Penerapan Algoritma *Dynamic Programming* Pada Catur

Muhammad Sohibul Maromi / 13510061
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510061@std.stei.itb.ac.id

***Abstract*—The abstract is to be in fully-justified italicized text, at the top of the left-hand column as it is here, below the author information. The abstract is to be in 9-point, single-spaced type, and may be up to 8 cm long. Define all symbols used in the abstract. Do not cite references in the abstract. Do not delete the blank line immediately above the abstract; it sets the footnote at the bottom of this column. Leave two blank lines after the index terms, then begin the main text. All manuscripts must be in English.**

***Index Terms*—About four key words or phrases in alphabetical order, separated by commas.**

I. PENDAHULUAN

Catur sering disebut sebagai "*perfect information game*", karena kedua pemain dapat mengetahui seluruh kondisi sepanjang waktu permainan. Hanya dengan melihat papan catur, pemain dapat mengetahui bidak apa saja yang hidup dan posisi-posisinya. Tic-tac-toe, Backgammon dan Othello juga merupakan jenis *perfect information game*, tetapi tidak dengan permainan kartu domino (para pemain tidak dapat mengetahui kartu yang dipegang oleh pemain lain).

Gambar 1. Permainan catur vs computer
Sumber: http://www.thechesswebsite.com/

Selain manusia, komputer juga dapat bermain catur walau hanya dengan waktu terbatas. Namun untuk waktu yang lama (hingga ber jam-jam) komputer tidak mampu untuk bersaing dalam permainan yang rumit dengan master catur. Bermula dari Torres y Quevedo pada tahun 1914 yang menciptakan catur otomatis (komputer) pertama yang dapat memainkan raja dan benteng lawan raja. Kemudian pada tahun 1950 Claude Shannon melontarkan ide untuk dapat menciptakan program catur yang dapat bermain sampai selesai. Namu saat itu tidak ada perangkat keras yang cukup cepat untuk menjalankan program tersebut. Pada tahun 1989 IBM membuat sebuah komputer catur, Deep Tought, untuk melawan seorang master catur dunia, Gary Kasparov, namun kalah pada kedua permainan. Riset terus berjalan dan kira-kira 10 tahun kemudian, IBM berhasil menciptakan computer pertama yang dapat mengalahkan juara dunia catur saat itu yang dinamakan Deep Blue.

Pada makalah ini penulis tidak berusaha membuat sebuah program catur yang akan melawan master catur seperti Kasparov. Tetapi penulis hanya akan mengimplementasi algoritma "Dynamic Programming" untuk melakukan pencarian langkah terbaik pada permainan catur.

II. ANALISIS PEMROGRAMAN CATUR

Untuk membuat sebuah program computer yang dapat bermain catur, ada beberapa komponen *software* yang harus dipenuhi, yaitu:

- Sebuah cara untuk merepresentasikan papan catur pada memory, sehingga dapat program dapat mengetahui setiap kondisi permainan.
- Aturan yang menentukan suatu langkah yang di-*generate* adalah legal.
- Teknik untuk memilih langkah dari seluruh kemungkinan langkah yang ada.
- Cara untuk memberikan penilaian terhadap kondisi tertentu, sehingga dapat ditentukan pemain yang sedang unggul.

Untuk memenuhi poin-poin di atas, penulis akan membagi analisis sebagai berikut.

A. Representasi Papan Catur

Pada tahun 60-an tim KAISSA dari Uni Soviet telah mengembangkan trik untuk permainan dengan menggunakan papan 64-kotak, yaitu **bitboard**. KAISSA menjalankannya menggunakan processor 64-bit. Jadi, 64 merupakan jumlah kotak pada papan catur, sehingga memungkinkan satu memory untuk merepresentasikan

predikat *true* atau *false* pada keseluruhan papan. Contoh, satu bitboard mungkin mengandung jawaban dari "apakah ada bidak putih di sini?" untuk setiap kotak pada papan catur. Oleh karena itu, setiap kondisi pada permainan catur dapat direpresentasikan oleh 12 bitboard, yaitu:

- raja putih
- ratu putih
- gajah putih
- kuda putih
- benteng putih
- raja hitam
- ratu hitam
- gajah hitam
- kuda hitam
- benteng hitam

dan dua bitboard untuk "semua bidak putih" dan "semua bidak hitam" yang digunakan untuk mempercepat perhitungan yang lebih lanjut.

Dengan menggunakan bitboard, banyak operasi-operasi penting yang dapat dijalankan dengan operator-operator logika dalam satu siklus set instruksi *processor*. Contohnya, untuk melakukan verifikasi apakah ratu putih men-skak raja hitam. Dengan representasi array 2D sederhana perlu:

- Menemukan posisi ratu, harus melakukan pencarian linear dan mungkin membutuhkan 64 kali *looping*.
- Memeriksa semua kemungkinan ratu dapat bergerak, ke 8 arah, sampai menemukan raja.

Proses ini selalu memebutuhkan waktu yang cukup banyak, terlebih jika ratu berada pada akhir array atau bahkan tidak ada skak yang merupakan hampir semua kasus.

Untuk memecahkan kasus tersebut dengan bitboard, yang perlu dilakukan adalah:

- Ambil bitboard dari "posisi ratu putih"
- Gunakan untuk melacak pada database bitboard yang merepresentasikan kotak-kotak yang dapat serang (dilalui) oleh ratu.
- Operasi AND bitboard tadi dengan bitboard dari "posisi raja hitam".

Jika hasilnya tidak sama dengan nol maka ratu putih men-skak raja hitam.

Contoh lain yaitu, jika hendak menggerakkan kuda putih maka carilah bitboard dari posisi yang dapat diserang oleh kuda kemudian lakukanlah operasi AND dengan NOT dari bitboard yang merepresentasikan semua kotak yang diduduki oleh bidak putih.

B. Generate Langkah

Pada tahun 1949, Claude Shannon menjelaskan dua cara untuk membuat algoritma bermain catur:

- Periksa semua kemungkinan langkah dan semua kemungkinan responnya secara rekursif
- Hanya periksa langkah "terbaik" dan hanya respon dari langkah "terbaik" tadi, secara rekursif juga.

Awalnya, cara kedua tampak akan sukses. Bagaimanapun, ini merupakan cara manusia bermain catur, dan secara pintas memeriksa beberapa langkah akan menghemat waktu dari pada memeriksa semuanya. Sayang sekali, hasilnya tidak membutikan teori. Program dengan selektif tadi tidak bermain dengan baik. Capaian terbaik hanya pada klub menengah kebawah, sering melakukan kesalahan-kesalahan yang memalukan. Mengalahkan juara dunia hanyalah angan-angan.

Masalahnya adalah bahwa, untuk "generator langkah terbaik" untuk menjadi baik, itu harus mendekati sempurna. Misalkan program ini dilengkapi dengan fungsi yang melihat untuk 5 langkah terbaik dan langkah terbaik tadi setidaknya 95%. Ini berarti probabilitas bahwa list generator akan selalu menghasilkan pilihan terbaik setiap saat, selama 40-langkah pertandingan, kurang dari 13%. Bahkan generator seperti dewa dengan akurasi 99% akan blunder setidaknya sekali dalam sekitar sepertiga dari permainan, sementara fungsi 90%-akurat lebih masuk akal akan memainkan seluruh permainan tanpa mengacaukan kurang dari 1,5% dari keseluruhan waktu!

Algoritma generator yang sebelumnya tidak digunakan lagi. Sehingga cara dengan full-width searching menjadi pilihan. Cara mengimplementasikannya yaitu:

- Mencari semua kemungkinan langkah yang legal
- Mengurutkannya dengan cara tertentu, berharap dapat meningkatkan kecepatan proses pencarian
- Cari semua kemungkinan

Bermula dari sebuah program yang bernama Sargon, melakukan ini dengan memindai papan satu kotak dengan waktu tertentu, mencari bidak yang bergerak, dan menghitung kemungkinan langkah dengan cepat. Hal ini sangat menghabiskan banyak memori.

Sekarang, struktur data dengan representasi papan catur yang baik dapat menghindari sejumlah besar perhitungan dan kompleksitas kode, dengan cost puluhan Kbytes. Ketika generator langkah yang super cepat ini dikombinasikan dengan tabel transposisi, pencarian langkah terbaik akan sangat mulus.

C. Mencari Langkah Terbaik

D. Evaluasi Kondisi

III. PSEUDO CODE

REFERENCES

[1] http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-i-getting-started-r1014
[2] http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-ii-data-structures-r1046
[3] http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-iii-move-generation-r1126
[4] http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-iv-basic-search-r1171
[5] http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-v-advanced-search-r1197
[6] http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/chess-programming-part-vi-evaluation-functions-r1208
[7] http://chessprogramming.wikispaces.com/
[8] http://www.fam-petzke.de/chess_home_en.shtml

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, 29 April 2010

Muhammad Sohibul Maromi
13510061