Penerapan Algoritma Runut-Balik Untuk Penyelesaian Teka-Teki Sudoku

Morenvino M[1], Ray A I[2], Anton R S[3]

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

E-mail : if14034@students.if.itb.ac.id[1], if14045@students.if.itb.ac.id[2], if14046@students.if.itb.ac.id[3]

Abstrak

Algoritma yang banyak diterapkan dalam mengaplikasikan permainan adalah algoritma runut-balik (*back-tracking)*. Dengan menggunakan algoritma ini, penyelesaian suatu permainan,yang notabene melibatkan banyak kemungkinan, dapat diselesaikan dengan lebih cepat. Hal ini dikarenakan, jika kita menggunakan algoritma runut-balik, tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang perlu dipertimbangkan. Salah satu jenis permainan yang dapat diselesaikan dengan algoritma runut-balik adalah permainan teka-teki Sudoku.

***Kata kunci**: runut-balik, teka-teki sudoku*

1. Pendahuluan

Permainan teka-teki merupakan salah satu permainan yang sering kita jumpai dalam kehidupan sehari-hari. Ada berbagai macam permainan teka-teki, salah satunya adalah Sudoku yang berasal dari Jepang, dan pemainnya dituntut untuk berpikir dengan logika. Secara umum, Sudoku merupakan permainan dalam bentuk tabel yang berukuran 9x9, dan dalam tabel tersebut terdapat 9 kelompok/daerah/blok yang berukuran 3x3. Untuk menyelesaikan teka-teki Sudoku, pemain "cukup" mengisi sel-sel yang kosong dengan angka 1-9 sedemikian sehingga dalam 1 baris dan 1 kolom, tidak ada angka yang sama, dan dalam 1 blok, tidak ada angka yang sama pula.

Untuk memecahkan teka-teki Sudoku, dapat digunakan algoritma runut-balik (*backtracking)*. Algoritma ini berbasis pada DFS (*Depth First Search)*, dimana solusi dapat ditemukan dengan penelusuran yang lebih sedikit dan dapat mencari solusi permasalahan secara lebih mangkus. Algoritma runut balik sendiri merupakan perbaikan dari algoritma *brute force* dan *DFS*.

2. Sudoku

2.1 Gambaran umum teka-teki Sudoku

Sudoku adalah suatu permainan teka-teki yang memiliki aturan sederhana. Teka-teki ini terdiri atas 9 buah blok yang berupa tabel 3 × 3. Sebagian sel tabel dalam teka-teki ini telah diisi dengan angka-angka yang berupa patokan untuk menyelesaikan tek-teki. Tujuan permainan ini adalah untuk mengisi setiap sel tabel yang masih kosong dengan angka-angka, sedemikian sehingga dalam 1 blok hanya terdiri atas angka 1-9 yang tidak berulang dan tidak ada angka yang berulang dalam 1 baris maupun kolom.

Meskipun aturannya sederhana namun penyelesaian teka-teki ini tidak semudah aturannya. Tentu saja tingkat kesulitan tiap teka-teki dapat bervariasi.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Gambar permainan Sudoku

2.2 Strategi umum penyelesaian teka-teki Sudoku

Secara umum, Sudoku dapat diselesaikan dengan kombinasi teknik pemindaian (*scanning)*, penandaan (*marking)*, dan analisa (*analysing)*. Beberapa teki-teki Sudoku yang tergolong mudah dapat diselesaikan hanya dengan salah satu proses, namun pada umumnya kita harus mengkombinasikan ketiga teknik tersebut.

a. Pemindaian

Berupa proses memindai baris atau kolom untuk mengindentifikasi baris mana dalam suatu blok yang terdapat angka-angka tertentu. Proses ini kemudian diulang pada setiap kolom (atau baris) secara sistematis. Kemudian menentukan nilai dari suatu sel dengan membuang nilai-nilai yang tidak mungkin.

b. Penandaan

Berupa analisa logika, dengan menandai kandidat angka yang dapat dimasukkan dalam sebuah sel.

c. Analisa

Berupa eliminasi kandidat, dimana kemajuan dicapai dengan mengeliminasi kandidat angka secara berturut-turut hingga sebuah sel hanya punya 1 kandidat.

2.3 Komputasi dalam teka-teki Sudoku

Penyelesaian teka-teki Sudoku $n^2 \times n^2$ adalah permasalahan *NP-complete*. Pada dasarnya, Sudoku dapat diselesaikan oleh algoritma traversal graf dengan membangun seluruh pohon kemungkinan. Penyelesaian Sudoku juga dapat dipandang sebagai masalah pewarnaan graf yaitu mewarnai graf dengan simpul $n^2 \times n^2$, dimana warna yang digunakan direpresentasikan dalam angka 1- 9, yang seluruhnya harus digunakan.

3. Algoritma runut-balik

3.1 Gambaran singkat algoritma runut-balik

Runut-balik *(backtracking)* adalah algoritma yang berbasis pada *DFS* untuk mencari solusi persoalan secara lebih mangkus. Runut balik, yang merupakan perbaikan dari algoritma brute force, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada. Dengan metoda ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Akibatnya, waktu pencarian dapat dihemat. Runut-balik lebih alami dinyatakan dalam algoritma rekursif. Kadang-kadang disebutkan pula bahwa runut balik merupakan bentuk tipikal dari algoritma rekursif.

3.2 Prinsip pencarian algoritma runut-balik

Di sini, kita hanya akan meninjau pencarian solusi pada pohon ruang status yang dibangun secara dinamis. Langkah-langkah pencarian solusi adalah sebagai berikut :

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pembentukan yang dipakai adalah mengikuti metode pencarian mendalam (*DFS*). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul-simpul yang sudah dilahirkan dinamakan simpul hidup (*live node*). Simpul hidup yang sedang diperluas dinamakan simpul-E (*expand node*). Simpul dinomori dari atas ke bawah sesuai dengan urutan kelahirannya.
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang sedang dibentuk tidak mengarah ke solusi, maka simpul-E tersebut dibunuh sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (bounding function). Simpul yang sudah mati tidak akan pernah diperluas lagi
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian diteruskan dengan membangkitkan simpul anak yang lainnya. Bila tidak ada lagi simpul anak yang dapat dibangkitkan, maka pencarian solusi dilanjutkan dengan melakukan runut balik ke simpul hidup terdekat (simpul orangtua). Selanjutnya, simpul ini menjadi simpul-E yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.
4. Pencarian dihentikan bila kita telah menemukan solusi atau tidak ada lagi simpul hidup untuk runut-balik

5. Penerapan algoritma runut-balik dalam teka-teki Sudoku

Algoritma runut balik sangat efektif dalam mengurangi jumlah pencarian kemungkinan solusi teka-teki. Menurut perhitungan ada sebanyak 6,670,903,752,021,072,936,960 jumlah kemungkinan status untuk teka-teki Sudoku berukuran 9 x 9. Suatu angka yang sangat besar jika ingin dicari secara *brute-force*, dengan kompleksitas. Garis besar algoritma runut-balik untuk menyelesaikan teka-teki diberikan di bawah ini:

Deklarasi global

```
N : integer
      {adalah ukuran papan sudoku }

tabel : array[1.. N^2] of
        array[1..N^2]
{representasi papan sudoku}
```

```
procedure solve()
{
menyelesaikan teka-teki  Sudoku
pada papan n^2 * n^2,
versi: rekursif
}

Algoritma

if( isDone() ) then
      exit()
      {papan sudah selesai diisi
      program selesai}

else

      updateTabel()
      {update sel tabel yang masih
       kosong}


      if( not isValidBaris() ) then
            backtrack()
      else
      {jika baris valid maka
       periksa kolom}
         if( not isValidKolom())
         then
            backtrack()
         else
         {jika kolom valid maka
          periksa blok}
            if ( not isValidBlok())
            then backtrack()
            else
            {jika blok valid maka
             panggil rekursif}
                  solve()
```

5. Kesimpulan

Permainan teka-teki Sudoku dapat diselesaikan dengan algoritma runut-balik. Pemilihan algoritma runut-balik dalam memecahkan teka-teki Sudoku dapat dibilang cukup baik jika dibandingkan dengan algoritma *brute force*. Hal ini dikarenakan dalam runut-balik, tidak perlu dicari semua kemungkinan solusi, berbeda dengan brute force yang mencari semua kemungkinan solusi. Dengan menggunakan algoritma ini, semua teka-teki Sudoku dapat dipecahkan dan waktu untuk memecahkan teka-teki ini lebih singkat dari brute-force.

Daftar Pustaka

1. Munir, Rinaldi, *Diktat kuliah IF2251 Strategi Algoritmik*, Bandung, 2005.
2. **http://www.wikipedia.org** diakses pada Kamis 18 Mei 2006.
3. **http://www.functologic.com** diakses pada hari Jumat 19 Mei 2006.
4. **http://bj.middleburry.edu** diakses pada hari Jumat 19 Mei 2006.