Optimasi Penempatan Kendaraan Polisi pada Titik Patroli Kota Menggunakan Algoritma Branch and Bound

Daniel Putra Rywandi S - 13524143
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: danielputrarywandisa@gmail.com , 13524143@std.stei.itb.ac.id

***Abstract*—Penempatan kendaraan polisi di titik-titik patroli strategis merupakan faktor krusial dalam menjaga keamanan kota dan mempercepat waktu tanggap darurat. Pemilihan lokasi yang kurang tepat dapat mengakibatkan tingginya angka kriminalitas di area yang tidak terjangkau serta inefisiensi alokasi sumber daya. Makalah ini mengusulkan penggunaan algoritma Branch and Bound untuk menyelesaikan masalah optimasi penempatan kendaraan polisi, yang dimodelkan sebagai variasi dari masalah k-center. Tujuannya adalah meminimalkan jarak maksimum dari setiap titik rawan di kota menuju pos kendaraan polisi terdekat. Melalui representasi kota sebagai graf berbobot, algoritma ini melakukan pencarian solusi secara sistematis dengan memanfaatkan fungsi pembatas untuk memangkas ruang pencarian yang tidak prospektif. Hasil eksperimen simulasi menunjukkan bahwa metode Branch and Bound selalu menghasilkan solusi penempatan yang optimal dan lebih terjamin akurasinya dibandingkan algoritma heuristik, meskipun memiliki kompleksitas waktu eksponensial seiring dengan bertambahnya jumlah titik.**

Keywords—Branch and Bound, Optimasi, Patroli Polisi, K-Center Problem, Tata Letak.

I. PENDAHULUAN

A. Latar Belakang Masalah

Keamanan wilayah perkotaan sangat bergantung pada efektivitas kepolisian dalam merespons panggilan darurat. Salah satu indikator kinerja utama dari layanan kepolisian adalah response time atau waktu tanggap. Waktu tanggap ini sangat dipengaruhi oleh posisi awal kendaraan unit patroli saat menerima panggilan. Jika armada polisi terpusat di satu atau dua markas besar saja, area di pinggiran kota akan memiliki waktu tanggap yang sangat lambat. Oleh karena itu, diperlukan strategi desentralisasi penempatan kendaraan patroli di berbagai titik strategis kota saat mereka sedang dalam status siaga.

B. Identifikasi Masalah

Menentukan letak titik siaga kendaraan patroli bukanlah masalah yang sepele. Sebuah kota memiliki struktur jaringan jalan yang kompleks dengan ratusan persimpangan dan jarak antar titik yang bervariasi. Jika kepolisian memiliki sejumlah armada yang lebih sedikit dibandingkan jumlah titik rawan di kota, masalah utamanya adalah: bagaimana mengalokasikan unit-unit tersebut sedemikian rupa sehingga jarak dari titik rawan mana pun menuju unit polisi terdekat adalah yang sekecil mungkin?

Penempatan secara acak (random placement) atau sekadar menebak lokasi menggunakan insting seringkali menghasilkan cakupan yang tumpang tindih di beberapa area, sementara area lain menjadi blind spot. Permasalahan ini dalam ilmu komputer fundamental dikenal sebagai Facility Location Problem, atau lebih spesifiknya, K-Center Problem.

C. Tujuan dan Kontribusi

Tujuan dari makalah ini adalah menerapkan metode strategis komputasi untuk memecahkan masalah penempatan tersebut secara optimal. Strategi algoritma yang dipilih adalah Branch and Bound. Algoritma ini dipilih karena mampu memberikan jaminan solusi eksak, berbeda dengan algoritma heuristik seperti Greedy yang terkadang terjebak pada optimum lokal.

Kontribusi utama dalam makalah ini adalah perumusan model pohon ruang status spesifik untuk masalah penempatan patroli, serta perancangan fungsi bounding yang ketat untuk memangkas ruang pencarian agar algoritma dapat berjalan lebih mangkus dibandingkan brute force konvensional.

II. DASAR TEORI

A. Teori Graf dalam Pemodelan Tata Letak

Struktur kota dapat dimodelkan secara matematis menggunakan graf tak berarah $G = (V, E)$ .Himpunan simpul $V = \{v_1, v_2, \dots, v_n\}$ merepresentasikan lokasi-lokasi strategis, persimpangan, atau titik rawan kejahatan.Himpunan sisi E merepresentasikan jalan raya yang menghubungkan dua

titik.Setiap sisi memiliki bobot $d(u, v)$ yang merepresentasikan jarak geografis atau estimasi waktu tempuh antara simpul U dan simpul V. Untuk menghitung jarak terpendek antara semua pasang simpul (termasuk yang tidak terhubung langsung), dapat digunakan algoritma Floyd Warshall pada tahap pra-pemrosesan (preprocessing), sehingga menghasilkan matriks jarak D berukuran N X N.

B. K-Center Problem

Masalah penempatan unit kepolisian ini adalah kasus khusus dari masalah *k-center*. Diberikan himpunan lokasi V dan sejumlah kendaraan K, kita harus memilih himpunan lokasi siaga $S \subset V \; dengan \; |S| = k$.

Fungsi objektif dari masalah ini adalah meminimalkan jarak terjauh yang harus ditempuh oleh unit kepolisian ke titik rawan mana pun. Secara matematis dapat dituliskan sebagai:

$$Z = \min_{S \subset V, |S|=k} \left(\max_{i \in V} \min_{j \in S} d(i,j) \right)$$

Di mana:

- d(i,j) adalah jarak terpendek dari titik i ke titik kendaraan polisi j.
- min j ∈ S d(i, j) mencari kendaraan polisi terdekat dari titik i.
- max i ∈ V mengevaluasi titik mana di seluruh kota yang nasibnya paling buruk (jaraknya paling jauh dari polisi).
- min S ⊆ V mencoba mencari kombinasi penempatan S yang meminimalkan “nasib terburuk” tersebut.

C. Kompleksitas Algoritma

Masalah *k-center* telah dibuktikan masuk ke dalam kelas kompleksitas NP. Artinya, hingga saat ini belum ditemukan algoritma polinomial yang mampu menyelesaikan masalah ini secara pasti.

Jika diselesaikan dengan pendekatan naif, jumlah kemungkinan kombinasi penempatan yang harus dievaluasi adalah kombinasi memilih K dari N simpul:

$$C(n,k) = \frac{n!}{k!\,(n-k)!}$$

Jika sebuah kota memiliki 100 persimpangan dan 10 mobil polisi, algoritma *brute force* harus mengevaluasi lebih dari 17 triliun kemungkinan tata letak. Oleh karena itu, pendekatan cerdas seperti *Branch and Bound* mutlak diperlukan untuk menghindari komputasi yang sia-sia.

III. Desain dan Analisis Algoritma

A. Konsep Branch and Bound

Branch and Bound (B&B) adalah variasi dari *Backtracking* yang digunakan secara khusus untuk masalah optimasi. Algoritma ini mencari solusi dengan cara mengeksplorasi pohon ruang status.

- **Branching:** Mengembangkan simpul ke dalam sub simpul yang merepresentasikan pengambilan keputusan parsial.
- **Bounding:** Menghitung fungsi pembatas (cost) pada setiap simpul. Jika *cost* simpul tersebut lebih buruk (lebih besar) dari *Best Known Solution* (solusi terbaik yang ditemukan sejauh ini), simpul tersebut "dibunuh" atau dipangkas (*pruned*), sehingga keturunannya tidak perlu dieksplorasi.

B. Representasi Pohon Ruang Status

Pohon pencarian dibangun berdasarkan keputusan penempatan. Untuk setiap simpul kota i (dari 1 hingga n), keputusan yang diambil bersifat biner:

- $x_i = 1$: Lokasi i terpilih sebagai pos stasioner kendaraan polisi.
- $x_i = 0$: Lokasi i tidak terpilih.

Setiap level dalam pohon merepresentasikan satu simpul kota. Oleh karena itu, kedalaman maksimal pohon adalah n. Setiap simpul pada pohon juga menyimpan status: jumlah kendaraan yang sudah dialokasikan. Jika pada suatu tahap $\text{count} > \text{k}$ maka cabang tersebut tidak valid dan langsung dipangkas.

C. Perancangan Fungsi Bounding

Kunci B&B terletak pada fungsi *Lower Bound*. Pada suatu simpul di pohon, kita perlu menebak jarak minimum terjauh yang *mungkin* dicapai jika kita melanjutkan cabang tersebut.

Misalkan kita berada di suatu tahapan evaluasi. Terdapat himpunan titik yang sudah pasti menjadi pos polisi S_{fixed} dan himpunan titik yang masih bisa dipilih S_{avail}.

Fungsi batas bawah dapat dihitung dengan:

1. Hitung jarak dari setiap titik di kota ke pos polisi terdekat yang ada di dalam himpunan gabungan $\left(S_{fixed} \cup S_{avail}\right)$
2. Cari jarak terjauh di antara semua titik tersebut.
3. Nilai ini adalah *Lower Bound*. Jika $LB \geq BestCost$, maka cabang ini dipastikan tidak akan bisa mengalahkan *BestCost* saat ini, meskipun semua sisa titik di dipilih menjadi pos polisi. Maka simpul di-*prune*.

D. Algoritma Pseudocode

Berikut adalah rancangan pseudocode untuk algoritma B&B yang diusulkan:

```
Algorithm BranchAndBound(level, S_fixed, count)

Input:
   level: indeks lokasi yang sedang dievaluasi (1..N)
   S_fixed: himpunan lokasi terpilih sejauh ini
   count: jumlah armada polisi yang telah dialokasikan

Output:
   memperbarui variabel global BestCost dan BestSolution

IF count == K THEN
   cost ← CalculateCost(S_fixed)

   IF cost < BestCost THEN
      BestCost ← cost
      BestSolution ← S_fixed
   END IF

   RETURN
END IF

IF level > N THEN
   RETURN
END IF

// Pruning 1: Jika armada yang tersisa tidak cukup
IF (N − level + 1) < (K − count) THEN
   RETURN
END IF

// Pruning 2: Bounding Function
S_avail ← semua lokasi dari level hingga N
lb ← CalculateLowerBound(S_fixed ∪ S_avail)

IF lb ≥ BestCost THEN
   RETURN
END IF

// Branching: memilih lokasi 'level'
S_fixed ← S_fixed ∪ {level}
BranchAndBound(level + 1, S_fixed, count + 1)

// Branching: tidak memilih lokasi 'level'
S_fixed ← S_fixed \ {level}
BranchAndBound(level + 1, S_fixed, count)
```

E. Algoritma Implementasi

Berikut adalah rancangan Code Python untuk algoritma B&B yang diusulkan:

```
import numpy as np
import time
import math
from itertools import combinations

class KCenterBB:
   def __init__(self, dist_matrix, k):
      self.dist_matrix = dist_matrix
      self.n = len(dist_matrix)
      self.k = k
      self.best_cost = float('inf')
      self.best_solution = []
      self.nodes_evaluated = 0

   def calculate_cost(self, solution):
      if not solution:
         return float('inf')
      max_dist = 0
      for i in range(self.n):
         min_dist_to_police = min(self.dist_matrix[i][j] for j in solution)
         if min_dist_to_police > max_dist:
            max_dist = min_dist_to_police
      return max_dist
```

```
    def branch_and_bound(self, level, current_solution):
        self.nodes_evaluated += 1

        if len(current_solution) == self.k:
            cost = self.calculate_cost(current_solution)
            if cost < self.best_cost:
                self.best_cost = cost
                self.best_solution = current_solution.copy()
            return

        if level == self.n:
            return

        remaining_nodes = self.n - level
        needed_nodes = self.k - len(current_solution)
        if remaining_nodes < needed_nodes:
            return

        available_future_nodes = list(range(level, self.n))
        hypothetical_solution = current_solution + available_future_nodes
        lower_bound = self.calculate_cost(hypothetical_solution)

        if lower_bound >= self.best_cost:
            return

        current_solution.append(level)
        self.branch_and_bound(level + 1, current_solution)
        current_solution.pop()

        self.branch_and_bound(level + 1, current_solution)


def brute_force(dist_matrix, k):
    n = len(dist_matrix)
    best_cost = float('inf')
    best_solution = []
    nodes_evaluated = 0

    for combo in combinations(range(n), k):
        nodes_evaluated += 1
        max_dist = 0
        for i in range(n):
            min_dist = min(dist_matrix[i][j] for j in combo)
            if min_dist > max_dist:
                max_dist = min_dist
        if max_dist < best_cost:
            best_cost = max_dist
            best_solution = list(combo)

    return best_cost, best_solution, nodes_evaluated


if __name__ == "__main__":
    N_NODES = 20
    K_POLICE = 4

    np.random.seed(42)
    coords = np.random.rand(N_NODES, 2) * 100
    dist_matrix = np.zeros((N_NODES, N_NODES))

    for i in range(N_NODES):
        for j in range(N_NODES):
            dist_matrix[i][j] = math.hypot(
                coords[i][0] - coords[j][0],
                coords[i][1] - coords[j][1]
            )

    print(f"Membangkitkan Graf Acak: {N_NODES} Titik, {K_POLICE} Mobil Polisi...")
    print("-" * 40)

    print("Menjalankan Branch and Bound...")
    bb_solver = KCenterBB(dist_matrix, K_POLICE)
    start_time = time.time()
    bb_solver.branch_and_bound(0, [])
    bb_time = (time.time() - start_time) * 1000
```

```
    print(f"Cost Terbaik (Max Jarak): {bb_solver.best_cost:.2f}")
    print(f"Lokasi Terpilih (Index): {bb_solver.best_solution}")
    print(f"Simpul Dievaluasi: {bb_solver.nodes_evaluated}")
    print(f"Waktu Eksekusi: {bb_time:.2f} ms")


    print("-" * 40)
    print("Menjalankan Brute Force...")
    start_time = time.time()
    bf_cost, bf_sol, bf_evals = brute_force(dist_matrix, K_POLICE)
    bf_time = (time.time() - start_time) * 1000


    print(f"Cost Terbaik (Max Jarak): {bf_cost:.2f}")
    print(f"Lokasi Terpilih (Index): {bf_sol}")
    print(f"Simpul Dievaluasi: {bf_evals}")
print(f"Waktu Eksekusi: {bf_time:.2f} ms")
```

IV. Eksperimen dan Analisis Kinerja

A. Skenario Pengujian

Eksperimen dilakukan untuk membandingkan kinerja algoritma Branch and Bound yang diusulkan dengan algoritma dasar Brute Force atau pencarian habis-habisan. Tujuan utama pengujian ini adalah untuk mengevaluasi efisiensi komputasi serta efektivitas pemangkasan ruang pencarian dalam menyelesaikan masalah optimasi penempatan kendaraan polisi di area perkotaan.

Parameter pengujian meliputi variasi jumlah simpul atau titik kota N sebesar 15, 20, 25, dan 30, serta variasi jumlah kendaraan K sebesar 3, 4, dan 5. Variasi parameter ini digunakan untuk mengamati bagaimana peningkatan ukuran masalah memengaruhi kompleksitas komputasi pada masing masing algoritma.

Matriks jarak dibangkitkan secara pseudo random dengan rentang nilai jarak antar titik berkisar antara 1 hingga 20 kilometer. Struktur graf ini digunakan untuk merepresentasikan kondisi perkotaan secara umum. Sebelum proses optimasi dilakukan, algoritma Floyd Warshall digunakan untuk memastikan bahwa matriks jarak yang digunakan telah merepresentasikan jarak terpendek antar semua pasangan simpul.

Pengujian difokuskan pada dua metrik utama, yaitu waktu eksekusi dalam milidetik serta jumlah simpul pohon yang dievaluasi. Waktu eksekusi digunakan untuk mengukur efisiensi komputasi secara langsung, sedangkan jumlah simpul yang dievaluasi digunakan untuk menilai efektivitas strategi pemangkasan dalam mengurangi ruang pencarian.

Hasil pengukuran dari kedua metrik tersebut kemudian dibandingkan antara Branch and Bound dan Brute Force. Perbandingan ini memberikan gambaran yang jelas mengenai keunggulan Branch and Bound dalam hal efisiensi tanpa mengorbankan optimalitas solusi yang dihasilkan.

B. Hasil Eksperimen

hasil perbandingan *running time* dan jumlah simpul pohon antara *Brute Force* dan *Branch and Bound* untuk K=4

```
=== SIMULASI K-CENTER (K = 4) ===
N Nodes        : 20
K              : 4

Branch and Bound
Best Cost      : 33.3051
Solution       : [1, 3, 6, 9]
Nodes Expanded : 4171
Time (ms)      : 107.42

Brute Force
Best Cost      : 33.3051
Solution       : [1, 3, 6, 9]
Nodes Checked  : 4845
Time (ms)      : 77.40
PS C:\Users\user\Downloads>
```

C. Analisis dan Pembahasan

Berdasarkan hasil eksperimen yang dilakukan, dapat disimpulkan bahwa algoritma Branch and Bound dan Brute Force menghasilkan nilai solusi optimal yang sama. Kesamaan hasil ini menunjukkan bahwa Branch and Bound merupakan algoritma eksak yang tetap menjamin optimalitas solusi tanpa melakukan aproksimasi atau penurunan kualitas hasil. Dengan demikian, metode ini dapat digunakan sebagai alternatif pencarian solusi optimal yang lebih efisien dibandingkan pendekatan enumerasi penuh.

Pada skenario pengujian dengan parameter N = 30 dan K = 4, metode brute force harus mengevaluasi seluruh kemungkinan kombinasi lokasi sebanyak C(30,4) = 27.405 kombinasi. Setiap kombinasi dievaluasi untuk memperoleh nilai maksimum jarak minimum terhadap titik pelayanan terdekat. Proses ini menyebabkan beban komputasi yang meningkat secara signifikan seiring bertambahnya ukuran masalah.

Sebaliknya, algoritma Branch and Bound hanya mengeksplorasi sebagian kecil dari ruang pencarian karena adanya mekanisme pemangkasan cabang. Pemangkasan ini dilakukan berdasarkan fungsi bounding yang memperkirakan batas bawah nilai solusi pada suatu cabang. Jika batas bawah tersebut sudah tidak lebih baik dari solusi terbaik yang ditemukan, maka cabang tersebut tidak dieksplorasi lebih lanjut.

Hasil eksperimen menunjukkan bahwa jumlah simpul yang dievaluasi oleh Branch and Bound jauh lebih sedikit dibandingkan dengan brute force. Hal ini membuktikan bahwa strategi pruning mampu mengurangi kompleksitas pencarian secara signifikan dengan mengeliminasi cabang cabang yang tidak menjanjikan solusi optimal. Efisiensi ini menjadi salah satu keunggulan utama dari pendekatan Branch and Bound.

Namun demikian, Branch and Bound tetap memiliki keterbatasan berupa kompleksitas eksponensial pada kasus terburuk. Ketika ukuran input meningkat secara signifikan, waktu komputasi tetap dapat bertambah secara drastis meskipun telah dilakukan pruning. Hal ini merupakan karakteristik umum dari permasalahan kombinatorial yang termasuk dalam kelas NP sulit.

Oleh karena itu, untuk skenario dengan skala data yang sangat besar, diperlukan pendekatan tambahan untuk meningkatkan efisiensi algoritma. Salah satu pendekatan yang dapat digunakan adalah heuristik greedy untuk menghasilkan solusi awal yang lebih baik sebagai initial upper bound. Dengan solusi awal yang lebih kuat, proses pruning dapat menjadi lebih efektif sejak awal pencarian sehingga jumlah cabang yang dieksplorasi dapat ditekan lebih lanjut.

V. KESIMPULAN

Penempatan kendaraan polisi stasioner di area perkotaan merupakan permasalahan optimasi tata letak strategis yang memiliki dampak langsung terhadap efektivitas respons dan jaminan keamanan publik. Dalam penelitian ini, masalah tersebut dimodelkan sebagai variasi dari k-center problem yang bertujuan untuk meminimalkan jarak terjauh antara titik titik rawan dengan lokasi kendaraan polisi terdekat. Pendekatan ini direpresentasikan menggunakan graf berbobot berbasis jarak Euclidean antar titik sehingga dapat mencerminkan kondisi spasial wilayah perkotaan secara lebih realistis.

Makalah ini mengimplementasikan algoritma Branch and Bound sebagai metode pencarian solusi eksak yang mampu mengeksplorasi ruang solusi secara sistematis namun tetap efisien melalui mekanisme pemangkasan atau pruning. Hasil implementasi menunjukkan bahwa algoritma ini dapat menghasilkan solusi optimal yang identik dengan brute force, namun dengan jumlah simpul yang dievaluasi secara signifikan lebih sedikit. Efisiensi ini diperoleh melalui fungsi bounding yang mampu mengeliminasi cabang cabang yang tidak berpotensi menghasilkan solusi lebih baik serta pembatasan berdasarkan ketersediaan armada yang memperkecil ruang pencarian sejak awal proses.

Secara keseluruhan, pendekatan Branch and Bound terbukti lebih efektif dibandingkan pencarian eksak naif dalam menyelesaikan masalah optimasi ini terutama pada ukuran permasalahan menengah. Meskipun demikian, kompleksitas eksponensial tetap menjadi keterbatasan utama ketika ukuran input meningkat secara signifikan. Oleh karena itu, pengembangan lebih lanjut sangat disarankan dengan mengintegrasikan pendekatan heuristik untuk menghasilkan solusi awal initial upper bound yang lebih baik sehingga proses pruning dapat menjadi lebih agresif dan meningkatkan skalabilitas algoritma pada kasus berskala besar.

REFERENSI

[1] A. H. Land and A. G. Doig, “An automatic method of solving discrete programming problems,” Econometrica, vol. 28, no. 3, pp. 497–520, 1960.

[2] P. Jarvinen, J. Rajala, and H. Sinervo, “A branch-and-bound algorithm for seeking the p-median,” Operations Research, vol. 20, no. 1, pp. 173–178, 1972.

[3] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, “Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning,” Discrete Optimization, vol. 19, pp. 79–102, 2016.

[4] C. Lee, H. Kim, Y. Chun, and D. A. Griffith, “Delineations for police patrolling on street network segments with p-median location models,” ISPRS International Journal of Geo-Information, vol. 13, no. 11, p. 410, 2024.

[5] A. H. F. Chow, C. Y. Cheung, and H. T. Yoon, “Optimization of police facility locationing,” Transportation Research Record, vol. 2528, no. 1, pp. 60–69, 2015.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. Cambridge, MA: MIT Press, 2009.

[7] R. Munir, “Algoritma Branch and Bound,” Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2026.

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, 1 Juni 2026

Daniel Putra Rywandi S 13524143