Route Optimization for aespa's Concert Tour: A Comparative Study of Greedy and Branch and Bound Algorithms

Amanda Aurellia Salsabilla - 13524131
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: amanda.utoro@gmail.com , 13524131@std.stei.itb.ac.id

***Abstract*—International K-Pop concert tours have grown into complex logistical operations, where routing decisions can significantly affect both operational costs and artist well-being. This study examines aespa's "SYNK : aeXIS LINE" and upcoming "SYNK : COMPLæXITY" tour as a real-world case for the Traveling Salesperson Problem, comparing three routing scenarios: the actual Hub-and-Spoke model, the Greedy nearest-neighbor algorithm, and Branch and Bound. Tour cities are represented as a weighted graph of estimated flight distances across two continent legs: Asia and North America. For the Asian leg, this study finds that the optimal route saves approximately 2,320 km over the baseline, a modest gap that helps explain why returning to Seoul between concerts remains a reasonable scheduling choice. The North American leg, however, shows a significant contrast: the Hub-and-Spoke model accumulates over 70,000 km while Branch and Bound reduce this to just 19,470 km, saving more than 50,000 km. This paper also demonstrates that while Greedy produces near-optimal results with faster computation, Branch and Bound consistently finds the exact shortest path, and recommends a mathematically optimized city sequence for the upcoming North American leg.**

Keywords—Traveling Salesperson Problem, Concert Logistics, Greedy Algorithm, Branch and Bound, Route Optimization

I. Introduction

International K-Pop concert tours have grown into genuinely large-scale operations, and aespa's "SYNK : aeXIS LINE" tour offers an interesting case study in how routing decisions play out differently across continents. The Asian leg covered cities like Jakarta, Osaka, Tokyo, and Macau, with concerts held mostly on weekends. Between stops, the group would return to Seoul, as evidenced by fan-documented airport sightings on March 9, just one day after their Macau concert on March 7 to 8 [1]. For the North American leg, however, the scheduling logic shifted: concert dates were packed more closely together so the group could complete the entire continent without flying back to Korea in between.

Returning to Seoul between concerts makes sense from an artist welfare standpoint, but it introduces an interesting routing inefficiency. Always flying back to a central hub after each stop is known as a Hub-and-Spoke model, and it tends to inflate total travel distance considerably [2]. Therefore, it is necessary to calculate the added distance and determine if a more optimal sequence exists. Efficient route planning has been shown to meaningfully reduce both operational costs and travel time [3][4], and in computer science, this type of problem is formally captured by the Traveling Salesperson Problem (TSP), where the goal is to find the shortest route that visits every location exactly once before returning to the starting point.

This study attempts to quantify that inefficiency by modeling the tour cities as a weighted graph and comparing the total flight distance of the real-world schedule against routes generated by two algorithms: Greedy and Branch and Bound. The goal is not just to run the algorithms, but to evaluate whether the mathematical optimum actually reflects practical touring constraints, and where it does not, to understand why.

II. Theoretical Framework

A. The Traveling Salesperson Problem (TSP)

The Traveling Salesperson Problem (TSP) addresses the problem of finding the shortest possible route given a set of cities and the distances between them, what is the shortest possible route that visits every city exactly once and returns to the starting point [5]? The problem is typically represented as a complete weighted graph, where cities are vertices and distances are the weighted edges. With n cities, there are $(n-1)!$ possible tours to evaluate [7], which means the search space grows factorially. For even a modest number of cities, brute-force enumeration becomes computationally infeasible, which is precisely why algorithmic approaches are necessary.

B. Greedy Algorithm

The Greedy algorithm approaches optimization problems by making the best available local choice at each step, without looking ahead to consider how that decision might affect future options [5]. Applied to TSP, this translates to always moving to the nearest unvisited city from the current position. The appeal of this approach is its simplicity and speed. The

drawback, however, is that locally optimal choices do not always chain together into a globally optimal solution [5]. In practice, the Greedy algorithm frequently produces sub-optimal tours because a short edge chosen early can force a much longer edge later in the route.

C. Branch and Bound Algorithm

Branch and Bound solve optimization problems by constructing a state space tree dynamically, where each node represents a partial solution [6]. Rather than expanding nodes in the order they were generated, the algorithm assigns a cost estimate to each node and always prioritizes the one with the lowest cost, a strategy sometimes called least-cost search [6]. In the context of TSP, this cost functions as a lower bound on the total distance from that partial route to a complete tour [7]. When a node's cost already exceeds the best complete solution found so far, that entire branch is pruned and no longer explored [8]. This bounding mechanism is what separates Branch and Bound from plain exhaustive search: it guarantees finding the exact optimal solution while avoiding the need to evaluate every possible permutation.

D. Hub-and-Spoke Model

The Hub-and-Spoke model is a routing strategy commonly used in transportation and logistics, where all trips originate from and return to a central hub before proceeding to the next destination [2]. Rather than traveling directly between stops in sequence, every leg of the journey passes through the hub. This structure simplifies scheduling and allows the hub to serve as a central coordination point, but it comes at the cost of increased total travel distance, particularly when the destinations are geographically spread out. In the context of this study, Seoul functions as the hub, and each concert city represents a spoke.

III. Implementation and Results

A. Data Modeling

The tour routing problem is modeled as a complete weighted graph, where each city is represented as a vertex and the estimated direct flight distance between any two cities serves as the edge weight. Flight distances are estimated using the great-circle (Haversine) formula based on the geographic coordinates of each city, and rounded to the nearest ten kilometers. The graph is divided into two separate legs drawn from the "SYNK : aeXIS LINE" and upcoming "SYNK : COMPLæXITY" tour schedules:

1. Asia Leg: Seoul (hub), Jakarta, Osaka, Tokyo.
2. North America Leg: Seoul (hub), Los Angeles, Oakland, Seattle, Vancouver.

Seoul is included in both graphs as the designated hub, reflecting its role as the group's home base and the origin point for all routing scenarios.

B. Implementation Code

```
import sys

# Data Asia Leg
cities_asia = ["Seoul", "Jakarta", "Osaka", "Tokyo"]
# Matriks Jarak Asia (Estimasi km udara, great-circle)
# 0:Seoul, 1:Jakarta, 2:Osaka, 3:Tokyo
dist_asia = [
    [0, 5300, 830, 1160],
    [5300, 0, 5400, 5800],
    [830, 5400, 0, 400],
    [1160, 5800, 400, 0]
]

# Data NA Leg
cities_na = ["Seoul", "Los Angeles", "Oakland", "Seattle", "Vancouver"]
# Matriks Jarak NA (Estimasi km udara, great-circle)
#  0:Seoul,  1:LA,  2:Oakland,  3:Seattle, 4:Vancouver
dist_na = [
    [0, 9600, 9000, 8300, 8200],
    [9600, 0, 540, 1540, 1740],
    [9000, 540, 0, 1080, 1280],
    [8300, 1540, 1080, 0, 190],
    [8200, 1740, 1280, 190, 0]
]

# Algoritma
def baseline_hub_spoke(dist_matrix, cities, hub_idx=0):
    # Hub-and-Spoke: Seoul -> Kota -> Seoul (untuk setiap kota)
    total_distance = 0
    route = [cities[hub_idx]]
    for i in range(len(cities)):
        if i != hub_idx:
            jarak_pp = dist_matrix[hub_idx][i] * 2
            total_distance += jarak_pp
            route.extend([cities[i], cities[hub_idx]])
    return route, total_distance

def greedy_tsp(dist_matrix, cities, start_idx=0):
    # Nearest Neighbor: selalu ke kota terdekat yang belum dikunjungi
    N = len(cities)
    visited = [False] * N
    visited[start_idx] = True
    route = [start_idx]
    total_distance = 0
    current = start_idx

    for _ in range(N - 1):
        next_city = -1
        min_dist = sys.maxsize
        for i in range(N):
            if not visited[i] and dist_matrix[current][i] < min_dist:
                min_dist = dist_matrix[current][i]
                next_city = i
        visited[next_city] = True
        route.append(next_city)
        total_distance += min_dist
        current = next_city

    total_distance += dist_matrix[current][start_idx]
    route.append(start_idx)
    route_names = [cities[i] for i in route]
    return route_names, total_distance
```

```
def branch_and_bound_tsp(dist_matrix, cities, start_idx=0):
    # DFS dengan upper bound pruning
    N = len(cities)
    min_cost = sys.maxsize
    best_route = []

    def bnb_util(current_city, visited, current_cost, path):
        nonlocal min_cost, best_route
        if current_cost >= min_cost:
            return
        if len(path) == N:
            total_cost = current_cost + dist_matrix[current_city][start_idx]
            if total_cost < min_cost:
                min_cost = total_cost
                best_route = path + [start_idx]
            return
        for i in range(N):
            if not visited[i]:
                visited[i] = True
                bnb_util(i, visited, current_cost + dist_matrix[current_city][i], path + [i])
                visited[i] = False

    visited = [False] * N
    visited[start_idx] = True
    bnb_util(start_idx, visited, 0, [start_idx])
    route_names = [cities[i] for i in best_route]
    return route_names, min_cost

# UI
def print_separator():
    print("-" * 55)

def print_result(label, route, distance):
    print(f"  Algoritma : {label}")
    print(f"  Jarak     : {distance} km")
    print(f"  Rute      : {' -> '.join(route)}")
    print()

def run_comparison():

    r_hub_asia, d_hub_asia = baseline_hub_spoke(dist_asia, cities_asia)
    r_gr_asia, d_gr_asia = greedy_tsp(dist_asia, cities_asia)
    r_bb_asia, d_bb_asia = branch_and_bound_tsp(dist_asia, cities_asia)

    r_hub_na, d_hub_na = baseline_hub_spoke(dist_na, cities_na)
    r_gr_na, d_gr_na = greedy_tsp(dist_na, cities_na)
    r_bb_na, d_bb_na = branch_and_bound_tsp(dist_na, cities_na)

    print("  [Asia Leg]")
    print(f"  Hub-and-Spoke  : {d_hub_asia} km")
    print(f"  Greedy         : {d_gr_asia} km (selisih: {d_hub_asia - d_gr_asia} km dari Hub-and-Spoke)")
    print(f"  Branch & Bound : {d_bb_asia} km (selisih: {d_hub_asia - d_bb_asia} km dari Hub-and-Spoke)")
    print(f"  >> Selisih relatif kecil, Hub-and-Spoke masih layak")
    print(f"     dipertimbangkan untuk Asia Leg.")
    print()
    print("  [North America Leg]")
    print(f"  Hub-and-Spoke  : {d_hub_na} km")
    print(f"  Greedy         : {d_gr_na} km (selisih: {d_hub_na - d_gr_na} km dari Hub-and-Spoke)")
    print(f"  Branch & Bound : {d_bb_na} km (selisih: {d_hub_na - d_bb_na} km dari Hub-and-Spoke)")
    print(f"  >> Selisih sangat besar, Hub-and-Spoke tidak praktis untuk NA Leg.")
    print(f"     >> Rute optimal: {' -> '.join(r_bb_na)}")
    print()

# Menu
SCENARIOS = [
    ("Asia Leg - Hub-and-Spoke (Baseline aespa)", "asia", "hub"),
    ("Asia Leg - Greedy", "asia", "greedy"),
    ("Asia Leg - Branch and Bound", "asia", "bnb"),
    ("NA Leg - Hub-and-Spoke (Hypothetical)", "na", "hub"),
    ("NA Leg - Greedy", "na", "greedy"),
    ("NA Leg - Branch and Bound", "na", "bnb"),
    ("Perbandingan Selisih per Leg", "both", "compare"),
]

def get_data(leg):
    if leg == "asia":
        return dist_asia, cities_asia
    elif leg == "na":
        return dist_na, cities_na
    else:
        return None, None

def show_menu():
    print("\n" + "=" * 55)
    print("   Program Optimasi Rute Tur Konser aespa")
    print("=" * 55)
    print("     Pilih skenario yang ingin dijalankan:\n")
    for i, (title, _, _) in enumerate(SCENARIOS, 1):
        print(f"  [{i}] {title}")
    print(f"  [8] Jalankan semua skenario")
    print(f"  [0] Keluar")
    print()

def run_single(pilihan):
    title, leg, mode = SCENARIOS[pilihan - 1]
    print_separator()
    print(f"  Skenario {pilihan}: {title}")
    print_separator()
    if mode == "compare":
        run_comparison()
        return
    dist, cities = get_data(leg)
    if mode == "hub":
        r, d = baseline_hub_spoke(dist, cities)
        print_result("Hub-and-Spoke (Baseline)", r, d)
    elif mode == "greedy":
        r, d = greedy_tsp(dist, cities)
        print_result("Greedy (Nearest Neighbor)", r, d)
    elif mode == "bnb":
        r, d = branch_and_bound_tsp(dist, cities)
        print_result("Branch and Bound", r, d)
```

```
def run_all():
    for i in range(1, 8):
        run_single(i)

def main():
    while True:
        show_menu()
        try:
            pilihan   =   int(input("     Masukkan
pilihan (0-8): "))
        except ValueError:
            print("  Input tidak valid. Masukkan
angka 0-8.")
            continue

        if pilihan == 0:
            print("\n  Program selesai.")
            break
        elif pilihan == 8:
            run_all()
        elif 1 <= pilihan <= 7:
            run_single(pilihan)
        else:
            print("    Pilihan  tidak  tersedia.
Masukkan angka 0-8.")

if __name__ == "__main__":
    main()
```

C. *Route Simulation Output*

A Python CLI program was written to simulate all routing scenarios across both continent legs. The program runs interactively: the user selects a scenario from a numbered menu, and the program outputs the route sequence and total distance for that scenario. A total of seven scenarios are available, covering all algorithm and leg combinations, plus one dedicated cross-comparison scenario.

```
==================================================
   Program Optimasi Rute Tur Konser aespa
==================================================
  Pilih skenario yang ingin dijalankan:

  [1] Asia Leg - Hub-and-Spoke (Baseline aespa)
  [2] Asia Leg - Greedy
  [3] Asia Leg - Branch and Bound
  [4] NA Leg  - Hub-and-Spoke (Hypothetical)
  [5] NA Leg  - Greedy
  [6] NA Leg  - Branch and Bound
  [7] Perbandingan Selisih per Leg
  [8] Jalankan semua skenario
  [0] Keluar

  Masukkan pilihan (0-8): 8
--------------------------------------------------
  Skenario 1: Asia Leg - Hub-and-Spoke (Baseline aespa)
--------------------------------------------------
  Algoritma : Hub-and-Spoke (Baseline)
  Jarak     : 14580 km
  Rute      : Seoul -> Jakarta -> Seoul -> Osaka -> Seoul -> Tokyo -> Seoul

--------------------------------------------------
  Skenario 2: Asia Leg - Greedy
--------------------------------------------------
  Algoritma : Greedy (Nearest Neighbor)
  Jarak     : 12330 km
  Rute      : Seoul -> Osaka -> Tokyo -> Jakarta -> Seoul

--------------------------------------------------
  Skenario 3: Asia Leg - Branch and Bound
--------------------------------------------------
  Algoritma : Branch and Bound
  Jarak     : 12260 km
  Rute      : Seoul -> Jakarta -> Osaka -> Tokyo -> Seoul
```

Fig. 1. Program Output Run All Scenarios (Part 1)

```
--------------------------------------------------
  Skenario 4: NA Leg  - Hub-and-Spoke (Hypothetical)
--------------------------------------------------
  Algoritma : Hub-and-Spoke (Baseline)
  Jarak     : 70200 km
  Rute      : Seoul -> Los Angeles -> Seoul -> Oakland -> Seoul -> Seattle -> Seoul -> Vancouver -> Seoul

--------------------------------------------------
  Skenario 5: NA Leg  - Greedy
--------------------------------------------------
  Algoritma : Greedy (Nearest Neighbor)
  Jarak     : 19610 km
  Rute      : Seoul -> Vancouver -> Seattle -> Oakland -> Los Angeles -> Seoul

--------------------------------------------------
  Skenario 6: NA Leg  - Branch and Bound
--------------------------------------------------
  Algoritma : Branch and Bound
  Jarak     : 19470 km
  Rute      : Seoul -> Oakland -> Los Angeles -> Seattle -> Vancouver -> Seoul

--------------------------------------------------
  Skenario 7: Perbandingan Selisih per Leg
--------------------------------------------------
  [Asia Leg]
  Hub-and-Spoke  : 14580 km
  Greedy         : 12330 km  (selisih: 2250 km dari Hub-and-Spoke)
  Branch & Bound : 12260 km  (selisih: 2320 km dari Hub-and-Spoke)
  >> Selisih relatif kecil, Hub-and-Spoke masih layak
     dipertimbangkan untuk Asia Leg.

  [North America Leg]
  Hub-and-Spoke  : 70200 km
  Greedy         : 19610 km  (selisih: 50590 km dari Hub-and-Spoke)
  Branch & Bound : 19470 km  (selisih: 50730 km dari Hub-and-Spoke)
  >> Selisih sangat besar, Hub-and-Spoke tidak praktis untuk NA Leg.
  >> Rute optimal: Seoul -> Oakland -> Los Angeles -> Seattle -> Vancouver -> Seoul
```

Fig. 2. Program Output Run All Scenarios (Part 2)

The complete distance results across all scenarios are summarized in Table I.

TABLE I. ROUTE DISTANCE COMPARISON

Leg / Continent	Algorithm / Scenario	Route Sequence	Total Distance (km)
Asia	Baseline (Hub-Spoke)	Seoul-Jakarta-Seoul-Osaka-Seoul-Tokyo-Seoul	14,580
Asia	Greedy	Seoul-Osaka-Tokyo-Jakarta-Seoul	12,330
Asia	Branch & Bound	Seoul-Jakarta-Osaka-Tokyo-Seoul	12,260
NA	Baseline (Hub-Spoke)	Seoul-LA-Seoul-Oakland-Seoul-Seattle-Seoul-Vancouver-Seoul	70,200
NA	Greedy	Seoul-Vancouver-Seattle-Oakland-LA-Seoul	19,610
NA	Branch & Bound	Seoul-Oakland-LA-Seattle-Vancouver-Seoul	19,470

D. *Scenario-Based Testing*

The seven scenarios tested in this study are structured as follows. Scenarios 1 through 3 apply the Hub-and-Spoke baseline, Greedy, and Branch and Bound algorithms to the Asian leg respectively. Scenarios 4 through 6 repeat the same sequence for the North American leg. Scenario 7 serves as a summary comparison, presenting the distance gap between Hub-and-Spoke and each algorithm side by side for both legs, making it easier to see how significant the routing inefficiency is on each continent.

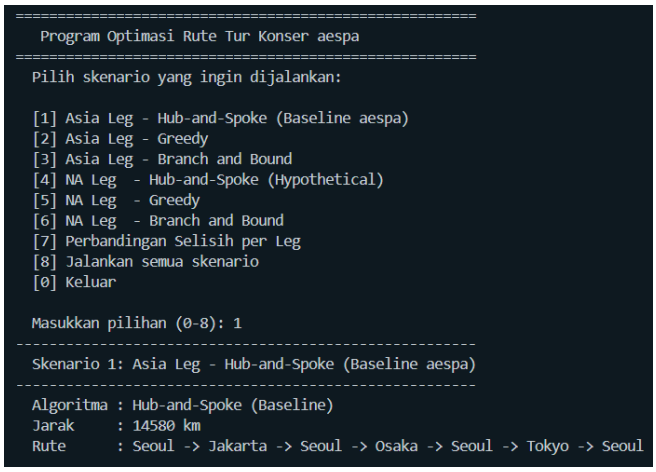

Fig. 3. Program Output Asia Leg - Hub-and-Spoke (Baseline aespa)

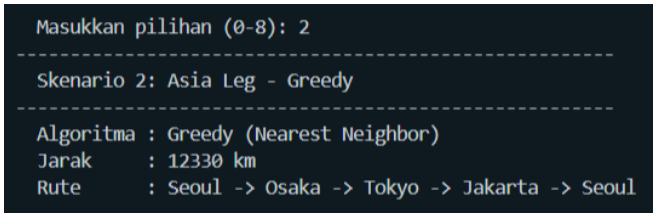

Fig. 4. Program Output Asia Leg – Greedy

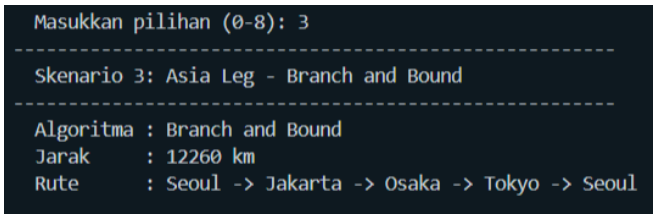

Fig. 5. Program Output Asia Leg - Branch and Bound

Fig. 6. Program Output NA Leg - Hub-and-Spoke (Hypothetical)

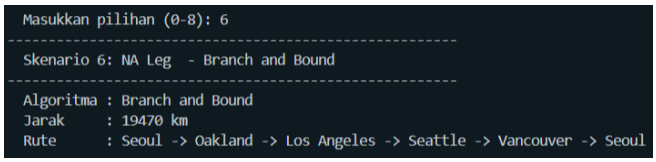

Fig. 7. Program Output NA Leg - Greedy

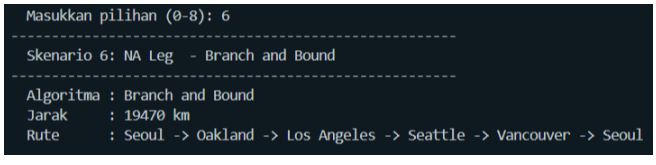

Fig. 8. Program Output NA Leg - Branch and Bound

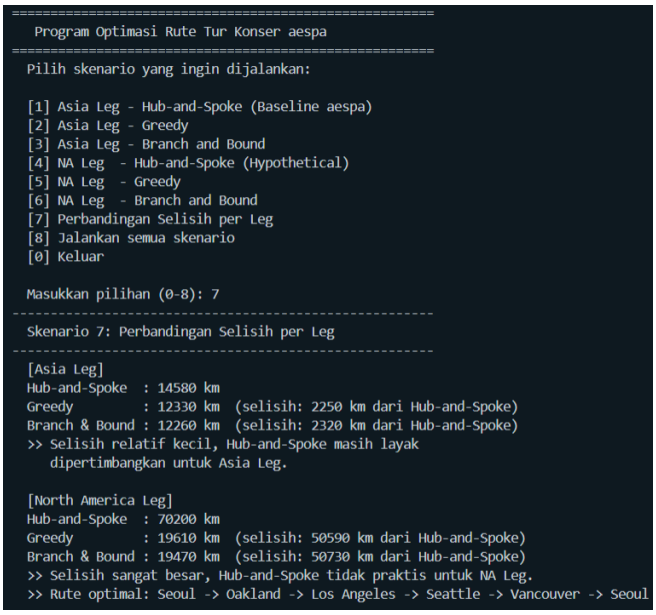

Fig. 9. Program Output Perbandingan Selisih per Leg

IV. Analysis and Discussion

A. *Distance Efficiency Comparison*

Table I highlights a significant difference in route efficiency between the two continents. For the Asian leg, switching from the Baseline to the Branch and Bound optimal route saves approximately 2,320 km, which is measurable but not dramatic. For the North American leg, the gap is a different story entirely. The Hub-and-Spoke Baseline accumulates 70,200 km, while the Branch and Bound route brings that down to 19,470 km, a reduction of 50,730 km. This 50,730 km difference demonstrates a massive logistical and financial inefficiency if the Hub-and-Spoke model were to be applied.

B. *Algorithmic Performance*

In terms of algorithmic behavior, the results align well with what the theory predicts. The Greedy algorithm produced near-optimal routes on both legs, 12,330 km for Asia and 19,610 km for North America, but fell short of the true optimum in both cases. Branch and Bound, on the other hand, found the exact shortest path each time: 12,260 km for Asia and 19,470 km for North America. The performance gap between the two is small in absolute terms, but the underlying reason matters. Greedy commits to each local decision permanently, which means an early suboptimal choice cannot be corrected later. Branch and Bound avoids this by pruning branches that cannot possibly beat the current best solution, allowing it to explore the solution space more thoroughly [8].

C. *Real-World Feasibility Analysis*

Looking at the numbers from a practical standpoint, SM Entertainment's scheduling choices actually make a lot of sense given the geography. For the Asian leg, the 2,320 km penalty for returning to Seoul after each concert is relatively

manageable, especially when the concerts are spaced roughly a week apart. Flying home gives the artists time to rest, attend domestic schedules, and prepare for the next stop without significant logistical cost.

The North American leg is a completely different situation. A round trip between Seoul and any North American city already exceeds 16,000 km on its own, so applying the Hub-and-Spoke model intercontinentally would be logistically and physically unreasonable. The 70,200 km Baseline figure confirms this: it is not just inefficient, it is impractical. This is likely why intercontinental tour legs are always scheduled as sequential runs rather than hub-based ones.

Based on the Branch and Bound result, the current planned sequence for the North American leg can still be improved. Rather than flying from Seoul to Los Angeles first, the optimal sequence is Seoul, Oakland, Los Angeles, Seattle, Vancouver, then back to Seoul. This routing minimizes total flight distance while following a geographically logical path down and then back up the West Coast.

D. Cross-Continental Comparison

Scenario 7 summarizes the cost difference between Hub-and-Spoke and the two algorithms for each leg. For the Asian leg, the gap between Hub-and-Spoke and Branch and Bound is only 2,320 km, small enough that the real-world scheduling choice to return to Seoul remains defensible. For the North American leg, that gap jumps to 50,730 km against Branch and Bound and 50,590 km against Greedy. Both algorithms arrive at roughly the same conclusion: Hub-and-Spoke is not viable intercontinentally. The difference in scale between the two legs is what ultimately justifies treating them with entirely different scheduling logic.

V. CONCLUSION

This study evaluates the mathematical efficiency of concert tour routing, revealing that the optimal strategy depends significantly on the continent's geographic scale.

For the Asian leg, the Hub-and-Spoke model incurs only a 2,320 km penalty over the optimal route. Given that concerts are spaced about a week apart and returning home supports artist well-being, this trade-off seems justified. This demonstrates that the mathematical penalty is acceptable for the artist's convenience.

However, the inefficiency becomes highly apparent in the North American leg. The Hub-and-Spoke Baseline would accumulate over 70,000 km, compared to just 19,470 km under the Branch and Bound optimal sequence. Sequential routing on intercontinental legs is not just mathematically preferable, it is the only practical option.

On the algorithmic side, both Greedy and Branch and Bound produced reasonable results, but Branch and Bound consistently found the exact shortest path. Greedy is faster to compute, but its tendency to commit to local optima makes it unreliable for absolute distance minimization. For the upcoming North American leg specifically, the optimal sequence this study identifies is Seoul, Oakland, Los Angeles, Seattle, Vancouver, and back to Seoul.

VIDEO LINK AT YOUTUBE

https://youtu.be/Ir6Kk9k5BUw

ACKNOWLEDGMENT

The author expresses her sincere gratitude to Allah SWT for the endless blessings and invaluable guidance she received throughout the entire research and writing of this paper. Furthermore, the author expresses gratitude to the lecturers of the IF2211 Strategi Algoritma course, especially Dr. Ir. Rinaldi Munir, M.T., Ir. Rila Mandala, M.Eng., Ph.D., Dr. techn. Muhammad Zuhri Catur Candra, S.T, M.T., Tricya Esterina Widagdo, S.T., M.Sc., and Dr. Fazat Nur Azizah, S.T., M.Sc. who dedicated themselves to helping their students by providing numerous learning resources and opportunities. Finally, the author would like to express sincere gratitude to her family for their unwavering support and constant encouragement throughout her entire academic journey.

REFERENCES

[1] snowdrop_010101 (@snowdrop_010101), "260309 Preview 안비 #WINTER #윈터 #aespa #에스파," X (formerly Twitter), Mar. 9, 2026. [Online]. Available: https://x.com/snowdrop_010101/status/2030802479160705125?s=20

[2] A. K. Wiramukti, S. F. Hidayah, N. A. Permana, and A. Soimun, "Analisis Efisiensi Distribusi Barang Dengan Pendekatan Saving Matrix Dan Milk Run Di CV XYZ," *Jurnal Transportasi*, vol. 25, no. 2, pp. 169-180, 2025.

[3] P. E. Haloho, "Peran Logistik Dalam Pembangunan Ekonomi: Analisis Terhadap Manajemen Risiko Dan Efisiensi Transportasi," *Jurnal Ilmiah Wahana Pendidikan*, vol. 10, no. 24, pp. 378-385, Dec. 2024.

[4] T. D. J. Saputra and D. A. W. Yanti, "Pengaruh Peran Manajemen Logistik Dalam Meningkatkan Efisiensi Distribusi Di PT Bahari Utama Samudra," *MBI*, vol. 19, no. 9, pp. 5669-5674, Apr. 2025.

[5] R. Munir, "Algoritma Greedy (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Sekolah Teknik Elektro dan Informatika ITB, 2026.

[6] R. Munir, N. U. Maulidevi, and M. L. Khodra, "Algoritma Branch and Bound (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Sekolah Teknik Elektro dan Informatika ITB, 2026.

[7] R. Munir, N. U. Maulidevi, and M. L. Khodra, "Algoritma Branch and Bound (Bagian 2)," Bahan Kuliah IF2211 Strategi Algoritma, Sekolah Teknik Elektro dan Informatika ITB, 2026.

[8] R. Munir, N. U. Maulidevi, and M. L. Khodra, "Algoritma Branch and Bound (Bagian 3)," Bahan Kuliah IF2211 Strategi Algoritma, Sekolah Teknik Elektro dan Informatika ITB, 2026.

STATEMENT

I hereby declare that this paper is my own writing, not an adaptation or translation of someone else's paper, and not plagiarized.

Jakarta, 19 June 2026

Amanda Aurellia Salsabilla (13524131)