ANALISIS PERBANDINGAN ALGORITMA DYNAMIC WEIGHTING A* DAN ALGORITMA SIMPLIFIED MEMORY-BOUNDED A* (SMA*) UNTUK MENCARI LANGKAH OPTIMAL DALAM PENYELESAIAN PERMAINAN PEG SOLITAIRE

Fandy Agung Prasetyo[1]

[1] Teknik Informatika – Univesitas Komputer Indonesia
Jl. Dipatiukur 112 - 114 Bandung
E-mail : fandyagungprasetyo.co.id[1]

ABSTRAK

Permainan peg solitaire merupakan salah satu board game populer di dunia yang dapat dimainkan oleh satu orang pemain. Permainan asah otak ini mempunyai aturan untuk mengeksekusi beberapa kelereng menjadi satu buah kelereng dengan melewati atau menepati jalur yang tidak terisi oleh kelereng lain. Kesalahan yang sangat mungkin terjadi pada permainan ini adalah adanya dua atau tiga peg yang letaknya saling berjauhan. Hal ini terjadi karena kesalahan dalam menentukan langkah dan menyebabkan permainan tidak dapat terselesaikan.

Kecerdasan buatan terdapatdua teknik pencarian yaitu pencarian buta dan pencarian terbimbing (heuristic). Algoritma yang dapat digunakan dalam pencairan terbimbing dua diantaranya adalah algoritma Dynamic Weighting A* (DWA*) dan algoritma Simplified Memory-Bounded A* (SMA*). Algoritma DWA* adalah algoritma pencarian rute terpendek yang baik dan dapat memberikan solusi yang tepat dengan penggunaan memori yang kecil. Sedangkan algoritma SMA* memiliki kelebihan yaitu dapat melakukan pencarian dengan cepat dan membutuhkan memori yang lebih kecil dibandingkan dengan algoritma A*, maka dilakukan perbandingan antara algoritma DWA* dan algoritma SMA* untuk mencari langkah optimal dalam menyelesaikan permainan peg solitaire untuk menganalisis algoritma mana yang lebih baik antara algoritma DWA* dan algoritma SMA*.

Berdasarkan hasil pengujian, untuk kasus model papan permainan dengan jumlah peg yang lebih sedikit yaitu pada papan plus, eight, dan fireplace algoritma DWA* dan SMA* masih dapat menemukan solusi. Akan tetapi, untuk kasus papan pyramid, arrow, dan solitaire algoritma DWA* dan SMA* tidak dapat menemukan solusi karena memori yang dibutuhkan melebihi kapasitas. Dilihat berdasarkan waktu penyelesaian algoritma DWA* lebih unggul dalam kecepatan daripada algoritma SMA* untuk menyelesaikan permainan peg solitaire untuk kasus pada papan plus, fireplace dan eight.

Kata kunci : Kecerdasan Buatan, DWA*, SMA*, *Peg Solitaire.*

1. PENDAHULUAN

Menentukan keberhasilan dari sebuah kecerdasan buatan (artificial intelligence) adalah kesuksesan dalam pencarian dan pencocokan[1]. Pada kecerdasan buatan terdapat 2 teknik pencarian yaitu pencarian buta (blind search) dan pencarian terbimbing (heuristic). Pencarian dengan teknik blind search merupakan teknik pencarian solusi tanpa syarat sedangkan heuristic search merupakan teknik pencarian solusi yang menggunakan aturan-aturan yang ditentukan pada rumus heuristic-nya.

Beberapa algoritma yang dapat digunakan dalam pencarian terbimbing dua diantaranya adalah algoritma Dynamic Weighting A* (DWA*) dan algoritma Simplified Memory-Bounded A* (SMA*). Algortima Dynamic Weighting A* (DWA*) adalah algoritma pencarian rute terpendek yang baik. Dalam penyelesaian masalah rute terpendek, algoritma penyelesaian Shortest Path Solution ini merupakan algoritma yang optimal (menemukan rute terpendek) dan complete (selalu menemukan solusi jika solusi ada)[2]. Pada kasus penentuan lokasi parkir pada smart parking system, algoritma DWA* dapat memberikan solusi yang tepat dengan waktu eksekusi yang cepat dan penggunaan memori yang kecil[3]. Sedangkan algoritma Simplified Memory-Bounded A* (SMA*) merupakan algoritma pencarian jalur terpendek dan merupakan pencarian heuristic. Algoritma ini memiliki kelebihan yaitu dapat melakukan pencarian dengan cepat dan

membutuhkan memori yang lebih kecil dibandingkan dengan algoritma A*. Untuk kasus pencarian lintasan terpendek dan efisiensi tarif perjalanan antar kota, algoritma SMA* bisa mengoptimalkan pencarian dengan menggunakan solusi yang telah ada sebelumnya[4].

Terdapat perbedaan algoritma DWA* dan algoritma SMA* dalam menyelesaikan masalah dan belum adanya penelitian dalam lingkup analisis performansi untuk membandingkan kedua algoritma ini khususnya pada permainan peg solitaire menjadi dasar penelitian ini. Oleh karena itu, kedua algoritma tersebut akan dilakukan perbandingan untuk mengukur performansi dalam kecepatan dan ketepatan.

Permainan peg solitaire merupakan salah satu board game populer di dunia yang hanya dapat dimainkan oleh satu orang pemain. Peg Solitaire dimainkan dengan menggunakan beberapa set kelereng dan papan yang berlubang. Dalam permainan ini kelereng, atau disebut sebagai peg, diisi sehingga membentuk berbagai macam model permainan. Peg solitaire adalah permainan asah otak atau logika yang menggunakan papan yang berukuran 7x7 yang biasa disebut dengan papan English Style. Pada permainan asah otak ini mempunyai aturan untuk mengeksekusi beberapa kelereng menjadi satu buah kelereng dengan melewati atau menempati jalur yang tidak terisi oleh kelereng lain. Kesalahan yang sangat mungkin terjadi pada permainan ini adalah adanya dua atau tiga peg yang letaknya saling berjauhan. Hal ini terjadi karena kesalahan dalam menentukan langkah dan menyebabkan permainan tidak dapat terselesaikan.

Atas dasar uraian yang telah dijelaskan tersebut, maka dilakukan perbandingan antara algoritma DWA* dan algoritma SMA* khususnya pada pencarian solusi untuk mencari langkah optimal dalam menyelesaikan permainan peg solitaire untuk menganalisis algoritma mana yang lebih baik antara algoritma DWA* dan algoritma SMA* dari segi kecepatan terutama pada model papan, jumlah kelereng dan arah pencarian solusi.

1.1 Rumusan Masalah

Berdasarkan latar belakang yang diuraikan diatas, permasalahan yang dapat rumuskan adalah bagaimana bagaimana menganalisis perbandingan algoritma DWA* dan algoritma SMA* untuk dapat mengidentifikasi langkah yang optimal untuk menyelesaikan permainan peg solitaire.

1.2 Maksud dan Tujuan

Berdasarkan permasalahan yang diteliti, maka maksud dari penelitian ini adalah untuk membandingkan algoritma Dynamic Weighting A* dan algoritma Simplified Memory-Bounded A* pada permainan peg solitaire.

Tujuan dari penelitian ini adalah :

1. Mendapatkan solusi dalam menyelesaikan masalah pada permainan peg solitaire.

2. Mengetahui performansi dalam kecepatan dan ketepatan dari perbandingan algoritma DWA* dan algoritma SMA* untuk mencari langkah yang optimal dalam menyelesaikan permainan peg solitaire.

2. ISI PENELITIAN

2.1 Algoritma

Dalam dunia komputasi, istilah algoritma menjadi dasar pemikiran sebuah formulasi. Algoritma dapat didefinisikan sebagai teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas tetapi tersusun secara logis dan sistematis. Kalimat-kalimat ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Algoritma harus berhenti setelah mengerjakan serangkaian tugas atau langkahnya terbatas dan setiap langkah harus didefinisikan dengan tepat sehingga tidak memiliki arti ganda. Algoritma sering mempunyai langkah pengulangan (iterasi) atau memerlukan keputusan (logika Boolean dan perbandingan) sampai tugasnya selesai. Desain dan analisis algoritma adalah suatu cabang khusus dalam ilmu computer yang mempelajari karakteristik dan performansi dari suatu algoritma dalam menyelesaikan masalah.

2.2 Artificial intelligence (AI)

Sebagian kalangan menerjemahkan Artificial Intelligence sebagai kecerdasan buatan, kecerdasan artifisial, intelijensia artifisial, atau intelejensia buatan. Istilah Artificial Intelligence sengaja tidak diterjemahkan ke bahasa Indonesia karena istilah tersebut sudah sangat akrab bagi orang Indonesia. Begitu juga dengan singkatan istilah tersebut, yaitu AI sudah sangat melekat di berbagai media ilmiah maupun non ilmiah.

Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia

saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia.

2.3 Dynamic Weighting A* (DWA*)

Algoritma DWA* adalah salah satu varian dari algoritma A* dan merupakan metode pencarian heuristik, asumsi untuk melakukan pencarian ke segala arah pada awal iterasi dan ketika goal state sudah dekat baru pencarian difokuskan kearah goal state merupakan karakteristik dari algoritma ini. Untuk itu digunakan pembobotan yang dinamis terhadap fungsi heuristik. Dalam notasi matematika dinyatakan seperti pada persamaan 2.1 berikut :

$$f(n) = g(n) + (w(n) * h(n)) \quad (2.1)$$

Dengan :

f(n) = hasil perhitungan dari *g(n)*, *h(n)*, dan *w(n)*

n = simpul saat ini

g(n) = biaya (cost) dari simpul awal ke simpul n sepanjang jalur pencarian

h(n) = perkiraan cost dari simpul n ke simpul tujuan (nilai heuristik)

w(n) = bobot dinamis

2.3.1 Array Open

Komponen array yang pertama adalah array Open, array ini berisi state atau suksesor yang dibangkitkan dan akan diperiksa. Dilakukan perulangan untuk memeriksa apakah berisi Goal State atau tidak, jika berisi Goal State maka state tersebut akan dihapus dari Open dan akan dimasukkan ke Closed dan proses pencarian selesai. Jika bukan, maka setiap tetangga atau state yang dibangkitkan dan terdapat di Open akan diperiksa apakah lebih baik atau tidak, state terbaik akan dihapus dari Open dan dimasukkan ke Closed.

State yang sudah berada di Open juga akan dilakukan pengecekan terhadap parent, yaitu terhadap nilai g(n) dari parent tersebut. Jika ada parent baru yang menghasilkan nilai g(n) yang lebih baik, maka akan dilakukan update parent. Nilai g(n) dan h(n) dari state tersebut juga akan diperbaharui, hal ini membuat state tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai state terbaik.

2.3.1 Array Closed

State yang sudah terpilih dan merupakan state yang lebih baik atau Best Node dibandingkan dengan tetangganya akan dimasukkan ke array Closed. State yang pernah berada di array Closed juga akan dilakukan pemeriksaan terhadap parentnya, apakah perlu ada perubahan atau tidak. Perubahan yang dilakukan juga akan berpengaruh terhadap nilai g(n) dan f(n) state sebelumnya. Sama halnya dengan array Open, perubahan parent tersebut juga akan menjadikan state tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai rute terbaik.

2.4. Algoritma Simplified Memory-Bounded A* (SMA*)

Untuk masalah tertentu, di mana memori komputer terbatas, algoritma A* mungkin tidak mampu menemukan solusi karena sudah tidak tersedia memori untuk menyimpan simpul-simpul yang dibangkitkan. Algoritma IDA* dapat digunakan untuk kondisi seperti ini karena IDA* hanya membutuhkan sedikit memori. Tetapi, satu kelemahan IDA* adalah bahwa pencarian yang dilakukan secara iteratif akan membutuhkan waktu yang lama karena harus membangkitkan simpul berulang kali.

Berlawanan dengan IDA* yang hanya mengingat satu f-limit, algoritma SMA* mengingat f-Cost dari setiap iterasi sampai sejumlah simpul yang ada di dalam memori. Karena batasan memori dalam jumlah tertentu, maka dapat membatasi pencarian hanya sampai pada simpul-simpul yang dapat dicapai dari root sepanjang suatu jalur yang memorinya masih mencukupi. Kemudian mengembalikan suatu rute terbaik diantara rute-rute yang ada dalam batasan jumlah simpul tersebut. Jika memori komputer hanya mampu menyimpan 100 simpul, maka bisa membatasi proses pencarian sampai level 99.

Pada algoritma SMA* terdapat sebuah queue yang digunakan untuk memanipulasi antrian simpul yang terurut berdasarkan f-cost. Di sini yang dimaksud f-cost adalah gabungan biaya sebenarnya dan biaya perkiraan, yang secara matematika dinyatakan seperti pada persamaan 2.3 berikut:

$$f(n) = g(n) + h(n) \quad (2.3)$$

Dengan:

n = simpul saat ini

g(n) = biaya (cost) dari simpul awal ke simpul *n* sepanjang jalur pencarian

h(n) = perkiraan cost dari simpul *n* ke simpul tujuan (nilai heuristik)

f(n) = total cost dari simpul *n* ke simpul tujuan

2.5 Permainan Peg Solitaire

Peg Solitaire merupakan permainan yang cukup populer sejak dulu. Menurut beberapa orang, permainan ini pertama kali diciptakan oleh seorang aristokrat Perancis pada abad ke 17. Namun, hal ini masih menjadi sebuah perdebatan di tahun 1800-an. Salah satu orang yang mencari pembuktian ini bernama John Beasley. John Beasley mengemukakan bahwa permainan ini bukanlah berasal dari Perancis, melainkan salah satu warisan turun- temurun dari suku asli Amerika. Walaupun pernyataannya ini juga belum memiliki bukti yang cukup kuat. Permainan ini terkenal di beberapa negara dan memiliki nama yang berbeda-beda pula tergantung tempat permainan ini dimainkan. Misalnya di Inggris, permainan ini bernama "Solitaire". Sedangkan permainan kartu yang selama ini kita kenal dengan nama "Solitaire" disebut sebagai "Patience" di Inggris. Di India, permainan ini dikenal dengan nama "Brainvita". Dalam permainan ini kelereng, atau disebut sebagai peg, diisi secara penuh pada papan permainan, kecuali pada lubang bagian tengah. Gambar 2.1 adalah contoh jenis papan Inggris.

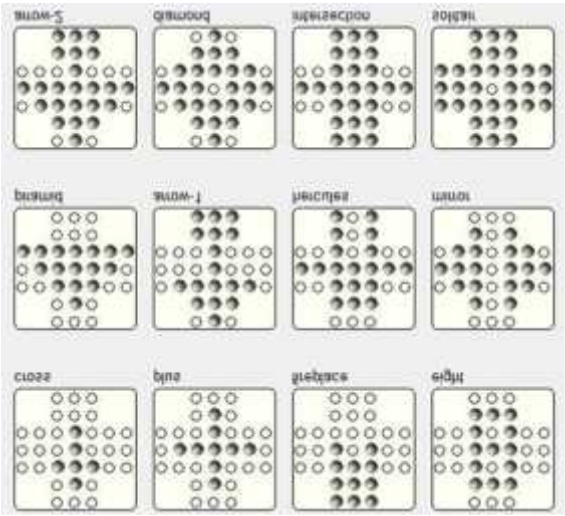


Gambar 2.1 Jenis Papan *Peg Solitaire* Inggris.

Permainan ini sendiri berkembang di seluruh dunia dengan nama yang berbeda dan unik, di Indonesia sendiri nama tersebut adalah asah otak dengan bermacam jenis bentuk permainannya. Namun dengan perkembangan teknologi maka permainan ini sudah mulai punah dikarenakan permainan sudah dianggap kuno dan akhirnya teknologi yang memegang peran dalam permainan modern.

Permainan ini pemainan yang individual dikarenakan hanya satu pemain saja yang dapat memainkannya, dari segi bentuk papan yang bervariasi dan permainan merupakan permainan yang individual. Permainan ini mengasah logika para pemain, menentukan langkah dalam menghabisikan kelereng, dalam menghabiskan jumlah kelereng yang ada dalam papan tersebut. Pemain akan dihadapi oleh kesulitan-kesulitan dalam penentuan langkah, hal ini berdampak pada segi waktu yang cukup lama dalam hal berfikir untuk menentukan langkah permainan.

Penggunaan model papan pada peg solitaire yang bervariasi sehingga pada penggunaan jumlah kelereng dan aturan cara permainannya berbeda-beda. Dalam analisis ini, digunakan papan English style dikarenakan standar permainan yang digunakan dalam asah otak dalam permainan ini. Gambar 2.2 adalah contoh model-model papan permainan dari peg solitaire.



Gambar 2.2 Variasi model papan pada permainan *peg solitaire.*

2.6 Fungsi Heuristik

Di dalam metode yang termasuk heuristic search, fungsi heuristik memainkan peranan yang sangat menetukan. Suatu fungsi dapat diterima sebagai fungsi heuristik jika biaya perkiraan yang dihasilkan tidak melebihi dari biaya sebenarnya. Ketika fungsi heuristik memberikan biaya perkiraan yang melebihi biaya sebenarnya (overestimate), maka proses pencarian bisa tersesat dan membuat heuristic search menjadi tidak optimal.

Suatu fungsi heuristik dikatakan baik jika bisa memberikan biaya perkiraan yang mendekati biaya sebenarnya, fungsi heuristik tersebut semakin baik.

Berikut ini adalah salah satu contoh fungsi heuristik yang dipakai pada masalah permainan peg solitaire, antara lain:

$h(n) = Ps - Pf$ (2.4)

Dengan :

Ps = Nilai Node Awal

Pf = Nilai Node Tujuan

Nilai Ps dan Pf adalah nilai yang tertera di dalam masing-masing peg yang ada didalam papan dan sudah di inisialisasi sebelumnya atau biasa disebut pagoda function. Gambar 2.3 berikut ini menunjukkan nilai dari masing-masing peg :

		−0.3	0.4	0		
		1.0	0	1.0		
0.5	0	0.5	0.4	0.1	0.3	−0.1
0	0.9	0.7	0.3	0.9	1.1	0.4
0.5	0.6	0.1	0.5	0.2	0.6	0.2
		0.8	0	0.8		
		0	0.5	−0.2		

Gambar 2.3 Pagoda Functions untuk papan *peg solitaire.*

2.7. C# *(C Sharp)*

C# (C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya (seperti Java, Delphi, Visual Basic, dan lain- lain) dengan beberapa penyederhanaan. Menurut standar ECMA-334 C# Language Specification, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan angka # (U+0023). Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik (U+266F), dan tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam keyboard standar.

2.7. Teknik Pengujian Perangkat Lunak

Ujicoba software merupakan elemen yang kritis dari SQA dan merepresentasikan tinjauan ulang yang menyeluruh terhadap spesifikasi,desain dan pengkodean. Ujicoba merepresentasikan ketidaknormalan yang terjadi pada pengembangan software. Selama definisi awal dan fase pembangunan, pengembang berusaha untuk membangun software dari konsep yang abstrak sampai dengan implementasi yang memungkin.

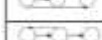
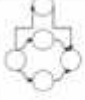
Para pengembang membuat serangkaian uji kasus yang bertujuan untuk ”membongkar” software yang mereka bangun. Kenyataannya, ujicoba merupakan salah satu tahapan dalam proses pengembangan software yang dapat dilihat (secara psikologi) sebagai destruktif, dari pada sebagai konstruktif.

2.7.1. Ujicoba Berbasis Alur (*Basis Path Testing*)

Ujicoba berbasis alur merupakan teknik ujicoba white box pertama yang diusulkan oleh Tom McCabe. Metode berbasis alur memungkinkan perancang kasus uji untuk menghasilkan ukuran kompleksitas logikal dari desain prosedural dan menggunakan ukuran ini untuk mendefinisikan himpunan basis dari alur eksekusi. Kasus uji dihasilkan untuk melakukan sekumpulan basis yang dijamin untuk mengeksekusi setiap perintah dalam program, sedikitnya satu kali selama ujicoba.

2.7.2. Notasi Graf Alur (*Path Graph Notation*)

Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah notasi diagram alir (atau grafik program), yang menggunakan notasi lingkaran (simpul atau node) dan anak panah (link atau edge). Notasi ini menggambarkan aliran control logika yang digunakan dalam suatu bahasa pemrograman. Gambar 2.4 dibawah ini adalah skema yang digunakan untuk menggambarkan notasi dari graf alur.

Notasi	Arti
	Skema Sequence
	Skema If
	Skema While (...) DO (...)
	Skema Repeat (...) Until (...)
	Skema Case (...) Of

Gambar 2.4 Notasi Diagram Alur

2.7.3. *Cyclomatic Complexity*

Cyclomatic complexity merupakan software metric yang menyediakan ukuran kuantitatif dari komplesitas logikal suatu program. Ketika digunakan dalam konteks metode ujicoba berbasis alur, nilai yang dikomputasi untuk kompleksitas cyclomatic mendefinisikan jumlah independent path dalam himpunan basis suatu program dan menyediakan batas atas untuk sejumlah ujicoba yang harus dilakukan untuk memastikan bahwa seluruh perintah telah dieksekusi sedikitnya satu kali.

Independent path adalah alur manapun dalam program yang memperkenalkan sedikitnya satu kumpulan perintah pemrosesan atau kondisi baru. Contoh independent path :

Path 1 : 1 – 11

Path 2 : 1 – 2 – 3 – 4 – 5 – 10 – 1 – 11

Path 3 : 1 – 2 – 3 – 6 – 8 – 9 – 10 – 1 – 11

Path 4 : 1 – 2 – 3 – 6 – 7 – 9 – 10 – 1 – 11

Misalkan setip path yang baru memunculkan edge yang baru, dengan path:

1 - 2 – 3 – 4 – 5 – 10 - 1 - 2 – 3 – 6 – 8 – 9 – 10 – 1 – 11

Path diatas tidak dianggap sebagai independent path karena kombinasi path diatas telah didefinisikan sebelumnya Ketika ditetapkan dalam graf alur, maka independent path harus bergerak sedikitnya 1 edge yang belum pernah dilewati sebelumnya. Kompleksitas cyclomatic dapat dicari dengan salah satu dari 3 cara berikut :

1. Jumlah region dari graf alur mengacu kepada kompleksitas cyclomatic
2. Kompleksitas cyclomatic untuk graf alur G didefinisikan:

$$V(G) = E – N + 2$$

Dimana E = jumlah edge, dan N = jumlah node

3. Kompleksitas cyclomatic untuk graf alur G didefinisikan:

$$V(G) = P + 1$$

Dimana P = jumlah predicates nodes

Berdasarkan hasil independent path diatas, maka kompleksitas cyclomatic-nya dapat di hitung sebagai berikut:

1. Flow graph diatas mempunyai 4 region
2. V(G) = 11 edges – 9 nodes + 2 = 4
3. V(G) = 3 predicates nodes + 1 = 4

Hasil kompleksitas cyclomatic menggambarkan banyaknya path dan batas atas sejumlah ujicoba yang harus dirancang dan dieksekusi untuk seluruh perintah dalam program.

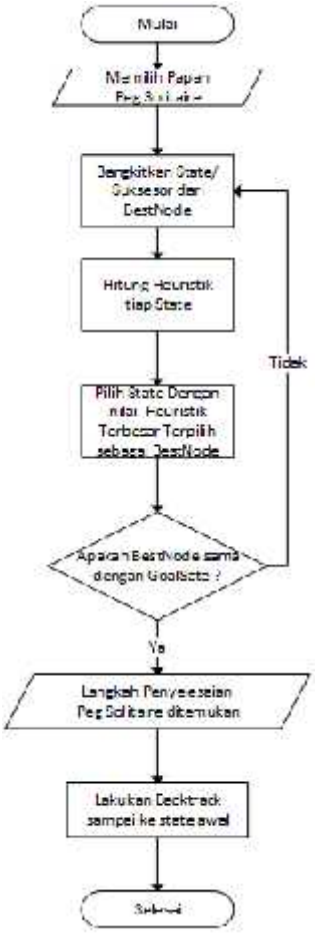
3. Analisis Sistem

3.1. Analisis Masalah

Permasalahan yang muncul pada penelitian ini adalah bagaimana algortima Dynamic Weighting A* dan algoritma Simplified Memory-Bounded A* mengindentifikasi langkah-langkah optimal dalam menyelesaikan permainan peg solitaire dan bagaimana mengukur performansi algortima Dynamic Weighting A* dan algoritma Simplified Memory-Bounded A* dari segi kecepatan dan ketepatan, untuk mengetahui algoritma mana yang lebih mangkus dalam menyelesaikan permainan peg solitaire, algoritma dapat dikatakan mangkus jika algoritma tersebut efektif dan efisisen. Suatu algoritma dikatakan efektif jika algoritma tersebut dapat menghasilkan solusi yang sesuai dengan masalah yang diselesaikan. Suatu algortima juga dikatakan efisien jika dapat meminimalkan kebutuhan waktu dan ruang dalam mencapai goal state yang diinginkan.

3.2 Analisis Algoritma DWA*

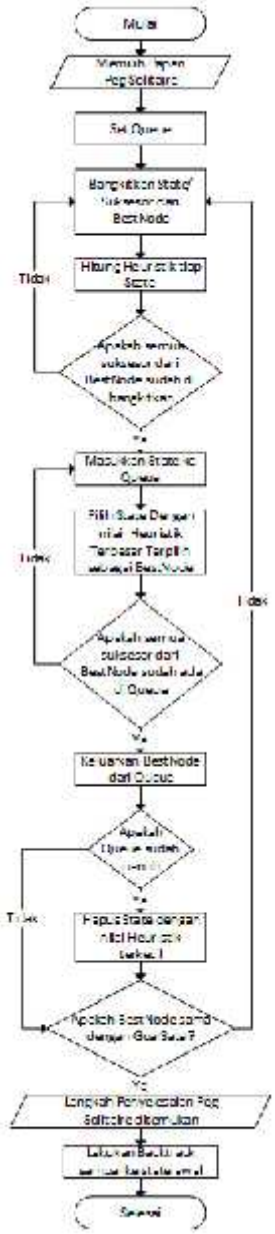
Analisis data keluaran adalah tahap akhir yang akan dilakukan. yaitu mencari nilai terbesar terhadap nilai probabilitas yang didapatkan pada saat analisis pengujian. berikut adalah alur proses mencari data keluaran. Algoritma DWA* menggunakan suatu *flowchart* yang berisi langkah-langkah manual dalam menyelesaikan permainan peg solitaire. Gambar 3.1 berikut ini menunjukkan langkah-langkah manual untuk penyelesaian permainan peg solitaire menggunakan algoritma DWA*.



Gambar 3.1 *Flowchart* Proses Penyelesaian Permainan Peg Solitaire menggunakan Algoritma DWA*.

3.3. Analisis Algoritma Simplified Memory-Bounded A* (SMA*)

Algoritma SMA* menggunakan suatu *flowchart* yang berisi langkah-langkah manual dalam menyelesaikan permainan peg solitaire. Gambar 3.2 berikut ini menunjukkan langkah-langkah manual untuk penyelesaian permainan peg solitaire menggunakan algoritma SMA*.



Gambar 3.2 *Flowchart* Proses Penyelesaian Permainan Peg Solitaire menggunakan Algoritma SMA*.

4. Implementasi dan Pengujian Sistem

4.1. Implementasi Sistem

Tahap implementasi sistem merupakan tahap pembuatan perangkat lunak, tahap lanjut dari tahap perancangan sistem. Tahap yang dilakukan untuk menerjemahkan perancangan berdasarkan hasil analisis dalam bahasa yang dimengerti oleh komputer serta penerapan perangkat lunak pada keadaan yang sebenarnya. Implementasi yang dilakukan antara lain menerapkan hasil analisis dan perancangan perangkat keras, perangkat lunak, dan antarmuka.

4.1.1. Implementasi Antarmuka

1. Antarmuka Utama

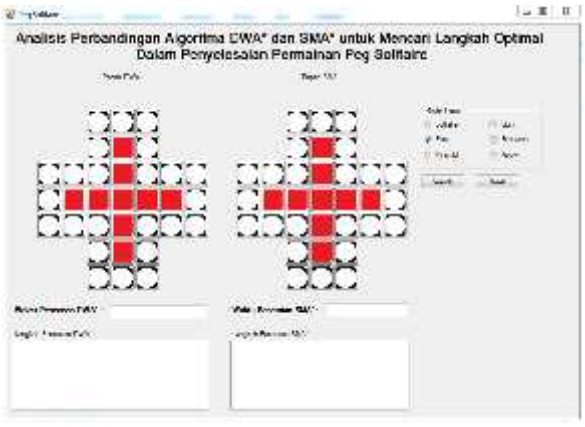
Antarmuka ini merupakan tampilan awal saat aplikasi dijalankan. Gambar 4.1 merupakan tampilan antarmuka utama system untuk implementasi algoritma *Dynamic Weighting A** (DWA*) dan algoritma *Simplified Memory-Bounded A** (SMA*) untuk mencari langkah optimal dalam penyelesaian permainan peg solitaire.



Gambar 4.1 Antarmuka Utama Papan Peg Solitaire.

2. Antarmuka Pilih Papan Permainan

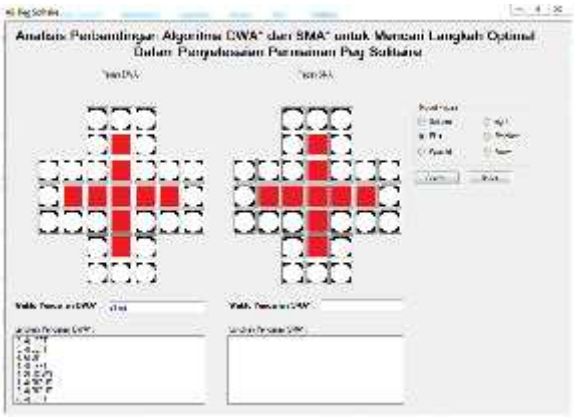
Antarmuka ini merupakan tampilan setelah pengguna menekan salah satu model papan Plus di RadioButton pada GroupBox Model Papan. Gambar 4.2 berikut ini adalah tampilan setelah user memilih papan permainan.



Gambar 4.2 Antarmuka Memilih Papan Permainan Peg Solitaire

3. Antarmuka Pencarian menggunakan Algoritma DWA*

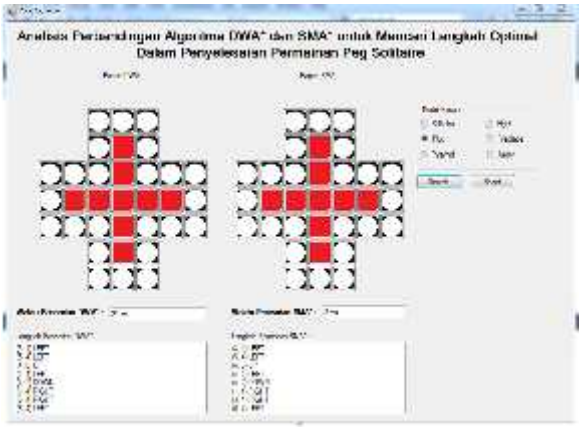
Antarmuka ini merupakan tampilan setelah pengguna menekan tombol Search. Gambar 4.3 berikut ini adalah tampilan pencarian menggunakan algoritma DWA*.



Gambar 4.3 Antarmuka Pencarian menggunakan Algoritma DWA*.

4. Antarmuka Pencarian menggunakan Algoritma SMA*

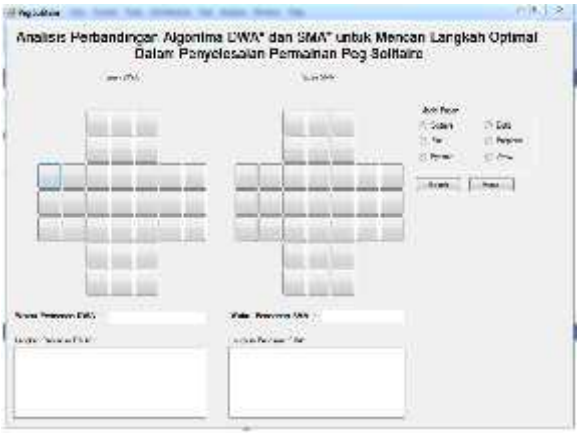
Antarmuka ini merupakan tampilan setelah pengguna menekan tombol *Search*. Gambar 4.4 berikut ini adalah tampilan pencarian menggunakan algoritma SMA*.



Gambar 4.4 Antarmuka Pencarian menggunakan Algoritma SMA*.

5. Antarmuka Hapus Papan Permainan

Antarmuka ini merupakan tampilan setelah pengguna menekan tombol *Reset*. Gambar 4.5 berikut ini adalah tampilan hapus papan permainan.



Gambar 4.5 Antarmuka Hapus Papan Permainan.

4.2. Kesimpulan Pengujian

Berdasarkan hasil pengujian, untuk kasus model papan permainan dengan jumlah peg yang lebih sedikit yaitu pada papan plus, eight, dan fireplace algoritma DWA* dan SMA* masih dapat menemukan solusi. Akan tetapi, untuk kasus papan pyramid, arrow, dan solitaire algoritma DWA* dan SMA* tidak dapat menemukan solusi karena memori yang dibutuhkan melebihi kapasitas. Dilihat berdasarkan waktu penyelesaian algoritma DWA* lebih unggul dalam kecepatan daripada algoritma SMA* untuk menyelesaikan permainan *peg solitaire* untuk kasus pada papan plus, fireplace dan eight.

5. Kesimpulan dan Saran

5.1. Kesimpulan

Berdasarkan hasil yang didapat dari penelitian secara keseluruhan, dapat ditarik kesimpulan sebagai berikut:

1. Algoritma DWA* dan SMA* dapat menemukan solusi untuk menyelesaikan permainan peg solitaire hanya untuk kasus model papan plus, eight, dan fireplace. Akan tetapi, untuk kasus papan pyramid, arrow, dan solitaire algoritma DWA* dan SMA* tidak dapat menemukan solusi karena memori yang dibutuhkan melebihi kapasitas.
2. Dilihat berdasarkan waktu penyelesaian, algoritma DWA* lebih unggul dalam kecepatan daripada algoritma SMA* untuk menyelesaikan permainan *peg solitaire* untuk kasus pada papan plus, fireplace dan eight.

5.2. Saran

Adapun saran yang dapat diberikan untuk pengembangan selanjutnya antara lain adalah sebagai berikut:

1. Menggunakan metode heuristik yang lain untuk memangkas pencarian sehingga memori yang dibutuhkan untuk melakukan pencarian solusi mencukupi.

6. DAFTAR PUSTAKA

Russell, S.J and Peter Norvig, P. 2010. *Artificial Intelligence: A modern Approach.* Third Edition.New Jersey:Pearson Education.

Suyanto, Artificial Intelligence Searching, Reasoning, Planning dan Learning, Bandung: Informatika Bandung, 2014.

Christian R, "Penentuan Lokasi Parkir Pada Smart Parking System Menggunakan Dynamic Weighting A*(DWA*)," *Skripsi Teknik Informatika, Universitas Telkom,* 2011.

Yoswara Y, "Penerapan Algoritma Simplified-Memory-Bounded A* dan Algoritma Greedy-Best First Search dalam Pencarian Lintasan Terpendek dan Efisiensi Tarif Perjalanan Antar Kota," *Jurnal Teknik Informatika, Institut Teknologi Bandung,* 2010.

R. S. Pressman, Rekayasa Perangkat Lunak Pendekatan Praktis, Yogyakarta: Andi, 2012.

Millington, I and Funge, J. 2009. *Artificial of Intelligence for Games.* Second Edition Massachusetts: Morgan Kaufmann Publishers.

I. Gilbran, "Algoritma Backtracking Pada Permainan Peg Solitaire," *Jurnal Teknik Informatika, Institut Teknologi Bandung,* 2010.

B.Hariyanto, Esensi-esensi Bahasa Pemrograman, Bandung: Informatika, 2014.

A. Nugroho, Rational Rose Untuk Pemodelan Berorientasi Objek, Bandung:Informatika, 2005.

Munawar, Pemodelan Visual dengan UML, Yogyakarta: Graha Ilmu, 2005.

K. Hamilton and R. Miles, Learning UML 2.0, Sebastopol: O'Reilly Media, Inc., 2006.

Artekom, (2012). Pengenalan C# (*C Sharp*). *Retreived from http://artekom.com/pengenalan-c-sharp/*

C#.NET Bagian 1. Pembuatan *Project* pada Visual Studio 2010 dengan C#.NET, (2013). *Retreived from http://www.wadinuryadi.com/2013/08/c-net-bagian-1-pembuatan-project-pada-visual-studio-2010-dengan-c-net.html*

Ayuliana, "Teknik Pengujian Perangkat Lunak," *Jurnal Teknik Informatika, Universitas Gunadarma*, 2009.

Kiyomi, M., dan Matsui, T.: Integer Programming Based Algorithms for Peg Solitaire Problems, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, Universitas Tokyo, 2001.