IMPLEMENTASI METODE *GENERATE AND TEST* DALAM MENYELESAIKAN *TRAVELLING SALESMAN PROBLEM* MENGGUNAKAN ROBOT BERSENSOR SONAR DAN WARNA

Ridho Rahmadi
Jurusan Teknik Informatika, Fakultas Teknologi Industri,Universitas Islam Indonesia
Jl. Kaliurang Km. 14 Yogyakarta 55501
E-mail: ridho.rahmadi@staff.uii.ac.id

ABSTRAKS
Masalah pencarian dan pelacakan merupakan hal penting dalam menentukan keberhasilan sebuah sistem yang berdasarkan Kecerdasan Buatan. Salah satu yang cukup dikenal adalah metode Generate and Test yang merupakan satu dari beberapa model pencarian heuristik dalam terminologi Kecerdasan Buatan. Travelling Salesman Problem (TSP) atau juga dipahami sebagai pencarian jalur terpendek sering diimplementasikan dalam dunia nyata seperti permasalahan distribusi produk perusahaan, pembuatan jaringan kabel telepon, dan pembuatan PCB dalam dunia elektronika. Tujuan penelitian ini adalah mencoba mengimplementasikan konsep pencarian heuristik dengan metode Generate and Test melalui sebuah robot yang dilengkapi sensor sonar untuk membaca jarak, dan sensor warna untuk membaca jalur sehingga dapat menemukan jalur terpendek dalam kasus TSP. Salah satu alasan mengapa menggunakan robot adalah selain melihat perkembangan implementasi Kecerdasan Buatan yang telah meluas ke ranah robotika, penulis juga mencoba membuat bentuk lain dari penyelesaian TSP ini. Dari hasil penelitian ini didapatkan sebuah robot cerdas yang dapat membaca jarak antar titik menggunakan sensor sonar kemudian mengkalkulasi lintasan terpendek dan pada akhirnya melintasinya dengan membaca jalur menggunakan sensor warna.

Kata Kunci: pencarian heuristik, generate and test, travelling saleman problem, robot, sensor sonar, sensor warna.

1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia Kecerdasan Buatan atau *Artificial Intelligence* (AI) yang demikian cepatnya secara tidak langsung memunculkan atmosfer kompetisi yang begitu kuat diantara para peneliti di bidang AI temasuk di dalamnya para akademisi. Salah satu arah perkembangan AI saat ini adalah bidang robotika. Sebagai satu contoh adalah robot humanoid keluaran Honda bernama Asimo yang cukup membuat dunia terduduk kagum dengan kemampuannya berjalan, berlari dan menuruni tangga seperti manusia.

Salah satu kasus yang sering digunakan dalam terminologi AI adalah *Travelling Salesman Problem* (TSP). Dalam kasus ini dianalogikan seorang *salesman* ingin mengunjungi sejumlah kota. Solusinya adalah berupa rute terpendek dari semua jalur antar kota tersebut dengan aturan setiap kota hanya boleh dilewat satu kali.

Generate and Test adalah sebuah metode dari beberapa konsep pencarian heuristik. Metode ini melakukan mekanisme kerja dengan cara membangkitkan atau *generate* kemungkinan solusi, kemudian melakukan pengujian atau *test*. Selama belum ditemukan solusi atau masih ada kemungkinan solusi maka akan terus dilakukan pencarian solusi. Jika ditemukan solusi maka berhasil, jika tidak maka gagal.

Dalam hal ini penulis mencoba untuk membuat konsep robot yang dapat menyelesaikan TSP dengan metode *Generate and Test.*

1.2 Tujuan

Penelitian ini secara garis besar bertujuan untuk mengimplementasikan konsep pencarian heuristik dengan metode *Generate and Test* pada sebuah robot sehingga dapat menyelesaikan *Travelling Salesman Problem.*

2. DASAR TEORI

2.1 *Generate and Test*

Metode *Generate and Test* adalah sebuah metode dari beberapa konsep pencarian heuristik. Menurut Kusumadewi (2003), pada prinsipnya metode ini merupakan penggabungan antara *depth-first search* dengan pelacakan mundur (*backtracking*) yaitu bergerak ke belakang menuju pada suatu keadaan awal. Nilai pengujian berupa jawaban ‘ya’ atau ‘tidak’. Algoritma dari metode *Generate and Test* ini adalah:

1. Bangkitkan suatu kemungkinan solusi (membangkitkan suatu titik tertentu atau lintasan tertentu dari keadaan awal)
2. Uji untuk melihat apakah node tersebut benar-benar merupakan solusinya dengan cara membandingkan node tersebut atau node akhir dari suatu lintasan yang dipilih dengan kumpulan tujuan yang diharapkan.

3. Jika solusi ditemukan, keluar. Jika tidak, ulangi kembali langkah yang pertama.

2.2 Robot

Robot berasal dari kata "robota" yang dalam bahasa Ceko yang berarti budak, pekerja atau kuli. Pertama kali kata "robota" diperkenalkan oleh Karel Capek dalam sebuah pentas sandiwara pada tahun 1921 yang berjudul RUR (Rossum's Universal Robot). Pentas ini mengisahkan mesin yang menyerupai manusia yang dapat bekerja tanpa lelah yang kemudian memberontak dan menguasai manusia. Istilah "robot" ini kemudian mulai terkenal dan digunakan untuk menggantikan istilah yang dikenal saat itu, yaitu automaton (Nugroho, 2009). Menurut Budiharto (2009), beberapa penerapan robot saat ini antara lain :

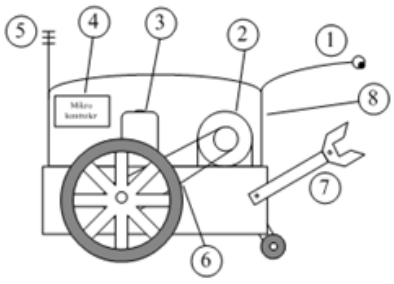
1. Merakit dan mengelas kerangka mobil di industri manufaktur.
2. Pencari dan pemadam sumber api.
3. Pelayan toko.
4. Robot boneka
5. Robot medis.
6. Robot perang.
7. *Multifunction Automated Crawling System* (MACS).
8. Robot penjelajah.
9. Robot intel.

Menurut pendapat Nugroho (2009), robot adalah sesuatu yang:

1. Dapat memperoleh informasi dari lingkungan.
2. Dapat diprogram.
3. Dapat melaksanakan beberapa tugas yang berbeda.
4. Bekerja secara otomatis.
5. Cerdas (Intelligent).
6. Digunakan di industri.

2.2.1 Anatomi Robot

Seperti halnya manusia, robot juga memiliki anatomi yang terdiri dari beberapa bagian yang menyusunnya. Bagian-bagian tersebut bekerja secara sinergis, sehingga dapat melakukan sebuah aktifitas tertentu. Gambar 2 menunjukkan skema anatomi robot.



Gambar 2. Anatomi robot

1. sensor
2. aktuator
3. catu daya
4. kontroler
5. sistem komunikasi
6. sistem pemindah
7. manipulator/end effector
8. rangka

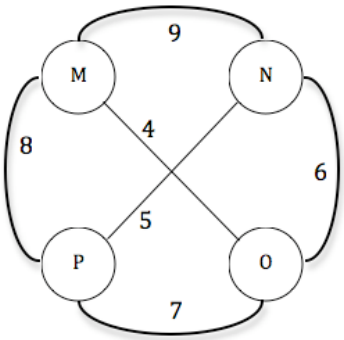
Di bawah ini adalah tabel 1 yang menunjukkan ekivalensi antara anatomi robot dengan anatomi manusia (Nugroho, 2009).

Tabel 1. Ekivalensi komponen robot dengan anggota tubuh manusia

Robot	*Manusia*
sensor	panca indera
aktuator	Otot
catu daya	sistem pencernaan
kontroler	Otak
sistem komunikasi	mulut dan telinga
sistem pemindah	kaki
manipulator/end effector	lengan/tangan
rangka	tulang
program	pikiran

3. PEMBAHASAN

Penelitian ini mencoba menyelesaikan kasus TSP dengan jumlah empat titik kota. Kit robot yang penulis gunakan adalah Mindstorms NXT 2.0 produksi LEGO. Robot beroda ini dilengkapi dengan sensor sonar yang akan digunakan membaca jarak antarkota dan sensor warna yang digunakan untuk membaca jalur antarkota. *Software* yang digunakan untuk memrogram robot tersebut adalah ROBOTC, menggunakan bahasa pemrograman C. Gambar 3 di bawah ini menggambarkan kasus *Travelling Salesman Problem* yang akan diselesaikan.



Gambar 3. Kasus TSP

3.1 Inisialisasi Jarak dan Jalur Oleh Robot

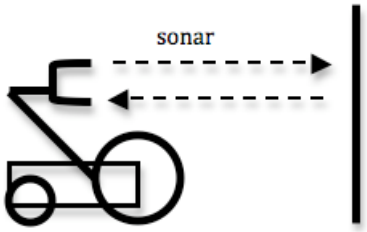
Secara teknis, tahap pertama adalah memastikan robot untuk mendapatkan nilai jarak antarkota. Gambar 3 di atas menunjukkan kasus TSP yang akan diselesaikan, satuan jaraknya dalam inci guna mengakomodasi simulasi pemecahan kasus yang

 ISSN: 1907-5022

dilakukan oleh robot. Dalam membaca jarak dan membedakan masing-masing titik kota, maka digunakan sebuah tiang berpapan disetiap kota dan memberikan warna yang berbeda antar kota. Untuk mendapatkan jarak tiap kota, yang pertama robot harus mengenali terlebih dahulu masing-masing titik kota. Dalam hal ini penulis menggunakan sensor warna untuk membaca warna jalur, dengan detail sebagai berikut:

1. Jalur M-N atau N-M, warna kuning
2. Jalur N-O atau O-N, warna hijau
3. Jalur O-P atau P-O, warna biru
4. Jalur P-M atau M-P, warna merah
5. Jalur M-O atau O-M, warna hitam
6. Jalur N-P atau P-N, warna abu-abu muda

Langkah selanjutnya adalah menginisialisasi jarak antar titik. Posisi robot untuk pertama kali harus ditentukan, yaitu di posisi titik M. gambar 4 menunjukkan posisi awal robot.



Gambar 4. Sensor sonar membaca jarak

Berikut ini adalah urutan-urutan perpindahan robot untuk menginisialisasi jarak tiap-tiap titik:

1. Robot di titik M, menghadap papan di titik N. Sensor sonar membaca jarak M-N = 9 inci.
2. Robot di titik M, berputar 45 derajat arah kanan menghadap papan di titik O. Sensor sonar membaca jarak M-O = 4 inci.
3. Robot di titik M, berputar 45 derajat lagi ke arah kanan menghadap papan di titik P. Sensor sonar membaca jarak M-P = 8 inci.
4. Robot berjalan dari M ke P lewat jalur M-P.
5. Robot di titik P, berputar ke kiri 45 derajat menghadap papan di titik N. Sensor sonar membaca jarak P-N = 5 inci.
6. Robot di titik P, berputar ke kanan 90 derajat menghadap papan di titik O. Sensor sonar membaca jarak P-O = 7 inci.
7. Robot berjalan dari P ke O lewat jalur P-O.
8. Robot di titik O, berputar ke kiri 90 derajat menghadap papan di titik N. Sensor sonar membaca jarak O-N = 6 inci.

Gambar 5 menunjukkan kode program yang memerintahkan robot untuk membaca jarak antar titik dan berpindah tempat.

```
task main(){
int MN, MO, MP, PN, PO, ON;
nSyncedMotors = synchBC;
SensorValue[sonar] = MN;
nSyncedTurnRatio = -100;
nMotorEncoder[motorB] = 0;
nMotorEncoderTarget[motorB] = -90;
motor[motorB] = -35;
while(nMotorRunState[motorB] != runStateIdle);
SensorValue[sonar] = MO;
nSyncedTurnRatio = -100;
nMotorEncoder[motorB] = 0;
nMotorEncoderTarget[motorB] = -90;
motor[motorB] = -35;
while(nMotorRunState[motorB] != runStateIdle);
SensorValue[sonar] = MP;
nSyncedTurnRatio = 100;
motor[motorB] = 35;
while(SensorValue[warna] == 8.5);
motor[motorB]=0;
nSyncedTurnRatio = -100;
nMotorEncoder[motorB] = 0;
nMotorEncoderTarget[motorB] = 135;
motor[motorB] = 35;
while(nMotorRunState[motorB] != runStateIdle);
SensorValue[sonar] = PN;
nSyncedTurnRatio = -100;
nMotorEncoder[motorB] = 0;
nMotorEncoderTarget[motorB] = -90;
motor[motorB] = -35;
while(nMotorRunState[motorB] != runStateIdle);
SensorValue[sonar] = PO;
nSyncedTurnRatio = 100;
motor[motorB] = 35;
while(SensorValue[warna] == 2.5);
nSyncedTurnRatio = -100;
nMotorEncoder[motorB] = 0;
nMotorEncoderTarget[motorB] = 90;
motor[motorB] = 35;
while(nMotorRunState[motorB] != runStateIdle);
SensorValue[sonar] = ON;}
```

Gambar 5. Kode program membaca jarak

3.1.1 Insialisasi Persamaan Lintasan dan Jarak

Dalam pembacaan jalur maka mekanisme yang dilakukan robot di sini adalah dua arah. Artinya:

1. MN = NM
2. NO = ON
3. OP = P
4. PM = MP
5. MO = OM
6. NP = PN

Gambar menunjukkan kode program untuk menginisialisasi variabel jalur.

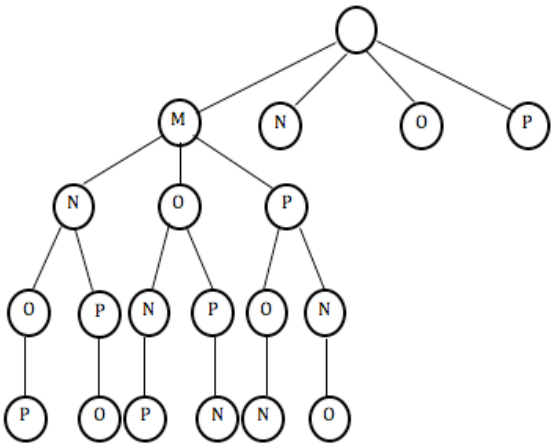
```
task inisialisasi(){
int NM, OM, PM, NP, OP, NO;
MN = NM;
OM = MO;
PM = MP;
NP = PN;
OP = PO;
NO = ON;
}
```

Gambar 6. Kode program inisialisasi jalur

 ISSN: 1907-5022

3.2 Pembangkitan Solusi *Generate and Test*

Penyelesaian dengan metode *Generate and Test* dilakukan dengan membangkitkan solusi-solusi melalui penyusunan titik-titik kota dalam urutan abjad. Gambar 7 merepresentasikan penyusunan titik-titik tersebut.



Gambar 7. Representasi *Generate and Test*

Dalam kasus ini dimulai dari node M. Dipilih sebagai keadaan awal adalah jalur MNOP dengan panjang lintasan 22 inci. Langkah selanjutnya adalah melakukan *backtracking* untuk mendapatkan lintasan MNPO dengan panjang 21 inci. Kedua lintasan dibandingkan, didapatkan MNPO < MNOP, sehingga yang dipilih adalah MNOP. Dilakukan lagi *backtracking* untuk mendapatkan lintasan MONP dengan panjang 15 inci. Didapatkan MONP < MNPO, maka dipilih MONP. Dilakukan lagi *backtracking* untuk mendapatkan lintasan MOPN dengan panjang 16 inci. Dibandingkan keduanya, didapatkan MONP < MOPN, maka dipilih MONP sebagai lintasan terpilih. Begitu seterusnya hingga ditemukan solusi lintasan terpendek.

Tabel 2 menunjukkan lintasan lintasan yang terbentuk dan yang terpilih.

Tabel 2. Alur pencarian metode *Generate and Test*

No	Lintasan	Panjang	Lintasan Terpilih	Panjang Terpilih
1	MNOP	22	MNOP	22
2	MNPO	21	MNPO	21
3	MONP	15	MONP	15
4	MOPN	16	MONP	15
5	NMOP	20	MONP	15
6	NMPO	24	MONP	15
7	NOPM	21	MONP	15
8	NOMP	18	MONP	15
9	NPMO	17	MONP	15
10	NPOM	16	MONP	15
11	OMNP	18	MONP	15
12	OMPN	17	MONP	15
13	ONMP	19	MONP	15
14	ONPM	19	MONP	15
15	OPMN	22	MONP	15
16	OPNM	24	MONP	15
17	PMNO	23	MONP	15
18	PMON	18	MONP	15
19	PNMO	23	MONP	15
20	PNOM	15	PNOM	15
21	POMN	20	PNOM	15
24	PONM	22	PNOM	15

Dari table di atas dapat dilihat lintasan yang terpendek adalah MNOP atau PNOM. Kedua-duanya dapat dijadikan pilihan sebagai solusi.

3.2.1 Inisialisasi Lintasan Terpilih Oleh Robot

Setelah sebelumnya menginisialisasi persamaan jalur seperti yang telah di jelaskan di sub-bab 3.1.1, maka langkah selanjutnya bagi robot adalah untuk menginisialisi jalur. Logika penginisialisasiannya adalah sebagai berikut:

1. Lintasan MNOP = MN+NO+OP.
2. Lintasan MNPO = MN+NP+PO.
3. Lintasan MPNO = MP+PN+NO.
4. Lintasan MPON = MP+PO+ON.
5. Lintasan MOPN = MO+OP+PN.
6. Lintasan MONP = MO+ON+NP.
7. Lintasan MOPN = MO+OP+PN.

Langkah di atas akan terus dilakukan sampai semua kemungkinan lintasan terbentuk seperti yang disampaikan di tabel 2. Dipastikan ada 24 lintasan yang akan terbentuk, begitu juga dengan insialisasinya. Gambar 8 menunjukkan kode program untuk menginisialisasikan lintasan yang terbentuk.

```
task inisialisasi2(){
int MNOP = MN+NO+OP;
int MNPO = MN+NP+PO;
int MOPN = MO+OP+PN;
int MONP = MO+ON+NP;
int MPON = MP+PO+ON;
int MPNO = MP+PN+NO;
int NMOP = NM+MO+OP;
int NMPO = NM+MP+PO;
int NOPM = NO+OP+PM;
int NOMP = NO+OM+MP;
int NPMO = NP+PM+MO;
int NPOM = NP+PO+OM;
int OMNP = OM+MN+NP;
int OMPN = OM+MP+PN;
int ONMP = ON+NM+MP;
int ONPM = ON+NP+PM;
int OPMN = OP+PM+MN;
int OPNM = OP+PN+NM;
int PMNO = PM+MN+NO;
int PMON = PM+MO+ON;
int PNMO = PN+NM+MO;
int PNOM = PN+NO+OM;
int POMN = PO+OM+MN;
int PONM = PO+ON+NM;
}
```

Gambar 8. Kode program inisialisasi lintasan

Langkah selanjutnya adalah mencari nilai terkecil dari variabel-variabel lintasan yang terbentuk seperti kode program di atas. Dari nilai terkecil tersebut dapat diartikan sebagai lintasan terpendek. Secara teknis, pencarian ini dapat dilakukan dengan dua cara, yang pertama membuat skema percabangan IF-THEN sejumlah banyak lintasan kemudian mencari nilai terkecilnya. Langkah kedua adalah membuat larik atau *array* dengan 24 indeks yang memetakan nilai lintasan dengan urutan pemasukan nilai lintasan. Dalam kasus ini digunakan metode yang kedua dimana nilai setiap lintasan dipetakan dalam sebuah larik agar dapat diterapkan mekanisme perulangan sehingga robot lebih cepat membaca. Indeks larik nantinya akan diasosiasikan dengan variabel lintasan yang telah dibuat. Berikut ini tabel 3 yang menunjukkan visualisasi lojik dari larik yang dibuat.

Tabel 3. Larik berisi lintasan yang terbentuk

indeks	elemen
1	22
2	21
3	15
4	16
5	20
6	24
7	21
8	18
9	17
…	…
24	22

Dari larik lintasan yang terbentuk, langkah selanjunya adalah mencari nilai terkecil dari 24 elemen yang ada. Gambar adalah kode program untuk mencari nilai elemen terkecil dari larik lintasan tersebut.

```
int Lintasan[24];
int i, min, noLintasan;
min = Lintasan[1];
for (i=1;i<=24;i++){
      if (Lintasan[i] < min){
            min = Lintasan[i];
            noLintasan = i;
      }
}
```

Gambar 9. Kode program nilai terkecil larik

Dari kode progam di atas, robot akan mendapatkan nilai terkecil dari larik lintasan sekaligus di indeks berapa nilai itu berada, yaitu indeks ke 3 dan 20 dengan nilai 15. Dengan kata lain robot telah mendapatkan lintasan mana yang terpendek. Langkah terakhir adalah membuat skema percabangan IF-THEN untuk mengecek nomor indeks lintasan terkecil itu berasosiasi dengan variable lintasan yang mana. Tabel 4 berikut ini, adalah asosiasi antara nomor indeks larik lintasan dengan lintasan.

Tabel 4. Asosiasi nomor indeks dengan variable lintasan

indeks	variabel
1	MNOP
2	MNPO
3	MONP
4	MOPN
5	NMOP
6	NMPO
7	NOPM
8	NOMP
9	NPMO
…	…
24	PONM

Dengan skema percangan IF-THEN maka didapatkan pengecekan kondisi seperti berikut:

IF <IndeksLintasanTerpendek>
THEN <lintasanTerpendek>

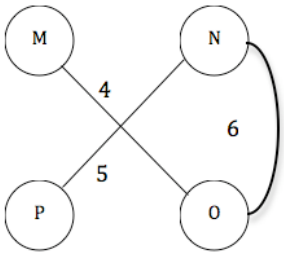
Kemudian selanjutnya skema percabangan ini direalisasikan berdasarkan hasil dari kode program seperti pada gambar 9, maka robot akan mendapatkan indeks ke 3. Dengan skema percabangan di atas penulis dapatkan:

IF 3 THEN MONP

Artinya robot akan mengenali lintasan MONP sebagai solusi (lintasan terpendek) dan akan bergerak melalui jalur sesuai hasil inisialisasi seperti yang dijelaskan pada sub-bab 3.2.1, yaitu MONP = MO+ON+NP. Selanjutnya robot akan bergerak dengan langkah sebagai berikut:

1. Robot di titik M, bergerak ke depan sepanjang sensor warna membaca warna hitam (jalur MO).
2. Robot di titik O, bergerak ke depan sepanjang sensor warna membaca warna hijau (jalur ON).
3. Robot di titik N, bergerak ke depan sepanjang sensor warna membaca warna abu-abu muda (jalur NP).
4. Robot berhenti.

 ISSN: 1907-5022



Gambar 10. Lintasan yang dilewati robot

4. KESIMPULAN

Dengan menggunakan metode *Generate and Test*, mampu diimplementasikan robot yang memilki kemampuan *self-recognizing* atau dapat mengenali lingkungan dengan sendirinya. Dari kemampuan ini selanjutnya robot dapat diprogram sedemikian rupa hingga dapat menyelesaikan permasalahan TSP. Dari segi efektifitas, penggunaan metode *Generate and Test* dalam kasus ini dapat dikatakan cukup baik dimana secara teknis walaupun membutuhkan banyak iterasi bagi robot dalam membaca jarak dan lintasan namun masih dalam batasan sehingga tidak menyebabkan waktu yang lama untuk robot bekerja.

Dari penelitian singkat ini didapatkan sebuah konsep yang diharapkan penulis sebagai embrio konsep yang lebih besar. Menurut penulis, memanfaatkan sebuah robot dalam melakukan pekerjaan yang membutuhkan kecerdasan atau pemikiran mandiri, kedepannya akan sangat bermanfaat. Penelitian ini mencoba melihat peluang bahwa dalam skala TSP yang lebih besar seperti dalam dunia nyata (seperti distribusi barang oleh perusahaan), akan sangat mungkin diimplementasikan pemecahannya berbasiskan robot, sehingga dapat dikatakan lebih efisien dan praktis.

PUSTAKA

Kusumadewi, S. (2003). *Artificial Intelligence*. Yogyakarta : Graha Ilmu.

Budiharto, W. (2009). *Membuat Sendiri Robot Cerdas*. Jakarta : PT Elex Media Komputindo.

Nugroho, A. (2009). *Apa Sih Robot Itu?*. Diakses pada 18 Mei 2010 dari http://nugroho.staff.uii.ac.id/category/mekatronika-dan-robotika

McCarthy, J. (2007). *What is Artificial Intelligence?*. Diakses pada 22 Mei 2010 dari http://www-formal.stanford.edu/jmc/whatisai/whatisai.html