Optimalisasi Urutan Lagu dalam *Setlist* Musik Menggunakan Graf Berbobot dan Hamiltonian Path

Studi Kasus *P1Harmony at Busan One Asia Festival 2025*

Safira Berlianti - 13524128
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: fira.berlianti@gmail.com , 13524128@std.stei.itb.ac.id

***Abstract*— Penyusunan urutan lagu (*setlist*) dalam konser musik memiliki pengaruh besar terhadap alur emosi dan pengalaman penonton. Transisi antar lagu yang kurang mulus dapat mengganggu imersi dan mengurangi kekuatan artistik pertunjukan. Makalah ini mengusulkan pendekatan matematis untuk mengoptimalkan urutan lagu dalam sebuah *setlist* menggunakan graf berbobot dan algoritma lintasan Hamilton. Setiap lagu direpresentasikan sebagai simpul, sedangkan bobot sisi antar lagu dihitung berdasarkan selisih parameter musikal seperti BPM, key, energy, danceability, happiness, dan loudness. Studi kasus dilakukan pada *P1Harmony at Busan One Asia Festival 2025* dengan membandingkan urutan asli *setlist* dan urutan hasil optimasi menggunakan dua pendekatan algoritmik: brute force dan greedy.**

Keywords— graf berbobot, Hamiltonian Path, optimasi setlist, transisi lagu, algoritma greedy, brute force

I. PENDAHULUAN

Konser musik merupakan salah satu elemen penting dalam stategi promosi suatu grup musik atau musisi, sekaligus menjadi momen interaksi langsung yang dinantikan oleh penggemar. Pengalaman konser yang berkesan tidak hanya bergantung pada kualitas vokal dan visual, tetapi juga pada alur musik yang kuat melalui susunan lagu atau *setlist*. Transisi antar lagu yang kurang tepat dapat mengganggu imersi penonton dan melemahkan kesan artistik yang dibangun oleh musisi. Oleh karena itu, penting untuk mempertimbangkan transisi lagu secara musikal. Tantangan dalam menyusun *setlist* adalah bagaimana menentukan urutan lagu yang mampu mempertahankan alur emosional dan energi konser. Faktor seperti keselarasan nada (key), tempo (BPM), energi, serta mood dari masing-masing lagu harus dipertimbangkan agar transisi antar lagu terasa alami.

Untuk mengatasi tantangan ini, permasalahan penyusunan *setlist* dapat dimodelkan menggunakan lintasan Hamilton dalam graf berbobot. Dalam model ini, setiap lagu direpresentasikan sebagai simpul, sementara sisi menggambarkan kemungkinan transisi antar lagu dengan bobot yang mencerminkan tingkat ketidakmulusan transisi. Bobot dapat ditentukan berdasarkan selisih parameter seperti perbedaan tempo, kunci musik, kesamaan tema dan lain-lain. Dengan pendekatan ini, akan dicari jalur terpendek yang melewati semua simpul tepat sekali sehingga total bobot transisi minimal. Untuk kasus dengan jumlah lagu terbatas, lintasan Hamilton dapat ditentukan melalui pendekatan brute force atau algoritma greedy. Meskipun pendekatan ini memiliki keterbatasan dalam skala besar, pendekatan ini cukup efektif untuk eksplorasi awal terhadap *setlist* dengan jumlah lagu yang tidak terlalu banyak.

Makalah ini mengusulkan pendekatan optimasi urutan lagu menggunakan teori graf berbobot dan Hamiltonian Path, dengan studi kasus *setlist P1Harmony at Busan One Asia Festival 2025*. Pemilihan studi kasus ini didasarkan pada data parameter musikal yang ada. Makalah ini diharapkan dapat menunjukkan bagaimana konsep matematika diskrit, khususnya teori graf, dapat diterapkan dalam bidang yang kreatif seperti musik.

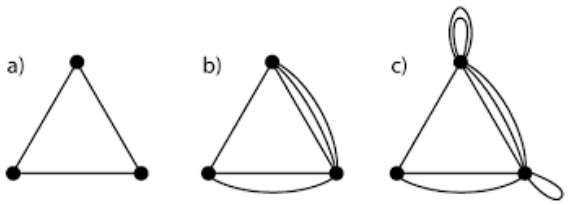
Makalah ini terdiri dari lima bagian: (1) Pendahuluan, (2) Studi Literatur, (3) Metodologi, (4) Hasil dan pembahasan, dan (5) Kesimpulan.

II. STUDI LITERATUR

A. Definisi Graf

Graf dapat didefinisikan sebgai pasangan himpunan (V, E). Dalam hal ini, V adalah himpunan tidak kosong dari simpul-simpul (*vertices*), sementara E adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul.

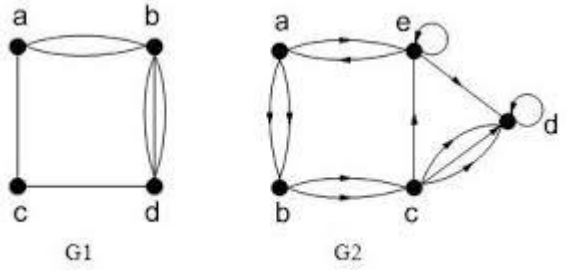
Graf dapat digolongkan menjadi dua macam berdasarkan ada tidaknya gelang atau sisi ganda di dalam graf, yaitu graf sederhana dan graf tak-sederhana. Graf sederhana adalah graf yang tidak mengandung gelang maupun sisi ganda. Graf tak-sederhana adalah graf yang mengandung sisi ganda atau gelang. Graf tak-sederhana dapat dibedakan lagi menjadi dua, yaitu graf ganda—graf yang mengandung sisi ganda—dan graf semu—graf yang mengandung sisi gelang.



Gambar 1 Ilustrasi graf sederhana (a), graf ganda (b), dan graf semu (c)

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Graf dapat digolongkan menjadi dua macam berdasarkan orientasi arah pada sisinya, yaitu graf tak-berarah dan graf berarah. Graf tak-berarah adalah graf yang sisinya tidak memiliki orientasi arah. Graf berarah adalah graf yang setiap sisinya memiliki orientasi arah.



Gambar 2 Ilustrasi graf tak-berarah (G1) dan graf berarah (G2)

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

B. Terminologi Graf

1) Ketetanggaan
Dua simpul dikatakan bertetangga apabila terdapat sisi yang menghubungkan kedua simpul tersebut tanpa melalui simpul perantara.

2) Bersisian
Suatu sisi dikatakan bersisian dengan simpul tertentu apabila sisi tersebut menghubungkan simpul yang dimaksud dengan simpul lain.

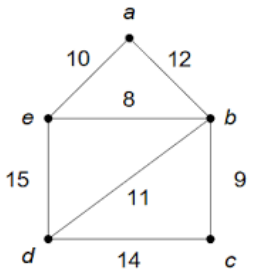
3) Derajat
Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut.

4) Lintasan
Lintasan adalah suatu rangkaian simpul dan sisi yang membentuk jalur dari satu simpul ke simpul lainnya.

5) Siklus
Siklus adalah suatu lintasan yang berawal dan berakhir pada simpul yang sama.

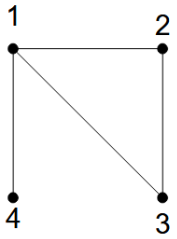
6) Keterhubungan
Dua simpul dikatakan terhubung jika terdapat lintasan yang menghubungkan kedua lintasan tersebut, baik secara langsung maupun tidak langsung.

7) Graf Berbobot
Graf berbobot adalah graf yang setiap sisinya memiliki nilai tertentu (bobot). Graf berbobot sangat berguna dalam pemodelan masalah optimasi seperti pencarian rute terpendek.



Gambar 3 Ilustrasi graf berbobot
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

8) Lintasan Hamilton
Lintasan Hamilton adalah lintasan yang melalui tiap simpul di dalam graf tepat satu kali.



Gambar 4 Ilustrasi graf yang memiliki lintasan Hamilton (misal: 3, 2, 1, 4)
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian3-2024.pdf

C. Algoritma Greedy

Algoritma greedy adalah pendekatan pemecahan masalah yang membuat pilihan terbaik pada setiap langkah, tanpa mempertimbangkan konsekuensi jangka panjang dari pilihan tersebut. Pada algoritma ini, keputusan yang sudah dibuat tidak dapat diubah di langkah selanjutnya. Beberapa contoh penerapan algoritma greedy yaitu sebagai berikut.

- Masalah knapsack: memilih barang dengan nilai tertinggi per berat untuk memaksimalkan kapasitas tas.
- Penjadwalan aktivitas: memilih aktivitas dengan waktu selesai tercepat untuk memaksimalkan jumlah aktivitas yang bisa dilakukan.
- Pohon Merentang Minimum (MST): algoritma Prim dan Kruskal menggunakan pendekatan greedy untuk menghubungkan semua simpul dengan bobot total.

Berikut langkah-langkah dari algoritma greedy.

1. Solusi awal dimulai dengan himpunan kosong (belum ada keputusan yang dibuat).
2. Pada setiap iterasi, algoritma menambahkan satu elemen ke dalam himpunan solusi. Elemen ini dipilih berdasarkan kriteria *optimal lokal* (misalnya: nilai terbesar, biaya terkecil, atau parameter lain yang relevan dengan masalah).

3. Setelah elemen dipilih, diperiksa apakah himpunan solusi tetap *feasible* (layak/memenuhi constraints masalah).
4. Proses berhenti ketika himpunan solusi telah memenuhi kondisi penyelesaian masalah.

Algoritma greedy efisien secara komputasi karena kompleksitas waktunya yang rendah. Namun, algoritma ini tidak selalu menghasilkan solusi optimal global (misalnya pada masalah *Travelling Salesman).*

D. Algoritma Brute Force

Algoritma brute force adalah pendekatan pemecahan masalah yang mengevaluasi seluruh kemungkinan solusi secara menyeluruh untuk menemukan hasil yang optimal. Berikut langkah-langkah utama dari algoritma brute force dalam konteks lintasan Hamilton.

1. Membangun semua urutan mungkin dari n simpul (n! kemungkinan).
2. Menghitung total bobot dari setiap lintasan.
3. Memilih lintasan dengan bobot terkecil sebagai solusi.

Kelebihan utama dari pendekatan ini adalah kemampuannya menjamin solusi optimal global karena tidak ada kemungkinan solusi terbaik terlewatkan. Pendekatan ini juga sederhana untuk diimplementasikan karena tidak membutuhkan logika optimasi yang kompleks. Namun, kelemahannya terletak pada kompleksitas waktunya yang faktorial, yaitu (O(n!)). Hal ini menjadikan algoritma brute force tidak cocok digunakan untuk jumlah elemen yang besar.

III. METODOLOGI

A. Pemodelan Graf

Dalam makalah ini, susunan lagu dalam konser dimodelkan sebagai graf berbobot tak berarah, di mana:

- Setiap simpul mereprepresentasikan lagu yang ada di *setlist* konser .
- Setiap sisi menunjukkan kemungkinan transisi antara satu lagu dengan lagu yang lainnya.
- Bobot pada setiap sisi dihitung berdasarkan perbedaan nilai parameter antara kedua lagu yang bertetangga.

Tujuan dari pemodelan ini adalah mencari lintasan Hamilton berbobot minimum, yaitu lintasan yang melewati setiap simpul tepat satu kali dan meminimalkan total bobot transisi antar lagu.

B. Penentuan Parameter

Bobot antar lagu dihitung berdasarkan selisih dari beberapa parameter musikal yang diperoleh dari Tunebat. Berikut parameter-parameter yang digunakan untuk pemodelan graf.

- *Key,* menunjukkan tangga nada utama dari suatu lagu, dikodekan ke angka 0–11. Perbedaan antar key dihitung menggunakan *circular distance* dalam sistem 12 nada.
- *BPM,* menunjukkan tempo lagu dalam ketukan per menit.
- *Energy,* menunjukkan seberapa intens dan aktif suatu lagu berdasarkan *general entropy*, *onset rate*, timbre, kekerasan suara, dan rentang dinamis. Nilai parameter ini berkisar antara 0—100.
- *Danceability,* menunjukkan seberapa cocok suatu lagu untuk menari berdasarkan kestabilan irama, kekuatan ketukan, dan tempo keseluruhan. Nilai parameter ini berkisar antara 0—100.
- *Happiness,* menunjukkan seberapa ceria dan positif suasana lagu. Nilai parameter ini berkisar antara 0—100.
- *Loudness,* menunjukkan amplitudo rata-rata dalam desibel. Nilai parameter ini berkisar antara -60 dB hingga 0 dB.

C. Penentuan Bobot Transisi

Untuk menghitung bobot sisi antara dua lagu A dan B, digunakan rumus:

$$Bobot(A,B) = \sum_{i=1}^{6} \alpha_i \left(\frac{\triangle Parameter_i}{Range_i}\right)$$

dengan

Parameter	Bobot (α)	Range
BPM	0.30	50 (dari 80–130 BPM)
Key	0.20	6
Energy	0.20	100
Danceability	0.10	100
Happiness	0.10	100
Loudness	0.10	60

Tabel 1 Parameter bobot transisi graf

Berdasarkan perhitungan bobot antar semua pasangan lagu, dibentuk sebuah graf lengkap berbobot. Graf ini direpresentasikan dalam bentuk matriks ketetanggaan, dengan setiap elemen [i][j] berisi nilai *cost* antara lagu ke-i dan ke-j.

D. Strategi Algoritma

1) Pendekatan greedy

Algoritma greedy tidak menjamin solusi optimal secara global, tetapi memberikan hasil yang cukup baik dengan kompleksitas waktu yang jauh lebih rendah ($O(n^2)$) dibandingkan pendekatan brute force (O(n!)). Metode ini cocok digunakan untuk jumlah lagu yang tidak terlalu besar

Dalam implementasi algoritma ini, lagu "DUH!" ditetapkan sebagai simpul awal. Penetapan ini didasarkan pada dua pertimbangan utama. Pertama, setlist asli konser P1Harmony di Busan One Asia Festival 2025 memang dibuka dengan lagu tersebut, sehingga mempertahankan urutan ini memberikan landasan yang lebih realistis dan relevan dengan konteks acara sebenarnya. Kedua, lagu *"DUH!"* merupakan lagu comeback terbaru P1Harmony pada saat konser tersebut berlangsung. Oleh karena itu, memulai pertunjukan dengan lagu ini diharapkan memiliki signifikansi promosi yang tinggi.

Urutan pertama dalam *setlist* sendiri sering dipilih secara strategis untuk menarik perhatian penonton dan media.

Implementasi algoritma tersebut dapat dilihat pada potongan kode berikut.

a) Data lagu beserta parameternya

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

# Data 6 lagu
songs = {
    "DUH!": {
        "key": 2, "bpm": 103, "energy": 73,
        "danceability": 80, "happiness": 70, "loudness": -3
    },
    "Heartbeat Drum": {
        "key": 8, "bpm": 130, "energy": 74,
        "danceability": 88, "happiness": 77, "loudness": -4
    },
    "SAD SONG": {
        "key": 10, "bpm": 80, "energy": 95,
        "danceability": 62, "happiness": 91, "loudness": -1
    },
    "Pretty Boy": {
        "key": 8, "bpm": 102, "energy": 84,
        "danceability": 86, "happiness": 49, "loudness": -3
    },
    "Follow Me": {
        "key": 4, "bpm": 128, "energy": 80,
        "danceability": 66, "happiness": 69, "loudness": -2
    },
    "Last Call": {
        "key": 7, "bpm": 130, "energy": 86,
        "danceability": 62, "happiness": 86, "loudness": -2
    },
}
```

Gambar 4 Potongan kode data lagu beserta parameternya

b) Bobot transisi antar lagu

```
# Hitung bobot transisi antar lagu
def calculate_weight(song1, song2):
    bpm_diff = abs(song1["bpm"] - song2["bpm"]) / 50
    key_diff = min(abs(song1["key"] - song2["key"]), 12 - abs(song1["key"] - song2["key"])) / 6
    energy_diff = abs(song1["energy"] - song2["energy"]) / 100
    dance_diff = abs(song1["danceability"] - song2["danceability"]) / 100
    happy_diff = abs(song1["happiness"] - song2["happiness"]) / 100
    loud_diff = abs(song1["loudness"] - song2["loudness"]) / 60

    weight = (0.30 * bpm_diff + 0.20 * key_diff + 0.20 * energy_diff +
              0.10 * dance_diff + 0.10 * happy_diff + 0.10 * loud_diff)
    return weight

# Matriks bobot antar lagu
song_names = list(songs.keys())
n = len(song_names)
weight_matrix = np.zeros((n, n))

for i in range(n):
    for j in range(n):
        if i != j:
            weight_matrix[i][j] = calculate_weight(songs[song_names[i]], songs[song_names[j]])

print("Matriks Bobot:\n", weight_matrix, "\n")
for i in range(n):
    for j in range(i+1, n):
        print(f"Bobot({song_names[i]}, {song_names[j]}): {weight_matrix[i][j]:.4f}")
```

Gambar 5 Potongan kode bobot transisi antar lagu

c) Penerapan algoritma greedy Hamiltonian Path

```
# Algoritma greedy Hamiltonian Path (simpul pertama = "DUH!")
def greedy_hamiltonian(weight_matrix, song_names, start="DUH!"):
    visited = set()
    path = [start]
    total_weight = 0
    current = song_names.index(start)
    visited.add(current)

    while len(visited) < n:
        min_weight = float('inf')
        next_node = None
        for i in range(n):
            if i not in visited and weight_matrix[current][i] < min_weight:
                min_weight = weight_matrix[current][i]
                next_node = i
        if next_node is not None:
            path.append(song_names[next_node])
            total_weight += weight_matrix[current][next_node]
            visited.add(next_node)
            current = next_node

    return path, total_weight

optimal_path, total_weight = greedy_hamiltonian(weight_matrix, song_names, start="DUH!")
print(f"\nUrutan Optimal : {optimal_path}")
print(f"Total Bobot: {total_weight:.4f}")
```

Gambar 6 Potongan kode penerapan algoritma greedy Hamiltonian Path

d) Perbandingan dengan setlist yang sebenarnya

```
# Bandingkan dengan setlist asli
setlist_asli = ["DUH!", "Heartbeat Drum", "SAD SONG", "Pretty Boy", "Follow Me", "Last Call"]

def hitung_bobot_setlist(setlist):
    total = 0
    for i in range(len(setlist) - 1):
        idx1 = song_names.index(setlist[i])
        idx2 = song_names.index(setlist[i+1])
        total += weight_matrix[idx1][idx2]
    return total

bobot_asli = hitung_bobot_setlist(setlist_asli)
print(f"\nSetlist Asli: {setlist_asli}")
print(f"Bobot Setlist Asli: {bobot_asli:.4f}")
```

Gambar 7 Potongan kode perbandingan dengan *setlist* asli

e) Visualisasi graf

```
# Visualisasi graf
G = nx.Graph()
for i in range(n):
    for j in range(i+1, n):
        G.add_edge(song_names[i], song_names[j], weight=weight_matrix[i][j])

pos = nx.spring_layout(G, seed=42)
plt.figure(figsize=(10, 8))
nx.draw(G, pos, with_labels=True, node_color='lightcoral', node_size=1500)
edge_labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels={k: f"{v:.2f}" for k, v in edge_labels.items()}, font_size=8)
plt.title("Graf Transisi Antar Lagu (Bobot = Ketidakmulusan)")
plt.show()
```

Gambar 8 Potongan kode visualisasi graf

2) Pendekatan brute force

Pendekatan brute force menawarkan solusi optimal secara global dengan cara mengevaluasi semua kemungkinan lintasan Hamilton yang mungkin. Dalam kasus ini, karena jumlah lagu hanya enam, maka jumlah permutasi lintasan yang dapat dievaluasi adalah 6! = 720. Kompleksitas waktu dari algoritma brute force berskala faktorial (O(n!)) sehingga tidak efisien jika diterapkan pada kasus dengan jumlah lagu yang jauh lebih banyak.

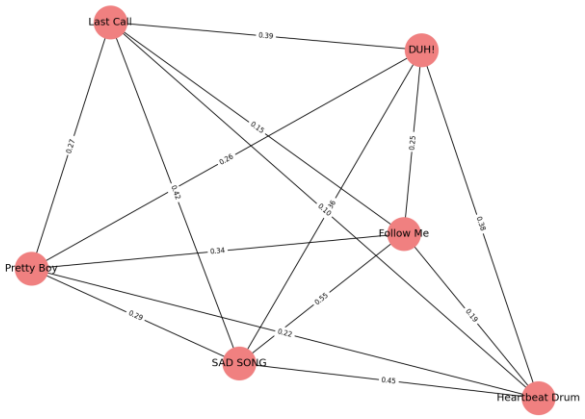
Perbedaan antara pendekatan ini dan pendekatan greedy terletak pada bagian penerapan algoritmanya. Tidak ada perubahan yang signifikan pada bagian lainnya. Berikut potongan kodenya.

```python
# Cari transisi paling optimal
def find_optimal_path(weight_matrix, song_names):
    n = len(song_names)
    min_weight = float('inf')
    best_path = []

    for path in permutations(range(n)):
        total_weight = 0
        for i in range(n-1):
            total_weight += weight_matrix[path[i]][path[i+1]]

        if total_weight < min_weight:
            min_weight = total_weight
            best_path = [song_names[idx] for idx in path]

    return best_path, min_weight

optimal_path, total_weight = find_optimal_path(weight_matrix, song_names)
print(f"\nUrutan Optimal: {optimal_path}")
print(f"Total Bobot: {total_weight:.4f}")
```

Gambar 9 Potongan kode penerapan algoritma brute force

```
Matriks Bobot:
 [[0.         0.38066667 0.35766667 0.255      0.24733333 0.39033333]
 [0.38066667 0.         0.45366667 0.21966667 0.19066667 0.09566667]
 [0.38066667 0.         0.45366667 0.21966667 0.19066667 0.09566667]
 [0.35766667 0.45366667 0.         0.29       0.54566667 0.42466667]
 [0.255      0.21966667 0.29       0.         0.339      0.268     ]
 [0.24733333 0.19066667 0.54566667 0.339      0.         0.145     ]
 [0.39033333 0.09566667 0.42466667 0.268      0.145      0.        ]]

Bobot(DUH!, Heartbeat Drum): 0.3807
Bobot(DUH!, SAD SONG): 0.3577
Bobot(DUH!, Pretty Boy): 0.2550
Bobot(DUH!, Follow Me): 0.2473
Bobot(DUH!, Last Call): 0.3903
Bobot(Heartbeat Drum, SAD SONG): 0.4537
Bobot(Heartbeat Drum, Pretty Boy): 0.2197
Bobot(Heartbeat Drum, Follow Me): 0.1907
Bobot(Heartbeat Drum, Last Call): 0.0957
Bobot(SAD SONG, Pretty Boy): 0.2900
Bobot(SAD SONG, Follow Me): 0.5457
Bobot(SAD SONG, Last Call): 0.4247
Bobot(Pretty Boy, Follow Me): 0.3390
Bobot(Pretty Boy, Last Call): 0.2680
Bobot(Follow Me, Last Call): 0.1450
```

Gambar 10 Hasil output bobot graf



Gambar 11 Hasil graf berdasarkan program yang dibuat

1) Urutan optimal dengan pendekatan greedy

```
Urutan Optimal: ['DUH!', 'Follow Me', 'Last Call', 'Heartbeat Drum', 'Pretty
Boy', 'SAD SONG']
Total Bobot: 0.9977

Setlist Asli: ['DUH!', 'Heartbeat Drum', 'SAD SONG', 'Pretty Boy', 'Follow Me
', 'Last Call']
Bobot Setlist Asli: 1.6083
```

Gambar 12 Urutan optimal dengan pendekatan brute force

2) Urutan optimal dengan pendekatan greedy

```
Urutan Optimal: ['DUH!', 'Follow Me', 'Last Call', 'Heartbeat Drum', 'Pretty
Boy', 'SAD SONG']
Total Bobot: 0.9977

Setlist Asli: ['DUH!', 'Heartbeat Drum', 'SAD SONG', 'Pretty Boy', 'Follow Me
', 'Last Call']
Bobot Setlist Asli: 1.6083
```

Gambar 13 Urutan optimal dengan pendekatan greedy

IV. HASIL DAN PEMBAHASAN

A. Persiapan Data

Makalah ini menggunakan data enam lagu yang dibawakan oleh P1Harmony pada *Busan One Asia Festival 2025,* yang didapat dari situs setlist.fm. Setiap lagu memiliki enam parameter sebagai dasar penentuan bobot transisi antar lagu, data ini didapat dari situs Tunebat. Berikut tabel parameter untuk setaiap lagu.

Lagu	Key	BPM	Ener-gy	Dance-ability	Happi-ness	Loud-ness
DUH!	B Minor	103	73	80	70	-3
Heartbeat Drum	A♭ Major	130	74	88	77	-4
SAD SONG	B♭ Minor	80	95	62	91	-1
Pretty Boy	F Minor	102	84	86	49	-3
Follow Me	C# Minor	128	80	66	69	-2
Last Call	G Major	130	86	62	86	-2

Tabel 2 Data lagu yang dibawakan P1Harmony pada *Busan One Asia Festival 2025* beserta parameternya

B. Hasil Output Program

Berikut hasil eksekusi program yang telah dibuat.

C. Pembahasan

Bobot transisi antar lagu dihitung berdasarkan perbedaan parameter musikal yang telah dinormalisasi. Hasil perhitungan tersebut direpresentasikan dalam bentuk visualisasi graf berbobot. Semakin besar bobot transisi, semakin besar perbedaan musikal antara dua lagu dan semakin tidak nyaman transisinya bagi pendengar. Transisi antara lagu “Follow me” dan “Last Call” memiliki bobot transisi paling rendah, yaitu

sebesar 0.145. Sementara itu, transisi antara lagu “SAD SONG” dan “Follow Me” memiliki bobot transisi paling tinggi, yaitu sebesar 0.5457.

Menariknya, baik algoritma greedy maupun brute-force menghasilkan urutan optimal yang sama: “DUH!”, “Follow Me”, “Last Call”, “Heartbeat Drum”, “Pretty Boy”, “SAD SONG”, dengan total bobot transisi sebesar 0.9977. Hal ini menunjukkan bahwa untuk kasus ini algoritma greedy berhasil menemukan solusi global optimal meskipun bersifat heuristik dan sruktur graf yang terbentuk memungkinkan pendekatan greedy bekerja dengan sangat efektif. Urutan ini berbeda *setlist* asli dalam konser, yaitu “DUH!”, “Heartbeat Drum”, “SAD SONG”, “Pretty Boy”, “Follow Me”, “Last Call”, yang memiliki total bobot sebesar 1.6683. *Setlist* yang menerapkan algoritma greedy Hamiltonian Path 37.97% lebih optimal dibandingkan *setlist* asli.

Meskipun memberikan hasil yang optimal untuk jumlah lagu kecil, pendekatan ini memiliki beberapa keterbatasan. Algoritma brute-force dengan kompleksitas O(n!) tidak praktis untuk *setlist* dengan jumlah lagu yang banyak. Algoritma greedy ($O(n^2)$) lebih skalabel tetapi tidak selalu menjamin optimalitas. Selain itu, model ini hanya mempertimbangkan aspek musikal teknis tanpa memperhitungkan faktor artistik lain seperti lirik lagu, emosi penonton, tema konser, atau dinamika panggung.

V. KESIMPULAN

Melalui pemodelan graf berbobot dan penerapan algoritma Hamiltonian Path, penelitian ini berhasil mengoptimalkan urutan lagu untuk konser P1Harmony at Busan One Asia Festival 2025. Hasilnya menunjukkan bahwa pendekatan ini 37.97% lebih optimal dibandingkan *setlist* asli. Menariknya baik algoritma greedy maupun brute force menghasilkan solusi identik, membuktikan bahwa dalam kasus khusus ini, pendekatan heuristik sederhana mampu mencapai optimalitas global. Namun demikian, implementasi praktis memerlukan pertimbangan lebih luas. Meskipun penting, aspek teknis seperti keselarasan nada dan tempo hanyalah sebagian dari hal yang mempengaruhi optimalnya urutan suatu *setlist*. Faktor artistik seperti alur emosional, variasi dinamika, dan interaksi dengan penonton juga memegang peranan krusial, tetapi belum sepenuhnya terakomodasi dalam model ini.

REFERENCES

[1] M. Bevec, M. Tkalčič, dan M. Pesek, “Hybrid music recommendation with graph neural networks,” *User Modeling and User-Adapted Interaction*, vol. 34, pp. 1891–1928, 2024. [Online]. Tersedia di: https://link.springer.com/article/10.1007/s11257-024-09410-4 [Diakses: 19 Juni 2025].

[2] Programiz, "Greedy Algorithm," Programiz. [Online]. Tersedia di: https://www.programiz.com/dsa/greedy-algorithm [Diakses: 20 Juni 2025].

[3] R. Munir, *Materi Kuliah Matematika Diskrit*, Program Studi Teknik Informatika STEI ITB. [Online]. Tersedia di: https://informatika.stei.itb.ac.id/~rinaldi.munir/ [Diakses: 18 Juni 2025].

[4] S. Carmesin, D. Woller, D. Parker, M. Kulich, dan M. Mansouri, “The Hamiltonian Cycle and Travelling Salesperson problems with traversal-dependent edge deletion,” *Journal of Computational Science*, vol. 74, p. 102156, 2023. [Online]. Tersedia di: https://www.sciencedirect.com/science/article/pii/S1877750323002168 [Diakses: 19 Juni 2025].

[5] Setlist.fm, "P1Harmony Setlist at Busan Exhibition and Convention Center," Setlist.fm. [Online]. Tersedia di: https://www.setlist.fm/setlist/p1harmony/2025/busan-exhibition-and-convention-center-busan-south-korea-43479763.html [Diakses: 19 Juni 2025].

[6] Tunebat, *Tunebat.com*. [Online]. Tersedia di: https://www.tunebat.com [Diakses: 19 Juni 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, 20 Juni 2025

Safira Berlianti
13524128