Penelusuran Hasil Asesmen *Talents Mapping* dengan Menggunakan Algoritma *Uniform Cost Search*

Muhammad Alfian Rasyidin, 13516104
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung – Bandung, Indonesia
malfianrasyidin@gmail.com, 13516104@std.stei.itb.ac.id

Abstrak* – Setiap orang memiliki bakat yang berbeda-beda, seseorang dapat sangat baik dalam menggeluti suatu bidang, namun mungkin tidak pada bidang tertentu. Bahkan, walaupun terdapat dua orang berbeda memiliki bakat yang sama, sangat memungkinkan urutan kekuatan bakatnya berbeda. Perbedaan inilah yang menjadikan seseorang unik dengan orang lainnya. Terdapat banyak metode untuk mengetahui kekuatan bakat seseorang yang telah beredar luas, baik secara gratis atau pun berbayar. Salah satu diantaranya ialah Talents Mapping dari situs temubakat.com. Seseorang memiliki 34 bakat yang diurutkan berdasarkan kekuatan dari bakat tersebut pada dirinya. Adakalanya, organisasi, perusahaan, dan perkumpulan menginginkan untuk mencari seorang dengan kriteria tertentu untuk bekerja sama, memegang amanah, atau pun untuk hal lainnya. Hal ini tentunya membutuhkan metode pencarian data yang efektif dan efisien, agar sesuai dengan yang diharapkan. Terdapat banyak algoritma pencarian untuk melakukannya,2 makalah ini akan membahas metode pencarian data tersebut dengan menggunakan algoritma uniform cost search (UCS).

Kata Kunci* — bakat; talents mapping; algoritma pencarian; uniform cost search;

I. PENDAHULUAN

Kamus Besar Bahasa Indonesia (KBBI) mendefinisikan bakat sebagai kepandaian, sifat, dan pembawaan yang dibawa sejak lahir.[1] Banyak teori dan metode telah diujicobakan dan selalu dikembangkan hingga saat ini untuk mengelompokkan sekelompok orang ke dalam bakat tertentu. Kelompok bakat-bakat tersebut lalu diurutkan sesuai dengan individu masing-masing. Umumnya, banyak orang yang ingin mengetahui bakat yang dimilikinya dengan tujuan agar lebih terarah dan optimal dalam menggunakan bakatnya dalam kehidupan karirnya.

Talents mapping merupakan salah satu metode untuk mengetahui urutan bakat yang kita miliki. Metode ini menggunakan kuesioner sebagai basis untuk melakukan analisis terhadap individu tersebut. Urutan bakat ini menggambarkan peluang bakat-bakat tersebut menonjol dalam diri individu tersebut. Walaupun hingga saat ini, belum ada teori yang sangat mendukung hasil dari analisis bakat dengan metode apa pun dengan keterkaitannya dengan kesuksesan. Namun, banyak orang yang merasa dirinya dapat lebih terarah jika telah mengetahui bakat yang mungkin dimilikinya tersebut.[2]

Selain bermanfaat untuk masing-masing individu, tes bakat juga biasanya dimanfaatkan oleh organisasi, perusahaan, atau perkumpulan lainnya. Umumnya, terdapat bagian khusus seperti bagian personalia atau manajemen sumber daya manusia yang bertugas dalam bidang ini. Tujuannya tidak lain agar organisasi tersebut dapat lebih efektif dalam menempatkan individu pada bidang-bidang yang memang mereka kompeten dan memiliki bakat di bidang tersebut. Tentu, akan terdapat situasi saat mereka membutuhkan pencarian seseorang yang memenuhi kriteria bakat-bakat tertentu.

Terdapat beragam cara dalam melakukan pencarian data dengan syarat memenuhi kriteria-kriteria tersebut. Algoritma yang dapat digunakan seperti *brute force*, *greedy*, *breadth-first-search*, dan lainnya. Tentu banyak aspek dalam memilih algoritma yang cocok untuk mendapatkan data yang diinginkan. Aspek tersebut dapat berupa banyak hasil yang diinginkan, seberapa optimal pencarian tersebut baik dalam segi hasil, waktu, ruang memori, dan aspek lainnya. Makalah ini akan membahas aspek-aspek tersebut, jika pencarian dilakukan dengan menggunakan algoritma *uniform cost search (UCS).*

II. DASAR TEORI

A. *Talent Mapping*

Temubakat.com merupakan sebuah situs penyedia asesmen bakat secara luar jaringan *(online).* Asesmen *Talents Mapping* merupakan salah satu asesmen yang ada di situs tersebut. Asesmen ini bertujuan untuk menggali sifat produktif yaitu bakat dari dalam diri seseorang. Selain itu juga, asesmen ini dapat menginterpretasikan kekuatan terkait peran melalui bakat dominan yang dimiliki oleh individu tersebut. Asesmen ini merupakan *self-assesment* yang didalamnya memiliki 170 pernyataan.[3]

Salah satu hasil asesmen tersebut adalah 34 bakat yang terurut berdasarkan kekuatan bakat tersebut dalam individu itu.

Bakat dengan nomor urut 1 hingga 14 merupakan bakat yang paling menonjol dalam diri individu tersebut. Sedangkan, 14 bakat yang berada pada urutan terakhir merupakan bakat yang tidak menonjol pada dirinya. Berikut salah satu hasil dari asesmen tersebut:

1	DEVELOPER	18	COMMAND
2	BELIEF	19	STRATEGIC
3	RELATOR	20	SIGNIFICANCE
4	ACTIVATOR	21	FUTURISTIC
5	RESPONSIBILITY	22	INTELLECTION
6	INCLUDER	23	ANALYTICAL
7	CONNECTEDNESS	24	CONSISTENCY
8	INPUT .	25	FOCUS
9	LEARNER .	26	CONTEXT
10	INDIVIDUALIZATION	27	DELIBERATIVE
11	ADAPTABILITY .	28	ACHIEVER
12	HARMONY .	29	MAXIMIZER
13	SELF-ASSURANCE .	30	IDEATION
14	WOO	31	ARRANGER
15	EMPATHY	32	COMMUNICATION
16	RESTORATIVE	33	COMPETITION
17	POSITIVITY	34	DISCIPLINE

Gambar 1. Contoh Hasil Asesmen *Talents Mapping*
sumber: Talents Mapping BMKA Salman ITB

B. Basis Data Relasional

Basis data relasional terdiri dari tabel-tabel, yang masing-masing tabel tersebut menggunakan nama yang unik.[6] Terdapat dua istilah penting dalam basis data relasional ini yaitu *tuple* yang merupakan baris atau *row* dari tabel dan *attribute* yang merupakan kolom atau *column* dari tabel. Selain itu, terdapat beberapa jenis *key* dalam skema basis data ini yaitu:

1. Super key

   Super key merupakan semua himpunan *key* yang dapat membuat *tuple* dalam tabel tersebut unik.

2. Candidate Key

   Candidate key ialah himpunan *super key* yang memiliki jumlah atribut yang paling minimum.

3. Primary Key

   Primary key merupakan salah satu anggota dari himpunan *candidate key* yang dipilih.

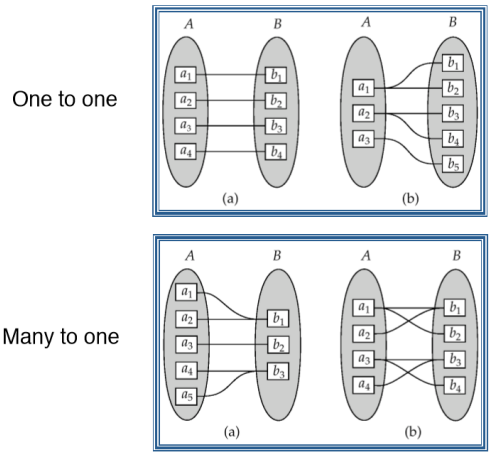
4. Alternate Key

   Alternate key merupakan seluruh anggota dari *candidate key* yang tidak dipilih menjadi *primary key.*

5. Foreign Key

   Foreign key merupakan *primary key* dari tabel lain yang disimpan sebagai *reference table.*

Agar dapat melakukan modifikasi, setiap tabel harus memiliki sebuah *primary key* yang dapat menjadikan *tuple* tersebut menjadi unik atau tidak akan ada *tuple* yang akan memiliki semua data yang sama persis.



Gambar 2. *Mapping Cardinalities*
sumber: Database System Concepts 6th Edition

Basis data relasional juga mengenal *mapping cardinalities* yang menggambarkan keterhubungan antar tabel pada basis data. *Mapping cardinalities* dapat dijelaskan sebagai berikut:

1. *One-to-one relationship*

   One-to-one relationship merupakan suatu kondisi saat masing-masing tabel memiliki tepat satu *tuple* yang saling berhubungan dengan tabel referensinya. Sebagai contoh: misalkan terdapat tabel mahasiswa dan tabel tugas akhir nya. Setiap *tuple* pada mahasiswa akan memiliki tepat satu *tuple* pada tugas akhir.

   Pada saat implementasinya pada *Database Management System (DBMS),* salah satu *primary key* diantara keduanya dapat diletakkan sebagai *foreign key* pada tabel lainnya. Pemilihan tabel mana yang harus menyimpan informasi tersebut sangat bergantung dengan banyaknya potensi *null key* yang dihasilkan.

   Sebagai contoh: pada tabel mahasiswa dan tabel tugas akhir tersebut, sebaiknya *primary key* dari tabel mahasiswa disimpan sebagai *foreign key* pada tabel tugas akhir. Hal ini mengingat bahwa jika *primary key* disimpan sebaliknya akan lebih banyak menghasilkan *null key* untuk mahasiswa yang belum memiliki tugas akhirnya.

2. *One-to-many relationship*

   One-to-many relationship merupakan kondisi pada saat salah satu tabel untuk setiap *tuple*-nya mempunyai hubungan dengan beberapa *tuple* pada tabel lainnya. Sebagai contoh: missal terdapat tabel mahasiswa dan tabel dosen wali. Setiap satu *tuple* pada tabel dosen wali memiliki hubungan dengan beberapa *tuple* pada mahasiswa. Hal ini mengingat setiap dosen wali dapat

memiliki lebih dari satu mahasiswa, namun setiap mahasiswa tepat hanya memiliki satu dosen wali saja.

Pada saat implementasi pada *DBMS*, *primary key* dari tabel yang mempunyai hubungan relasi ke beberapa tabel lainnya harus disimpan di tabel yang setiap *tuple*-nya hanya memiliki hubungan dengan satu tabel pada tabel tersebut. Sebagai contoh: pada hubungan tabel mahasiswa dan tabel dosen wali, *primary key* dari tabel dosen wali akan disimpan pada tabel mahasiswa sebagai *foreign key*. Hal ini mengingat setiap mahasiswa hanya memiliki tepat satu *tuple* pada tabel dosen wali.

3. *Many-to-one relationship*

Many-to-one relationship sama hal nya dengan relasi *one-to-many relationship* baik dalam segi defisini atau pun implementasi pada *database management system.*

4. Many-to-many relationship

Many-to-many relationship merupakan suatu kondisi dimana masing-masing *tuple* pada setiap tabel dapat memiliki hubungan relasi dengan beberapa *tuple* pada tabel lainnya. Sebagai contoh: misal terdapat tabel peserta dan tabel bakat, setiap *tuple* pada tabel peserta akan memiliki hubungan dengan beberapa *tuple* pada tabel bakat. Begitu pun sebaliknya, setiap *tuple* pada tabel bakat akan memiliki hubungan pada beberapa *tuple* pada tabel peserta.

Pada saat melakukan implementasi pada *database management system (dbms)*, hubungan relasi ini akan ditempatkan pada suatu tabel baru yang sering dinamakan dengan tabel pivot. Pada tabel pivot ini akan disimpan *primary key* dari masing-masing *tuple* pada kedua tabel yang saling berhubungan. Di dalam tabel ini juga dapat ditambahkan atribut yang terikat dengan hubungannya dengan relasi dari kedua tabel.

Sebagai contoh: relasi *many-to-many* pada tabel peserta dan tabel bakat akan diletakkan pada sebuah tabel pivot yang akan berisikan *primary key* dari masing-masing tabel yaitu tabel peserta dan tabel bakat, ditambah dengan atribut prioritas yang juga merupakan informasi yang melekat pada setiap relasi tersebut.

C. Algoritma Uniform Cost Search

Algoritma *Uniform Cost Search* yang selanjutnya disingkat dengan algoritma *UCS* merupakan salah satu algoritma pencarian yang cukup efektif dalam hal mendapatkan hasil yang terbaik. Algoritma ini menggabungkan prinsip pencarian dengan menggunakan algoritma *Breadth First Search (BFS)* dan *Iterative Deepening Search (IDS)* dengan mempertimbangkan langkah-langkah yang menghasilkan nilai akhir yang optimal.[4] Selain itu, algoritma ini tidak membutuhkan data *heuristic* untuk melakukan perhitungan dalam memperkirakan *cost* ke simpul tujuan.

Algoritma ini mirip dengan algoritma *Depth First Search (DFS)* yang membangkitkan *node* berdasarkan kriteria tertentu. Pada DFS kriteria tersebut ialah seberapa dalam *node* tersebut dari akar *(root)*, sedangkan pada *UCS* kriteria tersebut menjadi total *cost* yang telah dilalui hingga mencapai *node* tersebut. Sehingga, ide utama dari algoritma ialah hanya membangkitkan *node-node* yang memiliki total *cost* untuk mencapai *node* tersebut yang paling minimum. Tidak ada *node* selain yang mempunyai nilai minimum yang dapat dibangkitkan. Hal ini menyebabkan *node* yang dibangkitkan dapat secara acak, dalam arti lain pembangkitan tidak harus urut sesuai dengan *level* dari *tree.*

Sedangkan dalam implementasinya, algoritma ini perlu menyimpan suatu tabel yang berisi urutan-urutan *node* yang menjadi kandidat untuk dilakukan ekspansi berikutnya. Umumnya, table ini diimplementasikan sebagai *priority queue* dengan akumulasi total *cost* yang paling minimum berada di paling depan. Algoritma ini berhenti melakukan pencarian pada saat telah menemukan satu hasil yang sesuai dengan kriteria pencarian.[5]

Berikut algoritma *Uniform Cost Search* dalam bentuk pseudocode:

```
add all root into priority queue
while priority queue is not empty:
    take a very top element in the priority queue
    if top element reaches the goal state:
        exit and result has found
    else:
        add all child of current element to the
        priority queue
```

sumber: [5] dengan modifikasi oleh penulis.

III. PEMBAHASAN

A. Lingkungan Pengujian dan Sampel Data

Penyelesaian masalah ini dilakukan dengan mempersiapkan data uji dan membuat sebuah program uji. Program uji berbasis *web* agar lebih mudah untuk dilakukan visualisasi terhadap hasil uji. Program dibuat dengan sebuah *framework* PHP yaitu Laravel versi 5.6. Sedangkan, terdapat tiga tabel data yang digunakan yaitu tabel peserta, tabel bakat, dan tabel pivot untuk kedua tabel sebelumnya.

Tabel pivot digunakan karena hubungan relasi antara tabel peserta dengan tabel bakat ialah *many-to-many* yang memungkinkan banyak peserta dapat memiliki banyak bakat, dan tidak lupa dengan prioritas bakat yang dimilikinya. Dalam kasus uji kali ini, setiap orang memiliki 5 bakat yang dipilih secara acak dari tabel bakat. Setiap orang tidak mungkin memiliki lebih dari satu bakat yang sama dari kumpulan 5 bakat tersebut.

Berikut merupakan tabel data uji yang telah dimodifikasi untuk memudahkan pembacaan:

Tabel 1. Tabel Peserta

ID	Nama Lengkap
K	Saige Berge
L	Nella Shields
M	Amalia Walsh
N	Aurelia Johnson
O	Mr. Garret Reynolds I
P	Morgan Blanda

Tabel 2. Tabel Bakat

ID	Nama Bakat
A	Activator
B	Belief
C	Command

Tabel 3. Tabel Pivot

ID_Peserta	Prioritas		
	A	B	C
K	2	1	4
L	3	5	4
M	5	2	1
N	3	5	4
O	2	3	5
P	3	2	1

B. Algoritma Penyelesaian

Berikut merupakan algoritma dari program pengujian yang digunakan:

1. Penyelesaian dimulai pada akar yaitu simpul X.
2. Membangkitkan simpul anak dari X untuk dimasukkan ke dalam *priority queue.*
3. Pengurutan posisi di *priority queue* ialah indeks pertama diisi oleh akumulasi bobot paling minimum, selanjutnya tersusun menaik.
4. Jika ada dua atau lebih elemen *queue* yang memiliki bobot yang sama, maka akan diurutkan berdasarkan ID Peserta.
5. *Cost* dihitung dengan menjumlahkan nilai prioritas yang terdapat pada tabel pivot.
6. *Goal state* yaitu jumlah indeks bakat yang dicari dengan jumlah indeks iterasi simpul adalah sama.
7. Cek jika *top element* dari *priority queue* telah memenuhi syarat *goal state.*
8. Jika belum, maka lakukan ekspansi terhadap simpul *top element priority queue* dan letakkan kembali simpul anak-anaknya beserta akumulasi dari dari *total cost* yang telah dilewati.
9. Ulangi langkah 6 dan 7, hingga *top element queue* memenuhi persyaratan *goal state.*
10. Jika sudah dipenuhi, maka keluar dari pengulangan dan cetak hasil.

C. Langkah Penyelesaian

Berikut merupakan tabel langkah-langkah yang mendeskripsikan algoritma penyelesaian masalah yang digunakan:

Tabel 4. Langkah Penyelesaian

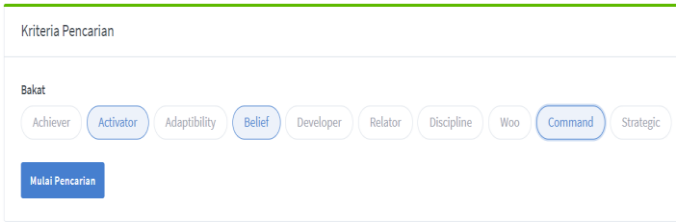
i	Simpul-E	Simpul Hidup
1	X	K_{A-2}, O_{A-2}, L_{A-3}, N_{A-3}, P_{A-3}, M_{A-5}
2	K_{A-2}	O_{A-2}, K_{AB-3}, L_{A-3}, N_{A-3}, P_{A-3}, M_{A-5}
3	O_{A-2}	K_{AB-3}, L_{A-3}, N_{A-3}, P_{A-3}, M_{A-5}, O_{AB-5}
4	K_{AB-3}	L_{A-3}, N_{A-3}, P_{A-3}, M_{A-5}, O_{AB-5}, K_{ABC-7}
5	L_{A-3}	N_{A-3}, P_{A-3}, M_{A-5}, O_{AB-5}, K_{ABC-7}, L_{AB-8}
6	N_{A-3}	P_{A-3}, M_{A-5}, O_{AB-5}, K_{ABC-7}, L_{AB-8}, N_{AB-8}
7	P_{A-3}	M_{A-5}, O_{AB-5}, P_{AB-5}, K_{ABC-7}, L_{AB-8}, N_{AB-8}
8	M_{A-5}	O_{AB-5}, P_{AB-5}, K_{ABC-7}, M_{AB-7}, L_{AB-8}, N_{AB-8}
9	O_{AB-5}	P_{AB-5}, K_{ABC-7}, M_{AB-7}, L_{AB-8}, N_{AB-8}, O_{ABC-10}
10	P_{AB-5}	P_{ABC-6}, K_{ABC-7}, M_{AB-7}, L_{AB-8}, N_{AB-8}, O_{ABC-10}
11	P_{ABC-6}	solusi ditemukan

Pada tabel diatas, simpul yang dilakukan ekpansi pertama kali ialah simpul akar yaitu X. Simpul ini melakukan ekpansi terhadap simpul-simpul anak yang dimilikinya. Simpul hidup pada tabel tersebut menggambarkan letak simpul yang disimbolkan oleh huruf kapital, dan *path* yang telah dilaluinya dituliskan pada *subscriptnya* beserta bobot *cost* dari *path* yang telah dilaluinya tersebut dengan dipisahkan tanda '-'.

Sebagai contoh, P_{AB-5} menggambarkan simpul saat ini berada pada simpul P dengan *path* P yang telah dilaluinya yaitu P_A dan P_B dengan bobot perjalanan tersebut yaitu sebesar 5. Kasus uji ini dapat diselesaikan dalam iterasi ke-11, yaitu saat jumlah *path* yang telah dilaluinya sama dengan jumlah kriteria bakat yang diinginkan.

D. Hasil Penyelesaian

Berikut merupakan beberapa *screenshoot* dari hasil program pengujian:



Gambar 3. Formulir Kriteria Pencarian
sumber: penulis

NAMA LENGKAP	NAMA BAKAT (PRIORITAS)
Saige Berge	Activator (2) Belief (1) Command (4)
Nella Shields	Activator (3) Belief (5) Command (4)
Amalia Walsh	Activator (5) Belief (2) Command (1)
Aurelia Johnson	Activator (3) Belief (5) Command (4)
Mr. Garret Reynolds I	Activator (2) Belief (3) Command (5)
Morgan Blanda	Activator (3) Belief (2) Command (1)
NAMA LENGKAP	NAMA BAKAT (PRIORITAS)

Gambar 4. Tabel Data Keseluruhan Pencarian
sumber: penulis

Gambar 5. Hasil Pencarian dengan Menggunakan Algoritma *Uniform Cost Search*
sumber: penulis

E. Analisis

Berdasarkan hasil pengujian yang telah dilakukan, algoritma *Uniform Cost Search* selalu menghasilkan hasil yang paling optimal dengan tingkat keberhasilan mencapai 100%. Di sisi lain, algoritma ini cukup optimum dalam aspek ruang memori dibandingkan dengan *breadth-first-search*. Hal ini disebabkan algoritma *UCS* tidak pernah membangkitkan *node* yang menurutnya tidak akan mengarah ke solusi, yaitu *node* yang lebih besar dari *node* yang memiliki *cost* yang lebih murah.

Namun, dalam hal ruang memori, algoritma *UCS* tidak lebih baik daripada *greedy*. Algoritma *UCS* membutuhkan ruang alokasi memori yang lebih daripada *greedy* karena algorima *UCS* membutuhkan ruang penyimpanan sementara yaitu *priority queue* yang tentunya dapat saja menggunakan memori yang banyak, terutama jika *node* yang harus diselesaikan sangat banyak. Begitu pun dengan waktu yang dihabiskan untuk menyelesaikan algoritma *UCS*, secara rata-rata, akan jauh lebih lama dari pada menggunakan algoritma *greedy*. Namun, sayangnya algoritma *greedy* tidak selalu mendapatkan hasil yang paling optimal.

Algoritma *UCS* jika dibandingkan dengan algoritma *brute force* akan jauh lebih unggul dalam hal ruang memori dan waktu penyelesaian, terutama jika *node* yang ada sangatlah banyak. Namun, dalam hal mendapatkan hasil yang paling optimal, kedua algoritma ini akan selalu menghasilkan hasil yang paling optimal. Namun, perbedaan yang paling menonjol ialah algoritma *UCS* hanya memungkinkan menampilkan satu solusi optimum yaitu solusi yang ditemukan paling pertama saja, sedangkan algoritma *brute force* dapat menampilkan keseluruhan kemungkinan solusi optimum yang ada.

Berikut merupakan tabel perbandingan antara ketiga algoritma tersebut:

Tabel 5. Perbandingan Algoritma *Brute Force*, *Greedy*, dan *Uniform Cost Search*

Aspek	***Brute Force***	***Greedy***	***UCS***
Ruang memori	Sangat boros, jika n sangat besar	Lebih hemat, karena hanya melakukan ekspansi terhadap satu jalur saja	Bergantung dengan akumulasi *cost*
Waktu penyelesaian	Sangat lama, jika n sangat besar	Relatif hampir sama, karena hanya melakukan ekpansi pada satu jalur	Bergantung dengan banyaknya *node* yang dilakukan ekspansi
Hasil yang didapatkan	Paling optimal, karena melakukan pemeriksaan terhadap seluruh kemungkinan	Relatif optimum, terkadang menghasilkan hasil yang merupakan bukan hasil yang optimal	Paling optimal, karena *path* yang dilalui mempunyai *cost* paling minimum
Banyak hasil yang didapatkan	Memungkinkan lebih dari satu kemungkinan solusi optimum	Hanya satu solusi yang berada pada jalur *greedy*	Hanya satu solusi pertama yang didapatkan

Dalam hal ini, jika terdapat banyak data pada tabel peserta, algoritma *brute force* sangatlah tidak efisien untuk digunakan dalam melakukan pencarian, karena akan menghabiskan waktu yang lama. Namun, jika memang harus mendapatkan semua kemungkinan solusi, maka secara tidak langsung, sistem harus melakukan evaluasi terhadap seluruh kemungkinan solusi. Namun, jika pencarian hanya bertujuan mencari seorang yang memiliki bakat yang terkuat, maka algoritma *uniform cost search* sangat dianjurkan untuk digunakan. Selain karena waktu penyelesaian yang tidak terlalu memakan waktu yang lama, algoritma ini juga menghasilkan sebuah solusi yang paling optimal pertama.

IV. SIMPULAN

Talents mapping merupakan salah satu metode untuk mengetahui kekuatan bakat dalam diri seseorang. Kumpulan data kekuatan bakat ini dapat dimanfaatkan bukan hanya individu yang bersangkutan, namun juga organisasi atau perkumpulan lainnya yang membutuhkan untuk mengalokasikan anggotanya agar sesuai dengan kemampuan dan bakat yang dimilikinya. Salah satu fitur yang sangat mungkin diperlukan ialah melakukan pencarian seseorang yang memenuhi kriteria bakat tertentu. Pencarian ini lebih tepat jika menggunakan algoritma *uniform cost search*, karena menghasilkan hasil yang paling optimal dan juga tidak membutuhkan waktu yang cukup lama, dibandingkan algoritma pencarian lainnya.

UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Allah yang Maha Esa karena atas berkat dan rahmatnya, penulis dapat menyelesaikan makalah ini dengan baik dan lancar. Penulis juga ingin mengucapkan terima kasih untuk Ibu Nur Ulfa Maulidevi, Bapak Rinaldi Munir, dan Ibu Masayu Leylia Khodra selaku dosen mata kuliah IF2211 - Strategi Algoritma yang telah memberikan bimbingan dan ilmu dalam penulisan makalah ini.

Penulis juga berterima kasih kepada semua pihak yang tidak dapat saya tulis seluruhnya di bagian ini, terutama untuk penulis-penulis yang tulisannya menjadi referensi di dalam penulisan makalah ini.

REFERENSI

[1] https://kbbi.kemdikbud.go.id/entri/bakat diakses pada tanggal 10 Mei 2018 pukul 19.27.

[2] https://www.psychologytoday.com/us/blog/the-procrastination-equation/201110/hard-work-beats-talent-only-if-talent-doesn-t-work-hard diakses pada tanggal 9 Mei 2018 pukul 19.52.

[3] http://temubakat.com/tma diakses pada tanggal 9 Mei 2018 pukul 19.20.

[4] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/A-Star-Best-FS-dan-UCS-(2018).pdf diakses pada tanggal 11 Mei 2018 pukul 20.42.

[5] https://algorithmicthoughts.wordpress.com/2012/12/15/artificial-intelligence-uniform-cost-searchucs/ diakses pada tanggal 12 Mei 2018 pukul 20.13.

[6] Silberschatz, Korth, and Sudarshan, 2011, Database System Concepts, 6th Ed

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, 14 Mei 2018

Muhammad Alfian Rasyidin