Jurnal Riset Mahasiswa Matematika
Volume 5 (1) (2025), Pages 29-40
p-ISSN: 2808-1552; e-ISSN: 2808-4926

Pembobotan Ulang pada Graf Berbobot Negatif untuk Menerapkan Algoritma Dijkstra dalam Menentukan Lintasan Terpendek

Natasya Thalia Salsabillah, Mohammad Nafie Jauhari*, and Achmad Nashichuddin

Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia

Abstrak

Algoritma Dijkstra merupakan algoritma untuk mencari lintasan terpendek yang bekerja secara optimal pada graf berbobot non-negatif. Namun, dalam berbagai permasalahan nyata seperti sistem transportasi dan jaringan keuangan, sering ditemukan sisi dengan bobot negatif yang menyebabkan Algoritma Dijkstra tidak dapat diterapkan secara langsung. Penelitian ini bertujuan untuk mengatasi keterbatasan tersebut dengan menerapkan metode pembobotan ulang menggunakan Algoritma Johnson. Metode ini mengombinasikan Algoritma Bellman-Ford dan Dijkstra untuk mengubah bobot negatif menjadi non-negatif tanpa mengubah struktur solusi optimal. Data yang digunakan berupa dua graf acak berarah yang masing-masing terdiri dari 31 simpul, yang dibuat menggunakan algoritma Erdos-Renyi. Hasil penelitian menunjukkan bahwa pembobotan ulang berhasil membuat bobot graf menjadi non-negatif sehingga Algoritma Dijkstra dapat diterapkan, dan hasil lintasan terpendek yang diperoleh sama dengan hasil dari Algoritma Bellman-Ford. Dengan demikian, metode pembobotan ulang menggunakan Algoritma Johnson terbukti efektif dalam menangani bobot negatif dan tetap menjaga keakuratan hasil pencarian lintasan terpendek menggunakan Algoritma Dijkstra.

Kata Kunci: Algoritma Dijkstra; Bobot Negatif; Pembobotan Ulang; Algoritma Johnson; Lintasan Terpendek.

Abstract

Dijkstra's algorithm is an algorithm for finding the shortest path that works optimally on non-negative weighted graphs. However, in various real problems such as transportation systems and financial networks, edges with negative weights are often found which cause the Dijkstra Algorithm to be unable to be applied directly. This study aims to overcome these limitations by implementing a reweighting method using the Johnson Algorithm. This method combines the Bellman-Ford and Dijkstra Algorithms to change negative weights to non-negative without changing the structure of the optimal solution. The data used are two random directed graphs, each consisting of 31 nodes, which are created using the Erdos-Renyi algorithm. The results of the study show that reweighting successfully makes the graph weights non-negative so that the Dijkstra Algorithm can be applied, and the shortest path results obtained are the same as the results of the Bellman-Ford Algorithm. Thus, the reweighting method using the Johnson Algorithm is proven to be effective in handling negative weights and maintaining the accuracy of the shortest path search results using the Dijkstra Algorithm.

Keywords: Dijkstra's Algorithm; Negative Weight; Reweighting; Johnson's Algorithm; Shortest Path.

*Corresponding author. E-mail: nafie.jauhari@uin-malang.ac.id

Submitted: July 18, 2025 Reviewed: August 18, 2025 Accepted: October 24, 2025
DOI: https://doi.org/10.18860/jrmm.v5i1.34383

1 Pendahuluan

Algoritma Dijkstra merupakan salah satu metode yang digunakan untuk menyelesaikan masalah lintasan terpendek dalam graf berbobot. Namun, algoritma ini hanya dapat digunakan jika semua bobot sisi bernilai tidak negatif [1]. Bobot negatif pada graf mengacu pada nilai negatif yang diberikan sisi, yang dapat diartikan sebagai pengurangan jarak, biaya, atau waktu dalam konteks tertentu. Dalam berbagai aplikasi dunia nyata, bobot negatif sering muncul, misalnya dalam bentuk diskon dalam transaksi ekonomi, keuntungan dalam sistem keuangan, atau pengurangan waktu perjalanan dalam jaringan transportasi akibat faktor tertentu. Namun, jika tidak ditangani dengan benar, bobot negatif dapat menyebabkan ketidakakuratan dalam pencarian lintasan terpendek, terutama jika membentuk siklus negatif, yaitu jalur tertutup dengan total bobot negatif yang membuat algoritma terus berulang tanpa batas [2].

Ketidakmampuan Algoritma Dijkstra untuk menangani bobot negatif menjadi kendala dalam berbagai aplikasi. Sebagai contoh, dalam sistem transportasi, biaya perjalanan antar kota mungkin memiliki diskon dinamis yang membuat bobot lintasan negatif dalam kondisi tertentu . Jika siklus negatif terbentuk, Algoritma Dijkstra tidak dapat digunakan karena tidak dapat membedakan apakah jalur tersebut optimal atau tidak valid. Untuk mengatasi keterbatasan ini, metode pembobotan ulang diperlukan agar Algoritma Dijkstra tetap dapat digunakan dalam kondisi graf berbobot negatif [3].

Salah satu metode yang dapat digunakan untuk pembobotan ulang adalah Algoritma Johnson, yang menggabungkan keunggulan Algoritma Bellman-Ford dan Algoritma Dijkstra untuk mengatasi keterbatasan dalam graf berbobot negatif. Metode ini bekerja dengan mendeteksi siklus negatif dan mentransformasikan graf menjadi non-negatif, lalu menggunakan Algoritma Dijkstra untuk menentukan lintasan terpendek dengan lebih efisien [4]. Dengan demikian, Algoritma Johnson memungkinkan pencarian lintasan terpendek pada graf berbobot negatif tanpa mengubah struktur graf secara signifikan.

Penelitian ini bertujuan untuk menerapkan metode pembobotan ulang pada graf berbobot negatif agar Algoritma Dijkstra dapat digunakan secara efektif untuk menentukan lintasan terpendek. Dengan memanfaatkan keunggulan Algoritma Johnson, penelitian ini diharapkan dapat memberikan solusi yang efisien dan akurat untuk permasalahan lintasan terpendek pada graf dengan bobot negatif, sekaligus memberikan kontribusi terhadap pengembangan algoritma dalam teori graf dan aplikasinya di dunia nyata.

Dengan demikian, berdasarkan uraian tersebut, peneliti terinspirasi untuk meneliti pembobotan ulang pada graf berbobot negatif, sehingga Algoritma Dijkstra dapat digunakan untuk menemukan lintasan terpendek dengan hasil yang akurat dan efisien. Penelitian ini diharapkan dapat mengatasi keterbatasan Dijkstra yang tidak mampu menangani bobot negatif secara langsung, serta memberikan kontribusi terhadap pengembangan algoritma pencarian lintasan terpendek yang lebih adaptif dalam berbagai kondisi graf, khususnya dalam konteks aplikasi dunia nyata yang mengandung bobot negatif.

2 Metode

Penelitian ini menggunakan pendekatan kuantitatif dengan menganalisis efektivitas pembobotan ulang pada graf berbobot negatif agar dapat diterapkan dalam Algoritma Dijkstra. Pendekatan ini memungkinkan analisis numerik terhadap perubahan hasil lintasan terpendek sebelum dan sesudah dilakukan pembobotan ulang. Simulasi dilakukan pada graf acak yang dibangun menggunakan algoritma Erdos-Renyi dengan 31 simpul dan bobot sisi yang dapat bernilai positif maupun negatif. Parameter yang dianalisis mencakup waktu eksekusi, perubahan bobot lintasan, jumlah iterasi, dan akurasi hasil lintasan terpendek.

Penggunaan data simulasi dari graf acak Erdos-Renyi memberikan fleksibilitas dan validitas dalam menguji performa algoritma pada kondisi graf yang kompleks [5]. Proses ini diharapkan dapat memperlihatkan bahwa pembobotan ulang memungkinkan Algoritma Dijkstra berfungsi pada graf dengan bobot negatif secara akurat. Berikut tahapan penelitian implementasi pembobotan ulang pada graf berbobot negatif [4]:

1. Menentukan data graf berbobot menggunakan algoritma Erdos-Renyi.

2. Menambahkan simpul baru (s) yang dihubungkan ke semua simpul dalam graf dengan bobot 0.
3. Menjalankan algoritma Bellman-Ford dari simpul s untuk menghitung nilai potensial $h(v)$ setiap simpul.
4. Melakukan pembobotan ulang pada setiap jalur$w'(u,v) = w(u,v) + h(u) - h(v)$, sehingga semua bobot menjadi non-negatif.
5. Menjalankan algoritma Dijkstra pada graf yang sudah dilakukan pembobotan ulang untuk menentukan lintasan terpendek.
6. Interpretasi dan Analisis Hasil.
   (a) Hasil dari penerapan algoritma Dijkstra pada graf yang telah dilakukan pembobotan ulang akan dibandingkan dengan hasil lintasan terpendek pada graf asli (sebelum pembobotan ulang).
   (b) Analisis ini bertujuan untuk menguji apakah proses pembobotan ulang memungkinkan algoritma Dijkstra menemukan lintasan terpendek dengan benar, meskipun graf yang digunkan sebelumnya memiliki bobot negatif.

3 Hasil dan Pembahasan

3.1 Deskripsi Data

Data dalam penelitian ini menggunakan dua graf acak berarah yang diperoleh menggunakan algoritma Erdos-Renyi dengan 31 simpul pada masing-masing graf. Setiap sisi dalam graf memiliki bobot yang dapat bernilai negatif maupun positif, sehingga merepresentasikan kondisi graf yang kompleks sebagaimana ditemukan dalam berbagai permasalahan dunia nyata, seperti jaringan transportasi atau sistem keuangan [6]. Perbedaam struktur antara graf pertama dan graf kedua mencakup variasi pada nilai bobot, simpul asal dan simpul tujuan. Graf pertama menunjukkan distribusi bobot yang relatif seimbang antara nilai negatif dan positif, sedangkan graf kedua memiliki konsentrasi bobot negatif yang lebih tinggi serta pola konektivitas yang lebih rapat antar simpul.

Sebagaimana dijelaskan dalam teori graf berbobot, nilai bobot pada sisi graf dapat direpresentasikan dalam bentuk matriks ketetanggaan (adjacency matrix) yang mempermudah proses perhitungan dan pemrograman [7]. Dalam konteks ini, kedua graf yang digunakan dalam direpresentasikan dalam bentuk matriks tersebut agar proses transformasi bobot dan penerapan algoritma lebih sistematis. Kondisi ini penting karena, menurut kajian teori, Algoritma Dijkstra tidak dapat diterapkan secara langsung jika terdapat bobot negatif [8]. Oleh karena itu, diperlukan teknik pembobotan ulang yang sesuai seperti Algoritma Johnson, yang mengubah bobot negatif menjadi non-negatif tanpa mengubah solusi optimal. Pada Tabel 1 disajikan data graf acak pertama dan pada Tabel Tabel 2 disajikan data graf acak kedua.

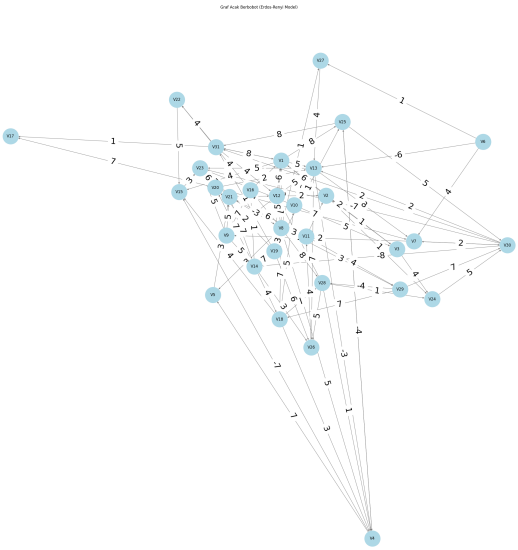
Tabel 1: Data Graf Acak Pertama

No	Simpul Asal	Simpul Tujuan	Bobot
1	V_1	V_{16}	2
2	V_1	V_{26}	5
3	V_2	V_4	4
4	V_2	V_{19}	8
5	V_3	V_9	5
⋮	⋮	⋮	⋮
76	V_{28}	V_2	8
77	V_{28}	V_{19}	4
78	V_{29}	V_{19}	4
79	V_{29}	V_{25}	1
80	V_{31}	V_{18}	7

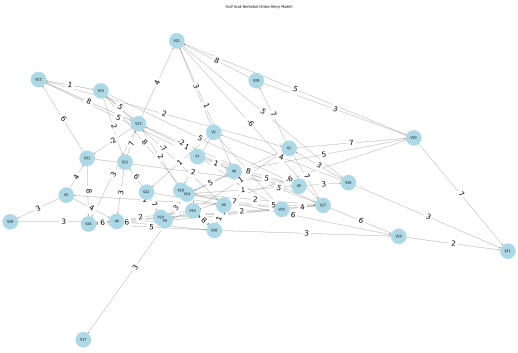
Tabel 2: Data Graf Acak Kedua

No	Simpul Asal	Simpul Tujuan	Bobot
1	V_1	V_{11}	6
2	V_1	V_{15}	7
3	V_1	V_{23}	-5
4	V_1	V_{26}	4
5	V_1	V_{29}	8
⋮	⋮	⋮	⋮
76	V_{30}	V_6	4
77	V_{30}	V_{17}	5
78	V_{30}	V_{20}	1
79	V_{31}	V_{18}	3
80	V_{31}	V_{19}	8

Gambar 1 dan Gambar 2 menunjukkan representasi graf berbobot berarah yang dibangun menggunakan model Erdos-Renyi.



Gambar 1: Matriks Model Graf Acak Pertama



Gambar 2: Matriks Model Graf Acak Kedua

3.2 Penerapan Algoritma Johnson

Algoritma Johnson diterapkan untuk menentukan jarak terpendek antara setiap pasangan simpul dalam graf berbobot, termasuk yang memiliki bobot negatif. Algoritma ini mengombinasikan Algoritma Bellman-

Ford dengan Algoritma Dijkstra guna memastikan bahwa bobot negatif dapat ditangani dengan pembobotan ulang sebelum pencarian jalur terpendek dilakukan. Keunggulan Algoritma Johnson dibandingkan metode lain, seperti menggunakan Bellman-Ford untuk setiap pasangan simpul, terletak pada efisiensinya. Dengan kompleksitas $O(V \times E)$ untuk satu kali Bellman-Ford dan $O(VlogV + E)$ untuk setiap iterasi Dijkstra, algoritma ini lebih cepat untuk graf besar yang jarang (*sparse*) [9].

1. Penambahan simpul baru dalam graf
   Graf asli diperluas dengan menambahkan simpul tambahan s, yang terhubung ke seluruh simpul yang ada dengan sisi berbobot nol.

$$\begin{aligned} s \to V_1 &= 0 \\ s \to V_2 &= 0 \\ s \to V_3 &= 0 \\ &\vdots \\ s \to V_{31} &= 0 \end{aligned}$$

2. Menjalankan Algoritma Bellman-Ford
   Algoritma Bellman-Ford dijalankan dari simpul s untuk menghitung nilai potensial $h(v)$ bagi setiap simpul v. Jika ditemukan siklus negatif, maka proses pencarian rute terpendek dihentikan karena graf tidak dapat dihitung dengan metode ini [10].
   (a) Inisialisasi
   Langkah pertama dalam Algoritma Bellman-Ford yaitu melakukan inisialisasi terhadap semua simpul dalam graf. Inisialisasi ini bertujuan untuk menetapkan nilai awal jarak dari simpul sumber ke semua simpul lainnya sebelum dilakukan proses iterasi.

$$\begin{aligned} \text{dist}(s) &= 0 \\ \text{dist}(V_1) &= \infty \\ \text{dist}(V_2) &= \infty \\ &\vdots \\ \text{dist}(V_{31}) &= \infty \end{aligned}$$

   (b) Proses Iterasi
   Setelah proses inisialisasi selesai, dilakukan iterasi sebanyak $|V| - 1$ kali untuk memperbarui nilai jarak pada setiap simpul dalam graf. Pada setiap iterasi, dilakukan relaksasi terhadap semua sisi (u, v, w) dalam graf. Relaksasi bertujuan untuk memastikan bahwa nilai jarak dari simpul awal ke setiap simpul lainnya selalu dalam kondisi optimal. Adapun aturan relaksasi dalam Algoritma Bellman-Ford sebagai berikut.

$$\text{Jika } \text{dist}(u) + w < \text{dist}(v), \text{ maka diperbarui menjadi } \text{dist}(v) = \text{dist}(u) + w$$

   Artinya, jika ditemukan jalur baru menuju simpul v yang lebih pendek daripada jalur sebelumnya, maka nilai $dist(v)$ diperbarui [11].
   Iterasi 1
   Pada iterasi pertama, algoritma mulai mengevaluasi setiap simpul berdasarkan bobot sisi yang ada. Jika ditemukan jalur dengan bobot lebih kecil daripada nilai jarak sebelumnya, maka nilai$Dist(v)$ diperbarui. Dalam Tabel 3, beberapa simpul masih memiliki bobot nol karena belum dilakukan pembaruan nilai jarak yang signifikan. Selain itu, keberadaan bobot negatif menunjukkan perlunya proses pembobotan ulang sebelum Algorima Dijkstra dapat berfungsi dengan benar.

Tabel 3: Nilai Iterasi Pertama

Simpul Asal	Simpul Tujuan	Jarak	Bobot
V_1	V_{16}	Dist(16)	0
V_1	V_{26}	Dist(26)	0
V_2	V_4	Dist(4)	0
⋮	⋮	⋮	⋮
V_{29}	V_{19}	Dist(19)	-2
V_{29}	V_{25}	Dist(25)	-7
V_{31}	V_{18}	Dist(18)	0

Iterasi 2

Pada proses iterasi kedua, proses relaksasi dalam Algoritma Bellman-Ford kembali dilakukan untuk memperbarui nilai jarak setiap simpul dalam graf berbobot negatif. Pada tahap ini, setiap simpul diperiksa ulang untuk memastikan apakah ada jalur baru yang lebih pendek dibandingkan dengan jarak yang telah tercatat sebelumnya. Jika ditemukan jalur dengan bobot yang lebih kecil, maka nilai jarak diperbarui agar lebih optimal. Hasil dari iterasi kedua menunjukkan bahwa beberapa simpul mengalami perubahan jarak akibat adanya jalur yang lebih efisien yang ditemukan dalam proses relaksasi. Selain itu, masih terdapat bobot negatif dalam graf, yang menunjukkan bahwa proses pembobotan ulang tetap diperlukan agar Algoritma Dijkstra dapat diterapkan dengan benar. Proses iterasi ini akan terus berlangsung hingga tidak ada lagi perubahan nilai jarak, yang menandakan bahwa jalur terpendek telah ditemukan secara optimal.

Tabel 4: Nilai Iterasi Kedua

Simpul Asal	Simpul Tujuan	Jarak	Bobot
V_1	V_{16}	Dist(16)	0
V_1	V_{26}	Dist(26)	-6
V_2	V_4	Dist(4)	0
⋮	⋮	⋮	⋮
V_{29}	V_{19}	Dist(19)	-2
V_{29}	V_{25}	Dist(25)	-7
V_{31}	V_{18}	Dist(18)	0

Iterasi 3

Pada iterasi ketiga, proses relaksasi dalam Algoritma Bellman-Ford kembali dilakukan dengan prinsip yang sama seperti iterasi sebelumnya, yaitu memeriksa setiap sisi dalam graf dan memperbarui jarak jika ditemukan jalur yang lebih pendek. Namun, hasil iterasi ketiga menunjukkan bahwa tidak ada perubahan nilai jarak dibandingkan dengan iterasi kedua. Hal ini menunjukkan bahwa proses relaksasi telah mencapai kestabilan, di mana semua jalur terpendek telah ditemukan dan tidak ada lagi pembaruan nilai jarak yang diperlukan. Dengan demikian, iterasi berikutnya tidak lagi menghasilkan perubahan, sehingga iterasi dapat dihentikan lebih awal karena telah mencapai konvergensi.

Tabel 5: Nilai Iterasi Ketiga

Simpul Asal	Simpul Tujuan	Jarak	Bobot
V_1	V_{16}	Dist(16)	0
V_1	V_{26}	Dist(26)	-6
V_2	V_4	Dist(4)	0
⋮	⋮	⋮	⋮
V_{29}	V_{19}	Dist(19)	-2
V_{29}	V_{25}	Dist(25)	-7
V_{31}	V_{18}	Dist(18)	0

Pada tahap ini, nilai potensial $h(v)$ untuk setiap simpul telah diperoleh dari hasil akhir Algoritma Bellman-Ford. Nilai ini mencerminkan jarak minimum dari simpul sumber tambahan ke setiap simpul dalam graf, yang selanjutnya digunakan untuk melakukan pembobotan ulang agar bobot sisi menjadi non-negatif sebelum penerapan Algoritma Dijkstra.

Tabel 6: Nilai Potensial $h(v)$

Nilai Potensial $h(v)$	Bobot
h(16)	= 0
h(26)	= -6
h(4)	= 0
	⋮
h(19)	= -2
h(25)	= -7
h(18)	= 0

3. Proses Pembobotan Ulang
Setelah menjalankan Algoritma Bellman-Ford dari simpul tambahan s, diperoleh nilai potensial $h(v)$ untuk setiap simpul dalam graf. Nilai ini digunakan untuk melakukan pembobotan ulang pada setiap sisi dalam graf guna memastikan bahwa semua bobot menjadi non-negatif. Proses pembobotan ulang dilakukan dengan menerapkan transformasi bobot menggunakan rumus [12]:

$$\begin{aligned} w'(u,v) &= w(u,v) + h(u) - h(v) \\ w'(V_1,V_{16}) &= 2 + 0 - 0 \\ w'(V_1,V_{16}) &= 2 \end{aligned}$$

Dengan transformasi ini, bobot negatif yang terdapat dalam graf asli diubah menjadi bobot non-negatif tanpa mengubah jalur terpendek yang sebenarnya. Hal ini memungkinkan Algoritma Dijkstra digunakan secara optimal untuk menentukan lintasan terpendek pada graf yang sebelumnya mengandung bobot negatif. Setelah proses pembobotan ulang selesai, langkah selanjutnya adalah memverifikasi apakah semua bobot telah menjadi non-negatif, yang kemudian akan ditampilkan dalam tabel berikut.

Tabel 7: Hasil Pembobotan Ulang

Simpul Asal	Simpul Tujuan	Bobot Baru
V_1	V_{16}	2
V_1	V_{26}	11
V_2	V_4	1
⋮	⋮	⋮
V_{29}	V_{19}	6
V_{29}	V_{25}	8
V_{31}	V_{18}	7

4. Menjalankan Algoritma Dijkstra
   Setelah seluruh bobot pada graf diubah menjadi non-negatif melalui proses pembobotan ulang, langkah selanjutnya adalah menerapkan Algoritma Dijkstra untuk menentukan lintasan terpendek dari simpul sumber ke seluruh simpul lainnya. Dengan bobot yang telah dimodifikasi, Algoritma Dijkstra dapat bekerja secara optimal tanpa terpengaruh oleh nilai negatif [13].
   (a) Inisialisasi
   Langkah pertama dalam penerapan Algoritma Dijkstra adalah melakukan inisialisasi terhadap setiap simpul dalam graf. Inisialisasi dilakukan dengan memberikan nilai jarak awal sebesar 0 pada simpul sumber, sedangkan semua simpul lainnya diberi nilai jarak tak hingga ∞ [14] Hal ini bertujuan untuk memastikan bahwa algoritma dapat membandingkan dan memperbarui nilai jarak yang lebih kecil secara iteratif.

Tabel 8: Inisialisasi Algoritma Dijkstra

Simpul Asal	Simpul Tujuan	Bobot Baru
V_1	0	Sudah dikunjungi (simpul sumber)
V_2	∞	Belum dikunjungi
V_3	∞	Belum dikunjungi
$\vdots$	$\vdots$	$\vdots$
V_{29}	∞	Belum dikunjungi
V_{30}	∞	Belum dikunjungi
V_{31}	∞	Belum dikunjungi

Setelah proses inisialisasi selesai, Algoritma Dijkstra dijalankan untuk menentukan lintasan terpendek dari simpul sumber ke seluruh simpul lainnya. Proses ini dilakukan secara iteratif, di mana pada setiap langkah dipilih simpul dengan jarak terpendek yang belum diproses, kemudian dilakukan pembaruan (relaksasi) terhadap jarak menuju simpul-simpul tetangganya.

(b) Proses Relaksasi
Relaksasi dilakukan untuk memperbarui nilai jarak apabila ditemukan lintasan baru yang lebih pendek dari jarak sebelumnya. Jika terdapat lintasan alternatif menuju simpul tertentu dengan total bobot lebih kecil, maka nilai jarak lama digantikan dengan nilai baru. Proses ini diulangi hingga semua simpul telah dikunjungi.

Tabel 9: Proses Relaksasi Algoritma Dijkstra

Iterasi	Simpul Diproses	Simpul Diperbarui	Jarak Sebelumnya	Jarak Terpendek	Lintasan	Keterangan
1	V_{10}	V_{20}	∞	4	$V_{10} \rightarrow V_{20}$	Diperbarui
1	V_{10}	V_{22}	∞	7	$V_{10} \rightarrow V_{22}$	Diperbarui
1	V_{10}	V_4	∞	8	$V_{10} \rightarrow V_4$	Diperbarui
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
27	V_{27}	V_{29}	∞	27	$V_{10} \rightarrow V_{22} \rightarrow V_{15} \rightarrow V_{28} \rightarrow V_{19} \rightarrow V_7 \rightarrow V_8 \rightarrow V_{27} \rightarrow V_{29}$	Diperbarui
28	V_{29}	V_{19}	13	33	-	Tidak Diperbarui
29	V_{29}	V_{25}	15	35	-	Tidak Diperbarui

Berdasarkan Tabel 9, proses relaksasi dalam penerapan Algoritma Dijkstra menunjukkan bahwa jarak dari simpul sumber ke simpul lainnya mengalami pembaruan secara bertahap. Pada setiap iterasi, simpul dengan jarak terkecil yang belum dikunjungi dipilih, lalu dilakukan pembaruan nilai jarak (relaksasi) ke simpul-simpul tetangganya jika ditemukan lintasan yang lebih efisien. Proses ini berlangsung hingga tidak ada lagi nilai jarak yang dapat diperbarui, yang menandakan bahwa algoritma telah mencapai konvergensi dan seluruh lintasan terpendek berhasil ditemukan secara optimal.

Keberhasilan proses relaksasi ini menandakan bahwa pembobotan ulang menggunakan Algoritma Johnson telah berhasil mengubah seluruh bobot sisi negatif menjadi non-negatif tanpa mengubah struktur solusi optimal. Dengan demikian, Algoritma Dijkstra dapat berjalan secara efektif dan menghasilkan lintasan terpendek yang valid pada graf yang sebelumnya mengandung bobot negatif.
Salah satu hasil yang ditinjau secara khusus dalam penelitian ini adalah lintasan terpendek dari simpul V_10 ke simpul V_31 diperoleh melalui jalur:

$$V_{10} \rightarrow V_{22} \rightarrow V_{15} \rightarrow V_{28} \rightarrow V_{19} \rightarrow V_{31}$$

Lintasan tersebut merupakan jalur dengan bobot total setelah dilakukan pembobotan ulang yang paling minimum di antara seluruh alternatif lintasan yang memungkinkan. Total bobot lintasan ini adalah 16. Hasil ini memperkuat bahwa penerapan pembobotan ulang tidak hanya menyelesaikan permasalahan nilai bobot negatif, tetapi juga tetap menjaga keakuratan hasil perhitungan lintasan terpendek. Dengan demikian, metode ini memungkinkan Algoritma Dijkstra digunakan secara lebih fleksibel, bahkan pada graf yang kompleks dan mengandung sisi berbobot negatif.

Dengan tahapan yang sama menggunakan graf acak yang memiliki bobot berbeda, pada graf acak kedua proses relaksasi menggunakan Algoritma Bellman-Ford untuk memperoleh nilai potensial $h(v)$ selesai pada iterasi ke-delapan. Setelah nilai potensial diperoleh dan graf dilakukan pembobotan ulang, lintasan terpendek dari simpul V_15 ke simpul V_31 kemudian diperoleh menggunakan Algoritma Dijkstra melalui jalur:

$$V_{15} \rightarrow V_{26} \rightarrow V_{28} \rightarrow V_{20} \rightarrow V_{17} \rightarrow V_{31}$$

Lintasan tersebut merupakan jalur dengan bobot total setelah dilakukan pembobotan ulang yang paling minimum di antara seluruh alternatif lintasan yang memungkinkan. Total bobot lintasan ini adalah 1.

3.3 Perbandingan Lintasan Terpendek Algoritma Bellman-Ford dengan Algoritma Dijkstra

Keakuratan hasil lintasan terpendek setelah dilakukan pembobotan ulang menggunakan Algoritma Johnson diuji dengan membandingkannya terhadap hasil dari Algoritma Bellman-Ford yang diterapkan secara langsung. Pemilihan Algoritma Bellman-Ford didasarkan pada kemampuannya menangani bobot negatif tanpa proses pembobotan ulang [15]. Setelah dilakukan pembobotan ulang menggunakan Algoritma Johnson dan diterapkan pada Algoritma Dijkstra, lintasan terpendek yang dihasilkan pada kedua graf dibandingkan untuk setiap pasangan simpul.

Perbandingan dilakukan dengan menghitung total panjang lintasan yang dihasilkan oleh kedua algoritma menggunakan bobot pada graf asli sebelum pembobotan ulang. Meskipun lintasan dari Algoritma Dijsktra diperoleh melalui graf yang telah dilakukan pembobotan ulang, penilaian terhadap panjang lintasannya tetap mengacu pada graf asli. Langkah ini bertujuan untuk memastikan bahwa lintasan terpendek yang diperoleh dari Algoritma Dijkstra setelah pembobotan ulang tetap mencerminkan struktur optimal yang sama dengan lintasan hasil Algoritma Bellman-Ford. Apabila total panjang lintasan dari kedua algoritma menunjukkan kesamaan nilai untuk setiap pasangan simpul, maka dapat disimpulkan bahwa pembobotan ulang tidak memengaruhi keakuratan hasil lintasan terpendek.

Tabel 10: Perbandingan Lintasan Terpendek Bellman-Ford dan Dijkstra pada Graf Acak Pertama

No	Simpul Asal → Simpul Tujuan	Algoritma Bellman-Ford	Algoritma Dijkstra	Selisih Panjang Rute
1	$V_1 \rightarrow V_1$	0	0	0
2	$V_1 \rightarrow V_{16}$	2	2	0
3	$V_1 \rightarrow V_{26}$	11	11	0
⋮	⋮	⋮	⋮	⋮
959	$V_{31} \rightarrow V_{21}$	18	18	0
960	$V_{31} \rightarrow V_{28}$	30	30	0
961	$V_{31} \rightarrow V_{23}$	17	17	0
		Total Selisih		**0**

Tabel 11: Perbandingan Lintasan Terpendek Bellman-Ford dan Dijkstra pada Graf Acak Kedua

No	Simpul Asal → Simpul Tujuan	Algoritma Bellman-Ford	Algoritma Dijkstra	Selisih Panjang Rute
1	$V_1 \rightarrow V_1$	0	0	0
2	$V_1 \rightarrow V_{11}$	6	6	0
3	$V_1 \rightarrow V_{15}$	7	7	0
⋮	⋮	⋮	⋮	⋮
959	$V_{31} \rightarrow V_{19}$	4	4	0
960	$V_{31} \rightarrow V_{24}$	9	9	0
961	$V_{31} \rightarrow V_{27}$	12	12	0
		Total Selisih		**0**

Hasil menunjukkan bahwa lintasan terpendek yang diperoleh dari metode pembobotan ulang sama dengan hasil dari Algoritma Bellman-Ford secara langsung. Hal ini membuktikan bahwa transformasi bobot pada Algoritma Johnson tidak mengubah struktur optimal dari lintasan terpendek.

4 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan, dapat disimpulkan bahwa pembobotan ulang pada graf berbobot negatif dapat dilakukan dengan menggunakan Algoritma Johnson, sehingga Algoritma Dijkstra dapat diterapkan secara efektif untuk menentukan lintasan terpendek. Hasil penerapan Algoritma Dijkstra menunjukkan bahwa lintasan terpendek tetap dapat ditemukan secara akurat, sebagaimana ditunjukkan pada graf pertama dari simpul V_10 ke V_31melalui lintasan $V_10 \rightarrow V_22 \rightarrow V_15 \rightarrow V_28 \rightarrow V_19 \rightarrow V_31$ dengan total bobot yaitu 16. Sementara itu, pada graf kedua dari simpul V_15 ke V_31 melalui lintasan $V_15 \rightarrow V_26 \rightarrow V_28 \rightarrow V_20 \rightarrow V_17 \rightarrow V_31$ dengan total bobot yaitu 1. Dengan demikian, pembobotan ulang terbukti efektif dalam mengatasi kendala bobot negatif pada graf tanpa mengubah struktur solusi optimal. Selain itu, hasil lintasan terpendek yang diperoleh setelah dilakukan pembobotan ulang menggunakan Algoritma Johnson memiliki kesesuaian yang sempurna dengan hasil lintasan yang dihitung langsung menggunakan Algoritma Bellman-Ford. Hal ini menunjukkan bahwa metode pembobotan ulang tidak memengaruhi keakuratan hasil perhitungan, namun memungkinkan pemanfaatan Algoritma Dijkstra yang lebih efisien. Dengan demikian, pendekatan ini tidak hanya efektif secara teoritis, tetapi juga terbukti secara praktis dalam menjaga keakuratan dan efisien pencarian lintasan terpendek pada graf berbobot negatif.

Pernyataan Kontribusi Penulis (CRediT)

Penulis Pertama: Konseptualisasi, Metodologi, Penulisan–Draf Awal.
Penulis Kedua: Kurasi Data, Analisis Formal, Penulisan–Telaah dan Penyuntingan.

Penulis Ketiga: Perangkat Lunak, Validasi, Visualisasi.
Penulis Keempat: Supervisi, Administrasi Proyek, Perolehan Pendanaan.

Deklarasi Penggunaan AI atau Teknologi Berbasis AI

Tidak ada teknologi AI generatif atau berbasis AI yang digunakan dalam penulisan naskah ini.

Deklarasi Konflik Kepentingan

Penulis menyatakan tidak ada konflik kepentingan.

Pendanaan dan Ucapan Terima Kasih

Penelitian ini tidak menerima pendanaan eksternal. Penulis mengucapkan terima kasih kepada pihak yang telah memberikan dukungan dan masukan selama penelitian berlangsung.

Ketersediaan Data

Data pendukung penelitian ini tersedia atas permintaan yang wajar kepada penulis korespondensi.

Daftar Pustaka

[1] D. T. Salaki, "Penentuan lintasan terpendek dari fmipa ke rektorat dan fakultas lain di unsrat manado menggunakan algoritma dijkstra," *Jurnal Ilmu Sains*, vol. 11, no. 1, pp. 73–76, 2011.

[2] A. F. Aji and W. Gozali, *Pemrograman Kompetitif Dasar: Panduan Memulai OSN Informatika, ACM-ICPC, dan Sederajat*. CV. Nulisbuku Jendela Dunia, 2017. Available online.

[3] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *Journal of the Association for Computing Machiner*, vol. 24, no. 1, pp. 1–13, 1997.

[4] M. Ilhamsyah, R. Mangkudjaja, and S. N. Hertiana, "Simulasi dan uji kinerja algoritma johnson untuk penentuan rute terbaik pada jaringan software defined network," *Jurnal Sistem Komputer*, vol. 6, no. 2, pp. 77–79, 2016.

[5] A. Frieze and M. Karonski, *INTRODUCTION TO RANDOM GRAPHS*. n.p., 2024.

[6] M. C. Bunaen, H. Pratiwi, and Y. F. Riti, "Penerapan algoritma dijkstra untuk menentukan ruteterpendek dari pusat kota surabaya ke tempat bersejarah," *Jurnal Teknologi Dan Sistem Informasi Bisnis*, vol. 4, no. 1, pp. 213–223, 2022. Available online.

[7] G. Chartrand, L. Lesniak, and P. Zhang, *Graphs Digraphs (5th ed.)* Taylor Francis Group, 2011.

[8] S. Sunardi, A. Yudhana, and A. A. Kadim, "Implementasi algoritma dijkstra untuk analisis rute transportasi umum transjogja berbasis android," *Jurnal Sistem Informasi Bisnis*, vol. 9, no. 1, pp. 32–38, 2019. DOI: https://doi.org/10.21456/vol9iss1pp32-38

[9] Y. Retanto, "Algoritma dijkstra dan bellman-ford dalam pencarian jalur terpendek," *Institut Teknologi Bandung*, 2009.

[10] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*. n.p., 2006.

[11] D. J. Bawole and H. P. Chernovita, "Algoritma bellman-ford untuk menentukan jalur terpendek dalam survey klaim asuransi (studi kasus : Pt. asuransi sinar mas, jakarta)," *INOBIS: Jurnal Inovasi Bisnis dan Manajemen Indonesia*, vol. 3, no. 1, pp. 41–51, 2019.

[12] J. Kleinberg and E. Tardos, *Algorithm Design*. Pearson Education, Inc., 2005.

[13] S. Rahayuainglih, *Teori Graph dan Penerapannya*. Unidha Press, 2018. Available online.

[14] R. Sedgewick and K. Wayne, *Algorithms Fourth Edition, 4th ed.* Pearson Education, Inc., 2011.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms Third Edition*. Massachusetts Institute of Technology, 2009.