Penyelesaian Permasalahan 8 Puzzle dengan Menggunakan Algoritma A* (A Star)

Igor Bonny Tua Panggabean[1], Yoseph Suryadharma[2], Prasetyo Nugroho[3]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if14022@students.if.itb.ac.id[1], if14037@students.if.itb.ac.id[2], if14039@students.if.itb.ac.id[3]

Abstrak

Dalam dunia Informatika, penyelesaian masalah adalah bagian yang paling penting dalam membuat suatu aplikasi. Terutama dalam sebuah aplikasi yang diharapkan untuk memberikan sebuah atau beberapa buah solusi. Dalam mencari cara penyelasaian masalah, ada beberapa algoritma yang dapat digunakan. Diantaranya adalah algoritma A* (A Star) yang paling mangkus dalam mengerjakan persoalan yang membutuhkan kecerdasan buatan, terutama untuk traversal pada graf. Banyak persoalan yang bisa direpresentasikan dengan graf. Solusinya didapatkan dengan mengunjungi simpul-simpul pada graf. Algoritma A* mengunjungi simpul dalam graf dengan cara mengunjungi simpul yang paling mendekati solusi. Makalah ini menganalisa algoritma A* dalam menyelesaikan permasalahan yang membutuhkan traversal pada graf, yaitu kasus 8 Puzzle. Algoritma A* menerapkan heuristik untuk menemukan solusi yang paling optimum. Heuristik ini yang menyebabkan pohon ruang status tidak perlu dibangkitkan seluruhnya, hanya yang mendekati solusi terbaik saja. Pada kasus ini solusi terbaik dapat dicapai dengan menggeser kotak sampai didapatkan solusi dengan langkah sesedikit mungkin.

***Kata kunci**: algoritma, A*, Permasalahan Traversal*

1. Pendahuluan

Salah satu algoritma yang dipelajari untuk menyelesaikan permasalahan adalah algoritma A* (A Star). Algoritma A* menyelesaikan masalah yang menggunakan graf untuk perluasan ruang statusnya. Dengan kata lain digunakan untuk menyelesaikan permasalah yang bisa direpresentasikan dengan graf.

Algoritma A* adalah sebuah algoritma yang telah diperkaya. Dengan menerapkan suatu heuristik, algoritma ini membuang langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak akan mencapai solusi yang diinginkan.

Algoritma A* membangkitkan simpul yang paling mendekati solusi. Simpul ini kemudian disimpan suksesornya ke dalam list sesuai dengan urutan yang paling mendekati solusi terbaik. Kemudian, simpul pertama pada list diambil, dibangkitkan suksesornya dan kemudian suksesor ini disimpan ke dalam list sesuai dengan urutan yang terbaik untuk solusi. List simpul ini disebut dengan *simpul terbuka(open node).*

Simpul pada list bisa berasal dari kedalaman berapapun dari graf. Algoritma ini akan mengunjungi secara mendalam (mirip DFS) selama simpul tersebut merupakan simpul yang terbaik. Jika simpul yang sedang dikunjungi ternyata tidak mengarah kepada solusi yang diinginkan, maka akan melakukan runut balik ke arah simpul akar untuk mencari simpul anak lainnya yang lebih menjanjikan dari pada simpul yang terakhir dikunjungi. Bila tidak ada juga, maka akan terus mengulang mencari ke arah simpul akar sampai ditemukan simpul yang lebih baik untuk dibangkitkan suksesornya. Strategi ini berkebalikan dengan algoritma DFS yang mencari sampai kedalaman yang terdalam sampai tidak ada lagi suksesor yang bisa dibangkitkan sebelum melakukan runut balik, dan BFS yang tidak akan melakukan pencarian secara mendalam sebelum pencarian secara melebar selesai. A* baru berhenti ketika mendapatkan solusi yang dianggap solusi terbaik.

Algoritma A* menerapkan teknik heuristik dalam membantu penyelesaian persoalan. Heuristik adalah penilai yang memberi harga pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan heuristik yang benar, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya) yang dicari. Dengan kata lain, heuristik adalah fungsi optimasi yang menjadikan algoritma A* lebih baik dari pada algoritma lainnya. Namun heuristik masih merupakan estimasi / perkiraan biasa saja. Sama sekali tidak ada rumus khususnya. Artinya,

setiap kasus memiliki fungsi heuristik yang berbeda-beda.

Algoritma A* ini bisa dikatakan mirip dengan algoritma Dijkstra, namun pada algoritma Dijkstra, nilai fungsi heuristiknya selalu 0 (nol) sehingga tidak ada fungsi yang mempermudah pencarian solusinya.

1.1 Tujuan Penulisan Makalah

Makalah ini ditulis untuk mengenalkan Algoritma A* yang termasuk algoritma baru kepada para pembaca. Dengan harapan mereka menjadi lebih tahu dan mengerti tentang penerapan algoritma ini dan mampu membandingkannya dengan algoritma traversal graf yang lainnya.

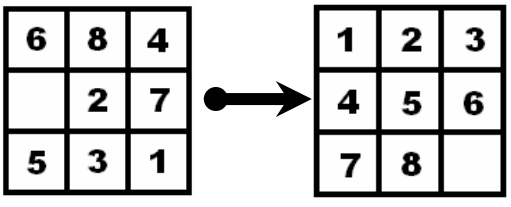
Makalah ini juga dituliskan untuk memenuhi tugas mata kuliah IF2251 Strategi Algoritmik pada semester genap tahun ajaran 2005/2006 di Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

1.2 Ruang Lingkup

Ruang lingkup pembahasan makalah ini adalah penggunaan algoritma A* untuk menyelesaikan salah satu permasalah yang dapat diimplementasikan dengan menggunakan graf untuk mencari solusinya. Permasalah yang dibahas di makalah ini adalah permasalahan 8 puzzle.

2. Pembahasan Masalah

Permasalahan 8 Puzzle ditemukan oleh Sam Lloyd. Puzzle ini berukuran 3 x 3 kotak dengan tiap kotak dinomori dari 1 sampai 8, dan kotak yang terakhir dibiarkan kosong. Tiap kotak yang bernomor bisa saling berpindah tempat dengan kotak yang kosong, dengan syarat kotak yang kosong ada di sebelah kotak yang bernomor. Pertama kali permainan, posisi kotak tersebut sudah teracak. Maka itu hasil akhir dari permainan ini adalah posisi kotak sudah berada di tempatnya yang sesuai (terurut).



Posisi Awal Puzzle Posisi Akhir Puzzle

2.1 Penentuan Rumus Heuristik

Rumus heuristik yang digunakan pada kasus ini adalah

|(AbsAwal – OrdAkhir)| + |(OrdAwal – AbsAkhir)| *(1)*

Rumus heuristik ini diterapkan pada kotak yang mungkin untuk digerakkan. Kemudian dipilih heuristik yang paling besar diantara semua kemungkinan tadi. Kotak yang terpilih, akan digerakkan ke kotak yang kosong, lalu akan dibangkitkan lagi anak pohon dari status yang sekarang. Dan memulai lagi proses penentuan heuristik untuk kemungkinan kotak yang baru.

Contoh

Dari gambar di atas :
heuristik kotak 6 adalah |(0 – 1)| + |(0 – 2)| = 3,
heuristik kotak 2 adalah |(1 – 0)| + |(1 – 1)| = 1,
heuristik kotak 5 adalah |(0 – 1)| + |(2 – 1)| = 2.
Dari perhitungan di atas, maka kotak yang dipilih untuk digerakkan adalah kotak 6.

2.2 Penerapan Algoritma A*

Setelah menentukan heuristik, kita menggerakkan kotak yang terpilih. Setiap pergerakan yang dilakukan statusnya akan disimpan pada suatu list. List ini akan digunakan untuk melakukan pengecekan apakah kita sudah pernah membangun status tersebut atau belum agar kita tidak menggerakkan kotak yang sama berkali-kali ke status yang sama.

Dengan menerapkan strategi ini, selain menemukan solusi, algoritma ini juga bisa menemukan langkah terpendek untuk mencapai solusi tersebut.

2.3 Algoritma A* untuk Persoalan 8 Puzzle

1. Jadikan status awal sebagai akar pohon persoalan.
2. Tentukan heuristik untuk tiap kotak yang mungkin untuk digerakkan.
3. Dari kemungkinan yang ada, lakukan perbandingan heuristiknya.
4. Bila ada nilai heuristik yang sama, maka yang digerakkan adalah kotak yang nilai heuristiknya ditentukan pertama kali.
5. Bila tidak ada nilai yang sama, maka yang digerakkan adalah kotak yang nilai heuristiknya yang terbesar
6. Bangkitkan anak pohon status dari simpul saat ini, dengan status baru adalah status setelah kotak digerakkan
7. Ulangi prosedur 2-6 sampai ditemukan solusi yang paling optimum.

3. Kesimpulan

1. Ada banyak algoritma yang bisa digunakan untuk memecahkan masalah yang dapat direpresentasikan dengan menggunakan graf.
2. A* adalah suatu teknik optimasi dari algoritma traversal graf.
3. Optimasi algoritma ini disebabkan oleh penerapan heuristik pada saat pembangkitan pohon statusnya.
4. Untuk mendapatkan solusi yang optimum, diperlukan heuristik yang optimum pula.
5. Untuk setiap simpul daun, nilai heuristiknya merupakan nilai aktual solusi (nilai yang menandakan simpul tersebut adalah solusi atau bukan solusi)
6. Gunakan urutan yang benar dari nilai heuristik untuk menentukan simpul yang akan dibangkitkan. Terurut secara maksimum, atau secara minimum, tergantung dari permasalahan yang harus dipecahkan.
7. Semakin akurat heuristiknya, semakin cepat solusi didapatkan.
8. Algoritma A* mirip dengan algoritma Branch and Bound, tetapi mengekspan pohon secara DFS, bukan BFS.

4. Daftar Pustaka

1. Lester, Patrick. *"A* Pathfinding for Beginners"*. **http://www.policyalmanac.org/games/aStarTutorial.htm**. Diakses tanggal 17 Mei 2006 pukul 11.00
2. Munir, Rinaldi.2006. *"Strategi Algoritmik"*. Departemen Teknik Informatika, Institut Teknologi Bandung
3. Patel, Amit J. *"AStar Comparison Introduction"*. **http://theory.stanford.edu/~amitp/GameProgramming/** . Diakses tanggal 17 Mei 2006 pukul 10.30
4. Wikimedia Foundation, Inc. *"A * Search Algorithm"*. **http://en.wikipedia.org/wiki/A-star_search_algorithm**. Diakses tanggal 17 Mei 2006 pukul 10.45