Penerapan Struktur Pohon dan Pencarian Solusi Langkah pada *Engine* Catur

Tony / 13512018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512018@std.stei.itb.ac.id

***Abstrak*—Pohon adalah sebuah penerapan lanjutan dari graf. Penerapan struktur pohon banyak sekali ditemukan pada berbagai macam hal seperti misalnya dalam pembuatan *engine* catur. Pohon yang diterapkan disini adalah pohon n-ary yang berbasis pada pencarian solusi(node) pada pohon. Makalah ini bertujuan untuk menganalisis penerapan struktur pohon yang digunakan sebagai representasi solusi-solusi langkah sebuah *engine* catur. Berdasarkan solusi-solusi yang ada, dipilihlah solusi terbaik berdasarkan statistik posisi dan keunggulan setelah langkah tersebut diambil.**

Kata Kunci*—*engine* catur, IDDFS, pohon n-ary, strukturisasi pohon, *minimax searching

I. PENDAHULUAN

Catur adalah permainan yang cukup terkenal di seluruh dunia dari berbagai kalangan. Permainan ini dimainkan oleh 2 orang: satu orang memegang buah putih dan satunya lagi memegang buah hitam. Di Indonesia, permainan berbasis papan ukuran 8x8(a1-h8) ini mempunyai penggemarnya tersendiri. Catur pun berkembang dari waktu ke waktu dengan segala jenis variasi taktiknya yaitu *opening*, *middle-game*, dan *ending*.

Gambar 1 : Ilustrasi Papan Catur
(Sumber Gambar : http://en.wikipedia.org/wiki/File:Chess_board_opening_staunton.jpg)

Taktik dan strategi dalam permainan catur mulai berkembang pesat ketika pertama kali ditemukannya *engine* catur pertama di dunia yang diciptakan oleh seorang bernama Alan Turing. Semenjak itu, terbukalah cakrawala baru tentang taktik-taktik yang sebelumnya dianggap kurang bagus menjadi taktik yang patut diperhitungkan, seperti contohnya pembukaan pertahanan Sisilia yang dulunya bahkan dianggap buruk, namun sekarang menjadi pembukaan terbaik untuk pemegang buah hitam dalam menghadapi pembukaan putih e4. Semua ini karena perkembangan kemajuan teknologi kalkulasi komputer yang semakin canggih yang membuat kecepatan perhitungan meningkat dan banyaknya kombinasi langkah menjadi semakin bervariasi.

Era baru dalam keberhasilan teknologi *engine* catur ini dibuktikan pertama kali oleh kemenangan *engine* buatan IBM yaitu DeepBlue yang berhasil menang melawan seorang juara dunia catur saat itu, Garry Kasparov pada 10 Februari 1996. Setelah itu, perkembangan *engine* catur pun semakin pesat seiring dengan pesatnya perkembangan teknologi komputer yang sekarang sudah menembus pada generasi *quad-core* ini. Generasi *engine* yang sekarang menjadi *engine* catur terkuat di dunia, yaitu Houdini 4.0 64-bit 4CPU menjadi yang tak terkalahkan sampai sekarang dengan rating mencapai 3266. Kecepatan kalkulasi yang dapat menembus ratusan juta atau bahkan miliaran langkah per detik yang dimiliki *engine* catur ini membuat manusia sekelas GrandMaster sekalipun yang 'hanya' dapat mengkalkulasi 2 langkah per detik dengan bantuan 100 milyar saraf otaknya tak berdaya.

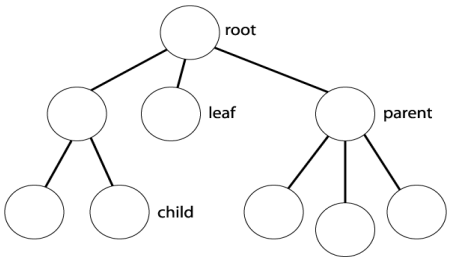
II. TEORI DASAR POHON

2.1. *Pohon*

Pohon dalah sebuah graf tidak berarah yang terhubung dan tidak mengandung sirkuit. Pohon yang tidak mengandung sirkuit adalah pohon yang tidak mengandung sisi ganda ataupun kalang (*loop*). Oleh sebab itu, pohon bisa juga disebut sebagai sebuah graf sederhana. Pohon mengacu pada teori graf sehingga sebuah pohon dapat juga hanya memiliki sebuah simpul tanpa sebuah sisipun. Dengan kata lain, jika G = (*V*, *E*) adalah sebuah pohon, maka *V* yang merupakan nodesnya tidak boleh merupakan himpunan kosong, namun *E* yang sisi dari sebuah pohon dalam hal ini boleh kosong. Sifat-sifat pohon bila *G* = (*V*, *E*) adalah graf tak-berarah sederhana dan jumlah simpulnya *n* :

1.) Setiap pasang simpul pada G terhubung dengan hanya

tepat satu lintasan.
2.) G terhubung dan memiliki *n*-1 buah sisi.
3.) G tidak mengandung sebuah sirkuit atau lintasan tertutup.
4.) Penambahan satu sisi saja pada graf di node manapun akan membuat hanya satu sirkuit.
5.) G terhubung dan semua sisinya bersifat sebagai jembatan yang bila dihapus akan menyebabkan graf terbagi menjadi 2 komponen graf.

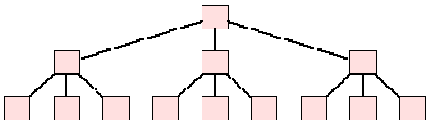


Gambar 2 : Ilustrasi struktur pohon
(Sumber gambar :
http://stackoverflow.com/questions/19330731/tree-implementation-in-java-root-parents-and-children)

Salah satu jenis pohon yang paling umum dan general adalah pohon biner, yaitu pohon yang hanya memiliki maksimum 2 buah anak. Pada makalah ini, penulis akan lebih membahas pohon n-ary dan pencarian node pohon dengan algoritma *Minimax* (*The Basic Search Algorithm*).

2.2. *Pohon N-Ary*

Pohon *n-ary* (baca: ener) adalah sebuah pohon berakar yang setiap simpul cabangnya mempunyai maksimal *n* buah anak. Jika n = 2, maka pohon tersebut disebut sebagai pohon biner. Sebuah pohon n-ary dikatakan teratur atau penuh jika setiap simpul cabangnya mempunyai tepat *n* buah anak.



Gambar 3 : Ilustrasi pohon *n-ary* teratur dengan *n* = 3
(Sumber gambar :
http://homepages.ius.edu/rwisman/C251/html/ch10/10_1_23.gif)

Pada penerapannya, pohon *n-ary* banyak digunakan pada berbagai ilmu dan kehidupan sehari-hari termasuk pada penggunaan struktur pohon *n-ary* sebagai representasi kemungkinan solusi langkah pada *engine* catur yang menjadi pokok bahasan dalam makalah ini. Pada representasi solusi langkah tersebut, simpul-simpul berperan sebagai posisi-posisi pada papan catur dan cabangnya berperan sebagai langkah yang diperbolehkan untuk dijalankan (*legal chess moves*). Simpul daun berperan sebagai posisi akhir yang tercipta setelah langkah tersebut dijalankan.

2.3. *Metode Pencarian Mendalam Berulang / Iterative deepening depth-first search (IDDFS)*

Algoritma dengan metode Pencarian Mendalam Berulang (*IDDFS*) adalah metode pencarian dengan menggunakan status ruang dimana pencarian dilakukan secara mendalam ke arah bawah dengan pencarian dilakukan secara iteratif sambil membentuk sub-sub pohon di bawahnya secara dinamis. Kedalaman pertama yang paling dangkal yang merupakan *goal state* dari sebuah persoalan pohon yang akan diambil sebagai solusi. Metode pencarian *IDDFS* ini mengkombinasikan antara metode urutan pencarian *DFS* untuk hal efisiensi ruang(memori) dan urutan kumulatif *BFS* dalam hal efektivitas prioritas pencarian pertama dalam sebuah kedalaman.

Kompleksitas ruang dari metode *IDDFS* ini adalah O(bd) dan kompleksitas waktu mencapai $O(b^d)$ dimana b adalah banyaknya cabang dan d adalah kedalaman dari solusi yang paling dangkal. Pendekatan secara iteratif dengan mengunjungi berkali-kali ini terlihat kurang efektif. Namun, kunjungan simpul-simpul di kedalaman yang dangkal tidaklah bermasalah karena kebanyakan simpul selalu berada pada kedalaman yang dalam, seperti daun.

Keuntungan dari metode *IDDFS* ini adalah bahwa estimasi pencarian solusi dapat lebih akurat dan pencarian menjadi lebih cepat karena dilakukan dari urutan pencarian yang lebih baik berdasarkan tingkat prioritas. Sebagai contohnya adalah alpha-beta pruning yang mencari langkah terbaik pertama kali sebagai prioritas. Selain itu, keuntungan yang lainnya adalah bahwa metode *IDDFS* ini metode yang cukup responsif. Contohnya untuk nilai kedalaman *d* yang kecil, metode ini mengkalkulasi dengan sangat cepat. Dengan nilai d yang terus bertambah dalam pencarian seiring langkah terbaik yang ditemukan. Hal ini cocok untuk algoritma pencarian solusi dalam *engine* catur.

Metode IDDFS ini menelusuri sebuah pohon secara interatif dengan mengikuti persamaan berikut :

$$(d+1)1+(d)b+(d-1)b^2+\cdots+3b^{d-2}+2b^{d-1}+b^d$$

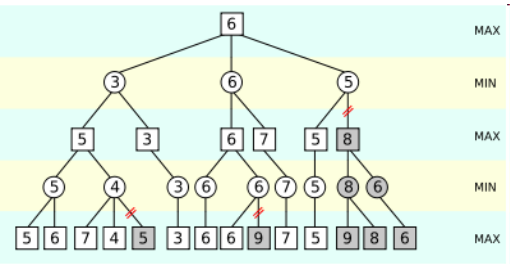
$$\sum_{i=0}^{d}(d+1-i)b^i$$

III. PENERAPAN POHON PADA *ENGINE* CATUR

3.1. *Kompleksitas Stukturisasi Pohon pada Engine Catur*

Pohon yang terbentuk dari solusi-solusi langkah yang terjadi di papan catur tentunya sangat banyak sekali kombinasinya. Untuk langkah pertama saja, ada 20 kombinasi langkah untuk buah putih dan 20 langkah jawaban dari buah hitam. Jadi, untuk langkah pertama saja, kita sudah mendapatkan 400 buah kemungkinan

terjadinya sebuah langkah. Untuk langkah selanjutnya pun semua kemungkinan semakin bertambah secara eksponensial sehingga menurut kalkulasi dapat mencapai angka 10^{120} total kemungkinan permainan catur di papan 8x8 tersebut. Tentu saja, untuk memudahkan dalam strukturisasi kemungkinan langkah yang begitu banyaknya, dibutuhkan sebuah representasi dimana pohon menjadi representasi dalam hal ini pada *engine* catur dengan simpul-simpulnya yang menjadi setiap kemngkinan langkah putih dan hitam berselang seling tiap kedalaman. Daun menjadi jawaban akhir dari kondisi posisi permainan di papan catur tersebut.



Gambar 4 : Ilustrasi Pohon pada *Engine* Catur
(Sumber gambar : http://www.uio.no/studier/emner/matnat/ifi/INF4130/h12/undervisningsmateriale/chess-algorithms-theory-and-practice-ver2012.pdf)

Cabang pada struktur pohon di *engine* catur merepresentasikan langkah legal dalam sebuah permainan catur. Langkah yang tidak legal direpresentasikan dengan tanda cabang yang dicoret dengan warna merah seperti yang terlihat pada ilustrasi Gambar 4.

3.2. *Pencarian pada Pohon dan Evaluasi Posisi*

Metode pencarian pada pohon tersebut berbasis pada nilai-nilai yang ada di setiap simpul yang ada di pohon solusi langkah pada *engine* catur. Daun-daun yang merepresentasikan posisi akhir dari sebuah pencarian solusi yang diinginkan tersebut dievaluasi apakah sudah terjadi posisi menang (skak-mat) atau belum. Jika belum terjadi posisi kemenangan, maka akan dihitung keuntungan material yang dipunya maupun keuntungan posisi dengan menghitung akumulasi nilai dari setiap simpul yang merupakan nilai /bobot dari setiap langkah yang dijalankan untuk mencapai titik posisi di daun.

Selain itu, algoritma yang dilakukan dalam pencarian solusi langkah ini menggunakan teknik *Minimax* dengan pertama mengasumsikan bahwa setiap langkah putih dan hitam adalah langkah terbaik. Lalu, diterapkanlah prinsip pencarian dengan *DFS* untuk mencari solusi secara mendalam ke bawah terlebih dahulu dengan memperhatikan prioritas nilai yang ada pada setiap simpul yang dilewati pada kedalaman tertentu. Pada penelusuran pencarian ini, bila putih yang melangkah, maka akan diambil poin terbaik untuk setiap langkah putih dan akan diambil poin terburuk untuk setiap langkah hitam. Hal ini bila *engine* tersebut memainkan buah putih sehingga membuat langkah yang akan 'menguntungkan' pihak putih dan merugikan pihak hitam. Ilustrasinya masih tergambarkan pada ilustrasi di Gambar 4 dimana pencarian selang seling Max Min di setiap kedalaman pencarian. Sebaliknya, jika *engine* tersebut berperan menjalankan buah hitam, maka akan dicari solusi dengan nilai terbaik untuk hitam dan nilai terburuk untuk putih sehingga membuat hitam menang.

Teknik Pruning dengan kompleksitas waktu pencarian $O(b^d)$ pada pohon ini juga membutuhkan waktu yang tidak sedikit dengan jumlah faktor per cabang dari setiap simpul dapat mencapai kurang lebih 40 langkah legal di setiap kemungkinan langkahnya(40 cabang per simpul).

Perlu diingat juga bahwa setiap buah catur memiliki nilai-nilai tersendiri baik itu berdasarkan nilai materialnya maupun nilai posisi dimana buah tersebut berada pada papan catur. Misalnya, nilai sebuah kuda di petak tengah pada papan catur membuat nilai posisinya meningkat sehingga nilai kemungkinan langkah kuda ke petak tengah sebagai solusi pada pencarian pohon tersebut meningkat pula. Sebaliknya, nilai material yang lebih tinggi akan berkurang nilai kemungkinannya menjadi solusi jika buah tersebut berada pada tempat yang buruk secara posisi. Misalnya, kuda yang berada di pinggir tentu hanya akan 'menjaga' petak dengan lingkup area yang kecil saja.

Gambar 5 : Ilustrasi posisi kuda yang bagus
(Sumber gambar : http://www.thechessdrum.net/chessacademy/diagrams/Board_knight3.jpg)

Pada ilustrasi Gambar 5, terlihat bahwa kuda putih pada petak D6 bekerja aktif di petak tengah di wilayah musuh dengan menjaga 8 petak, yang 4 petaknya merupakan wilayah pertahanan buah hitam. Hal ini tentunya akan menjadi evaluasi dalam bobot dalam simpul solusi langkah yang membuat posisi perhitungan keadaan papan oleh kuda putih menjadi lebih superior, menambah persentase keunggulan posisi putih di papan.

3.3. *Analisa Langkah Terbaik Pertama*

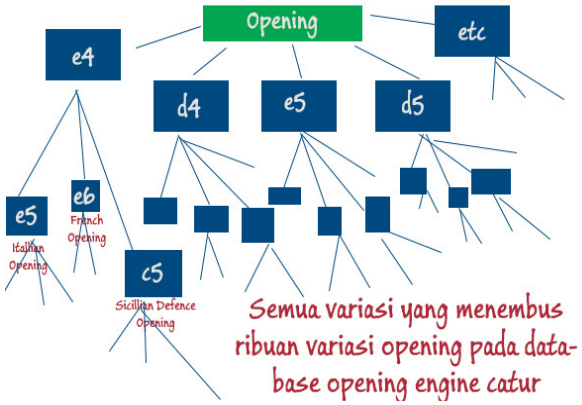
Analisa ini menggunakan teknik bernama *Alpha-Beta Pruning*. Pertama-tama dimulai dari nilai *alpha* yaitu nilai skor dari buah putih yang dimulai dari nilai minus tak hingga (*start value*: -∞) dan nilai *beta* yang dimulai dari

nilai plus tak hingga (*start value*: +∞). Jika nilai beta kurang dari alpha, maka posisi tidak pernah terjadi bila diasumsikan dalam permainan terbaik(tidak ada kesalahan langkah). Maka dalam hal ini, kita dapat mengabaikan langkah-langkah yang tidak penting yang tidak berpengaruh pada evaluasi skor akhir posisi permainan. Dengan *Alpha-Beta Pruning,* bahkan bila kita memulai dengan langkah terburuk di awal, maka kita mendapatkan kompleksitas perhitungan waktu sebesar O(b*b*b*...*b) = $O(b^d)$ dengan b sebanyak d kali(kedalaman). Jika kita memulai langkah awal dengan langkah terbaik, maka balasannya hanya ada satu langkah terbaik yang pasti sehingga kompleksitas perhitungan waktunya menjadi O(b*1*b*1*b*1...) = $O(b^{d/2})$. Jadi, kita dapat melipatgandakan kedalaman dari sebuah pohon solusi langkah tanpa menambah kompleksitas yang dibutuhkan bila kita memulainya dengan langkah yang terbaik pada langkah awal.

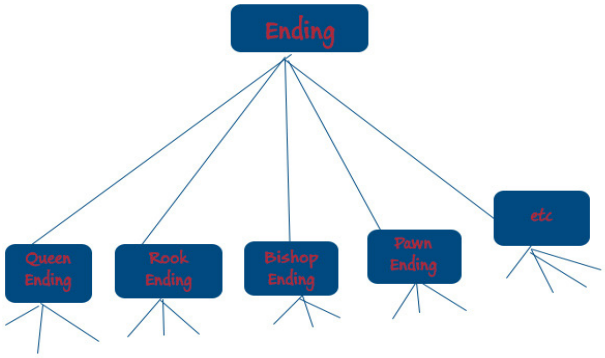
3.4. *Penelusuran berdasarkan data empiris permainan*

Seperti yang kita tahu, bahwa selain penilaian secara matematis berdasarkan kualitas material dan posisi yang menjadi patokan untuk pencarian solusi langkah, juga dibutuhkan sumber lain untuk pencarian yang lebih cepat dan baik dengan kepastian atau akurasi yang tinggi.

Penelusuran tersebut berdasarkan data empiris permainan-permainan catur di seluruh dunia yang masuk ke database *engine* catur tersebut. Biasanya, untuk opening dan ending yang biasanya merujuk pada database yang pasti karena langkah-langkah dalam opening itu pasti dan jumlahnya 'hanya' ribuan dengan perbandingan jumlah kemungkinan langkah yang ada pada middle game yang mencapai ratusan juta kemungkinan. Selain itu, ending juga sangatlah pasti dengan jumlah material yang sedikit sehingga bentuk ending sangatlah tidak jauh berbeda dari satu pertandingan dengan pertandingan catur lainnya, baik dari segi posisi maupun dari segi keunggulan material.



Gambar 6 : Ilustrasi kemungkinan opening pada database *engine* catur
(Sumber gambar : dibuat sendiri)



Gambar 7 : Ilustrasi kemungkinan ending pada database *engine* catur
(Sumber gambar : dibuat sendiri)

Biasanya, *engine* catur memainkan *ending* dengan tiga sampai enam buah tersisa di papan dan melihat tabel database ending yang besar untuk mencari kemungkinan langkah yang sesuai dengan posisi dan material saat *ending*.

Database *ending* yang biasa dipanggil *Tablebases* ini dimulai pencarian solusinya dari posisi final / akhir. *Tablebases* tersebut akan memprediksi posisi akhir dari sebuah permainan catur tersebut apakah berakhir dengan kemenangan, remis dengan *stalemate* ataupun material tidak mencukupi. Jadi, bila melihat dengan representasi pohon, maka penelusuran langkah yang diambil berbasis dari solusi yang akan diinginkan terlebih dahulu karena solusi pada *ending* itu biasanya hanya sedikit sehingga tingkat akurasi untuk prediksinya sangat tinggi. Algoritma yang dipakai dalam penelusuran dari solusi di *Tablebases* yang berupa pohon ini dimulai dari daun hingga ke posisi pada papan catur sekarang dengan menggunakan Complex Compression Algorithms (Eugene Nalimov). Biasanya, besar data dari semua ending untuk 3-6 buah catur yang tersisa di papan adalah 'hanya' sebesar 21 GB.

V. KESIMPULAN

Stuktur Pohon banyak digunakan sebagai representasi konkrit dalam sebuah permasalahan yang menyediakan banyak kemungkinan solusi. Strukturisasi pohon juga memudahkan dalam pencarian berdasarkan langkah per langkah tiap kedalaman maupun berdasarkan kategori per simpul dan anak simpulnya. Penerapan struktur pohon dalam *engine* catur termasuk salah satunya. Pohon merupakan pemodelan yang digunakan sebagai representasi solusi langkah-langkah dalam *engine* catur. Pemodelan dengan pohon ini mempermudah pencarian solusi-solusi langkah yang tepat dalam menghitung ketepatan langkah tersebut dengan akurasi yang tinggi.

Pencarian solusi dari pohon ini ternyata lebih efektif dengan metode *IDDFS* dengan metode pencarian dan penelusuran berdasarkan kombinasi antara metode pencarian *DFS* (berdasarkan kedalaman) untuk efektivitas kompleksitas spasial dan metode *BFS* (berdasarkan lebar / banyaknya cabang) yang berperan dalam hal efisiensi

kompleksitas waktu. Hal ini terbukti dengan penggunaan metode ini pada engine-engine catur terkemuka seperti *Deep Rybka 4, Houdini 1.5,* dan *Fritz 13*. Penerapan ini tentunya ada sesuai perkembangan zaman dari yang hanya menerapkan algoritma untuk keputusan 1 sampai 2 langkah terbaik ke depan saja, sampai pada pencarian langkah terbaik untuk berpuluh-puluh langkah ke depan dengan efektivitas yang baik dengan menerapkan struktur pohon dengan perkembangan algoritmanya.

Rekor yang tak terpatahkan dari *engine-engine* dengan kekuatan prosesor yang sudah mumpuni dengan algoritma yang terstruktur dan efisien tersebut berhasil membuktikan kecanggihan teknologi di masa kini dan masa depan.

REFERENCES

[1] http://www.geocities.com/SiliconValley/Lab/7378/comphis.htm, diakses tanggal 14 Desember 2013 pukul 22:03 WIB

[2] http://mufidnilmada.staff.gunadarma.ac.id/Downloads/files/9717/Penerapan+BFS+dan+DFS+pada+Pencarian+Solusi.ppt. diakses tanggal 15 Desember 2013 pukul 13.49 WIB

[3] JKorf, Richard (1985). "Depth-first Iterative-Deepening: An Optimal Admissible Tree Search". *Artificial Intelligence* **27**: 97–109.

[4] http://www.uio.no/studier/emner/matnat/ifi/INF4130/h12/undervisningsmateriale/chess-algorithms-theory-and-practice-ver2012.pdf diaskses tanggal 15 Desember 2013 pukul 15:23 WIB

[5] http://www.frayn.net/beowulf/theory.html diakses tanggal 15 Desember 2013 pukul 15:31 WIB

[6] http://www.tcm.phy.cam.ac.uk/BIG/pob24/deep_blue.pdf diakses tanggal 15 Desember 2013 pukul 15:37 WIB

[7] http://www.geocities.com/SiliconValley/Lab/7378/comphis.htm diakses tanggal 15 Desember 2013 pukul 15:39 WIB

[8] http://www.cruxis.com/chess/houdini.htm diakses tanggal 15 Desember 2013 pukul 15:41 WIB

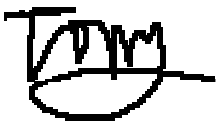
[9] http://www.computerchess.org.uk/ccrl/4040/ diakses tanggal 15 Desember 2013 pukul 15:43 WIB

[10] http://musaqif.blogspot.com/2010/04/search-carilacak-dalam-bidang_20.html diakses tanggal 15 Desember 2013 pukul 15:49 WIB

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, 15 Desember 2013



Tony
13512018