Process Discovery untuk *Streaming Event log* Menggunakan Model *Markov* Tersembunyi

Kelly R. Sungkono, Riyanarto Sarno, dan Abdul Munif
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: riyanarto@if.its.ac.id

***Abstrak*—*Process discovery* adalah teknik penggalian model proses dari rangkaian aktivitas yang tercatat dalam *event log*. Saat ini, sistem informasi menghasilkan streaming *event log* dimana *Online Heuristic Miner* adalah algoritma *process discovery* yang mampu menghasilkan model proses dari *streaming event log*. Algoritma *Online Heuristic Miner* memiliki kelemahan yaitu ketidakmampuan mengatasi *incomplete trace*. *Incomplete trace* adalah rangkaian aktivitas pada *event log* yang terpotong di bagian awal ataupun di bagian akhir. *Incomplete trace* mengakibatkan proses tidak dapat ditampilkan secara utuh dalam model proses. Algoritma yang memanfaatkan Model *Markov* Tersembunyi digunakan untuk membentuk model proses yang dapat menangani *incomplete trace*. Algoritma yang memanfaatkan Model *Markov* Tersembunyi terdiri atas gabungan dari metode pembentukan model proses serta metode yang dimodifikasi. Metode yang dimodifikasi adalah metode *Baum-Welch*, *Backward* serta *Viterbi*. Metode *Backward* dan *Viterbi* yang dimodifikasi digunakan untuk memperbaiki *incomplete trace* sedangkan metode *Baum-welch* yang dimodifikasi dan metode pembentukan model proses digunakan untuk membangun model proses dari Model *Markov* Tersembunyi. Hasil uji coba menunjukkan bahwa dengan adanya perbaikan *incomplete trace*, nilai kualitas dari sisi *fitness*, presisi, generalisasi, dan *simplicity* model proses dari algoritma yang memanfaatkan Model *Markov* Tersembunyi lebih tinggi dibandingkan model proses dari algoritma *Online Heuristic Miner*.**

***Kata Kunci*—*Incomplete Trace*, Model *Markov* Tersembunyi, *Process Discovery*, *Streaming Event log*.**

I. PENDAHULUAN

TEKNIK penggalian model proses bisnis dari beberapa rangkaian aktivitas yang tercatat pada *event log* adalah *process discovery* [1]. Model proses bisnis memiliki dua manfaat utama, yaitu sebagai alat bantu untuk menjelaskan proses yang terjadi pada sistem serta sebagai modal utama dalam menganalisa permasalahan terkait pengaturan rangkaian aktivitas.

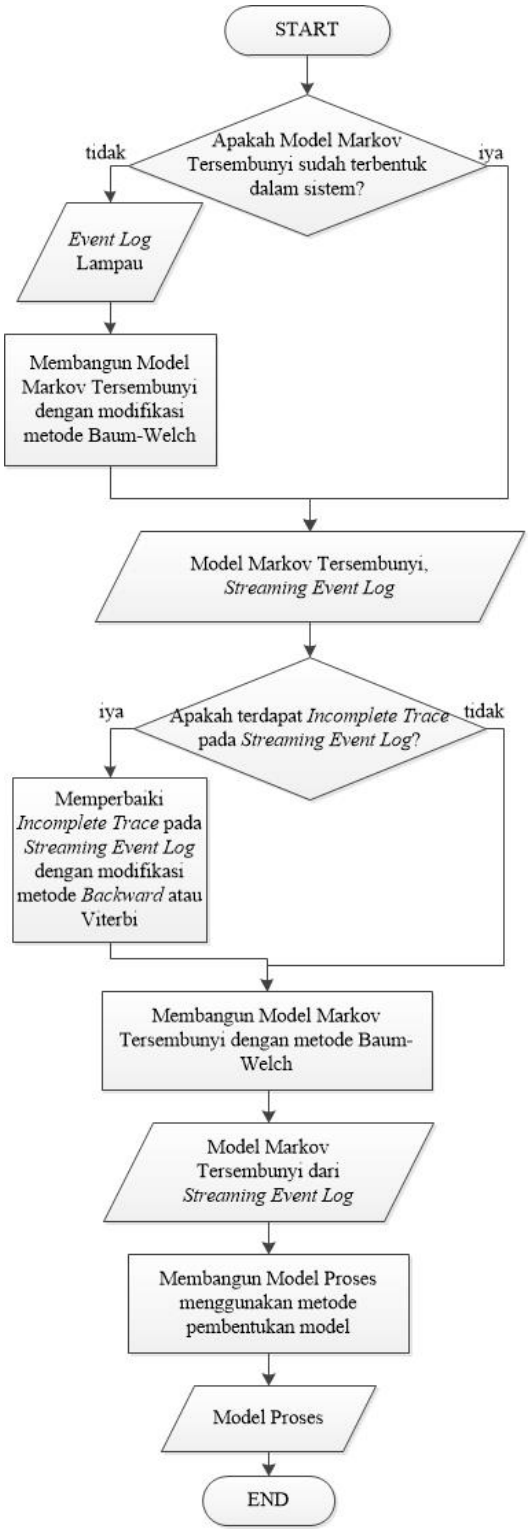
Faktanya, sistem infomasi saat ini menghasilkan *streaming event log* dimana keseluruhan *event log* tidak dapat tersimpan di dalam sistem [2]. Selain itu, adanya kemungkinan pertambahan aktivitas di tengah-tengah proses sehingga model proses diharuskan mampu menampilkan pertambahan aktivitas tersebut.

Online Heuristic Miner adalah algoritma *process discovery* untuk *streaming event log* dimana algoritma ini mampu beradaptasi dengan pertambahan aktivitas di tengah proses [2]. Kelemahan dari *Online Heuristic Miner* adalah kemunculan *incomplete trace* pada *streaming event log*. *Incomplete trace* adalah rangkaian aktivitas (*trace*) pada *event log* yang terpotong di bagian awal ataupun di bagian akhir *trace*. *Incomplete trace* dapat mengakibatkan model proses bisnis tidak akurat dikarenakan adanya aktivitas yang tidak terekam pada saat *process discovery*.

Oleh karena itu, algoritma yang memanfaatkan Model *Markov* Tersembunyi digunakan untuk membangun model proses yang mampu mengatasi kemunculan *incomplete trace*. Model *Markov* Tersembunyi telah digunakan di berbagai bidang untuk peramalan, seperti *speech recognition* [3], *gene prediction* [4] ataupun *stock market forecasting* [5]. Penelitian [6] telah memanfaatkan Model *Markov* Tersembunyi dalam *process discovery*. Akan tetapi, *process discovery* pada penelitian tersebut belum menggunakan data *streaming event log*, melainkan menggunakan data *event log* tetap.

Algoritma yang memanfaatkan Model *Markov* Tersembunyi digunakan untuk *process discovery* menggunakan data *streaming event log* dimana *process discovery* ini dapat menangani *incomplete trace*. Algoritma yang memanfaatkan Model *Markov* Tersembunyi terdiri dari metode pembentukan model proses serta metode yang dimodifikasi. Metode yang dimodifikasi adalah metode *Baum-welch* [7], metode *Viterbi* [7] dan metode *Backward* [7]. Metode *Baum-welch* digunakan untuk menentukan probabilitas pada Model *Markov* Tersembunyi, metode *Viterbi* digunakan untuk memperbaiki *incomplete trace* akibat terpotongnya aktivitas di bagian akhir dan metode *Backward* digunakan untuk memperbaiki *incomplete trace* akibat terpotongnya aktivitas di bagian awal. Pemodifikasian metode *Viterbi* maupun metode *Backward* digunakan untuk menanggulangi aktivitas baru yang belum terbentuk di Model *Markov* Tersembunyi dan pemodifikasian metode *Baum-welch* digunakan untuk mengatur kelompok *observer* yang bergantung pada *state* Model *Markov* Tersembunyi. Sedangkan, metode pembentukan model proses digunakan untuk menentukan relasi antar aktivitas sebagai bahan pembentukan model proses. Relasi antar aktivitas terdiri dari empat macam, yaitu relasi XOR, OR, AND dan *Sequence*. Uji coba akan dilakukan sebanyak tiga kali, dimana perbedaan dari ketiga uji coba tersebut adalah jangka waktu antar pembentukan model proses. Pengujian dimaksudkan untuk memunculkan beragam kemunculan *incomplete trace* pada *streaming event log*. Hasil uji coba menunjukkan bahwa nilai kualitas model proses dari algoritma yang memanfaatkan Model *Markov* Tersembunyi lebih tinggi dibandingkan model proses dari algoritma *Online Heuristic Miner*. Kualitas model

proses diukur berdasarkan 4 sisi yaitu *fitness* [8], presisi [1], generalisasi [9] dan *simplicity* [9].



Gambar 1. Alur Proses dari Algoritma yang memanfaatkan Model *Markov* Tersembunyi

Gambar 2. Gambaran Model *Markov* Tersembunyi

Tabel 1.
Metode *Baum-welch* yang dimodifikasi

Bagian	Penjelasan
Masukan Metode	*Event log* lampau.
Keluaran Metode	Model *Markov* Tersembunyi dari *event log* lampau.
Alur Metode	1. Pengelompokkan aktivitas sebagai *observer* dimana aktivitas yang mempunyai relasi paralel dengan aktivitas lain dimasukkan dalam kelompok yang sama. 2. Jumlah *state* adalah jumlah kelompok *observer* dimana *observer* dalam satu kelompok bergantung pada *state* yang sama. 3. Pembangunan Model *Markov* Tersembunyi menggunakan metode *Baum-welch* [11].

II. METODE PENELITIAN

A. Gambaran Umum Metode Penelitian

Metode untuk pembentukan model proses dari *streaming event log* adalah algoritma yang memanfaatkan Model *Markov* Tersembunyi, seperti yang dirunjukkan pada Gambar 1.

Algoritma yang memanfaatkan Model *Markov* Tersembunyi menggunakan *event log* lampau dan *streaming event log* sebagai masukan. *Event log* lampau adalah *event log* tetap, dimana seluruh *trace* (rangkaian aktivitas) sudah tersimpan secara keseluruhan dalam sistem. Sedangkan *streaming event log* adalah *event log* yang berjalan sesuai waktu eksekusi aktivitas pada *event log* tersebut.

B. Membangun Model Markov Tersembunyi berdasarkan Event log Lampau

Membangun Model *Markov* Tersembunyi berdasarkan *event log* lampau merupakan proses pertama dari algoritma yang memanfaatkan Model *Markov* Tersembunyi. Gambar 2 menunjukkan gambaran Model *Markov* Tersembunyi yang digunakan. Bentuk Model *Markov* Tersembunyi yang digunakan didasarkan pada model Petri net, dimana *state* pada Model *Markov* Tersembunyi merupakan *place* sedangkan *observer* pada Model *Markov* Tersembunyi merupakan nama aktivitas. Setiap *observer* bergantung pada *state* yang sama apabila *observer* tersebut memiliki relasi paralel. Penentuan relasi paralel mengacu pada metode *Process discovery based on Activity Lifespan* [10].

Metode *Process discovery based on Activity Lifespan* [10] mengemukakan bahwa suatu aktivitas memiliki relasi paralel dengan aktivitas lain apabila waktu mulai aktivitas tersebut terjadi sebelum waktu selesai aktivitas lain. Sesuai dengan metode *Process Discovery based on Activity Lifespan*, maka *event log* yang digunakan, baik *event log* lampau maupun *streaming event log*, menggunakan pencatatan waktu ganda (waktu mulai dan waktu selesai suatu aktivitas).

Tabel 2.
Pseudo-code metode *Backward* dan *Viterbi* yang dimodifikasi

Bagian	Penjelasan
Masukan Metode	A_L, B_L, π_L, *seq_stream* (*streaming event log*) .
Keluaran Metode	*seq_repair* (*streaming event log* yang telah diperbaiki).
Alur Metode	1. $X_0 \leftarrow seq\_stream[0]$ 2. while $\sum_{i=1}^{N} b_i(X_0) = 0$ do 3. *remove* X_0 4. $X_0 \leftarrow seq\_stream[0]$ 5. while $\pi_{stateX_0} = 0$ do 6. $S_S \leftarrow \arg\max_{j=0,...,N} Backward(j, stateX_0)$ 7. *addFront(seq_stream,*O_{S_S}*)* 8. $stateX_0 \leftarrow S_S$ 9. $X_T \leftarrow seq\_stream[-1]$ 10. while $\sum_{i=1}^{N} b_i(X_T) = 0$ do 11. *remove* X_T 12. $X_T \leftarrow seq\_stream[-1]$ 13. while $\sum_{i=1}^{N} a_{stateX_T j} > 0$ do 14. $S_S \leftarrow Viterbi(stateX_T)$ 15. *addLast(seq_stream,*O_{S_S}*)* 16. $stateX_T \leftarrow S_S$ 17. $seq\_repair \leftarrow seq\_stream$

dimana:

π_L	:	vektor probabilitas *state* sebagai *state* awal dari *event log* lampau.
$a_{stateX_T j}$	:	nilai probabilitas transisi dari *state* yang memiliki *observer* X_T ke *state j*.
addFront(seq_stream,O_{S_S}*)*	:	*observer* dari *state S* ditambahkan pada bagian pertama dari *incomplete trace*.
A_L	:	matrik probabilitas transisi *state* dari *event log* lampau.
addLast(seq_stream,O_{S_S}*)*	:	*observer* dari *state S* ditambahkan pada bagian akhir dari *incomplete trace*.
$b_i(X_0)$	:	nilai probabilitas *observer* X_0 terhadap *state i*.
B_L	:	matrik probabilitas *observer* dari *event log* lampau.
$Backward(j, stateX_0)$	:	nilai metode *Backward* [7] dari *state j* dimana $stateX_0$ merupakan *state* yang terjadi setelah *state j*.
N	:	jumlah *state* pada A_L .
seq_repair	:	hasil perbaikan *incomplete trace*.
$seq\_stream[0]$	:	aktivitas pertama pada *incomplete trace*.
$seq\_stream[-1]$	:	aktivitas terakhir pada *incomplete trace*.
π_{stateX_0}	:	nilai probabilitas *state* yang memiliki *observer* X_0 sebagai *state* awal.
$Viterbi(stateX_T)$	:	hasil dari metode *Viterbi* berupa *state* dimana $stateX_T$ merupakan *state* yang terjadi sebelum hasil metode *Viterbi* [11].
X_0	:	nama aktivitas awal pada *trace* di *event log*.
X_T	:	nama aktivitas akhir pada *trace* di *event log*.

Tabel 3.
Pseudo-code Pembangunan Model *Markov* Tersembunyi berdasarkan *Streaming Event log*

Bagian	Penjelasan
Masukan Metode	A_L, B_L, π_L, *seq_repair* .
Keluaran Metode	Model *Markov* Tersembunyi dari *Streaming Event log* (A_S, B_S, π_S) dan Model *Markov* Tersembunyi dari *Event log* Lampau (A_L, B_L, π_L).
Alur Metode	1. while $seq\_repair \neq null$ do 2. //Pembentukan Model Markov Tersembunyi (A_S, B_S, π_S)menggunakan metode Baum – Welch [10]. 3. if *length*(A_S)>*length*(A_L) then 4. $A \leftarrow A_S + A_L$ 5. else 6. $A \leftarrow A_L + A_S$ 7. if *length*(B_S)>*length*(B_L) then 8. $B \leftarrow B_S + B_L$ 9. else 10. $B \leftarrow B_L + B_S$ 11. $\pi \leftarrow \pi_S + \pi_L$ 12. $A_L \leftarrow A$ 13. $B_L \leftarrow B$ 14. $\pi_L \leftarrow \pi$

dimana:

π_L	:	vektor probabilitas *state* sebagai *state* awal dari *event log* lampau.
π_S	:	vektor probabilitas *state* sebagai *state* awal dari *streaming event log*.
A_L	:	matrik probabilitas transisi *state* dari *event log* lampau.
A_S	:	matrik probabilitas transisi *state* dari *streaming event log*.
B_L	:	matrik probabilitas *observer* dari *event log* lampau.
B_S	:	matrik probabilitas *observer* dari *streaming event log*.
seq_repair	:	rangkaian *observer* pada *streaming event log* yang mengalami perbaikan apabila terdapat *incomplete trace*.

Pembangunan Model *Markov* Tersembunyi berdasarkan *event log* lampau menggunakan metode *Baum-welch* yang dimodifikasi. Modifikasi yang dilakukan adalah penambahan ketentuan kebergantungan *observer* terhadap *state* pada Model *Markov* Tersembunyi berdasarkan metode *Process discovery based on Activity Lifespan* sebelum metode *Baum-welch* dijalankan. Alur Modifikasi Metode *Baum-welch* dipaparkan pada Tabel 1.

C. Memperbaiki Incomplete Trace pada Streaming Event log

Memperbaiki *incomplete trace* merupakan proses kedua dari alur algoritma yang memanfaatkan Model *Markov* Tersembunyi. Metode yang digunakan adalah modifikasi dari metode *Backward* [7] dan metode *Viterbi* [11]. Tabel 2 menunjukkan modifikasi dari metode *Backward* dan metode *Viterbi*.

Apabila *incomplete trace* terjadi diakibatkan terpotongnya aktivitas di bagian awal, maka menggunakan metode *Backward* yang dimodifikasi (alur *pseudo-code* baris 1-8). Sedangkan apabila *incomplete trace* terjadi akibat terpotongnya aktivitas di bagian akhir, maka menggunakan metode *Viterbi* yang dimodifikasi (alur *pseudo-code* 9-16). Pemodifikasian baik pada metode *Backward* mapupun *Viterbi* dilakukan dengan menambahkan aturan untuk menanggulangi aktivitas baru di tengah-tengah proses. Pemodifikasian dapat dilihat pada alur *pseudo-code* baris 2-4 dan baris 10-12 pada Tabel 2.

Tabel 4.
Pseudo-code Penentuan Relasi

Bagian	Penjelasan
Masukan Metode	A_S, B_S.
Keluaran Metode	Relasi antar aktivitas
Alur Metode	1. for S_i in [0,...,n_{A_S}] do 2. for S_j in [0,...,n_{A_S}] do 3. if $a_{S_iS_j}$ >0 then 4. if $(S_i \neq S_j) \vee ((S_i = S_j) \wedge n_{O_{S_i}} = 1)$ then 5. graph[act_before,act_after]=[O_{S_i}, O_{S_j}] 6. for act_before in graph do 7. if $n_{act\_after} > 1$ then 8. if act_before=act_after then 9. graph[act_before → act_after] = relasi *sequence* 10. else 11. graph[act_before → act_after] = R(A_S, B_S, act_before, act_after)+"Split" 12. if $n_{act\_before} > 1$ then 13. if act_before=act_after then 14. graph[act_before → act_after] = relasi *sequence* 15. else 16. if graph[act_before → act_after] = relasi XOR Split then 17. relation_before = graph[act_before,act_after] 18. graph[act_before,act_after] = relasi_before, R(A_S, B_S, act_before, act_after)+"Join" 19. else 20. graph[act_before,act_after] = R(A_S, B_S, act_before, act_after)+"Join"

dimana:

$a_{S_iS_j}$	:	nilai probabilitas transisi *state* S_i ke *state* S_j .
A_S	:	matrik probabilitas transisi *state* dari *streaming event log*.
B_S	:	matrik probabilitas *observer* dari *streaming event log*.
graph[act_before, act_after]	:	penampung aktivitas awal dan aktivitas akhir untuk penentuan relasi.
graph[act_before → act_after]	:	penampung relasi antara aktivitas awal menuju aktivitas akhir.
n_{act_after}	:	jumlah aktivitas akhir untuk pembentukan suatu relasi.
n_{act_before}	:	jumlah aktivitas awal untuk pembentukan suatu relasi.
n_{A_S}	:	jumlah *state* pada A_S .
$n_{O_{S_i}}$	:	jumlah *observer* yang bergantung pada *state* S_i .
O_{S_i}	:	kumpulan *observer* yang bergantung pada *state* S_i .
R(A_S, B_S, act_before, act_after)	:	fungsi untuk memanggil *pseudo-code* Penentuan Relasi XOR, OR, AND, dan *Sequence*.

Tabel 5.
Pseudo-code Penentuan Relasi XOR, OR, AND, dan *Sequence*

Bagian	Penjelasan
Masukan Metode	A_S, B_S, act_before, act_after
Keluaran Metode	*relation*
Alur Metode	1. i ← S_{act_before} 2. j ← S_{act_after} 3. if $(n_i = 1) \vee (n_j = 1)$ then 4. if $(n_i = 1 \wedge n_{j\rightarrow} > 1) \vee (n_j = 1 \wedge n_{i\rightarrow} > 1)$ then 5. *relation* = relasi XOR 6. else 7. graph[act_before → act_after] = relasi *sequence* 8. else 9. //menghitung RM(i → j) dan avgPP sesuai (1) dan (2) 10. if RM(i → j) ≥ avgPP then 11. *relation* = relasi AND 12. else 13. *relation* = relasi OR

dimana:

A_S	:	matrik probabilitas transisi *state* dari *streaming event log*.
B_S	:	matrik probabilitas *observer* dari *streaming event log*.
graph[act_before → act_after]	:	penampung relasi antara aktivitas awal menuju aktivitas akhir.
$n_{j\rightarrow}$	:	jumlah *state* yang bertransisi dengan *state j*.
relation	:	variabel untuk menyimpan relasi antar aktivitas.
S_{act_before}	:	*state* yang memiliki *observer* dimana *observer* tersebut merupakan aktivitas awal pada relasi.

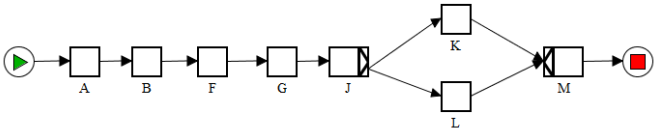
D. Membangun Model Markov Tersembunyi berdasarkan Streaming Event log

Membangun Model *Markov* Tersembunyi berdasarkan *Streaming Event Log* merupakan proses ketiga dari alur algoritma yang memanfaatkan Model *Markov* Tersembunyi. *Streaming event log* yang digunakan adalah *streaming event log* yang sudah mengalami perbaikan apabila *streaming event log* tersebut memiliki *incomplete trace*. Model *Markov* ini akan digunakan untuk membangun model proses.

Pembangunan Model *Markov* Tersembunyi berdasarkan *streaming event log* menggunakan metode *Baum-welch* dengan tambahan aturan untuk memperbaharui Model *Markov* Tersembunyi dari *event log* lampau yang digunakan untuk *streaming event log* selanjutnya. Tambahan aturan dilakukan dengan cara menggabungkan Model *Markov* Tersembunyi dari *event log* lampau dengan Model *Markov* Tersembunyi dari *streaming event log*. Aturan dapat dilihat pada alur *pseudo-code* Pembangunan Model *Markov* Tersembuyi baris 3-14. *Pseudo-code* tersebut dipaparkan pada Tabel 3.

E. Membangun Model Proses menggunakan Metode Pembentukan Model Proses

Penelitian mengajukan metode pembentukan model proses yang digunakan untuk membangun model proses. Metode pembentukan model proses memanfaatkan Model *Markov* Tersembunyi dari *streaming event log*. Metode ini dibagi ke dalam dua *pseudo-code* yaitu *pseudo-code* Penentuan Relasi Aktivitas dan *pseudo-code* Penentuan Relasi XOR, OR, AND, dan *Sequence.* Kedua *pseudo-code* tersebut dipaparkan pada Tabel 4 dan Tabel 5.



Gambar 3. Model Proses dari *Event log* Lampau

Pseudo-code Penentuan Relasi merupakan inti dari metode Pembentukan Model Proses, dimana alur *pseudo-code* baris 1-3 merupakan penentuan aktivitas yang berelasi dan alur *pseudo-code* baris 4-20 merupakan penentuan relasi antar aktivitas. Penambahan kata "*Split*" seperti pada *pseudo-code* baris 11 menunjukkan bahwa aktivitas pilihan termasuk dalam aktivitas akhir suatu relasi. Sebaliknya, kata "*Join*" pada *pseudo-code* baris 18 dan 20 menunjukkan bahwa aktivitas pilihan termasuk dalam aktivitas awal suatu relasi.

Penentuan relasi menggunakan *pseudo-code* Penentuan Relasi XOR, OR, AND dan *Sequence*. Alur *pseudo-code* baris 3-5 merupakan ketentuan pengunaan relasi XOR dan alur *psudo-code* baris 3-7 merupakan ketentuan penggunaan relasi *sequence*. Ketentuan penggunaan relasi OR dan AND diatur pada alur *pseudo-code* baris 8-13 dimana melibatkan (1) dan (2).

$$\mathrm{RM}(i \rightarrow j) = \frac{a_{ii}}{a_{ij}} \times \frac{1}{n_{O_i}} \quad (1)$$

$$\mathrm{avgPP} = \frac{\sum_{i=0}^{n_A} \sum_{j=0}^{n_A} a_{ij}}{n_{a_{ij}}} \quad (2)$$

Keterangan pada (1) dan (2):

a_{ii} : nilai probabilitas transisi *state i* ke *state* itu sendiri.

a_{ij} : nilai probabilitas transisi *state i* ke *state j* (a_{ij}) dengan syarat nilai a_{ij} lebih dari 0.

n_A : jumlah *state* pada matrik probabilitas transisi *state* (A)

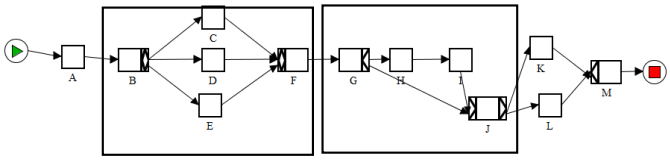
n_{O_i} : jumlah *observer k* yang bergantung pada *state i*.

$n_{a_{ij}}$: jumlah banyaknya a_{ij} yang bernilai lebih dari 0.

III. HASIL UJI COBA

Penelitian ini menggunakan data studi kasus yang merupakan modifikasi dari data penanganan ulasan jurnal.

Data studi kasus ini diambil dari *event log Handling for A Journal* pada *website* http://www.processmining.org/. Data studi kasus terdiri dari 100 *trace* dimana 20 *trace* digunakan sebagai *event log* lampau (*data training*) dan 80 *trace* digunakan sebagai *streaming event log* (*data testing*). Setiap nama aktivitas diinisialisasi dengan abjad dan aktivitas pada *streaming event log* disimulasikan berjalan dalam kurun waktu 0,5 detik.



Gambar 4. Model Proses dari *Streaming Event log*

Pemodifikasian dilakukan dengan mengembangkan proses pada model proses dari *event log* lampau. Model proses dari *event log* lampau ditunjukkan pada Gambar 3 sedangkan alur proses dalam kotak pada Gambar 4 menunjukkan pengembangan proses pada *streaming event log*.

Pemodifikasian dilakukan dengan mengembangkan proses pada model proses dari *event log* lampau. Model proses dari *event log* lampau ditunjukkan pada Gambar 3 sedangkan alur proses dalam kotak pada Gambar 4 menunjukkan kejadian pengembangan proses pada *streaming event log*.

Pada kotak pertama Gambar 4, dapat dilihat bahwa pengembangan proses dilakukan dengan penambahan aktivitas C,D,E diantara alur proses aktivitas B ke aktivitas F sehingga terdapat relasi tambahan yaitu relasi OR. Selain itu, terdapat pengembangan proses lainnya yaitu penambahan aktivitas H dan I diantara alur proses aktivitas G ke aktivitas J. Hal itu mengakibatkan munculnya relasi baru yaitu XOR.

Penelitian ini melakukan uji coba sebanyak tiga kali, yaitu pembentukan model proses menggunakan algoritma yang memanfaatkan Model *Markov* Tersembunyi dan algoritma pembanding, *Online Heuristic Miner* setiap 5 detik, setiap 10 detik dan setiap 15 detik. Variasi uji coba tersebut digunakan untuk memunculkan variasi data masukan dalam pembentukan model proses. Setiap model proses yang dihasilkan akan diukur kualitasnya dari sisi *fitness*, presisi, generalisasi dan *simplicity*. Hasil kualitas model proses tersebut ditunjukan pada Tabel 6.

IV. KESIMPULAN/RINGKASAN

Penelitian ini memaparkan algoritma yang digunakan untuk menghasilkan model proses dari *streaming event log* dengan memanfaatkan Model *Markov* Tersembunyi.

Algoritma yang memanfaatkan Model *Markov* Tersembunyi memiliki beberapa tahapan. Tahapan pertama adalah pembentukan Model *Markov* Tersembunyi dari *event log* lampau dengan menggunakan modifikasi metode *Baum-Welch*. Pemodifikasian metode *Baum-welch* adalah penambahan aturan penentuan parameter Model *Markov* Tersembunyi sebelum dilakukan metode *Baum-welch*. Tahapan kedua adalah perbaikan *incomplete trace* dari *streaming event log* dengan menggunakan modifikasi metode *Viterbi* dan metode *Backward*. Pemodifikasian metode adalah penambahan aturan untuk menanggulangi aktivitas yang belum diinisialisasi pada Model *Markov* Tersembunyi serta penambahan pencarian *observer* sebagai pengganti aktivitas terpotong pada *incomplete trace*. Tahapan ketiga adalah pembentukan Model *Markov* Tersembunyi dari *streaming event log* yang telah diperbaiki dan tahapan terakhir adalah pembentukan model proses dengan menggunakan Model *Markov* Tersembunyi dari *streaming event log* dan metode

pembentukan proses. Metode pembentukan proses terdiri dari peraturan serta rumus yang merupakan gagasan penelitian.

Tabel 6.
Hasil kualitas model proses dari sisi *fitness*, presisi, generalisasi dan *simplicity*

Metode	Uji Coba	Total Q_f	Total Q_p	Total Q_g	Total Q_s
Algoritma yang memanfaatkan Model *Markov* Tersembunyi	5 detik	0,982	0,448	0,887	0,947
	10 detik	0,983	0,435	0,862	0,94
	15 detik	0,985	0,553	0,861	0,937
	Rata-rata	0,983	0,479	0,87	0,941
Online Heuristic Miner	5 detik	0,912	0,406	0,88	0,927
	10 detik	0,885	0,375	0,851	0,925
	15 detik	0,839	0,461	0,854	0,92
	Rata-rata	0,879	0,414	0,862	0,924

dimana:

Total Q_f : nilai rata-rata dari kualitas sisi *fitness* semua model proses pada masing-masing uji coba.

Total Q_p : nilai rata-rata dari kualitas sisi presisi semua model proses pada masing-masing uji coba.

Total Q_g : nilai rata-rata dari kualitas sisi generalisasi semua model proses pada masing-masing uji coba.

Total Q_s : nilai rata-rata dari kualitas sisi generalisasi semua model proses pada masing-masing uji coba.

Rata-rata : nilai rata-rata dari kualitas model proses untuk masing-masing sisi kualitas pada uji coba dengan jangka waktu antar pembentukan model proses selama 5 detik, 10 detik dan 15 detik.

Berdasarkan nilai rata-rata pada Tabel 6, hasil uji coba menunjukkan bahwa nilai kualitas dari sisi *fitness*, presisi, generalisasi dan *simplicity* model proses yang dibentuk oleh algoritma yang memanfaatkan Model *Markov* Tersembunyi lebih tinggi dibandingkan model proses yang dibentuk oleh algoritma *Online Heuristic Miner*. Hal tersebut membuktikan bahwa perbaikan *incomplete trace* pada algoritma yang memanfaatkan Model *Markov* Tersembunyi mengakibatkan kualitas model proses yang dibentuk oleh algoritma tersebut lebih baik dibandingkan kualitas model proses dari algoritma *Online Heuristic Miner*.

UCAPAN TERIMA KASIH

Penulis K.R.S mengucapkan terima kasih kepada Tuhan Yang Maha Esa, Orang Tua, Bapak Dosen Pembimbing, Bapak Ibu Dosen Teknik Informatika ITS serta Teman-Teman Teknik Infomatika ITS yang mendukung penelitian ini.

DAFTAR PUSTAKA

[1] W. M. P. van der Aalst, *Process Mining - Discovery, Conformance and Enhancement of Business Processes*, Schleiden: Springer (2010).

[2] A. S. Burattin dan W. M. P. van der Aalst, "Heuristic Miners for Streaming Event Data," *ArXiv CoRR*, 2012.

[3] K. Audhkhasi, O. Osoba dan B. Kosko, "Noisy hidden *Markov* models for speech recognition," dalam *Neural Networks (IJCNN), The 2013 International Joint Conference*, 2013.

[4] A. C. Testa, J. Hane, S. Ellwood dan R. P. Oliver, "Coding Quarry: highly accurate hidden *Markov* model gene prediction in fungal genomes using RNA-seq transcripts," *BMC genomics*, vol. 16, no. 1, p. 1, 2015.

[5] M. R. Hassan, K. Ramamohanarao, J. Rahman dan M. M. Hossain, "A HMM-based adptive fuzzy inference system for stock market forecasting," *Neurocomputing*, vol. 104, pp. 10-25, 2013.

[6] R. Sarno and K. R. Sungkono, "Hidden *Markov* Model for Process Mining of Parallel Business Processes," *International Review on Computers and Software (IRECOS)*, vol. 11, no. 4, pp. 290-300, 2016.

[7] A. Tenyakov, "Estimation of Hidden *Markov* Models and Their Applications in Finance," Electronic Thesis and Dissertation Repository, 2014.

[8] S. De Cnudde, J. Claes, dan G. Poels, "Improving the quality of the Heuristics Miner in ProM 6.2," *Expert Systems with Applications*, vol. 41, no. 17, pp. 7678–7690, 2014.

[9] C. A. M. Buijs, B. F. van Dongen dan W. M. P. van der Aalst, "On The Role of Fitness, Precision, Generalization and Simplicity in Process Discovery," dalam *Lecture Notes in Computer Science*, Berlin, Springer-Verlag, 2012, pp. 305-322.

[10] R. A. Sutrisnowati, H. Bae, L. Dongha dan K. Minsoo, "Process Model Discovery based on Activity Lifespan," dalam *International Conference on Technology Innovation and Industrial Management*, Seoul, 2014.

[11] P. Dymarski, Hidden *Markov* Model, Theory and Application, India: InTech, 2011.