Penggunaan Algoritma *Branch and Bound* dan Program Dinamis Dalam Pemecahan Masalah *Rubik's Cube*

Ibnul Qoyyim[1], Arief Pratama[2], Raden Tomi Akhmad Fadlan[3]

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14066@students.if.itb.ac.id[1], if14070@students.if.itb.ac.id[2], if14094@students.if.itb.ac.id[3]

Abstrak

Persoalan Rubik's Cube mengimplementasikan beberapa algoritma, seperti brute force, greedy, DFS, BFS, Branch and Bound, dan sebagainya. Dalam makalah ini, implementasi yang digunakan adalah gabungan algoritma Branch and Bound dan program dinamis. Algoritma Branch and Bound ini digunakan dalam mencari langkah terpendek menyelesaikan sebuah sisi sedangkan program dinamis digunakan dalam mencari sisi yang diselesaikan terlebih dahulu. Persoalan dipermudah dengan dua metode, yaitu atas-tengah-bawah dan atas-bawah-tengah yang menunjukkan urutan sisi yang akan diselesaikan relatif terhadap sisi yang dipilih pertama kali.

***Kata kunci**: Rubik's cube, Branch and Bound, Program Dinamis.*

1. Pendahuluan

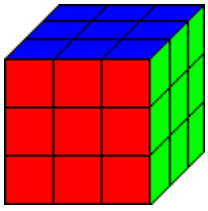
Terdapat banyak persoalan kombinatorial dalam sains komputer, misalnya N-Puzzle, Maze, Pentomino Puzzle, dan sejenisnya. Persoalan-persoalan ini mengimplementasikan bermacam-macam algoritma kombinatorial seperti DFS, BFS, dan program dinamis.

Rubik's Cube merupakan satu dari sekian persoalan tersebut. Persoalan ini cukup dikenal karena kerumitannya. Rubik's Cube ditemukan oleh Erno Rubik, seorang profesor desain interior berkewarganegaraan Hungaria.

Permainan *Rubik's Cube* sudah lama dikenal masyarakat. Meskipun begitu, banyak orang yang belum dapat menyelesaikan persoalan tersebut. Orang yang telah menyelesaikan persoalan tersebut pun, umumnya tidak menggunakan langkah terpendek untuk menyelesaikannya.

2. Persoalan

Rubik's Cube adalah persoalan dimana sebuah kubus yang setiap sisinya terdiri dari 9 upakubus yang lebih kecil dan harus diselesaikan dari keadaan acak menjadi bentuk yang tersusun. Bentuk tersusun adalah keadaan dimana setiap sisi kubus memiliki warna yang sama. Masalah Rubik's Cube merupakan masalah yang cukup rumit dimana hanya ada 1 konfigurasi yang benar dari 43 *quintillion* konfigurasi. Bila persoalan ini dipecahkan menggunakan brute force, akan dibutuhkan waktu yang sangat lama.



Rubik's Cube yang bentuknya tersusun.

Pada bentuk tersusun (tidak teracak), tiap sisinya mempunyai satu warna yang sama. Rubik's cube dapat diputar pada setiap sisinya, baik horizontal maupun vertikal, dengan arah putaran bebas (searah jarum jam atau berlawanan arah jarum jam sejauh 0°, 90°, 180°, atau 270°). Setelah sebuah sisi diputar, keadaan Rubik's Cube berubah.

Susunan upakubus-upakubus pada sisi-sisi selain sisi yang diputar dan sisi diseberangnya, akan berubah sehingga terdapat 4 buah sisi yang susunan upakubus-upakubusnya terdiri dari 6 buah upakubus berwarna tetap (seperti sebelumnya) dan 3 buah upakubus yang berbeda warnanya. Apabila hal tersebut dilakukan secara acak dan berulang-ulang pada semua sisi, maka akan didapatkan susunan kubus yang sulit diselesaikan. Menurut Erno Rubik, apabila seseorang memutar kubus secara acak untuk mengembalikan ke keadaan semula (bentuk tersusun), maka waktu yang diperlukan diperkirakan melebihi masa hidupnya.

3. Penyelesaian Masalah Rubik's cube

Penyelesaian masalah Rubik's Cube ini tidak mungkin menggunakan *brute force* murni karena hanya ada 1 konfigurasi yang benar dibandingkan dengan banyak konfigurasi yang salah sehingga membutuhkan waktu yang sangat lama. Oleh karena itu, penulis menyelesaikan masalah Rubik's Cube ini dengan cara menyelesaikan satu per satu sisi dari kubus. Untuk memilih sisi yang akan diselesaikan digunakan program dinamis sementara untuk mendapatkan jumlah langkah terpendek untuk menyelesaikan suatu sisi tanpa mengubah sisi yang telah terselesaikan digunakan *Branch and Bound*. Dengan demikian, hasil dari perhitungan keseluruhan diharapkan merupakan langkah optimum.

Ada 2 cara penyelesaian Rubik's Cube yang pada umumnya digunakan pemula:

1. Penyelesaian dengan menyelesaikan lapisan atas terlebih dahulu, kemudian lapisan bawah, terakhir lapisan tengah.
2. Penyelesaian dengan urut dari atas ke bawah: mula mula lapisan atas, tengah, kemudian bawah

3.1. Metode Atas-Bawah-Tengah

Algoritma ini dilakukan dengan cara:

1. Susun sisi bagian atas sehingga memiliki warna sama dan sesuai.
2. Susun sisi bagian bawah sehingga memiliki warna sama dan sesuai tanpa mengubah keteraturan sisi bagian atas.
3. Susun lapisan bagian tengah tanpa mengubah keteraturan sisi bagian atas maupun bawah.

Pemilihan sisi mana yang atas dan bawah dilakukan dengan program dinamis, begitu juga dengan sisi berikutnya. Kemudian untuk menentukan banyak langkah untuk menyelesaikan setiap sisi digunakan *branch and bound.*

3.2. Metode Atas-Tengah-Bawah

Algoritma dengan cara ini adalah :

1. Susun sisi bagian atas sehingga memiliki warna sama dan sesuai.
2. Susun lapisan bagian tengah tanpa mengubah keteraturan sisi bagian atas.
3. Susun sisi bagian bawah sehingga memiliki warna sama dan sesuai tanpa mengubah keteraturan sisi bagian atas maupun lapisan tengah.

Cara pemilihan sisi dan penentukan langkah penyelesaian sama dengan metode sebelumnya.

3.3. Penyelesaian dengan Branch and Bound

Cara Penyelesaian dengan branch and bound akan bergantung pada sisi yang akan diselesaikan, apakah sisi pertama, sisi kedua, atau sisi setelahnya.

3.3.1. Penyelesaian Branch and Bound untuk Sisi Pertama

Algoritma untuk penyelesaian sisi pertama dengan cabang dari pohon tingkat pertama adalah memilih bagian sisi dari cube yang masih acak untuk digunakan sebagai tempat penggabungan tile yang berwarna sama (warna tile ditentukan oleh program dinamis). Cabang-cabang berikutnya menentukan tile yang akan dipindah menuju posisi goal (tempat seharusnya). Fungsi adalah jumlah langkah yang dibutuhkan untuk memindahkan tile ke posisi tersebut (memutar 90 maupun 180 dianggap sama), sementara ongkos adalah jumlah tile dalam bagian sisi tersebut yang belum sesuai dengan posisi goal.

3.3.2. Penyelesaian Branch and Bound untuk Sisi Bawah Puzzle

Algoritma untuk menyelesaikan sisi bawah adalah dengan menggerakkan sisi bawah yang bersesuaian dengan menggunakan langkah langkah yang diketahui tidak akan mengubah sisi atas.

3.3.3. Penyelesaian Branch and Bound untuk Lapisan Tengah

Algoritma untuk menyelesaikan sisi bawah adalah dengan menggerakkan lapisan tengah sehingga bersesuaian dengan menggunakan langkah-langkah yang diketahui tidak akan mengubah sisi lapisan bawah maupun atas.

Fungsi heuristik untuk menghitung taksiran cost:

$C(i) = f(i) + g(i)$

C(i) = ongkos untuk simpul i

F(i) = ongkos mencapai simpul i dari akar = jumlah langkah yang diperlukan untuk mencapai konfigurasi di simpul i

G(i) = ongkos mencapai simpul tujuan dari simpul i = jumlah ubin yang belum sesuai dengan keadaan lengkap untuk sisi tersebut.

3.4. Pemakaian Program

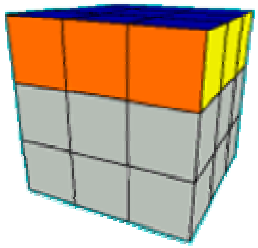
Program dinamis digunakan untuk memilih salah satu sisi sebagai sisi atas. Kemudian memilih berikutnya yang akan diselesaikan dan seterusnya. Ongkos dari program dinamis didapat dari hasil branch and bound pada penyelesaian sisi yang bersangkutan.

4. Analisis

Kedua penyelesaian sebenarnya hanya berbeda pada urutan lapisan kubus yang harus disesuaikan. Untuk membandingkan kedua jenis penyelesaian ini, tolok ukurnya adalah jumlah langkah yang diambil dalam penyelesaian. Jumlah langkah yang diambil mencerminkan keefektifan penyelesaian. Berikut ini adalah contoh penggunaan 2 cara :

1. Penyelesaian Atas-Bawah-Tengah

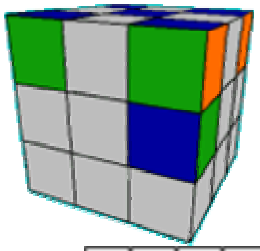
Mula mula buat sisi bagian atas agar sesuai dengan konfigurasi yang diinginkan sehingga hasil akhirnya seperti berikut :



(catatan : *warna bervariasi tergantung dari warna yang dipilih, warna abu abu menyatakan bahwa warna tersebut tidak diperdulikan.*)

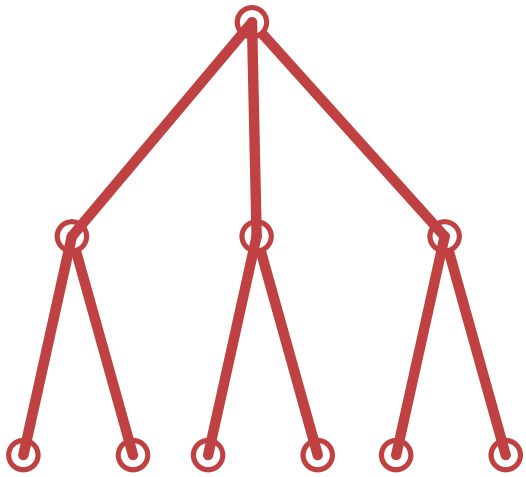
Gunakan branch and bound untuk mendapatkan jumlah langkah minimum untuk menyelesaikan satu sisi.

Untuk menghitung jumlah langkah yang diperlukan untuk mencapai konfigurasi digunakan tabel yang menunjukkan jumlah langkah yang diperlukan
Misal : untuk memindahkan



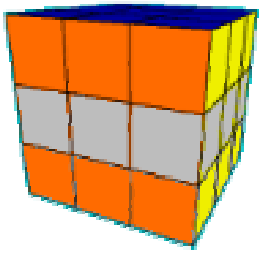
Diperlukan langkah: 
(5 langkah) sehingga f(i)= (f(parent)) + 5 (jumlah langkah) diketahui f(parent) = 7 . dari hasil tersebut didapat f(i) = 7 + 5 = 12
dari gambar diatas dapat dilihat bahwa ada 3 ubin pada sisi atas yang beluam sesuai maka
g(i) = jumlah ubin yang belum sesuai = 3
c(i) = f(i) + g(i) = 3 + 12 = 15
dan demikian seterusnya.
hasil dari perhitungan branch and bound tersebut dimasukkan pada tabel pada program dinamis

Misal



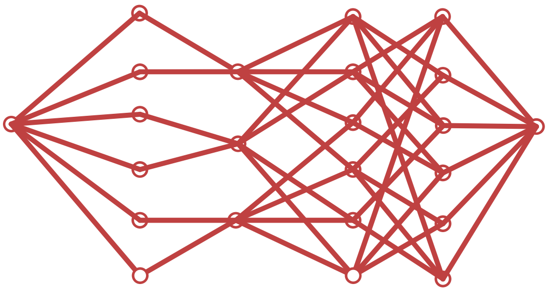
Dari hasil tree diatas dapat diketahui bahwa simpul terpendek adalah simpul yang paling kanan dengan banyak langkah 1 + 4 + 8 = 13. namun branch and bound masih mencari solusi lainnya sehingga cabang cabang lain juga diexpand.

Langkah selanjutnya adalah menyelesaikan sisi bawah tanpa merubah sisi atas dengan menggunakan branch and bound untuk mendapatkan langkah optimal. Bila lapisan dan sisi bawah telah selesai maka akan tampil seperti :



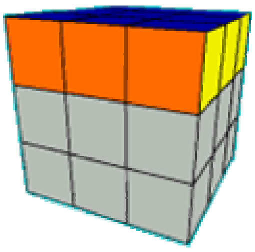
Langkah terakhir adalah menyelesaikan sisi sisi pada lapisan tengah untuk mendapatkan langkah optimal untuk menyelesaikan sebuah sisi digunakan branch and bound hasilnya dicatat dalam tabel pada program dinamis kemudian selesaikan sisi yang lainnya menggunakan branch and bound dan catat hasilnya pada program dinamis dan demikian seterusnya sehingga lapisan tengah telah selesai dengan berbagai urutan pengerjaan sisi sehingga seluruh Cube terselesaikan dan masukkan hasilnya kedalam table pada program dinamis

Untuk mendapatkan langkah paling optimal telusuri graf pada program dinamis



Catatan : graf disederhanakan untuk mempermudah pemahaman.
Ket : M : merah, B : biru, H : hijau, P : putih, J : jingga, K : kuning.

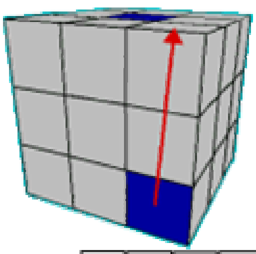
2. Penyelesaian Atas-Tengah-Bawah
Mula mula buat sisi bagian atas agar sesuai dengan konfigurasi yang diinginkan sehingga hasil akhirnya seperti :



catatan : warna bervariasi tergantung dari warna yang dipilih, warna abu abu menyatakan bahwa warna tersebut tidak diperdulikan.

Gunakan branch and bound untuk mendapatkan jumlah langkah minimum untuk menyelesaikan satu sisi.

Untuk menghitung jumlah langkah yang diperlukan untuk mencapai konfigurasi digunakan tabel yang menunjukkan jumlah langkah yang diperlukan
Misal : untuk memindahkan

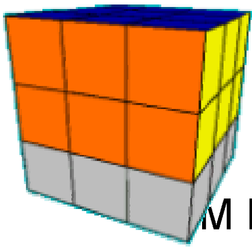


Diperlukan langkah: 
(4 langkah) sehingga f(i)=(f(parent)) + 4(jumlah langkah) karena ini langkah pertama maka parent daru i adalah akar sehingga f(parent) = 0 . dari hasil tersebut didapat f(i) = 0 + 4 = 4

dari gambar diatas dapat dilihat bahwa ada 7 ubin pada sisi atas yang beluam sesuai maka
g(i) = jumlah ubin yang belum sesuai = 7
c(i) = f(i) + g(i) = 4 + 7 = 11
dan demikian seterusnya.

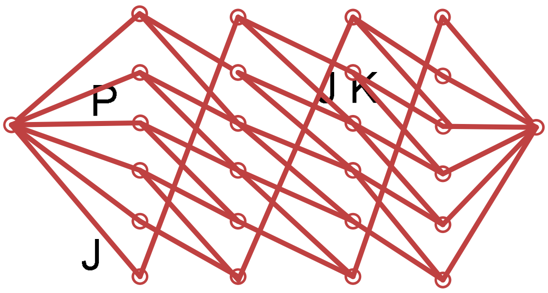
Kemudian masukkan hasil dari perhitungan branch and bound tersebut pada tabel pada program dinamis

Langkah selanjutnya adalah menyelesaikan sisi sisi pada lapisan tengah untuk mendapatkan langkah optimal untuk menyelesaikan sebuah sisi digunakan branch and bound hasilnya dicatat dalam tabel pada program dinamis kemudian selesaikan sisi yang lainnya menggunakan branch and bound dan catat hasilnya pada program dinamis dan demikian seterusnya sehingga lapisan tengah telah selesai dengan berbagai urutan pengerjaan sisi. Bila lapisan tengah telah selesai maka akan tampil seperti :



Langkah terakhir adalah menyelesaikan sisi bawah seperti biasa gunakan branch and bound untuk mendapatkan langkah tercepat sehingga seluruh Cube terselesaikan dan masukkan hasilnya kedalam table pada program dinamis

Untuk mendapatkan langkah paling optimal telusuri graf pada program dinamis



Catatan : graf disederhanakan untuk mempermudah pemahaman.
Ket : M : merah, B : biru, H : hijau, P : putih, J : jingga, K : kuning.

5. Kesimpulan

Masing masing penyelesaian memiliki kelebihan dan kekurangan. Untuk penyelesaian atas – bawah – tengah memiliki kelebihan dalam tingkat kerumitan

saat mencari solusi, namun banyak langkah tidak dapat optimalkan lagi. Sementara penyelesaian atas – tengah – bawah memiliki kelebihan dalam jumlah langkah yang dapat dioptimalkan, namun kekurangannya cukup rumit untuk mencari langkah. Dalam menyelesaikan rubik's cube dengan menggunakan algoritma branch and bound dan program dinamis diharapkan menghasilkan solusi dengan jumlah langkah terkecil dengan kerumitan yang tidak terlalu tinggi. Program dinamis menjamin pilihan sisi yang dipilih optimal, sementara branch and bound menjamin dalam menyelesaikan sebuah sisi didapat jumlah langkah sesedikit mungkin.

6. Daftar Pustaka

Munir, Rinaldi. 2005. Strategi Algoritmik. Teknik Informatika ITB : Bandung

Monroe, Matthew ,Direction for Solving Rubic Cube, Alchemistmatt.com

http://www.puzzlesolver.com