Pengembangan Algoritma *Best First Search* dalam Pencarian Solusi *Game Tree*

Tobias Natalio Sianipar - 13521090
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (outlook): tobiasnatalio@outlook.com

***Abstract*—Algoritma *best first search* adalah salah satu algoritma pencarian solusi dengan informasi (*informed search*) yaitu metode pencarian yang berbasis heuristik. *Game tree* atau AND/OR tree merupakan representasi grafis dari permainan yang berurutan, tiap level pada *game tree* secara bergantian menunjukkan giliran pemain. PNS (proof number search) merupakan algoritma *best first seacrh* yang menggunakan *proof number* serta *disproof number* untuk menentukan *node* yang akan ditelusuri selanjutnya.**

Keywords—Best First Search, Proof-Number Search, Game Tree, AND/OR Tree

I. PENDAHULUAN

Permainan dengan urutan main bergantian yang dimainkan oleh dua pemain merupakan permainan yang sering dijumpai dari zaman dahulu hingga pada saat ini. *Checkers, 9x9 Hex, Tsume-Shogi,* serta *Endgame Problem* pada permainan seperti catur dan *Go* merupakan jenis-jenis permainan yang dapat di representasikan menggunakan *game tree*. Hal ini dikarenakan permainan-permainan tesebut memiliki karakteristik yang sama yaitu dimainkan oleh dua pemain dan memiliki urutan yang giliran bergantian secara teratur.

Game Tree atau *AND/OR Tree* adalah salah satu bentuk representasi langkah-langkah pada suatu permainan yang memiliki dua pemain serta urutan giliran yang bergantian secara teratur. *AND/OR Tree* merupakan pohon berbatas yang terdiri atas simpul *OR* dan *AND*. Simpul *OR* merepresentasikan pemain pertama atau biasa dikaitkan dengan pencari solusi, sedangkan simpul *AND* merepresentasikan pemain kedua atau lawan bermain. *AND/OR Tree* dinyatakan *solved* ketika nilai simpul akar memenuhi suatu nilai spesifikasi kemenangan atau kekalahannya.

Algoritma *Best First Search* merupakan salah satu algoritma yang dapat mencari solusi dari suatu masalah yang dapat direpresentasikan menjadi sebuah pohon. Algoritma ini merupakan salah satu bagian dari algoritma pencarian dengan informasi (*informed search*) dengan memanfaatkan fungsi heuristik yang dapat membantu menentukan langkah pencarian selanjutnya pada suatu simpul pohon. Algoritma *Best First Search* telah banyak dikembangkan menjadi algoritma-algoritma baru yang memiliki fungsi lebih spesifik dalam suatu pencarian solusi masalah seperti, *Conspiracy Number Search, A*, B*, dll.* Algoritma ini memanfaatkan konsep *Priority Queue* yang dipadukan dengan fungsi heuristik tertentu untuk menentukan simpul pencarian yang akan ditelusuri lebih dahulu. Pada tahun 1994 telah dikembangkan salah satu algoritma terbaik yang didasari algoritma *best first search* untuk memecahkan masalah pencarian solusi *game tree* yang bernama *Proof-Number Search*.

Proof-Number Search (*PNS)* yang ditemukan oleh Allis *et al* pada tahun 1994 merupakan algoritma yang dikembangkan dari algoritma *Conspiracy Number Search* yang ditemukan oleh McAllester pada tahun 1985-1988. PNS memiliki nilai heuristik yang menunjukan daun yang paling menjanjikan dengan memilih *most-proving node (MPN)*. MPN merupakan daun yang dapat berkontribusi dalam penentuan keberhasilan atau kegagalan nilai akarnya. MPN ditemukan dengan mengeksploitasi dua karakteristik utama *search tree*, yaitu bentuk pohon dan nilai dari daunnya.

II. TEORI DASAR

A. Algoritma Best First Search

Algoritma *Best First Search* merupakan algoritma *searching* yang biasanya dilakukan pada *search tree* memanfaatkan fungsi heuristik untuk menentukan simpul yang akan ditelusuri. Algoritma ini menghitung heuristik simpul-simpul yang telah ditemukan dan mengurutkan simpul-simpul tersebut berdasarkan nilai heuristiknya dalam *priority queue*, kemudian simpul yang memiliki prioritas tertinggi akan diekspansi untuk menemukan simpul-simpul baru yang kemudian akan dihitung nilai heuristiknya dan dimasukkan kedalam *priority queue*, hal tersebut terus dilakukan berulang-ulang sampai memenuhi suatu simpul yang mencapai *goals* tertentu. Berbagai variasi algoritma *best first search* adalah *A*, B*, Conspiracy Number Search, Proof-Number Search, dll*. Terdapat 4 elemen utama yang digunakan pada algoritma *best first search*, yaitu

1) Himpunan kandidat: merupakan himpunan yang berisi daftar kandidat solusi pencarian yang diperbaharui tiap langkah dan disusun mengurut menggunakan *priority queue* berdasarkan nilai heuristik total solusi tersebut, himpunan ini juga dapat disebut sebagai himpunan simpul hidup.

2) Himpunan solusi: merupakan himpunan yang berisi daftar simpul yang telah dilalui untuk mencapai suatu simpul tertentu, himpunan ini merupakan solusi akhir dari masalah yang ada pada sebuah *search tree*.

3) Fungsi Evaluasi: merupakan fungsi yang memberikan nilai pada suatu kandidat solusi, fungsi ini mengolah nilai-nilai heuristik tiap simpul pada kandidat solusi menjadi suatu nilai evaluatif yang dapat menetukan keunggulan suatu daun dari daun lainnya.

4) Fungsi Heuristik: merupakan fungsi yang dibuat untuk memprediksi bobot nilai suatu simpul terhadap pencarian solusi berdasarkan informasi yang telah diketahui pembuat melalui fakta-fakta atau *rules* pada persoalan yang akan dipecahkan.

B. *Identify the Headings*

Game Tree atau *AND/OR Tree* merupakan salah satu jenis *search tree* yang dikembangkan dengan tujuan merepresentasikan langkah-langkah pada permainan dengan giliran teratur yang dimainkan dua orang. *AND/OR Tree* adalah *search tree* dengan akar dan memiliki batas yang terdiri atas simpul simpul *OR* sebagai representasi pemain pertama (pemain yang mencari langkah solusi) dan simpul *AND* sebagai representasi pemain kedua (pemain yang menjadi lawan). Semua simpul selain simpul akar memiliki *parent*. Setiap sisi simpul dengan *parent*-nya merepresentasikan langkah yang dapat dilakukan pemain, sedangkan simpul-simpul pada *AND/OR Tree* merepresentasikan *gamestate* permainan. Pada beberapa sumber, akar dari *Game Tree* selalu merupakan simpul *OR*. Setiap nilai pada simpul *AND/OR* memiliki nilai *win*, *loss*, atau *unknown*. Setiap simpul dengan nilai *win* atau *loss* mengindikasikan langkah yang diketahui dapat menyebabkan kemenangan atau kekalahan pada pemain pertama sedangkan simpul dengan nilai *unknown* perlu diekspansi terlebih dahulu untuk menentukan apakah simpul tersebut menyebabkan kemenangan atau kekalahan bagi pemain pertama. Simpul-simpul pada *Game Tree* diklasifikasikan menjadi tiga, yaitu

1) Simpul terminal leaf (terminal): merupakan simpul yang tidak memiliki *children*.

2) Simpul internal (internal): merupakan simpul yang memiliki paling sedikit satu *child*.

3) Simpul non-terminal leaf (leaf): merupakan simpul yang belum diekspansi dalam tahap pencarian, simpul tidak dapat diketahui apakah merupakan simpul *terminal* atau *internal* sampai simpul ini diekspansi.

Proof tree merupakan *subtree* dari *AND/OR Tree* yang tersusun atas simpul-simpul yang merepresentasikan langkah-langkah yang dilakukan untuk mencapai kemenangan bagi pemain pertama. Sebuah *proof tree T* dengan akar *r* memiliki sifat-sifat sebagai berikut:

1) Simpul akar berada dalam *T*.

2) Untuk setiap simpul internal *OR* dalam *T*, sedikitnya satu *child* simpul tersebut berada dalam *T*.

3) Untuk setiap simpul internal *AND* dalam *T*, seluruh *child* simpul tersebut berada dalam *T*.

4) Untuk setiap simpul terminal dalam *T* memiliki nilai *win*.

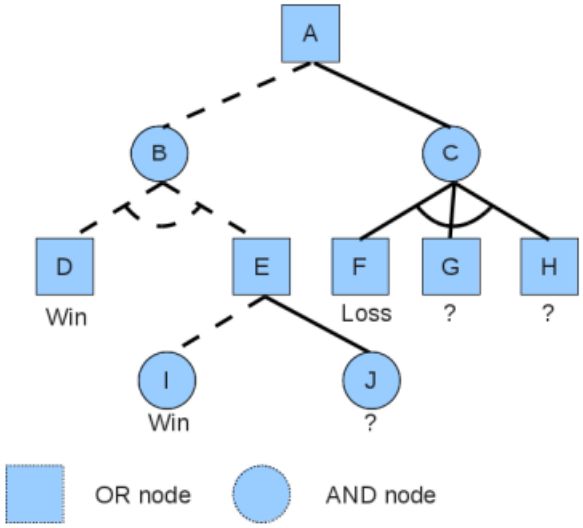
Disproof tree merupakan *subtree* dari *AND/OR Tree* yang tersusun atas simpul-simpul yang merepresentasikan langkah-langkah yang dilakukan untuk mencapai kemenangan bagi pemain kedua. Sebuah *disproof tree T* dengan akar *r* memiliki sifat-sifat sebagai berikut:

1) Simpul akar berada dalam *T*.

2) Untuk setiap simpul internal *AND* dalam *T*, sedikitnya satu *child* simpul tersebut berada dalam *T*.

3) Untuk setiap simpul internal *OR* dalam *T*, seluruh *child* simpul tersebut berada dalam *T*.

4) Untuk setiap simpul terminal dalam *T* memiliki nilai *loss*.



Gambar 2.1. Contoh *AND/OR* Tree. Lingkaran merepresentasikan *AND*. Persegi merepresentasikan *OR*

Sumber: https://webdocs.cs.ualberta.ca/~mmueller/ps/ICGA2012PNS.pdf

C. *Proof-Number Search*

Proof-Number Search (*PNS*) adalah algoritma *best first search* yang didasari algoritma *Conspiracy Number Search*. Algoritma ini menggunakan fungsi heuristik yang menghitung *proof number* dari suatu simpul pada *game tree* dan memilih *most-proving node* (*MPN*) dari antaranya. Untuk menemukan MPN, PNS menyimpan dua angka penting pada tiap simpul *game tree*, yaitu *proof number* (*pn*) dan *disproof number* (*dpn*). Pada fungsi heuristik ini, *dpn* dianggap sebagai *lower bound* dari jumlah simpul turunan yang perlu diekspansi untuk membuktikan simpul dengan *dpn* tersebut tidak menghasilkan solusi yang diinginkan, sedangkan *pn* dianggap sebagai *lower bound* dari jumlah simpul turunan yang perlu diekspansi untuk membuktikan simpul dengan *pn* tersebut menghasilkan solusi yang diinginkan.

Untuk setiap simpul *internal* pada *Game Tree* yang ditelusuri oleh algoritma ini akan diinisialisasi dengan nilai *pn* dan *dpn* sama dengan satu. Kemudian nilai *pn* dan *dpn* pada tiap simpul dikomputasi ulang melalui *backpropagation* dengan rumus sebagai berikut:

1) Untuk simpul *OR N* dengan himpunan *child succ(N)*,

$$pn(N) = min(\, pn(C) \,)_{C \in succ(N)}$$

$$dpn(N) = \sum_{C \in succ(N)} dpn(C)$$

2) Untuk simpul *AND N* dengan himpunan *child succ(N)*,

$$pn(N) = \sum_{C \in succ(N)} pn(C)$$

$$dpn(N) = min(\, dpn(C) \,)_{C \in succ(N)}$$

Untuk setiap simpul *terminal*, jika *gamestate* simpul menunjukkan kemenangan bagi pemain pertama, maka nilai *pn* simpul tersebut adalah 0 dan nilai *dpn* simpul tersebut adalah ∞, hal ini menunjukkan bahwa tidak ada lagi simpul yang perlu dibuktikan untuk menyetujui simpul tersebut bernilai *win* dan tidak mungkin menyangkal bahwa simpul itu tidak bernilai *win* karena perlu tak hingga jumlahnya simpul untuk menyangkalnya. Begitu pula sebaliknya bila suatu simpul *terminal* menyatakan kekalahan bagi pemain pertama, maka nilai *pn* pada simpul tersebut adalah ∞ dan nilai *dpn* pada simpul tersebut adalah 0.

Setelah menemukan *pn* dan *dpn* tiap simpul yang telah di ekspansi, MPN ditentukan dengan logika berikut:

1) Untuk simpul *OR*, mengubah nilai *dpn* dari simpul *child* manapun akan mengubah *dpn* simpul *parent*-nya, sedangkan perubahan pada nilai *pn* simpul *child* hanya akan berpengaruh bila simpul *child* tersebut mempunyai nilai *pn* minimum diantara *child* lainnya, sehingga MPN dari simpul *OR* adalah *child* dengan nilai *pn* paling minimum sehingga perubahan pada simpul ini akan berpengaruh terhadap simpul akar.

2) Untuk simpul *AND*, mengubah nilai *pn* dari simpul *child* manapun akan mengubah *pn* simpul *parent*-nya, sedangkan perubahan pada nilai *dpn* simpul *child* hanya akan berpengaruh bila simpul *child* tersebut mempunyai nilai *dpn* minimum diantara *child* lainnya, sehingga MPN dari simpul *AND* adalah *child* dengan nilai *pn* paling minimum sehingga perubahan pada simpul ini akan berpengaruh terhadap simpul akar.

Dengan kata lain, algoritma ini memfokuskan simpul yang dapat merubah nilai *pn* dan *dpn* simpul akar karena penentuan *win* atau *loss* suatu *game tree* bergantung pada nilai *win/loss* akarnya.

Secara garis besar, algoritma ini merupakan loop yang terdiri atas empat langkah, yaitu

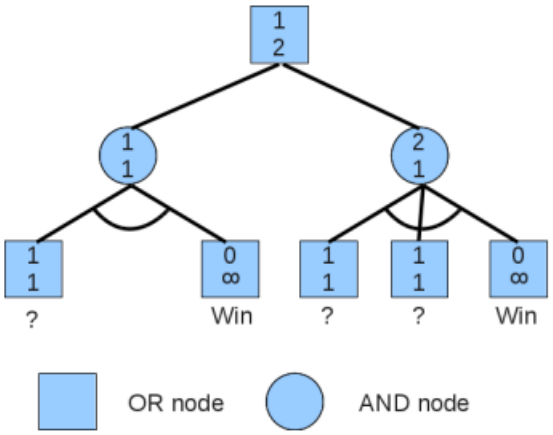
1) Pemilihan MPN berdasarkan fungsi heuristik yang telah dijelaskan diatas.

2) Ekspansi MPN dengan membuat simpul *child* untuk tiap langkah yang dapat dilakukan dari *gamestate* simpul MPN.

3) Evaluasi tiap simpul *child* untuk menentukan *win*, *loss*, atau inisiasi nilai *pn* dan *dpn* sama dengan 1.

4) Backpropagate dan mengupdate tiap simpul yang telah diekspansi pada *game tree*.

Hal ini akan terus dilakukan sampai nilai akar *game tree* mencapai *win* atau *loss*.



Gambar 2.2. Contoh *PNS*

Sumber: https://webdocs.cs.ualberta.ca/~mmueller/ps/ICGA2012PNS.pdf

III. Pembahasan Pencarian Solusi Game Tree Menggunakan Algoritma Best First Search

Pada pembahasan pencarian solusi *game tree* pada makalah ini, saya akan menggunakan salah satu algoritma *best first search*, yaitu *Proof-Number Search* terhadap suatu persoalan *endgame Tic Tac Toe* berikut.

O	X	O
X	X	a
c	O	b

Gambar 3.1. Persoalan *endgame tic tac toe*

A. *Batasan Masalah*

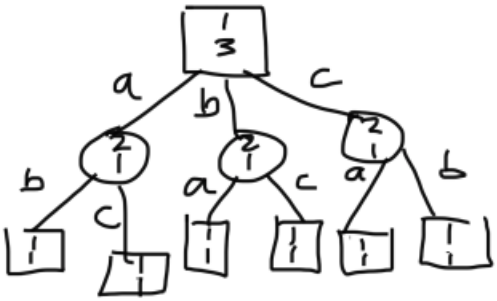
Dalam persoalan *endgame tic tac toe* tersebut akan diterapkan aturan sebagai berikut:

- Pada gambar diatas a, b, c merepresentasikan grid yang belum terisi.

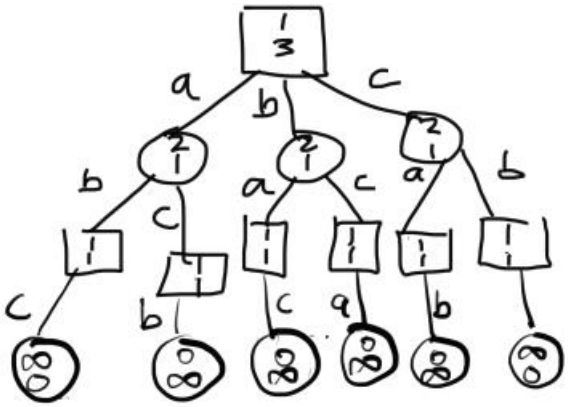
- Permainan *tic tac toe* ini dimulai dengan ‘O’ sebagai pemain pertama.
- Permainan dimenangkan bila terdapat simbol ‘X’ atau ‘O’ yang berbentuk horizontal, vertikal, maupun diagonal.

B. Penyelesaian Masalah

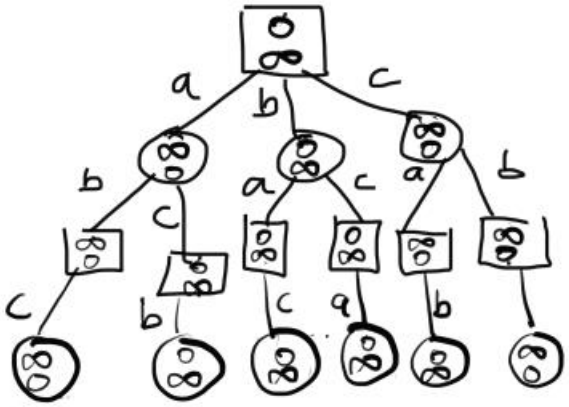
Pertama kita perlu menginisiasi *game tree* 3 level sebagai berikut, kemudian tentukan *pn dan dpn* sesuai algoritma yang dijelaskan pada landasan teori.



Pada gambar tersebut dapat titentukan MPN adalah semua daun dimulai dari daun paling kiri, karena semua daun *OR* bernilai *pn* sama. Kemudian lakukan ekspansi satu persatu daun sehingga menghasilkan *game tree* berikut.



Lakukan *backpropagation* dan *update* semua nilai *dpn* dan *pn* pada tiap simpul sehingga dihasilkan *game tree* sebagai berikut.



Perhatikan bahwa simpul akar pada *game tree* tersebut telah mencapai nilai *win* dimana *pn* sama dengan 0 dan *dpn* sama dengan ∞, hal ini menunjukkan bahwa *endgame tic tac toe* tersebut dapat dimenangkan dengan penempatan ‘O’ pada posisi *b* pada awal giliran.

IV. KESIMPULAN

Salah satu variasi dari algoritma *best first search*, yaitu *Proof-Number Search* merupakan algoritma yang cocok digunakan untuk mencari solusi dari permainan yang dimainkan oleh dua pemain serta memiliki giliran yang berurutan.

Penggunaan fungsi heuristik pada algoritma ini dibuat sedemikian rupa sehingga penentuan solusi dapat dilakukan dengan iterasi yang seminimal mungkin. Masalah pada algoritma ini berada pada pemakaian memori yang besar seiring bertambahnya tingkat kerumitan solusi dari suatu *game tree*.

Game tree merupakan representasi *search tree* yang sangat baik digunakan untuk menggambarkan *game state* serta *langkah-langkah* pemain pada suatu permainan yang dimainkan dua orang dan memiliki giliran yang terurut.

VIDEO LINK AT YOUTUBE *(Heading 5)*

Makalah ini tidak memiliki video.

ACKNOWLEDGMENT *(Heading 5)*

Puji dan syukur kita pajatkan kepada Tuhan YME, karena atas berkat rahmat dan karuniaNya, penulis dapat menyelesaikan tugas makalah mata kuliah Strategi dan Algoritma tahun ajaran 2022/2023 ini dengan tepat waktu. Selain itu, penulis juga mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc., dosen pengajar mata kuliah Strategi Algoritma K02, yang telah membimbing penulis dalam pemahaman teoriteori yang telah dimanfaatkan untuk pengerjaan makalah ini. Kemudian, saya juga berterimakasih kepada Bapak Dr. Ir. Rinaldi M. T., yang telah menyediakan situs yang memuat berbagai macam slide yang menjelaskan teori-teori yang telah digunakan pada makalah ini dengan sangat baik. Semoga kebaikan Ibu dan Bapak dapat dibalas oleh Tuhan YME dengan

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors).*

sebaik-baiknya balasan. Penulis juga meminta maaf apabila dalam kepenulisan makalah ini, penulis memiliki banyak kesalahan karena sejatinya penulis adalah pelajar yang masih harus terus mencari ilmu. Semoga makalah ini dapat menjadi manfaat untuk orang banyak, tidak hanya dalam lingkup mahasiswa informatika saja, tetapi juga untuk masyarakat umum. Harapan dari penulis, makalah ini dapat meningkatkan pemahaman orang terhadap implementasi ilmu matematika pada kehidupan nyata karena sejatinya matematika adalah ilmu yang dapat tergantikan pada kehidupan ini. Penulis juga berharap ilmu yang dikaji dalam makalah ini tidak digunakan untuk tindakan yang melanggar hukum.

REFERENCES

[1] https://minimax.dev/docs/ultimate/pn-search/#a-sketch-of-pn-search diakses pada 22 Mei 2023

[2] https://webdocs.cs.ualberta.ca/~mmueller/courses/2014-AAAI-games-tutorial/slides/AAAI-14-Tutorial-Games-4-Proof-Number-Search.pdf diakses pada 22 Mei 2023

[3] The PN∗-search algorithm: Application to tsume-shogi, Masahiro Seo, Hiroyuki Iida, Jos W.H.M. Uiterwijk. Osaka: Japan 29 December 1999.

[4] https://webdocs.cs.ualberta.ca/~mmueller/ps/ICGA2012PNS.pdf diakses pada 22 Mei 2023.

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, 22 Mei 2023

Tobias Natalio Sianipar 13521090