Penerapan Algoritma BestFast dalam Pemodelan Permainan Rubik's Cube

Apri Kamayudi

Program Studi Informatika, STEI ITB, Bandung
Jl.Taman Malaka Utara D16 no.13 Jakarta Timur.
E-mail : if15009@students.if.itb.ac.id

Abstraksi

Rubik's cube merupakan permainan puzzle mekanis yang berbentuk kubus terdiri dari 27 kotak kecil dan memiliki 6 warna pada kubus utama. Posisi setiap kotak kecil dapat diubah dengan merotasikannya sesuai sumbu. Dengan perhitungan matematis didapatkan 4x 10^{12} kemungkinan posisi yang dihasilkan puzzle ini. Besarnya ruang solusi akibat banyaknya kemungkinan tersebut memerlukan algoritma yang mangkus untuk memberikan solusi secara cepat. Algoritma BestFast yang merupakan perbaikan dari algoritma A* BestFirst mampu menjawab masalah yang ada. BestFast menggunakan tabel profil sebagai penyimpan solusi hasil penjelajahan setiap kemungkinan pada ruang solusi.

Kata kunci *: tabel profil, goal distances, change-over distance, heuristik, rubik's cube, ruang status*

1. Pendahuluan

Rubik's Cube adalah puzzle mekanik yang ditemukan pada tahun 1974 oleh pemahat dan profesor arstitekur Hungarian, Erno Rubik. Penemunya sendiri menamkannya Magic Cube. Pada tahun 1976, temuan tersebut dijual di Inggris dengan nama Riubik's cube. Rubik's cube memenangkan penghargaan khusus Game of the Year di German untuk kategori puzzle terbaik pada tahun 1980.

Rubik's cube memiliki 9 potongan kotak kecil pada setiap sisinya, dan terdiri dari 27 kubus kecil. Umumnya, setiap sisi dari kubus ini memiliki 6 warna berbeda. Saat puzzle ini dipecahkan maka akan diperoleh warna yang sama pada masing-masing sisi.

2. Ruang Lingkup

Kubus standar memiliki ukuran rusuk 5.7 cm. Puzzle memiliki 26 miniatur kubus pada permukaannya. Kubus pusat dari setiap permukaannya adalah kotak utama yang posisinya sudah tetap sebagai mekanisme pusat. Dengan struktur tersebut, kubus-kubus kecil lainnya dapat diputar. Tersisa 21 potongan kubus yang dapat diubah posisinya sedangkan 6 lainnya sebagai sumbu utama yang tetap tetapi tetap dapat dirotasikan.

Rubik's cube normal (3x3x3) memiliki $(8! \times 3^{8-1}) \times (12! \times 2^{12-1})/2 =$ 43,252,003,274,489,856,000 posisi berbeda (permutasi). Berdasarkan besarnya jmlah posisi tersebut, semua kubus dapat dipecahkan dengan 27 langkah atau kurang. Jika setiap permutasi dicoba 1 per satu maka akan membutuhkan waktu 261 tahun cahaya. Dan apabila permukaan kubus dari setiap permutasinya dibentangkan maka akan sebanding dengan 256 kali luas bumi

Faktanya terdapat $(8! \times 3^8) \times (12! \times 2^{12}) =$ 519,024,039,293,878,272,000 kemungkinan penyusunan dari setiap potongan untuk membangun kubus ini, tapi 1 diantara 12 yang dapat dicapai. Hal ini dikarenakan tidak ada langkah yang dilakukan secara berurutan sehingga dapat menukar sepasang atau merotasikan potongan yang terdapat di pojok.

Rubik's cube dapat direpresentasikan sebagai mesin status terhingga dan menerima bahasa seperti mesin lainnya. Bahasa yang diterima tersusun atas string setiap langkah. Walau begitu, banyaknya status yang menjadi asumsi mesin ini sangatlah besar. Hal ini menyebabkan proses pencarian ruang statusnya sangat sulit dilakukan kecuali jika sebagian besar kemungkinan dihilangkan pada saat pengecekan.

Mesin ini dapat dimodelkan dengan fokus kepada kotak-kotak kecil yang membangun koyal besar. Kotak-kotak kecil tersebut juga merupakan mesin status berhingga yang menerima bahasa tertentu. Semua irisan dari bahasa yang diterima oleh setipa kotak kecil adalah bahasa yang diterima oleh kotak besar.

3. Pemodelan Rubik's Cube

Terdapat 4 tipe dasar potongan kubus yang berbeda. Kubus pojok, kubus rusuk, kubus tengah (pada setiap permukaan), dan kubus pusat yang tidak terlihat dari luar.

Sebagai konvensi, kubus tengah memiliki posisi yang tetap (walau dapat dirotasikan). Hal ini dimaksudkan bahwa setiap potongan kubus lainnya tetap dapat dipindahkan tanpa harus ikut memindahkan kubus tengah. Faktanya, kubus tengah selalu sama bila direlasikan dengan sesamanya.

Perhatikan setiap kubus tengah yang tetap dapat disimbolkan sebagai arah. Sebagai contoh, kubus hijau yang berarah depan.

warna	**huruf**	**arah**
Green (hijau)	g	Front (depan)
White (putih)	w	Top (atas)
Orange (Oranye)	o	Left (kiri)
Yellow (Kuning)	y	Hidden* (belakang)
Blue (Biru)	b	Bottom (bawah)
Red (Merah)	r	Right (kanan)

*arah "hidden" adalah bagian yang tidak tampak saat kita melihat kubus dari depan

Setiap potongan kubus memiliki 6 sisi. Jika kita memindahkan potongan kubus yang tersedia dari keseluruhan kubus (), kita dapat menemukan 6 sisi kubus dan mengidentifikasikan warna menunjukkan arah apa. Informasi ini dapat digunakan untuk mendefinisikan status dari sebuah potongan kubus dengan contoh sebagai berikut.

mc(Front,Top,Left,Hidden,Bot,Right) atau mc(F,T,L,H,B,R)

dimana setiap argumen memiliki arti: green (g), white (w), orange (o), yellow(y), blue (b), red (r), none (n).

Misalkan instansiasi dari potongan kubus pertama sebagai **mc(n,n,o,y,b,n).** Hal ini berarti bawah permukaan yang menghadap depan dianggap tidak berwarna (atau “none”), bagian atas juga “none”, bagian kiri berwarna oranye, bagian “hidden” berwarna kuning, bagian bawah berwarna biru, bagian kanan adalah “none”. (inistansiasi ini menghasilkan potongan kubus pertama sebagai posisi yang berhasil ditemukan)

Hasilnya dari semua ini adalah kita tidak perlu mendeskripsikan seluruh status dengan warna pada setiap permukaannya: kita dapat memecahkan potongan kubus sebagai kubus kecil untuk kemudian dicatat orientasinya.

Sebagai contoh , potongan kubus pertama dapat diposisikan dengan warna oranye sebagai bagian depan. Tetapi kita juga harus mengetahui orientasinya intuk mengetahui statusnya. Pada saat sisi oranye berada di depan, sisi kuning akan menghadap atas, kanan, bawh, atau kiri. Dengan kata lain potongan kubus memiliki total status 24 buah.

mc(n,b,y,o,n,n)

mc(n,y,n,o,n,b)

mc(n,n,n,o,b,y)

mc(n,n,b,o,y,n)

mc(b,n,n,n,o,y)

mc(b,y,n,n,n,o)

mc(b,o,y,n,n,n)

mc(b,n,o,n,y,n)

mc(y,n,b,n,o,n)

mc(y,b,o,n,n,n)

mc(y,o,n,n,n,b)

mc(y,n,n,n,b,o)

mc(o,b,n,n,n,y)

mc(o,y,b,n,n,n)

mc(o,n,y,n,b,n)

mc(o,n,n,n,y,b)

mc(n,n,y,b,o,n)

mc(n,y,o,b,n,n)

mc(n,o,n,b,n,y)

mc(n,n,n,b,y,o)

mc(n,n,n,y,o,b)

mc(n,b,n,y,n,o)

mc(n,o,b,y,n,n)

mc(n,n,o,y,b,n)

Representasi dari status-status dari setiap potongan kubus dapat dimodelkan jika kita men-assign sebuah huruf ke setiap status. Kemudian potongan kubus dapat ditrepresentasikan dengan huruf-huruf “a” sampai “x”. Hal ini harus dicatat sebagai metoda berbeda yang memiliki beberapa kelebihan dan kekurangan.Setiap tahapan yang ditemukan mengizinkan setiap potongan kubus untuk menggunakan skema dasar yang sama dimana status “a” untuk potongan kubus memiliki kesamaan dengan status “a” untuk potongan kubus yang berbeda.

Status dari keseluruhan kubus dapat direpresentasikan sebagai berikut:

st(MC1,MC2,MC3,MC4,MC5,MC6,MC7,MC8,MC9,MC10,MC11,MC12,MC13,MC14,MC15,MC16,MC17,MC18,MC19,MC20)

dimana MC1 yang melalui MC20 adalah status-status dari potongan kubus pertama sampai yang ke-20.

Variabel "MC" diatas hanya digunakan untuk menandai bahwa setiap variabel memiliki posisi tertentu pada 20 tupel relasi yang disebut sebagai status. Pada penerapannya, kita harus menandai status-status variabel dari potongan kubus dengan warnanya masing-masing:

st(OYB,YB,YBR,OY,YR,WOY,WY,WYR,OB,BR,
WO,WR,GOB,GB,GBR,GO,GR,WO,GW,GWR)

untuk mempermudah, pandang "A" sebagai status dari potongan kubus pertama, "B" sebagai potongan kedua, dan seterusnya

st(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T)

Jika kita periksa topografi kubus, kita akan menyetujui bahwa pada setiap posisi yang dipecahkan, potongan kubus pertama berwarna orangye, biru dan kuning, sedangkan potongan kedua hanya bitu dan kuning. Tidak ada lagi potongan kubus yang memiliki warna yang sama.

00

Sebagai contoh status dari keseluruhan kubus adalah sebagai berikut:

st(a,a,c,b,c,b,g,h,e,d,l,p,l,j,f,v,f,v,w,o)
dimana potongan kubus pertama berada pada status "a" dan potongan terakhir berada pada status "o".

Pada saat kita menemukan kubus yang belum terpecahkan, kita dapat menempatkan semua potongan pada baris yang berorientasi pada arah sehingga menjadi kubus awal secara keselituhan dan menyelesaikan puzzle dengan mencari paket langkah-langkah yang menyebabkan setiap potongan sesuai dengan orientasi.Analogi ini sesuai dengan penggunakan tahapan pada program Rubik. Dengan asumsi bahwa kita telah mempelajari setiap langkah memiliki efek apa pada status apa, kita tidak perlu lagi memperhatikan keseluruhan kubus. Biasanya kita hanya perlu memperhatikan orientasi dari potong kubus secara terpisah.

Kita akan menyadari bahwa efek dari setiap langkah yang akan membentuk potongan kubus bergantung pada orientasinya dan tipe potongan kubus tersebut. Pada orientasi tertentu, langkah aopaun tidak akan berpengaruh.

Untuk mengetahui efek dari setiap langkah, kita perlu mengenali langkah apa saja yang ada. Terdapat 12 langkah. Terdapat 6 permukaan yang masung-masingnya dapat diputar 90 derajat ke kiri (berlawanan arah ajrum jam) atau ke kanan (searah jarum jam). Kita akan menanmai setiap permukaan dari keseluruhan kubus berdasarkan warna dari kotak tetapnya. Langkah-langkah yang dapat kita lakukan diantaranya reen-right (gr), green-left (gl), white-right (wr), white-left (wl), orange-right (or), orange-left (ol), yellow-right (yr), yellow-left (yl), blue-right (br), blue-left (bl), red-right (rr), dan red-left (rl). Tentunta, kita dapat menggerakkan setap permuakan sebesar 180 derajat atau kelipatan dari 90.

Tandai kotak berwarna hijau yang tidak dapat bergeser untuk selalu menghadap ke depan, langkah green-kanan seharusnya dapat dilakukan dengan mudah dengan memperhatikan langkah depan-kanan. Hal ini akan berubah jika potongan kubus tidak terpengaruh oleh langkah green-right (atau left) kecuali jika sisi yang mengahadap depan berwarna. Seperti halnya, potongan kubus tidak akan dipengaruhi oleh langkah putih kecuali jika bagian atas berwarna. Piola yang sama akan benar untuk setiap langkah.

Dengan mencoba, kita akan mendapatkan langkah green-right tidak akan mempengaruhi bagian depan maupun belakang permukaan dari potongan kubus manapun. Jika potongan kubus dipengaruhi oleh setiap langkah yang dilakukan terhadapnya, hanya bagian atas, kanan, bawa dan kiri saja yang akan berubah. Pada kasus ini, bagian atas akan mengambil warna dari bagian kiri, bagian kanan akan mengambil warna dari bagian atas, bagian bawah akan mengambil watna dari bagian kiri, dan bagian kiri akan mengambil warna dari bagian bawah. Paket aturanyang sama dapat menjadi status untuk setiap langkah

Kita dapat mengunakan informasi untuk membangun tabel yang memberitahukan status potongan kubus yang berubah ketika digerakkan. Paket dari semua langkah yang menybabkan semua potongan-kubus berubah menjadi statis terpecahkannya adalah solusi yang dibutuhkan untuk menyelesaikan puzzle Rubik.

Tujuan akhir dari masing-masing potongan kubus adalah untuk ditempatkan sesuai orientasinya (posisi terpecahkan). Sebagai contoh, potongan kubus pertama tidak akan sesuai jika warna kuninf mengahadap kiri. Jika warna oranye menghadap kiri, maka bagian kuning akan tersembunyi dan bagian biru menghadap bawah. Hal ini dapat diekspresikan sebagai mc(_,_,o,y,b,_) pada relasi di bawah ini.

4. Algoritma BestFast

Algoritma Bestfast adalah algoritma A* bertipe BestFirst yang dapat digunakan untuk menyelesaikan puzzle Rubik's cube. Walau begitu, Bestfast dapat memecahkan masalah lebih cepat dibandingkan dengan BestFirst itu sendiri. Hal tersebut dikarenakan BestFast menggunakan struktur data pohon a-b untuk sebagai pohon pembentuk simpul-simpul yang membangkitkan nilai-nilai heuristik. Dengan begitu, BestFast memiliki keunggulan dalam menyelesaikan persoalan yang besar dan rumit.

Perbedaan lainnya, adalah penggunaan (tabel profil) pada algoritma heuristik. Tabel profil adalah representasi relasi antara status-status yang memiliki karakteristik tertentu dengan jumlah langkah yang dibentuk pada saat menyelesaikan posisi (dari Rubik's cube) tertentu. Jumlah langkah tersebut dapat dianggap sebagai jarak yang dituju atau ongkos. Seperti kebanyakan algoritma heuristik lainnya, profil tidaklah selalu benar. Bagaimanapun juga, *goal distance* terpendek yang dibangkitkan selalu menghasilkan perkiraan dari jumlah langkah yang diperlukan untuk menyelesaikan Rubik's cube. Ketika nilai pada tabel profil tidak ditemukan, maka digunakan heuristik konvensional. Sayangnya, heuristik konvensional menghabiskan banyak waktu untuk memperkirakan *goal distance*. Kelebihan dari profil yang adalah hasil perkiraan yang lebih baik dan perolehan solusi yang lebih cepat ketika tabel muali menjelajahi seriap ruang status.

Tabel profil mempunyai batasan ruang status. Hal ini dibuktikan dengan *goal distance* terbesar yang tercantum pada tabel. Pada saat inilah heuristik konvensional harus digunakan, oleh karenanya hal tersebut dikenal sebagai *"change-over distance"*.

Untuk membangkitkan tabel profil, diperlukan nilai-nilai yang hanya dihasilkan pada saat nilai tersebut berasosiasi dengan status yang diketahui untuk menghasilkan *goal distance*. Untuk setiap *goal distance* yang tercantum pada tabel, satu atau lebih profil harus dibangkitkan. Profil yang paling optimal adalah profil yang memiliki korespondensi satu-ke-banyak pada setiap status yang ditandai dengan banyak langkah pada satus yang berhasil diselesaikan. Tetapi hal ini tidak dapat menjamin solusi terpendek (banyak langkah) pada setiap masalah. Nilai yang dibangkitkan mungkin memiliki asosiasi yang tidak tepat dengan status-status pada *goal distance*. Walau begitu paling tidak terdapat profil yang benar untuk setiap status pada *change-over distance.* Suatu profil dinyatakan benar apabila *goal distance* yang ada cocok dengan status yang berasosiasi dengannya.

Untuk menciptakan tabel profil dengan lebih cepat, digunakan nilai profil sebagai kunci. Untuk membuat tabel profil lebih akurat (serta pencarian yang lebih cepat), maka dilakukan penjelajahan dinamis untuk memeriksa ketepatan dari setiap asumsi yang dibangkitkan oleh tabel profil. Dengan cara tersebut , maka setiap asumsi dapat dibuktikan salah jika suksesir dari profil status tidak konsisten dengan asumsi yang telah diperoleh oleh predesornya. Asumsi baru dapat dibuat dan informasi baru dapat dipropagasi melalui pohon pencari. Walau perunutan dinamis tidak diimplementasikan pada saat itu juga oleh program BestFast, program telah dibuat untuk mengakomodasikan algoritma tersebut. Pada saat itu, pengakomodasian akan memperlambat eksekusi. Bagaimanapun juga, penerapan perunutan profil secara sepenuhnya dapat meningkatkan eksekusi.

Pada Rubik's cube, solusi-solusi akan memiliki batasan waktu dan memori sampai dengan 9 langkah pada mesin yang lambat. Solusi terpendek (pencarian A*) dapat dipastikan dengan menggunakan parameter berikut. Batasan kedalaman akan meningkat dengan menggunakan tabel profil yang lebih besar dan penerapan perunutan profil. Karena program dibuat dengan Arity Prolog versi 16 bi, penggunaan versi 32 bit dapat meningkatkan kedalaman perunutan. Penggunaan supercomputer dalam membangkitkan tabel profil akan memberikan pengaruh yang sangat besar. Data yang dihasilkan akan sangat ekonomis karena ukuran tabel profil yang sedang dapat mencakup ruang status yang besar.

5. Kesimpulan

Untuk meyelesaikan puzzle Rubik's cube, setiap potongan kubus harus didefinisikan statusnya. Setiap ruang status yang ada akan dijelajahi oleh Algoritma pencarian BestFast yang akan membangkitkan solusi pada tabel profil dengan langkah terpendek.

Daftar Pustaka

[1] Miller, David Lee Winston. Solving Rubik's Cube Using Bestfast Algorithm and Profile Tables. State University of New York Institute of Technology at Utica/Rome, 1998

[2] http://en.wikipedia.org diakses pada tanggal 21 Mei 2007 jam 10.00

[3] http://fang.sunyit.edu/Faculty/novillo.html diakses pada tanggal 21 Mei 2007 jam 10.00