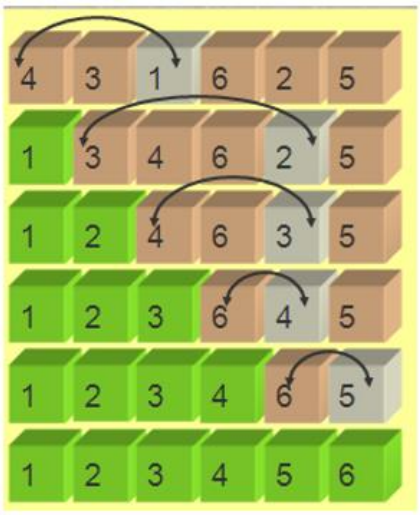
Sebelum kita masuk ke dalam Algoritma *MiniMax* itu sendiri ada baiknya kita mempelajari dahulu dasar-dasar dari penentuan algoritma untuk memecahkan masalah yang dasar :

A. Algoritma Brute Force

Algoritma *Brute Force* adalah algoritma yang menyelesaikan masalah dengan cara yang paling *straightforward*, yaitu dengan mencoba semua kemungkinan yang ada dari solusi suatu masalah.

Algoritma *Brute Force* ini mempunyai beberapa karakteristik :

1. Tidak efektif dan efisien, karena mempunyai kompleksitas yang paling kompleks dari antara semua algoritma yang ada, sehingga sering juga disebut sebagai Algoritma Naif.
2. Digunakan untuk persoalan yang cenderung kecil karena algoritma Brute Force sangat mudah diimplementasikan dan sederhana.
3. Sering digunakan sebagai pembanding dengan suatu algoritma yang lebih efisien.
4. Hampir semua persoalan dapat diselesaikan dengan Algoritma *Brute Force*.



Gambar 2. Algoritma *Selection Sort* yang merupakan *Brute Force*
Source : Referensi [1].

Dilihat dari karakteristiknya, dapat disimpulkan bahwa Algoritma *Brute Force* memiliki kelebihan :

1. Dapat dipakai untuk memecahkan jenis persoalan apapun.
2. Sangat sederhana sehingga mudah untuk dimengerti.
3. Algoritma ini dapat dinilai layak untuk pemecahan masalah yang kecil namun penting, seperti *search*, *sort*, dan lain-lain.
4. Menghasilkan algoritma baku untuk tugas-tugas komputasi.

Namun, Algoritma *Brute Force* ini mempunyai kelemahan yang cukup penting, yaitu:

1. Sangat jarang menghasilkan algoritma yang efektif dan efisien.
2. Cenderung terlalu lambat sehingga menjadi algoritma yang tak layak.
3. Tidak terstruktur dan kreatif seperti algoritma yang lainnya

B. Algoritma Greedy

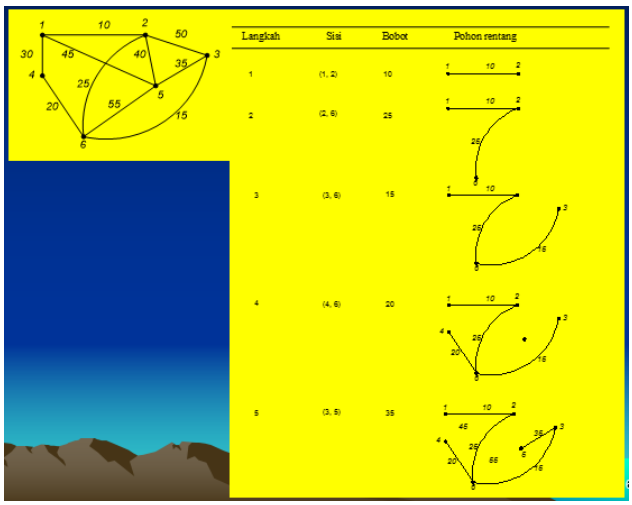
Algoritma *Greedy* ini merupakan algoritma yang paling cocok digunakan untuk masalah optimisasi, baik me-maksimal-kan suatu hasil ataupun me-minimal-kannya. Sesuai dengan namanya, Algoritma *Greedy* berprinsip untuk mengambil hasil yang terbaik pada saat itu. Algoritma ini membentuk suatu solusi langkah per langkah, di mana setiap langkahnya, selalu diambil langkah yang terbaik saat itu juga, dengan harapan hasil akhir dari langkah-langkah yang telah diambil merupakan hasil yang terbaik pula. Jika hasil dari Algoritma *Greedy* memang selalu hasil yang terbaik, hasil tersebut selalu dapat diturunkan secara matematis.

Namun, ada kalanya pula hasil akhir dari Algoritma *Greedy* ini tidak merupakan hasil yang terbaik. Walau begitu, algoritma ini tetap dipakai karena walau tidak optimal, hasil tersebut masih berguna sebagai nilai hampiran dari nilai optimal karena Algoritma *Greedy* cenderung lebih sederhana walau tidak eksak dibanding dengan algoritma rumit yang menghasilkan hasil yang eksak.

Elemen-elemen yang terdapat pada Algoritma *Greedy* meliputi:

1. Himpunan Kandidat (*C*)
2. Himpunan Solusi (*S*)
3. Fungsi Seleksi
4. Fungsi Kelayakan
5. Fungsi Objektif

Melihat elemen-elemen yang terdapat pada Algoritma *Greedy* ini, dapat dikatakan bahwa Algoritma *Greedy* adalah algoritma untuk mencari himpunan solusi *S* dari himpunan kandidat *C* dengan cara mengaplikasikan fungsi seleksi pada *C*. Dalam hal ini, semua anggota dari *S* harus *feasible*, ditentukan oleh fungsi kelayakan, dan kemudian *S* dioptimisasi menggunakan fungsi objektif.

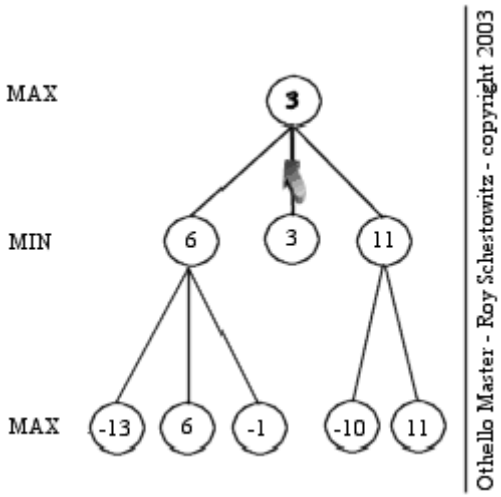


Gambar 3. Algoritma *Prim* yang merupakan Algoritma *Greedy*
Source : Referensi [1].

III. ALGORITMA MINIMAX

Nama Algoritma *MiniMax* merupakan perpaduan dari dua kata yang sebenarnya mudah ditebak, yaitu Minimum dan Maksimum.

Algoritma ini termasuk ke dalam Algoritma *Greedy* karena merupakan algoritma yang pada dasarnya digunakan untuk optimisasi. Optimisasi yang dilakukan oleh algoritma ini mencari kedua nilai optimum untuk mencari langkah terbaik yang memungkinkan. Algoritma ini dijalankan pada suatu pohon prediksi kemungkinan.



Gambar 4. Pohon prediksi kemungkinan dari Algoritma *MiniMax*
Source : http://othellomaster.com/OM/Report/HTML/

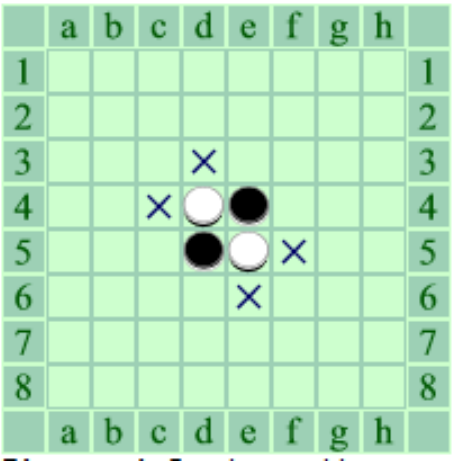
Pohon prediksi kemungkinan ini dibuat untuk menentukan langkah mana yang harus diambil untuk mencapai daun terakhir yang optimum. Sebelum memulai Algoritma *MiniMax* harus mengetahui terlebih dahulu semua kemungkinan yang ada pada permainan tersebut dan dipetakan menjadi pohon prediksi kemungkinan. Namun, mustahil bagi memori komputer untuk menyimpan semua kemungkinan yang ada pada permainan tersebut. Oleh karena itu, pemetaan pohon kemungkinan dibatasi pada level tertentu untuk mencegah penuhnya memori komputer, sehingga komputer seolah-olah dapat memprediksi gerakan lawan ke depannya sebanyak *n* giliran, di mana *n* merupakan batas level dari pemetaan pohon kemungkinan.

Algoritma *MiniMax* akan mencari langkah untuk memperoleh hasil maksimum pada daun di pohon kemungkinan di saat gilirannya, dan akan mencari langkah untuk memperoleh hasil minimum pada daun di pohon kemungkinan di saat giliran musuh. Oleh karena itu, Algoritma *MiniMax* ini hanya bisa diaplikasikan pada permainan yang dimainkan oleh dua orang saja.

IV. APLIKASI ALGORITMA MINIMAX

A. Reversi

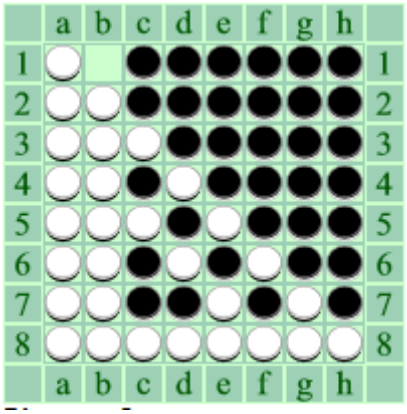
1. Aturan



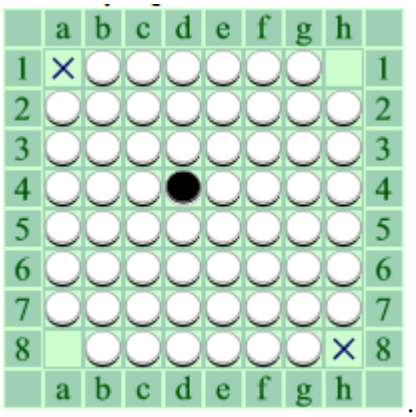
Gambar 5. Posisi awal dari Reversi
Source : Ref. [2]

Seperti yang dikatakan sebelumnya, permainan Reversi ini membutuhkan dua pemain, seperti catur, pemain dengan warna putih melawan pemain dengan warna hitam. Perbedaannya dari catur adalah di permainan Reversi ini, pemain dengan warna hitam memulai permainan dahulu.

Setiap giliran, pemain harus menaruh koin dengan warnanya menghadap ke atas jika tersedia langkah yang legal baginya. Yang dimaksud dengan langkah legal di sini adalah pemain harus mengubah koin di papan dengan warna musuhnya menjadi warnanya sendiri dengan cara menaruh koin dengan warnanya sendiri di papan sedemikian rupa sehingga ada koin warna lawan yang "terkepung" baik secara vertikal, diagonal, maupun vertikal sekalipun. Koin yang terkepung inilah yang nantinya akan dibalik menjadi warnanya sendiri. Pada gambar di atas, dapat dilihat bahwa tanda silang pada gambar tersebut merupakan langkah legal yang dapat dilakukan oleh si hitam. Pemain tidak dapat melewat gilirannya jika pemain tersebut masih punya langkah yang legal. Permainan akan berakhir ketika sudah tidak ada pemain yang mempunyai langkah legal, dengan pemenangnya adalah pemain dengan koin warnanya sendiri paling banyak di papan.



Gambar 6. Permainan berakhir karena tidak ada legal move lagi
Source : Ref.[2]



Gambar 7. Pada saat keadaan seperti ini, hitam dapat memenangkan permainan dengan poin 40 vs 24
Source : Ref.[2]

2. Aplikasi Algoritma *MiniMax*

Pada permainan Reversi ini, algoritma *MiniMax* ini dapat diterapkan pada dua hal, yaitu *MiniMax* menurut banyak poin, dan *MiniMax* menurut *margin* dari posisi yang diambil.

Penerapan *MiniMax* menurut banyak poin ini berprinsip untuk memaksimalkan poin sendiri dan meminimalkan poin lawan pada saat langkah itu diambil, sehingga hasil akhir yang diharapkan adalah koin dengan warna sendiri akhirnya lebih banyak dan menang..

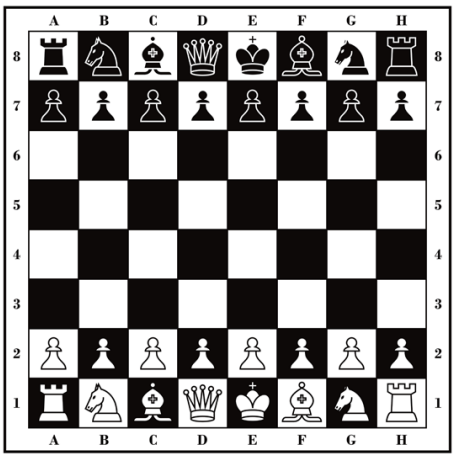
Namun, untuk memori komputer yang terbatas, cara itu tidaklah efektif melawan seorang lawan dengan teknik evaporasi yang baik. Yang dimaksud dengan teknik evaporasi di sini adalah teknik dalam bermain reversi, di mana pemain akan mengambil langkah legal yang mengambil sedikit dari koin lawan setiap langkahnya dengan hati-hati (jangan sampai tiba-tiba terhapus dari papan seluruhnya). Teknik ini bertujuan untuk meminimalkan langkah legal yang dapat diambil oleh lawan, dan nantinya saat terakhir, pemain yang bermain dengan teknik ini dapat mengambil lebih banyak koin lawan karena mayoritas koin di papan yang hampir penuh tersebut diisi oleh koin lawan, sehingga pada algoritma *MiniMax*, lawan seolah-olah terpojok, padahal sebenarnya itu merupakan taktik bermain dari sang lawan.

Oleh karena itu, algoritma *MiniMax* ini lebih cocok diterapkan menurut margin posisi, yaitu dengan menilai setiap grid pada papan reversi dengan suatu margin. Margin posisi dari grid yang dekat sudut akan bernilai rendah karena merupakan posisi yang tidak menguntungkan, sedangkan margin posisi dari grid sudut akan memiliki margin lebih besar dari yang lain karena merupakan kunci kemenangan dari reversi.

Gambar 8. Contoh margin posisi
Source : Ref.[2]

B. Catur

1. Aturan



Gambar 9. Posisi awal dari Catur
Source : http://www.chess-space.com/

Catur adalah permainan papan 8×8 yang sudah diakui dunia sebagai salah satu dari permainan olah raga otak (olah batin). Catur membutuhkan dua orang pemain, di mana pemain yang satu memegang sisi putih, dan pemain yang lain memegang sisi hitam. Dalam catur, pemain yang memegang sisi putih mendapatkan giliran pertama.

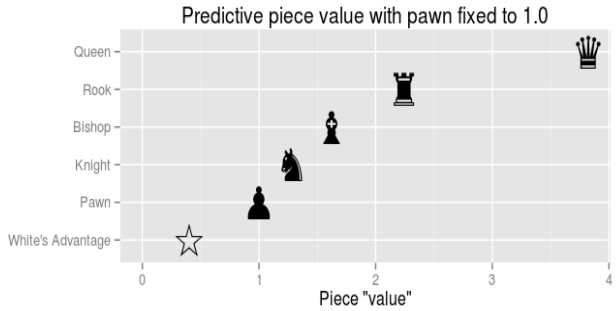
Permainan catur ini merupakan simulasi dari strategi peperangan, di mana masing masing pion berperan sesuai kemampuannya masing-masing.

Prajurit (*pawn*) adalah pion yang paling banyak dan paling lemah. Prajurit hanya bisa berjalan maju jika tidak ada yang menghalangi dan dapat memakan pion yang berada di diagonal depannya. Namun, jika belum digerakkan dan tidak ada yang menghalangi, prajurit dapat langsung maju dua langkah, dan jika prajurit berhasil mencapai ujung depan arena, prajurit dapat di-*promote* menjadi pion pilihan kita. Benteng (*rook*) merupakan salah satu pion kuat yang dapat bergerak secara horizontal dan vertikal dan dapat memakan pion yang menghalanginya. Menteri (*bishop*) mempunyai gerak yang serupa dengan benteng, namun geraknya berarah diagonal. Kuda (*knight*) merupakan pion yang unik karena gerakannya yang tidak terduga, yaitu berbentuk L (dua petak lurus, satu petak belok). Ratu (*queen*) merupakan pion terkuat yang gerakan perpaduan antara benteng dan menteri. Raja (*king*) adalah pion yang harus kita lindungi dan dapat bergerak satu petak ke segala arah.

Aturan dari catur ini cukup rumit, namun tujuan utama dari permainan ini adalah mengepung raja sedemikian rupa sehingga raja tidak dapat bergerak lagi.

2. Aplikasi Algoritma *MiniMax*

Ide dari aplikasi algoritma ini pada permainan catur adalah memberi nilai pada masing-masing pion sesuai dengan kekuatan-kekuatannya. Algoritma ini akan mencari langkah di mana total dari nilai-nilai semua pionnya tetap paling besar (bertahan) dan di saat yang sama, mencari langkah yang membuat total nilai-nilai semua pion lawan menjadi paling kecil (menyerang).



Gambar 10. Nilai pion pada catur (relatif)
Source : http://www.sumsar.net/figures/2015-06-10-big-data-and-chess/

V. APENDIKS

1) Algoritma : proses dan aturan yang dijalankan dalam suatu perhitungan atau operasi pemecahan masalah lainnya.
2) Optimum : nilai maksimum dan nilai minimum
3) Optimisasi : nencarian nilai optimum
4) Pohon : graf tanpa sirkuit

VI. UCAPAN TERIMA KASIH

Kepada Pak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi selaku Dosen IF 2211 – Strategi Algoritma, saya ucapkan terima kasih karena telah memberikan saya tugas makalah ini untuk memperluas wawasan saya mengenai ilmu-ilmu yang berkaitan dengan Strategi Algoritma ini.

Tidak lupa saya ucapkan terima kasih pada teman-teman dan keluarga serta Tuhan Yang Maha Esa karena dukungan merekalah saya dapat mengerjakan tugas makalah ini dengan baik.

REFERENSI

[1] Munir, Rinaldi, Slide Kuliah IF 2211 Strategi Algoritma .

[2] http://www.samsoft.org.uk/reversi/strategy.htm, 20:05, 8 Mei 2016

dan sumber-sumber gambar yang dicantumkan langsung sumbernya.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2015

Gaudensius Dimas Prasetyo Suprapto - 13514059