Penggunaan Heuristik Skor Scrabble untuk Optimisasi Pengecek Feasibility Konstruksi Puzzle Teka-Teki Silang

Dzubyan Ilman Ramadhan- 10122010
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: author@gmail.com , author@std.stei.itb.ac.id

***Abstract*— Dalam implementasi pencarian solusi dari suatu konfigurasi puzzle teka teki silang, dibutuhkan suatu heuristik yang memperkirakan seberapa sering suatu kombinasi huruf muncul pada kata-kata. Karena sifat dari skor Scrabble, skor tersebut cocok untuk dijadikan heuristik dalam masalah ini. Diperoleh bahwa pencarian dengan heuristik lebih cepat dan optimal daripada pencarian biasa**

Keywords—crosswords; heuristics; optimization; puzzle; search;

I. Introduction

Pengaplikasian algoritma untuk pencarian, logika dan pemenuhan konstrain pada puzzle telah menjadi *benchmark* dalam ilmu komputer sains. Di antaranya adalah puzzle yang berkaitan dengan kata-kata seperti teka-teki silang dan *Scrabble*.

A. Teka Teki Silang

Teka-teki silang telah menjadi puzzle kata yang populer. Untuk memecahkan suatu teka-teki silang, kita harus mengisi barisan kotak-kotak yang mendatar atau menurun dengan huruf-huruf, satu huruf untuk satu kotak, yang membentuk suatu kata yang sesuai dengan petunjuk yang diberikan, sehingga semua kotak terisi tanpa adanya konflik perbedaan huruf.

Secara umum, pada konstruksi puzzle teka-teki silang, konstruktor puzzle akan membuat grid yang akan digunakan (biasanya simetris) dan memasukkan beberapa kata yang akan menjadi tema dari puzzle tersebut. Tahap ini disebut “seeding”. Kata-kata ini biasanya adalah kata-kata yang mungkin panjang atau jarang digunakan di kehidupan sehari-hari. Setelah itu, konstruktor akan mengisi kata-kata lain yang menjadikan grid tersebut sebagai kunci jawaban dari puzzle itu sendiri. Kata-kata yang dimasukkan pada tahap ini adalah kata-kata yang biasanya pendek dan cukup muncul di kehidupan sehari-hari sebagai bantuan untuk memecahkan kata-kata “seeding” yang lebih panjang tadi. Tahap tersebut biasa disebut “filling” [1]. Tahap inilah yang menjadikan puzzle teka-teki silang sebuah puzzle.

Dalam tahap “filling” inilah, terdapat kesulitan yang sering muncul. Konstruktor harus memilah kata-kata dengan hati-hati dan memastikan grid teka-teki silang dapat diisi agar tidak ada konflik di pertemuan suatu kata mendatar dengan suatu kata menurun dan memastikan suatu slot kata mendatar atau menurun dapat terisi, sehingga tak terjadi *dead end*. Pertemuan 2 kata ini biasanya disebut “crossing”. Sebagai contoh, konstruktor puzzle harus menghindari menaruh kata “JARS” dan “RUE” secara bertetanggaan seperti pada Fig 1. Karena kata menurun yang crossing dengan huruf “J “ di “JARS” harus memiliki “JR” sebagai awalannya.

Umumnya, kita ingin agar terdapat banyak pilihan kata untuk suatu slot agar konflik seperti Fig 1. tidak terjadi. Maka dari itu, apabila kita ingin mengoptimalisasikan automasi proses ini, kita perlu suatu heuristik pencarian untuk mengukur seberapa sering suatu huruf dan kombinasinya sering muncul di seluruh kata. Dengan ini, kita bisa memprioritaskan kata-kata yang memiliki banyak huruf umum sehingga deadlock dalam pencarian solusi dapat dihindari.

B. Scrabble

Pada permainan Scrabble, pemain membentuk kata dan memperoleh skor berdasarkan total dari skor yang tertera dari setiap huruf. Skot tersebut diberikan berdasarkan frekuensi seberapa sering huruf tersebut tersebut muncul dalam kamus. Huruf seperti “T”, “S” yang sering muncul dalam kata-lata diberikan skor kecil, sedangkan huruf seperti “Q”,”Z” yang jarang muncul diberikan skor yang besar.

Maka dari itu, skor ini cocok bila dijadikan heuristik dalam mencari solusi dari suatu konfigurasi puzzle teka-teki silang. Kita buat sebuah hipotesis bahwa penggunaan heuristik skor ini dapat mempercepat proses “fill” daripada algoritma dengan brute forcing mencocokan untuk tiap kata.

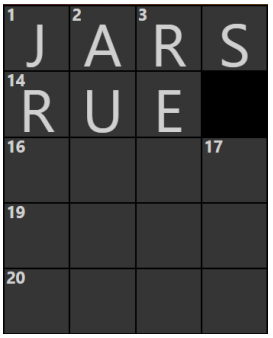


Fig 1. Contoh peletakan yang buruk

C. *Algoritma Backtracking*

Backtracking adalah sebuah metode untuk traversal suatu ruang solusi. Pada algoritma ini, hanya pilihan yang mengarah ke solusi yang dieksplorasi pilihan yang tidak mengarah ke solusi tidak dipertimbangkan akan lagi [2]. Algoritma ini akan kita gunakan untuk melakukan pencarian kata karena setiap pemilihan kata akan mempengaruhi pilihan kata pada slot selanjutnya. Apabila suatu konfigurasi menyebabkan tidak adanya solusi yang tersedia, kita backtrack ke simpul parent dari pohon tersebut dan mencoba kemungkinan solusi lainnya.

Pencarian dengan DFS juga memberikan pencarian yang lebih hemat memori, karena DFS hanya menyimpan suatu cabang saja. Pada masalah ini branching factor-nya akan menjadi sangat besar. Oleh karena itu, pencarian DFS lebih cocok ketimbang pencarian BFS yang mengharuskan kita untuk menyimpan dan mempertimbangkan seluruh cabang terlebih dahulu sebelum meluaskan suatu simpul lebih dalam..

II. Metodologi

Untuk menguji hipotesis tersebut, kita akan melakukan perbandingan sederhana dalam Python terhadap waktu yang dibutuhkan metode brute forcing dan metode dalam mencari solusi mana saja, bila diberikan suatu konfigurasi teka-teki silang yang diisi sebagian. Sebagai persiapan, kita akan menggunakan daftar kata dari sumber [3] yang akan menjadi daftar kata yang valid untuk dimasukkan ke suatu slot dari puzzle. Daftar kata tersebut memiliki sekitar 130000 kata dalam bahasa Inggris. Untuk menyiapkan daftar katanya, kita urutkan dari panjangnya, lalu secara alfabetis.

Sementara itu, untuk teknik heuristik, kita akan gunakan daftar kata yang sama yang telah ditambahkan skornya. Skor ini dihitung dengan cara berikut, untuk suatu kata, hitung total skor Scrabble dari setiap huruf unik dari kata tersebut (Detail skor dapat dilihat pada Tabel I). Lalu, bagi dengan banyak huruf di kata tersebut. Skor akan dikalikan 10 dan dibulatkan agar kita bekerja dengan bilangan bulat. Sebagai contoh, kata “POSTS” memiliki total skor 3 + 1 + 1 + 1 = 6. Maka skornya adalah 6/5 * 10 = 12. Selanjutnya, semua kata diurutkan berdasarkan panjang, lalu skor yang diperoleh, lalu diurutkan secara leksikografis.

Pengurutan dilakukan agar kita bisa melakukan binary search untuk memvalidasi kata di suatu slot yang terisi penuh daripada mengecek secara manual.

TABEL I. SKOR SCRABBLE TIAP HURUF

Huruf	Skor	Huruf	Skor
A	1	N	1
B	3	O	1
C	3	P	3
D	2	Q	10
E	1	R	1
F	4	S	1
G	2	T	1
H	4	U	1
I	1	V	4
J	8	W	4
K	5	X	8
L	1	Y	4
M	3	Z	10

A. *Metode Backtracking Biasa*

Algoritma backtracking yang digunakan adalah sebagai berikut:

1) Untuk tiap slot yang tersedia, bangkitkan semua kata yang memiliki panjang yang sama dengan slot, lalu iterasikan semua kata untuk mengecek apakah kata tersebut memenuhi konstrain slot tersebut pada saat itu.
2) Lanjutkan untuk slot selanjutnya
3) Apabila tidak ada kata yang bisa dimasukkan, back track ke slot sebelumnya dan ganti kata tersebut dengan kata selanjutnya menurut daftar kata biasa
4) Ulangi langkah 2 dan 3 hingga seluruh slot terisi oleh kata yang valid.

B. Metode Heuristik

Algoritma yang digunakan adalah sebagai berikut:

1) Untuk tiap slot yang tersedia, bangkitkan semua kata yang memiliki panjang yang sama dengan slot yang tersedia, lalu iterasikan semua kata untuk mengecek apakah kata tersebut memenuhi konstrain slot tersebut pada saat itu dari urutan skornya dari terkecil.
2) Lanjutkan untuk slot selanjutnya
3) Apabila tidak ada kata yang bisa dimasukkan, back track ke slot sebelumnya dan ganti kata tersebut dengan kata selanjutnya menurut urutan daftar kata yang telah disortir berdasarkan skor yang telah dihitung
4) Ulangi langkah 2 dan 3 hingga seluruh slot terisi oleh kata yang valid.

C. Konstrain Masalah

Konfigurasi teka-teki silang berukuran m x n akan direpresentasikan dengan n string masing-masing dengan panjang m. Karakter "." melambangkan kotak putih yang dapat diisi oleh huruf. Karakter "#" melambangkan kotak hitam yang tidak akan diisi oleh apapun. Karakter-karakter huruf mungkin akan diisi di beberapa tempat. Suatu konfigurasi dapat dikatakan solusi apabila semua karakter "." telah diganti dengan huruf dan tiap baris atau kolom dari huruf yang panjangnya lebih dari sama dengan 3 disebut valid apabila kata tersebut termasuk dalam daftar kata yang telah disebutkan. Tidak boleh ada 2 slot kata yang sama,

Sebagai contoh, konfigurasi berikut

```
H..##
....#
....#
.##.#
```

Mungkin dapat diselesaikan dengan jawaban berikut

```
HIT##
AREA#
TEEMS
S##P#
```

Namun, solusi berikut tidak valid

```
HAT##
ARCH#
TOPAZ
E##T#
```

Karena terdapat 2 kata yang identik.

III. Hasil

Berdasarkan metode yang telah disebutkan dan spesifikasi solusi yang telah dipaparkan, hasil dari tiap metode pencarian tertera pada Tabel II-V. Tiap tabel akan berisi konfigurasi yang diujikan, hasil dari tiap metode pencarian serta waktu yang dibutuhkan.

TABEL II Perbandingan Pencarian Kasus 1

Konfigurasi
R... .#.# ...Y L#.#
Solusi Backtracking Biasa
RABE A#A# IMAY L#S# Elapsed time: 0.007253 seconds
Solusi Heuristik Scrabble
RIRI E#A# EASY L#A# Elapsed time: 0.007015 seconds

TABEL III Perbandingan Pencarian Kasus 2

Konfigurasi
B.... .#.#. ..M.. .#.#. ..... .#.#.
Solusi Backtracking Biasa
BAAED A#D#A AHMAD B#I#B AMNIO A#S#D Elapsed time: 0.052803 seconds
Solusi Heuristik Scrabble
BABAR

```
A#A#E
AIMEE
B#B#S
ALONE
A#O#S
Elapsed time: 0.011614 seconds
```

TABEL IV Perbandingan Pencarian Kasus 3

Konfigurasi
S.....S .#P#.#. .#...#. L#####. ####... ##G..#. ####P..
Solusi Backtracking Biasa
SABINES A#P#A#A I#APB#B L#####E ####AER ##GAL#E ####PAD Elapsed time: 4.103653 seconds
Solusi Heuristik Scrabble
SEERESS E#P#E#E L#ALL#S L#####T ####ALE ##GAP#T ####PPS Elapsed time: 0.008314 seconds

TABEL V Perbandingan Pencarian Kasus 4

Konfigurasi
AFRAIDS .#...## .#...## .....
Solusi Backtracking Biasa
No solution Elapsed time: 5.160664 seconds
Solusi Heuristik Scrabble
No solution Elapsed time: 4.896823 seconds

IV. Analisis Hasil

Berdasarkan dari hasil pada Bab III, dapat diperhatikan bahwa metode pencarian dengan heuristik selalu lebih cepat dibandingkan metode pencarian biasa. Bahkan, untuk beberapa kasus bisa sampai 50x lebih cepat. Memang, apabila grid teka teki silang semakin besar dan semakin sedikit petunjuk yang diberikan, maka akan semakin lama waktu yang dibutuhkan untuk program menyelesaikan program tersebut.

Sesuai hipotesis, pencarian heuristik akan memberikan pilihan yang lebih banyak untuk suatu kata di crossing yang sama bila dibandingkan dengan pencarian biasa. Sehingga, kata yang memberikan pilihan yang banyak akan dihentikan untuk diuji lebih dahulu. Akibatnya, pencarian dengan heuristik akan mencapai solusi lebih dahulu daripada pencarian backtracking biasa. Walaupun memang terdapat tradeoff dimana kita harus mengecek lebih banyak kemungkinan daripada solusi backtracking biasa. Akan tetapi, pencarian yang kita lakukan adalah sejatinya DFS, sehingga banyak cabang baru untuk solusi kotak selanjutnya tidak akan mempengaruhi lama pencarian solusi secara langsung, hanya besar grid dan banyaknya slot yang akan mempengaruhi lama waktu dalam pencarian.

Masalah ini memiliki

V. Kesimpulan

Pada makalah ini, kita telah menganalisis dan ingin mengkonfirmasi hipotesis penggunaan heuristik akan mengoptimalisasikan pencarian solusi untuk proses “filling” pada konfigurasi puzzle teka-teki silang. Berdasarkan hasil yang kita peroleh, heuristik yang kita berikan berhasil mengoptimalisasikan pencarian solusi.

Namun, metode yang digunakan pada makalah ini mungkin memberikan solusi yang kurang beragam dan kurang acak bila dibandingkan dengan puzzle teka-teki silang pada umumnya, karena kita hanya ingin mencari solusi mana saja yang memenuhi. Dapat dilihat dari bab III bahwa terdapat bias pada huruf yang memiliki skor rendah seperti “A”, “S”, dan “E”.Lalu, penggunaan heuristik skor

Scrabble mungkin kurang mencerminkan frekuensi huruf pada kata-kata [4].

Untuk pengembangan kedepannya, kita dapat menganalisis penggunaan tiap huruf dalam suatu daftar kata untuk mencari heuristik yang lebih bagus dan akurat agar pencarian dapat dilaksanakan lebih optimal. Bisa juga dengan mengimplementasikan early pruning dengan memperhatikan crossing tiap kata.

References

[1] B. Tausig and F. Vigeland, "How to Make a Crossword Puzzle," *The New York Times*, Apr. 11, 2018. Available: https://www.nytimes.com/2018/04/11/crosswords/constructing-themes.html

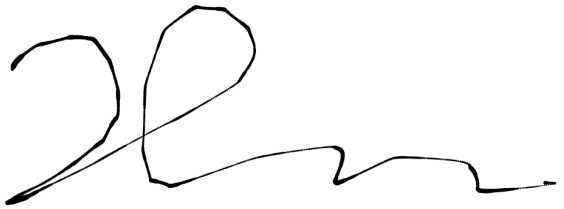
[2] R. Munir, "Algoritma Runut-balik (Backtracking)," in *Kuliah IF2211 Strategi Algoritma*, Jun. 24, 2025

[3] B. Husic and E. H. Anguiano, "our free wordlist," spread the word(list), Apr. 01, 2025. https://www.spreadthewordlist.com/wordlist (accessed Jun. 24, 2025).

[4] L. Gray, "Scrabble: Should letter values change?," *BBC*. https://www.bbc.com/news/magazine-20984707 (accessed Jun. 24, 2025).

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, 24 Juni 2025
Ttd
Dzubyan Ilman Ramadhan
10122010