Perbandingan Perbedaan Heuristik pada Algoritma A* dalam Penyelesaian Labirin

Mahanti Indah Rahajeng 13517085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517085@std.stei.itb.ac.id

***Abstrak*—Permasalahan labirin sudah sering dibahas hingga saat ini. Penyelesaian labirin ini tentunya bisa didapatkan dengan menggunakan beragam cara dan pendekatan. Cara yang paling umum adalah memodelkan labirin menjadi sebuah graf. Lalu, graf ini akan ditelusuri dengan algoritma pencarian tertentu. Algoritma pencarian yang dapat digunakan juga bermacam-macam. Pada makalah ini, penulis akan membahas penyelesaian labirin dengan menggunakan algoritma A* dan membandingkan hasil dari perbedaan fungsi heuristik yang digunakan dalam penyelesaian labirin.**

***Kata Kunci*—heuristik; A*; labirin**

I. PENDAHULUAN

Graf merupakan suatu dasar teori yang digunakan dalam berbagai pemodelan permasalahan. Salah satunya dalam persoalan penyelesaian labirin. Bukan hanya dalam aplikasi permainan dan taman hiburan, persoalan labirin ini juga bahkan muncul pada sebuah pertandingan robot *micro maze solver*.

Gambar 1 Micromouse USA

Sumber : http://micromouseusa.com/?m=201509

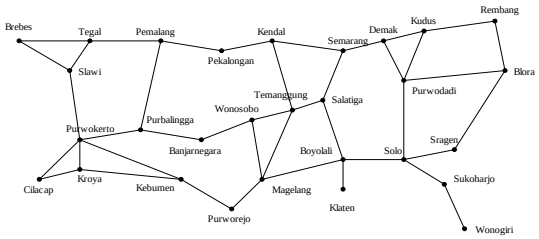
Dalam pencarian jalan keluar, labirin dapat dimodelkan menjadi sebuah graf sedemikian rupa sehingga dapat dilihat rute atau jalur dari titik masuk ke titik keluar. Banyak algoritma pencarian yang dapat digunakan untuk menyelesaikan labirin, seperti A*, BFS, dan DFS. Tentunya algoritma-algoritma tersebut memiliki karakteristiknya masing-masing dalam menyelesaikan masalah pencarian. Tidak seperti algoritma BFS dan DFS, algoritma A* memiliki informasi tambahan berbasis heuristik dalam proses pencarian jalur.

Dalam penyelesaian labirin, labirin itu sendiri akan dipresentasikan menjadi petak dua dimensi lalu dimodelkan dalam sebuah graf. Pencarian jalur akan dimulai dari titik awal tertentu dan berakhir pada titik tujuan. Pada tulisan ini, penulis akan membahas dan membandingkan perbedaan heuristik pada algoritma A* dalam penyelesaian labirin.

II. TEORI DASAR

A. *Graf*

Sebuah graf biasa digunakan untuk merepresentasikan objek-objek diskrit dan hubungan diantara objek-objek tersebut. Secara visual, objek dapat direpresentasikan menjadi sebuah simpul dan garis yang menghubungkan dua simpul menggambarkan hubungan antara kedua objek tersebut.



Gambar 2 Peta Jaringan Jalan Raya Provinsi Jawa Tengah

Sumber : [1]

Gambar 1 di atas merupakan sebuah peta jaringan jalan raya di Provinsi Jawa Tengah. Peta tersebut termasuk sebuah representasi nyata dari graf. Objek simpul adalah kota dan garis antara kedua kota menggambarkan adanya jalan yang menghubungkan keduanya.

Secara matematis, graf dapat didefinisikan sebagai pasangan himpunan (V, E) yang dapat ditulis dengan notasi sebagai berikut.

$$G = (V, E)$$

V adalah himpunan tidak kosong dari simpul dan E adalah himpunan sisi yang menghubungkan sepasang simpul. [1]

B. *Algoritma Path-Planning*

Secara singkat, algoritma path-planning adalah sebuah algoritma pencarian rute atau lintasan dari suatu titik awal ke titik tujuan tertentu. Algoritma pencarian ini secara garis besar dibagi menjadi dua, yaitu tanpa informasi (*uninformed search*) dan dengan informasi (*informed search*). Perbedaan dari kedua algoritma ini terletak pada ada tidaknya informasi tambahan

yang dimiliki saat melakukan pencarian. Tidak seperti *uninformed search*, algoritma *informed search* memiliki informasi tambahan berbasis heuristik. Contoh algoritma *uninformed search* adalah BFS dan DFS. Sedangkan contoh dari algoritma *informed search* adalah algoritma A* dan Greedy Best-First Search. [2]

C. Algoritma A*

Algoritma A* merupakah salah satu algoritma pencarian *informed search.* Ide dari algoritma A* ini adalah menghindari untuk menelusuri jalur yang sudah jelas tidak optimal atau memiliki *cost* yang besar. Fungsi evalusi yang digunakan adalah sebagai berikut.

$$f(n) = g(n) + h(n)$$

$g(n)$ adalah *cost* yang dikeluarkan untuk mencapai titik n. $h(n)$ adalah fungsi heuristik yang berupa estimasi *cost* dari titik n untuk mencapai titik tujuan. Sehingga $f(n)$ adalah total estimasi *cost* yang dikeluarkan untuk melalui jalur dari titik n ke titik tujuan.

Algoritma A* ini dinilai selalu menghasilkan solusi optimal dalam mencari jalur dengan total *cost* minimum. Kompleksitas waktu dari algoritma ini adalah $O(b^m)$. Kompleksitas ruang yang dimiliki juga $O(b^m)$ karena algoritma ini menyimpan semua simpul pada memori. [2]

D. Heuristik

Fungsi heuristik $h(n)$ memberi algoritma A* estimasi *cost* minimum dari sembarang titik n menuju titik tujuan. Pemilihan fungsi heuristik perlu diperhatikan karena heuristik yang digunakan mempengaruhi bagaimana algoritma A* akan berjalan.

Beberapa kasus yang mungkin terjadi dengan perbedaan fungsi heuristik adalah sebagai berikut.

- Jika $h(n)$ bernilai 0, maka $f(n) = g(n)$. Algoritma A* akan berjalan seperti algoritma Dijkstra.
- Jika $h(n)$ bernilai sangat besar dibandingkan dengan nilai $g(n)$, maka $f(n) = h(n)$. Algoritma A* akan berjalan seperti algoritma Greedy Best-First Search.
- Jika $h(n)$ bernilai kurang dari atau sama dengan *cost* dari titik n ke titik tujuan, maka algoritma A* dijamin dapat menemukan jalur terpendek. Semakin kecil nilai $h(n)$, pencarian akan semakin lama.
- Jika $h(n)$ bernilai lebih besar dibandingkan *cost* dari titik n ke titik tujuan, maka algoritma A* tidak dijamin dapat menemukan jalur terpendek. Akan tetapi, pencarian akan berjalan lebih cepat. [3]

E. Labirin

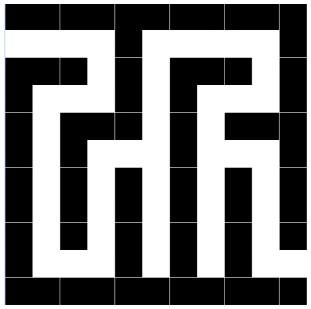
Labirin merupakan sebuah jalan atau lorong yang rumit, berliku-liku dan memiliki banyak jalan buntu. Labirin sederhana pada umumnya memiliki satu titik masuk dan satu titik keluar. Sangat memungkinkan untuk suatu labirin memiliki banyak jalur yang dapat ditempuh dari titik masuk menuju titik keluar. Permasalahan yang sering dibahas adalah bagaimana menemukan jalur terpendek untuk menuju titik keluar.

III. Implementasi Algoritma

Pada makalah kali ini, akan diberi batasan permasalahan yaitu labirin akan direpresentasikan menjadi *two-dimensional grids serta* gerakan pencarian yang diperbolehkan hanya gerakan empat arah yaitu atas, kanan, bawah, dan kiri.

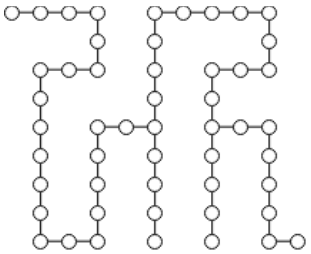
A. Translasi Labirin menjadi Graf

Diberikan contoh labirin sebagai berikut.



Gambar 3 Contoh Labirin

Gambar 3 memperlihatkan bahwa petak yang berwarna hitam adalah halangan dan petak yang berwarna abu-abu adalah jalan yang dapat dilalui. Mentranslasikan labirin di atas menjadi sebuah graf cukup mudah. Setiap petak yang berwarna abu-abu akan ditandai sebagai satu simpul. Tiap-tiap petak abu-abu yang bertetangga akan membentuk garis diantaranya. Sehingga akan terbentuk graf sebagai berikut.



Gambar 4 Labirin dalam Graf

B. Variasi Heuristik

Beberapa fungsi heuristik yang akan dibandingkan yaitu

1. *Manhattan distance*

```
function heuristic(node, goal) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return (dx + dy)
```

2. *Euclidean distance*

```
function heuristic(node, goal) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return sqrt(dx * dx + dy * dy)
```

3. *Tie-breaking scaling*

```
function heuristic(current, start,
goal) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    heuristic = dx + dy
    heuristic *= 1.001
    return heuristic
```

4. *Tie-breaking cross-product*

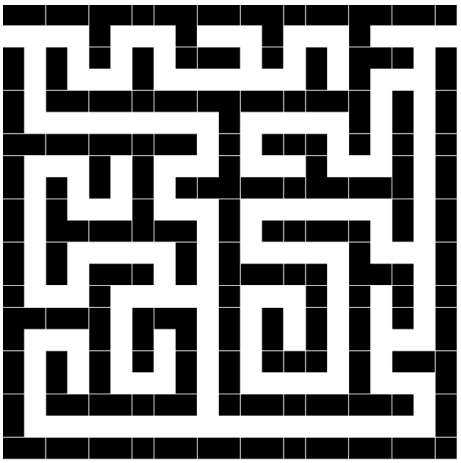
```
function heuristic(current, start,
goal) =
    dx1 = current.x - goal.x
    dy1 = current.y - goal.y
    dx2 = start.x - goal.x
    dy2 = start.y - goal.y
    heuristic = abs(dx1) + abs (dy1)
    cross = abs(dx1*dy2 - dx2*dy1)
    heuristic += cross*0.001
    return heuristic
```

C. *Source Code Algoritma A* dalam Bahasa Python*

```
def AStar():
  global jalur, matriks, queue
  global masuk, keluar
  found = False
  x = masuk
  y = 0
  while not found:
jalur.append([x,y])
matriks[x][y] = 2
if x==keluar and y==len(matriks[0])-1:
  found = True
else:
  gn = len(jalur)
  if (matriks[x][y+1] == 0):
   y += 1
   queue.append([x,y,gn+hn(x,y),jalur])
  if (matriks[x+1][y] == 0):
   x += 1
   queue.append([x,y,gn+hn(x,y),jalur])
  if (matriks[x-1][y] == 0):
   x -= 1
   queue.append([x,y,gn+hn(x,y),jalur])
  if (matriks[x][y-1] == 0):
   y -= 1
  queue.append([x,y,gn+hn(x,y),jalur])
queue.sort(key=takeLast)
x = queue[0][0]
y = queue[0][1]
jalur = queue[0][3]
queue.pop(0)
```

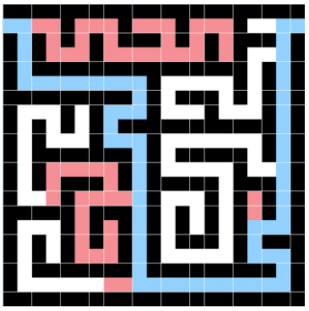
IV. HASIL DAN ANALISIS

A. *Test Map 1 (21 x 21)*

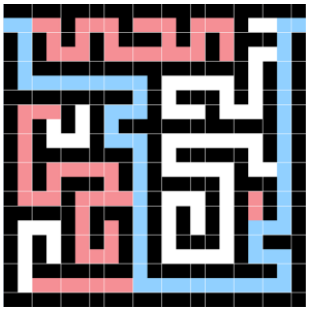


Gambar 5 Labirin Test 1

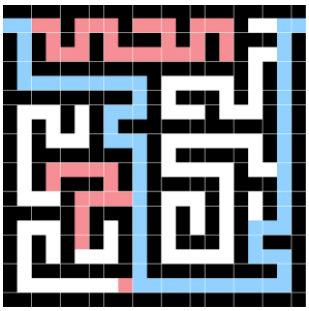
Hasil Pengujian:



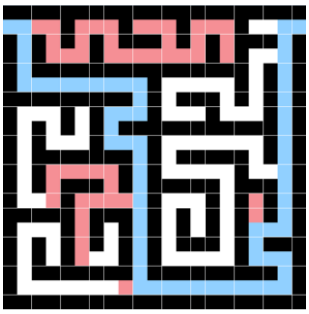
Gambar 6 Manhattan distance



Gambar 7 Euclidean distance



Gambar 8 Tie-breaking scaling



Gambar 9 Tie-breaking cross-product

Manhattan distance

Waktu	: 0.03358912467956543 detik
Jumlah langkah	: 117
Lintasan terpendek	: 65

Euclidean distance

Waktu	: 0.07155203819274902 detik
Jumlah langkah	: 132
Lintasan terpendek	: 65

Tie-breaking scaling

Waktu : 0.032544851303100586 detik

Jumlah langkah : 108

Lintasan terpendek : 65

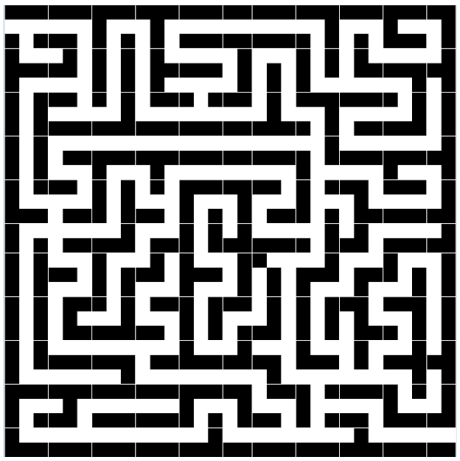
Tie-breaking cross-product

Waktu : 0.031914472579956055 detik

Jumlah langkah : 112

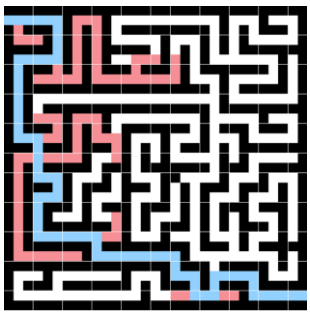
Lintasan terpendek : 65

B. *Test Map 2 (31 x 31)*

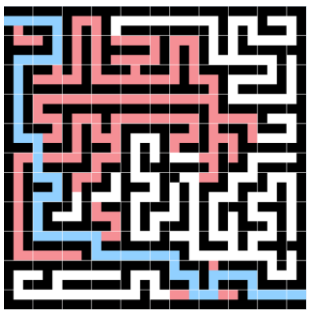


Gambar 10 Labirin Test 2

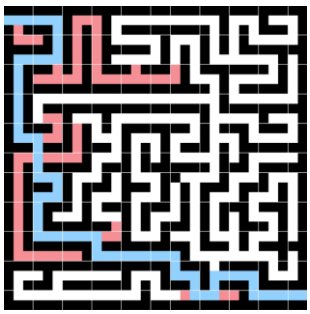
Hasil Pengujian:



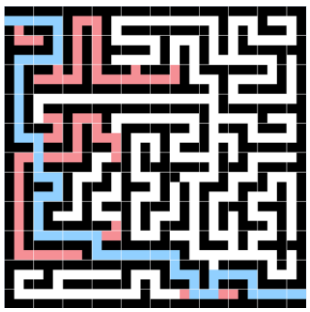
Gambar 11 Manhattan distance



Gambar 12 Euclidean distance



Gambar 13 Tie-breaking scaling



Gambar 14 Tie-breaking cross-product

Manhattan distance

Waktu : 0.05662989616394043 detik

Jumlah langkah : 161

Lintasan terpendek : 75

Euclidean distance

Waktu : 0.0658254623413086 detik

Jumlah langkah : 267

Lintasan terpendek : 75

Tie-breaking scaling

Waktu : 0.05699467658996582 detik

Jumlah langkah : 145

Lintasan terpendek : 75

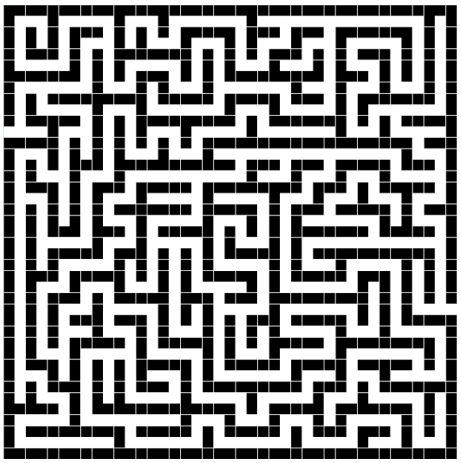
Tie-breaking cross-product

Waktu : 0.04549670219421387 detik

Jumlah langkah : 154

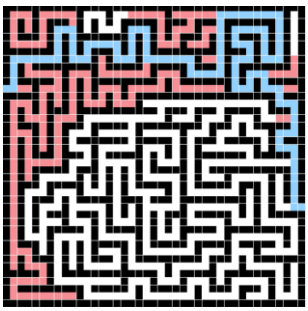
Lintasan terpendek : 75

C. *Test Map 3 (41 x 41)*

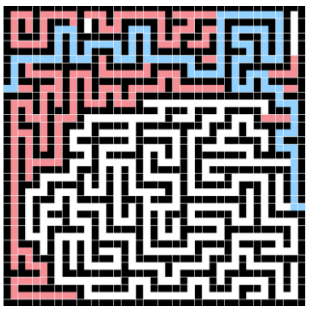


Gambar 15 Labirin Test 3

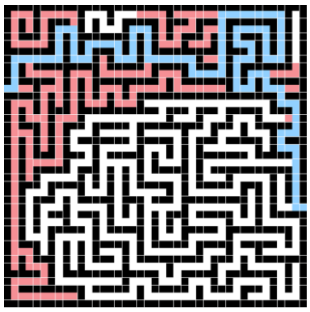
Hasil Pengujian:



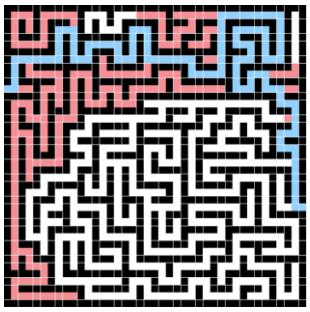
Gambar 16 Manhattan distance



Gambar 17 Euclidean distance



Gambar 18 Tie-breaking scaling



Gambar 19 Tie-breaking cross-product

Manhattan distance

Waktu : 0.11285543441772461 detik

Jumlah langkah : 337

Lintasan terpendek : 133

Euclidean distance

Waktu : 0.14263176918029785 detik

Jumlah langkah : 349

Lintasan terpendek : 133

Tie-breaking scaling

Waktu : 0.1311020851135254 detik

Jumlah langkah : 334

Lintasan terpendek : 133

Tie-breaking cross-product

Waktu : 0.11132669448852539 detik

Jumlah langkah : 334

Lintasan terpendek : 133

D. Tabel Perbandingan

Dari hasil pengujian di atas, didapatkan tabel perbandingan hasil uji sebagai berikut.

Tabel 1 Tabel Perbandingan Hasil Uji

Test Map	Manhattan distance			Euclidean distance			Tie-breaking scaling			Tie-breaking cross-product		
	Waktu	Jumlah Langkah	Lintasan Terpendek	Waktu	Jumlah Langkah	Lintasan Terpendek	Waktu	Jumlah Langkah	Lintasan Terpendek	Waktu	Jumlah Langkah	Lintasan Terpendek
1	0.033	117	65	0.071	132	65	0.032	108	65	0.0319	112	65
2	0.056	161	75	0.065	267	75	0.056	145	75	0.0454	154	75
3	0.112	337	133	0.142	349	133	0.131	334	133	0.111	334	133

E. Analisis

Dari tabel perbandingan hasil uji, dapat dilihat data waktu eksekusi, jumlah langkah yang dilalui dan lintasan terpendek dari tiap-tiap heuristik yang digunakan. Beberapa informasi yang dapat ditarik dari tabel di atas adalah sebagai berikut.

1. Apapun fungsi heuristik yang digunakan akan selalu didapatkan lintasan terpendek dari titik awal ke titik tujuan.
2. Waktu eksekusi yang diperlukan untuk fungsi heuristik *Euclidean distance* relatif paling besar daripada fungsi heuristik yang lain.
3. Jumlah langkah yang diperlukan untuk fungsi heuristik *Euclidean distance* juga relatif paling besar daripada fungsi heuristik yang lain.
4. Waktu eksekusi yang diperlukan untuk fungsi heuristik *Tie-breaking cross-product* relatif paling kecil daripada fungsi heuristik yang lain.
5. Jumlah langkah yang diperlukan untuk fungsi heuristik *Tie-breaking scaling* relatif paling kecil daripada fungsi heuristik yang lain.

KESIMPULAN

Dari hasil analisis tabel pengujian di atas, dapat dibuktikan bahwa algoritma A* selalu menghasilkan solusi yang optimal yaitu jalur terpendek dalam penyelesaian labirin. Selain itu, heuristik *Euclidean distance* memiliki *cost* yang paling besar. Hal ini dikarenakan fungsi heuristik tersebut menghitung nilai jarak asli dari titik n ke titik tujuan. Sedangkan batasan yang digunakan dalam makalah ini adalah gerakan pencarian yang hanya empat arah, dibandingkan dengan gerakan atas-kanan-bawah-kiri gerakan diagonal akan memakan *cost* yang lebih besar. Heuristik *tie-breaking* bekerja sangat baik pada kasus labirin makalah ini karena memang fungsi heuristik *tie-breaking* menyelesaikan masalah dari *grid-based map* yang cenderung memiliki nilai $f(n)$ sama sehingga program akan menelusuri seluruh lintasan dengan nilai $f(n)$ yang sama bukan malah memilih satu lintasan yang terbaik.

UCAPAN TERIMA KASIH

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa, yang telah melimpahkan rahmat dan karunia-Nya berupa waktu dan kesempatan, sehingga penulis dapat menyelesaikan penulisan makalah ini dengan tuntas.

Dalam penulisan makalah ini tentunya banyak pihak yang telah membantu dan memberikan bantun serta dorongan yang sangat berarti bagi penulis. Untuk itu, penulis menyampaikan rasa terima kasih kepada para dosen pengajar mata kuliah Strategi Algoritma, orang tua penulis, dan semua pihak yang ikut membantu dalam penyusunan makalah ini.

Penulis menyadari dalam pembuatan makalah ini masih terdapat kekurangan baik dalam sistematika penulisan maupun isi makalah. Maka dari itu, dengan terbuka penulis menerima saran dan kritik yang membangun atas penulisan makalah ini. Harapan penulis adalah semoga makalah ini bermanfaat bagi kita semua.

DAFTAR PUSTAKA

[1] R. Munir, *Matematika Diskrit, Edisi 3*, Bandung: Penerbit Informatika, 2010, ch 8. Diakses pada 25 April 2019.

[2] Slide Kuliah IF2211 Strategi Algoritma 2018/2019 : *Path-Planning*. Diakses pada 25 April 2019.

[3] *http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html*. Diakses pada 25 April 2019.

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, 25 April 2019

Mahanti Indah Rahajeng 13517085