Penentuan Keputusan dalam Permainan Gomoku dengan Program Dinamis dan Algoritma *Greedy*

Atika Yusuf 13510055
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13510055@std.stei.itb.ac.id

Abstract—**Gomoku is a strategy-based board game also named Gobang or Five in a Row with white and black stones on a 15x15 board. Gomoku is a game for two and the winner is the first player to place five of his stones in a row horizontally, vertically, or diagonally.**

***Index Terms*— gomoku, dynamic programming, greedy algorithm**

I. INTRODUCTION

Pada mata kuliah IF3051 Strategi Algoritma diajarkan beberapa algoritma yang dapat digunakan untuk menyelesaikan masalah dalam programming maupun kehidupan sehari-hari. Algoritma ini antara lain adalah: algoritma *brute force, greedy, BFS, DFS, backtrack, dynamic programming.*

Salah satu persoalan kehidupan sehari-hari yang dapat diselesaikan dengan algoritma adalah *board games. Board games* sangat populer karena merupakan permainan yang membutuhkan ketelitian, menuntut kreatifitas, dan strategi.

Permainan catur adalah salah satu contoh *board games* yang membutuhkan strategi yang baik, sama halnya dengan permainan gomoku. Tantangan dari permainan gomoku adalah ukuran papan yang lebih besar dari permainan tic tac toe. Ketelitian dibutuhkan untuk mendeteksi apakah ada batu lawan yang sudah tersusun berurutan sekaligus menyusun batu kita sendiri. Oleh karena itu diperlukan algoritma yang dapat menganalisis kondisi di papan dan memberikan keputusan yang terbaik.

Beberapa dekade yang lalu, pemain professional gomoku menyebutkan bahwa gomoku dapat dimenangkan oleh pemain dengan giliran pertama. Selain itu ada *paper* yang khusus membahas bagaimana cara memenangkan permainan gomoku dengan teknik *Threat-Space Search* oleh L.V Allis H.J. van den Herik dan M.P.H. Huntjens. Karena teknik tersebut diluar *scope* kuliah, penulis hanya akan membahas penentuan keputusan terbaik.

II. GOMOKU

Gomoku adalah permainan strategi yang disebut juga Gobang atau *Five in a* Row yang merupakan permainan untuk dua orang. Pemain dapat menempatkan batu sesuai warna miliknya (hitam atau putih) pada papan 15x15 dan batu yang sudah diletakkan tidak dapat dipindahkan lagi. Objektif dari permainan ini adalah berlomba-lomba menempatkan batunya sejajar lima buah secara diagonal, horizontal, atau diagonal.

Tantangan dari permainan ini sama dengan permainan Tic Tac Toe, dimana pemain dihadapkan antara dua tantangan pada setiap giliran: berusaha menyusun batunya sejajar atau menghadang batu pemain lain. Ketelitian sangatlah diperlukan oleh karena itu memenangkan permainan ini lebih sulit dilakukan dengan komputer daripada manusia. Pada hakikatnya jika batu lawan sudah sejajar tiga buah dengan lowongan kosong dikiri-kanannya, pemain harus menghadang batu lawan karena jika tidak kondisi pemain di giliran selanjutnya mungkin saja tidak aman.

III. PROGRAM DINAMIS

A. Definisi

Program dinamis adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahap sehingga solusi dari persoalan dapat dilihat dari serangkaian keputusan yang saling berkaitan. Karakteristik penyelesaian persoalan dengan program dinamis

1. terdapat sejumlah berhingga pilihan yang mungkin,
2. solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangan pada suatu tahap.

Perbedaaan algoritma greedy dengan program dinamis adalah algoritma greedy hanya menghasilkan satu rangkaian keputusan sedangkan program dinamis menghasilkan satu rangkaian keputusan dengan mempertimbangkan banyak rangkaian keputusan.

B. Solusi

Algoritma program dinamis dapat membantu pengambilan keputusan, hanya saja dalam permainan Gomoku sulit ditentukan langkah mana yang memiliki profit lebih besar karena tidak ada cost seperti graf dan selain itu maksimal ada 132 tahap. Jadi dalam penyelesaian yang saya tawarkan ada dua pilihan dalam tiap langkah:

1. Berusaha menyusun batu berurutan
2. Berusaha menghadang batu

Kemudian kedua pilihan tersebut dapat ditentukan costnya dengan situasi tertentu, cost yang mungkin adalah 10, dan 20. Cost paling kecillah yang terbaik. Berikut kondisi yang menentukan cost sebuah pilihan:

a. Batu pemain bisa ditambah/bebas
b. Batu musuh sudah ada yang berurutan tiga buah dan belum dihadang
c. Kondisi a dan b muncul bersamaan

Alasan mengapa kondisi b muncul adalah karena ketika kondisi tersebut terjadi, pemain masuk ke dalam *state* yang tidak aman dimana jika batu musuh yang berurutan tidak segera dihadang, pada giliran selanjutnya batu musuh pasti sudah berurutan empat buah dan pemain tidak mungkin dapat menghadang musuh lagi kecuali diujung urutan batu musuh tersebut sudah ada batu pemain.

Berikut cost yang didapat berdasarkan pilihan dan kondisi:

Table 1

Kondisi	Pilihan	
	1	2
a	10	20
b	20	10
c	10	10

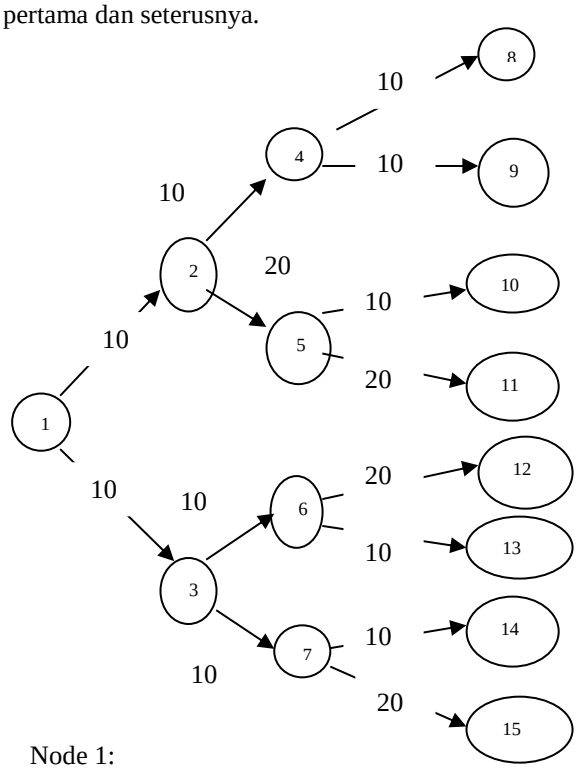
Pseudo-code:

```
function GetMinCost(State s) → integer
{fungsi yang menerima state board
gomoku dan mengembalikan keputusan
dengan cost minimum (keputusan
terbaik)}

ALGORITMA
      if (State=a) {bisa mengurutkan
batu sendiri}
            return 1
      endif

      else if (State=b) {batu musuh
berurutan tiga then}
            return 2
      endelse

      else {State=c}
            return 1 or 2
      endelse
```

Dalam permainan Gomoku pada http://gomoku.yjyao.com memilki intelegensi buatan yang kira-kra hampir sama dengan keputusan yang dibuat pada rancangan program dinamis diatas,. Pada bahasan ini, diasumsikan langkah pemain dan langkah musuh adalah satu kesatuan yang direpresentasikan sebagai satu node dalam sebuah graf.

C. Contoh Penyelesaian

Misal tahap direpresentasikan sebagai graf dengan node 1 merupakan langkah pertama dengan meletakkan batu pemain tepat di tengah papan. Lalu node 2 dan 3 merupakan representasi pilihan 1 dan 2 setelah langkah pertama dan seterusnya.



Node 1:

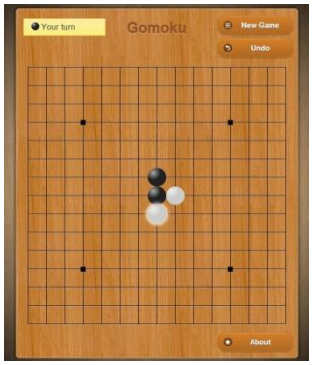
Sumber: gomoku.yjyao.com

Tahap 1

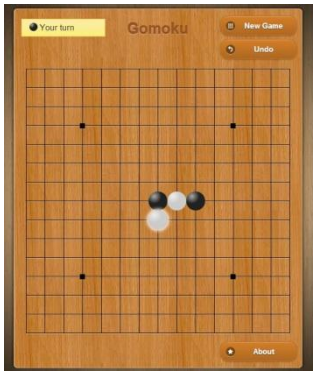
S	Solusi optimum	
	$F_1(s)$	X_1

2	10	1
3	10	1

Node 2:



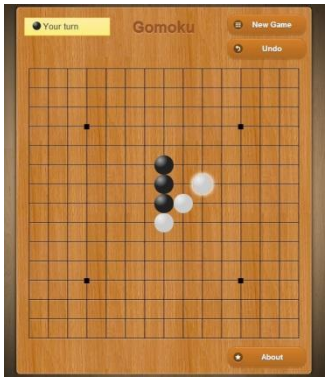
Node 3:



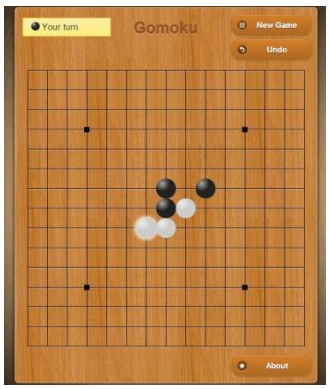
Tahap 2

X_2	$F_2(x_2,s)= C_{x2}s+f_1(x_2)$		Solusi optimum	
S	2	3	$F_2(s)$	X_2
4	20	∞	20	2
5	30	∞	30	2
6	∞	20	20	3
7	∞	20	20	3

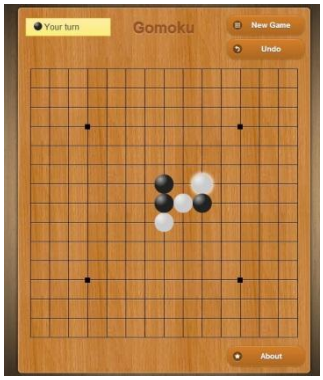
Node 4:



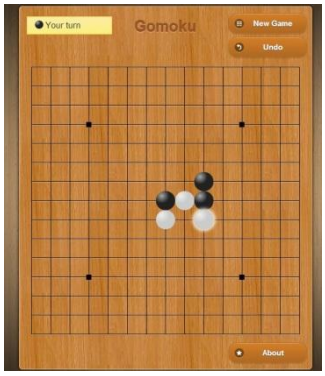
Node 5:



Node 6:



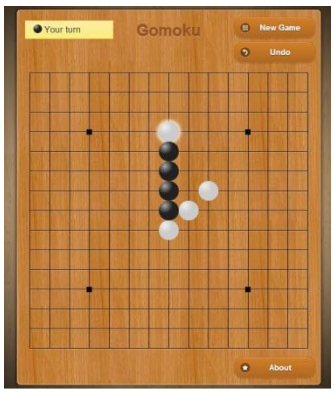
Node 7:



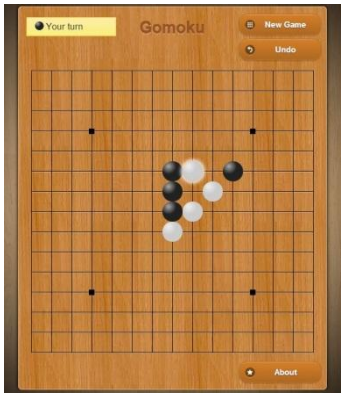
Tahap 3

X_2	$F_2(x_3,s)= C_{x3}s+f_2(x_3)$				Solusi optimum	
S	4	5	6	7	$F_3(s)$	X_3
8	30	∞	∞	∞	30	4
9	30	∞	∞	∞	30	4
10	∞	40	∞	∞	40	5
11	∞	50	∞	∞	50	5
12	∞	∞	40	∞	40	6
13	∞	∞	30	∞	30	6
14	∞	∞	∞	30	30	7
15	∞	∞	∞	40	40	7

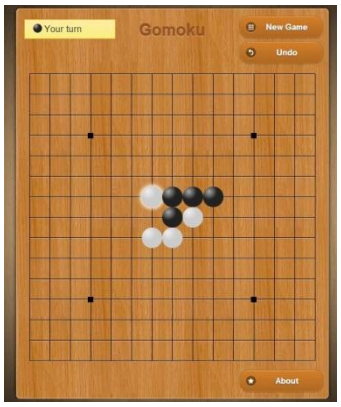
Node 8:



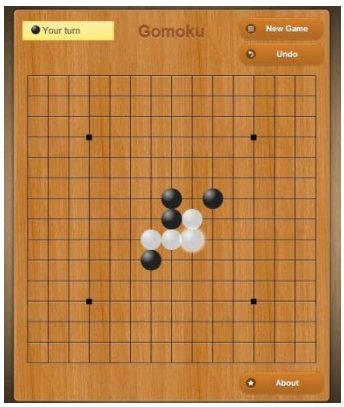
Node 9:



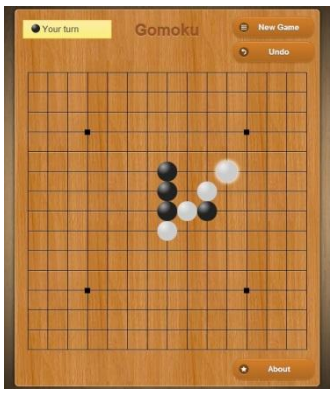
Node 10:



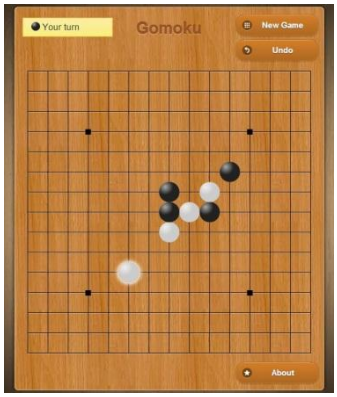
Node 11:



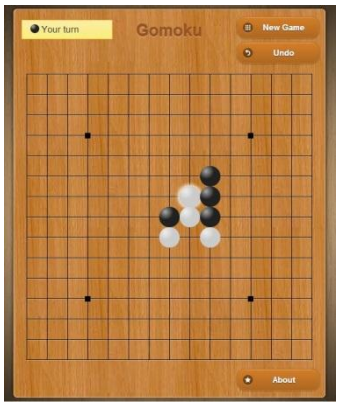
Node 12:



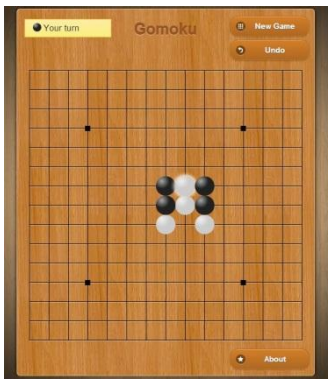
Node 13:



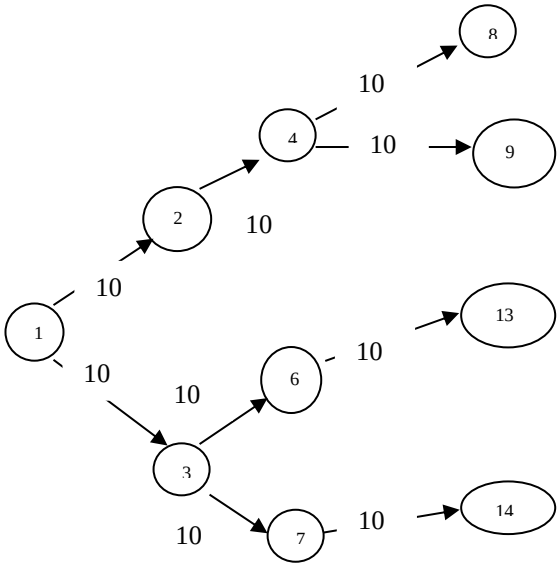
Node 14:



Node 15:



Node-node diatas tidak dilanjutkan karena akan menghasilkan tahap yang terlalu banyak. Saya anggap bahwa optimum lokal dapat memenuhi optimum global sesuai dengan prinsip program dinamis. Dari fungsi rekursif diatas, dapat disimpulkan bahwa tahap terbaik yang dapat diambil adalah:



IV. ALGORITMA GREEDY

A. Definisi

Algoritma *greedy* adalah metode untuk memecahkan persoalan optimasi dengan prinsip "*take what you can get now"*. Algoritma *greedy* membentuk satu buah solusi dari beberapa langkah. Pada setiap langkah algoritma *greedy* akan mengambil pilihan terbaik pada saat itu tanpa memperhatikan langkah selanjutnya. Algortima *greedy* hampir sama dengan program dinamis dalam hal memilih optimum lokal pada setiap langkah untuk mendapatkan optimum global, hanya saja tidak ada jaminan algoritma *greedy* akan mendapatkan solusi paling optimum.

B. Solusi

Solusi dengan algoritma *greedy* menggunakan table 1 dan juga menggunakan fungsi GetMinCost sebagai fungsi obyektif.

- Himpunan kandidat = {pilihan 1, pilihan 2}
- Himpunan solusi = pilihan yang sesuai kondisi
- Fungsi seleksi: minimum dari GetMinCost()
- Fungsi layak: langkah yang diambil tidak menyebabkan musuh langsung menang
- Fungsi obyektif: langkah yang diambil minimum

Pseudo-code

```
function greedy(input
C:himpunan_kandidat)→himpunan_kandidat
{mengembalikan himpunan solusi dari
persoalan gomoku sesuai dengan kondisi
board dan pilihan}

Deklarasi
      x: kandidat
      S: himpunan_kandidat

ALGORITMA
      S ← {}  {inisialisasi S}
      while ((not GAMEOVER) and (C ≠
{})) do
            x ← SELEKSI(C)
            C ← C - {x}
            if LAYAK(S U {x}) then
                  S ← S U {x}
            endif
      endwhile

      return S
```

C. Contoh Penyelesaian

Dengan algoritma *greedy*, disini diasumsikan jika ada cost minimum yang sama maka akan diprioritaskan pilihan 1 yaitu menyusun batu pemain berurutan (supaya lebih cepat menang) daripada menghadang batu musuh. Maka solusi yang dihasilkan dengan contoh yang sama dengan kasus pada program dinamis adalah node {1, 2, 4, 8} atau dalam pilihan {1, 1, 1, 1}.

V. KESIMPULAN

Program dinamis jika dibandingkan dengan algoritma *greedy* kurang mangkus dan lebih memakan banyak waktu karena ada banyak sekali kemungkinan. Algoritma *greedy* mungkin tidak menghasilkan hasil yang optimum tetapi lebih mudah dalam penentuan keputusannya. Selain itu penentuan cost pilihan masih cenderung subjektif sehingga tidak menjamin akan tercapainya kondisi yang diinginkan. Dengan kata lain, diperlukan teknik yang lebih mendalam dari algoritma *greedy* untuk memenangkan permainan gomoku dengan intelegensi buatan pada gomoku.yjyao.com. Diperlukan juga algoritma yang merupakan serangkaian keputusan seperti program dinamis yang dapat menjebak musuh.

REFERENCES

[1] http://gomoku.yjyao.com/ diakses pada 21 Desember 2012
[2] http://home.mit.bme.hu/~gtakacs/download/allis1994.pdf diakses pada 21 Desember 2012
[3] Munir, Rinaldi, *Diktat Kuliah IF3051 Strategi Algoritma*, Penerbit Informatika : Bandung, 2009

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 21Desember 2012

Atika Yusuf