ALGORITMA PARALEL *ODD EVEN TRANSPOSITION* PADA MODEL JARINGAN NON-LINIER

Ernastuti, Ravi A. Salim, dan Haryanto

Pusat Studi Komputasi Matematika, Universitas Gunadarma, Jalan Margonda Raya 100, Depok, 16424, Indonesia

E-mail : emas@staff.gunadarma.ac.id

Abstrak

Odd-even-transposition adalah suatu algoritma *paralel* yang merupakan pengembangan dari algoritma sekuensial "*bubble sort*". Algoritma *odd-even-transposition* ini didesain khusus untuk model jaringan *array* linier (homogen). Untuk n elemen data, kompleksitas waktu dari algoritma *bubble sort* adalah $O(n^2)$, sedangkan pada *odd-even-transposition* yang bekerja di atas *n* prosesor adalah $\Theta(n)$. Ada peningkatan kecepatan waktu pada kinerja algoritma paralel ini sebesar n kali dibanding algoritma sekuensialnya. *Hypercube* dimensi *k* adalah model jaringan non-linier (non-homogen) terdiri dari $n = 2^k$ prosesor, di mana setiap prosesor berderajat k. Model jaringan *Fibonacci cube* dan *extended Lucas cube* masing-masing merupakan model subjaringan *hypercube* dengan jumlah prosesor $< 2^k$ prosesor dan maksimum derajat prosesornya adalah k. Pada *paper* ini, diperlihatkan bagaimana algoritma *odd-even-transposition* dapat dijalankan juga pada model jaringan komputer *cluster* non-linier *hypercube*, *Fibonacci cube,* dan *extended Lucas cube* dengan kompleksitas waktu O(n).

Kata Kunci: *bubble sort, extended lucas cube, fibonacci cube, jaringan cluster non-linier hypercube, odd even transposition*

Abstract

Odd-even-transposition is a parallel algorithm which is the development of sequential algorithm "bubble sort". Odd-even transposition algorithm is specially designed for linear array network model (homogeneous). For n data elements, the time complexity of bubble sort algorithm is $O(n^2)$, while the odd-even-transposition that works with n processor is $\Theta(n)$. There in an increase in the speed of time on the performance of this parallel algorithms for n times than its sequential algorithm. K-dimensional hypercube is a non-linear network model (non-homogeneous) consists of $n = 2^k$ processors, where each processor has k degree . Network model of Fibonacci cube and extended Lucas cube are the hypercube sub-network model with the number of processors $<2^k$ processors and maximum processor degree is k. In this paper, it is shown how the odd-even-transposition algorithm can also be run on non-linear hypercube cluster, Fibonacci cube, and extended Lucas cube computer network model with time complexity O(n).

Keywords: *bubble sort, extended lucas cube, fibonacci cube, non-linier hypercube cluster network, odd even transposition*

1. Pendahuluan

Sorting merupakan salah satu operasi paling penting dalam sistem basis data dan efisiensinya dapat memengaruhi kinerja sistem secara drastis. Untuk meningkatkan kinerja sistem basis data, paradigma *paralellism* (komputasi secara paralel) digunakan untuk mengeksekusi operasi-operasi pada pengadministrasian data [1-3].

Sekumpulan komputer yang terhubung dalam suatu jaringan area lokal disebut jaringan komputer *cluster* yang memungkinkan untuk meningkatkan kecepatan waktu pemrosesan aplikasi. Untuk melihat kekuatan komputasi paralel pada jaringan komputer *cluster*, penelitian-penelitian yang terkait telah dipelajari dan dievaluasi pada berbagai aplikasi ilmiah.

Mesin paralel yang didiskusikan pada *paper* ini khusus diaplikasikan untuk sistem basis data paralel. Sebelum ini telah banyak penelitian yang dialamatkan untuk permasalahan terkait dengan sistem basis data. Namun hanya sedikit yang mendiskusikan tentang evaluasi kinerja algoritma *sorting* paralel pada jaringan komputer *cluster* non-linier.

Odd-even-transposition adalah suatu algoritma paralel *sorting* yang merupakan pengembangan dari algoritma sekuensial *bubble*

sort, di mana algoritma *odd-even-transposition* ini didesain khusus untuk model jaringan komputer *array* linier (homogen). Untuk *n* elemen data, kompleksitas waktu dari algoritma *bubblesort* pada prosesor tunggal adalah $O(n^2)$, sedangkan pada *odd-even-transposition* yang bekerja di atas *n* prosesor adalah $O(n)$. Terlihat peningkatan kecepatan waktu pada kinerja algoritma paralel ini sebesar *n* kali dibanding algoritma sekuensialnya [1-3].

Pokok persoalan utama arsitektur komputer paralel adalah pada desain topologi (model) jaringan interkoneksi prosesor. Topologi jaringan interkoneksi biasanya dinyatakan sebagai graf, di mana simpul menyatakan elemen prosesor dan busur menyatakan saluran komunikasi dalam jaringan [2].

Komputer-komputer yang mendukung komputasi paralel saat ini telah tersedia dengan berbagai macam topologi interkoneksi antara lain, topologi *hypercube* yang merupakan topologi populer dan paling banyak digunakan [4]. Jumlah prosesor dalam *hypercube* dimensi *k* adalah 2^k. *Hypercube* memiliki banyak sifat istimewa dan menarik, antara lain simpul dan busurnya simetri, memiliki struktur rekursif yang sederhana, memiliki jalur dan siklus Hamiltonian yang dapat ditanam (*embedded*) model jaringan lain seperti *linear array, ring, mesh* dan *tree* secara efisien. Namun kelemahan *hypercube* adalah ketika dimensinya diperbesar, jumlah prosesornya meningkat secara eksponensial. Hal ini sangat membatasi ukuran sistem jaringan yang akan dibangun [5-7]. Untuk mengatasi kelemahan ini, telah diperkenalkan model-model jaringan interkoneksi dengan jumlah simpul yang jauh lebih sedikit dari 2^k serta memiliki hampir semua sifat istimewa *hypercube* yaitu *Fibonacci cube* (FC), *Extended Fibonacci cube* (EFC) dan *Extended Lucas cube* (ELC), masing-masing diperkenalkan oleh Hsu pada tahun 1993 [5], Wu (1997) [7], dan Ernastuti (2007) [4][8-10]. Graf-graf ini masing-masing adalah subgraf bentukan dari *hypercube*.

Dengan kata lain, *hypercube* dimensi *k* adalah model jaringan non-linier terdiri dari $n = 2^k$ prosesor. Sedangkan jaringan kubus FC, EFC, dan ELC, masing-masing merupakan model subjaringan *hypercube* dengan jumlah prosesor $< 2^k$ dan maksimum derajat prosesornya adalah *k*. Dari penelitian yang telah dilakukan membuktikan bahwa pada subjaringan ini dapat juga ditanamkan model jaringan linier *array*, *ring*, *mesh* dan *tree*.

Pada *paper* ini diperlihatkan bagaimana algoritma *odd-even-transposition* dapat dijalankan pada model jaringan komputer *cluster* non-linier *hypercube*, FC dan ELC dengan kompleksitas waktu $O(n)$, di mana $n = 2^k$.

Sampai saat ini, teori graf masih digunakan untuk menyelesaikan masalah dalam berbagai bidang. Pada umumnya, teori graf digunakan untuk memodelkan persoalan dan mencari solusinya. Beberapa persoalan pada zaman modern yang menggunakan teori graf antara lain adalah permasalahan jaringan *web*, prosesor, telepon, dan utilitas.

Dari banyak masalah yang menggunakan graf sebagai alat bantu pemodelan dan penyelesaiannya, penelitian ini khusus membahas mengenai aplikasi teori graf dalam topologi jaringan, khususnya model jaringan interkoneksi prosesor atau komputer non-linier *hypercube, Fibonacci cube* dan *extended Lucas cube*. Pada penelitian ini, ketiga model jaringan tersebut dibuktikan dapat dimanfaatkan untuk menjalankan algoritma paralel *sorting Odd-even-transposition.*

Definisi 1, misal A adalah suatu daftar dari *n* elemen A_1, A_2 , …, A_n dalam memori. Mengurutkan *(sorting)* A berkaitan dengan operasi menyusun elemen-elemen di dalam A demikian sehingga elemen-elemen tersebut terurut secara numerik atau secara leksikografik, yaitu sedemikian sehingga $A_1 \leq A_2 \leq A_3 \leq \ldots \leq A_n$.

Dengan kata lain definisi 1 dapat dinyatakan dengan tiga buah asumsi pertanyaan. Pertama, tabel dengan *n* elemen: A_1, A_2 , …, A_n. Kedua, untuk dua elemen A_i dan A_j, salah satu kondisi ini pasti benar: $A_1 \leq A_2$, $A_1 = A_2$ atau $A_1 \geq A_2$. Ketiga, tujuan *sorting*: menemukan permutasi (π_0, π_1 , …, π_{n-1}) sedemikian sehingga $A\pi_0 \leq A\pi_1 \leq \ldots \leq A\pi_{n-1}$.

Definisi 2 yaitu sebuah graf G=(V,E) terdiri dari himpunan simpul V dan himpunan busur E; sebuah busur adalah suatu pasangan tidak terurut (x,y), di mana $x \neq y \in V_G$. Untuk menghindari pengertian yang ambigu, V dan E dalam graf G masing-masing dinotasikan V_G dan E_G.

Definisi 3 yaitu jalur Hamiltonian pada graf tak berarah $G(V_G, E_G)$ didefinisikan sebagai suatu perjalanan yang melewati setiap simpul $v_i \in V_G$ tepat satu dan satu kali. Jika jalur Hamiltonian dalam G dimulai dari simpul v_1 dan berakhir disimpul v_n, dengan (v_1,v_n) adalah busur $\in E_G$, maka jalur ini disebut *Hamiltonian cycle*. Graf yang memiliki *Hamiltonian cycle* disebut graf Hamiltonian.

Masalah pencarian jalur Hamiltonian dan *Hamiltonian cycle* pada suatu graf membutuhkan waktu *nonpolynomial*, bahkan masalah ini terbukti termasuk dalam kelas *NP-complete*. Namun, untuk beberapa graf tertentu masalah ini bisa diselesaikan dalam waktu *polynomial* [8][10].

Contohnya, perhatikan graf-graf berikut: *Hypercube* dimensi *n Q(n)* adalah suatu graf tak berarah di mana setiap simpulnya diwakili oleh suatu *string biner* panjang-*n* bit $\in \{0,1\}^n$. Berdasarkan sifat bipartisi, graf *hypercube* dibuktikan sebagai graf Hamiltonian. Beberapa model subgraf bentukan *hypercube* juga telah dibuktikan oleh para peneliti sebagai graf Hamiltonian. Beberapa subgraf tersebut adalah graf *Fibonaccicube* (FC), graf *Extended Fibonacci cube* (EFC), dan graf *extended Lucas cube* (ELC). Hsu [5] menunjukkan kurang dari sepertiga graf FC adalah Hamiltonian, walau untuk semua graf FC memiliki jalur Hamiltonian. Sedangkan Wu [7] telah membuktikan semua EFC adalah Hamiltonian. Graf EFC ini merupakan pengembangan dari graf FC. Untuk sifat Hamiltoniannya terlihat EFC lebih baik dari FC. *Jean L Baril dan Vincent Vajnovzki* [11] di dalam *paper*-nya pada tahun 2005 memperlihatkan pemanfaatan konsep *minimal change list* dan *kode gray* untuk menentukan jalur Hamiltonian pada *graf* FC dan *Lucas cube* (LC). Selanjutnya, pada *paper* [8][10] dengan pendekatan 1-kode *gray* dan sifat bipartisi telah membuktikan bahwa semua ELC adalah Hamiltonian.

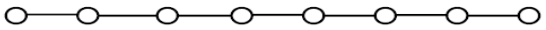
Definisi 4 yaitu bit adalah suatu digit biner 0, 1, atau λ. S*tring* biner panjang-*n* adalah suatu barisan hingga bit dengan panjang-*n*., atau dengan kata lain *string* biner panjang-*n* adalah suatu barisan $a_1\ a_2 \ldots a_n$ dengan $a_i \in \{0,1\}$.

String λ adalah *string* yang berisi bit kosong dengan panjang 1. Jika α adalah *string* biner dan V adalah himpunan *string* biner, maka operasi αV menyatakan *concatenation* antara string α dan setiap *bit-string* dalam *V*. Contoh: 01.{0,1}= {010, 011}.

Definisi 5 yaitu sebuah *string Fibonacci* panjang-*n* adalah string biner $a_1\ a_2 \ldots a_n$ yang tidak mengandung dua bit '1' berurutan. Berikut adalah semua *string Fibonacci* panjang-3: 000, 001, 010,100,101.

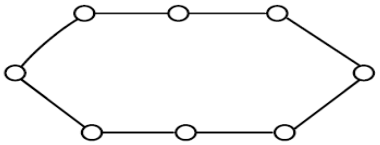
Definisi 6 yaitu jarak *Hamming* di antara dua string biner x dan y panjang-*n* adalah jumlah dari posisi bit berbeda diantara keduanya; notasi: *H(x,y)*. Contoh: H(001,100)=2.

Definisi 7 yaitu model jaringan *array* linier LA(n), n ≥ 2, adalah graf yang memiliki *n* simpul terdiri dari dua simpul berderajat satu dan *n-2* simpul berderajat dua. Gambar 1 memperlihatkan model jaringan LA (8).



Gambar 1. Model jaringan linier *array* LA(8).

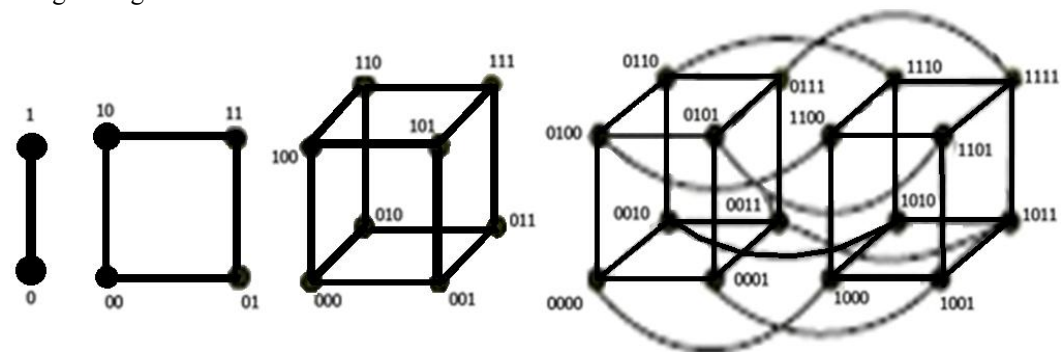
Definisi 8 yaitu model jaringan ring R(n), n ≥ 2, adalah graf yang memiliki *n* simpul terdiri dari *n* simpul berderajat dua. Gambar 2 memperlihatkan model jaringan R(8).



Gambar 2. Model jaringan ring R(8).

Model-model jaringan non-linier. *Hypercube* dimensi *k*, dinotasikan *Q(k)* adalah suatu graf tak berarah yang setiap simpulnyadiwakili oleh suatu *string* biner panjang-*k* bit. Setiap dua simpul dalam *Q(k)* akan terhubung jika *string-string* yang mewakilinya berbeda satu posisi bit.

Jumlah simpul dalam *Q(k)* adalah 2^k, di mana setiap simpul memiliki keterhubungan dengan *k* simpul lainnya. Derajat setiap simpul adalah *k*. *Hypercube* dapat didefinisikan secara rekursif sebagai berikut:



(a) (b) (c) (d)

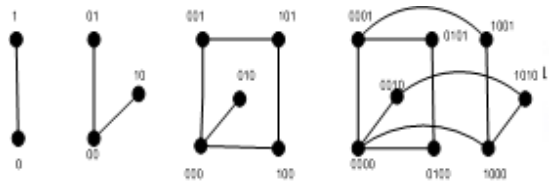
Gambar 3. *Graph hypercube* Q(k): (a) Q(1), (b) Q(2), (c) Q(3), (d) Q(4).

Definisi 9 yaitu *(Hypercube*) Untuk $k \geq 0$, $Q(k) = (V_Q(k), E_Q(k))$ dapat didefinisikan : Himpunan simpul $V_Q(k)$ dinyatakan secara rekursif sebagai berikut: $V_Q(k) = 0.V_Q(k\text{-}1) \cup 1 \cdot V_Q(k\text{-}1)$, di mana $V_Q(0) = \{\}$; $V_Q(1) = \{0,1\}$; dua simpul x,y $\in V_Q(k)$ dihubungkan oleh suatu busur $\in E_Q(k)$ jika dan hanya jika jarak *Hamming* H(x,y)=1.

Contoh: untuk *hypercube* dimensi3, *Q(3)*, maka himpunan simpulnya adalah { 000, 001, 010, 100, 101, 110, 011, 111}. Gambar 3 memperlihatkan *graph hypercube* Q_k, untuk dimensi k=1,2,3,4.

Hsu mengembangkan FC yang jumlah simpulnya jauh lebih sedikit (*sparse*) daripada *hypercube.* Graf *Fibonacci cube* dimensi*k FC(k)* ini adalah graf tak berarah.

Definisi 10 yaitu [5] (*Fibonacci cube*). Untuk $k \geq 0$, $FC(k) = (V_{FC}(k), E_{FC}(k))$ dapat didefinisikan: $V_{FC}(k)$ dinyatakan secara rekursif sebagai berikut $V_{FC}(k) = 0 \cdot V_{FC}(k\text{-}1) \cup 10 \cdot V_{FC}(k\text{-}2)$, di mana $V_{FC}(0) = \{\}$, $V_{FC}(1) = V_{FC}(2) = \{\lambda\}$ dan $V_{FC}(3) = \{0,1\}$; dua simpul x,y $\in V_{EFC}(k)$ dihubungkan oleh suatu busur $\in E_{FC}(k)$jika dan hanya jika jarak *Hamming* H(x,y)=1. FC(k) adalah subgraf bentukan dari *hypercube* Q(k-2), untuk k $\geq$ 3. Gambar 4 memperlihatkan graf FC(k), untuk k=3,4,5,6. Selanjutnya model graf FC diperluas oleh Wu ke dalam graf yang disebut *extended Fibonacci cube* (EFC).



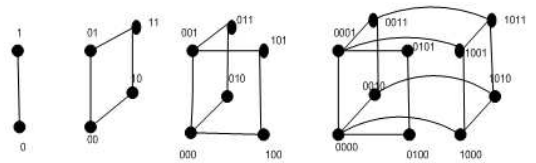
(a) (b) (c) (d)

Gambar 4. Graf FC (a) FC(3), (b) FC(4), (c) FC(5), dan (d) FC(6).

Definisi 11 yaitu [7] (*Extended Fibonacci cube*)**.** Untuk $k \geq 0$, $EFC(k) = (V_{EFC}(k), E_{EFC}(k))$ dapat didefinisikan: $V_{EFC}(k)$ dinyatakan secara rekursif sebagai berikut $V_{EFC}(k) = 0 \cdot V_{EFC}(k\text{-}1) \cup 10 \cdot V_{EFC}(k\text{-}2)$, di mana $V_{EFC}(0) = \{\}$, $V_{EFC}(1) = V_{EFC}(2) = \{\lambda\}$, $V_{EFC}(3) = \{0,1\}$ dan $V_{EFC}(4) = \{00,10,11,01\}$; dua simpul x,y $\in V_{EFC}(k)$ dihubungkan oleh suatu busur $\in E_{EFC}(k)$jika dan hanya jika jarak *Hamming* H(x,y)=1.

EFC(k) adalah subgraf bentukan dari *hypercube* Q(k-2), untuk $k \geq 3$. Gambar 5 memperlihatkan graf EFC(k), untuk k=3,4,5,6.

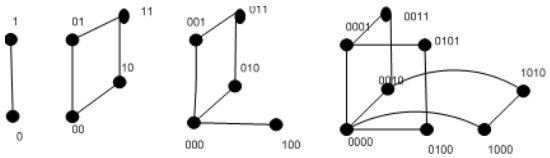
Definisi 12 yaitu [9][10] *(Extended Lucas cube).* Graf ELC dinyatakan sebagai berikut: Untuk $k \geq 0$, $ELC(k) = (V_{ELC}(k), E_{ELC}(k))$, maka himpunan simpul ELC(k) didefinisikan secara rekursif : $V_{ELC}(k) = 0 \cdot V_{EFC}(k\text{-}1) \cup 10 \cdot V_{EFC}(k\text{-}3) \cdot 0$, dengan $V_{EFC}(0) = \{\}$, $V_{EFC}(1) = \{0, 1\}$ dan $V_{EFC}(2) = \{00, 10, 11, 01\}$, di mana $V_{EFC}(k)$ adalah himpunan simpul graf EFC(k).Gambar 6 memperlihatkan model graf ELC(k) untuk k = 3,4,5,6.



(a) (b) (c) (d)

Gambar 5. Graf EFC (a) EFC(3), (b) EFC(4), (c)EFC(5) , (d) EFC(6).

Algoritma sorting terdiri dari algoritma *bubble sort* dan algoritma *odd even transposition*. Algoritma *bubble sort* diperuntukkan untuk komputer prosesor tunggal. Algoritma *bubble sort* adalah jika *n* adalah jumlah elemen data dengan elemen-elemennya adalah T_1 , T_2 , ..., T_{n-1} , T_n maka, pertama dilakukan traversal dari belakang untuk membandingkan dua elemen yang berdekatan. Kedua, jika elemen pada $T_{n-1} > T_n$, maka dilakukan proses pertukaran data (*swap*). Jika tidak, lanjutkan ke data berikutnya sampai bertemu dengan data yang telah diurutkan. Ketiga ulangi langkah tersebut untuk semua data yang tersisa.



(a) (b) (c) (d)

Gambar 6. Graf ELC(k). (a) ELC(3), (b) ELC(4), (c) ELC(5), dan (d) ELC(6).

Algoritma *bubble sort* untuk struktur data linier *array* dapat dilihat pada gambar 7:

```
For i ← 1 to (n-1) do
    For j ← n downto (i+1) do
 If  data[j] < data [j-1] then
   Temp ← data[J]
        data[j]← data[j-1]
    data[j-1]← Temp
   Endif
     Endfor
Endfor
```

Gambar 7. Algoritma *bubble sort.*

Gambar 9 mengilustrasikan contoh penyelesaian masalah *sorting* dengan algoritma sekuensial *bubble sort*. Pada kondisi awal bilangan-bilangan 4, 2, 7, 8, 5, 1, 3, 6 masing-masing ditempatkan pada *array linier* 8 elemen. Jumlah iterasi yang dibutuhkan untuk menyelesaikan masalah *sorting* dengan teknik *bubble sort* pada gambar 1 adalah $8 \times (8\text{-}1) / 2$. Secara umum kompleksitas algoritma sekuensial *bubble sort* pada prosesor tunggal dari *n* bilangan adalah $O(n^2)$.

Algoritma *odd even transposition* digunakan untuk jaringan komputer *cluster* linier homogen. Algoritma ini dirancang untuk model *array* prosesor dengan elemen-elemen *processing* yang disusun menjadi *mesh* satu dimensi. Anggap $A = (a_0, a_1, ..., a_{n-1})$ adalah himpunan *n* elemen yang akan di-*sort*. Setiap *n* elemen *processing* berisi dua variabel lokal yaitu *a* yang merupakan elemen unik dari *arrayA*, dan *t* merupakan variabel berisi nilai yang diambil dari elemen *processing* tetangganya. Algoritma melakukan *n*/2 iterasi, dan setiap iterasi memiliki dua fase, yaitu:

Fase 1, *Odd-even exchange* merupakannilai *a* pada setiap prosesor bernomor ganjil (kecuali prosesor *n*-1) dibandingkan dengan nilai *a* yang tersimpan di prosesor berikutnya. Nilai-nilai ditukar, jika perlu, sehingga prosesor dengan nomor lebih kecil berisi nilai yang lebih kecil.

```
Parameter n
Global i
Local a {elemen yang di-sort}
t  {elemen  yang  diperoleh  dari  prosesor
tetangganya }
   begin
   for i ← 1 to n/2 do
   for all Pj, di mana 0 ≤ j ≤ n-1 do
if   j  <  n–1  and  odd(j)  then  {Odd-even
exchange}
 t⇐successor(a) {Ambil nilai dari prosesor
tetangga}
   successor(a)⇐max(a,t)
   a ← min(a,t)
    endif
   if              even(j)              then
{Even-odd exchange}
  t⇐successor(a)
{Ambil   nilai   dari   prosesor   tetangga}
successor(a)⇐max(a,t)
   a ← min(a,t)
         endif
   endfor
   endfor
end
```

Gambar 8. Algoritma *odd even transposition.*

Fase 2, *Even-odd exchange* merupakan nilai *a* pada setiap prosesor bernomor genap dibandingkan dengan nilai *a* yang tersimpan di prosesor berikutnya. Nilai-nilai ditukar jika perlu, sehingga prosesor dengan nomor lebih kecil berisi nilai yang lebih kecil. Setelah *n*/2 iterasi, nilai-nilai harus terurut.

Algoritma *odd even transposition* untuk *array* prosesor linier dapat dilihat pada gambar 8. Gambar 10 mengilustrasikan contoh penyelesaian masalah *sorting* dengan algoritma paralel *odd-even transposition*. Pada kondisi awal bilangan-bilangan 4, 2, 7, 8, 5, 1, 3, 6 masing-masing ditempatkan pada prosesor P_0 , P_1 , P_2 , P_3 , P_4 , P_5 , P_6 , P_7 . Jumlah iterasi yang dibutuhkan untuk menyelesaikan masalah *sorting* dengan teknik *odd even transposition* pada gambar 2 adalah 8 iterasi. Oleh karena itu, secara umum kompleksitas waktu pada algoritma paralel *odd even transposition* di jaringan komputer *cluster* homogen *n* prosesor dari *n* bilangan adalah $\Theta(n)$.

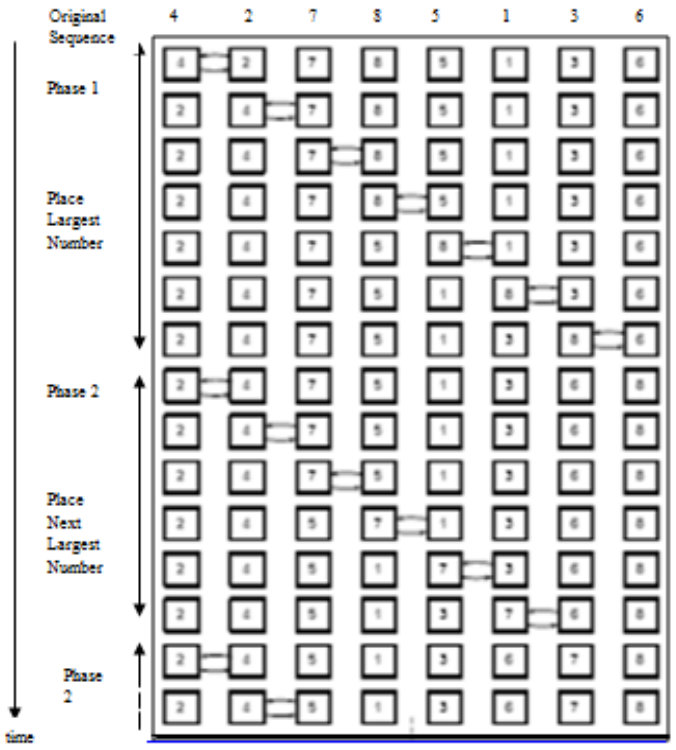
2. Metodologi

Agar dapat menjalankan algoritma paralel *odd even transposition* pada model-model jaringan non-linier *hypercube*, FC, dan ELC, pertama harus dibuktikan bahwa ketiga model jaringan tersebut dapat ditanam *(embedding*) model jaringan *array* linier. Kedua, untuk mengetahui kompleksitas waktu algoritma, jumlah simpul (prosesor) masing-masing model jaringan tersebut harus dihitung. Teorema dan *lemma* berikut dimanfaatkan sebagai langkah-langkah untuk menyelesaikan masalah ini.

TABEL I
BILANGAN *FIBONACCI* F_N, $1 \le N \le 8$

n	Bilangan *fibonacci* f_n
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34

Teknik penanaman (*embedding*) didefinisikan sebagai suatu fungsi pemetaan *g* yang memetakan simpul-simpul dari suatu graf *guest G* ke simpul-simpul dalam suatu graf *host H*. Teorema 1 yaitu [9] Jika suatu *graf* G dengan *n* simpul adalah graf Hamiltonian, maka model jaringan ring *R(n)* dapat ditanamkan pada G. Pada Teorema 1, jaringan ring *R(n)* dipandang sebagai *guest*, sedangkan graf G sebagai *host*. Karena model jaringan linier *array LA(n)* adalah subgraf bentukan dari *R(n)*, maka : Akibat 1 yaitu Model jaringan *array* linier *LA(n)* dapat pula ditanamkan pada graf G.



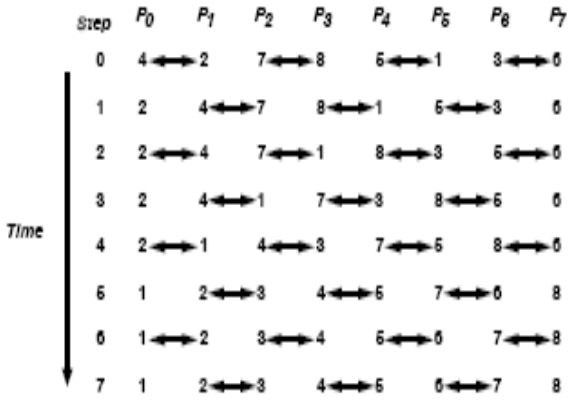
Gambar 9. Aplikasi algoritma *bubble sort* pada prosesor tunggal dengan struktur data *array*.

Teorema 2 yaitu[6] *Hypercube Q(k)* adalah graf Hamiltonian untuk semua *k*. *Lemma* 1 yaitu [6] Jumlah simpul *hypercube Q(k)* adalah 2^k. Teorema 2 dan *Lemma* 1 menghasilkan akibat. Akibat 2 yaitu model jaringan *array* linier LA(2^k) dapat ditanamkan pada model jaringan non-linier*hypercubeQ(k)*.

Gambar 11 memperlihatkan model jaringan *ringR(8)* dapat ditanam pada *hypercube Q(3)*, dengan kata lain model jaringan *array* linier *LA(8)* dapat pula ditanam pada model jaringan non-linier *hypercube Q(3)*. Teorema 3 yaitu[5] untuk k > 2, *Fibonacci cube* FC(*k*) adalah graf Hamiltonian hanya pada beberapa *k*, namun FC(*k*) memiliki jalur Hamiltonian di setiap *k*.

Definisi 13 yaitu bilangan *Fibonacci* f_n didefinisikan sebagai relasi rekursif berikut $f_{n+1} = f_n + f_{n-1}$, di mana nilai awal $f_1 = f_2 = 1$. Tabel I memperlihatkan tabel bilangan-bilangan *Fibonaccif*$_n$, di mana $1 \le n \le 8$.

Lemma 2 yaitu [6][12] untuk k > 2, jumlah simpul model jaringan *Fibonacci cube* FC(*k*) adalah f_{k-2}. Teorema 3 dan *Lemma* 2 menghasilkan akibat 3 yaitu model jaringan *array* linier LA(f_{k-2}.) dapat ditanamkan pada model jaringan non-linier *Fibonaccicube* FC(k), k > 2.



Gambar 10. Aplikasi algoritma paralel *odd even transposition* pada 8 prosesor *array* linier

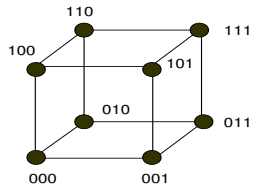
Teorema 4 yaitu [8][10] untuk k > 2, *extended Lucas cube* ELC(*k*) adalah *graf Hamiltonian* untuk semua k, kecuali pada k = 5. *Lemma* 3 yaitu [8-10] untuk k > 6, jumlah simpul model jaringan *extended Lucas cube* ELC(*k*) adalah $2 f_{k-2} + 2 f_{k-4}$. Sedangkan untuk k = 3,4,5,6 , jumlah simpul ELC(*k*) masing-masing adalah 2, 4, 5, 8.

Kemudian Teorema 4 dan *Lemma* 3 menghasilkan akibat 5. Akibat 5 yaitu [9] model jaringan *array* linier LA($2f_{k-2}.+ 2f_{k-4}$) dapat ditanamkan pada model jaringan non-linier *extended Lucas cube* ELC(k), k > 6. Sedangkan LA(2), LA(4), LA(5) dan LA(8) masing-masing dapat ditanamkan pada ELC(3), ELC(4), ELC(5) dan ELC(6) yang dapat dilihat pada gambar 12.

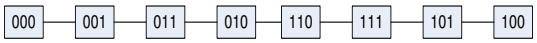
3. Hasil dan Pembahasan

Dari pembahasan sebelumnya, model-model jaringan non-linier *hypercube Q*, *Fibonacci cube FC* dan *extended Lucas cube ELC* masing-masing memiliki jalur Hamiltonan. Hal ini mengakibatkan model jaringan *array* linier LA dapat ditanam pada ketiga jaringan tersebut.

Untuk mengaplikasikan algoritma *odd even transposition* pada *hypercube* Q, pengujian algoritma dicobakan pada *hypercube* Q(3). *Hypercube* Q(3) pada gambar 13 memiliki jalur Hamiltonian berbentuk model jaringan *linier array* LA(8) pada gambar 14. Ilustrasi aplikasi algoritma *odd-even- transposition* pada Q(3) dapat dilihat pada gambar 15.



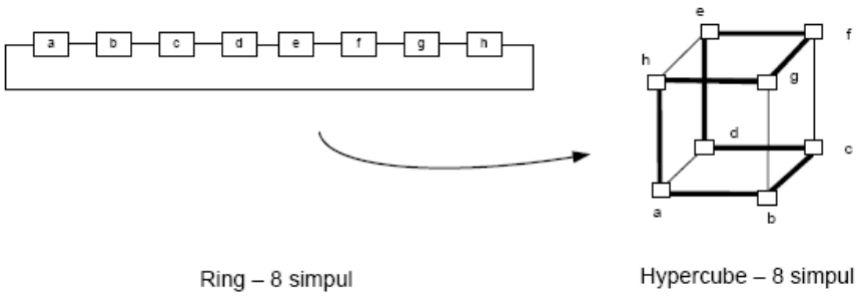
Gambar 13. *Hypercube* Q(3).



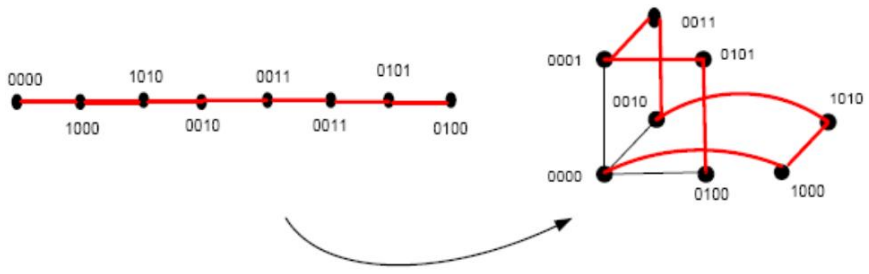
Gambar 14. Linier *array* LA(8).

Teorema 5, anggap ada *n* elemen yang akan di-*sort* pada *array* prosesor yang disusun menjadi linier *array LA* serta asumsikan bahwa sebelum dan sesudah *sort*, elemen akan didistribusikan merata, satu elemen per prosesor. Dengan demikian, batas bawah kompleksitas waktu pada algoritma *sorting* yang mana pun adalah $\Theta(n)$.

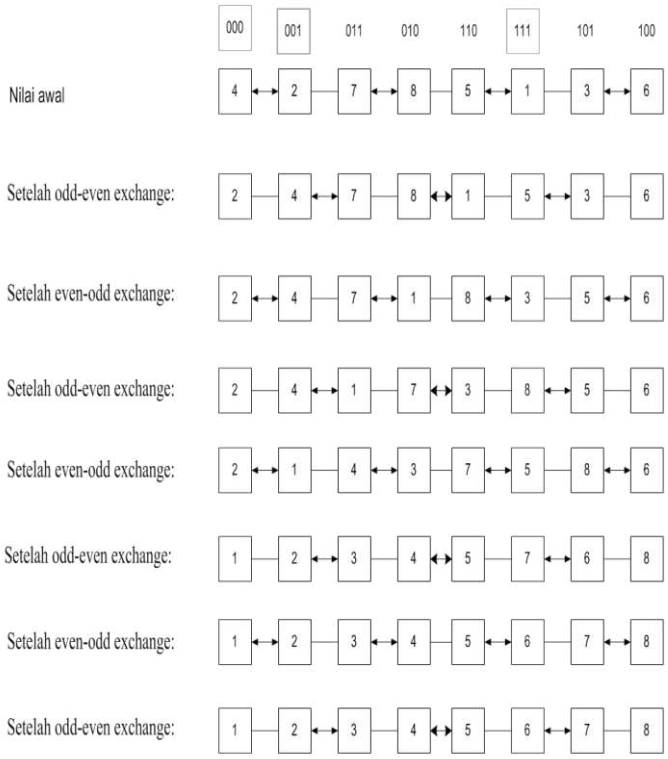
Buktinya yaitu lebar biseksi dari jaringan *linier array LA* adalah 1. Misalkan posisi yang ter-*sort* dari semua elemen yang pada awalnya ada pada satu sisi biseksi adalah pada sisi biseksi yang lain dan sebaliknya, maka seluruh *n* elemen harus melewati satu *link* untuk mencapai sisi yang lain. Karena *link* hanya dapat membawa satu elemen sekali, jumlah langkah yang diperlukan untuk menukar elemen melalui biseksi paling tidak adalah sebesar *n*. Dengan demikian, batas bawah kompleksitas jaringan linier *array LA* dengan syarat-syarat tersebut adalah $\Theta(n)$.



Gambar 11. Model jaringan *ringR*(2^3) ditanam pada *hypercubeQ(3)*.



Gambar 12. Model jaringan linier *array* LA(8) ditanam pada ELC(6).



Gambar 15. Ilustrasi aplikasi algoritma *odd even transposition* pada Q(3).

Teorema 6**,** kompleksitas *sorting n* elemen pada linier *array* prosesor *LA(n)* dengan *n* prosesor menggunakan *odd-even–transposition sort* adalah *Θ(n).* Buktinya yaitu berdasarkan fakta bahwa setelah *i* iterasi *loop for* luar pada algoritma *odd-even transposition* pada seksi 4.2, tidak ada elemen yang bisa lebih jauh dari *n*-2*i* posisi dari posisi akhir dan ter-*sort*-nya. Dengan demikian, *n*/2 iterasi cukup untuk men-*sort* elemen-elemen tersebut, dan kompleksitas waktu algoritma paralel adalah Θ(*n*) jika diberikan *n* elemen *processing*.

Akibat 6 yaitu kompleksitas *sorting n* elemen pada jaringan non-linier *hypercube* Q(k) dengan $n = 2^k$ prosesor menggunakan *odd-even-transposition sort* adalah Θ(*n*). Buktinya yaitu jelas terbukti benar dari akibat 2 dari Teorema 6.

Lemma 4 yaitu untuk k > 2 dan $n = 2^k$, kompleksitas *sorting*f_{k-2} elemen pada model jaringan non-linier *Fibonacci cube FC(k)* menggunakan *odd-even transposition sort* adalah *O(n)*. Buktinya yaitu dari definisi 10 terlihat bahwa $FC(k) \subseteq Q(k-2)$ atau dengan kata lain *FC(k)* adalah subgraf bentukan *hypercube Q(k-2). Lemma* 3 menyatakan bahwa untuk k > 2, jumlah simpul model jaringan *Fibonaccicube* FC(k) adalah f_{k-2}. Sedangkan $f_{k-2} < 2^k$, atau dengan kata lain $f_{k-2} < n$.

Teorema 3 menyatakan bahwa untuk semua k > 2, *Fibonacci cube* FC(*k*) adalah graf Hamiltonian pada beberapa *k*, namun FC(*k*) memiliki jalur Hamiltonian di setiap *k*. Dengan kata lain model jaringan *array* linier *LA*(f_{k-2}) dapat ditanamkan pada model jaringan *non-linierFibonacci cube FC(k)*, *k* > *2*. Karena $f_{k-2} < n$ dan teorema 6 menyatakan kompleksitas *sorting n* elemen pada *linier array* prosesor *LA(n)* dengan *n* prosesor menggunakan *odd—even-transposition sort* adalah Θ(*n*), maka kompleksitas *sorting* f_{k-2} elemen pada model jaringan non-linier *Fibonacci cube FC(k)* menggunakan *odd-even transposition sort* adalah *O(n).*

Lemma 5, untuk k > 4, kompleksitas *sorting* 2 f_{k-2} + 2 f_{k-4} elemen pada model jaringan non-linier *extended Lucas cube ELC(k)* menggunakan *odd-even-transposition sort* adalah O(*n*), di mana $n = 2^k$. Buktinya yaitu menggunakan cara yang sama dengan pembuktian *lemma* 4, terbukti bahwa untuk semua k > 4, kompleksitas *sorting* 2 f_{k-2} + 2 f_{k-4} elemen pada model jaringan non-linier *extended Lucas cube ELC(k)* menggunakan *odd-even transposition sort* adalah O(*n*).

4. Kesimpulan

Penelitian ini memberikan kesimpulan bahwa, pertama, kompleksitas *sorting n* elemen pada jaringan non-linier *hypercube* Q(k) dengan $n = 2^k$ prosesor menggunakan *odd-even transposition sort* adalah Θ(*n*). Kedua, untuk *k* > 2

dan $n = 2^k$, kompleksitas *sorting* f_{k-2} elemen pada model jaringan non-linier *Fibonacci cube FC(k)* menggunakan *odd-even transposition sort* adalah *O(n)*. Ketiga, untuk $k > 4$, kompleksitas *sorting* $2 f_{k-2} + 2 f_{k-4}$ elemen pada model jaringan non-linier *extended Lucas cube ELC(k)* menggunakan *odd-even transposition sort* adalah O(n), di mana $n = 2^k$. Keempat, secara umum kompleksitas algoritma *odd-even-transposition* pada model-model jaringan non-linier subgraf bentukan dari *hypercube* yang memiliki jalur Hamiltonian adalah *O(n),* di mana $n = 2^k$.

Referensi

[1] J. Modi & R. Prager, "Implementation of Bubble Sort and the Odd-even Transposition Sort on a Rack of Transputers," *R Parallel Comput,* vol. 4, pp. 345–348, 1987.

[2] S.H. Roosta, *Parallel Processing and Parallel Algorithms: Theory and Computation,* Springer Inc, New York, 2000.

[3] S.N.Salloum & D.Wang, "Fault Tolerance Analysis of Odd-even Transposition Sorting Networks with Single Pass and Multiple Passes" *In Proceeding of IEEE,* vol. 1, pp. 193-196, 1997.

[4] Ernastuti & V. Vajnovzki, "Pemanfaatan Topologi Jaringan Interkoneksi Graf Bentukan dari Objek Kombinatorial pada komputasi paralel" *In Proceeding of Konferensi Nasional Matematika XIII*, pp. 123-132, 2006.

[5] W.J. Hsu, "Fibonacci Cubes—a New Interconnection Topology," *IEEE Trans. Parallel Distr. Systems*, vol. 4, pp. 3-12, 1993.

[6] H.P. Katseff, *"Incomplete Hypercube, " IEEE Transaction On Computer,* vol. 37, pp. 604-607, 1998.

[7] J. Wu, "Extended Fibonacci Cubes, " *IEEE Transaction Parallel Distributed Systems,* vol. *8,* pp. 3-9, 1997.

[8] Ernastuti, B.H. Widjaja, & D. Kerami, "Extended Lucas Cube: A New Hamiltonian Graph," *Journal of the Indonesian Mathematical Society,* vol. 14, pp. 25-35, 2008.

[9] Ernastuti & V.Vajnovzki, "Embedding of Linear Arrays, Rings, and 2D Meshes on Extended Lucas Cube Networks" *In Proceedings of the International Conference on Electrical Engineering and Informatics*, pp. 78-81, 2007.

[10] Ernastuti, "Hamiltonicity on Interconnection Network: Extended Lucas Cube" *In Proceedings of the International Conference on Soft Computing, Intelligent System and Information Technology*, pp. 368-371, 2007.

[11] L.J. Baril & V. Vajnovzki, "Minimal Change List for Lucas Strings and Some Graph Theoretic Consequences," *Elsevier Theoritical Computer Science*, vol. 346, pp. 189-199, 2005.

[12] S. Klavzar, "On Median Nature and Enumerative Properties of Fibonacci-like-cubes," *Elsevier Discrete Mathematics*, vol. 299, pp. 145-153, 2005.