Penyelesaian Permainan Bantumi dengan Algoritma *Expand, Mark and Select*

Zulfikar Hakim

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung
e-mail : zulfikar.hakim@gmail.com

ABSTRAK

BFS adalah salah satu cara iterasi dalam pohon ruang solusi secara melebar. BFS banyak digunakan dalam permasalahan mencari solusi. Solusi yang pertama kali ditemukan oleh iterator akan menjadi solusi atas permasalahan.

Namun demikian, BFS konvensional tidak dapat memenuhi kebutuhan optimasi. Oleh karena itu, penulis melakukan modifikasi pada algoritma BFS yang diberi nama algoritma Expand, Mark, and Select (EMS), yang merupakan salah satu variasi dari algoritma Branch and Bound. Algoritma akan ini digunakan untuk memecahkan permainan tradisional Afrika yang dikembangkan lebih dari 3000 tahun yang lalu di daerah Mesir.

Permainan yang dimaksud adalah *bantumi*, sebuah permainan yang diadaptasi dari permainan tradisional Afrika, *Mancala*. Prinsip permainan Mancala adalah dengan memindahkan sejumlah batu-batu dari salah satu lubang ke dalam lubang lainnya. Terdapat dua lubang utama di tepi papan yang diberi nama sama dengan permainannya : lubang Mancala. Pemain yang berhasil memasukkan paling banyak batu ke dalam Mancala menjadi pemenangnya.

Permainan ini sederhana, namun cukup sukar dimainkan. Di sini akan dibahas bagaimana memainkan permainan klasik Bantumi, atau Mancala, dengan teori teknik perancangan algoritma modern.

Kata Kunci : BFS, EMS, Mancala, Bantumi

1. PENDAHULUAN

Mancala adalah permainan papan (*board games*). Mancala telah dikenal di daerah barat dengan sebutan *Kalah* dan *Oware.* Konon, kata *Mancala* merupakan adapatasi dari bahasa Arab *naqala* yang artinya "bergerak".

Berikut ini adalah sejarah Mancala dan bagaimana perkembangan Mancala dewasa ini.

1.1. Sejarah Mancala

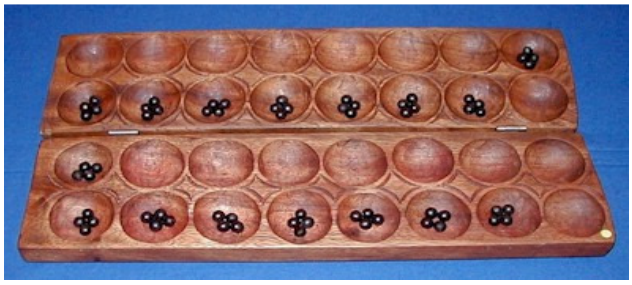
Mancala mungkin menjadi salah satu permainan strategi tertua di dunia. Mancala klasik dapat dimainkan di berbagai macam media yang ada di sekitar kita. Di Afrika misalnya, Mancala biasa dimainkan dengan batu kerikil sebagai bijinya dengan lubang kecil yang digali di atas sepetak tanah. Bantumi juga biasa dimainkan di atas papan kayu dengan bebijian sebagai biji permainan.

Gambar 1 Media permainan bantumi berupa lubang di atas tanah

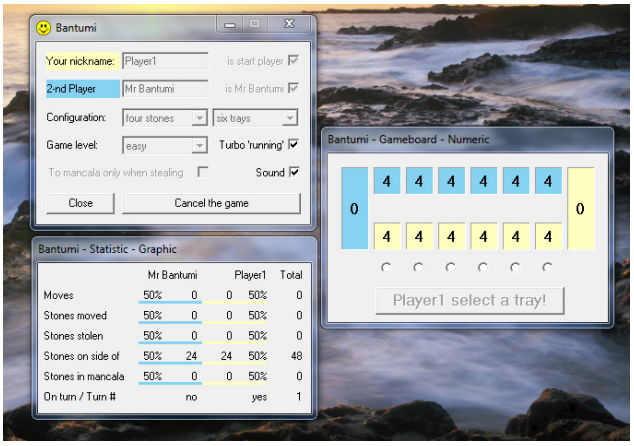
Mancala telah ditemukan di atas atap kuil di daerah Memphis, Thebes, dan Luxor. Setelah diidentifikasi, permainan ini telah dimainkan oleh penduduk setempat semenjak 1400 SM, atau sekitar 3400 tahun yang lalu.

1.2. Permainan Mancala Kini

Variasi dari permainan Mancala telah dimainkan di seluruh Afrika. Papan permainan Mancala telah dapat dimainkan di seluruh Karibean dan Pantai Timur Afrika Selatan. Versi lainnya dari permainan ini terdapat di India, Sri Lanka, Filipina, bahkan di Indonesia dan Malaysia. Variasi dari Mancala di Indonesia umumnya disebut dengan Congklak, dan di Malaysia biasa disebut dengan Congkak. Berikut ini akan ditunjukkan beberapa variasi permainan Mancala :



Gambar 2 Permainan Omweso dari Uganda. Gambar papan permainan ini diambil dari pameran di 4th Mind Sport Olimpiad, Alexandra Palace, Londong di tahun 2000



Gambar 3 Permainan Bantumi, versi digital dari Mancala. Biasa dimainkan di bawah sistem operasi windows

Kini, sebuah permainan Mancala versi digital yang diberi nama Bantumi telah dikembangkan. Sayangnya, versi digital ini berlisensi freeware (source code tidak dibuka) sehingga tidak dapat diulik algoritma di balik permainan ini.

1.3. Peraturan Permainan

Permainan Bantumi direpresentasikan di sebuah perangkat lunak melalui kotak dan angka. Kotak merepresentasikan tiap-tiap lubang di papan permainan Mancala, sedangkan angka merepresentasikan total biji yang ada di dalam kotak. Sebuah persegi panjang besar di tepi jendela program merepresentasikan kotak Mancala.

Pada aplikasi dalam gambar tiga, kotak direpresentasikan dengan dua warna. Kotak dengan warna yang berbeda dimiliki oleh pemain yang berbeda, satu set kotak untuk seorang pemain, dan satu set kotak lainnya untuk pemain lawannya.

Tidak berbeda jauh dengan salah satu permainan Mancala di Indonesia, Congklak, pemain pada gilirannya diminta untuk mengambil seluruh kumpulan biji yang ada di salah satu lubang yang dimilikinya (selain lubang Mancala).

Kemudian, dengan arah pergerakan berlawanan dengan jarum jam, pemain meletekkan satu per satu biji yang ada di tangannya ke lubang yang ada, termasuk lubang Mancala. Ketentuannya adalah sebagai berikut :

1. Arah pergerakan selalu berlawanan dengan jarum jam.
2. Pergerakan harus disertai dengan peletakkan satu buah biji pada setiap lubang yang dilalui, termasuk kotak milik lawan.
3. Biji tidak diletakkan di lubang Mancala milik pemain lawan.
4. Jika biji jatuh di lubang Mancala, maka poin pemain bertambah satu. Giliran kembali milik pemain yang sedang bergerak
5. Jika biji jatuh pada kotak milik pemain, maka biji yang ada pada kotak tersebut dan kotak di seberangnya (milik musuh) dimasukkan ke dalam lubang Mancala milik pemain.
6. Jika salah satu pemain kehabisan biji pada kotak yang dimilikinya (selain Mancala) maka seluruh biji akan diakumulasikan bagi salah satu pemain

2. METODE

2.1. Ide Dasar dan Terminologi

Ide dasar dari algoritma Expand, Mark, and Select (selanjutnya disebut algoritma EMS) adalah pencarian ruang solusi dengan metode iterasi pencarian melebar (Breadth-First Search) dengan menambahkan beberapa tahap dasar yaitu Expand (mengekspansi simpul daun dan menentukan nilai simpulnya), Mark (menandai kandidat solusi), dan juga Select (memilih solusi optimum dari kandidat solusi). Berbeda dengan BFS yang hanya memilih sebuah solusi atas masalah yang diberikan, EMS harus memilih sekian kandidat solusi dari pohon ruang solusi yang memiliki nilai optimum (optimum minimal maupun optimum maksimal). Berbeda pula dengan algoritma *greedy*, algoritma ini memilih nilai optimum dari seluruh kemungkinan yang ada dari kandidat solusi. Kandidat solusi dipilih melalui pemilihan setelah seluruh daun dibentuk.

Berikut ini adalah terminologi dalam algoritma EMS :

1. **Pohon ruang solusi**. Pohon yang dibentuk berdasarkan prinsip BFS.
2. **Nilai simpul** (*node value*). Nilai simpul berisi sebuah nilai integer [0..∞]yang menjadi sebuah ukuran kuantitatif dari alternatif solusi – yang direpresentasikan oleh sebuah simpul di pohon ruang solusi. Nilai simpul akan menjadi patokan dasar pemilihan solusi optimal. Nilai simpul dilambangkan dengan C_i di mana i adalah simpul di dalam pohon ruang status.
3. **Fungsi penandaan** (*Marking Function*). Fungsi penandaan adalah sebuah fungsi pembatas seperti fungsi pembatas pada algoritma *branch and bound*. Namun, berbeda dengan fungsi pembatas pada algoritma *branch and bound* yang mematikan sebuah simpul dalam pohon, fungsi penandaan bertugas untuk menandai sebuah simpul agar tidak diekspansi dan kemudian memasukkan properti simpul tersebut ke dalam himpunan kandidat solusi. Fungsi

penandaan dilambangkan dengan M(i) di mana i adalah sebuah simpul yang akan dicek.

4. **Fungsi ekspansi** (*expanding function*). Fungsi ekspansi menentukan berapa jumlah anak yang akan diekspansi, berapakah (atau apakah) properti sisi (edge) yang menghubungkan simpul tersebut dengan simpul orang tuanya, dan berapakah bobot dari simpul tersebut. Fungsi ekspansi dilambangkan dengan E(i) → (n, {($P_{i,j}$, C_j)}) di mana i adalah subjek ekspansi (siapa yang melakukan ekspansi) dan j adalah objek ekspansi (subjek ekspansi mengekspansi siapa)
5. **Fungsi pembatas** *(bounding function)***.** Fungsi pembatas di sini berperilaku mirip dengan fungsi pembatas pada algoritma Branch and Bound**.** Fungsi pembatas berfungsi untuk mengecek apakah satu simpul menawakan solusi yang *feasible* atau tidak. Jika tidak *feasible*, maka simpul tersebut akan dimatikan dan jalur menuju simpul tidak boleh digunakan sama sekali. Fungsi pembatas dilambangkan dengan B(i) di mana i adalah simpul yang akan dicek.
6. **Kondisi berhenti** (*Stop Condition*)**.** Kondisi berhenti menjadi sebuah acuan di mana pembangkitan simpul pada ruang solusi harus dihentikan. Kondisi berhenti dipenuhi jika seluruh simpul daun telah ditandai atau dibunuh.
7. **Kandidat solusi** (*Solution Candidate*) merupakan himpunan simpul dan propertinya yang menjadi dipilih menjadi kandidat solusi oleh fungsi penandaan. Kandidat solusi dilambangkan dengan SC yang berisi himpunan {$(Pa,C_i)_1$, $(Pa,C_i)_2$, $(Pa,C_i)_3$, $(Pa,C_i)_4$, ... , (Pa, $C_i)_n$}
8. **Fungsi pemilihan** (*Selecting function*). Fungsi pemilihan adalan fungsi yang mencari solusi optimum dari kandidat solusi. Fungsi ini dapat menjadi fungsi mencari nilai simpul maksimal dari himpunan kandidat solusi, atau dapat juga mencari nilai simpul minimal. Fungsi pemilihan dilambangkan dengan S(SC) → (Pa, C_i). Selain memilih solusi optimal, fungsi pemilihan juga menambahkan pertimbangan pengecualian (exception) dari solusi yang dipertimbangkan)
9. **Solusi optimal** (*optimum solution*) atau S. Solusi optimal berisi informasi (Pa, $C_i)_{Optimal}$ yang menjadi solusi atas permasalahan.

2.2. Properti Algoritma EMS

Properti algoritma EMS tidak jauh berbeda dengan poin terminologi EMS. Properti algoritma ini adalah :

1. Fungsi penandaan

M(i)

2. Fungsi ekspansi

E(i) = (n, {($P_{i,j}$, C_j)})

3. Fungsi pemilihan

S(k) → (Pa, C_i)

4. Kandidat solusi

SC

5. Solusi optimal

S(SC)

6. Fungsi pembatas

B(i)

2.3. Prinsip pencarian solusi

Prinsip pencarian solusi pada algoritma EMS adalah sebagai berikut :

1. Sebuah simpul dipilih untuk mewakili suatu kondisi awal. Akar dinomori dengan angka 1.
2. Simpul tersebut diekspansi dengan fungsi ekspansi E(1). Akar akan memiliki n buah anak yang masing-masing bernilai C_j dan yang dihubungkan dengan sisi berproperti $P_{i,j}$.
3. Setiap anak dicek dengan fungsi pembatas B(i) di mana i untuk iterasi pertama adalah [2..n]. Fungsi pembatas akan mematikan node yang tidak *feasible* untuk dijadikan solusi.
4. Setiap anak yang tidak dimatikan oleh fungsi pembatas dicek kembali dengan fungsi penandaan M(i) di mana i untuk iterasi pertama adalah adalah [2..n]. Fungsi penandaan akan menandai node yang menjadi kandidat solusi dan memasukkannya ke dalam *array* kandidat solusi. Node ini tidak akan diperluas lagi pada kesempatan berikutnya
5. Fungsi ekspansi kembali mengekspansi seluruh simpul daun yang ada. Kemudian langkah 2 dan langkah 5 diiterasi sampai ditemui kondisi berhenti.
6. Fungsi solusi optimal S(SC) akan mencari nilai optimal dari seluruh array yang ada di dalam kandidat solusi dan memberikan solusi optimal dari permasalahan yang ada.

2.4. Pseudocode Skema Umum EMS

Pseudocode skema umum algoritma EMS berdasarkan struktur data yang ada adalah sebagai berikut:

```
Function getEMSSolution () → elmt_SC
{mendapatkan element solusi kandidat
yang memenuhi fungsi seleksi}

Kamus lokal
i:node

Algoritma
i ← 1 {i diassign dengan akar pohon
ruang solusi}

while not semua simpul dikunjungi do
   if (not B(i) and not M(i)) then
      E(i)
   endif
   i← i+1
endwhile

return S(SC)
```

2.5. Waktu yang tepat untuk menggunakan algoritma EMS

Algoritma EMS sangat tepat digunakan untuk kasus-kasus di mana nilai simpul bukan menjadi sebuah tolak ukur utama dalam menentukan solusi. Misalnya ada kondisi khusus di mana nilai simpul yang tinggi justru diprediksi menimbulkan hasil yang tidak optimal pada optimal global. Kondisi ini biasanya terjadi pada pemrograman *game*.

Dalam makalah ini, karakteristik fungsi seleksi sangat terlihat ketika fungsi pemilihan harus memilih secara sekaligus maksimal dari nilai simpul namun sekaligus minimal biji yang jatuh ke lubang lawan. Dua kondisi kontradiksi ini secara eksplisit dinyatakan dalam fungsi seleksi.

3. Analisis penyelesaian masalah

3.1. Logika dasar

Pemilihan suatu kotak sebagai solusi dalam sebuah giliran akan berimplikasi kepada salah satu kemungkinan dari dua kejadian di bawah ini (dengan masing-masing beberapa subkejadian)

1. Biji terakhir jatuh di kotak milik sendiri (giliran tidak berpindah)
    a. Biji melewati Mancala dan biji di dalam Mancala bertambah satu. Biji jatuh di kotak milik sendiri yang tidak kosong
    b. Biji melewati Mancala dan biji dalam Mancala bertambah satu. Biji jatuh di kotak milik sendiri yang kosong. Mancala ditambahkan biji di kotak tersebut dan kotak di seberangnya (milik lawan)
2. Biji terakhir jatuh di kotak milik lawan (giliran berpindah)
    a. Biji melewati Mancala dan biji di dalam Mancala bertambah satu

Analisis ini berusaha memecahkan bagaimana biji dapat masuk ke dalam kotak Mancala dalam jumlah yang maksimal. Diharapkan, dengan cara ini permainan dapat dimenangkan oleh pemain/ aplikasi yang menggunakan algoritma ini. Prinsip pencarian solusi untuk kasus ini sama dengan prinsip pencarian solusi pada algoritma EMS secara umum. Sehingga pada bab ini hanya akan dijelaskan properti dari algoritma EMS yang akan digunakan spesifik terhadap masalah yang dihadapi.

3.2. Struktur Data

Banyaknya biji dalam sebuah giliran akan menjadi faktor ukur kuantitatif sehingga banyaknya biji yang dapat ditambahkan ke dalam kotak Mancala dalam satu giliran adalah cost dari sebuah simpul atau jalur.

Sementara itu, pohon ruang solusi didefinisikan sebagai seluruh kemungkinan pemilihan biji dalam satu giliran. Dengan demikian, pilihan dapat menjadi sangat lebar, yaitu beberapa kali pilihan sampai tidak ada pilihan lagi kecuali biji jatuh di tempat lawan.

Sementara itu, kalaupun biji harus jatuh di tempat lawan, maka diprioritaskan biji jatuh pada posisi di mana kotak lawan bernilai 0. Hal ini dimaksudkan agar lawan tidak dapat memenuhi kondisi 1b yang sangat merugikan pemain. Untuk melakukan prioritisasi ini dibutuhkan strukutr data tambahan pada setiap elemen kandidat solusi.

Sisi (*edge*) direpresentasikan oleh integer [1..6] yang menggambarkan index pengambilan biji pada kotak pemain (Kotak_pemain). Sedangkan pohon direpresentasikan oleh matrix ketetanggaan di mana isi matrix adalah sisi yang menghubungkan simpul bapak dengan simpul anaknya. Berikut ini adalah struktur data dari algoritma yang akan dibangun :

```
const jumlah_biji=48

{Struktur data node}
typedef node
{
    Nilai_simpul : integer
    no_simpul : integer
    path : Array[] of integer
}
```

```
const jumlah_biji=48

{Struktur data node}
typedef node
{
    Nilai_simpul : integer
    no_simpul : integer
    path : Array[] of integer
}

{Struktur data elemen array kandidat solusi}
typedef elmt_SC
{
    simpul : node
    isLastKotakKosongMusuh : boolean
    {bernilai true jika biji terkahir jatuh di kotak lawan yang kosong (tidak berisi biji)}
}

{struktur data kotak pemain, kotak musuh, serta Mancala pemain dan musuh}
Kotak_pemain : array [6] of integer
Kotak_lawan  : array [6] of integer
Mancala_pemain : integer
Mancala_lawan  : integer

{struktur data matrix ketetanggaan}

Ruang_solusi : matrix[][] of integer [1..6]
```

3.3. Properti algoritma EMS untuk pemasalahan Bantumi

Efektifitas algoritma EMS sangat bergantung kepada properti yang ditetapkan bagi algoritma tersebut oleh programmer. Berikut ini adalah properti algoritma EMS yang diajukan :

3.3.1. Fungsi penandaan

Simpul yang ditandai sebagai kandidat solusi adalah simpul yang mau tidak mau harus menjatuhkan biji terakhirnya di kotak lawan. Logikanya, tidak mungkin seluruh biji yang ada di kotak pemain dimasukkan ke dalam Mancala secara sekaligus. Sehingga suatu saat giliran pasti diakhiri dengan menjatuhkan biji terakhir ke kotak lawan.

3.3.2. Fungsi ekspansi

Fungsi ekspansi selalu menghasilkan n=6 dan properti bernilai 1 sampai dengan 6 untuk masing-masing node. Nilai simpul selalu dihitung berdasarkan aturan perhitungan nilai simpul suatu node ditambah dengan simpul di atasnya. Perhitungan nilai simpul didapatkan dari jumlah biji yang dapat dimasukkan ke dalam Mancala dalam sebuah pilihan gerakkan.

3.3.3. Fungsi Pembatas

Jika suatu kotak tidak berisi satu pun biji, maka kotak tersebut tidak dapat diperluas. Fungsi pembatas membatasi perluasan perluasan sebuah simpul jika kotak yang diambil bernilai 0.

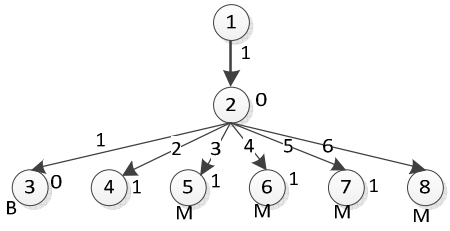
3.3.4. Fungsi pemilihan

Fungsi pemilihan adalah untuk memaksimalkan biji yang masuk ke dalam Mancala dalam setiap gilirannya. Oleh karena itu, fungsi pemilihan berfungsi untuk mendapatkan **nilai simpul maksimal** dari seluruh elemen kandidat solusi yang ada.

Di samping itu, jika terdapat nilai maksimal berada pada lebih dari satu kandidat solusi, maka biji yang jatuh di tempat lawan harus **diminimalkan**. Hal ini bertujuan untuk memperkecil kesempatan lawan untuk menempatkan biji di Mancala dan juga mengantisipasi kejadian pada aturan nomor enam.

4. PROSES PEMBENTUKAN RUANG SOLUSI

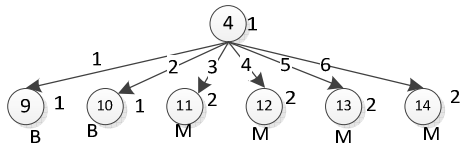
Proses pembentukan ruang solusi menghasilkan pohon yang cukup besar untuk index maksimal kotak pemain 6. Oleh karena itu, proses pembentukan ruang solusi dibatasi hanya pada pengambilan pertama adalah kotak pertama. Pengambilan solusi dinyatakan dalam array of integer. Masing-masing dinyatakan sebagai x_q=s di mana s adalah index kotak dan q adalah adalah yang menyatakan urutan pengambilan biji pada kotak dengan index x_1=1. Berikut ini adalah proses pembentukan pohon pada level 0 (akar) dan 1 :



Gambar 4 Pohon ruang solusi level 0 (akar), 1, dan 2

Setiap simpul memiliki informasi nilai simpul yang dimilikinya. Angka pada sisi menyatakan indeks kotak yang akan diambil bijinya.

Pada kondisi di atas, simpul 3 terkena fungsi pembatas karena isi dari kotak nomor 1 kosong. Simpul ini mati dan tidak dapat dipergunakan. Simpul 5, 6, 7, dan 8 jatuh pada kotak lawan. Hal ini menyebabkan giliran harus ganti dan simpul tersebut ditandai sebagai kandidat solusi. Simpul 4 yang tidak ditandai tersebut kemudian diperluas :



Gambar 5 Pohon ruang solusi level 0 (akar), 1, dan 2

Pada tahap ini, simpul 9 dan 10 berisi kotak kosong, sehingga simpul tersebut terkena fungsi pembatas. Sedangkan simpul 11, 12, 13, dan 14 ditandai sebagai kandidat solusi karena jatuh di kotak lawan.

Untuk memperjelas terjadinya fungsi penanda (marking function) dan fungsi pembatas (bounding function) berikut ini akan digambarkan kondisi kotak dan Mancala pemain pada setiap simpul.

Tabel 1 Kondisi kotak mancala pada setiap simpul

No. Simpul	Kondisi	#Mancala	Ket.
1	4 4 4 4 4 4	0	Kondisi awal
2	0 5 5 5 5 5	0	Kotak 1 dipilih
3	**0** 5 5 5 5 5	0	Kotak 1 kosong, terkena B(3)
4	0 0 6 6 6 6	1	
5	0 5 0 6 6 6	1	Jatuh di tempat lawan, ditandai M(5)
6	0 5 5 0 6 6	1	Jatuh di tempat lawan, ditandai M(6)
7	0 5 5 5 0 6	1	Jatuh di tempat lawan, ditandai M(7)
8	0 5 5 5 5 0	1	Jatuh di tempat lawan, ditandai M(8)
9	**0** 0 6 6 6 6	1	Kotak 1 kosong, terkena B(9)
10	0 **0** 6 6 6 6	1	Kotak 2 kosong, terkena B(10)
11	0 0 0 7 7 7	2	Jatuh di tempat lawan, ditandai M(11)
12	0 0 6 0 7 7	2	Jatuh di tempat lawan, ditandai M(12)
13	0 0 6 6 0 7	2	Jatuh di tempat lawan, ditandai M(13)
14		2	Jatuh di
	0 0 6 6 6 0		tempat lawan, ditandai M(14)

Dengan demikian kandidat solusi dapat ditentukan, yaitu himpunan yang digambarkan pada tabel 2.

Tabel 2 Himpunan Kandidat Solusi (SC)

Jalur	Nilai simpul
1 → 2 → 3	1
1 → 2 → 4	1
1 → 2 → 5	1
1 → 2 → 6	1
1 → 2 → 4 → 11	2
1 → 2 → 4 → 12	2
1 → 2 → 4 → 13	2
1 → 2 → 4 → 14	2

Sesuai dengan properti fungsi pemilihan (selecting function), maka solusi yang dipilih adalah solusi dengan jalur 1 → 2 → 4 → 11 karena memiliki nilai simpul yang terbesar dan biji yang jatuh ke tempat lawan terkecil (2 buah).

Pohon di atas adalah penyederhanaan (hanya mengambil ruang solusi pada pengambilan pertama kotak pertama). Ruang solusi yang sesungguhnya akan jauh lebih besar namun menjamin optimisitas dari permasalahan yang didefinisikan.

5. KESIMPULAN

Algoritma EMS adalah algoritma perbaikan secara signifikan dari algoritma BFS konvensional. EMS dapat menjadi alternatif dari algoritma Branch and Bound terutama jika kemungkinan kandidat solusi cukup besar, dan dibutuhkan penanganan khusus dalam mematikan, atau menjadikan sebuah node menjadi kandidat solusi. Skema ini tidak diakomodir oleh algoritma Branch and Bound konvensional. Skema algoritma EMS yang unik tersebut kemudian dapat digunakan untuk menyelesaikan permasalahan salah satu permainan Mancalu, Bantumi.

REFERENSI

[1] Munir, Rinaldi. "Diktat Kuliah IF3051 Strategi Algoritma". Institut Teknologi Bandung. 2009

[2] http://www.andybell.ch/e_bantumi.htm
Waktu Akses : 04-01-2009 15:03

[3] http://www.tradgames.org.uk/games/Mancala.htm
Waktu Akses : 04-01-2009 15:30

[4] http://en.wikipedia.org/wiki/Mancala
Waktu Akses : 04-01-2009 15:45