Jurnal Teknik Informatika (JUTIF)
Vol. 3, No. 1, Februari 2022, hlm. 91-98

DOI : https://doi.org/10.20884/1.jutif.2022.3.1.151
p-ISSN: 2723-3863
e-ISSN: 2723-3871

PARALLEL COMPUTATION DESIGN OF DETERMINANT AND INVERSE OF A MATRICE USING COMPUTATIONAL GEOMETRY ANALYSIS TECHNIQUE

Eddy Maryanto*[1], Yogiek Indra Kurniawan[2]

[1,2]Jurusan Informatika, Fakultas Teknik, Universitas Jenderal Soedirman, Indonesia
Email: [1]eddy.maryanto@unsoed.ac.id, [2]yogiek@unsoed.ac.id

(Naskah masuk: 8 Februari 2022, Revisi : 16 Februari 2022, diterbitkan: 25 Februari 2022)

Abstract

The aim of this research is to study the use of computational geometry analysis technique in detail to design a parallel computation to determine determinant and inverse of a matrice using Gauss-Jordan method. In comparing with dependence graph technique, it is well known that computational geometry analysis gives us an easier way in determining determinant and inverse of a matrice, moreover for handling a computation domain with dimension of higher than 3. The type of scheduling function used in this research is of linear type. Some important facts are discovered in this research, the most important one is, this technique gives us the ease in arranging data scheduling scheme based on the observation of the computation domain. The other important fact is, it is also give us the ease to understand the scheme of the data scheduling since this technique can be collaborated with dependence graph technique for visualizing the data scheduling.

Keywords: *computation domain, computational geometry analysis technique, dependence acyclic graph, Gauss-Jordan method, and parallel computation.*

PERANCANGAN KOMPUTASI PARALEL DETERMINAN DAN BALIKAN MATRIKS MENGGUNAKAN TEKNIK ANALISIS GEOMETRI KOMPUTASIONAL

Abstrak

Pada penelitian ini dilakukan studi teknik analisis geometri komputasional untuk perancangan komputasi paralel untuk perhitungan determinan dan balikan matriks menggunakan metode Gauss-Jordan. Teknik analisis geometri komputasional memberikan kemudahan terutama untuk masalah dengan domain komputasi berdimensi lebih dari 3, apabila dibandingkan dengan teknik graf dependensi. Fungsi penjadwalan yang digunakan adalah fungsi linier. Berdasarkan hasil studi diperoleh kesimpulan bahwa perancangan komputasi paralel untuk masalah perhitungan determinan dan balikan matriks dengan menggunakan teknik analisis geometri komputasional memberikan kemudahan dalam pembuatan penjadwalan data, dan dikolaborasikan dengan teknik graf dependensi menjadikan penjadwalan data dapat divisualisasi sehingga memudahkan untuk memahami skema penjadwalan.

Kata kunci: *domain komputasi, fungsi, graf asiklis berarah, metode Gauss-Jordan, komputasi paralel, dan teknik analisis geometri komputasional.*

1. PENDAHULUAN

Sejalan dengan perkembangan ilmu pengetahuan dan teknologi yang semakin kompleks, serta kebutuhan pemrosesan data dengan volume yang besar, maka kebutuhan akan komputasi berkinerja tinggi merupakan kebutuhan mutlak. Salah satu upaya untuk menyediakan komputasi berkinerja tinggi adalah dengan komputasi paralel. Motivasi dari komputasi paralel adalah semakin banyak hal-hal yang dapat dilakukan secara simultan (dilakukan pada waktu yang bersamaan), maka semakin banyak perkerjaan yang terselesaikan [1].

Faktor utama untuk meningkatkan kinerja dari komputasi paralel terletak pada bagaimana mendesain teknik penjadwalan yang inovatif yang bisa memaksimalkan penggunaan unit pemroses [2]. Selain masalah penjadwalan pekerjaan, upaya untuk meningkatkan kinerja komputasi paralel juga bisa dilakukan dengan cara menentukan granularitas optimal dari pekerjaan [3]. Pada saat ini ada banyak jenis komputer yang memiliki multi-processor atau multicore yang memungkinkan komputer bisa bekerja lebih cepat, akan tetapi pemrosesan paralel yang memanfaatkan pemrograman paralel juga bisa

digunakan untuk meningkatkan kinerja komputasi [4]. Kinerja komputasi saat ini bisa ditingkatkan dengan diciptakannya komputer *multi-core* yang memungkinkan untuk dilakukannya komputasi paralel. Tanpa komputasi paralel, prosesor *multi-core* tidak mungkin dapat dimanfaatkan secara maksimal [5]. Pada komputasi paralel, sebuah program dipecah menjadi bagian-bagian dan kemudian dieksekusi secara simultan [6]. Pada penelitian ini, upaya yang dilakukan dalam rangka menciptakan komputasi berkinerja tinggi yaitu dengan cara mengkonversi algoritma sekuensial menjadi algoritma paralel menggunakan teknik analisis geometri komputasional.

2. METODE PENELITIAN

Penggunaan perkalian dan balikan dari matriks berukuran ultra sebagai operasi dasar pada berbagai bidang semakin meningkat, termasuk *big data* [7][8][9]. Ada banyak metode untuk menentukan balikan matriks, pada penelitian ini metode yang digunakan untuk menentukan balikan matriks didasarkan pada operasi baris elementer.

Masalah utama lainnya pada sistem komputasi paralel adalah penjadwalan pekerjaan dikarenakan adanya dependensi waktu eksekusi antar pekerjaan. Terdapat beberapa metode yang dapat digunakan untuk penjadwalan pekerjaan, antara lain *Genetic Algorithm based task scheduling* (GATS), *Master-Slave Genetic Algorithm* (MSGA), *Contention-aware task scheduling with task Duplication* (CA-D), *Node Duplication genetic algorithm* (NGA), dan *Real time controlled Duplication Algorithm* (RTCDA) [10]. Pada *cloud computing* atau *grid computing* penjadwalan pekerjaan bertujuan mendistribusikan beberapa pekerjaan besar yang ada tanpa dipecah menjadi pekerjaan yang lebih kecil dan independen untuk diselesaikan menggunakan sumberdaya yang tersedia dan digunakan bersama secara efisien [11][12]. Sedangkan pada komputasi paralel, sebuah pekerjaan besar diupayakan untuk dapat diselesaikan secepat mungkin dengan cara memecah pekerjaan menjadi pekerjaan yang lebih kecil dan independen sehingga bisa diproses secara paralel menggunakan sumberdaya yang tersedia. Pada penelitian ini, penjadwalan pekerjaan dilakukan pada sebuah sistem komputasi paralel dan proses penjadwalan pekerjaan dilakukan dengan menggunakan fungsi linier, yang mana vektor penjadwalan ditentukan berdasarkan observasi terhadap permasalahan yang dihadapi, dan indeks waktu penjadwalan pekerjaan direpresentasikan dalam bentuk *Directed Acyclic Graph* (DAG).

Convex hull pada penelitian ini digunakan untuk mendefinisikan domain komputasi dari algoritma. Komputasi *convex hull* merupakan isu fundamental pada berbagai bidang seperti komputasi geometri, grafik komputer, dan visi komputer [13][14].

Langkah-langkah yang digunakan dalam perancangan komputasi paralel dengan geometri komputasi [15] dalam penelitian ini adalah sebagai berikut:

(1) Nyatakan formula matematis yang ada pada masalah sebagai algoritma iteratif reguler
(2) Definisikan domain komputasi D berdasarkan indeks pada algoritma iteratif reguler.
(3) Identifikasikan matriks dependensi **A** dari tiap peubah. Selanjutnya, berdasarkan matriks dependensi dapat ditentukan vektor null dari matriks tersebut yang merepresentasikan subdomain *broadcast B* dari peubah. Identifikasikan juga perpotongan antara *B* dan D, yang mana perpotongan ini dapat membantu untuk mensuplai input atau mengekstrak output. Pada tahapan ini kita dapat mengambil keputusan apakah akan melakukan *broadcast* atau *pipeline* terhadap peubah.
(4) Melakukan penjadwalan atau *scheduling* data
Fungsi yang sering digunakan pada penjadwalan adalah fungsi linier. Fungsi penjadwalan harus memenuhi 5 syarat sebagai berikut

Indeks waktu positif: $\mathbf{sp} \geq s, \forall \mathbf{p} \in \mathrm{D}$ (1)

Restriksi *broadcast*: $\mathbf{se} = 0$ (2)

Restriksi *pipelining*: $\mathbf{sf} \neq 0$ (3)

Restriksi proyeksi: $\mathbf{sd} \neq 0$ (4)

Kausalitas: $\mathbf{sR} > 0$ (5)

dimana **e** adalah vektor null *broadcast*, **f** adalah vektor null pipelining, **d** adalah arah proyeksi, dan **R** adalah sinar ekstremal dari D.

(5) Melakukan proyeksi domain D pada domain D'.

3. HASIL DAN PEMBAHASAN

3.1 PERANCANGAN KOMPUTASI PARALEL PERHITUNGAN DETERMINAN MATRIKS

(a) Menghitung Determinan Matriks Menggunakan Metode Gauss-Jordan

Sebelum menentukan balikan dari sebuah matriks kita harus terlebih dahulu memeriksa apakah matriks tersebut memiliki invers atau tidak. Untuk mengetahui apakah sebuah matriks memiliki balikan atau tidak dilakukan dengan cara menghitung determinannya. Jika sebuah matriks memiliki determinan yang bernilai nol, maka matriks tersebut tidak memiliki balikan. Ada beberapa metode yang dapat digunakan untuk menghitung determinan dari sebuah matriks. Pada penelitian ini untuk menghitung determinan dari sebuah matriks digunakan metode Gauss-Jordan. Metode ini menerapkan operasi baris elementer (OBE) untuk menghitung determinan matriks. OBE digunakan untuk mengubah sebuah matriks menjadi matriks segitiga (atas atau bawah) yang mana proses ini disebut proses triangularisasi. Determinan matriks diperoleh dengan mengalikan elemen-elemen matriks yang ada pada diagonal utama dari matriks

segitiga. Berikut ini diberikan contoh matriks segitga atas (kiri) dan matriks segitiga bawah (kanan) dengan orde 4, yang mana × merepresentasikan elemen-elemen matriks.

$$\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \qquad \begin{bmatrix} \times & 0 & 0 & 0 \\ \times & \times & 0 & 0 \\ \times & \times & \times & 0 \\ \times & \times & \times & \times \end{bmatrix}$$

(b) Algoritma Gauss-Jordan untuk Menghitung Determinan Matriks

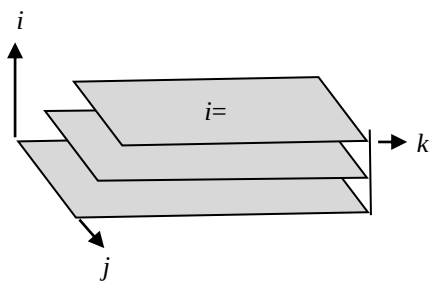
Berikut ini disajikan algoritma Gauss-Jordan untuk menghitung determinan dari sebuah matriks dengan mengubah matriks menjadi matriks segitiga atas dan pada saat yang bersamaan mengalikan elemen-elemen pada diagonal utama untuk memperoleh determinan.

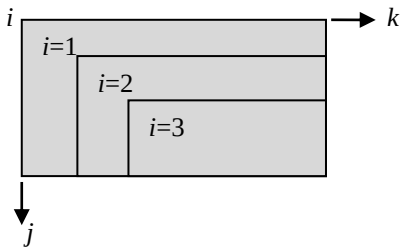
Algortima 1 Metode Gauss-Jordan untuk menghitung determinan dari matriks persegi M

Kebutuhan Input: Matriks persegi **M** dengan orde N, $m_{11} \neq 0$

1: $det = m_{11}$;
2: **for** $i = 1 : N-1$ **do**
3: **if(** $m_{ii} \neq 0$ **){**
4: **for** $j = i+1 : N$ **do**
5: $t = m_{ji}$;
6: **for** $k = i : N$ **do**
7: $m_{jk} = m_{jk} - (t / m_{ii}).m_{ik}$;
8: **end for**
9: $det = det.m_{(i+1)(i+1)}$;
10: **end for**
11: **else {** $det = 0$;
12: $i = N$;**}**
13: **end if**
14: end for
15: print("$|\mathbf{M}| =$", det**);**

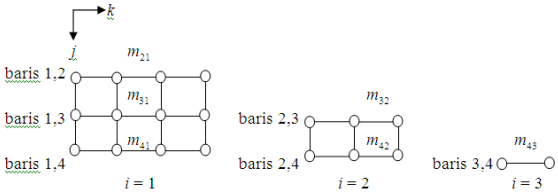
Algoritma 1 memiliki dimensi 3 dengan indeks $1 \le i \le N-1$, $i+1 \le j \le N$, dan $i \le k \le N$. *Convex hull* yang mendefinisikan domain komputasi dari algoritma 1 berbentuk piramid sebagaimana diperlihatkan pada Gambar 1. Gambar 1 memperlihatkan *bird's eye* dan *plan views* dari struktur berlapis graf dependensi dari algoritma triangularisasi matriks untuk matriks berorde 4.





Gambar 1 *Bird's eye* dan *plan views* struktur berlapis graf dependensi dari algoritma triangularisasi matriks untuk matriks berorde 4

Gambar 2 memperlihatkan detil dari graf dependensi lapisan demi lapisan untuk matriks dengan orde 4. Dalam hal ini kita membayangkan graf dependensi sebagai dependensi graf dua dimensi ganda agar mudah untuk divisualisasikan. Jika kita amati maka kita dapatkan fakta bahwa peubah m_{ji} merambat pada arah j dan peubah m_{ik} merambat pada arah k. Peubah m_{ik} ini merepresentasikan suatu elemen pada baris teratas pada setiap iterasi, dimana peubah ini digunakan untuk menerapkan OBE pada baris-baris di bawahnya. Indeks i menunjukkan posisi iterasi, dan nilai terbesar dari indeks i menunjukkan jumlah iterasi yaitu sebanyak $N-1$.



Gambar 2 Graf dependensi dari algoritma triangularisasi matriks untuk matriks berorde 4

(c) Fungsi Penjadwalan Proses Triangularisasi Matriks

Vektor penjadwalan **s** yang memberikan nilai indeks waktu kepada setiap titik pada domain komputasi berbentuk

$$t(\mathbf{p}) = \mathbf{sp} = s_1 i + s_2 j + s_3 k \qquad (6)$$

Nilai-nilai s_1, s_2 dan s_3 akan ditentukan dengan melakukan beberapa observasi. Berdasarkan observasi, pertama-tama kita peroleh fakta bahwa pada setiap iterasi *i*, baris-baris yang ada bisa kita proses secara paralel, sehingga kita peroleh

$$t(i, j, k) > t(i, j+1, k) \text{ atau}$$
$$t(i, j, k) < t(i, j+1, k) \text{ atau}$$
$$t(i, j, k) = t(i, j+1, k) \qquad (7)$$
$$\text{dan } s_2 = 0, \pm 1 \qquad (8)$$

Selanjutnya, kita peroleh fakta bahwa titik-titik pada sebuah baris bisa kita proses secara paralel, sehingga kita peroleh

$t(i,j,k) > t(i,j,k+1)$ atau

$t(i,j,k) < t(i,j,k+1)$ atau

$$t(i,j,k) = t(i,j,k+1) \quad (9)$$

dan $s_3 = 0, \pm 1$ (10)

Terakhir, kita peroleh fakta bahwa iterasi ke-$(i+1)$ hanya bisa diproses setelah pemrosesan iterasi ke-i selesai, sehingga kita peroleh

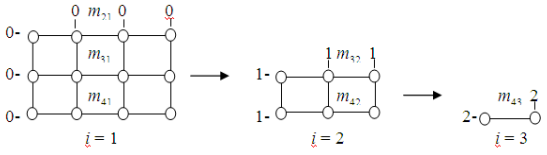
$$t(i+1,j,k) > t(i,j,k) \quad (11)$$

dan $s_1 = 1$ (12)

Berdasarkan (8), (10) dan (12), kita peroleh 9 kemungkinan vektor penjadwalan yaitu

$$\mathbf{s}_1 = [1 \quad 1 \quad 0] \quad (13)$$
$$\mathbf{s}_2 = [1 \quad 0 \quad 0] \quad (14)$$
$$\mathbf{s}_3 = [1 \quad -1 \quad 0] \quad (15)$$
$$\mathbf{s}_4 = [1 \quad 1 \quad -1] \quad (16)$$
$$\mathbf{s}_5 = [1 \quad 0 \quad -1] \quad (17)$$
$$\mathbf{s}_6 = [1 \quad -1 \quad -1] \quad (18)$$
$$\mathbf{s}_7 = [1 \quad 1 \quad 1] \quad (19)$$
$$\mathbf{s}_8 = [1 \quad 0 \quad 1] \quad (20)$$
$$\mathbf{s}_9 = [1 \quad -1 \quad 1] \quad (21)$$

Kita bisa bebas memilih salah satu dari 9 vektor penjadwalan yang ada, akan tetapi berdasarkan pertimbangan kesederhanaan visualisasi dari fungsi penjadwalan maka dipilih vektor penjadwalan $\mathbf{s}_2 = [1 \quad 0 \quad 0]$. Graf asiklis berarah dari fungsi penjadwalan untuk matriks berorde 4 disajikan pada Gambar 3 berikut ini.



Gambar 3 Graf asiklis berarah dari fungsi penjadwalan $\mathbf{S}_2$ untuk matriks berorde 4

Berdasarkan Gambar 3 kita ketahui bahwa apabila pemroses yang tersedia dalam jumlah yang mencukupi yaitu sebanyak N^2 maka kita dapat memaksimalkan *speedup*. Akan tetapi apabila jumlah pemroses yang kita miliki terbatas, maka pemrosesan paralel bisa kita terapkan antar baris matriks.

(d) Arah Proyeksi Triangularisasi Matriks

Sebelum menentukan matriks proyeksi terlebih dahulu harus ditentukan arah proyeksi **d**. Arah proyeksi ditentukan berdasarkan vektor penjadwalan $\mathbf{s}_2 = [1 \quad 0 \quad 0]$yang sudah kita pilih sebagai vektor penjadwalan, akan tetapi harus mengikuti ketentuan bahwa arah proyeksi tidak boleh tegak lurus terhadap vektor penjadwalan, sehingga diperoleh 3 kemungkinan arah proyeksi yaitu

$$\mathbf{d}_1 = [1 \quad 0 \quad 0]^T \quad (22)$$
$$\mathbf{d}_2 = [1 \quad 0 \quad 1]^T \quad (23)$$
$$\mathbf{d}_3 = [1 \quad 1 \quad 0]^T \quad (24)$$

Berdasarkan pertimbangan kesederhanaan, kita pilih $\mathbf{d}_1 = [1 \quad 0 \quad 0]^T = \mathbf{d}$ sebagai arah proyeksi, yang mana dengan pemilihan $\mathbf{d} = [1 \quad 0 \quad 0]^T$ sebagai arah proyeksi, maka semua titik yang terletak pada arah sumbu *i* akan dipetakan ke domain komputasi D'. Selanjutnya kita tentukan 3 buah vektor basis untuk domain komputasi D, yang mana salah satunya adalah $\mathbf{d} = [1 \quad 0 \quad 0]^T$, sedangkan 2 vektor basis lainnya harus tegak lurus terhadap $\mathbf{d}_1 = [1 \quad 0 \quad 0]^T$, sehingga kita peroleh vektor-vektor basis sebagai berikut

$$\mathbf{b}_0 = [1 \quad 0 \quad 0]^T = \mathbf{d} \quad (25)$$
$$\mathbf{b}_1 = [0 \quad 1 \quad 0]^T \quad (26)$$
$$\mathbf{b}_2 = [0 \quad 0 \quad 1]^T \quad (27)$$

Selanjutnya, karena proses perhitungan pada titik-titik baik pada arah *j* maupun *k* pada sebuah iterasi bisa dilakukan secara paralel, maka untuk mendapatkan matriks proyeksi **P**, kita gunakan persamaan matriks sebagai berikut

$$\mathbf{Pb} = \mathbf{b}'$$

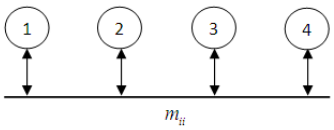
$$\Leftrightarrow [p_1 \quad p_2 \quad p_3] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [1 \quad 0 \quad 0]$$

$$\Leftrightarrow [p_1 \quad p_2 \quad p_3] = [1 \quad 0 \quad 0]$$

Jadi kita peroleh matriks proyeksinya adalah

$$\mathbf{P} = [1 \quad 0 \quad 0] \quad (27)$$

Matriks proyeksi **P** akan memetakan setiap titik $\mathbf{p} = [i \quad j \quad k]^T \in D$ ke titik $\mathbf{p}' = [i] \in$ D'. Gambar 3 memperlihatkan graf asiklis tereduksi dari graf asiklis pada Gambar 2. Penerapan proyeksi **P** menghasilkan implementasi berbasis kolom dan baris, yang mana pada setiap iterasi pemroses beroperasi pada setiap titik dan dilakukan *broadcast* terhadap elemen diagonal m_{ii} ke semua pemroses.



Gambar 3 Graf asiklis tereduksi untuk matriks berorde 4 dengan fungsi penjadwalan $\mathbf{s}_2 = [1 \quad 0 \quad 0]$ dan arah proyeksi $\mathbf{P} = [1 \quad 0 \quad 0]$

3.2 PERANCANGAN KOMPUTASI PARALEL PERHITUNGAN BALIKAN MATRIKS

(a) Menghitung Balikan Matriks Menggunakan Metode Gauss-Jordan

Pada penelitian ini, untuk menghitung balikan matriks digunakan metode Gauss-Jordan yang pada dasarnya menggunakan OBE. Untuk menentukan

balikan matriks berorde *N*, pertama-tama dibentuk sebuah matriks yang merupakan penggabungan antara matriks yang akan ditentukan balikannya dengan matriks identitas berorde *N* sehingga matriks gabungan kedua matriks memiliki dimensi $N \times 2N$. Selanjutnya diterapkan OBE pada matriks gabungan sedemikian sehingga matriks asal (sebelah kiri) dengan elemen-elemen yang direpresentasikan dengan $\times$ berubah menjadi matriks identitas, dan separuh matriks gabungan pada bagian kanan dengan elemen-elemen yang direpresentasikan dengan $\otimes$ merupakan matriks balikannya, sebagaimana diperlihatkan pada gambar berikut ini untuk matriks berorde 4.

$$\left[\begin{array}{cccc|cccc} \times & \times & \times & \times & 1 & 0 & 0 & 0 \\ \times & \times & \times & \times & 0 & 1 & 0 & 0 \\ \times & \times & \times & \times & 0 & 0 & 1 & 0 \\ \times & \times & \times & \times & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & \otimes & \otimes & \otimes & \otimes \\ 0 & 1 & 0 & 0 & \otimes & \otimes & \otimes & \otimes \\ 0 & 0 & 1 & 0 & \otimes & \otimes & \otimes & \otimes \\ 0 & 0 & 0 & 1 & \otimes & \otimes & \otimes & \otimes \end{array}\right]$$

(b) Algoritma Gauss-Jordan untuk Menentukan Balikan Matriks

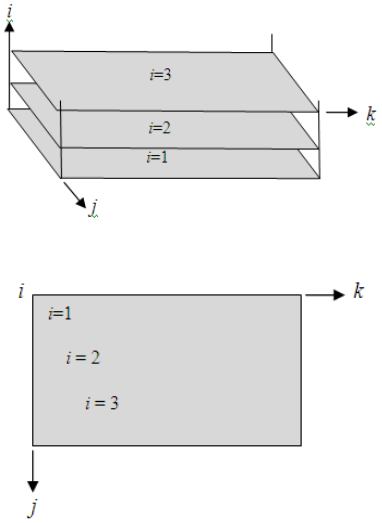
Berikut ini disajikan algoritma Gauss-Jordan untuk menentukan balikan dari sebuah matriks dengan menerapkan OBE sedemikian sehingga matriks asal yang akan ditentukan balikannya berubah menjadi matriks identitas.

Algortima 2 Metode Gauss-Jordan untuk menentukan balikan dari matriks M

Kebutuhan Input: Matriks **M** dengan orde $N \times 2N$, $m_{11} \neq 0$

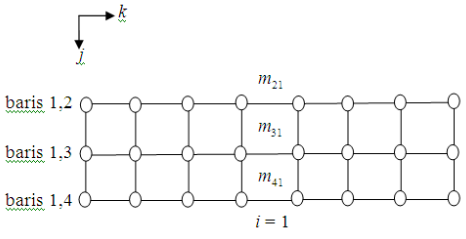
1: **for** $i = 1:N$ **do**
2: $a = m_{ii}$;
3: **for** $l = 1:2N$ **do**
4: $m_{il} = m_{il} / a$;
5: **end for**
6: **for** $j = 1:N$ **do**
7: **if(** $j \neq i$ **){** $b = m_{ji}$;
8: **for** $k = 1:2N$ **do**
9: $m_{jk} = m_{jk} - b.m_{ik}$;
10: **end for}**
11: **end if**
12: **end for**
13: **end for**
14: //Matriks **L** merupakan balikan dari matriks **M**
15: **for** $i = 1:N$ **do**
16: **for** $j = N+1:2N$ **do**
17: $l_{i(j-N)} = m_{ij}$;
18: **end for**
19: **end for**

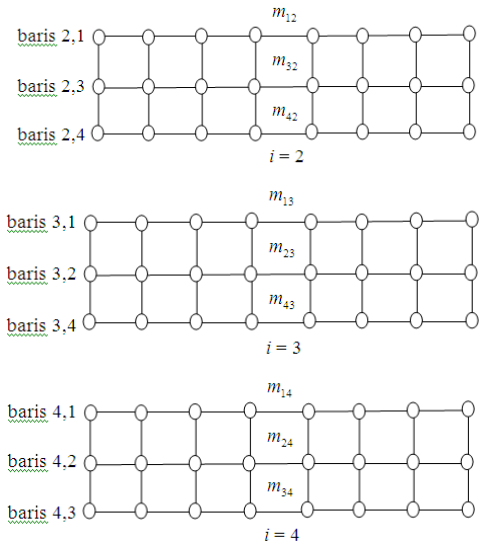
Algoritma 2 (pada bagian perhitungan balikan) memiliki dimensi 3 dengan indeks $1 \le i \le N$, $i \le j \le N$, dan $1 \le k \le 2N$. *Convex hull* [14][15] yang mendefinisikan domain komputasi dari algoritma 2 berbentuk balok memanjang sebagaimana diperlihatkan pada Gambar 4. Gambar 4 memperlihatkan *bird's eye* dan *plan views* dari struktur berlapis graf dependensi dari algoritma balikan matriks untuk matriks berorde 4.



Gambar 4 *Bird's eye* dan *plan views* struktur berlapis graf dependensi dari algoritma balikan matriks untuk matriks berorde 4

Gambar 5 memperlihatkan detil dari graf dependensi lapisan demi lapisan untuk matriks dengan orde 4. Dalam hal ini kita membayangkan graf dependensi sebagai dependensi graf dua dimensi ganda agar mudah untuk divisualisasikan. Jika kita amati maka kita dapatkan fakta bahwa peubah m_{ji} merambat pada arah j dan peubah m_{ik} merambat pada arah *k*. Peubah m_{ik} ini merepresentasikan suatu elemen pada baris kunci pada setiap iterasi, dimana peubah ini digunakan untuk menerapkan OBE pada baris-baris lainnya. Indeks *i* menunjukkan posisi iterasi, dan nilai terbesar dari indeks *i* menunjukkan jumlah iterasi yaitu sebanyak N.





Gambar 5 Graf dependensi dari algoritma perhitungan balikan matriks untuk matriks berorde 4

(c) Fungsi Penjadwalan Proses Perhitungan Balikan Matriks

Vektor penjadwalan **s** yang memberikan nilai indeks waktu kepada setiap titik pada domain komputasi berbentuk

$$t(\mathbf{p}) = \mathbf{sp} = s_1 i + s_2 j + s_3 k \quad (28)$$

Nilai-nilai s_1, s_2 dan s_3 akan ditentukan dengan melakukan beberapa observasi. Berdasarkan observasi, pertama-tama kita peroleh fakta bahwa pada setiap iterasi *i*, baris-baris yang ada bisa kita proses secara paralel, sehingga kita peroleh

$$t(i,j,k) > t(i,j+1,k) \text{ atau}$$
$$t(i,j,k) < t(i,j+1,k) \text{ atau}$$
$$t(i,j,k) = t(i,j+1,k) \quad (29)$$
$$\text{dan } s_2 = 0, \pm 1 \quad (30)$$

Selanjutnya, kita peroleh fakta bahwa titik-titik pada sebuah baris bisa kita proses secara paralel, sehingga kita peroleh

$$t(i,j,k) > t(i,j,k+1) \text{ atau}$$
$$t(i,j,k) < t(i,j,k+1) \text{ atau}$$
$$t(i,j,k) = t(i,j,k+1) \quad (31)$$
$$\text{dan } s_3 = 0, \pm 1 \quad (32)$$

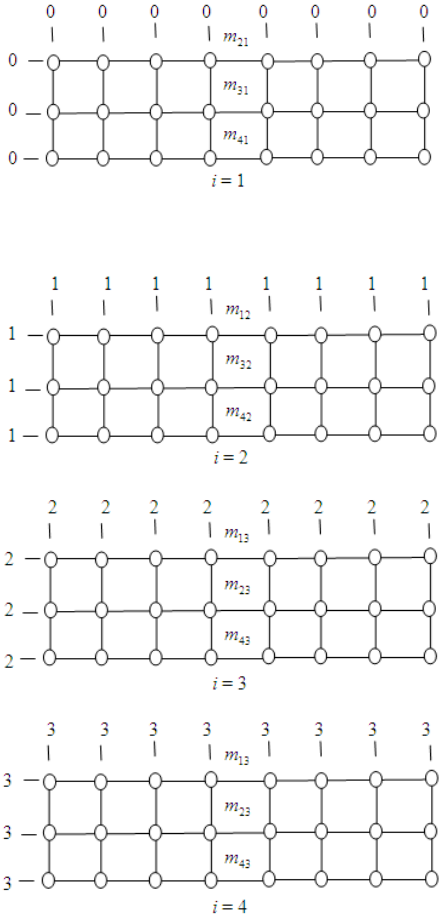
Terakhir, kita peroleh fakta bahwa iterasi ke-$(i+1)$ hanya bisa diproses setelah pemrosesan iterasi ke-i selesai, sehingga kita peroleh

$$t(i+1,j,k) > t(i,j,k) \quad (33)$$
$$\text{dan } s_1 = 1 \quad (34)$$

Berdasarkan (30), (32) dan (34), kita peroleh 9 kemungkinan vektor penjadwalan yaitu

$$\mathbf{s}_1 = [1 \quad 1 \quad 0] \quad (35)$$
$$\mathbf{s}_2 = [1 \quad 0 \quad 0] \quad (36)$$
$$\mathbf{s}_3 = [1 \quad -1 \quad 0] \quad (37)$$
$$\mathbf{s}_4 = [1 \quad 1 \quad -1] \quad (38)$$
$$\mathbf{s}_5 = [1 \quad 0 \quad -1] \quad (39)$$
$$\mathbf{s}_6 = [1 \quad -1 \quad -1] \quad (40)$$
$$\mathbf{s}_7 = [1 \quad 1 \quad 1] \quad (41)$$
$$\mathbf{s}_8 = [1 \quad 0 \quad 1] \quad (42)$$
$$\mathbf{s}_9 = [1 \quad -1 \quad 1] \quad (43)$$

Kita bisa bebas memilih salah satu dari 9 vektor penjadwalan yang ada, akan tetapi berdasarkan pertimbangan kesederhanaan visualisasi dari fungsi penjadwalan maka dipilih vektor penjadwalan $\mathbf{s}_2 = [1 \quad 0 \quad 0]$. Graf asiklis berarah dari fungsi penjadwalan untuk matriks berorde 4 disajikan pada Gambar 6 berikut ini.



Gambar 6 Graf asiklis berarah dari fungsi penjadwalan $\mathbf{S}_2$ untuk matriks berorde 4

Berdasarkan Gambar 6 kita ketahui bahwa apabila pemroses yang tersedia dalam jumlah yang mencukupi yaitu sebanyak $2N^2$ maka kita dapat memaksimalkan *speedup*. Akan tetapi apabila jumlah pemroses yang kita miliki terbatas, maka pemrosesan paralel bisa kita terapkan antar baris matriks pada setiap iterasi.

(d) Arah Proyeksi Perhitungan Balikan Matriks

Sebelum menentukan matriks proyeksi terlebih dahulu harus ditentukan arah proyeksi **d**. Arah proyeksi ditentukan berdasarkan vektor

penjadwalan $\mathbf{s}_2 = [1 \quad 0 \quad 0]$yang sudah kita pilih sebagai vektor penjadwalan, akan tetapi harus mengikuti ketentuan bahwa arah proyeksi tidak boleh tegak lurus terhadap vektor penjadwalan, sehingga diperoleh 3 kemungkinan arah proyeksi yaitu

$$\mathbf{d}_1 = [1 \quad 0 \quad 0]^T \tag{44}$$

$$\mathbf{d}_2 = [1 \quad 0 \quad 1]^T \tag{45}$$

$$\mathbf{d}_3 = [1 \quad 1 \quad 0]^T \tag{46}$$

Berdasarkan pertimbangan kesederhanaan, kita pilih $\mathbf{d}_1 = [1 \quad 0 \quad 0]^T = \mathbf{d}$ sebagai arah proyeksi, yang mana dengan pemilihan $\mathbf{d} = [1 \quad 0 \quad 0]^T$ sebagai arah proyeksi, maka semua titik yang terletak pada arah sumbu *i* akan dipetakan ke domain komputasi D'. Selanjutnya kita tentukan 3 buah vektor basis untuk domain komputasi D, yang mana salah satunya adalah $\mathbf{d} = [1 \quad 0 \quad 0]^T$, sedangkan 2 vektor basis lainnya harus tegak lurus terhadap $\mathbf{d}_1 = [1 \quad 0 \quad 0]^T$, sehingga kita peroleh vektor-vektor basis sebagai berikut

$$\mathbf{b}_0 = [1 \quad 0 \quad 0]^T = \mathbf{d} \tag{47}$$

$$\mathbf{b}_1 = [0 \quad 1 \quad 0]^T \tag{48}$$

$$\mathbf{b}_2 = [0 \quad 0 \quad 1]^T \tag{49}$$

Selanjutnya, karena proses perhitungan pada titik-titik baik pada arah *j* maupun *k* pada sebuah iterasi bisa dilakukan secara paralel, maka untuk mendapatkan matriks proyeksi **P**, kita gunakan persamaan matriks sebagai berikut

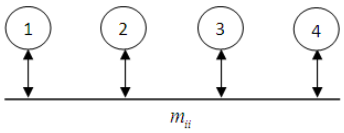
$$\mathbf{Pb} = \mathbf{b}'$$

$$\Leftrightarrow [p_1 \quad p_2 \quad p_3] = [1 \quad 0 \quad 0]$$

Jadi kita peroleh matriks proyeksinya adalah

$$\mathbf{P} = [1 \quad 0 \quad 0] \tag{50}$$

Matriks proyeksi **P** akan memetakan setiap titik $\mathbf{p} = [i \quad j \quad k]^T \in D$ ke titik $\mathbf{p}' = [i] \in$ D'. Gambar 7 memperlihatkan graf asiklis tereduksi dari graf asiklis pada Gambar 6. Penerapan proyeksi **P** menghasilkan implementasi berbasis kolom dan baris, yang mana pada setiap iterasi pemroses beroperasi pada setiap titik dan dilakukan *broadcast* terhadap elemen diagonal m_{ii} ke semua pemroses.



Gambar 7 Graf asiklis tereduksi untuk matriks berorde 4 dengan fungsi penjadwalan $\mathbf{s}_2 = [1 \quad 0 \quad 0]$ dan arah proyeksi $\mathbf{P} = [1 \quad 0 \quad 0]$

4. KESIMPULAN

Berdasarkan hasil penelitian ini dapat disimpulkan bahwa perancangan proses komputasi paralel dari perhitungan determinan dan balikan matriks berbasis metode Gauss-Jordan menjadi mudah untuk dilakukan dengan menggunakan teknik geometri komputasional. Skema penjadwalan data juga dapat dirancang dan direpresentasikan secara visual dengan baik, karena teknik ini juga bisa mengelaborasi graf asiklis berarah untuk memvisualisasi skema penjadwalan data sehingga lebih mudah untuk dipahami.

Pada penelitian selanjutnya, hasil perancangan komputasi paralel perhitungan determinan dan balikan matriks ini akan diimplementasikan dengan menggunakan teknik *multithread* dan *multiprocessor* dengan berbasiskan teknologi *remote method invocation* (RMI) pada bahasa pemrograman Java dengan tujuan untuk mengetahui pengaruh dari komputasi paralel terhadap *speedup*. Selain itu juga akan dilakukan penelitian tentang perancangan dan implementasi teknik ini untuk bidang lainnya seperti pemodelan ilmiah, pengolahan citra digital, dan basis data terdistribusi, termasuk meneliti pengaruh penggunaan fungsi penjadwalan non linier terhadap efektifitas penjadwalan data.

DAFTAR PUSTAKA

[1] E. Fernando, D.F. Murad, and B.D. Wijanarko, "Classification and Advantages Paralel in Process Computing: A Systematic Literature Review," *2018 4th International Conference on Computing, Engineering, and Design (ICCED)*, 2018, doi: 10.1109/ICCED.2018.00036.

[2] A. Ahmad, M. M. Yousaf, S. Sarwar, W.U. Qounain, L. Aslam, and M. Khali, "Optimized Scheduling for Parallel Computing Environment," *Sindh University Research Journal (SCIENCESERIES), SindhUniv. Res. Jour. (Sci. Ser.) Vol. 50 (002) 295-302 (2018*, 2018, doi: 10.26692/sujo/2018.06.0050.

[3] B. Bramas and A. Ketterlin, "Improving parallel executions by increasing task granularity in task-based runtime systems using acyclic DAG clustering," *PeerJ Comput. Sci.* 6:e247, pp. 22-47, 2020, doi:10.7717/peerj-cs.247.

[4] Rihartanto, A. Susanto, and A. Rizal, "Performance of Parallel Computing in Bubble Sort Algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, no. 3, pp. 861-866, 2017, doi: 10.11591/ijeecs.v7.i3.

[5] R. Pandey and N. Badal, "Understanding the Role of Parallel Programming in Multi-core Processor-based Systems," *2nd International Conference on Advanced Computing and Software Engineering (ICACSE-2019)*, 2019, pp. 365-368. https://ssrn.com/abstract=3350311.

[6] A.M. Bhugul, "Parallel Computing Using Open MP," *International Journal of Computer Science and Mobile Computing*, 2017, vol.6, issue.2, pp. 90-94.

[7] H. Wang, Y. Guo and H. Guo , "A Method of Ultra-Large-Scale Matrix Inversion Using Block Recursion," *Information Journal*, pp. 523-527, 2020, doi:10.3390/info11110523.

[8] E.G. Almeida, Y. B. Moreno, Y.H. Erfano, M. Vera, M` Mora, and R. Barrientos, "Parallel methods for linear systems solution in extreme learning machines: an overview," *Journal of Physics: Conference Series*, 1702 (2020) 012017, 2020, doi:10.1088/1742-6596/1702/1/012017.

[9] H. Singh, D. Chander, and R. Bhatt, "Parallel Computing of Matrix Multiplication in OPEN MP Supported Codeblocks," *Advances and Applications in Mathematical Sciences*, vol. 18, issue 8, Pages 775-787, 2019, Mili Publications.

[10] R. Singh, "Task Scheduling Techniques Based On Parallel Genetic Algorithm Approaches : A Review," *International Journal of Computer Applications & Information Technology*, vol. 11, issue no. 1, pp.229-243, 2018, http://www.ijcait.com/IJCAIT/11/112.pdf.

[11] B. Anitha and G.K.M. Kamalam, "Heuristic Algorithm for Independent Task Scheduling in Grid Computing," *International Journal of Rescent Technology and Engineering (IJRTE)*, vol.8, no.4, pp.12861-12866, 2019, doi: 10.35940/ijrte.D9411. 118419.

[12] R.A.A. Arasi, A. Saif, "Task scheduling in cloud computing based on meta-heuristic techniques: A review paper," *EAI Endorsed Transactions on Cloud Systems*, vol. 6, issue 17, 2020, doi: 10.4108/eai.13-7-2018. 162829.

[13] R. Alshamrani, F. Alshehri , and H. Kurdi, "A Preprocessing Technique for Fast Convex Hull Computation," *Procedia Computer Science*, vol.170, pp.317–324, 2020, doi: 10.1016/j.procs.2020.03.046.

[14] A.N. Gamby and J. Katajainen, "Convex-Hull Algorithms: Implementation, Testing and Experimentation Convex-Hull Algorithms: Implementation, Testing and Experimentation," *MDPI Journal*, vol. 11, no. 195, pp. 747–775, 2018, doi:10.3390/a11120195.

[15] F. Gebali, *Algorithms and Parallel Computing*, 2011, John Wiley and Sons, USA.