

BAB II TINJAUAN PUSTAKA

A. Penelitian Terkait

Penelitian ini merujuk pada penelitian sebelumnya yang dijadikan acuan dalam penelitian ini. Persamaan dan perbedaan dapat dilihat sebagai berikut:

Tabel 1. Daftar penelitian terkait

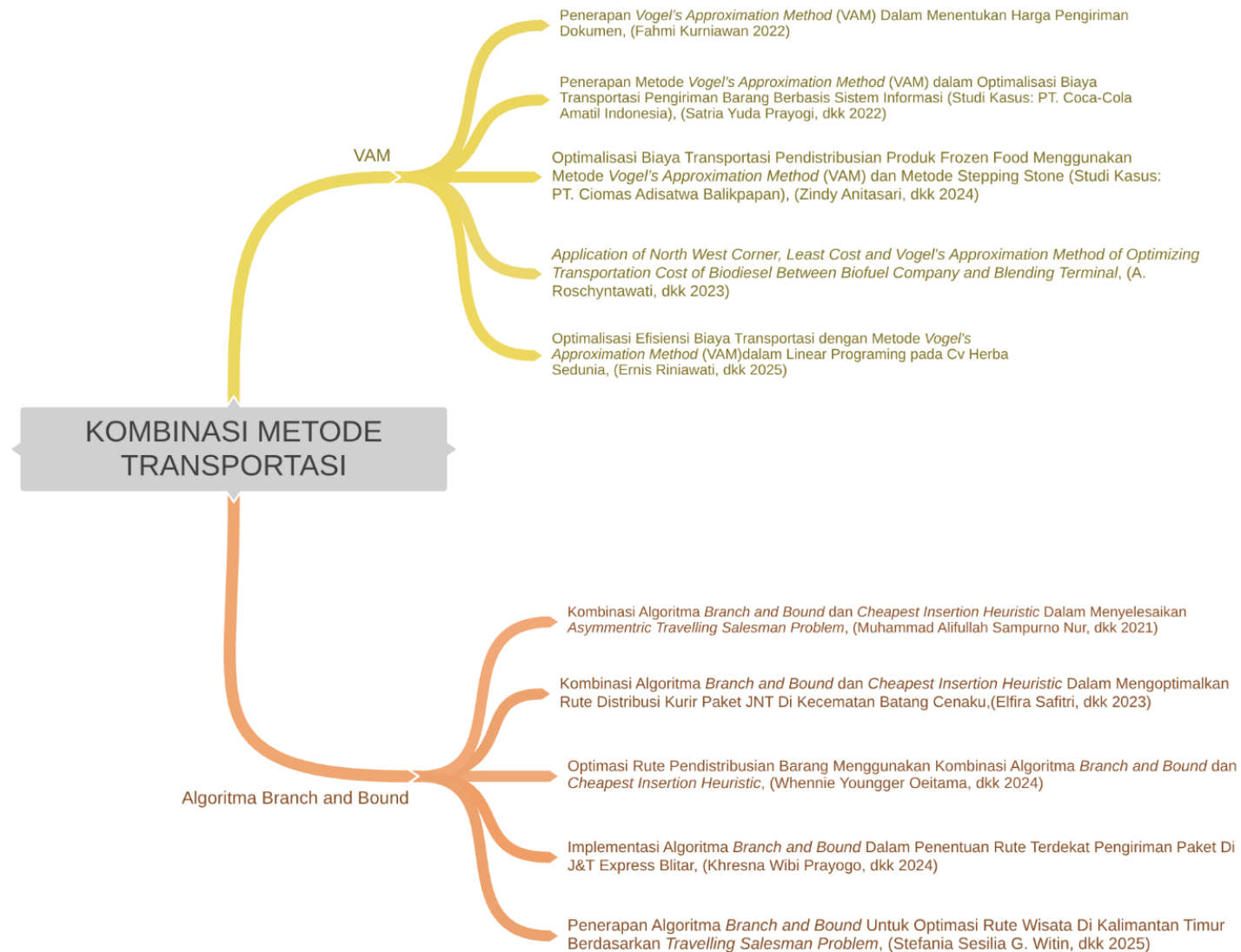
No	Peneliti	Keterangan Penelitian
(1)	(2)	(3)
1	Fahmi Kurniawan (2022) [2]	Judul: Penerapan <i>Vogel's Approximation Method</i> (VAM) Dalam Menentukan Harga Pengiriman Dokumen Metode: <i>Vogel's Approximation Method</i> (VAM). Hasil Penelitian: Penerapan VAM dapat membantu perusahaan dalam menghitung biaya pengiriman dengan lebih akurat dan efisien, sehingga dapat meningkatkan daya saing perusahaan.
2	Satria Yuda Prayogi & Muhammad Iqbal Panjaitan (2022) [5]	Judul: Optimalisasi Biaya Transportasi Pengiriman Barang Berbasis Sistem Informasi di PT. Coca-Cola Amatil Indonesia. Metode: VAM. Hasil Penelitian: Penggunaan metode VAM dan MODI menghasilkan penghematan yang signifikan dengan biaya transportasi yang optimal sebesar Rp. 2.719.900,00.
3	Zindy Anitasari, Wasono Wasono & Fidya Deny Tisna Amijaya (2024) [1]	Judul: Optimalisasi Distribusi Produk <i>Frozen Food</i> di PT. Ciomas Adisatwa Balikpapan. Metode: NWC, <i>Least Cost</i> , VAM, dan <i>Stepping Stone</i> . Hasil Penelitian: Biaya dengan NWC sebesar Rp. 9.211.000,00, <i>Least Cost</i> sebesar Rp. 10.436.400,00 dan VAM sebesar Rp. 6.249.000,00. Uji optimalitas dengan VAM dan <i>Stepping Stone</i> menurunkan biaya

No	Peneliti	Keterangan Penelitian
(1)	(2)	(3)
		menjadi Rp. 6.098.200,00, Lebih hemat dibanding VAM saja.
4	A. Roschyntawati et al. (2023) [12]	Judul: <i>Application of North West Corner, Least Cost and Vogel's Approximation Method for Optimizing Transportation Cost of Biodiesel.</i> Metode: NWC, Least Cost, dan VAM. Hasil Penelitian: Kombinasi metode menghasilkan penghematan biaya sebesar 31,05%.
5	Whennie Youngger, Oeitama et al. (2024) [13]	Judul: Optimasi Rute Pendistribusian Barang Dengan Kombinasi Algoritma <i>Branch and Bound</i> serta <i>Cheapest Insertion Heuristic</i>. Metode: Algoritma <i>Branch and Bound</i> dan <i>Cheapest Insertion Heuristic</i>. Hasil Penelitian: Rute optimal untuk pendistribusian barang di kantor pos Palembang diperoleh jarak sepanjang 24,3 km dengan efisiensi lebih tinggi dibanding penggunaan <i>Branch and Bound</i> saja.
6	Muhammad Alifullah Sampurno Nur & Budi Rahadjeng (2021) [14]	Judul: Kombinasi Algoritma <i>Branch and Bound</i> serta <i>Cheapest Insertion Heuristic</i> pada <i>Asymmentric Travelling Salesman Problem</i>. Metode: Algoritma <i>Branch and Bound</i> & <i>Cheapest Insertion Heuristic</i>. Hasil Penelitian: Rute distribusi diperoleh dengan total jarak 54,9 km.
7	Elfira Safitri et al. (2023) [15]	Judul: Optimasi Rute Distribusi Kurir JNT dengan Algoritma <i>Branch and Bound</i> serta <i>Cheapest Insertion Heuristic</i>. Metode: Algoritma <i>Branch and Bound</i> & <i>Cheapest Insertion Heuristic</i>. Hasil Penelitian: Rute distribusi di Kecamatan Batang Cenaku diperoleh dengan total jarak 172 km.
8	Khresna Wibi Prayogo et al. (2024) [8]	Judul: Implementasi Algoritma <i>Branch and Bound</i> untuk Penentuan Rute J&T Express Blitar.

No	Peneliti	Keterangan Penelitian
(1)	(2)	(3)
		Metode: Algoritma <i>Branch and Bound</i>. Hasil Penelitian: Rute yang dihasilkan oleh Algoritma <i>Branch and Bound</i> dengan total jarak 8,360 m, menghasilkan penghematan selisih jarak 1.760 m (16,78%) pada trip 1 dan 542 m (6,09%) pada trip 2 dibanding pilihan kurir J&T Exspress.
9	Ernis Riniawati et al. (2025) [6]	Judul: Efisiensi Biaya Transporrtasi dengan VAM di Cv Herba Sedunia. Metode: VAM. Hasil Penelitian: Distribusi 1.250 unit produk menghasilkan biaya Rp 18.700.000,00. VAM terbukti dapat mengurangi biaya operasional dan memberikan fleksibilitas strategis distribusi.
10	Stefania Sesilia G. Witin et al. (2025) [16]	Judul: Penerapan <i>Branch and Bound</i> untuk Optimasi Rute Wisata di Kalimantan Timur. Metode: <i>Branch And Bound</i>. Hasil Penelitian: Rute terpendek adalah Bandara Sepinggan hingga Bukit Bingkirai dengan total jarak tempuh 148,6 km.

Sumber: Buku Panduan Penulisan Karya Ilmiah 2023

Diagram *mind map* dicontohkan pada gambar berikut ini:



Gambar 1. *Mind map* penelitian terkait

B. Landasan Teori

1. Optimasi

Optimalisasi adalah suatu proses pengoptimalan untuk mencari solusi terbaik. Tujuan dari optimalisasi ini adalah mencari rute. Dengan meminimalkan jarak tempuh, waktu yang digunakan akan menjadi lebih baik singkat dan tidak terjadi pemborosan biaya [17].

2. Masalah Transportasi

Masalah transportasi umumnya diselesaikan dengan beberapa metode untuk menghasilkan solusi awal yang optimal. Beberapa metode populer adalah *North West Corner* (NWC), *Least Cost Method* (LCM), dan *Vogel's Approximation Method* (VAM) [3].

- a. NWC merupakan metode sederhana yang memulai alokasi dari pojok kiri atas (barat laut) pada tabel distribusi tanpa mempertimbangkan biaya transportasi, sehingga meskipun sederhana dan mudah diterapkan, hasilnya kurang efisien dan tidak optimal.
- b. LCM yakni dengan memilih alokasi berdasarkan biaya transportasi terendah terlebih dahulu, sehingga memberikan hasil yang lebih baik dari NWC, tetapi tidak selalu menjamin solusi optimal, terutama saat terdapat biaya minimum yang sama.
- c. VAM menggunakan konsep biaya penalti yang merupakan selisih antara dua biaya transportasi terkecil dalam satu baris atau kolom, kemudian mengalokasikan alokasi berdasarkan penalti tertinggi sehingga solusi awal yang dihasilkan lebih mendekati optimal atau bahkan optimal.

Secara umum, masalah transportasi adalah proses pengiriman barang dari beberapa sumber ke tujuan secara optimal dengan mempertimbangkan biaya minimal, pengalokasian yang tepat, dan rute pengiriman yang efektif [18]. Dalam masalah transportasi klasik adalah menentukan jadwal optimal dari pengiriman yang [4]:

- a. Berasal dari sumber yang memiliki jumlah barang tertentu yang tersedia.
- b. Barang dialokasikan dan dikirim langsung ke masing-masing tujuan di mana jumlah total yang diterima harus sama dengan jumlah total yang dibutuhkan.
- c. *Supply* dan *demand* harus dihabiskan. Oleh karena itu, total permintaan yang diberikan harus sama dengan total persediaan yang diberikan.
- d. Biaya setiap pengiriman dari sumber ke tujuan dihitung berdasarkan kapasitas gudang dan total biaya alokasi adalah jumlah biaya dari semua biaya.

Masalah transportasi dapat disederhanakan m adalah jumlah barang yang harus dikirim ke tujuan n untuk memenuhi permintaan. Khususnya, sumber i memuat barang berjumlah a_i , dan tujuan j membutuhkan barang sebesar b_j . Diasumsikan bahwa sistem tersebut seimbang, artinya total pemasok sama dengan total permintaan [3]. Masalah ini dapat dirumuskan sebagai berikut:

$$\sum_{i=1}^m a_i = \sum_{j=1}^n b_j \quad (1)$$

Sehingga secara model matematis diperoleh [1],

$$\text{Minimumkan } Z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (2)$$

Dengan kendala:

$$\sum_{j=1}^n x_{ij} = a_i, \quad i = 1, 2, 3, \dots, m \quad (3)$$

$$\sum_{i=1}^m x_{ij} = b_j, \quad j = 1, 2, 3, \dots, n \quad (4)$$

$$x_{ij} \geq 0 \quad (i = 1, 2, 3, \dots, m \text{ dan } j = 1, 2, 3, \dots, n)$$

Keterangan:

- a_i : Banyaknya barang yang tersedia di tempat asal ke- i ,
- b_j : Banyaknya permintaan barang di tempat tujuan ke- j ,
- c_{ij} : Biaya distribusi per satuan barang dari sumber i ke tujuan j ,
- x_{ij} : Satuan barang yang akan didistribusikan dari sumber i ke tujuan j ,
- m : Jumlah tempat asal,
- n : Jumlah tempat tujuan, dan
- Z : Biaya total transportasi.

3. *Vogel's Approximation Method* (VAM)

Vogel's Approximation Method (VAM) merupakan salah satu metode transportasi yang populer karena pada penerapannya menghasilkan solusi mendekati optimal atau bahkan optimal [4]. Berikut ini terdapat beberapa asumsi yang dimiliki Algoritma *Vogel's Approximation Method* (VAM) yaitu:

- 1) Ada penawaran dan permintaan.
- 2) Jumlah permintaan dan penawaran sama.
- 3) Terdapat Gudang dan tujuan.
- 4) Adanya biaya distribusi.

Berikut langkah-langkah VAM [3]:

- a) Hitung selisih C_{ij} dengan biaya terkecil pada seluruh baris atau kolom.
- b) Tentukan penalti C_{ij} yang selisihnya terbesar. Jika terdapat nilai terbesar lebih dari satu, maka pilihlah sembarang yang dapat memindahkan barang paling banyak.
- c) Alokasikan sebanyak mungkin pada sel yang mempunyai biaya terkecil pada C_{ij} yang dipilih. Jumlah alokasi adalah sebesar *supply* atau *demand* yang mana lebih kecil.
- d) Jika alokasi telah memenuhi *supply* atau *demand*, maka hapus C_{ij} yang sudah memenuhi syarat sebelumnya.
- e) Ulangi langkah (a) sampai (d) hingga semua alokasi terpenuhi.

Berikut contoh implementasi VAM [19],

Tabel 2. Biaya distribusi

Gudang	Agen					Supply
	1	2	3	4	5	
A	100	150	200	140	35	400
B	50	70	60	65	80	200
C	40	90	100	150	130	150
Demand	100	200	150	160	140	

Penyelesaian:

Tabel 3. Iterasi 1 Algoritma VAM

Gudang	Agen					Supply	
	1	2	3	4	5		
A	100	150	200	140	35	400	100 - 35 = 65
B	50	70	60	65	80	200 - 160 = 40	60 - 50 = 10
C	40	90	100	150	130	150	90 - 40 = 50
Demand	100	200	150	160 - 160 = 0	140		
50 - 40 = 10 90 - 70 = 20 100 - 60 = 40 140 - 65 = 75 80 - 35 = 45							

Tabel 4. Iterasi 2 Algoritma VAM

Gudang	Agen					Supply
	1	2	3	4	5	
A	100	150	200	140	35	400
B	50	70	60	65	80	40
C	40	90	100	150	130	150
Demand	100	200	150	0	140	

$100 - 35 = 65$
 $60 - 50 = 10$
 $90 - 40 = 50$
 $50 - 40 = 10$
 $90 - 70 = 20$
 $100 - 60 = 40$
 $80 - 35 = 45$

Tabel 5. Iterasi 3 Algoritma VAM

Gudang	Agen					Supply
	1	2	3	4	5	
A	100	150	200	140	35	400 - 140 = 260
B	50	70	60	65	80	40
C	40	90	100	150	130	150
Demand	100	200	150	0	140 - 140 = 0	

$100 - 35 = 65$
 $60 - 50 = 10$
 $90 - 40 = 50$
 $50 - 40 = 10$
 $90 - 70 = 20$
 $100 - 60 = 40$
 $80 - 35 = 45$

Tabel 6. Iterasi 4 Algoritma VAM

Gudang	Agen					Supply
	1	2	3	4	5	
A	100	150	200	140	35	260
B	50	70	60	65	80	40
C	40	90	100	150	130	150
Demand	100	200	150	0	0	

$150 - 100 = 50$
 $60 - 50 = 10$
 $90 - 40 = 50$
 $50 - 40 = 10$
 $90 - 70 = 20$
 $100 - 60 = 40$

Tabel 7. Iterasi 5 Algoritma VAM

Gudang	Agen					Supply	
	1	2	3	4	5		
A	100	150	200	140	35	260	150 - 100
					140		= 50
B	50	70	60	65	80	40	60 - 50
				160			= 10
C	40	90	100	150	130	150 - 100	90 - 40
	100					= 50	= 50
Demand	100 - 100 = 0	200	150	0	0		
	50 - 40 = 10	90 - 70 = 20	100 - 60 = 40				

Tabel 8. Iterasi 6 Algoritma VAM

Gudang	Agen					Supply	
	1	2	3	4	5		
A	100	150	200	140	35	260	200 - 150
					140		= 40
B	50	70	60	65	80	40	70 - 60
				160			= 10
C	40	90	100	150	130	50	100 - 90
	100						= 10
Demand	0	200	150	0	0		
		90 - 70 = 20	100 - 60 = 40				

Tabel 9. Iterasi 7 Algoritma VAM

Gudang	Agen					Supply	
	1	2	3	4	5		
A	100	150	200	140	35	260 - 200 = 60	200 - 150
		200			140		= 50
B	50	70	60	65	80	40	70 - 60
				160			= 10
C	40	90	100	150	130	50	100 - 90
	100						= 10
Demand	0	200 - 200 = 0	150	0	0		
		90 - 70 = 20	100 - 60 = 40				

Tabel 10. Iterasi 8 Algoritma VAM

Gudang	Agen					Supply	
	1	2	3	4	5		
A	100	150	200	140	35	60	200
		200			140		
B	50	70	60	65	80	40	60
				160			
C	40	90	100	150	130	50	100
	100						
Demand	0	0	150	0	0		

$100 - 60 = 40$

Tabel 11. Iterasi 9 Algoritma VAM

Gudang	Agen					Supply	
	1	2	3	4	5		
A	100	150	200	140	35	$60 - 60 = 0$	200
		200	60		140		
B	50	70	60	65	80	40	60
				160			
C	40	90	100	150	130	50	100
	100						
Demand	0	0	$150 - 60 = 90$	0	0		

$100 - 60 = 40$

Tabel 12. Iterasi 10 Algoritma VAM

Gudang	Agen					Supply	
	1	2	3	4	5		
A	100	150	200	140	35	0	
		200	60		140		
B	50	70	60	65	80	40	60
				160			
C	40	90	100	150	130	50	100
	100						
Demand	0	0	90	0	0		

$100 - 60 = 40$

Tabel 13. Iterasi 11 Algoritma VAM

Gudang	Agen					Supply
	1	2	3	4	5	
A	100	150	200	140	35	0
		200	60		140	
B	50	70	60	65	80	40
				160		
C	40	90	100	150	130	50 - 50 = 0
	100		50			
Demand	0	0	$90 - 50 = 40$ $100 - 60 = 40$	0	0	

60
100

Tabel 14. Iterasi 12 Algoritma VAM

Gudang	Agen					Supply
	1	2	3	4	5	
A	100	150	200	140	35	0
		200	60		140	
B	50	70	60	65	80	40 - 40 = 0
			40	160		
C	40	90	100	150	130	0
	100		50			
Demand	0	0	$40 - 40 = 0$	0	0	

60

Tabel 15. Iterasi 13 Algoritma VAM

Gudang	Agen					Supply
	1	2	3	4	5	
A	100	150	200	140	35	0
		200	60		140	
B	50	70	60	65	80	0
			40	160		
C	40	90	100	150	130	0
	100		50			
Demand	0	0	0	0	0	

Dari tabel di atas diperoleh total biaya minimum:

$$\begin{aligned}
 Z &= 160(65) + 140(35) + 100(40) + 200(150) + 60(200) + 50(100) + 40(60) \\
 &= 10.400 + 4.900 + 4.000 + 30.000 + 12.000 + 5.000 + 2.400 \\
 &= 68.700
 \end{aligned}$$

Berdasarkan perhitungan pada pendistribusian dengan VAM didapatkan total biaya distribusi sebesar 68.700. Berikut tabel pendistribusian dengan VAM.

Tabel 16. Pendistribusian

Gudang ke Agen	Alokasi	Harga Distribusi Satuan	Total Biaya
$A \rightarrow 2$	200	150	30.000
$A \rightarrow 3$	60	200	12.000
$A \rightarrow 5$	140	35	4.900
$B \rightarrow 3$	40	60	2.400
$B \rightarrow 4$	160	65	10.400
$C \rightarrow 1$	100	40	4.000
$C \rightarrow 3$	50	100	5.000
Total Keseluruhan Biaya Distribusi			68.700

4. Teori Dasar Graf

Menurut G.Chartrand [20] meliputi definisi graf, *adjacent*, *incident*, $u - v$ walk, *circuit*, *cycle*, graf terhubung, dan graf berbobot.

Definisi 2.1 Graf dapat didefinisikan sebagai himpunan tak kosong berhingga $V(G)$ disertai dengan relasi yang *symmetric* dan *irreflexive* pada V . Dalam graf G , $V(G)$ merupakan himpunan semua *vertex* pada graf G dan $E(G)$ adalah himpunan semua *edge* pada graf G . Sedangkan banyaknya *edge* disebut dan dinotasikan $|E(G)|$ dan banyaknya *edge* dalam G disebut *size* G .

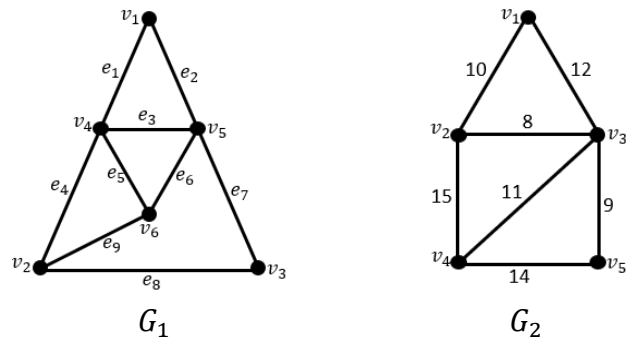
Definisi 2.2 Dua *Vertex* u dan v dikatakan *adjacent* pada graf G jika $uv \in E(G)$. Apabila *Vertex* u *incident* dengan *edge* e dan *vertex* v *incident* dengan *edge* e , maka $e = uv \in E(G)$ atau u dan v *incident* dengan e .

Definisi 2.3 Suatu $u - v$ walk pada graf G adalah barisan bergantian dari *vertex* dan *edge* G dimulai dari *vertex* u dan diakhiri *vertex* v , dengan setiap *edge* menghubungkan *vertex* terhadap barisan tersebut. Jika $u - v$ trail tidak ada pengulangan *edge* manapun, sedangkan suatu $u - v$ path pada graf adalah $u - v$ walk yang tidak ada pengulang *vertex*.

Definisi 2.4 *Circuit* adalah u - v *trail* dimana $u = v$ yang memiliki sedikitnya tiga *edge*. Jika *circuit* tidak ada pengulangan *vertex* di dalamnya (kecuali *vertex* awal dan akhir), maka disebut *cycle*.

Definisi 2.5 Graf G dikatakan terhubung jika setiap dua *vertex* pada graf G dapat ditentukan suatu *path* yang menghubungkan kedua *vertex* tersebut. Jika tidak, maka graf G tidak terhubung.

Definisi 2.6 Graf G dikatakan berbobot dimana setiap *edge* nya memiliki nilai (bobot). Nilai setiap *edge* dapat berbeda-beda tergantung permasalahan yang dimodelkan dengan graf.



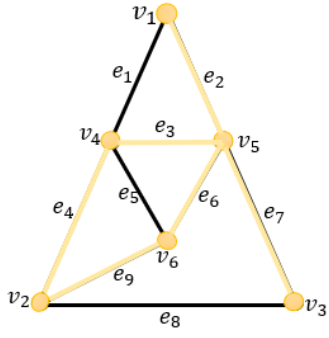
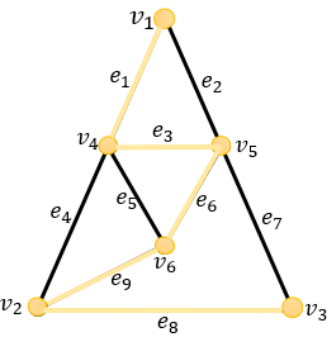
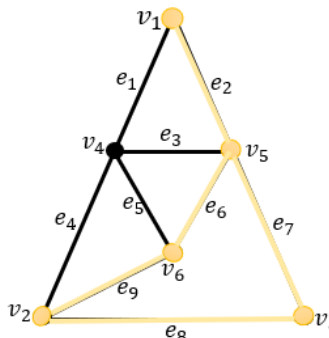
Gambar 2. Graf G_1 dan Graf G_2

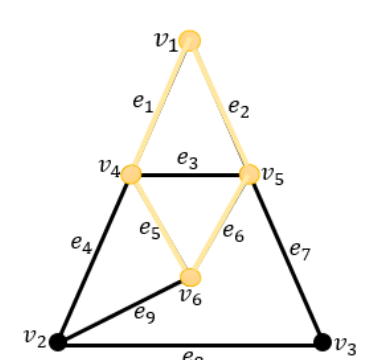
Berdasarkan Gambar 2. Graf tersebut termasuk contoh graf terhubung dengan $|V(G_1)| = 6$ dan $|E(G_1)| = 9$. Dari graf G_1 diperoleh bahwa v_1 *adjacent* dengan v_4 dan v_5 , e_3 *incident* dengan *vertex* v_1 dan v_4 .

Tabel 17. Memuat beberapa contoh lintasan pada graf G_1 yaitu *walk*, *trail*, *path*, *circuit*, dan *cycle*.

Tabel 17. Contoh *walk*, *trail*, *path*, *circuit*, dan *cycle*

No.	Contoh	Perbedaan	Rute
1	<i>Walk</i>	<i>Vertex dan edge</i> boleh diulang	Gambar 3. <i>Walk</i> $v_1, e_2, v_5, e_3, v_4, e_5, v_6, e_6, v_5$

2	<i>Trail</i>	Tidak ada pengulangan vertex	 Gambar 4. <i>Trail</i> $v_1, e_2, v_5, e_3, v_4, e_4, v_2, e_9, v_6, e_6, v_5, e_7, v_3$
3	<i>Path</i>	Tidak ada pengulangan edge	 Gambar 5. <i>Path</i> $v_1, e_1, v_4, e_3, v_5, e_6, v_6, e_9, v_2, e_8, v_3$
4	<i>Circuit</i>	Suatu <i>trail</i> yang memuat sedikitnya tiga edge	 Gambar 6. <i>Circuit</i> $v_1, e_2, v_5, e_6, v_6, e_9, v_2, e_8, v_3, e_7, v_5, e_3, v_4, e_1, v_1$

5	Cycle	Suatu <i>circuit</i> tidak ada pengulangan <i>vertex</i> didalamnya, kecuali <i>vertex</i> awal dan akhir	 Gambar 7. Cycle $v_1, e_1, v_4, e_5, v_6, e_6, v_5, e_2, v_1$
---	-------	---	---

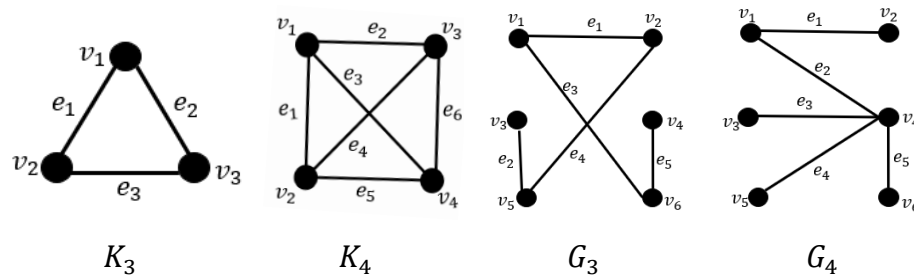
Sedangkan graf G_2 merupakan contoh dari graf berbobot. Contoh penentuan bobot, misal bobot $v_2 \rightarrow v_4$ adalah 15.

5. Kelas Graf

Pada bagian ini fokus pada dua kelas graf meliputi graf lengkap dan graf pohon:

Definisi 2.7 Graf lengkap merupakan graf sederhana dimana setiap *vertex* nya memiliki *edge* terhadap semua *vertex* lainnya. Graf lengkap dengan n buah *vertex* yaitu K_n .

Definisi 2.8 Graf pohon adalah graf sederhana yang terhubung dan tidak memuat *circuit*.



Gambar 8. Graf K_3 , Graf K_4 , Graf G_3 dan Graf G_4 .

Berdasarkan Gambar 8. Graf K_3 dan graf K_4 merupakan contoh graf lengkap, sedangkan graf G_3 dan graf G_4 merupakan contoh graf pohon.

6. Algoritma Branch and Bound

Definisi 2.9 Algoritma *Branch and Bound* merupakan algoritma yang membagi suatu masalah menjadi sub masalah lebih kecil yang kemudian mengarah pada penyelesaian dengan cara melakukan percabangan

(*branching*) dan pembatasan (*bounding*) untuk mencapai solusi yang optimal.

Percabangan (*branching*) yaitu proses membentuk suatu masalah menjadi struktur pohon pencarian (*search tree*). Proses percabangan dilakukan untuk membangun semua cabang pohon yang mengarah ke solusi, sedangkan proses pembatasan dilakukan dengan menghitung estimasi nilai (*cost*) *vertex* dengan memperhatikan *edge*. Algoritma *Branch and Bound* pertama kali dikemukakan oleh A. H. Land dan A. G. Doig pada tahun 1960 [7].

Secara umum, Algoritma *Branch and Bound* dalam mencari solusi menggunakan teknik *Least Cost Search* atau pencarian nilai terkecil, teknik ini akan menghitung nilai (*cost*) setiap *vertex*. *Vertex* yang mempunyai nilai terkecil dikatakan mempunyai kemungkinan terbesar untuk menghasilkan solusi. Setiap *vertex* aktif (*current node*) memiliki nilai yang menyatakan nilai batas (*bound*). Sebuah *vertex* aktif merupakan *vertex* yang mempunyai nilai batas terkecil karena teknik pencarian solusi (*Least Cost Search*) [21].

Misal $c(i)$ adalah nilai estimasi lintasan minimum dari *vertex* ke *vertex* tujuan, maka $c(i)$ menyatakan batas (*bound*) nilai pencarian solusi dari *vertex* i sehingga dapat dirumuskan fungsi heuristik untuk menghitung nilai (*cost*) sebagai berikut:

$$c(i) = f(i) + g(i)$$

dengan:

$c(i)$: Nilai untuk *vertex* i .

$f(i)$: Nilai lintasan dari *vertex* akar ke *vertex* i .

$g(i)$: Nilai untuk mencapai *vertex* tujuan dari *vertex* i .

Pada permasalahan yang lebih kompleks, sistem digambarkan dengan matriks A berukuran $n \times n$. *Vertex* yang dieksplorasi adalah *vertex* dengan nilai batas terkecil. Nilai batas didapatkan dari reduksi baris dan kolom matriks A yang merepresentasikan graf. Reduksi dilakukan dengan mengurangi nilai C_{ij} pada baris atau kolom dengan nilai C_{ij} terkecil pada baris atau kolom tersebut, sedemikian sehingga didapatkan matriks tereduksi $A(t)$ dengan sebuah nilai nol pada setiap baris dan kolom. Matriks A dikatakan tereduksi apabila setiap kolom dan barisnya memuat minimal satu nilai nol dan semua elemen lainnya non-negatif [7]. Selanjutnya, total nilai pereduksi menjadi nilai batas simpul akar. Untuk setiap simpul anak yang dibangkitkan dengan mengunjungi $A(i, j)$, dilakukan reduksi matriks $A(t)$ untuk mendapatkan matriks tereduksi $A(j, \text{simpul awal})$ diubah menjadi ∞ .

Berdasarkan persamaan fungsi heuristik untuk menghitung nilai estimasi yang telah dijelaskan sebelumnya, maka nilai batas *vertex* anak dihitung menggunakan rumus:

$$\hat{c}(S) = \hat{c}(R) + A(i, j) + r$$

dengan:

$\hat{c}(S)$: Nilai perjalanan minimum yang melalui *vertex* S, dimana S adalah anak dari R.

$\hat{c}(R)$: Nilai perjalanan minimum yang melalui *vertex* R, dimana R adalah orang tua dari S.

$A(i, j)$: Bobot *edge* (i, j) pada matriks tereduksi.

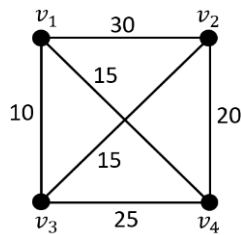
r : Jumlah semua pereduksi pada proses memperoleh matriks tereduksi untuk *vertex* S.

Berikut ini adalah contoh penerapan Algoritma *Branch and Bound*. Data yang digunakan, disajikan pada:

Tabel 18. Jarak antar *vertex*

<i>Vertex</i>	V_1	V_2	V_3	V_4
V_1	0	30	10	15
V_2	30	0	15	20
V_3	10	15	0	25
V_4	15	20	25	0

Tabel 18. Merepresentasikan jarak antar v_1 sampai v_4 , kemudian diubah dalam bentuk graf sebagai berikut:



Gambar 9. Graf G_5

Gambar 9. Mepresentasikan graf dari tabel 18. Kemudian, diubah dalam bentuk matriks sebagai berikut:

$$M_1 = \begin{bmatrix} 0 & 30 & 10 & 15 \\ 30 & 0 & 15 & 20 \\ 10 & 15 & 0 & 25 \\ 15 & 20 & 25 & 0 \end{bmatrix}$$

Matriks M_1 merupakan representasi matriks dari gambar 9. Selanjutnya, dicari rute terpendek dari v_1 ke semua *vertex* kemudian kembali ke v_1 menggunakan Algoritma *Branch and Bound* sebagai berikut:

a. Langkah 1

- 1) Memberikan label pada v_1 ke v_1 , v_2 ke v_2 dan seterusnya dengan label ∞ untuk setiap sel dalam matriks.

$$\begin{bmatrix} \infty & 30 & 10 & 15 \\ 30 & \infty & 15 & 20 \\ 10 & 15 & \infty & 25 \\ 15 & 20 & 25 & \infty \end{bmatrix}$$

- 2) Pada permasalahan minimalisasi terlebih dahulu mencari nilai terkecil disetiap baris, kemudian nilai terkecil tersebut digunakan untuk mengurangi semua nilai jarak yang ada pada baris yang sama.

$$\begin{bmatrix} \infty & 30 & 10 & 15 \\ 30 & \infty & 15 & 20 \\ 10 & 15 & \infty & 25 \\ 15 & 20 & 25 & \infty \end{bmatrix} \begin{matrix} B_1 - 10 \\ B_2 - 15 \\ B_3 - 10 \\ B_4 - 15 \end{matrix} \Rightarrow \begin{bmatrix} \infty & 20 & 0 & 5 \\ 15 & \infty & 0 & 5 \\ 0 & 5 & \infty & 15 \\ 0 & 5 & 10 & \infty \end{bmatrix}$$

- 3) Memastikan semua baris dan kolom sudah memiliki nilai nol. Apabila masih ada kolom yang belum memiliki nilai nol, maka cari nilai terkecil pada kolom tersebut lalu gunakan untuk mengurangi semua nilai yang ada pada kolom yang sama. Hasil matriks yang dikurangkan elemen terkecil disetiap baris dan setiap kolom disebut matriks (C_{ij}) . Semua angka yang digunakan untuk mengurangi setiap baris dan setiap kolom tersebut dijumlahkan, kemudian hasil penjumlahan inilah dijadikan sebagai r (root) atau bobot dari *vertex* awal atau akar. Hal ini juga berarti bahwa solusi pada persoalan ini paling tidak memiliki bobot minimum sebesar r (root).

$$\begin{bmatrix} \infty & 20 & 0 & 5 \\ 15 & \infty & 0 & 5 \\ 0 & 5 & \infty & 15 \\ 0 & 5 & 10 & \infty \end{bmatrix} \begin{matrix} K_2 - 5 \\ K_4 - 5 \end{matrix} \Rightarrow \begin{bmatrix} \infty & 15 & 0 & 0 \\ 15 & \infty & 0 & 0 \\ 0 & 0 & \infty & 10 \\ 0 & 0 & 10 & \infty \end{bmatrix} = A.$$

Total jumlah semua pengurangan = $10 + 15 + 10 + 15 + 5 + 5 = 60$

$$\hat{c}(1) = 60$$

b. Langkah 2

Misalkan A adalah matriks tereduksi untuk *vertex* R . Misal adalah anak dari *vertex* R sedemikian sehingga *edge* (R, S) pada ruang pohon pencarian berkoresponden dengan *edge* (i, j) . Jika bukan *vertex* daun, maka matriks bobot tereduksi untuk *vertex* S dapat dihitung dengan langkah sebagai berikut:

- a. Ubah semua nilai pada baris i dan kolom j menjadi ∞ , untuk mencegah agar tidak ada lintasan yang keluar dari *vertex* i atau masuk pada *vertex* j ,

- b. Ubah $A(j, 1)$ menjadi ∞ , untuk mencegah penggunaan *edge* $(j, 1)$, dan
- c. Reduksi kembali semua baris dan kolom pada matriks A kecuali untuk elemen ∞ .

Jika r adalah total pengurangan, maka nilai batas untuk *vertex* S adalah:

$$\hat{c}(S) = \hat{c}(R) + A(i, j) + r.$$

Hasil reduksi ini menghasilkan matriks B .

Perhitungan bobot lintasan antar *vertex* dapat ditunjukkan sebagai berikut:

- 1) Lintasan $v_1 - v_2, A(1,2) = 15$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 0 \\ 0 & \infty & \infty & 10 \\ 0 & \infty & 10 & \infty \end{bmatrix} = B$$

Nilai batas *vertex* 2 pada *search tree* adalah:

$$\hat{c}(2) = \hat{c}(1) + A(1,2) + r = 60 + 15 + 0 = 75$$

- 2) Lintasan $v_1 - v_3, A(1,3) = 0$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 15 & \infty & \infty & 0 \\ \infty & 0 & \infty & 10 \\ 0 & 0 & \infty & \infty \end{bmatrix}$$

Nilai batas *vertex* 3 pada *search tree* adalah:

$$\hat{c}(3) = \hat{c}(1) + A(1,3) + r = 60 + 0 + 0 = 60$$

- 3) Lintasan $v_1 - v_4, A(1,4) = 0$

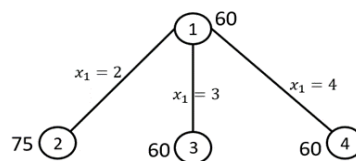
$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 15 & \infty & 0 & \infty \\ 0 & 0 & \infty & \infty \\ \infty & 0 & 10 & \infty \end{bmatrix}$$

Nilai batas untuk *vertex* 4 pada *search tree* adalah:

$$\hat{c}(4) = \hat{c}(1) + A(1,4) + r = 60 + 0 + 0 = 60$$

Dari 1-3 *search tree* terpendek hasilnya sama bernilai 60, diperoleh dari $v_1 - v_3$ dan $v_1 - v_4$.

Search tree yang terbentuk sampai saat ini adalah:



Gambar 10. *Search tree* sementara pertama

- 4) Lintasan $v_1, -v_3 - v_2, A(3,2) = 0$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \end{bmatrix} = C$$

Nilai batas *vertex* 5 pada *search tree* adalah:

$$\hat{c}(5) = \hat{c}(3) + A(3,2) + r = 60 + 0 + 0 = 60$$

- 5) Lintasan $v_1, -v_3 - v_4, A(3,4) = 10$

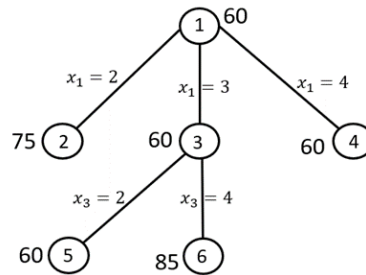
$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 15 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \end{bmatrix} B_3 - 15 \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \end{bmatrix}$$

Diperoleh $r = 15$, sehingga nilai batas untuk *vertex* 6 pada *search tree* adalah:

$$\hat{c}(6) = \hat{c}(3) + A(3,4) + r = 60 + 15 + 10 = 85$$

Dari 4-5 *search tree* terpendek bernilai 60, diperoleh dari $v_1, -v_3 - v_2$.

Search tree yang terbentuk sampai saat ini adalah:



Gambar 11. *Search tree* sementara kedua

- 6) Lintasan $v_1, -v_4 - v_2, A(4,2) = 0$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} = D$$

Nilai batas *vertex* 7 pada *search tree* adalah:

$$\hat{c}(7) = \hat{c}(4) + A(4,2) + r = 60 + 0 + 0 = 60$$

- 7) Lintasan $v_1, -v_4 - v_3, A(4,3) = 10$

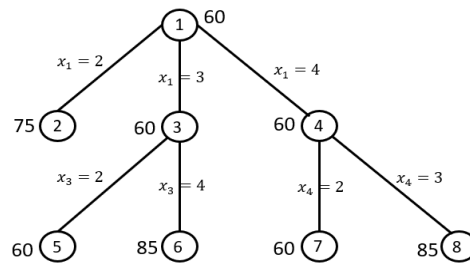
$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 15 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} B_2 - 15 \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \end{bmatrix}$$

Diperoleh $r = 15$, sehingga nilai batas untuk *vertex* 8 pada *search tree* adalah:

$$\hat{c}(8) = \hat{c}(4) + A(4,3) + r = 60 + 15 + 10 = 85$$

Dari 6-7 *search tree* terpendek bernilai 60, diperoleh dari $v_1, -v_4 - v_2$.

Search tree yang terbentuk sampai saat ini adalah:



Gambar 12. *Search tree* sementara ketiga

8) Lintasan $v_1, -v_3 - v_2 - v_4, A(2,4) = 0$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Nilai batas *vertex* 9 pada *search tree* adalah:

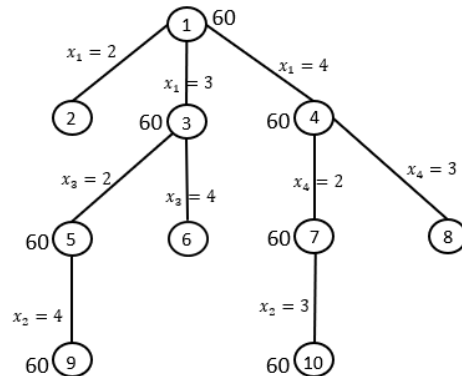
$$\hat{c}(9) = \hat{c}(5) + A(2,4) + r = 60 + 0 + 0 = 60$$

9) Lintasan $v_1, -v_4 - v_2 - v_3, A(2,3) = 0$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

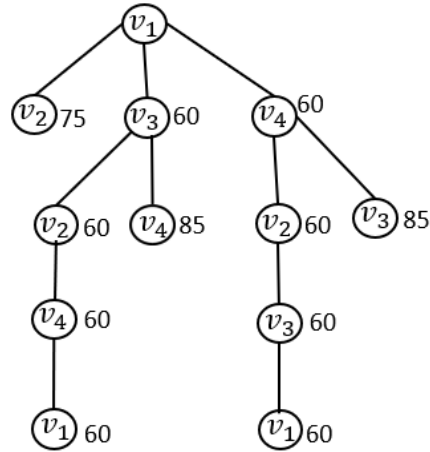
Nilai batas *vertex* 10 pada *search tree* adalah:

$$\hat{c}(10) = \hat{c}(7) + A(2,3) + r = 60 + 0 + 0 = 60$$



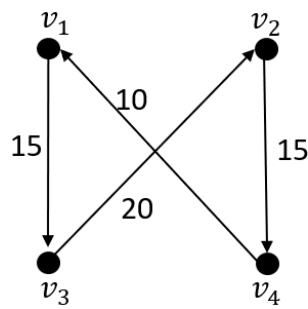
Gambar 13. *Search tree* terakhir

Karena tidak terdapat lagi *vertex* pada *search tree*, maka solusi terbaik yang dipilih adalah $v_1, - v_3 - v_2 - v_4, - v_1$ atau $v_1, - v_4 - v_2 - v_3 - v_1$ dengan total bobot minimal 60. Jadi, diperoleh panjang lintasannya 60.

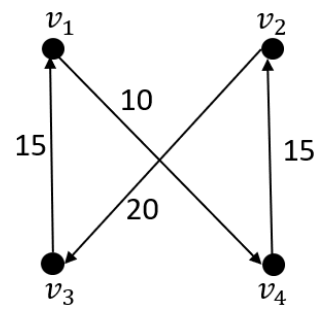


Gambar 14. Hasil akhir model graf Algoritma *Branch and Bound*

Berdasarkan contoh kasus di atas, diperoleh rute sebagai berikut:



Gambar 15. Rute pendistribusian utama



Gambar 16. Rute alternatif