Implementasi Algoritma Greedy, BFS, Branch and Bound, dan Metode Heuristik dalam Permainan *Reversi*

Gilang Julian Suherik - 13512045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
gilang.9h@gmail.com

***Abstrak*—Dalam makalah ini dibahas mengenai permainan reversi, beberapa algoritma yang mungkin digunakan dalam pembuatan *Artificial Intelligence* untuk game tersebut, kelemahan dan kelebihan dari masing-masing algoritma, serta bagaimana metode heuristik akan mempengaruhi kinerja dari setiap algoritma.**

Index Terms*—*Reversi, Greedy, Breadth First Search, Branch and Bound, Heuristik, minimax.

I. PENDAHULUAN

Reversi adalah salah satu *puzzle board game* yang didesain untuk dimainkan dua orang pemain. Permainan ini terdiri atas sebuah papan dengan kotak 8x8, 64 buah kepingan dengan dua warna berbeda. Permainan dimulai dengan meletakkan 4 buah kepingan di bagian tengah papan dengan warna yang berselang-seling.

Masing-masing pemain kemudian memilih warna yang berbeda. Tujuan dari permainan ini adalah mendapatkan sebanyak mungkin kepingan dengan warna yang dipilih oleh masing-masing pemain di akhir permainan. Pemain dengan jumlah kepingan yang lebih banyak di akhir permainan adalah pemain yang menjadi pemenang.

Pemain dapat menaruh kepingan saat gilirannya berlangsung dengan aturan kepingan dengan warna miliknya mengapit satu atau lebih kepingan dengan warna lawannya yang membentuk barisan vertikal, horizontal, ataupun diagonal. Saat seorang pemain meletakkan kepingannya (dengan posisi yang valid) maka kepingan milik lawan yang berada di antara dua kepingan pemain tersebut akan dibalik (berubah warna) menjadi kepingan dengan warna sebaliknya.

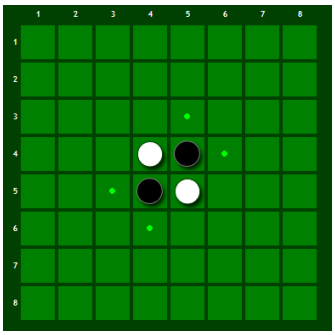
Permainan berlanjut selama masih ada pemain yang dapat meletakkan kepingannya sesuai dengan syarat mengapit minimal satu buah kepingan milik lawan. Jika hanya ada seorang pemain yang tidak memiliki langkah valid untuk meletakkan kepingannya, maka giliran pemain tersebut akan dilewati. Sehingga seorang pemain mungkin mendapat beberapa giliran berturut-turut.

Reversi pertama kali ditemukan pada tahun 1883 oleh salah satu dari dua orang berkebangsaan Inggris Lewis Waterman / John W. Mollet yang masing-masing dari mereka menuduh satu sama lain sebagai penipu. Saat ini versi digital dari permainan reversi jauh lebih terkenal dibandingkan dengan *board game* aslinya. Bahkan banyak orang mungkin hanya mengetahui versi digital dari permainan ini.

Bentuk digital dari sebuah game tidak mungkn hanya dapat dimainkan berdua. Idealnya sebuah game yang hanya bisa dimainkan berdua harus memiliki AI (*Artificial Intelligence*) sebagai lawan.

Artificial Intelligence dalam sebuah game tentu harus dibuat oleh menggunakan algoritma-algoritma tertentu. Beberapa jenis algoritma yang dipakai dalam pembuatan *Artificial Intelligence* reversi diantaranya adalah algoritma greedy, minimax, BFS, dan Branch and Bound.

.



Gambar 1.1 Awal permainan reversi
Sumber: http://webversi.com

II. DASAR TEORI

2.1. Algoritma Greedy

Algoritma greedy adalah salah satu algoritma yang banyak sekali dipakai dalam menyelesaikan suatu masalah optimasi. Algoritma ini berkerja dengan cara membentuk solusi secara langkah-per langkah dengan mengambil langkah yang dianggap paling menguntungkan di saat itu namun tidak melewati batasan-batasan yang telah ditentukan.

Dengan memanfaatkan pilihan paling menguntungkan di setiap langkahnya diharapkan solusi yang dihasilkan dari penerapan algoritma ini juga merupakan solusi yang paling menguntungkan / optimal.

Namun dalam kenyataannya tidak selalu pilihan paling menguntungkan di setiap saatnya akan mengarah

kepada solusi yang paling menguntungkan karena kita tidak dapat mengetahui arah yang dituju dari setiap pilihan yang diambil, mungkin saja di akhir suatu pilihan yang tidak menguntungkan akan didapat suatu hasil yang besar.

Salah satu faktor yang membuat algoritma greedy tidak dapat menghasilkan solusi yang paling optimal adalah batasan bahwa algoritma ini tidak memperbolehkan untuk melakukan perhitungan kembali kepada langkah yang telah dilewati.

2.2. Algoritma Breadth First Search (BFS)

Algoritma Breadth First Search merupakan algoritma pencarian yang memanfaatkan pohon ruang status. Pencarian dimulai dari state awal kemudian dilanjutkan dengan pembangkitan seluruh subpohon berdasarkan state yang mungkin didapat dari state awal tersebut. Selanjutnya dari masing-masing subpohon tersebut dibentuk lagi subpohon-subpohon lainnya hingga solusi yang dicari ditemukan dalam pohon.

Algoritma Breadth First Search meliputi simpul-simpul state, antrian simpul yang akan diperluas, serta daftar simpul yang telah dikunjungi untuk menecgah terjadinya penelusuran ulang sebuah simpul yang sama.

2.3. Algoritma Branch and Bound

Branch and Bound merupakan algoritma hasil dari pengembangan algoritma pencarian Breadth First Search (BFS) yang memanfaatkan pohon ruang status. Perkembangan yang terdapat dalam algoritma ini adalah berkurangnya perluasan pohon / subpohon yang sebenarnya tidak perlu diperluas dengan memanfaatkan sistem prioritas.

Setiap simpul dari pohon akan memiliki sebuah nilai / cost yang didapat dari perhitungan terhadap kondisi / status dari simpul tersebut. Proses perluasan pohon kemudian tidak dilakukan secara keseluruhan, melainkan hanya subpohon dengan cost terendah saat itu saja yang akan diperluas.

III. PENERAPAN BEBERAPA ALGORITMA DALAM *ARTIFICIAL INTELLIGENCE* REVERSI

Dalam setiap algoritma selalu dibutuhkan koordinat kotak-kotak yang dapat diisi oleh kepingan pada suatu giliran. Maka apapun algoritma yang digunakan pasti meliputi sebuah fungsi yang berguna untuk melakukan pengecekan terhadap setiap kotak dan menghasilkan kumpulan koordinat dari kotak yang dapat diisi.

Selanjutnya algoritma-algoritma yang digunakan hanyalah berfungsi untuk menentukan kotak mana yang paling menguntungkan untuk diisi menurut algoritma tersebut.

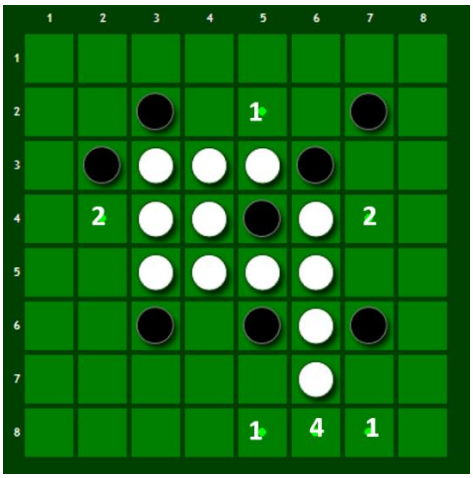
3.1. Penerapan Algoritma Greedy

Penerapan greedy dalam AI reversi relatif mudah karena yang perlu kita lakukan hanyalah memilih kotak yang jika diisi saat itu akan menghasilkan skor terbesar. Langkah-langkah dari algoritma *greedy* kurang lebih adalah sebagai berikut:

- Tentukan kotak-kotak yang valid untuk diisi pada giliran tersebut.
- Coba satu-persatu mengisi kotak-kotak tersebut sambil mencatat skor yang dihasilkan saat melakukan pengisian di kotak tersebut.
- Bandingkan skor dari setiap pengisian tersebut, pilih pengisian yang menghasilkan skor tertinggi.

Kelebihan dari penggunaan algoritma ini adalah pada setiap gilirannya akan selalu didapatkan skor tertinggi yang mungkin diperoleh pada saat itu. Namun secara langsung kelebihan ini juga mencerminkan kelemahannya, yaitu tidak diperhitungkannya langkah-langkah untuk waktu jauh ke depan (keadaan dalam permainan mungkin bisa berubah menjelang akhir permainan dan skor bisa berbalik).

Kelemhan lain dari algoritma greedy adalah algoritma ini juga tidak dapat memperhitungkan keadaan yang mungkin menguntungkan lawan. Misalnya dengan mengambil langkah secara greedy, lawan mungkin akan dapat menguasai kotak sudut yang tidak mungkin untuk direbut kembali.



Gambar 3.1.1 Perhitungan keuntungan algoritma greedy
Sumber: http://webversi.com

3.2. Penerapan Algoritma Branch and Bound

Algoritma branch and bound dalam reversi memiliki dasar yang mirip dengan greedy, namun pertimbangan untuk melangkah tidak hanya yang menguntungkan untuk saat ini saja, melainkan memungkinkan untuk beberapa langkah ke depan tergantung pengaturan kedalaman yang dipilih.

Seperti algoritma branch and bound pada umumnya, perhitungan dimulai dengan membuat simpul-simpul yang mungkin dicapai dari simpul akar, menghitung nilai dari setiap simpul, dan melakukan pembentukan simpul-simpul selanjutnya.

Algoritma branch and bound diimplementasi dengan langkah-langkah sebagai berikut:

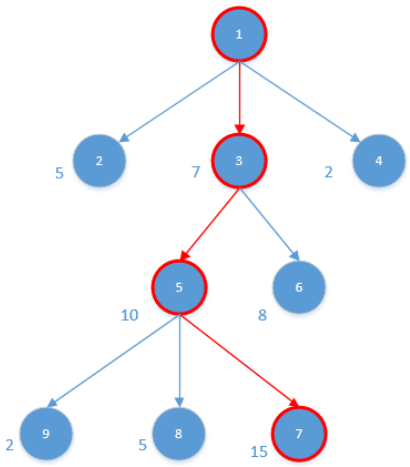
- Tentukan batasan kedalaman pembentukan pohon ruang status.

- Buat semua simpul yang mungkin dicapai dari simpul saat ini (seluruh tempat dimana kepingan mungkin diletakkan).
- Hitung nilai masing-masing simpul serperti pada algoritma greedy.
- Pilih simpul dengan nilai yang paling tinggi, dan lakukan perluasan pada simpul tersebut.
- Penghitungan nilai simpul selanjutnya (yang merupakan simpul kemungkinan langkah lawan) adalah dengan cara semakin besar kemungkinan lawan menghasilkan skor besar, maka semakin kecil nilai dari simpul tersebut.
- Pilih kembali simpul dengan nilai tertinggi dari semua simpul yang telah dibangkitkan dan kembali lakukan langkah 2.
- Perluasan berhenti saat pohon telah mencapai level yang ditentukan.
- Selanjutnya simpul yang dipilih adalah simpul level 1 dari simpul yang mencapai level paling dalam.

Kelebihan dari penggunaan algoritma ini sama dengan kelebihan algoritma branch and bound pada umumnya, yaitu berkurangnya jumlah simpul yang harus dibangkitkan sehingga dapat mengurangi waktu komputasi.

Kekurangan dari algoritma ini adalah adanya kemungkinan langkah menguntungkan yang mungkin sama sekali tidak dievaluasi jika untuk mencapai langkah tersebut harus melewati langkah yang tidak menguntungkan terlebih dahulu.

Selain itu, masalah posisi-posisi yang tidak menguntungkan seperti terambilnya kotak yang berada di sudut oleh lawan juga tidak dipertimbangkan dalam algoritma ini.



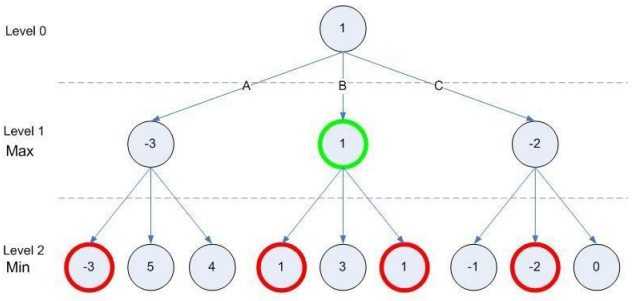
Gambar 3.2.1 Ilustrasi pemilihan keputusan algoritma branch & bound

3.3. Penerapan Algoritma BFS

Algoritma BFS sebenarnya tidak digunakan secara langsung dalam AI reversi. Karena pada dasarnya BFS adalah algoritma yang digunakan untuk mencari suatu nilai yang sudah pasti, sedangkan dalam reversi yang dibutuhkan adalah sebuah nilai yang optimal sehingga dibutuhkan metode lain untuk mencari nilai optimal tersebut.

Algoritma yang populer digunakan dalam reversi salah satunya yaitu algoritma minimax yang meliputi algoritma BFS untuk membuat pohon ruang status yang diperlukan. Proses yang dilakukan algoritma ini secara umum adalah sebagai berikut:

- Pertama-tama tentukan batas kedalaman dari pohon yang akan dibentuk.
- Bentuk pohon ruang status dengan kedalaman yang telah ditentukan menggunakan algoritma BFS.
- Setiap level dari pohon merepresentasikan pemain mana yang sedang aktif secara berselang-seling.
- Setiap simpul merepresentasikan state hasil langkah yang mungkin diambil pemain dan setiap simpul mengandung sebuah nilai yang merupakan hasil pengurangan skor lawan terhadap skor kita.
- Idealnya jika level simpul yang sedang dievaluasi saat ini merepresentasikan giliran kita, maka yang menguntungkan adalah simpul dengan nilai tertinggi.
- Sebaliknya, kita juga mengasumsikan lawan bersifat “cerdas” artinya kita asumsikan saat level simpul yang dievaluasi merepresentasikan giliran lawan, maka ambil kemungkinan terburuk (nilai simpul terkecil) sebagai langkah preventif.
- Dimulai dari simpul daun, ubah nilai simpul parent masing-masing daun dengan nilai tertinggi dari simpul anaknya. Kemudian hapus seluruh daun sehingga simpul parent dari daun akan menjadi daun. Ulangi langkah tersebut hingga tersisa simpul-simpul yang merupakan anak dari simpul akar. Pilih simpul dengan nilai tertinggi sebagai solusi.



Gambar 3.3.1 Ilustrasi pemilihan keputusan algoritma minimax
Sumber: http://i175.photobucket.com/albums/w139/habsq/minimax-2.jpg

Kelebihan dari algoritma ini adalah teratasinya langkah yang mungkin diambil lawan yang akan menyulitkan kita di beberapa langkah ke depan.

Namun algoritma ini juga tetap memiliki kelemahan. Salah satunya sama seperti kelemahan algoritma lainnya, yaitu saat terjadinya pemilihan kotak yang akan menguntungkan lawan untuk menguasai kotak sudut.

3.4. Penerapan Metode Heuristik

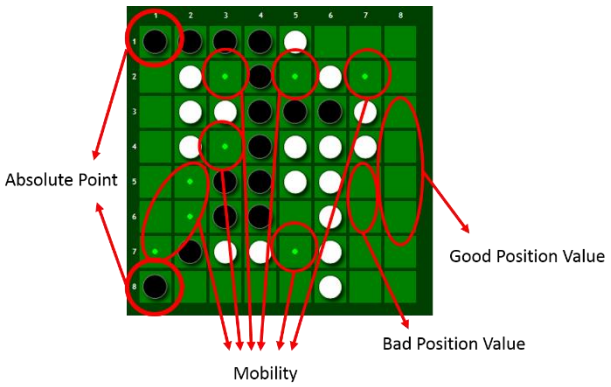
Pada setiap algoritma selalu dimungkinkan suatu langkah yang menguntungkan menurut algoritma tersebut, namun sebenarnya merugikan akibat adanya kemungkinan dikuasainya kotak yang berada di sudut papan, kemungkinan hilangnya langkah valid yang dapat kita lakukan pada giliran selanjutnya, dan lain-lain.

Ketika ada langkah kita yang memungkinkan terjadinya kondisi-kondisi merugikan tersebut, asalkan langkah tersebut menguntungkan bagi kita (dalam hal skor) maka akan tetap dijadikan solusi. Walaupun memang sekilas tidak ada yang salah pada pilihan ini, namun dari pengalaman kita bermain kita ketahui bahwa pilihan semacam ini sangat fatal.

Akibat adanya kesalahan semacam ini yang tidak dapat dihitung oleh sistem, maka harus kita definisikan sendiri aturan-aturan tambahan dalam perhitungan. Aturan-aturan tambahan yang bersifat intuitif semacam ini disebut metode heuristik.

Beberapa aturan-aturan yang didapat secara heuristik pada setiap pemeriksaan simpulnya antara lain:

1. Absolute Point
   adalah jumlah dari kepingan yang tidak dapat diubah oleh lawan / pada saat itu telah menjadi point yang pasti didapatkan oleh kita.
2. Mobility
   adalah tingkat kebebasan / banyaknya titik yang dapat ditempati oleh kepingan yang akan ditaruh oleh kita pada saat itu.
3. Position Value
   adalah perbedaan prioritas bagi penempatan kepingan. Misalnya akan lebih bagus jika menempati kotak di sudut daripada kotak yang berada di sampingnya.
4. Blocking Point
   adalah cara yang memastikan jika ditemukan langkah yang akan menghilangkan kemungkinan langkah valid bagi lawan pada giliran selanjutnya, maka langkah tersebut harus diambil.



Gambar 3.2.2 Penerapan heuristik pada reversi
Sumber: http://webversi.com

Metode heuristik sebenarnya agak sulit untuk diimplementasikan secara pasti karena tidak ada tolak ukur yang jelas (berupa nilai eksplisit) seberapa merugikan / menguntungkannya suatu langkah saat dieksekusi. Sehingga implementasi metode heuristik seseorang kemungkinan besar akan berbeda dengan orang yang lain walaupun algoritma yang digunakan sama persis.

Cara yang paling mudah untuk mengimplementasikan metode heuristik yaitu dengan memberikan pengaruh penilaian terhadap setiap keputusan menggunakan intuisi yang kita miliki. Setiap algoritma yang telah dibahas sebelumnya menggunakan sebuah "nilai" yang cara perhitungannya ditentukan oleh masing-masing algoritma. Contohnya dibahas sebagai berikut:

- Heuristik pada Algoritma Greedy

  Sebelumnya algoritma greedy hanya akan memilih langkah yang menghasilkan skor terbesar saat itu. Misalnya ada dua kotak yang dapat kita isi saat ini, saat mencoba mengisi kotak pertama didapat bahwa kita akan mendapat 4 poin, dan saat mengisi kotak kedua didapat angka 7 poin. Tentu greedy akan memilih kotak yang menghasilkan 7 poin.

  Namun ternyata 7 poin tersebut banyak didapat dari kotak-kotak yang berada di tengah, sedangkan pilihan lainnya yang bernilai 4 poin banyak mendapatkan poinnya dari kotak-kotak yang berada di sisi-sisi papan, dan ini lebih menguntungkan.

  Sehingga bisa kita tambahkan aturan, misalnya untuk poin yang didapat dari kotak yang berada di sisi akan dikali 1.5, poin dari kotak yang berada di daerah tengah papan akan dikali 1, dan poin dari kotak yang berada di sudut akan dikali 3. Kemudian hasil dari perhitungan inilah yang digunakan untuk menentukan pilihan secara greedy.

- Heuristik pada Algoritma Branch and Bound

  Algoritma greedy hanya dapat melakukan evaluasi terhadap keadaan dari permainan pada suatu saat saja dan tidak dapat melakukan evaluasi jauh ke depan karena keputusan dibentuk secara langkah demi langkah dan tidak dapat mengevaluasi kembali langkah yang telah diambil apalagi mengubah keputusan.

  Akibat adanya keterbatasan itu, metode heuristik yang memungkinkan untuk diimplementasikan terhadap algoritma greedy hanyalah metode perhitungan positional value.

  Karena algoritma branch and bound dapat memperhitungkan kemungkinan langkah untuk beberapa giliran ke depan, maka bisa dilakukan implementasi dari metode hauristik lainnya. Misalnya selain menambahkan prinsip positional value, dapat juga ditambahkan prinsip *mobility* pada algoritma ini.

  Dalam kondisi-kondisi tertentu, penggunaan prinsip mobility akan sangat mempengaruhi permainan. Misalkan hanya digunakan algoritma branch and bound dengan penambahan aturan

positional value. Suatu saat mungkin terjadi keadaan dimana langkah yang diambil sangat menguntungkan, namun untuk beberapa langkah ke depan yang tidak dievaluasi ternyata kemungkinan tidak adanya langkah valid yang dapat dilakukan pada saat itu sangat tinggi. Dilewatinya giliran melangkah akibat tidak adanya langkah valid yang dapat dilakukan jelas sangat merugikan.

Kemungkinan terjadinya keadaan semacam ini dapat diperkecil dengan memberikan prioritas bagi langkah yang menghasilkan tingkat mobilitas yang tinggi. Misalnya untuk pemberian nilai sebuah simpul dapat ditambahkan dengan banyaknya kotak yang mungkin dapat diisi pada giliran kita berikutnya.

- Heuristik pada Algoritma Minimax

Penggunaan metode heuristik pada algoritma minimax mungkin adalah yang paling menguntungkan dibandingkan dengan algoritma lainnya. Hal ini disebabkan pada algoritma minimax, pohon ruang status dibentuk secara keseluruhan menggunakan BFS, sehingga simpul yang dievaluasi akan lebih banyak. Banyaknya simpul yang dievaluasi sedikit menguntungkan bagi penggunaan heuristik karena aturan yang digunakan pada metode heuristik terkesan tidak memiliki nilai yang pasti, sehingga simpul yang diprediksikan tidak menguntungkan (walaupun pada kenyataannya menguntungkan) pun akan dievaluasi.

Keseluruhan aturan heuristik yang telah dituliskan di atas dapat diimplementasikan pada algoritma minimax ini. Misalnya saja pertimbangan langkah yang memungkinkan kita mendapatkan kotak yang berada di sudut untuk beberapa langkah ke depan. Keadaan semacam ini akan menambah nilai simpul yan dipilih dengan sangat besar karena pertama lokasi kotak yang berada di sudut menambah nilai simpul (lokasi yang strategis) serta tambahan dari absolute point (kotak yang berada di sudut tidak mungkin untuk direbut).

Metode lain yang dapat diimplementasikan yaitu dengan melihat kemungkinan langkah yang kita pilih yang dapat membuat lawan tidak dapat bergerak pada giliran berikutnya. Karena algoritma minimax mengevaluasi seluruh alternatif solusi, maka akan lebih mudah didapatkan simpul yang memiliki kemungkinan tinggi lawan tidak dapat bergerak.

IV. PENGUJIAN

Untuk melakukan pengujian, diimplementasikan beberapa algoritma dalam permainan reversi. Kemudian selanjutnya pengujian terhadap beberapa algoritma dilakukan sebagai berikut:

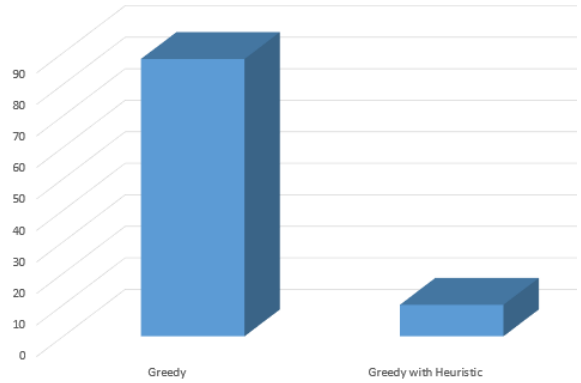
4.1. Algoritma Greedy dengan dan tanpa Metode Heuristik

Pengujian pertama dilakukan dengan "mengadu" algoritma greedy yang sama dengan perbedaan algoritma pertama tidak menerapkan metode heuristik, sedangkan algoritma kedua telah menggunakan metode heuristik.

Dari 100 permainan yang dilakukan didapat bahwa algoritma tanpa heuristik menang sebanyak 10 kali, algoritma dengan heuristik menang sebanyak 88 kali, dan terjadi hasil seri sebanyak 2 kali.

Heuristik yang dipakai dalam algoritma ini hanya prinsip positional value. Bisa kita lihat, dengan penambahan satu jenis metode heuristik saja telah memberikan suatu hasil yang memiliki perbedaan sangat signifikan.

Sebagai pembanding, dilakukan percobaan terhadap 100 permainan dengan algoritma yang sama, skor antara pemain 1 dan pemain 2 sekitar 48:52, perbedaan yang tidak terlalu jauh.



Gambar 4.1.1 Perbandingan algoritma greedy dengan dan tanpa heuristik

4.2. Algoritma Minimax dengan dan tanpa Metode Heuristik

Pengujian selanjutnya dilakukan dengan "mengadu" algoritma minimax biasa, dengan algoritma minimax yang telah ditambahkan prinsip heuristik blocking point.

Sama seperti sebelumnya, pengujian dilakukan dengan 100 permainan. Dari permainan-permainan tersebut didapat data sebagai berikut. Algoritma minimax tanpa heuristik berhasil menang 39 kali, algoritma dengan heuristik berhasil menang 59 kali, dan didapat juga hasil seri sebanyak 2 kali.

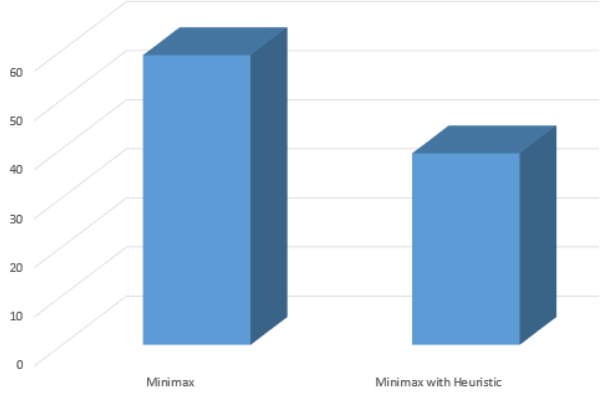
Sebagai pembanding dilakukan juga percobaan permainan dengan 2 algoritma yang sama, hasilnya adalah antara pemain 1 dan pemain 2 didapat perbandingan kemenangan adalah 49:51. Sekali lagi, penambahan prinsip sederhana berpengaruh besar pada hasil permainan.

Percobaan lainnya dilakukan dengan menghitung skor yang didapat dalam setiap permainan. Pengujian ini dilakukan dengan "mengadu" algoritma greedy dengan algoritma minimax biasa dilanjutkan dengan algoritma greedy melawan minimax dengan heuristik.

Pada percobaan antara algoritma greedy dengan algoritma minimax biasa, didapat hasil bahwa rata-rata

algoritma greedy akan kalah dengan hasil skor 1:3. Kemudian dilakukan percobaan antara algoritma greedy dengan algoritma minimax dengan heuristik, didapat bahwa rata-rata algoritma greedy akan kalah dengan perbandingan skor 1:5.

Dari hasil tersebut terlihat bahwa perolehan skor pada setiap permainannya meningkat drastis dengan digunakannya metode heuristik.



Gambar 4.1.1 Perbandingan algoritma greedy dengan dan tanpa heuristik

V. KESIMPULAN

Banyak algoritma yang dapat digunakan untuk membuat sebuah *Artificial Intelligence* untuk permainan reversi. Masing-masing algoritma juga memiliki kelebihan dan kelemahannya masing-masing. Kelebihan dan kelemahan ini bisa ditinjau dari beberapa aspek, misalnya kecepatan komputasi, kebutuhan memori, serta kemungkinan untuk menang.

Namun terlepas dari algoritma apa yang digunakan, ternyata penambahan metode heuristik dalam setiap algoritma dapat meningkatkan kemungkinan untuk menang dengan cukup signifikan. Sehingga pengunaan metode heuristik dalam *Artificial Intelligence* dari permainan reversi dianggap sangat efektif.

REFERENSI

[1] Munir, Rinaldi. 2009. ”Diktat Kuliah IF 2211 Strategi Algoritma”. Bandung:Program Studi Teknik Informatika STEI ITB.
[2] https://github.com/luugiathuy/ReversiGame, 17 Mei 2014.
[3] http://yabingilroi.blogspot.com/2013/04/reversi-othello.html, 17 Mei 2014.

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,19 Mei 2014

Gilang Julian S. (13512045)