Penerapan Algoritma A* dalam Pencarian Jalan untuk Permainan Lose Your Marble

Arnold Nugroho Sutanto

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10, Bandung
e-mail: hey_nold@hotmail.com

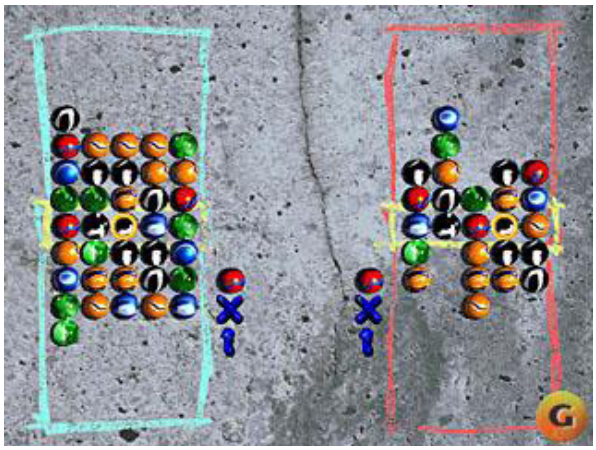
ABSTRAK

Makalah ini memaparkan tentang penerapan Algoritma A* sebagai algoritma pencarian jalan yang digunakan dalam suatu Intelegensia Buatan sederhana untuk permainan Lose Your Marble. Permainan Lose Your Marble ini, secara sederhananya, merupakan permainan yang mengharuskan setiap permain untuk berlomba memecahkan marble dengan cara menyesuaikan 3 atau lebih buah marble dengan warna yang sama, dengan batasan besarnya papan permainan dan jenis pergerakan pemain yang dapat dilakukan. Algoritma A* ini dipakai sebagai dasar dari algoritma pencarian cara tercepat untuk memecahkan marble-marble tersebut. Dalam ilmu komputer, Algoritma A* merupakan algoritma pencarian graf terbaik hingga saat ini untuk mencari jalan dengan biaya (*cost*) terkecil. Penerapan Algoritma pencarian A* pada Lose Your Marble ini sudah dicoba diimplementasikan dalam bahasa Java dan didapatkan Intelegensi Buatan yang cukup mangkus dalam menemukan cara tercepat bahkan lebih baik daripada Intelegensia Buatan yang dipakai dalam permainan aslinya. Hanya saja, memori dan waktu yang dibutuhkan cukup besar, jika dibandingkan dengan kesederhanaan permainan.

Kata kunci: Algoritma A*, Lose Your Marble, BFS, penentuan bobot.

1. PENDAHULUAN

Permaianan *Lose Your Marble* merupakan game PC yang cukup populer pada tahun 1997. Game ini memiliki alur permainan yang unik dan menarik sehingga berpotensi besar untuk dapat dipopulerkan lagi saat ini. Berikut adalah screen shotnya.

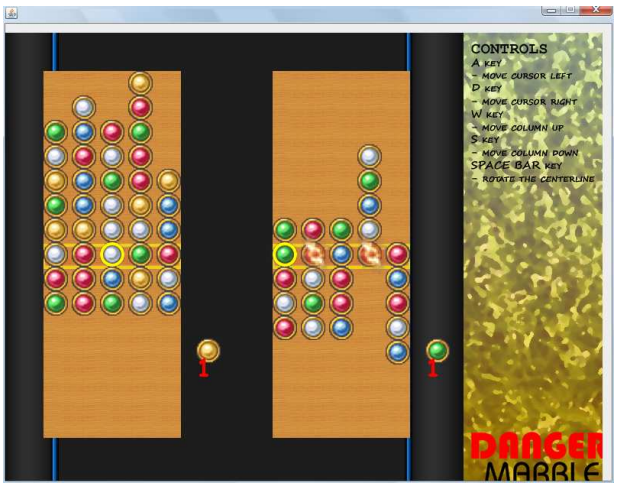


Gambar 1 Lose Your Marble Keluaran Tahun 1997

Satu hal yang dapat saya amati dari permainan ini adalah AI yang disediakan kurang efisien dalam melakukan pergerakan permainan. Bahkan, game ini menuai kritikan "lambat" karena kurang efisiennya AI yang ada. Memang, cara termudah untuk mengatasi hal ini adalah dengan mempercepat langkah AI (delay setiap langkah dikurangi). Tetapi tentu saja cara seperti ini tidaklah bagus karena akan mengurangi realitas AI yang berperan sebagai peran lawan pemain (tidak mungkin manusia dapat berpikir secepat itu). Oleh karena itu, satu cara yang paling efektif untuk mengurangi kelemahan AI ini adalah membuat AI baru yang melakukan langkah-langkah yang paling efisien.

Algoritma A* telah sering digunakan untuk pathfinding atau pencarian jalan suatu karakter dalam game. Algoritma ini juga dapat dipakai untuk pencarian langkah pada berbagai macam puzzle, salah satunya pada Lose Your Marble ini. Meskipun tidak dapat dibuktikan efisiensinya, pemakaian Algoritma A* yang telah diimplementasikan dalam permainan ini memberikan hasil yang lebih baik dari pada AI yang ada pada game aslinya yaitu dalam pengambilan langkah seminim mungkin untuk memecahkan bola-bola marble.

2. Lose Your Marble



Gambar 2 Danger Marble yang Merupakan Implementasi Lose Your Marble

Untuk memperjelas inti dari permainan Lose Your Marble, dapat dijabarkan aturan utama dari permainan ini:
- Tujuan permainan adalah menggeser deretan kelereng agar didapat satu baris kelereng berpola sama.
- Barisan kelereng yang dilihat kesamaan polanya adalah kelereng-kelereng yang terdapat dalam kotak kuning pada gambar di atas.
- Pergeseran kelereng dilakukan per kolom, kolom yang akan digeser ditandai dengan lingkaran berwarna kuning di bagian dalam kotak kuning.
- Setiap kali satu baris berhasil dicocokkan, kelereng akan hilang.

Kelereng yang hilang akan berimbas pada player musuh dengan memunculkan kelereng baru.

3. Algoritma Pencarian A*

A* sebenarnya merupakan algoritma pengembangan dari BFS (Breadth-First-Search) untuk menemukan jalan dengan biaya terkecil dari titik awal ke titik tujuan (bisa lebih dari 1 titik tujuan). Hanya saja, A* menggunakan fungsi heuristis (yang sering juga disebut f(x)) yang menentukan urutan titik mana yang akan dikunjungi terlebih dahulu. Fungsi heuristis ini sebenarnya menyimbolkan seberapa baik/mungkin titik itu dikunjungi untuk mencapai ke titik tujuan. Fungsi ini terdiri dari penjumlahan 2 fungsi :
Fungsi biaya jalan tersebut, yang merupakan biaya dari titik awal sampai titik sekarang (biasanya disebut dengan g(x))
Fungsi perkiraan jarak menuju titik tujuan dari titik yang sekarang (biasanya disebut h(x)). Sesuai namanya, h(x) hanyalah fungsi perkiraan yang dapat diterima.

3.1 Deskripsi Algoritma

Telah dijelaskan bahwa A* merupakan kerabat dari BFS. Oleh karena itu, A* memiliki sifat utama dari BFS yaitu memasukkan semua tetangga yang mungkin dari titik yang sekarang sedang dikunjungi ke dalam list yang akan dicek (disebut "open list). Perbedaanya dari BFS, A* tidak sembarangan memilih titik yang akan dikunjungi, melainkan ia akan memilih titik berdasarkan nilai f(x) nya. Untuk itu, open list selalu terurut berdasarkan nilai f(x)nya.
Dengan adanya seleksi pemilihan titik ini, algoritma A* memiliki efisiensi yang jauh lebih besar daripada BFS. Pemilihan yang dilakukan BFS membuat pengunjungan titik-titik lebih terfokus agar dapat mencapai tujuan yang diinginkan. Pemilihan titik-titik ini dan peninjauan semua tetangganya akan terus berlangsung hingga titik tujuanlah yang terpilih.

3.2 Kompleksitas Algoritma

Kompleksiatas waktu dari A* tergantung pada ketepatan fungsi heuristiknya. Pada kasus terburuk, banyaknya titik membesar secara eksponensial sesuai panjang jalan menuju solusi, tetapi A* bersifat polinomial ketika ruang pencariannya berbentuk pohon, yaitu terdapat 1 titik tujuan, dan fungsi heuristik h memenuhi kondisi ini :

$$| h(x) - h^*(x) | = O(\log h^*(x)) \qquad (1)$$

dimana *h** adalah heuristik optimal, atau cost pasti untuk menuju tujuan dari *x*. Dengan kata lain, kesalahan (*error*) dari *h* tidak boleh tumbuh lebih cepat dari logaritma "*perfect heuristic*" *h** yang mengembalikan jarak sebenarnya dari *x* menuju tujuan.

4. Penerapan Algoritma Pencarian A* dalam Permainan Lose Your Marble

Pencarian jalan cara yang dilakukan dalam AI Lose Your Marble, sebenarnya terbagi menjadi 2 yaitu pencarian jalan biasa yang diperuntukkan mendapatkan penghilangan marbel minimum 3 dan pencarian jalan spesial untuk mendapatkan penghilangan marbel sebanyak 5. Pencarian jalan biasa untuk penghilangan marbel minimum 3 inilah yang menggunakan Algoritma A* (Algoritma untuk pathfinding). Untuk pencarian jalan

spesial yang mengharuskan 5 marbel berpola sama beerurutan, saya belum sempat untuk menerapkannya.

4.1 Definisi Istilah

Untuk memahami algoritma pencarian jalan ini, ada baiknya kita perlu memahami beberapa istilah penting dalam Algoritma A* dengan hubungannya dalam penerpannya ini. Adapun istilah-istilah yang akan dibahas yaitu Path, Open List, Closed List, Nilai F, G dan H.
Path yang dimaksud di sini sama dengan istilah titik yang telah kita bahas sebelumnya. Dalam penerapannya, suatu path berarti suatu *state*/kondisi marble matriks (posisi-posisi marble yang teruabah) dan perintah yang diperlukan untuk menuju ke sana dari path sebelumnya. Nilai F, G, dan H juga disimpan dalam suatu path ini.

- ✓ *Open* list adalah list yang menyimpan kemungkinan path yang akan diperiksa. Open List dibuat terurut berdasarkan nilai F. Open list digunakan untuk menentukan secara selektif (berdasarkan nilai F) jalan yang dikira lebih dapat menuju pada path tujuan.
- ✓ *Closed* list adalah list yang menyimpan jalan yang sudah diperiksa dari open list. Kedua list (open list dan closed list) ini bertujuan juga untuk menghindari penelusuran berkali-kali jalan-jalan yang memang sudah diidentifikasi agar tidak usah masuk lagi ke dalam open list.
- ✓ Nilai F adalah cost perkiraan suatu path yang teridentifikasi. Nilai F merupakan hasil dari f(x). Misal suatu path yang memiliki marble matriks dengan marble yang diinginkan untuk dihilangkan berada jauh pada garis tengah sehingga diperkirakan memerlukan banyak langkah akan memiliki nilai F yang lebih besar jika dibandingkan dengan path (dengan jumlah langkah yang sama dengan path sebelumnya) yang memiliki marble-marble tujuan yang dekat dengan garis tengah. Nilai F, sesuai dengan fungsi f(x), ini diperoleh dengan penjumlahan nilai G dan H.
- ✓ G, hasil dari fungsi g(x), adalah banyaknya langkah yang diperlukan untuk menuju ke path sekarang (marble matriks yang sudah diubah sedemikian rupa) dari matriks marble awalnya. Banyaknya langkah yang dimaksud di sini dapat disamakan dengan banyaknya seorang player harus menekan keyboard entah untuk menggeser bola atau memindahkan kursor. Nilai G ini akan mempengaruhi apakah sebuah path layak untuk dimasukkan ke open list tergantung dari batas maksimum langkah yang diinginkan.
- ✓ H adalah harga yang diperkirakan dibutuhkan (dari matriks sekarang) untuk menuju ke marble matriks tujuan dari jalan yang ditempuh sekarang. Nilai ini didapatkan dari ketersediaan 3 marbel dengan warna yang dicari pada kolom yang berdekatan dan seberapa dekat marbel tersebut dengan garis tengah. Semakin tidak ada marbel yang berdekatan dan semakin jauh marbel tersebut maka semakin tinggi nilai H nya.

4.2 Definisi Fungsi h(x)

Fungsi h(x), sebagai penghasil nilai H, didefinisikan sebagai berikut

$$h(x) = 25 - x*5 \quad (2)$$

dengan x adalah jumlah marble yang bersebelahan di daerah tengah dengan pola yang diinginkan

Nilai 25 ini didapat dari jumlah marbel yang bersebelahan di garis tengah yang maksimum sebanyak 5 sedangkan setiap marbel di garis tengah yang bersebelahan diberi nilai 5 (5 buah marbel bersebelahan * 5 nilai per marbel = 25). Maksud dari nilai 5 ini adalah saya memprediksikan membutuhkan rata-rata 5 langkah untuk membawa sebuah marbel ke garis tengah.

Penentuan h(x) ini bisa dibilang sangat sederhana dan memungkinkan besarnya perkiraan yang error. Oleh karena hal ini, kompleksitas waktu dan memori dari penerapan algoritma A* yang saya implementasikan tergolong terlalu boros.

4.3 Algorima Penerapan A*

Dengan mengetahui istilah-istilah ini, kita akan dapat mengerti Algoritma A* yang diterapkan dalam permainan ini. Hal yang perlu diperhatikan, dalam melakukan pencarian jalan ini kita dibatasi oleh maksimum langkah yang diinginkan (agar efisien), sehingga setiap path yang memiliki nilai G lebih besar dari pada ini tidak terkualifikasi. Algoritma penerapan A* ini merupakan suatu prosedur untuk menemukan jalan memecahkan marbel-marbel dengan masukkan pola/jenis/warna marbel tersebut.

Algoritma penerapan A* yang diimplementasikan kurang lebih dapat dijelaskan sebagai berikut

1) Tambahkan jalan awal yang terdiri atas Marble Matriks awal dan null command (tidak ada perintah yang perlu dilakukan untuk menuju ke posisi awal)
2) Lakukan pengulangan berikut:
   a) Cari jalan yang bernilai F paling kecil di open list.
   b) Jadikan path tersebut ke closed list, dan keluarkan dari open list.
   c) Untuk setiap path munculkan semua kemungkinan path yang sekiranya membantu untuk mendapatkan tujuan. Setiap path yang

saya pikirkan dapat mencapai tujuan adalah sebagai berikut:

- Menggeser semua marbel dengan warna yang diinginkan (warna masukan) dan yang terdekat dengan garis tengah, yaitu pada atas garis tengah maupun pada bawah garis tengah, dari setiap kolom ke garis tengah.
- Merotasi marbel
- Menurunkan setiap marbel yang berada pada garis tengah.

Setiap path ini kemudian akan dicek apakah dapat dilakukan (jika dalam menggeser bola dan ternyata stuck karena melebihi batas bola maka path tersebut tidak dapat dilakukan). Jika path tersebut tidak dapat dilakukan atau nilai G nya lebih besar dari jumlah langkah yang diharapkan, path itu dilewati. Tetapi jika bisa, path itu akan melewati beberapa tahap seleksi yaitu :

- Jika path tidak terdapat pada open list, tambahkan ke open list dan jadikan path yang sekarang parent dari kotak yang sekarang. Hitung dan simpan nilai F,G dan H nya.
- Jika path terdapat pada closed list, lewati path ini.
- Jika path ini sudah ada pada open list, periksa apakah path sekarang dalam menuju goal ini lebih baik menggunakan nilai G (nilai G yang lebih kecil, lebih baik). Dan jika benar, maka rekalkulasi nilai G dan F nya.

d) Berhenti jika:

- Terdapat path yang memiliki adjective marble dengan warna yang sama lebih banyak dari 3.
- Open list kosong, yang berarti gagal untuk mencari target tujuan.

Kita lihat di sini bahwa algoritma ini berhenti ketika terdapat path yang memiliki adjective marble dengan warna yang sama lebih banyak dari 3 entah itu warna marbel yang dimaksud atau bukan. Hal ini dilakukan karena marble matriks akan berubah saat hal ini terjadi, sehingga meskipun kita meneruskan pencarian jalan untuk warna yang dimaksud akan pasti menghasilkan kekeliruan karena acuan marble matriksnya salah.

Setelah berhenti, untuk mendapatkan perintah-perintah yang diperlukan, kita menelusuri dari belakang (path terakhir) sampai kita menemukan path paling awal berdasarkan parentnya. Untuk itu setiap path perlu ditambahkan atribut ID dia sendiri dan ID orang tua dengan ID yang bersifat uniq (tidak ada yang sama).

5. Kesimpulan

Penggunaan algoritma A* untuk pencarian jalan pada permainan Lose Your Marble memberikan hasil pencarian jalan yang cukup efisien, dalam artian memberikan langkah tercepat untuk memecahkan marble. Meskipun begitu, dalam khasus tertentu algoritma ini tidak selalu memberikan jalan yang paling cepat. Hal ini dikarenakan fungsi heuristik (h(x)) untuk perkiraan bobot ke titik tujuan yang dibuat kurang bagus sehingga masih terdapat error yang cukup besar dengan bobot seharusnya. Oleh karena fungsi heuristik ini yang kurang baik, kompleksitas waktu juga tidak begitu baik. Memori yang digunakan juga cukup besar karena pada tiap titik simpul berisi matriks marblenya. Jadi, penerapan Algoritma A* ini dapat diperbaiki dengan memperbaiki fungsi heuristiknya (h(x)) sehingga memperkecil kemungkinan error.

REFERENSI

1. Munir, Rinaldi. 2006. Diktat Kuliah IF 2251 Strategi Algoritmik. Bandung: Laboratorium Ilmu dan Rekayasa Komputasi. Hal. 108, 120. Program Studi Teknik Informatika ITB.
2. http://en.wikipedia.org/wiki/A%2A_search_algorithm Diakses tanggal 2 Januari 2009 ~ 12.05
3. http://www.policyalmanac.org/games/aStarTutorial.htm Diakses tanggal 2 Januari 2009 ~ 12.25