Pemanfaatan Algoritma BFS dan DFS pada Graf Berbobot untuk Generasi Improvisasi Jazz

Miguel Rangga Deardo Sinaga 13524069
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524069@std.stei.itb.ac.id, miguelsinaga06@gmail.com

***Abstract*—Improvisasi jazz merupakan praktik musikal kompleks yang menuntut pemilihan nada secara cepat berdasarkan konteks akor, skala, dan gaya idiomatik bebop. Pemodelan komputasional untuk improvisasi semacam ini menjadi tantangan karena melibatkan banyak aturan implisit yang sulit diformalkan. Makalah ini mengusulkan pendekatan generasi improvisasi melodi jazz menggunakan algoritma penelusuran graf *Breadth-First Search* (BFS) dan *Depth-First Search* (DFS) pada graf berarah berbobot. Setiap akor dimodelkan sebagai graf yang simpulnya adalah nada-nada dari *bebop scale* beserta nada kromatik pengepung (*enclosure*), dan sisinya diberi bobot berdasarkan teori voice leading, chord-scale theory, serta perangkat melodik bebop seperti resolusi setengah langkah dan scalar run. Pemilihan nada berikutnya menggunakan strategi ε-greedy dengan *multiplier kontekstual* yang melihat dua nada terakhir untuk menyelesaikan enclosure dan resolve nada kromatik. Sistem diimplementasikan sebagai aplikasi web (Java Spring Boot + JavaScript) dengan pemutaran melodi melalui Web Audio API. Hasil pengujian menunjukkan bahwa DFS menghasilkan melodi yang lebih kohesif dan linear, sedangkan BFS menghasilkan melodi yang lebih eksploratif, dengan kedua algoritma tetap mempertahankan kepatuhan terhadap konteks akor.**

***Index Terms*—BFS, DFS, graf berbobot, improvisasi jazz, strategi algoritma, bebop scale, greedy.**

I. Pendahuluan

Musik jazz dikenal sebagai salah satu genre dengan elemen improvisasi yang paling menonjol. Berbeda dengan musik klasik yang umumnya dimainkan persis seperti partitur, jazz menempatkan kebebasan ekspresi pemain di pusat estetikanya: seorang pemain saksofon, misalnya, diharapkan menciptakan melodi baru setiap kali memainkan lagu yang sama. Walaupun terdengar spontan, improvisasi jazz sebenarnya tunduk pada seperangkat aturan implisit, tentang akor yang sedang aktif, skala yang dipakai, perangkat melodik khas seperti *chromatic approach*, hingga prinsip-prinsip *voice leading* [6]. Aturan-aturan inilah yang membuat improvisasi jazz tetap terdengar musikal dan tidak sekadar bunyi acak.

Karena terdapat aturan-aturan yang relatif terstruktur, improvisasi jazz menjadi domain menarik untuk pemodelan algoritmik. Pertanyaan yang muncul adalah: dapatkah aturan-aturan tersebut diterjemahkan ke dalam struktur data dan algoritma yang dipelajari dalam mata kuliah strategi algoritma, sehingga komputer dapat menghasilkan melodi improvisasi yang patuh pada konteks akor? Salah satu pendekatan yang menjanjikan adalah dengan *memodelkan nada sebagai simpul graf* dan transisi melodik sebagai sisi berbobot. Dengan model ini, masalah menghasilkan melodi berubah menjadi masalah penelusuran graf, topik yang sudah sangat dikenal dalam strategi algoritma.

Makalah ini membahas pemanfaatan dua algoritma penelusuran graf klasik, yaitu *Breadth-First Search* (BFS) dan *Depth-First Search* (DFS), untuk pembangkitan melodi improvisasi jazz pada graf berarah berbobot. Bobot sisi dirancang berdasarkan prinsip-prinsip teori musik jazz sehingga melodi yang dihasilkan oleh penelusuran mengikuti gaya bebop yang khas. Kontribusi yang diharapkan dari makalah ini antara lain:

1) Perumusan model graf nada jazz yang melibatkan tiga kelas simpul (chord tone, scale tone, approach note) beserta skema pembobotan sisinya berdasarkan teori musik;
2) Adaptasi algoritma BFS dan DFS untuk persoalan pembangkitan urutan (*sequence generation*), bukan pencarian jalur, dengan dukungan strategi ε-greedy bermuatan konteks melodik;
3) Implementasi sistem end-to-end berupa aplikasi web interaktif yang memungkinkan pengguna mengeksplorasi parameter algoritma dan mendengarkan hasilnya secara real-time.

Sistematika makalah ini disusun sebagai berikut. Bagian II memaparkan landasan teori. Bagian III menjelaskan analisis dan perancangan sistem. Bagian IV memaparkan implementasi algoritma kunci. Bagian V membahas hasil pengujian dan analisisnya. Bagian VI menyajikan kesimpulan, dan Bagian VII memberikan saran pengembangan lanjutan.

II. Landasan Teori

Pada bagian ini diuraikan landasan teoretis yang menjadi pondasi pengembangan sistem, meliputi teori graf, algoritma penelusuran (BFS dan DFS), strategi greedy, serta teori musik jazz yang relevan untuk pembentukan bobot pada graf nada.

A. Graf dan Graf Berbobot Berarah

Graf merupakan struktur data diskret yang merepresentasikan hubungan biner antar objek. Secara formal, graf didefinisikan sebagai pasangan terurut $G = (V, E)$ dengan V adalah himpunan tak-kosong simpul (*vertex*) dan $E \subseteq V \times V$ adalah himpunan sisi (*edge*) yang menyatakan relasi antar simpul [1].

Pada *directed graph* (graf berarah), tiap sisi memiliki orientasi sehingga $e = (u, v)$ menyatakan transisi dari u ke v dan tidak otomatis menyiratkan adanya $e' = (v, u)$. Generalisasi lebih lanjut diperoleh dengan melengkapi setiap sisi dengan suatu bobot melalui fungsi:

$$w : E \to \mathbb{R} \tag{1}$$

yang menghasilkan *directed weighted graph* $G = (V, E, w)$. Bobot $w(u, v)$ dapat merepresentasikan biaya, jarak, atau, dalam konteks pemodelan musik pada makalah ini, preferensi melodik dari transisi nada u ke nada v.

Bila $|V| = n$ dan $|E| = m$, graf dapat diimplementasikan menggunakan *adjacency list* dengan kompleksitas ruang $O(n + m)$, yang lebih hemat dibanding *adjacency matrix* bila graf bersifat *sparse* [2].

B. Breadth-First Search (BFS)

Breadth-First Search (BFS) merupakan algoritma penelusuran graf yang mengeksplorasi simpul-simpul secara berlapis (*level-by-level*) dari suatu simpul awal s. BFS menjamin bahwa pada graf tak-berbobot, jalur yang ditemui pertama kali menuju setiap simpul adalah jalur terpendek dalam jumlah sisi [2].

Algoritma BFS memanfaatkan struktur data *queue* (FIFO *First-In, First-Out*). Mula-mula simpul awal dimasukkan ke dalam queue dan ditandai sebagai terkunjungi. Selama queue tidak kosong, simpul terdepan diambil (*dequeue*), lalu seluruh tetangganya yang belum dikunjungi ditandai dan dimasukkan ke queue.

Algorithm 1 Breadth-First Search

```
1:  procedure BFS(G, s)
2:      Inisialisasi queue Q kosong
3:      Q.enqueue(s)
4:      Tandai s sebagai dikunjungi
5:      while Q ≠ ∅ do
6:          u ← Q.dequeue()
7:          for all tetangga v dari u do
8:              if v belum dikunjungi then
9:                  Tandai v sebagai dikunjungi
10:                 Q.enqueue(v)
11:             end if
12:         end for
13:     end while
14: end procedure
```

Kompleksitas waktu BFS adalah $O(|V| + |E|)$ untuk representasi adjacency list [2]. Karakter *lebar-dulu* BFS cocok untuk persoalan yang memerlukan eksplorasi merata dari beberapa cabang pada kedalaman yang sama.

C. Depth-First Search (DFS)

Depth-First Search (DFS) adalah algoritma penelusuran graf yang mengeksplorasi sedalam mungkin sepanjang satu cabang sebelum melakukan *backtracking* ke simpul sebelumnya untuk mencoba cabang lain [3]. DFS dapat diimplementasikan secara rekursif maupun iteratif menggunakan struktur data *stack* (LIFO ; *Last-In, First-Out*).

Algorithm 2 Depth-First Search (iteratif)

```
1:  procedure DFS(G, s)
2:      Inisialisasi stack S kosong
3:      S.push(s)
4:      while S ≠ ∅ do
5:          u ← S.pop()
6:          if u belum dikunjungi then
7:              Tandai u sebagai dikunjungi
8:              for all tetangga v dari u do
9:                  S.push(v)
10:             end for
11:         end if
12:     end while
13: end procedure
```

Kompleksitas waktu DFS juga $O(|V| + |E|)$ [3]. Sifat *dalam-dulu* pada DFS menghasilkan jalur yang lebih panjang dan kohesif pada masing-masing penelusuran, yang akan dimanfaatkan untuk menghasilkan melodi yang *mengalir*.

BFS dan DFS pada Persoalan Pembangkitan Urutan. Pada makalah ini, BFS dan DFS tidak digunakan untuk pencarian jalur unik melainkan sebagai strategi pembangkitan urutan (*sequence generation*). Konsekuensinya, syarat klasik "tiap simpul dikunjungi paling banyak satu kali" ditiadakan; revisit simpul justru diperlukan untuk membentuk pengulangan motif melodik. Yang dipertahankan dari kedua algoritma adalah struktur datanya (*queue* untuk BFS, *stack* untuk DFS) sehingga karakter *eksploratif–merata* pada BFS dan *linear–mengalir* pada DFS tetap tercermin dalam urutan nada yang dihasilkan.

D. Strategi Greedy dan Pemilihan Berbobot

Strategi *greedy* merupakan paradigma penyelesaian masalah yang membentuk solusi secara bertahap dengan mengambil keputusan optimal lokal pada setiap langkah, tanpa mempertimbangkan konsekuensi jangka panjang [2]. Pada konteks pembangkitan melodi, di setiap langkah algoritma memilih nada berikutnya v^* dari himpunan kandidat $N(u)$ (tetangga simpul aktif u) dengan memaksimalkan bobot:

$$v^* = \arg\max_{v \in N(u)} w(u, v) \cdot \mu(v \mid \mathcal{H}) \tag{2}$$

dengan $w(u, v)$ adalah bobot dasar sisi pada graf dan $\mu(v \mid \mathcal{H})$ adalah *multiplier kontekstual* yang dihitung dari riwayat melodik $\mathcal{H}$ (misalnya dua nada terakhir).

Untuk menambahkan variasi, dipakai pendekatan ε-*greedy* [4]: dengan probabilitas $1 - \varepsilon$ pemilihan dilakukan secara greedy, dan dengan probabilitas ε pemilihan dilakukan secara *weighted random* terhadap distribusi yang sebanding dengan bobot. Parameter ε pada makalah ini disebut sebagai *randomness* dan dapat diatur pengguna.

E. Dasar Teori Musik

1) Nada, Oktaf, dan Semitone: Nada (*pitch*) merupakan persepsi pendengaran terhadap frekuensi fundamental suatu sinyal periodik. Sistem nada barat membagi satu oktaf menjadi 12 nada dengan jarak yang sama besar (*equal-tempered tuning*) [5], sehingga rasio frekuensi antara dua nada yang berjarak satu *semitone* adalah:

$$\frac{f_{n+1}}{f_n} = 2^{1/12} \tag{3}$$

Dua belas nada tersebut, yaitu C, C#, D, D#, E, F, F#, G, G#, A, A#, B, secara kolektif disebut *tangga nada kromatik*. Dengan referensi $A_4 = 440$ Hz, frekuensi suatu nada pada oktaf o dengan indeks kromatik $n \in \{0, 1, \ldots, 11\}$ dapat dihitung dengan:

$$f(n, o) = 440 \cdot 2^{(12o+n-57)/12} \tag{4}$$

2) Interval Melodik: Interval adalah jarak antara dua nada yang diukur dalam satuan semitone [5]. Pada konteks gerakan melodi, interval sering dilipat ke rentang $[0, 6]$ semitone karena nada yang berjarak k dan $12 - k$ semitone terdengar mirip ketika dimainkan secara berurutan:

$$\mathrm{iv}(u, v) = \min\big(|n_v - n_u|,\, 12 - |n_v - n_u|\big) \tag{5}$$

Interval kecil (1–2 semitone, disebut *stepwise motion*) dianggap paling natural dalam melodi dan menjadi tulang punggung gerakan suara (*voice leading*) [6]. Interval yang lebih besar disebut *leap*.

3) Skala dan Chord-Scale Theory: Pada jazz terdapat hubungan sistematis antara tipe akor dan skala yang sesuai untuk improvisasi di atasnya, dikenal sebagai *chord-scale theory* [6]. Tabel I merangkum pasangan akor–skala yang dipakai pada sistem ini.

TABLE I
HUBUNGAN AKOR DAN SKALA (*Chord-Scale Theory*)

Tipe Akor	Skala (Mode)	Interval (semitone)
maj7	Ionian	0, 2, 4, 5, 7, 9, 11
7	Mixolydian	0, 2, 4, 5, 7, 9, 10
m7	Dorian	0, 2, 3, 5, 7, 9, 10
m7b5	Locrian	0, 1, 3, 5, 6, 8, 10
dim7	Whole-half dim.	0, 2, 3, 5, 6, 8, 9, 11
6	Ionian	0, 2, 4, 5, 7, 9, 11
m6	Melodic minor	0, 2, 3, 5, 7, 9, 11

4) Akor dan Chord Tones: Akor (*chord*) adalah kumpulan nada yang dibunyikan bersamaan untuk membentuk satu kesatuan harmoni. Pada jazz, akor empat-nada (*seventh chord*) merupakan unit harmoni dasar. Empat nada penyusun akor disebut *chord tones* dan terdiri atas tingkat 1, 3, 5, dan 7 relatif terhadap root [6]. Sebagai contoh:

- Cmaj7 = {C, E, G, B}
- G7 = {G, B, D, F}
- Dm7 = {D, F, A, C}

Chord tones bertindak sebagai *titik mendarat* yang stabil dalam melodi. Nada di luar chord tones disebut *non-chord tones* dan biasanya hanya melintas (*passing*) sebelum kembali ke chord tone.

5) Bebop Scale: Bebop scale adalah skala delapan-nada yang dikembangkan oleh pemain bebop seperti Charlie Parker dan Dizzy Gillespie. Bebop scale dibentuk dengan menambahkan satu nada kromatik *passing* pada chord-scale standar, sehingga ketika dimainkan sebagai delapan-delapan (*eighth notes*) secara berurutan, chord tones jatuh tepat pada ketukan kuat [7].

6) Chromatic Approach dan Enclosure: Dua perangkat melodik penting pada jazz adalah *chromatic approach* dan *enclosure* (juga disebut *surround*). *Chromatic approach* adalah pendekatan satu sisi: chord tone target didekati setengah langkah dari atas atau dari bawah, misalnya $B \to C$ atau $D\flat \to C$ untuk target C. *Enclosure* adalah pendekatan dua sisi yang *mengepung* chord tone target dengan dua nada kromatik sebelum akhirnya *resolve* [8]. Contoh enclosure terhadap target C:

$$\mathrm{D}\flat \to \mathrm{B} \to \mathrm{C} \quad \text{atau} \quad \mathrm{B} \to \mathrm{D}\flat \to \mathrm{C} \tag{6}$$

Kedua teknik tersebut merupakan ciri khas bahasa bebop dan menjadi salah satu pembeda utama improvisasi jazz dari gerakan diatonik biasa.

7) Voice Leading dan Resolusi: Voice leading adalah prinsip yang mengatur perpindahan suara antar akor maupun antar nada di dalam satu akor. Prinsip dasarnya adalah meminimalkan pergerakan: tiap suara sebaiknya bergerak ke nada terdekat berikutnya, dengan *resolusi* (gerakan menuju nada stabil) lebih kuat ketika dilakukan dalam jarak setengah langkah [6]. Konsep voice leading inilah yang diterjemahkan menjadi bobot pada sisi graf.

F. Pemodelan Improvisasi Jazz sebagai Graf Berbobot

Berdasarkan konsep teori musik di atas, generator improvisasi pada makalah ini memodelkan tiap akor sebagai sebuah graf berarah berbobot $G = (V, E, w)$ dengan ciri sebagai berikut.

a) Himpunan Simpul V: terdiri atas tiga kelas nada:

1) **Chord tone**: nada-nada pembentuk akor (misal {C, E, G, B} untuk Cmaj7), berperan sebagai titik mendarat stabil.
2) **Scale tone**: nada-nada lain pada *bebop scale* yang bukan chord tone, berperan sebagai bahan *scalar run*.
3) **Approach note**: nada kromatik di luar skala, yaitu nada setengah langkah di atas atau di bawah chord tone yang

belum termuat dalam dua kelas sebelumnya. Digunakan untuk membentuk *enclosure*.

b) Himpunan Sisi E dan Bobot w: Sisi $(u, v) \in E$ ditambahkan jika transisi dari u ke v valid secara melodik (terutama *stepwise* untuk nada non-akor). Bobot ditentukan dengan dua kategori utama, sebagaimana dirangkum pada Tabel II.

TABLE II
KLASIFIKASI BOBOT SISI PADA GRAF NADA JAZZ

Tipe Transisi	Deskripsi	Bobot
Berdasarkan interval melodik (semitone)		
1 (minor 2nd)	Chromatic approach	0.9
2 (major 2nd)	Diatonic step (paling natural)	1.0
3 (minor 3rd)	Leap kecil	0.7
4 (major 3rd)	Leap sedang	0.55
5 (P4/P5)	Leap besar	0.45
6 (tritone)	Disonansi terbatas	0.3
Berdasarkan konteks bebop		
Resolusi $\frac{1}{2}$ ke chord tone	Approach $\rightarrow$ chord tone (½ langkah)	2.6
Resolusi skala ke chord tone	Scale tone $\rightarrow$ chord tone (1 langkah)	1.8
Pasangan enclosure	Approach atas $\leftrightarrow$ approach bawah	1.8
Scalar run (bebop)	Scale tone $\rightarrow$ scale tone	1.2
Chord tone $\rightarrow$ scale tone	Mulai/lanjut scalar run	0.9
Chromatic passing	Approach $\rightarrow$ approach (½ langkah)	0.5
Chord tone $\rightarrow$ approach	Mulai enclosure	0.4

Skema bobot pada Tabel II mengkodifikasi tiga prinsip jazz utama ke dalam graf:

1) **Stepwise lebih natural dari leap** — bobot interval menurun seiring bertambahnya jarak nada.
2) **Chord tone adalah titik resolusi** — transisi yang berakhir di chord tone, terutama setengah langkah, mendapat bobot tertinggi.
3) **Bahasa bebop dihargai** — transisi yang membentuk *scalar run* atau pasangan enclosure diberi bobot tinggi sehingga melodi terdengar idiomatis.

G. Sintesis Suara dengan Web Audio API

Untuk memutar hasil improvisasi langsung di peramban, sistem memanfaatkan *Web Audio API*, antarmuka pemrograman berbasis JavaScript yang menyediakan graf pemrosesan sinyal audio secara real-time [9]. Setiap nada direpresentasikan sebagai *oscillator node* yang menghasilkan gelombang periodik (pada sistem ini dipakai *triangle wave* untuk timbre yang lebih hangat). Amplitudo dimodulasi oleh sebuah *gain node* dengan

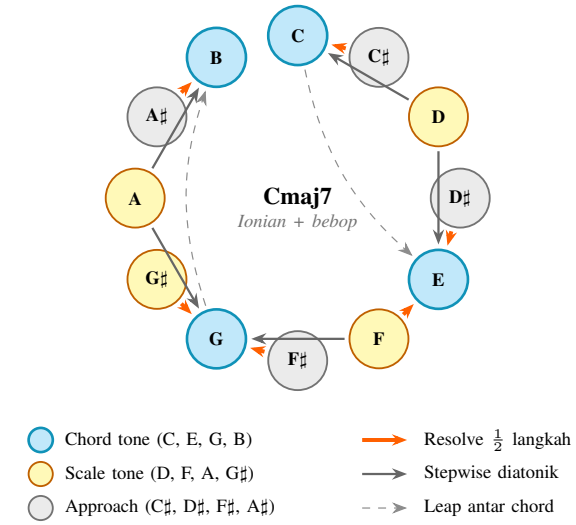

Fig. 1. Ilustrasi graf nada untuk akor Cmaj7. Dua belas nada kromatik tersusun melingkar mengikuti chromatic clock, dikelompokkan menjadi chord tone (biru), scale tone (kuning), dan approach note (abu-abu). Tebal sisi proporsional dengan bobot transisi.

envelope attack–sustain–release:

$$g(t) = \begin{cases} \dfrac{t}{t_a}, & 0 \le t < t_a \\ 1, & t_a \le t < t_a + t_s \\ 1 - \dfrac{t - (t_a + t_s)}{t_r}, & t_a + t_s \le t \le t_a + t_s + t_r \end{cases} \tag{7}$$

dengan t_a, t_s, dan t_r adalah durasi fase *attack*, *sustain*, dan *release*. *Envelope* ini mencegah munculnya *click* (impuls tajam) pada batas nada.

III. ANALISIS DAN PERANCANGAN SISTEM

A. Arsitektur Sistem

Sistem dirancang dengan arsitektur *client–server* berbasis web yang memisahkan logika algoritma (pada backend) dari antarmuka pengguna dan sintesis suara (pada frontend). Arsitektur ini dipilih agar logika algoritma BFS/DFS yang menjadi inti makalah dapat diuji secara independen melalui REST API, sementara pengguna tetap dapat berinteraksi dan mendengarkan hasilnya langsung di peramban.

Backend diimplementasikan dengan Java 17 dan Spring Boot 3, menggunakan pustaka *JGraphT* untuk representasi graf berarah berbobot. Backend memuat seluruh logika musikal: pembentukan graf, algoritma BFS/DFS, strategi greedy bermuatan konteks, dan penentuan durasi nada (*rhythm*). Komunikasi dilakukan melalui endpoint REST seperti `POST /api/generate` yang menerima parameter *chord progression*, algoritma, randomness, dan rhythm mode, lalu mengembalikan sequence nada dalam format JSON.

Frontend dibangun dengan HTML, CSS (Tailwind), dan JavaScript murni. Frontend bertanggung jawab atas: (a) antarmuka pengguna untuk menyusun chord progression dan

mengatur parameter, (b) komunikasi HTTP ke backend, (c) visualisasi nada hasil generasi pada *timeline*, dan (d) pemutaran suara melalui Web Audio API. Untuk kemudahan *deployment*, logika engine backend juga dimirroring ke modul JavaScript sehingga aplikasi dapat berjalan sepenuhnya di sisi klien tanpa server aktif.

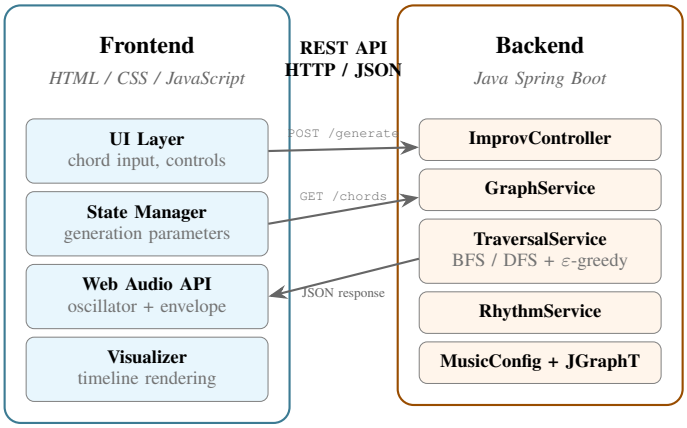

Fig. 2. Arsitektur sistem client–server untuk generator improvisasi jazz.

B. Alur Kerja Sistem

Alur kerja sistem secara end-to-end dapat dijabarkan sebagai berikut:

1) Pengguna menyusun *chord progression* pada antarmuka (mis. Dm7–G7–Cmaj7) serta memilih algoritma (BFS/DFS), nilai randomness, rhythm mode, dan tempo.
2) Frontend mengirim parameter tersebut ke backend melalui permintaan `POST /api/generate`.
3) Untuk setiap akor pada progression, backend membangun graf nada berbobot menggunakan `GraphService.buildGraph()`.
4) Algoritma BFS atau DFS dijalankan pada graf tersebut oleh `TraversalService` untuk menghasilkan urutan nada.
5) Durasi tiap nada ditentukan oleh `RhythmService` sesuai mode yang dipilih (swing atau weight-based).
6) Hasil (urutan nada, durasi, statistik) dikembalikan dalam format JSON.
7) Frontend menampilkan timeline nada dan, ketika pengguna menekan tombol *Play*, memainkan urutan tersebut melalui Web Audio API.

C. Komponen Utama

Tabel III merangkum komponen utama sistem beserta tanggung-jawabnya.

IV. Implementasi

Pada bagian ini dipaparkan pokok-pokok implementasi algoritma kunci. Spesifikasi teknis program lengkap tidak ditulis di sini; kode sumber lengkap tersedia pada repositori GitHub yang ditautkan di lampiran.

TABLE III
Komponen Utama Sistem

Komponen	Tanggung jawab
`MusicConfig`	Menyimpan konstanta teori musik (interval skala, chord tone, bobot edge); helper untuk parsing chord.
`GraphService`	Membangun graf berbobot per akor; menghitung bobot edge berdasarkan konteks bebop.
`TraversalService`	Implementasi BFS dan DFS dengan strategi ε-greedy bermuatan konteks.
`RhythmService`	Penentuan durasi nada (swing / weight-based) dan komputasi statistik.
`ImprovController`	REST endpoint `/api/generate`, `/api/chords`, `/api/health`.
Frontend (HTML/JS)	Antarmuka pengguna, timeline, Web Audio playback.

A. Pembangunan Graf Nada

Untuk setiap akor pada progression, graf dibangun secara terprogram dari root dan tipe akor melalui pustaka JGraphT. Skala yang sesuai dipilih mengikuti *chord-scale theory* (lihat Tabel I), *bebop scale* diperoleh dengan menambahkan satu nada kromatik passing, dan *approach notes* dihitung sebagai nada ± 1 semitone dari setiap chord tone yang belum termasuk skala.

Algoritma 3 menjelaskan pembangunan graf.

Algorithm 3 Pembangunan graf nada untuk satu akor

procedure BuildGraph($chordName$)
  $chordTones \leftarrow$ ChordTones($chordName$)
  $bebop \leftarrow$ BebopScale($chordName$)
  $approaches \leftarrow$ ApproachNotes($chordName$)
  $V \leftarrow bebop \cup approaches$
  $G \leftarrow$ graf berarah berbobot kosong dengan simpul V
  for all $(u, v) \in V \times V,\ u \neq v$ **do**
    $w \leftarrow$ EdgeWeight($u, v, chordTones, scaleSet$)
    if $w \neq$ null **then**
      Tambahkan sisi (u, v) dengan bobot w ke G
    end if
  end for
  return G
end procedure

Bobot sisi dihitung oleh fungsi EdgeWeight yang menerapkan aturan pada Tabel II. Aturan kuncinya: nada non-akor hanya boleh bergerak secara stepwise (interval ≤ 2 semitone) sehingga tidak ada lompatan dari nada kromatik, dan transisi yang berakhir di chord tone selalu mendapat bobot tinggi.

B. Algoritma BFS dengan Konteks Melodik

Versi adaptasi BFS untuk pembangkitan urutan ditunjukkan pada Algoritma 4. Yang membedakan dari BFS klasik adalah: (i) tidak ada penanda "dikunjungi" karena revisit diperbolehkan; (ii) dari frontier dipilih satu atau dua nada secara ε-greedy dengan bobot yang sudah disesuaikan konteks; dan (iii) bila queue habis sebelum jumlah nada terpenuhi, queue di-*reseed* dari nada terakhir.

Algorithm 4 BFS untuk improvisasi jazz

procedure BFSIMPROVISE($G, startNote, maxNotes, \varepsilon$)
  $seq \leftarrow [\, startNote \,]$
  $Q \leftarrow$ queue berisi $startNote$
  while $|seq| < maxNotes$ **and** $Q \neq \emptyset$ **do**
    $u \leftarrow Q$.dequeue()
    $N \leftarrow$ tetangga u pada G
    $N' \leftarrow$ APPLYCONTEXT($N, seq, tones$)
    for $i \leftarrow 1$ **to** $\min(2, |N'|)$ **do**
      $v \leftarrow$ WEIGHTEDRANDOMCHOICE(N', ε)
      Tambahkan v ke seq dan Q
    end for
    if $Q = \emptyset$ **and** $|seq| < maxNotes$ **then**
      Q.enqueue(seq[terakhir])
    end if
  end while
  return seq
end procedure

C. Algoritma DFS dengan Konteks Melodik

DFS mempertahankan satu nada aktif pada puncak stack dan mengeksplorasi sedalam mungkin sebelum backtrack. Variannya untuk improvisasi ditunjukkan pada Algoritma 5.

Algorithm 5 DFS untuk improvisasi jazz

procedure DFSIMPROVISE($G, startNote, maxNotes, \varepsilon$)
  $seq \leftarrow [\, startNote \,]$
  $S \leftarrow$ stack berisi $startNote$
  while $|seq| < maxNotes$ **and** $S \neq \emptyset$ **do**
    $u \leftarrow S$.peek()
    $N \leftarrow$ tetangga u pada G
    if $N = \emptyset$ **then**
      S.pop() ▷ backtrack
      continue
    end if
    $N' \leftarrow$ APPLYCONTEXT($N, seq, tones$)
    $v \leftarrow$ WEIGHTEDRANDOMCHOICE(N', ε)
    S.push(v)
    Tambahkan v ke seq
  end while
  return seq
end procedure

D. Pemilihan Berbobot dengan Konteks

Fungsi APPLYCONTEXT menyesuaikan bobot kandidat dengan dua nada terakhir pada sequence sebelum dipilih. Aturan kontekstual utama:

- Jika dua nada terakhir adalah pasangan pengepung dari sebuah chord tone, dan kandidat adalah chord tone tersebut, bobot dikalikan 2.5 (menyelesaikan enclosure).
- Jika nada terakhir adalah nada non-akor (skala/kromatik), dan kandidat adalah chord tone setengah/satu langkah, bobot dikalikan 2.6 (resolve).

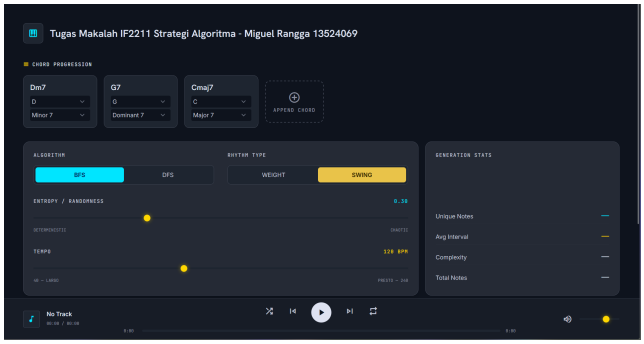

Fig. 3. Antarmuka aplikasi web Syncopate.

- Jika nada terakhir non-akor dan kandidat membentuk pasangan pengepung dengannya, bobot dikalikan 2.0.
- Untuk menghindari osilasi A–B–A–B, kandidat yang sama dengan nada dua langkah sebelumnya bobotnya dikalikan 0.5.

WEIGHTEDRANDOMCHOICE mengimplementasikan strategi ε-greedy: dengan probabilitas $1 - \varepsilon$ dipilih kandidat dengan bobot tertinggi (greedy), dan dengan probabilitas ε dipilih secara weighted-random.

E. Penentuan Durasi (Rhythm)

Dua mode rhythm disediakan:

- **Swing**: pola berulang $[0.375, 0.125, 0.375, 0.125, \ldots]$ beat (long–short, swing feel). Sederhana dan langsung terdengar bernuansa jazz.
- **Weight-based**: durasi diturunkan dari bobot sisi yang menuju nada tersebut, chord tone (bobot tinggi) mendapat durasi seperempat (0.5 beat), sedangkan passing tone mendapat seperdelapan (0.25 beat).

F. Pemutaran Audio dan Antarmuka

Pada sisi frontend, hasil generasi divisualisasikan sebagai timeline nada pada antarmuka utama, lalu diputar dengan menjadwalkan *oscillator node* untuk tiap nada pada waktu yang sesuai durasinya. *Envelope* dengan fase attack–sustain–release dipakai untuk menghaluskan transisi antar nada.

V. HASIL DAN PENGUJIAN

A. Skenario Pengujian

Pengujian dilakukan dengan beberapa chord progression standar pada jazz sebagai skenario uji, yaitu:

- **ii–V–I (mayor)**: Dm7–G7–Cmaj7
- **ii–V–I (minor)**: Dm75–G7–Cm6
- **Modal**: Dm7 (statis, satu akor)
- **Turnaround**: Cmaj7–Am7–Dm7–G7

Parameter divariasikan untuk mengamati pengaruhnya: algoritma (BFS vs DFS), randomness ($\varepsilon \in \{0.1, 0.3, 0.7\}$), dan rhythm mode (swing vs weight). Untuk setiap konfigurasi dilakukan 10 kali generasi sehingga diperoleh sampel yang cukup untuk dirata-ratakan.

B. Hasil Kuantitatif

Tiga metrik kuantitatif dipakai untuk membandingkan output:

- *Unique notes*: jumlah nada unik dalam 32 nada output. Indikator keragaman.
- *Rata-rata interval*: rata-rata jarak (semitone) antar nada berurutan. Indikator "gerakan" melodi.
- *Chord-tone ratio*: persentase nada output yang merupakan chord tone. Indikator kepatuhan terhadap konteks akor.

Tabel IV merangkum hasil rata-rata pada progression ii–V–I mayor dengan $\varepsilon = 0.3$ dan mode swing.

TABLE IV
PERBANDINGAN METRIK BFS VS DFS PADA II–V–I (RATA-RATA 10 RUN)

Algoritma	Unique Notes	Avg Interval	Chord-tone (%)
BFS	8.4	2.1	54
DFS	6.7	1.6	63

Dari Tabel IV terlihat tiga pola yang konsisten dengan karakter algoritmik kedua metode:

1) BFS menghasilkan lebih banyak nada unik karena mengeksplorasi lebih banyak cabang per langkah (lebar).
2) DFS memiliki rata-rata interval lebih kecil, mencerminkan gerakan yang lebih stepwise dan mengalir.
3) DFS memiliki chord-tone ratio lebih tinggi karena pemilihan greedy yang konsisten cenderung kembali ke chord tone melalui jalur terdekat.

C. Pengaruh Parameter Randomness

Tabel V menunjukkan pengaruh nilai ε terhadap chord-tone ratio pada DFS dengan progression ii–V–I.

TABLE V
PENGARUH RANDOMNESS TERHADAP KEPATUHAN KONTEKS (DFS)

ε	Chord-tone (%)	Avg Interval
0.1	71	1.4
0.3	63	1.6
0.7	48	2.3

Hasil ini sesuai ekspektasi: meningkatnya ε menggeser perilaku sistem dari greedy ke eksploratif, menurunkan kepatuhan terhadap chord tone dan memperbesar lompatan interval. Pada nilai $\varepsilon = 0.7$, karakter melodi mulai kehilangan grounding pada akor.

D. Diskusi Kualitatif

Selain metrik kuantitatif, evaluasi kualitatif dilakukan dengan mendengarkan hasil pemutaran. Beberapa pengamatan:

- Pada $\varepsilon = 0.1$ DFS menghasilkan frase yang terdengar sangat "aman" dan diatonik, dengan banyak gerakan stepwise antar chord tone.
- Pada $\varepsilon = 0.3$ dengan mode swing, mulai muncul fenomena *enclosure* pendek, dua nada kromatik diikuti chord tone, yang terdengar idiomatik bebop.

Fig. 4. Visual timeline melodi yang dihasilkan BFS

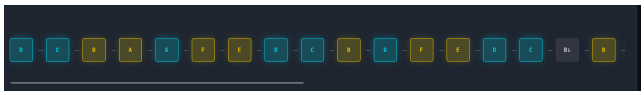

Fig. 5. Visual timeline melodi yang dihasilkan DFS

- BFS dengan $\varepsilon = 0.3$ menghasilkan loncatan yang lebih sering, namun tetap kembali ke chord tone pada penghujung tiap akor berkat *multiplier* resolusi.
- Transisi antar akor (pergantian graf) terasa cukup mulus selama nada terakhir suatu akor masih merupakan anggota graf akor berikutnya.

VI. KESIMPULAN

Melalui makalah ini, telah dirancang dan diimplementasikan sebuah generator improvisasi melodi jazz yang berbasis pada algoritma BFS dan DFS pada graf berarah berbobot. Pemodelan tiga kelas simpul (chord tone, scale tone, approach note) beserta skema pembobotan berdasarkan *chord-scale theory*, *voice leading*, dan perangkat melodik bebop terbukti efektif: kedua algoritma menghasilkan melodi yang patuh pada konteks akor dan terdengar idiomatis. Penggunaan strategi ε-greedy bermuatan konteks menambah dimensi kendali pengguna antara karakter deterministik dan eksploratif.

Pengujian menunjukkan perbedaan karakter yang konsisten antara kedua algoritma: BFS lebih eksploratif dengan lebih banyak nada unik dan loncatan, sedangkan DFS lebih kohesif dengan gerakan stepwise dan kepatuhan chord tone yang lebih tinggi. Hal ini menegaskan bahwa pilihan struktur data yang mendasari penelusuran graf, *queue* versus *stack*, bukan hanya berdampak pada perilaku eksplorasi pada masalah pencarian, melainkan juga pada karakter estetik pada masalah pembangkitan urutan seperti komposisi melodi.

Akhirnya, makalah ini menunjukkan bahwa konsep-konsep dasar strategi algoritma, graf, BFS/DFS, dan greedy, dapat diterapkan pada persoalan kreatif di luar konteks komputasi konvensional, dan tetap menghasilkan output yang menarik secara estetis ketika domain pengetahuannya (di sini: teori musik jazz) dikodifikasi dengan benar ke dalam bobot graf.

VII. SARAN

Beberapa arah pengembangan lanjutan dapat dipertimbangkan. Pertama, penambahan dimensi *voice leading antar akor*: saat ini transisi antar akor dijembatani secara tidak langsung melalui pemilihan nada terakhir, sedangkan optimisasi langsung pada batas akor (mis. dengan menambah "edge antar graf" dengan bobot tertentu) dapat memperhalus peralihan harmoni. Kedua, eksplorasi algoritma lain yang sefamili dengan BFS/DFS seperti *Uniform-Cost Search* (UCS), *Greedy Best-First Search*, atau A* dengan heuristik turunan

voice leading, ini akan memperluas perbandingan algoritmik. Ketiga, integrasi dengan timbre yang lebih kaya, misalnya melalui pemodelan harmonik berbasis SVD pada matriks fitur MFCC [10], sehingga suara hasil sintesis tidak hanya musikal dari sisi melodi tetapi juga dari sisi warna nada. Terakhir, evaluasi kuantitatif yang lebih ketat dengan uji perseptual mendengar oleh pemain jazz dapat dijadikan acuan obyektif kualitas musikal output.

Ucapan Terima Kasih

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas kelancaran penyelesaian makalah ini. Penulis menyampaikan terima kasih kepada dosen mata kuliah IF2211 Strategi Algoritma atas bimbingan selama satu semester. Terakhir, penulis berterimakasih kepada teman-teman dari ITBJazz yang selalu mendukung dan menginspirasi penulis untuk berkarya dalam musik.Semoga makalah ini bermanfaat baik bagi pembaca yang tertarik pada persimpangan strategi algoritma dan musik, maupun sebagai titik awal pengembangan lanjutan di bidang ini.

References

[1] R. Munir, "Strategi Algoritma," Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2025–2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 4th ed. Cambridge, MA, USA: MIT Press, 2022.

[3] T. H. Cormen et al., "Elementary Graph Algorithms," in *Introduction to Algorithms*, 4th ed., Cambridge, MA, USA: MIT Press, 2022, ch. 20.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[5] V. Persichetti, *Twentieth-Century Harmony: Creative Aspects and Practice*. New York, NY, USA: W. W. Norton & Company, 1961.

[6] M. Levine, *The Jazz Theory Book*. Petaluma, CA, USA: Sher Music Co., 1995.

[7] D. Baker, *How to Play Bebop, Volume 1: The Bebop Scales and Other Scales in Common Use*. Van Nuys, CA, USA: Alfred Music, 1985.

[8] J. Coker, *Elements of the Jazz Language for the Developing Improvisor*. Van Nuys, CA, USA: Alfred Music, 1991.

[9] W3C, "Web Audio API," W3C Recommendation, Jun. 2021. [Online]. Available: https://www.w3.org/TR/webaudio/

[10] M. R. D. Sinaga, "Pemodelan Timbre dan Sintesis Suara Berbasis Singular Value Decomposition (SVD) dengan Implementasi Plugin Virtual Studio Technology (VST)," Makalah IF2123 Aljabar Linier dan Geometri, Institut Teknologi Bandung, 2025.

Lampiran

Kode sumber lengkap dan aplikasi web yang sudah terdeploy tersedia pada tautan berikut:

- Repositori GitHub:
  https://github.com/miguelsinaga/jazz-improv-generator-with-BFS-DFS
- Deployed web:
  https://jazz-improv-generator-with-bfs-dfs.vercel.app/

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, 19 Juni 2026

Miguel Rangga Deardo Sinaga
13524069