JURNAL MATEMATIKA, STATISTIKA & KOMPUTASI E-ISSN: 2614-8811
https://journal.unhas.ac.id/index.php/jmsk/index
Vol. 18, No. 1, 22-30, September, 2021
DOI: 10.20956/j.v18i1.14003

Comparison of Serial and Parallel Computation on Predicting Missing Data with EM Algorithm

Perbandingan Komputasi Serial dan Paralel pada Pendugaan Data Hilang dengan *EM Algorithm*

Erna Nurmawati[1)], Robby Hasan Pangaribuan[2)], Ibnu Santoso[3)]

Abstract

One way to deal with the presence of missing value or incomplete data is to impute the data using EM Algorithm. The need for large and fast data processing is necessary to implement parallel computing on EM algorithm serial program. In the parallel program architecture of EM Algorithm in this study, the controller is only related to the EM module whereas the EM module itself uses matrix and vector modules intensively. Parallelization is done by using OpenMP in EM modules which results in faster compute time on parallel programs than serial programs. Parallel computing with 4 (four) thread or sub process increases speed up, reduces compute time, and reduces efficiency when compared to parallel computing by the number of threads 2 (two).

Keywords: Parallel Computing, Missing Data, EM Algorithm.

Abstrak

Salah satu cara untuk menangani adanya data hilang atau data yang tidak lengkap adalah dengan melakukan imputasi terhadap data tersebut dengan menggunakan *EM Algorithm*. Kebutuhan pemrosesan data dalam jumlah yang besar dan cepat perlu untuk menerapkan komputasi paralel pada program serial *EM Algorithm*. Pada arsitektur program paralel *EM Algorithm* dalam penelitian ini, *controller* hanya berhubungan dengan modul EM sedangkan modul EM sendiri menggunakan modul matrix dan vector secara intensif. Paralelisasi dilakukan dengan menggunakan OpenMP pada modul EM menghasilkan waktu komputasi pada program paralel lebih cepat daripada program serialnya. Komputasi paralel dengan jumlah *thread* atau sub proses sebanyak 4 (empat) lebih meningkatkan *speed up*, mengurangi lama waktu komputasi, dan mengurangi efisiensi jika dibandingkan dengan komputasi paralel dengan jumlah *thread* 2 (dua).

Kata kunci: Komputasi Paralel, Data Hilang, *EM Algorithm.*

1. PENDAHULUAN

Data yang dihasilkan dari rancangan percobaan, survei, dan sensus tidak selalu lengkap atau terdapat *missing data* (data hilang). Hal ini disebabkan karena terdapat permasalahan dalam proses

[*]*Politeknik Statistika STIS*
[**]*BPS Deli Serdang Sumatera Utara*
Email *: [1]erna.nurmawati@stis.ac.id, [2]robby@bps.go.id, [3]ibnu@stis.ac.id*

pengumpulan data seperti responden tidak bersedia memberikan jawaban dari butir pertanyaan tertentu pada survei, sampel rancangan percobaan (hewan atau tumbuhan) yang mati, alat pengukur yang rusak, dan data *outlier* yang dihilangkan karena kesalahan dalam input data [12] [8]. Kasus adanya data hilang hampir terjadi pada setiap penelitian sehingga penanganan terhadap data hilang tersebut menjadi sangat penting. Sering kali matrik data multivariat mengandung data hilang yang menghalangi penggunaan teknik standar yang ada sehingga data yang akan dianalisis harus lengkap atau tidak mengandung data hilang [5].

Jika data hilang terbukti random, dalam arti terjadi tidak sengaja dan tidak mengacu keadaan tertentu, maka berbagai perlakuan bisa dilakukan pada data-data yang hilang [13]. Ada sejumlah cara alternatif dalam penanganan data hilang, antara lain dengan *listwise deletion*, *pairwise deletion*, dan *EM algorithm* [9]. EM Algorithm menghasilkan estimasi data yang hilang paling dekat dengan nilai aslinya [7].

Dalam analisis multivariat, data umumnya didasarkan pada asumsi mengikuti peluang distribusi normal multivariat [12]. Jika *p* adalah jumlah variabel dan *n* adalah jumlah observasi maka fungsi peluang sebaran normal *p* -variat adalah sbb:

$$f(x) = \frac{1}{(2\pi)^{p/2}\left|\Sigma\right|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right] \qquad \text{j}$$

$$= 1,2,3,...,n$$

x dan μ pada fungsi di atas merupakan vektor data dan vektor rata-rata $p\times1$ dan Σ adalah sebuah matriks ragam-peragam yang berordo $p\times p$

$$\text{dimana } \Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{12} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1p} & \sigma_{2p} & \cdots & \sigma_{pp} \end{bmatrix} \qquad \text{dan } \sigma_{ik} = \frac{1}{n}\sum_{j=1}^{n}(X_{ij} - \overline{X}_i)(X_{kj} - \overline{X}_k)$$

Pendekatan umum untuk menduga data hilang atau data tidak lengkap yaitu data/nilai dari observasi yang hilang adalah dengan *Maksimum Likelihood Estimate* (MLE). Teknik ini disebut dengan *EM algorithm* yaitu penghitungan iterative (yang berulang) melibatkan dua tahap, yaitu tahap prediksi dan tahap estimasi [2].

Tahap prediksi dimulai dengan memberikan beberapa penduga $\tilde{\theta}$ dari parameter yang tidak diketahui yaitu dengan menghitung nilai rata-rata, ragam, dan peragam dari seluruh variabel yang kemudian dimasukkan dalam vektor rata-rata dan matrik ragam-peragam. Hal ini dilakukan guna memperkirakan kontribusi beberapa observasi yang hilang untuk statistik cukup data lengkap / *complete data sufficient statistics*. Tahap estimasi dilakukan setelah tahap prediksi dengan menggunakan perkiraan statistik cukup (*predicted sufficient statistic*) untuk menghitung penduga yang telah direvisi dari parameter.

Penghitungan berulang dari satu tahap ke tahap yang lainnya, sampai penduga yang telah direvisi tidak berbeda nilainya dari penduga yang didapat dari iterasi sebelumnya.,Jika beberapa observasi $X_1,X_2, \ldots, X_n$ adalah sampel acak dari populasi normal *p*-variate, algoritma prediksi dan estimasi bergantung pada *complete data sufficient statisic.*

$$T_1 = \sum_{j=1}^{n} X_j = n\overline{X}$$

dan

$$T_2 = \sum_{j=1}^{n} X_j X_j' = (n-1)S + n\overline{X}\overline{X}'$$

Dalam kasus ini, algoritma di proses sebagai berikut: adanya asumsi bahwa rata-rata populasi dan varians- μ dan Σ, masing-masing tidak diketahui dan harus diestimasi atau diduga. Pada tahap awal data yang hilang diisikan nilainya dengan rata-rata dari peubahnya. Berdasarkan data tersebut dicari vektor rata-rata dan matrik ragam-peragam (untuk data populasi) yang akan digunakan dalam tahap prediksi.

Selanjutnya pada tiap sampel atau observasi yang terdapat data hilang dilakukan partisi untuk $\tilde{\mu}$ dan $\tilde{\Sigma}$. Partisi ini dilakukan untuk mencari nilai prediksi dari data hilang tersebut. $\tilde{\mu}$ dipartisi menjadi dua, yaitu $\tilde{\mu}^{(1)}$ adalah vektor rata-rata dari peubah yang terdapat data hilang dan $\tilde{\mu}^{(2)}$ adalah vektor rata-rata dari peubah yang tidak terdapat data hilang.

$$\tilde{\mu} = \begin{bmatrix} \mu^{(1)} \\ \hline \mu^{(2)} \end{bmatrix} \qquad \text{dan} \qquad \tilde{\Sigma} = \left[\begin{array}{c:c} \tilde{\Sigma}_{11} & \tilde{\Sigma}_{12} \\ \hdashline \tilde{\Sigma}_{21} & \tilde{\Sigma}_{22} \end{array}\right]$$

a. Tahap Prediksi

Untuk masing-masing vektor x_j dengan data hilang, $x_j^{(1)}$ merupakan notasi dari komponen hilang dan $x_j^{(2)}$ adalah notasi dari komponen yang tersedia. Jadi $x_j = [x_j^{(1)\prime} x_j^{(2)\prime}]$. Selanjutnya penduga $\tilde{\mu}$ dan $\tilde{\Sigma}$ diberikan dari tahap estimasi dengan menggunakan rata-rata dari distibusi normal bersyarat dari $x_j^{(1)}$, dengan syarat $x_j^{(2)}$, untuk mengestimasi data hilang/*missing value*.

$$\tilde{x}_j^{(1)} = E(X_j^{(1)} \mid x_j^{(2)}; \tilde{\mu}, \tilde{\Sigma}) = \tilde{\mu}^{(1)} + \tilde{\Sigma}_{12}\tilde{\Sigma}_{22}^{-1}(x_j^{(2)} - \tilde{\mu}^{(2)}) \qquad (1.1)$$

Pers. 1 di atas digunakan untuk menduga kontribusi dari $x_j^{(1)}$ pada T_1.

Berikutnya, memperkirakan kontribusi dari $x_j^{(1)}$ pada T_2 yaitu :

$$\overbrace{x_j^{(1)} x_j^{(1)\prime}} = E(X_j^{(1)} X_j^{(1)\prime} \mid x_j^{(1)}; \tilde{\mu}, \tilde{\Sigma}) = \tilde{\Sigma}_{11} - \tilde{\Sigma}_{12}\tilde{\Sigma}_{22}^{-1}\tilde{\Sigma}_{21} + \tilde{x}_j^{(1)}\tilde{x}_j^{(1)\prime} \qquad (1.2)$$

dan

$$\overbrace{x_j^{(1)} x_j^{(2)\prime}} = E(X_j^{(1)} X_j^{(2)\prime} \mid x_j^{(1)}; \tilde{\mu}, \tilde{\Sigma}) = \tilde{x}_j^{(1)} x_j^{(2)\prime}$$

Kontribusi pada (1.1) dan (1.2) dijumlahkan pada semua x_j seluruhnya dengan komponen hilang. Hasilnya dikombinasikan dengan data sampel untuk menghasilkan $\tilde{T}_1$ dan $\tilde{T}_2$.

Jika semua data hilang untuk sebuah sampel maka prediksi untuk nilai $\tilde{x}_j = \tilde{\mu}$ dan $x_j x_j' = \tilde{\Sigma} + \tilde{\mu}\tilde{\mu}'$.

b. Tahap Estimasi

Menghitung revised MLE, sbb:

$$\tilde{\mu} = \frac{\tilde{T}_1}{n}, \qquad \tilde{\Sigma} = \frac{1}{n}\tilde{T}_2 - \tilde{\mu}\tilde{\mu}'$$

Proses iterasi antara tahap prediksi dan estimasi berlangsung terus sampai elemen dari $\tilde{\mu}$ dan $\tilde{\Sigma}$ menghasilkan perubahan yang tidak terlalu jauh. Sehingga pada tahap estimasi terakhir $\tilde{\mu}$ dan $\tilde{\Sigma}$ akan berubah $\hat{\mu}$ dan $\hat{\Sigma}$ yang merupakan penduga dari vektor rata-rata dan matrik ragam-peragam. Setelah iterasi mencapai konvergen maka nilai yang akan menggantikan data hilang pada data awal adalah nilai dari tahap prediksi pada iterasi terakhir.

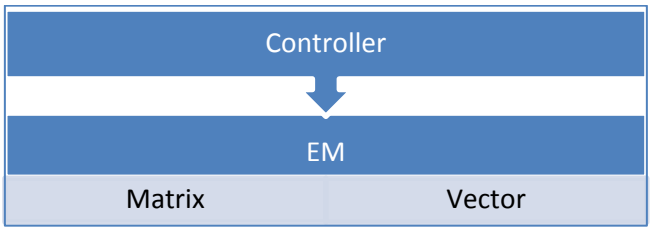
Penanganan data hilang khususnya dengan *EM algorithm* menggunakan penghitungan yang melibatkan operasi matrik yang sangat rumit sehingga membutuhkan tingkat ketelitian yang tinggi, waktu, dan tenaga untuk mendapatkan hasil yang benar jika dilakukan secara manual. Oleh karena itu, teori pendugaan data hilang dengan *EM Algorithm* pada penelitian ini akan dibuat dalam bentuk program aplikasi komputer.

Program komputer umumnya diproses oleh komputer secara berurutan dengan menggunakan satu prosesor atau disebut dengan komputasi serial. Seiring waktu, teknologi komputer multiprosesor banyak tersedia. Selain itu, kebutuhan pemrosesan data dalam jumlah yang sangat besar maka memerlukan cara komputasi baru sehingga dapat mempercepat proses komputasi. Algoritma pemrograman dalam komputasi serial dapat dipecah menjadi beberapa bagian sub proses (*thread*) dan dikerjakan secara bersamaan pada beberapa prosesor [14]. Sebuah *thread* adalah sebuah bagian dari program yang dapat berjalan mandiri, sehingga dua atau lebih *thread* dapat berjalan bersamaan, tanpa yang satu harus menunggu *thread* lain selesai [6]. Hal ini disebut dengan komputasi paralel yang dapat diterapkan untuk meningkatkan kecepatan proses komputasi [11]. Komputasi paralel dapat meningkatkan efisiensi waktu proses perhitungan sehingga penggunaan sumber daya komputasi lebih efektif dan efisien [1]. Untuk menghitung seberapa cepat proses komputasi paralel dibandingkan dengan komputasi serial maka digunakan *Speed Up* (S) dan *Efficiency* (E) [4]. Rumus untuk menghitung S dan E sebagai berikut $S = \frac{Ts}{Tp}$ dan $E = \frac{S}{P}$, dimana Ts adalah waktu yang dibutuhkan untuk komputas serial, Tp adalah waktu yang digunakan untuk komputasi paralel, dan p adalah jumlah prosesor. Efisiensi kinerja paralel menggunakan 2 *thread* diperoleh lebih baik daripada menggunakan 4 *thread*, dan 8 *thread* [10]. Hasil nilai Efesiensi yang didapat dari semua skenario pengujian membuktikan bahwa penggunaan dengan dua prosesor masih tergolong paling efesien dibandingkan empat prosesor [3].

2. HASIL PEMBAHASAN

Source code pada program aplikasi penanganan data hilang dengan metode *listwise*, *pairwise*, dan *EM Algorithm* pada penelitian sebelumya dikembangkan menggunakan bahasa pemrograman Visual Basic .Net 2005 secara serial yaitu eksekusi algoritma program dilakukan secara berurutan pada prosesor yang sama [9]. Pada penelitian ini, komputasi untuk menduga data hilang dengan *EM Algorithm* dilakukan secara paralel dengan menggunakan bahasa pemrograman C++. Oleh karena itu, kode program dengan algoritma yang dieksekusi serial juga akan diubah menjadi Bahasa pemrograman C++.

2.1. Pembuatan Program dengan Bahasa Pemrograman C++



Gambar 1. Arsitektur Program

Ada dua *project* program yang dibuat, yaitu untuk serial dan paralel. Kode program untuk serial dibuat terlebih dulu kemudian dilakukan analisis untuk menentukan bagian kode program yang perlu diparalelkan. Pada gambar arsitektur program di atas, *controller* hanya berhubungan dengan modul EM, sedangkan EM sendiri menggunakan modul Matrix dan Vector secara intensif.

Masing-masing *project* program memiliki modul yang sama. Hanya untuk *project* paralel disisipkan kode program untuk paralelisasi dengan OpenMP. Ada tiga bagian modul besar yang dibuat, yaitu: "matrix.h", "vector.h", dan "em.h". Modul yang lain adalah: "var.h" untuk variabel global untuk "matrix.h" dan "vector.h", modul "algebra.h" untuk mengkolaborasikan modul "matrix.h" dan "vector.h". Modul yang lain adalah "main.cpp" sebagai *controller* dalam program. Berikut adalah daftar modul seluruhnya:

- Serial dan paralel "matrix.h"
- Serial dan paralel "vector.h"
- Serial "em.h"
- Serial dan paralel "var.h"
- Serial dan paralel "algebra.h"
- Serial "main.cpp"
- Paralel "em.h"
- Paralel main.cpp

Masing-masing modul "matrix.h", "vector.h", dan "em.h" dibuat dalam *header file* dimana implementasinya disatukan juga dalam *header file* tersebut. Ketiga modul ini dibuat dalam bentuk *class module*. Program dibuat dengan menggunakan CodeBlocks 12.1 dengan *compiler* adalah gcc versi 4.4.1, dan untuk program paralelnya menggunakan OpenMP versi 2.0.

Untuk mendapatkan manfaat dari paralelisasi, peneliti membagi pekerjaan besar kepada *thread-thread* dengan beban yang sama. Dalam algoritma EM, terdapat suatu *loop* yang akan berhenti jika ditemukan data estimasi hilang atau jumlah *loop* telah mencapai maksimal iterasi yang telah ditentukan dalam program (ditetapkan maksimal iterasi adalah 1.000). Dalam algoritma EM, ada dua tahap dalam *loop*, yaitu tahap prediksi dan tahap estimasi. Kedua tahap itu saling bergantung bahwa harus dilakukan tahap prediksi dahuhlu kemudian dilakukan tahap estimasi. Selain itu, suatu *loop* selanjutnya memiliki ketergantungan dengan hasil dari *loop* sebelumnya. Sebelum *loop* ada variabel-variabel matrik dan vektor yang harus diisi terlebih dulu. Setelah kode program serial selesai dibuat, diperoleh bahwa pekerjaan terbesar yang dilakukan program adalah inisialisasi sebelum *loop* dibandingkan pekerjaan yang dilakukan pada satu *loop*. Berikut gambaran hasil analisis kode program serial:

Inisialisasi Sebelum Loop (*Heavy Job*)	
Loop	Tahap Prediksi *(Medium Job)*
	Tahap Estimasi (*Medium Job*)

Gambar 2. Gambaran Hasil Analisis Kode Program Serial

Dari hasil analisis alur proses dan konstrain maka bentuk paralelisasi sebagian besar dilakukan dengan *non-iterative worksharing*. Dalam OpenMP jenis ini menggunakan *directive* "#pragma omp sections". Level paralelisasi dilakukan pada modul EM bukan pada modul Matrik dan Vektor .

2.2. Pengujian

Uji Keluaran Program dengan Teori di Buku (Uji Validitas Hasil)

Set data yang dipakai dalam pengujian program seperti contoh pada buku referensi terkait *EM Algorithm* [12].

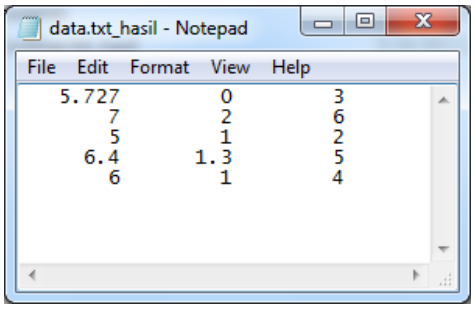
Data Percobaan

$$X = \begin{bmatrix} - & 0 & 3 \\ 7 & 2 & 6 \\ 5 & 1 & 2 \\ - & - & 5 \end{bmatrix}$$

Hasil output pada iterasi pertama dari contoh buku

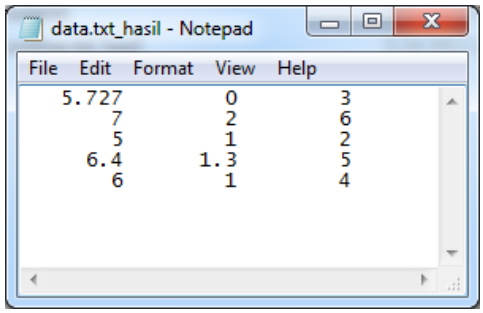
$$X = \begin{bmatrix} 5{,}73 & 0 & 3 \\ 7 & 2 & 6 \\ 5 & 1 & 2 \\ 6{,}4 & 1{,}3 & 5 \end{bmatrix}$$

Hasil output pada iterasi pertama dari program serial



Gambar 3. Hasil Keluaran pada Iterasi Pertama dari Program Serial

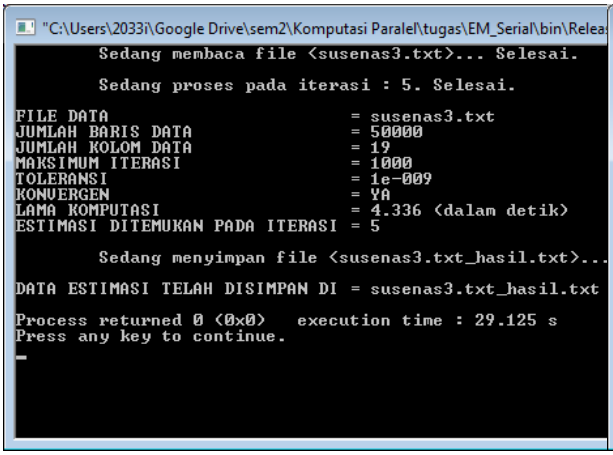
Hasil output pada iterasi pertama dari program paralel



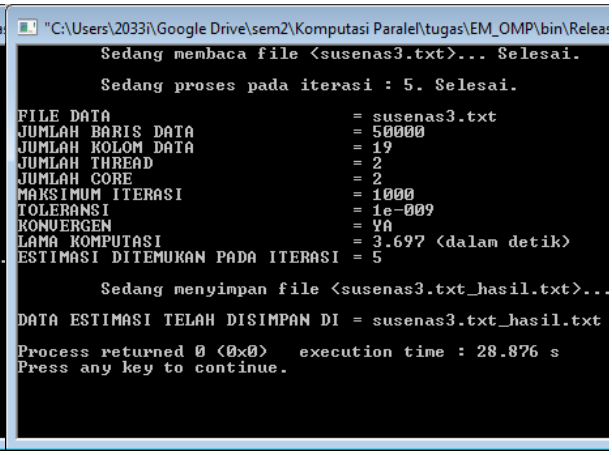
Gambar 4. Hasil Keluaran pada Iterasi Pertama dari Program Paralel

2.3. Uji Perbandingan Serial dan Paralel

Spesifikasi komputer yang digunakan untuk menguji program adalah MacBook Pro prosesor i7-2635QM 2GHz memori 4GB. Lama waktu komputasi adalah rata-rata waktu yang diperoleh dari lima kali percobaan. Data yang digunakan berjumlah 5000, 10.000, dan 50.000 baris yang berasal dari data bangkitan untuk masing masing pengujian program serial, program paralel dengan 2 *thread*, dan prorgam paralel dengan 4 *thread*. Jumlah variabel atau kolom untuk seluruh pengujian sama yaitu sebanyak 19 variabel, begitu juga dengan jumlah baris yang mengandung data hilang sama yaitu sebanyak 14 baris, sedangkan jumlah kolom yang mengandung data hilang juga sama yaitu sebanyak 18 kolom. Hasil keluaran program serial dan paralel dapat dilihat pada Gambar 5 dan Gambar 6.



Gambar 5. Contoh Keluaran Program Serial



Gambar 6. Contoh Keluaran Program Paralel

Tabel 1. Tabel Analisis dan Perbandingan Serial dan Paralel

Jumlah Sampel (baris)	Lama Komputasi Serial/ Ts (detik)	Komputasi Paralel		Persentase Selisih Serial dan Paralel (%)	S	E
		Jumlah *Thread* (p)	Lama Komputasi Paralel / Tp (detik)			
5000	0,078	2	0,066	15,38	1.18	0.59

5000	0,078	4	0,060	23,08	1.30	0.33
10000	0,150	2	0,122	18,67	1.23	0.61
10000	0,150	4	0,110	26,67	1.36	0.34
50000	0,611	2	0,515	15,71	1.19	0.59
50000	0,611	4	0,493	19,31	1.24	0.31

Berdasarkan Tabel 1 di atas dapat dilihat bahwa lama komputasi program paralel membutuhkan waktu yang lebih sedikit dibandingkan lama komputasi program serial. Rata-rata lama komputasi program serial dari seluruh sampel yang diuji membutuhkan waktu 0,2796 detik, sedangkan rata-rata lama komputasi program paralel adalah 0,2276 detik. Untuk persentase selisih lama waktu antara program serial dan program paralel yaitu 18,59 persen. Penggunaan jumlah thread pada program paralel juga membuat waktu komputasi semakin cepat jika dibandingkan dengan program serial. Rata-rata persentase selisih lama komputasi paralel yang menggunakan jumlah thread sebanyak 2 dengan komputasi serial sebesar 16,59 persen, sedangkan komputasi paralel yang menggunakan jumlah thread sebanyak 4 dengan komputasi serial sebesar 27,95 persen atau dengan kata lain komputasi paralel dengan jumlah *thread* 4 membuat komputasi 1,68 kali lebih cepat.

Jika dilihat berdasarkan nilai *speed up* untuk keseluruhan pengujian program paralel, seluruh hasilnya lebih dari 1 kali. Hal ini menunjukkan bahwa secara rata-rata komputasi paralel lebih cepat 1,25 kali dibandingkan komputasi serial. Begitu juga dengan efisiensi algoritma paralel secara rata-rata yaitu 0,46. Penambahan thread pada komputasi paralel juga terlihat mengurangi lama komputasi dan efesiensi serta meningkatkan *speed up*.

3. KESIMPULAN

Dari hasil penelitian paralelisasi kode program EM *Algorithm* diperoleh kesimpulan bahwa Alur proses kerja program serial dan pemilihan struktur data mempengaruhi jenis paralelisasi yang tepat untuk kode program paralel. Komputasi paralel pada penanganan data hilang dengan EM Algrithm dapat mempercepat waktu komputasi. Peningkatan jumlah *thread* pada kode program EM *Algorithm* dapat mengurangi lama komputasi, meningkatkan *speed up*, dan menurunkan efisiensi sumber daya yang digunakan.

Daftar Pustaka

[1] Andri Lesmana, W., Maria Angela, K., Sri, M. 2017. Komputasi Paralel Untuk Pengolahan Prestasi Akademik Mahasiswa. *Jurnal Teknologi Elektro* 8.

[2] Dempster, A.P., Laird, N.M., Rubin, D.B. 1977. Maximum Likelihood from Incomplete Data Via the Em Algorithm. *Journal of the Royal Statistical Society: Series B* 39, 1-22.

[3] Lumbanraja, F.R., Aristoteles, A., Muttaqina, N.R. 2020. Analisa Komputasi Paralel Mengurutkan Data Dengan Metode Radix Dan Selection. *Jurnal Komputasi* 8, 77-93.

[4] Lumbanraja, F.R., Aristoteles, A., Nadila Rizqi, M. 2020. Analisa Komputasi Paralel Mengurutan Data Dengan Metode Radix Dan Selection. *Jurnal Komputasi* 8, 77-93.

[5] Morrison, D.F., Marshall, L.C., Sahlin, H.L., 1976. Multivariate Statistical Methods. McGraw–Hill Book Company, New York.

[6] Mulya, M., Abdiansah, A. 2013. Penerapan Multi-Threading Untuk Meningkatkan Kinerja Pengolahan Citra Digital. *Jurnal Generic* 8, 230-237.

[7] Musil, C.M., Warner, C.B., Yobas, P.K., Jones, S.L. 2002. A Comparison of Imputation Techniques for Handling Missing Data. *Western Journal of Nursing Research* 24, 815-829.

[8] Nova, M., Mukid, M. 2011. Pendugaan Data Hilang Dengan Menggunakan Data Augmentation. *Jurnal Media Statistika* 4, 73-86.

[9] Nurmawati, E., 2006. *Rekayasa Program Penanganan Data Hilang (Dengan Metode Listwise, Pairwise, Dan Expectation-Maximization Algorithm)*. Sekolah Tinggi Ilmu Statistik, Jakarta.

[10] Rabbani, H., Gunawan, P.H. 2018. Kinerja Openmp Pada Pengolahan Citra Dengan Model Curvature Motion. *Indonesia Journal on Computing* 3, 97-102.

[11] Rastogi, S., Zaheer, H., Year. *Significance of Parallel Computation over Serial Computation.* pp. 2307-2310.

[12] Richard A. Johnson, D.W.W., 2002. Applied Multivariate Statistical Analysis Fifth Edition. Prentice Hall, Upper Saddle River, New Jersey, p.^pp. 235-255.

[13] Santoso, S., 2002. Spss Statistik Multivariat. Elex Media Komputindo, Jakarta.

[14] Yusman, M., Aristoteles, A., Irawati, A.R. 2012. Analisis Komputasi Paralel Dan Serial Pada Algoritma Merge Sort. *Jurnal Sains MIPA* 18.