Penempatan Kamera Minimum untuk Cakupan Pengawasan Melalui Branch-and-Bound pada Masalah Set-Covering

Muhammad Farrel Wibowo - 13523153
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: faawibowo@gmail.com , 13523153@std.stei.itb.ac.id

***Abstrak*—Masalah penempatan kamera pengawasan merupakan permasalahan optimasi yang dapat dimodelkan sebagai varian dari *Set Covering Problem* (SCP). Makalah ini membahas implementasi algoritma *Branch and Bound* untuk menyelesaikan permasalahan tersebut dengan pendekatan pemangkasan simpul pencarian dan estimasi solusi optimal berbasis metode greedy. Sistem menerima masukan berupa koordinat titik pengawasan dan radius jangkauan kamera, lalu menentukan jumlah minimum kamera yang diperlukan untuk mencakup seluruh titik. Evaluasi dilakukan dengan berbagai skenario jumlah titik, distribusi (acak/grid/spiral), serta variasi radius kamera. Hasil eksperimen menunjukkan bahwa algoritma mampu mencapai solusi optimal secara efisien, khususnya ketika radius kamera cukup besar atau distribusi titik mengikuti pola tertentu. Visualisasi cakupan turut disertakan untuk memperkuat interpretasi hasil. Kontribusi utama dari makalah ini adalah penerapan strategi estimasi *greedy* sebagai fungsi heuristik dalam algoritma *Branch and Bound* untuk meningkatkan efisiensi eksplorasi simpul solusi.**

Keywords—Branch and Bound, Set Covering, Penempatan Kamera, Greedy, Optimasi Kombinatorial

I. PENDAHULUAN

Sistem pengawasan kamera memainkan peran vital dalam keamanan berbagai sektor seperti fasilitas umum, institusi pendidikan, dan bangunan komersial. Efektivitas sistem ini sangat bergantung pada penempatan kamera yang optimal, karena penempatan yang keliru dapat menimbulkan titik buta dan mengurangi cakupan pengawasan. Oleh karena itu, optimasi penempatan kamera merupakan aspek krusial untuk memastikan visibilitas area yang ingin dipantau secara real-time.

Masalah penempatan kamera pengawasan dapat dimodelkan sebagai *Set Covering Problem* (SCP), yang dikenal sebagai masalah NP-hard. Kompleksitasnya meningkat seiring bertambahnya jumlah titik pengawasan dan parameter spasial kamera, sehingga diperlukan pendekatan algoritmik yang efisien. Salah satu pendekatan tersebut adalah algoritma *Branch and Bound* (B&B), yang memungkinkan pencarian solusi optimal melalui eksplorasi pohon keputusan dan strategi pemangkasan (pruning).

Makalah ini menyajikan model SCP untuk penempatan kamera minimum dan implementasi algoritma B&B sebagai metode penyelesaiannya. Evaluasi dilakukan terhadap berbagai skenario jumlah titik, pola distribusi, dan radius jangkauan kamera. Hasil eksperimen dianalisis berdasarkan efektivitas dan efisiensi solusi, serta dibandingkan secara visual. Diharapkan makalah ini dapat memberikan kontribusi terhadap pengembangan pendekatan optimasi pada sistem pengawasan berbasis algoritma pencarian sistematis.

II. DASAR TEORI

A. Set Covering Problem

Masalah Set Covering (SCP) adalah sebuah masalah optimisasi kombinatorial yang terkenal. Pentingnya masalah ini berasal dari banyaknya aplikasi di dunia nyata, mulai dari penjadwalan kru pada maskapai penerbangan atau perusahaan kereta api (Hoffmann & Padberg, 1993; Caprara et al., 1997; Housos & Elmoth, 1997) dan penjadwalan pengemudi pada transportasi umum (Lourenço et al., 2001) hingga penjadwalan dan perencanaan produksi di berbagai industri (Vasko dan Wolf, 1987).

SCP dapat diformalkan sebagai berikut. Diberikan sebuah himpunan hingga $A=\{a_1,\ldots,a_m\}$ dan sebuah keluarga $F=\{A_1,\ldots,A_n\}$ berupa himpunan bagian $A_i \subseteq A$ yang mencakup A, artinya setiap elemen dari A muncul dalam setidaknya satu himpunan pada F. Pada *Minimum SCP*, tujuannya adalah mencari himpunan bagian minimum $C^* \subseteq F$ yang mencakup A. Pada *Weighted SCP*, terdapat fungsi bobot $w : F \rightarrow \mathbb{R}^+$ yang memberikan bobot (atau biaya) positif pada setiap elemen F, dan tujuannya adalah menemukan set cover C^* dengan total bobot minimum. Jelas, *Minimum SCP* merupakan kasus khusus dari *Weighted SCP*, di mana semua elemen dari F memiliki bobot yang sama, sehingga sering disebut sebagai *Unicost SCP*.

Set Covering Problem (SCP) seringkali dimodelkan menggunakan pendekatan *integer programming* karena sifat masalahnya yang membutuhkan pemilihan kombinasi optimal dari sekumpulan himpunan bagian. Pada model ini, setiap himpunan bagian yang menjadi kandidat solusi direpresentasikan oleh variabel biner, di mana nilai 1 berarti himpunan tersebut dipilih dan nilai 0 berarti tidak dipilih. Dalam

konteks ini, tujuan utamanya adalah meminimalkan total biaya atau jumlah himpunan yang digunakan, dengan memastikan bahwa setiap elemen pada himpunan semesta tercakup oleh setidaknya satu himpunan bagian yang dipilih.

Secara teknis, dimisalkan kita memiliki himpunan semesta A yang berisi n elemen, dan terdapat sekumpulan kandidat himpunan bagian F={A_1, A_2, ..., A_n} di mana setiap $A_i \subseteq A$. Model *integer programming* akan memperkenalkan variabel x_i untuk setiap A_i, dengan x_i=1 jika Ai dipilih dan x_i=0 jika tidak dipilih. Setiap elemen A wajib tercover, yang diwujudkan dalam bentuk kendala, yaitu jumlah seluruh x_i untuk himpunan A_i yang mengandung elemen tersebut minimal harus satu. Jika setiap himpunan bagian memiliki bobot atau biaya c_i, maka tujuan optimisasi adalah meminimalkan:

$$f(x_1, \ldots, x_n) := \sum_{i=1}^{n} c_i \cdot x_i$$

Sebagai contoh untuk kasus sederhana set covering problem dapat diwakili dengan skenario Dimana kita memiliki masalah set covering dengan tujuh elemen yang harus dicakup, yaitu A={a,b,c,d,e,f,g}. Disediakan juga tujuh himpunan bagian kandidat, yaitu F={A_1,A_2,...,A_7} dengan rincian sebagai berikut:

A_1={a,b,f,g}

A_2={a,b,g}

A_3={a,b,c}

A_4={e,f,g}

A_5={f,g}

A_6={d,f}

A_7={d}

Pada kasus ini, beberapa himpunan kandidat merupakan subset dari himpunan lain (misal, A_2 dan A_5 merupakan subset dari A_1), namun hal tersebut tidak selalu menjamin efisiensi dalam penutupan seluruh elemen A. Jika kita ingin mencakup seluruh elemen pada A menggunakan jumlah himpunan paling sedikit, maka solusi optimalnya dapat berupa C_1*={A_3,A_4,A_6} atau C_2*={A_3,A_4,A_7} Kedua kombinasi ini hanya membutuhkan tiga himpunan untuk menutupi seluruh elemen A, dan jumlah ini lebih sedikit dibandingkan jika memilih A_1 sekaligus, yang walaupun lebih besar tetap memerlukan tambahan tiga himpunan lain sehingga totalnya empat.

Dalam pemodelan *integer programming*, contoh kasus SCP ini dapat direpresentasikan dengan sebuah matriks B. Matriks B berukuran m×n (dengan m adalah jumlah elemen yang harus dicakup, dan n adalah jumlah himpunan kandidat). Setiap kolom pada matriks B mewakili himpunan bagian Ai, sedangkan setiap baris merepresentasikan satu elemen dari himpunan semesta A. Matriks B untuk kasus ini adalah sebagai berikut:

$$B := \begin{pmatrix} 1110000 \\ 1110000 \\ 0010000 \\ 0000011 \\ 0001000 \\ 1001110 \\ 1101100 \end{pmatrix}$$

Persamaan tujuan (objektif) adalah meminimalkan jumlah himpunan yang dipilih:

$$\text{minimise} f(x_1, \ldots, x_7) := \sum_{i=1}^{7} x_i$$

Gambar 1. Rumus minimize contoh kasus

Sumber: https://www.sciencedirect.com/topics/computer-science/set-covering-problem

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &\geq 1 \\ x_1 + x_2 + x_3 &\geq 1 \\ x_3 &\geq 1 \\ x_6 + x_7 &\geq 1 \\ x_4 &\geq 1 \\ x_1 + x_4 + x_5 + x_6 &\geq 1 \\ x_1 + x_2 + x_4 + x_5 &\geq 1 \\ x_i \in \{0, 1\} \quad i &= 1, \ldots, 7 \end{aligned}$$

Gambar 2. Kendala contoh kasus

Sumber: https://www.sciencedirect.com/topics/computer-science/set-covering-problem

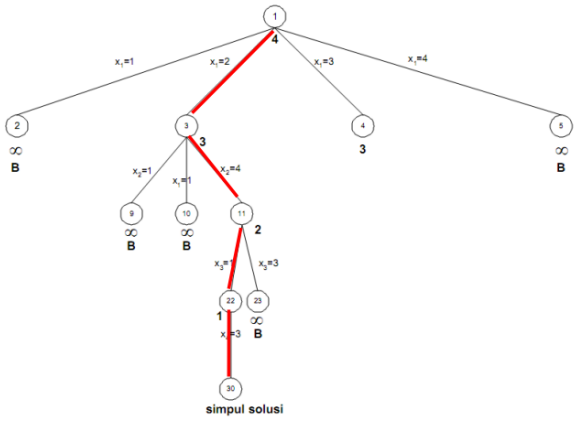
Pada contoh khusus, setiap baris kendala merepresentasikan bahwa satu elemen harus tercakup oleh minimal satu himpunan yang dipilih. Matriks ketetanggaan (covering matrix) yang digunakan pada model ini memungkinkan kita untuk memvisualisasikan relasi antara elemen dan himpunan bagian secara sistematis.

Meskipun pada solusi optimal mungkin terdapat elemen yang tercakup lebih dari satu kali, atau setiap elemen hanya tercakup sekali saja, kedua solusi dapat memiliki nilai fungsi tujuan yang sama dan dianggap sama-sama optimal. Hal ini karena yang menjadi fokus adalah jumlah minimum himpunan bagian yang digunakan untuk menutupi seluruh elemen, bukan seberapa sering sebuah elemen dicakup.

Set Covering Problem sendiri merupakan permasalahan optimasi kombinatorial yang NP-hard. Dalam kasus minimum set cover, tersedia algoritma aproksimasi yang bekerja dalam waktu polynomial dan dapat menjamin hasil yang maksimal hanya beberapa kali lipat lebih banyak dari solusi optimal, yakni dengan rasio paling buruk 1+log m, di mana m adalah jumlah elemen yang dicakup. Akan tetapi, telah terbukti secara teoritis bahwa tidak mungkin mendapatkan algoritma deterministik dengan rasio aproksimasi konstan untuk masalah ini, kecuali jika P = NP.

B. Algoritma Branch and Bound

Algoritma Branch and Bound (B&B) adalah metode algoritmik yang digunakan untuk menyelesaikan masalah optimisasi, yaitu mencari solusi yang memaksimalkan atau meminimalkan suatu fungsi objektif dengan batasan tertentu. Metode ini menggabungkan teknik pencarian Breadth-First Search (BFS) dengan strategi least cost search, yaitu pencarian yang selalu memilih simpul dengan nilai estimasi biaya (cost) paling kecil sebagai simpul berikutnya yang akan diekspansi. Dalam algoritma ini, setiap simpul dalam ruang pencarian (state space) diberikan sebuah nilai estimasi atau cost yang mencerminkan perkiraan jarak dari simpul tersebut menuju simpul tujuan (goal node) atau solusi terbaik yang mungkin diperoleh melalui simpul tersebut.



Gambar 3. Pembentukan Pohon Ruang Status 4-Ratu dengan Algoritma Branch & Bound

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/17-Algoritma-Branch-and-Bound-(2025)-Bagian1.pdf

Branch and Bound bekerja dengan membentuk pohon ruang status secara dinamis untuk menjelajahi berbagai kemungkinan solusi. Algoritma ini menerapkan prinsip pencabangan (branching), di mana persoalan besar dibagi menjadi subpersoalan yang lebih kecil, serta prinsip pembatasan (bounding), yang digunakan untuk menghentikan eksplorasi subpersoalan yang dipastikan tidak akan memberikan solusi lebih baik dari solusi terbaik yang telah ditemukan sebelumnya. Proses pencabangan dilakukan secara sistematis, sedangkan proses pembatasan memerlukan fungsi heuristik untuk memperkirakan batas bawah (lower bound) atau batas atas (upper bound) pada solusi optimal yang dapat dicapai dari subpersoalan tersebut.

Dalam algoritma ini, sebuah simpul dikatakan sebagai "simpul hidup" jika simpul tersebut masih berpotensi menghasilkan solusi yang optimal. Sebaliknya, simpul yang dianggap tidak lagi berpotensi memberikan solusi lebih baik akan "dibunuh" atau tidak dieksplorasi lebih lanjut. Proses pencarian dalam Branch and Bound dilakukan dengan memilih simpul hidup yang memiliki cost terkecil untuk diekspansi, sehingga memastikan bahwa pencarian selalu mengarah ke solusi paling menjanjikan terlebih dahulu (best-first rule).

Fungsi pembatas (bounding function) merupakan komponen penting dalam Branch and Bound yang menentukan apakah sebuah simpul layak dilanjutkan atau tidak. Fungsi ini secara heuristik menghitung nilai cost untuk setiap simpul berdasarkan estimasi ongkos termurah dari simpul tersebut ke simpul tujuan. Apabila nilai cost pada simpul tertentu sudah melebihi nilai cost dari solusi terbaik yang ditemukan sejauh ini, maka simpul tersebut akan langsung dipangkas dari pohon pencarian, karena tidak mungkin memberikan solusi yang lebih optimal dibandingkan dengan solusi yang sudah ada.

Secara umum, algoritma Branch and Bound mengikuti langkah-langkah berikut:

1. Mulailah dengan memasukkan simpul akar ke dalam antrian *Q*. Jika simpul tersebut sudah merupakan simpul tujuan (goal node), maka solusi telah ditemukan dan pencarian dapat dihentikan, terutama jika hanya satu solusi yang dicari.
2. Jika antrian *Q* kosong, maka proses pencarian berakhir karena tidak ada lagi simpul yang dapat dieksplorasi.
3. Jika *Q* masih memiliki isi, pilih simpul i dari antrian tersebut yang memiliki nilai estimasi biaya $\hat{c}(i)$ paling rendah. Apabila terdapat lebih dari satu simpul dengan nilai biaya sama rendahnya, pilih salah satu secara acak.
4. Apabila simpul *i* yang terpilih adalah simpul tujuan, maka solusi telah ditemukan. Jika hanya satu solusi yang dibutuhkan, maka proses berhenti. Dalam konteks optimasi dengan pendekatan least cost search, perlu dilakukan pemeriksaan terhadap semua simpul hidup: jika ada simpul dengan cost lebih tinggi daripada solusi sementara yang ditemukan, maka simpul tersebut dapat dieliminasi (pruned).
5. Jika simpul *i* belum mencapai solusi, maka lanjutkan dengan membangkitkan seluruh anak dari simpul tersebut. Jika simpul *i* tidak memiliki anak, lanjutkan kembali ke langkah kedua.
6. Untuk setiap simpul anak *j* yang dihasilkan, hitung nilai $\hat{c}(j)$ dan tambahkan ke dalam antrian *Q*.
7. Ulangi kembali proses dari langkah kedua.

Pendekatan ini telah banyak digunakan untuk menyelesaikan berbagai masalah optimisasi, seperti Traveling Salesperson Problem (TSP), masalah penugasan (Assignment Problem), dan persoalan Knapsack, serta dapat diaplikasikan dalam masalah Set Covering seperti penempatan kamera untuk cakupan pengawasan minimum.

III. Implementasi dan Pengujian

Untuk membuktikan efektivitas metode *Branch and Bound* dalam menyelesaikan permasalahan penempatan kamera pengawasan, dilakukan implementasi algoritma dan serangkaian eksperimen untuk mengamati kinerja algoritma terhadap berbagai skenario uji. Implementasi difokuskan pada proses pencarian solusi minimum dari himpunan titik kamera yang mampu mencakup seluruh titik pengawasan dalam radius tertentu. Pada penelitian ini tidak hanya mengadaptasi prinsip *least-cost search* dari algoritma Branch and Bound, tetapi juga mengembangkan fungsi estimasi *lower bound* berbasis heuristik greedy untuk mempercepat proses pruning. Seluruh pengujian dilakukan secara sistematis dengan konfigurasi input yang bervariasi, serta dilengkapi dengan visualisasi cakupan untuk memastikan validitas solusi. Bagian berikut akan menguraikan bagaimana desain solusi dirancang, bagaimana algoritma diimplementasikan, dan bagaimana pengujian dilakukan secara terstruktur.

A. Desain Solusi

Permasalahan penempatan kamera pengawasan dimodelkan sebagai persoalan *Set Covering Problem*, yaitu mencari himpunan minimum dari titik-titik kandidat kamera yang cakupannya mencakup seluruh titik pengawasan yang ada. Setiap kamera memiliki radius pengamatan tertentu, sehingga sebuah titik pengawasan dianggap tercakup apabila jarak Euclidean antara titik tersebut dan kamera tidak melebihi radius yang ditentukan.

Solusi dinyatakan sebagai subset dari titik kandidat kamera yang dipilih. Dalam konteks ini, setiap titik kandidat dipertimbangkan sebagai simpul dalam pohon pencarian solusi. Simpul awal (root node) tidak memiliki kamera dan tidak mencakup titik apa pun. Pada setiap langkah, satu titik kamera dipilih untuk ditambahkan ke dalam solusi, dan cakupan diperluas dengan menambahkan himpunan titik yang tercakup oleh kamera tersebut. Proses ini terus berlangsung hingga seluruh titik pengawasan telah tercakup.

Untuk mengefisienkan pencarian, digunakan metode *Branch and Bound* dengan strategi eksplorasi berdasarkan prinsip *least-cost search*. Setiap simpul dalam ruang solusi dievaluasi menggunakan fungsi biaya total sebagai berikut:

$$f(n) = g(n) + h(n)$$

di mana $g(n)$ adalah jumlah kamera yang telah dipilih sejauh ini, dan $h(n)$ adalah estimasi jumlah kamera tambahan minimum yang dibutuhkan untuk mencakup titik-titik yang belum tercakup. Estimasi $h(n)$ dihitung menggunakan pendekatan greedy, yaitu dengan secara iteratif memilih kamera yang mencakup jumlah titik maksimum yang belum tercakup hingga semua titik terliput. Pendekatan ini memberikan estimasi batas bawah (*lower bound*) yang cukup ketat tanpa membebani kinerja.

Untuk menjamin efisiensi pencarian, digunakan struktur data *priority queue* agar setiap simpul yang diekspansi memiliki nilai $f(n)$ paling kecil. Selain itu, dilakukan pemangkasan (*pruning*) terhadap simpul yang nilai $f(n)$ lebih besar atau sama dengan solusi terbaik sementara yang telah ditemukan. Dengan strategi ini, algoritma akan menghindari eksplorasi terhadap cabang solusi yang tidak menjanjikan.

B. Implementasi Algoritma Branch and Bound

Implementasi algoritma *Branch and Bound* dalam konteks penempatan kamera dilakukan dengan membentuk pohon pencarian solusi, di mana setiap simpul mewakili solusi parsial berupa daftar titik kamera yang telah dipilih dan cakupan titik pengawasan yang telah tercapai. Proses eksplorasi dilakukan secara sistematis dengan menggunakan antrian prioritas (*priority queue*) untuk memastikan bahwa simpul dengan estimasi total biaya terendah diekspansi terlebih dahulu.

Setiap simpul terdiri dari empat elemen: daftar titik kamera yang telah dipilih ($\boldsymbol{C}$), himpunan titik pengawasan yang telah tercakup (P), indeks level eksplorasi yang menunjukkan posisi terakhir kandidat kamera yang digunakan, dan nilai prioritas simpul. Nilai prioritas dihitung sebagai jumlah kamera saat ini ($\boldsymbol{g(n)}$) ditambah dengan estimasi jumlah kamera tambahan minimum yang dibutuhkan untuk mencakup titik-titik yang belum terliput ($\boldsymbol{h(n)}$) Estimasi ini dihitung dengan metode greedy, yaitu dengan memilih secara berurutan titik kamera yang memberikan cakupan maksimum terhadap titik yang belum terliput hingga semua titik tercakup.

Simpul awal (akar) dari pencarian berisi solusi kosong dan tidak mencakup titik mana pun. Selanjutnya, pada setiap iterasi, simpul dengan nilai $\boldsymbol{f(n)}$ terkecil diambil dari antrian prioritas. Jika simpul tersebut telah mencakup seluruh titik pengawasan, maka simpul dianggap sebagai solusi sementara. Jika belum, dilakukan perluasan terhadap simpul dengan menambahkan satu titik kamera dari indeks kandidat berikutnya. Untuk setiap simpul hasil perluasan, kembali dihitung nilai estimasi total biaya dan dimasukkan ke dalam antrian jika berpotensi menghasilkan solusi lebih baik.

Proses ini akan terus berlangsung hingga antrian prioritas kosong, atau seluruh simpul yang berpotensi menghasilkan solusi optimal telah dievaluasi atau dipangkas. Strategi pruning dilakukan dengan membandingkan nilai total biaya $\boldsymbol{f(n)}$ suatu simpul dengan jumlah kamera dari solusi terbaik sementara. Apabila nilai $\boldsymbol{f(n)}$ lebih besar atau sama, maka simpul tidak diekspansi lebih lanjut.

Sebagai contoh diberikan sebuah kasus sederhana yang dimisalkan terdapat 5 titik pengawasan pada koordinat berikut:

P0 = (0, 0)
P1 = (0, 1)
P2 = (1, 1)
P3 = (2, 0)
P4 = (2, 1)

Dengan radius kamera adalah 1 satuan.

Langkah Konkret yang dilakukan untuk penyelesaian contoh kasus sederhana adalah sebagai berikut:

(1) Inisialisasi Awal (Root Node):
Pada tahap awal, belum ada kamera yang terpasang sehingga daftar kamera adalah [], dan himpunan titik yang tercakup masih kosong ({}). Estimasi jumlah kamera tambahan yang dibutuhkan (h(n)) dihitung menggunakan metode *greedy*. Dalam hal ini, metode *greedy* bekerja dengan memilih titik kamera yang mampu mencakup jumlah titik belum tercakup (*uncovered*) paling banyak, lalu mengurangi titik-titik yang sudah tercakup dari himpunan. Proses ini diulang hingga seluruh titik tercover. Dalam kasus ini, diperoleh estimasi h(n) = 2. Maka, prioritas simpul awal (f(n)) adalah 0 (jumlah kamera) + 2 (estimasi) = 2. Simpul ini kemudian dimasukkan ke dalam antrian prioritas untuk dieksplorasi lebih lanjut.

(2) Perluasan Node Pertama:
Algoritma mulai mengeksplorasi kemungkinan penempatan kamera di berbagai titik pengawasan. Sebagai contoh, kamera dipasang di P0 (0,0), sehingga daftar

kamera menjadi [0] dan titik yang tercakup adalah {0, 1, 2} karena P0, P1, dan P2 berada dalam radius cakupan. Estimasi jumlah kamera tambahan dihitung kembali dengan pendekatan *greedy*, yaitu dengan memilih titik kamera berikutnya yang mampu mencakup sisa titik terbanyak (dalam hal ini, P3 dan P4 belum tercakup dan dapat dijangkau oleh P3), sehingga estimasi h(n) = 1. Maka, nilai f(n) untuk simpul ini adalah 1 (kamera saat ini) + 1 (estimasi) = 2. Simpul [0] kemudian dimasukkan ke dalam antrian prioritas.

(3) Ekspansi Node dengan Prioritas Terendah:

Simpul [0] yang memiliki nilai f(n) = 2 diambil dari antrian untuk diekspansi. Salah satu ekspansinya adalah menambahkan kamera di titik P3 (2,0), sehingga daftar kamera menjadi [0, 3] dan semua titik pengawasan {0, 1, 2, 3, 4} kini tercakup. Karena semua titik sudah tercakup, estimasi h(n) = 0, dan f(n) = 2 + 0 = 2. Simpul [0, 3] ini kemudian dicatat sebagai solusi terbaik sementara.

(4) Pruning Cabang Tidak Efisien:

Simpul-simpul lain yang belum dieksplorasi akan dievaluasi nilai f(n)-nya. Apabila nilai tersebut lebih besar atau sama dengan solusi terbaik sementara (dalam hal ini 2), simpul akan dipangkas (*pruned*) karena tidak mungkin menghasilkan solusi yang lebih baik. Sebagai contoh, simpul [1] yang hanya mencakup sebagian titik dan memiliki estimasi tinggi akan menghasilkan f(n) = 1 + 2 = 3, sehingga dipangkas. Demikian pula simpul [2] yang hanya mencakup P1 dan P2 akan menghasilkan f(n) > 2 dan juga dipangkas.

Sehingga didapatkan bahwa penempatan kamera pada titik P0 (0,0) dan P3 (2,0) merupakan solusi optimal yang mencakup seluruh titik pengawasan dengan jumlah kamera minimum, yaitu sebanyak 2 kamera.

C. *Rancangan Pengujian*

Pengujian terhadap algoritma Branch and Bound pada permasalahan penempatan kamera dilakukan untuk mengevaluasi efektivitas dan efisiensi algoritma dalam berbagai kondisi kompleksitas permasalahan. Pengujian ini dirancang untuk mengukur kemampuan algoritma dalam menemukan solusi optimal secara tepat waktu dan dengan jumlah kamera seminimal mungkin.

Pengujian dilakukan dengan memvariasikan jumlah titik pengawasan, yaitu sebanyak 5 hingga 16 titik, radius cakupan kamera, dengan nilai radius sebesar 1 hingga 5 satuan Euclidean, serta variasi terhadap distribusi pengawasan. Distribusi ini mencakup tiga pola utama, yaitu:

(1) pola grid seperti persegi 3×3 dan 4×4

(2) pola acak, yang meniru sebaran tidak teratur sebagaimana kondisi nyata di lapangan, serta

(3) pola spiral, yang mensimulasikan kasus dengan distribusi lebih kompleks dan melingkar.

Lingkungan pengujian dilakukan pada sistem operasi Windows dan dengan bahasa pemrograman yang digunakan adalah Python 3, dengan library tambahan seperti heapq untuk antrian prioritas, math untuk perhitungan Euclidean, random untuk distribusi titik acak, serta matplotlib.pyplot untuk keperluan visualisasi hasil eksperimen.

Matplotlib.pyplot digunakan untuk memvisualisasikan hasil pengujian yang menampilkan titik-titik pengawasan sebagai titik biru dan area cakupan kamera dalam bentuk lingkaran hijau. Visualisasi ini membantu memverifikasi bahwa seluruh titik telah tercakup oleh kamera yang dipasang.

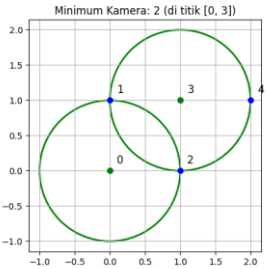
D. *Pengujian*

Berdasarkan rancangan pengujian, dilakukan tujuh uji coba pada berbagai konfigurasi jumlah titik, radius kamera, dan pola distribusi. Setiap uji menghasilkan data berupa jumlah kamera minimum yang dibutuhkan untuk mencakup seluruh titik pengawasan serta waktu eksekusi algoritma dalam satuan detik. Berikut adalah hasil eksperimen yang diperoleh:

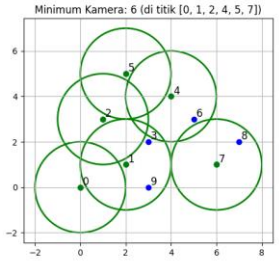
No	Jumlah Titik	Radius Kamera	Pola Distribusi	Kamera Minimum	Waktu Eksekusi (detik)
1	5	1	grid	2	0.0097
2	10	2	acak	6	0.0130
3	15	3	acak	5	0.0128
4	9	1	grid	3	0.0003
5	12	2	grid	3	0.0004
6	16	1	grid	4	0.0007
7	14	5	spiral	2	0.0003

Tabel 1. Hasil pengujian

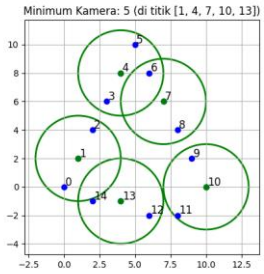
berikut disajikan hasil visualisasi dari pengujian. Pada gambar, titik biru merepresentasikan lokasi titik pengawasan, sedangkan lingkaran hijau menunjukkan cakupan dari kamera yang dipilih oleh algoritma. Visualisasi ini mengonfirmasi bahwa seluruh titik pengawasan berhasil dicakup oleh area jangkauan kamera secara optimal.



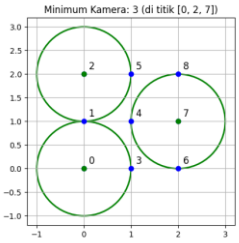
Gambar 4. Visualisasi pengujian 1



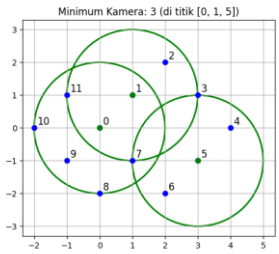
Gambar 5. Visualisasi pengujian 2



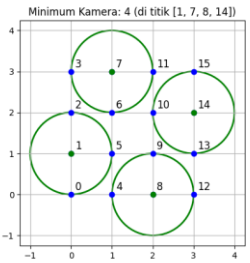
Gambar 6. Visualisasi pengujian 3



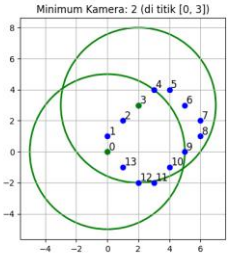
Gambar 7. Visualisasi pengujian 4



Gambar 8. Visualisasi pengujian 5



Gambar 9. Visualisasi pengujian 6



Gambar 10. Visualisasi pengujian 7

IV. Analisis dan Kesimpulan

Pengujian yang dilakukan pada tujuh skenario berbeda yang terdiri dari variasi jumlah titik, radius cakupan kamera, dan pola distribusi titik pengawasan menghasilkan bahwa jumlah kamera minimum sangat dipengaruhi oleh besar kecilnya radius cakupan kamera. Semakin besar radius yang digunakan, maka jumlah kamera yang dibutuhkan untuk mencakup seluruh titik pengawasan akan semakin sedikit. Hal ini tampak jelas pada kasus ke-7, di mana dengan 14 titik dan radius sebesar 5, hanya diperlukan dua kamera untuk mencakup seluruh area. Sebaliknya, pada radius yang kecil (misalnya 1), jumlah kamera yang dibutuhkan meningkat, seperti terlihat pada kasus ke-6 dengan 16 titik dan radius 1 yang memerlukan empat kamera.

Selain radius, pola distribusi titik juga mempengaruhi efisiensi penempatan kamera. Pola distribusi grid cenderung menghasilkan hasil yang lebih optimal karena keteraturan letak titik-titik pengawasan yang memungkinkan cakupan kamera lebih merata. Hal ini dapat dilihat pada kasus-kasus dengan pola grid (kasus 1, 4, 5, dan 6), yang menunjukkan bahwa jumlah kamera minimum relatif kecil dan waktu eksekusi sangat cepat. Sebaliknya, pola acak menyebabkan beberapa titik tersebar tidak merata, sehingga memaksa sistem untuk menempatkan kamera lebih banyak demi menjangkau semua titik, seperti yang terjadi pada kasus ke-2 dan ke-3.

Dari segi efisiensi waktu, algoritma Branch and Bound menunjukkan performa yang sangat baik, dengan seluruh kasus berhasil diselesaikan dalam waktu kurang dari 0,015 detik. Bahkan untuk kasus yang melibatkan 15 hingga 16 titik, waktu eksekusinya tetap berada di bawah ambang batas 0,013 detik. Hal ini menunjukkan bahwa algoritma yang digunakan mampu menyelesaikan permasalahan penempatan kamera secara optimal dalam waktu yang sangat singkat.

Berdasarkan analisis ini, dapat disimpulkan bahwa algoritma Branch and Bound sangat efektif untuk menyelesaikan masalah penempatan kamera pengawasan. Dengan penyesuaian parameter seperti radius kamera dan pengaturan distribusi titik, sistem dapat dengan cepat menemukan solusi optimal dengan jumlah kamera minimum yang dibutuhkan. Visualisasi yang menyertai pengujian juga memperkuat bukti bahwa setiap solusi memang mencakup seluruh titik secara menyeluruh.

Ucapan Terimakasih

Penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada Bapak Monterico Adrian dan Bapak Rinaldi Munir selaku dosen pengampu mata kuliah Strategi Algoritma.

Bimbingan, arahan, dan materi yang telah disampaikan selama satu semester ini sangat membantu dalam memahami konsep-konsep yang ada pada mata kuliah strategi algoritma, khususnya dalam penyusunan makalah ini. Penulis juga mengucapkan terima kasih kepada seluruh pihak yang telah memberikan dukungan selama proses penyusunan makalah ini berlangsung. Harapan penulis, makalah ini dapat memberikan kontribusi yang bermanfaat bagi pembaca dalam memahami penerapan algoritma Branch and Bound, khususnya pada kasus optimasi penempatan kamera pengawasan. Semoga tulisan ini tidak hanya menjadi bentuk pemenuhan tugas akademik, tetapi juga dapat menjadi rujukan awal yang inspiratif untuk pengembangan lebih lanjut di bidang ilmu komputer dan teknik algoritma. Saran dan kritik yang membangun sangat penulis harapkan guna penyempurnaan di masa yang akan datang.

REFERENCES

[1] [1] H. H. Hoos and T. Stützle, "Other combinatorial problems," in *Stochastic Local Search*, The Morgan Kaufmann Series in Artificial Intelligence, San Francisco, CA, USA: Morgan Kaufmann, 2005, pp. 467–525. doi: 10.1016/B978-155860872-6/50027-5.

[2] R. Munir, N. U. Maulidevi, dan M. L. Khodra, "Algoritma Branch & Bound (Bagian 1)", Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2025. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/17-Algoritma-Branch-and-Bound-(2025)-Bagian1.pdf

[3] R. Munir, N. U. Maulidevi, dan M. L. Khodra, "Algoritma Branch & Bound (Bagian 2)", Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2025. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/17-Algoritma-Branch-and-Bound-(2025)-Bagian2.pdf

[4] R. Munir, N. U. Maulidevi, dan M. L. Khodra, "Algoritma Branch & Bound (Bagian 3)", Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2025. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/17-Algoritma-Branch-and-Bound-(2025)-Bagian3.pdf

[5] R. Munir, N. U. Maulidevi, dan M. L. Khodra, "Algoritma Branch & Bound (Bagian 4)", Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2025. [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/17-Algoritma-Branch-and-Bound-(2025)-Bagian4.pdf

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, 24 Juni 2025

Muhammad Farrel Wibowo 13523153