Aplikasi Graf Terarah Berbobot Durasi untuk Penjadwalan Latihan Fullbody yang Efisien dengan Algoritma Dijkstra

Faqih Muhammad Syuhada and 13523057[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]faqihmsy@itb.ac.id

***Abstract*— Tidak hanya bagi atlet profesional tetapi juga bagi orang-orang yang ingin menjaga kualitas hidup mereka, latihan teratur dapat secara signifikan meningkatkan kesehatan, kebugaran, dan performa tubuh mereka. Latihan seluruh tubuh, juga dikenal sebagai "latihan seluruh tubuh", adalah salah satu pendekatan yang paling efektif karena dapat melibatkan hampir semua kelompok otot utama dalam satu sesi. Merancang jadwal latihan yang efektif, efisien, dan dapat disesuaikan dengan tujuan latihan adalah tantangan terbesar dalam implementasinya. Berbagai jenis latihan beban dalam satu sesi, seperti latihan beban berat, sedang, dan ringan, memerlukan perhatian khusus dalam hal durasi, intensitas, dan waktu istirahat antara latihan. Ini penting untuk perencanaan latihan karena transisi antara set latihan dan pemulihan otot sangat penting untuk menjaga efisiensi. Metode graf terarah berbobot durasi digunakan dalam penelitian ini untuk membuat jadwal latihan yang ideal. Algoritma Djikstra membantu dalam menemukan jalur terpendek, mengoptimalkan waktu, dan memastikan pemulihan otot yang tepat. Hasil dari penelitian ini diharapkan dapat membantu atlet dan orang-orang dengan waktu terbatas merancang program latihan seluruh tubuh yang efektif. Studi ini akan mempertimbangkan fisiologi olahraga dan memastikan hasil terbaik.**

***Keywords*—Dijkstra's algorithm, Teori Graf, penjadwalan latihan, Efisiensi Waktu, Pemulihan Otot, Periodisasi, Full-body Workout, Circuit Training.**

I. INTRODUCTION

Latihan dapat meningkatkan kesehatan, kebugaran, dan performa fisik seseorang secara signifikan. Ini berlaku tidak hanya bagi atlet profesional tetapi juga bagi individu yang berlatih untuk menjaga kualitas hidup mereka. Karena memungkinkan untuk melatih hampir semua kelompok otot utama dalam satu sesiv(FullBody), latihan seluruh tubuh telah menjadi salah satu jenis latihan kebugaran yang paling efektif. Namun, salah satu tantangan dalam menerapkan program latihan seluruh tubuh adalah mengetahui bagaimana membuat jadwal yang tidak hanya efisien dan efektif tetapi juga fleksibel sehingga dapat disesuaikan dengan tujuan latihan tertentu, yang dapat menyebabkan masalah.

Latihan seluruh tubuh biasanya terdiri dari berbagai jenis latihan. Latihan beban berat dapat digunakan untuk meningkatkan kekuatan dan tenaga, latihan beban sedang dapat digunakan untuk memperbesar otot, dan latihan beban ringan atau berat badan dapat digunakan untuk meningkatkan daya tahan dan pengondisian. Selain itu, latihan ini mencakup berbagai varian, seperti superset atau latihan sirkuit, yang secara berurutan memadukan latihan dengan intensitas yang berbeda-beda. Latihan ini biasanya juga melibatkan variasi. Ada banyak hal yang menarik dalam proses perencanaan karena setiap jenis latihan memiliki durasi, intensitas, dan waktu istirahat yang berbeda.

Penjadwalan latihan menjadi lebih sulit dalam situasi di mana harus mempertimbangkan efisiensi waktu dan waktu yang diperlukan untuk pemulihan otot. Misalnya, latihan beban berat seperti deadlift memerlukan waktu pemulihan yang lebih lama daripada latihan beban ringan seperti push-up. Selain itu, untuk menghindari kelelahan otot yang berlebihan, yang dapat mengurangi efisiensi sesi latihan, transisi antar set sangat penting. Ini terutama berlaku saat beralih di antara latihan dengan intensitas yang berbeda. Dalam situasi seperti ini, metodologi dan teknologi berbasis grafik dapat menawarkan solusi inovatif untuk membuat jadwal latihan yang efektif dan terukur secara ilmiah.

Metode metodis untuk memodelkan hubungan antara latihan berdasarkan intensitas latihan, durasi aktivitas, dan waktu istirahat yang dibutuhkan diberikan oleh grafik terarah berbobot durasi. Dalam kerangka model ini, setiap bentuk latihan digambarkan sebagai titik pada grafik, dan durasi setiap latihan dan jumlah waktu yang berlalu di antara latihan digambarkan sebagai bobot pada tepinya. Akibatnya, grafik ini dapat digunakan untuk menggambarkan berbagai kombinasi latihan yang mungkin, yang akan memberikan gambaran yang jelas tentang cara paling efektif untuk mencapai tujuan pelatihan tertentu.

Dalam studi ini, algoritma Dijkstra, yang terkenal di bidang pemrograman dan matematika diskrit, digunakan sebagai bagian penting dari metodologi ini untuk menemukan jalur paling efisien pada grafik terarah berbobot durasi. Algoritma ini memungkinkan untuk menemukan kombinasi latihan dalam penjadwalan latihan yang memperhitungkan tujuan dan kebutuhan relaksasi sambil mengurangi jumlah waktu yang dihabiskan untuk latihan. Dengan kata lain, model ini menawarkan solusi praktis untuk masalah utama perencanaan latihan seluruh tubuh.

Tujuan penelitian ini adalah untuk membantu orang atau pelatih membuat rejimen latihan seluruh tubuh yang ideal dengan membuat aplikasi berbasis grafik terarah berbobot durasi. Aplikasi ini fleksibel sehingga cocok untuk atlet yang membutuhkan rejimen latihan intensif dan orang-orang dengan

waktu terbatas tetapi ingin mendapatkan hasil terbaik. Selain itu, tujuan dari penelitian ini adalah untuk menunjukkan bagaimana algoritma Djikstra dapat digunakan dalam situasi yang tidak konvensional, misalnya dalam mengatur latihan yang membutuhkan fisik. Hasil yang diharapkan dari proyek penelitian ini tidak hanya akan memungkinkan optimalisasi manajemen waktu, tetapi juga akan memastikan bahwa jadwal akhir akan mengikuti prinsip-prinsip fisiologi olahraga, seperti mengendalikan kelelahan dan pemulihan otot.

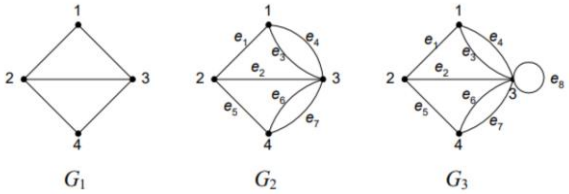
Diharapkan bahwa penelitian ini akan memberikan kontribusi besar untuk kemajuan teknologi di bidang kebugaran dengan menggunakan metode ini. Selain itu, akan membuka ruang untuk penelitian lebih lanjut tentang bagaimana model grafik dapat digunakan untuk aplikasi sehari-hari.

II. BASIC THEORY

A. Graf

Graf dapat didefinisikan sebagai suatu struktur diskrit yang terdiri dari kumpulan simpul (vertex) yang saling terhubung melalui asosiasi sisi (edge) dengan hubungan tertentu. Representasi graf ini dapat dituliskan dalam bentuk $G = (V, E)$, di mana G adalah graf, V adalah himpunan simpul $v1$, $v2$, … , vn yang tidak kosong, dan E adalah kumpulan sisi $e1$, $e2$, … , en yang menghubungkan sepasang simpul pada graf.

Dalam konteks graf, simpul dapat dihubungkan oleh dua sisi yang berbeda, yang dikenal sebagai multiple edge. Selain itu, terdapat sisi yang berawal dan berakhir pada simpul yang sama, yang disebut ring atau lingkaran (circles). Berdasarkan keberadaan sisi ganda atau sisi lengkung, graf dapat dibedakan menjadi graf sederhana tetapi tidak memiliki sisi ganda dan sisi lengkung, dan graf tidak sederhana yang memiliki sisi ganda atau lengkung, dan apabila graf tidak sederhana memiliki lengkung maka disebut graf semu.

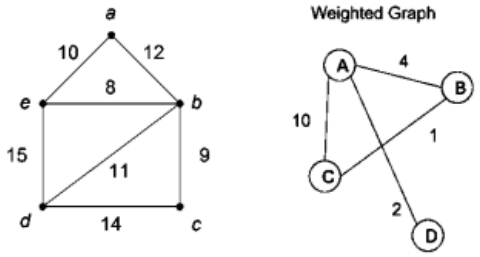


Gambar 2.1 (a) Graf sederhana (b) Graf tak-sederhana (c) Graf-semu (Sumber : [9])

Orientasi arah sisi juga menjadi faktor penting dalam mengelompokkan graf. Graf tak berarah memiliki sisinya tanpa orientasi arah, sementara graf berarah memiliki orientasi arah pada setiap sisinya

Beberapa terminologi dalam teori graf meliputi:

1. Ketetanggan (adjacency): Dua simpul dalam graf dikatakan bertetangga jika terhubung oleh setidaknya satu sisi.
2. Bersisian (incidency): Sebuah sisi dikatakan bersisian dengan dua simpul tertentu jika menghubungkannya.
3. Derajat (degree): Jumlah sisi yang bersisian dengan suatu simpul.
4. Lintasan (path): Barisan simpul dan sisi yang menghubungkannya dari simpul awal ke simpul tujuan.
5. Siklus (cycle atau circuit): Lintasan yang berawal dan berakhir pada simpul yang sama.
6. Keterhubungan (connected): Dua simpul dikatakan terhubung jika terdapat lintasan yang menghubungkannya.
7. Graf berbobot (weighted graph): Graf yang setiap sisinya memiliki nilai atau bobot spesifik.

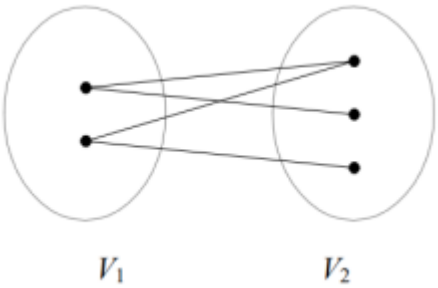


Gambar 2.2. Contoh orientasi graf berbobot (Sumber: [9])

8. Graf lengkap (complete graph): Graf yang setiap simpulnya terhubung dengan semua simpul lainnya dalam graf tersebut.

Beberapa jenis graf khusus, antara lain:

1. Graf Lengkap (Complete Graph)
   Graf lengkap adalah jenis graf sederhana di mana setiap simpulnya terhubung dengan semua simpul lainnya, sehingga jumlah sisi pada graf dengan n simpul selalu $n(n - 1)/2$. Graf lengkap yang memiliki n simpul dinotasikan sebagai Kn.
2. Graf Lingkaran
   Graf lingkaran adalah graf sederhana di mana setiap simpulnya memiliki derajat dua.
3. Graf Teratur (Regular Graph)
   Graf teratur adalah jenis graf yang setiap simpul memiliki derajat yang sama, sehingga jumlah sisi pada graf teratur adalah $nr/2$, di mana n adalah jumlah simpul dan r adalah derajat setiap simpul. Graf teratur berderajat r adalah graf di mana semua simpul memiliki derajat r.
4. Graf Bipartite (Bipartite Graph)
   Graf bipartite adalah graf G yang himpunan simpulnya dapat dibagi menjadi dua bagian, $V1$ dan $V2$, sehingga setiap sisi dalam G menghubungkan simpul dari $V1$ ke simpul dari $V2$. Graf bipartite direpresentasikan sebagai $G(V1, V2)$.



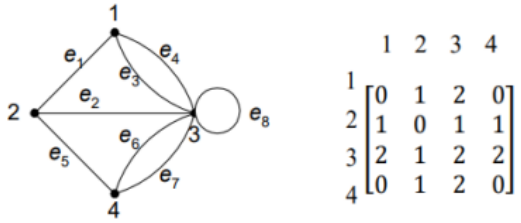
Gambar 2.3. Orientasi graf bipartite (Sumber: [9])

Beberapa representasi graf adalah sebagai berikut:

1. Matriks Ketetanggaan (Adjacency Matrix)
   Sebuah graf tak berarah dengan n simpul dapat diwakili oleh matriks ketetanggaan M $n \times n$. Elemen pada baris ke-i dan kolom ke-j bernilai 1 jika

terdapat sisi yang menghubungkan simpul ke-i dan simpul ke-j, dan 0 jika tidak. Pada graf berarah, elemen pada baris ke-i dan kolom ke-j akan bernilai 1 jika ada sisi yang mengarah dari simpul ke-i ke simpul ke-j. Jika graf memiliki bobot pada sisinya, elemen pada posisi yang sama menggambarkan bobot sisi yang menghubungkan simpul ke-i dan simpul ke-j.

2. Matriks Bersisian (Incidency Matrix)
   Graf tak berarah dengan n simpul dan m sisi dapat direpresentasikan oleh matriks bersisian M $n \times m$. Elemen pada baris ke-i dan kolom ke-j bernilai 1 jika simpul ke-i bersisian dengan sisi ke-j, dan 0 jika tidak.



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 0 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 2 & 2 \\ 0 & 1 & 2 & 0 \end{bmatrix} \end{array}$$

Gambar 2.4. Orientasi graf bipartite (Sumber : [9])

Graf terarah berbobot sering digunakan untuk memecahkan masalah optimasi, seperti penjadwalan, jalur transportasi, atau alokasi sumber daya. Dalam penjadwalan latihan, graf ini dapat digunakan untuk merancang rute latihan yang efisien, di mana bobot sisi menggambarkan total waktu yang diperlukan untuk transisi dari satu latihan ke latihan lainnya [4]. Selain itu, untuk memastikan seluruh latihan dikunjungi maka dapat menggunakan sirkuit Hamilton, ialah lintasan yang melalui tiap simpul didalam graf tepat sekali, kecuali simpul asal yang dilalui dua kali. Graf yang memiliki sirkuit Hamilton dinamakan graf Hamilton [9].

B. Shortest path

Shortest path merupakan salah satu permasalahan umum yang ada pada graf. Permasalahan shortest path dapat didefinisikan sebagai pencarisan lintasan terpendek diantara dua simpul tertentu pada sebuah graf. Jika graf merupakan graf berbobot, maka lintasan yang dicari haruslah lintasan yang jumlah bobotnya terkecil. Shortest path sendiri dapat dibedakan menajadi dua jenis, yaitu:

1. Single-source Shortest Path Yaitu pencarian lintasan terpendek dari sebuah simpul ke seluruh simpul yang ada pada graf.
2. Single-destination Shortest Path Yaitu pencarian lintasan terpendek dari seluruh simpul yang ada pada graf ke sebuah simpul tertentu.
3. All-pairs Shortest Path Yaitu pencarian lintasan terpendek dari setiap pasang simpul pada sebuah graf.

Salah satu cara untuk mencari shortest path, adalah dengan menggunakan Algoritma Dijkstra, Algoritma ini sering digunakan dalam permasalahan kehidupan sehari-hari seperti alat GPS untuk mencari rute terpendek dari suatu lokasi ke lokasi lain. Untuk algoritmanya sendiri, terdapat beberapa langkah yang dapat dilakukan sebagai berikut:

1. Pilih sebuah simpul source pada graf G yang akan dijadikan simpul sumber dan tetapkan $dist$= 0. Untuk simpul selain Source, tetapkan nilai $dist = \infty$. Masukkan seluruh simpul pada sebuah himpunan Q yang menunjukkan simpul-simpul yang masih belum ditentukan nilai shortest pathnya.
2. Pilih sebuah simpul v pada Q dengan nilai dist terkecil. Hapus v dari dari Q.
3. Cek setiap simpul u yang merupakan tetangga dari v yang belum ada pada Q. Nilai $dist_u$ diperbaharui apabila $dist_u < dist_v + W_{vu}$, dengan W_{vu} adalah besar bobot dari sisi yang menghubungkan v dan u.
4. Ulangi langkah 2 hingga 4 hingga himpunan Q kosong.

.

C. Teori Latihan Fisik dan Penjadwalan Latihan

Latihan fisik dibagi menjadi beberapa kategori berdasarkan tujuan dan intensitas, seperti latihan kekuatan (strength), hipertrofi (hypertrophy), dan ketahanan (endurance) [1]. Penjadwalan latihan yang efektif melibatkan pertimbangan durasi setiap latihan, waktu istirahat antar set, dan urutan latihan untuk mengoptimalkan hasil. Dalam studi fisiologi olahraga, perencanaan ini dikenal dengan istilah periodization, yang bertujuan untuk mengatur intensitas dan volume latihan sehingga tubuh dapat pulih dengan optimal sambil tetap mencapai target performa [2].

Latihan full-body merupakan metode yang melatih semua kelompok otot utama dalam satu sesi. Keuntungan dari metode ini adalah efisiensi waktu, tetapi tantangannya adalah mengelola transisi antar latihan yang berbeda intensitasnya. Studi menunjukkan bahwa waktu istirahat yang tepat di antara set sangat penting untuk pemulihan otot dan kinerja optimal [3]. Ada banyak cara melakukan Latihan full body, diantara ada Latihan circuit(circuit training).

Circuit training adalah teknik latihan yang melibatkan serangkaian gerakan fisik yang dilakukan secara bergantian tanpa istirahat yang lama. Latihan ini biasanya dimaksudkan untuk melatih berbagai kelompok otot secara bergantian dalam satu sesi latihan, sehingga memberikan manfaat lengkap bagi kebugaran fisik. Teknik ini juga dikenal sangat efektif dalam meningkatkan daya tahan tubuh dan mengoptimalkan waktu Latihan [4]. Banyak Manfaat Pelatihan Circuit, diantaranya ada :

Efisiensi Waktu, Latihan circuit melibatkan banyak kelompok otot dalam satu sesi, yang memungkinkan latihan lebih singkat dibandingkan dengan teknik tradisional tanpa mengurangi efektivitasnya [7].

Peningkatan Daya Tahan Otot, Latihan ini menggunakan gerakan yang kuat dengan sedikit istirahat, meningkatkan daya tahan otot dan kemampuan kardiovaskular [8].

Variasi Latihan, Latihan sirkuit menawarkan fleksibilitas dalam memilih jenis gerakan, yang mengurangi kemungkinan kebosanan selama Latihan [5].

Latihan circuit sendiri juga dapat terbagi sesuai tujuannya diantaranya yaitu, Latihan Kekuatan Otot Tangan termasuk push-up, angkat beban untuk biceps, atau triceps dip [4]. Latihan Daya Tahan Otot Kaki termasuk squat, lunges, atau box jump [4]. Latihan Hipertrofi Otot Perut termasuk sit-up, plank, atau angkat kaki tergantung [7].

Kekuatan, daya tahan, dan kebugaran secara

keseluruhan dapat ditingkatkan dengan cara ini. Studi menunjukkan bahwa latihan circuit dapat meningkatkan kemampuan fisik dalam olahraga tertentu, seperti memanah, di mana akurasi dan daya tahan otot sangat penting [4]. Selain itu, kombinasi latihan circuit dapat disesuaikan dengan kebutuhan seseorang, seperti berkonsentrasi pada meningkatkan atau mengurangi berat badan.

III. PEMBAHASAN

Pada bagian ini, metode yang digunakan untuk menyelesaikan masalah untuk mengoptimalkan durasi latihan dengan mempertimbangkan waktu istirahat antara latihan dibahas. Beberapa algoritma, seperti graf berbobot, algoritma Djikstra, dan pencarian jalur Hamiltonia, digunakan untuk mencapai solusi yang optimal.

A. Representasi Graf Berbobot

Replikasi grafik berbobot yang menghubungkan berbagai latihan merupakan komponen utama metode ini. Dalam skenario ini, setiap jenis latihan digambarkan sebagai simpul (node) pada graf. Setiap node berisi informasi tentang nama latihan, intensitasnya (misalnya, berat atau ringan), dan jumlah waktu yang diperlukan untuk menyelesaikannya.

```
exercises = {
    "Deadlift": ("heavy", 60),
    "Squat": ("heavy", 60),
    "Bench Press": ("heavy", 50),
    "Overhead Press": ("medium", 50),
    "Pull-Up": ("light", 30),
    "Push-Up": ("light", 30),
    "Barbell Row": ("medium", 45),
    "Dumbbell Bench Press": ("medium", 45),
    "Plank": ("light", 90),
    "Renegade Row": ("medium", 60),
    "Kettlebell Swing": ("medium", 60)
}
```

Gambar 3.1. Node atau simpul pada graf (Sumber : Lampiran)

Graf tersebut juga memiliki sisi, atau tepi, yang menghubungkan satu latihan ke latihan lain. Bobot ini menunjukkan jumlah waktu yang diperlukan untuk beralih dari satu latihan ke latihan berikutnya. Waktu istirahat antara dua latihan berbeda. Latihan intensitas tinggi membutuhkan waktu istirahat yang lebih lama daripada kombinasi latihan intensitas rendah.

```
rest_times = {
    ("heavy", "heavy"): 300,
    ("heavy", "medium"): 240,
    ("medium", "heavy"): 240,
    ("heavy", "light"): 180,
    ("light", "heavy"): 180,
    ("medium", "medium"): 180,
    ("medium", "light"): 120,
    ("light", "medium"): 120,
    ("light", "light"): 60
}
```

Gambar 3.2. Edge atau sisi pada graf (Sumber : Lampiran)

B. Pembentukan Graf Lengkap

Semua latihan saat ini dapat dihubungkan dengan menggunakan fungsi create_complete_graph(). Dalam graf ini, setiap pasangan latihan dihubungkan oleh sisi yang memiliki bobot yang ditentukan berdasarkan waktu istirahat yang diperlukan dan durasi latihan berikutnya.

```
def create_complete_graph(exercises, rest_times):
    graph = defaultdict(dict)
    for ex1 in exercises:
        for ex2 in exercises:
            if ex1 != ex2:
                intensity1 = exercises[ex1][0]
                intensity2 = exercises[ex2][0]
                try:
                    rest_time = rest_times[(intensity1, intensity2)]
                except KeyError:
                    rest_time = rest_times[(intensity2, intensity1)]
                graph[ex1][ex2] = rest_time + exercises[ex2][1]
    return graph
```

Gambar 3.1. Algoritma membangun graf (Sumber : Lampiran)

Setiap sisi dari graf lengkap berisi informasi tentang waktu istirahat antara dua latihan dan durasi latihan itu sendiri, sehingga setiap jalur yang melewati sisi tersebut dapat dihitung dengan tepat. Ini memungkinkan pencarian jalur terbaik antara latihan tanpa terbatas atau bergantung pada urutan latihan yang dilakukan.

C. Pencarian Jalur Terpendek Menggunakan Algoritma Dijkstra

Untuk menemukan waktu terpendek yang diperlukan untuk berpindah dari satu latihan ke latihan lainnya, algoritma pencarian jalur terpendek (Djikstra) digunakan untuk mengoptimalkan urutan latihan yang dilakukan. Algoritma ini bekerja dengan mengunjungi setiap titik dalam graf dan memperbarui jarak terpendek yang ditemukan menuju titik tetangga.

```
def dijkstra(graph, start, end):
    """Fungsi untuk menemukan jalur terpendek menggunakan algoritma Dijkstra"""
    pq = []  # Priority queue
    heapq.heappush(pq, (0, start))  # (total_cost, current_node)
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    visited = set()

    while pq:
        current_distance, current_node = heapq.heappop(pq)

        if current_node in visited:
            continue
        visited.add(current_node)

        if current_node == end:
            return current_distance

        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(pq, (distance, neighbor))

    return float('inf')  # Tidak ada jalur
```

Gambar 3.3. Algoritma Dijkstra yang diterapkan (Sumber : Lampiran)

Algoritma Dijkstra digunakan dalam fungsi Dijkstra() dalam kode ini. Fungsi ini memulai pencarian dari simpul latihan yang ditentukan dan kemudian mengunjungi semua simpul yang terkait sambil memperbarui jarak terpendek. Hasilnya adalah waktu terpendek untuk beralih antara dua latihan.

D. Pencarian Jalur Hamiltonian dengan Durasi Minimum

Dalam hal ini, masalah utama yang ingin diselesaikan adalah menemukan urutan latihan yang ideal yang memperhitungkan durasi total latihan, serta waktu istirahat yang diperlukan di antara latihan. Tujuan dari Hamiltonian Circuit adalah untuk menemukan jalur yang mengunjungi setiap simpul sekali dan hanya sekali, kemudian kembali ke simpul awal.

```python
def find_shortest_hamilton_circuit(exercises, rest_times):
    graph = create_complete_graph(exercises, rest_times)
    exercise_names = list(exercises.keys())
    min_duration = float('inf')
    best_path = None

    for path in permutations(exercise_names):
        duration = calculate_circuit_duration(path, graph, exercises)
        if duration < min_duration:
            min_duration = duration
            best_path = path

    return best_path, min_duration, graph
```

Gambar 3.4. Algortima sirkuit Hamilton (Sumber : Lampiran)

Untuk menyelesaikan masalah ini, teknik brute force digunakan untuk menghitung durasi total untuk setiap urutan latihan (permutasi) yang mungkin. Fungsi find_shortest_hamilton_circuit() mencoba semua urutan latihan, menghitung durasi total, dan kemudian memilih urutan dengan durasi terpendek. Fungsi ini menghitung durasi total, yang mencakup waktu latihan dan istirahat.

E. Penyusunan Durasi dan Pembagian Waktu

Setelah menemukan urutan latihan yang ideal, dibuat jadwal untuk latihan dan istirahat. Untuk menghitung total waktu yang diperlukan untuk menyelesaikan seluruh urutan latihan, fungsi calculate_circuit_duration() digunakan. Pembagian waktu ini mencakup waktu latihan untuk setiap latihan, serta waktu yang diperlukan untuk transisi antar latihan.

```python
def calculate_circuit_duration(path, graph, exercises):
    total_duration = exercises[path[0]][1]
    for i in range(len(path)-1):
        current = path[i]
        next_ex = path[i+1]
        total_duration += graph[current][next_ex]
    return total_duration
```

Gambar 3.5. Algoritma menghitung durasi sirkuit (Sumber : Lampiran)

Selain itu, sistem ini mengoptimalkan waktu tidur dengan mencari latihan tambahan yang dapat dilakukan saat tidur. Untuk memastikan bahwa sisa waktu istirahat tidak terbuang sia-sia, fungsi find_exercises_for_rest() mencari latihan yang dapat dimasukkan ke dalam waktu istirahat.

F. Implementasi Penggunaan Dijkstra untuk Jalur Terpendek

Kode ini menunjukkan penggunaan algoritma Dijkstra untuk mencari jalur terpendek antara dua latihan tertentu, misalnya, mencari jalur terpendek dari latihan Deadlift ke latihan Pull-Up menggunakan permutasi. Ini meningkatkan fleksibilitas analisis jalur antar latihan yang relevan.

```python
shortest_path = dijkstra(graph, "Deadlift", "Pull-Up")
print(f"\nJarak terpendek dari Deadlift ke Pull-Up menggunakan Dijkstra: {shortest_path} detik")
```

Gambar 3.6. Contoh penerapan dijkstra (Sumber : Lampiran)

G. Evaluasi Hasil

Program ini menghasilkan hasil setelah menemukan urutan latihan yang ideal, yang termasuk:

Urutan latihan menunjukkan jumlah waktu yang dibutuhkan untuk melakukan latihan dan waktu istirahat yang tepat untuk setiap transisi antara latihan.

```
Detail urutan dengan waktu istirahat:

Menit 0.0 - 1.0: Deadlift (heavy, 60s)
Istirahat: 300 detik (Transisi Deadlift dengan Pull-Up 30s -> Push-Up 30s -> Squat)

Menit 6.0 - 7.0: Squat (heavy, 60s)
Istirahat: 240 detik (Transisi Squat -> Overhead Press)

Menit 11.0 - 11.8: Overhead Press (medium, 50s)
Istirahat: 120 detik (Transisi Overhead Press -> Pull-Up)

Menit 13.8 - 14.3: Pull-Up (light, 30s)
Istirahat: 60 detik (Transisi Pull-Up -> Push-Up)

Menit 15.3 - 15.8: Push-Up (light, 30s)
Istirahat: 120 detik (Transisi Push-Up -> Barbell Row)

Menit 17.8 - 18.6: Barbell Row (medium, 45s)
Istirahat: 180 detik (Transisi Barbell Row -> Dumbbell Bench Press)

Menit 21.6 - 22.3: Dumbbell Bench Press (medium, 45s)
Istirahat: 120 detik (Transisi Dumbbell Bench Press -> Plank)

Menit 24.3 - 25.8: Plank (light, 90s)
Istirahat: 120 detik (Transisi Plank -> Renegade Row)

Menit 27.8 - 28.8: Renegade Row (medium, 60s)
Istirahat: 180 detik (Transisi Renegade Row -> Kettlebell Swing)

Menit 31.8 - 32.8: Kettlebell Swing (medium, 60s)
Istirahat: 240 detik (Transisi Kettlebell Swing -> Bench Press)

Menit 36.8 - 37.7: Bench Press (heavy, 50s)

Total durasi: 2260 detik (37.67 menit)
```

Gambar 4.1. Urutan latihan dan total durasi (Sumber : Lampiran)

Durasi Total adalah jumlah total waktu yang dibutuhkan untuk menyelesaikan urutan latihan, termasuk istirahat.

```
Breakdown waktu:
Total waktu exercise: 580 detik (9.67 menit)
Total waktu istirahat: 1680 detik (28.00 menit)
```

Gambar 4.2. Total waktu istirahan dan olahraga dalam satu set(Sumber : Lampiran)

Breakdown Waktu adalah pembagian waktu antara durasi latihan dan istirahat dalam satu set latihan, yang memberikan gambaran tentang bagian waktu yang terlibat dalam latihan secara keseluruhan sesuai kainginan pengguna.

IV. KESIMPULAN

Penelitian ini menemukan solusi yang optimal untuk optimalisasi durasi latihan dengan mempertimbangkan waktu istirahat antara latihan. Metode yang digunakan termasuk penggunaan graf berbobot dan algoritma Djikstra untuk menemukan jalur terpendek, serta pendekatan permutasi untuk menyelesaikan masalah jalur Hamiltonian. Dengan menggunakan sistem ini, Anda dapat membuat jadwal latihan yang efektif dan membantu atlet mencapai hasil terbaik dalam waktu yang singkat.

Sangat mungkin untuk menggunakan jadwal latihan ini untuk berbagai jenis pelatihan olahraga, baik untuk individu maupun kelompok, dengan mempertahankan keseimbangan antara latihan intensif dan pemulihan yang ideal.

V. LAMPIRAN

Program lengkap dari makalah ini dapat ditemukan di. https://github.com/FaqihMSY/Aplikasi-Graf-Terarah-Berbobot-Durasi-untuk-Penjadwalan-Latihan-Fullbody-yang-Efisien.git.

REFERENCES

[1] G. Haff and N. T. Triplett, *Essentials of Strength Training and Conditioning*, 4th ed. Champaign, IL: Human Kinetics, 2016. [Online]. Available:

https://www.ncbi.nlm.nih.gov/nlmcatalog?cmd=PureSearch&term=101647597%5Bnlmid%5D. [Accessed 5 January 2024].
[2] T. O. Bompa and C. Buzzichelli, *Periodization: Theory and Methodology of Training*. Champaign, IL: Human Kinetics, 2019. [Online]. Available: https://www.researchgate.net/publication/377319619_Periodization_Theory_and_Methodology_of_Training. [Accessed 5 January 2024].
[3] G. G. Haff and N. T. Triplett, *Essentials of Strength Training and Conditioning*, 4th ed. Champaign, IL: Human Kinetics, 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/nlmcatalog?cmd=PureSearch&term=101647597%5Bnlmid%5D. [Accessed 5 January 2024].
[4] Susanto, Siswantoyo, Prasetyo, Y., & Putranta, H. (2021). The Effect of Circuit Training on Physical Fitness and Archery Accuracy in Novice Athletes. *Physical Activity Review*, 9(1), 100-108. [Online]. Available: https://www.researchgate.net/publication/349408625_The_effect_of_circuit_training_on_physical_fitness_and_archery_accuracy_in_novice_athletes. [Accessed 5 January 2024].
[5] García-Pinillos, F., Martínez-Amat, A., Hita-Contreras, F., & Martínez-López, E. J. (2017). Effects of High-Intensity Circuit Training on Body Composition, Physical Fitness, and Quality of Life in Middle-Aged Adults. *Journal of Aging and Physical Activity*, 25(4), 731-740. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8145598. [Accessed 5 January 2024].
[6] Rahman, N. A., Wahid, A., & Rashid, A. (2020). The Effect of Circuit Training on Physical Fitness and Archery Accuracy in Novice Athletes. Journal of Physical Activity and Health, 17(1), 45-51. [Online]. Available: https://www.researchgate.net/publication/368822679_METODE_LATIHAN_CIRCUIT_TRAINING_UNTUK_MENINGKATKAN_DAYA_TAHAN_OTOT_LENGAN_AKURASI_MEMANAH. [Accessed 5 January 2024].
[7] Stanforth, D., Stanforth, P. R., & Hoemeke, M. P. (1998). Physiologic and Metabolic Responses to a Body-Composition Circuit. Journal of Strength and Conditioning Research, 12(3), 163-169. [Online]. Available: https://www.researchgate.net/publication/232107546_Physiologic_and_Metabolic_Responses_to_a_Body_Pump_Workout. [Accessed 5 January 2024].
[8] Gettman, L. R., & Pollock, M. L. (1981). Circuit Weight Training: A Critical Review of Its Physiological Benefits. The Physician and Sportsmedicine, 9(1), 44-60. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/27462744/. [Accessed 5 January 2024].
[9] R. Munir, "https://informatika.stei.itb.ac.id/," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/. [Accessed 8 January 2024].

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, 8 Januari 2025

Faqih Muhammad Syuhada (13523057)