ANALISIS PERBANDINGAN ALGORITMA *HEURISTIC MINER* DAN *INDUCTIVE MINER* DALAM *PROCESS MINING*

SKRIPSI

Oleh :
MUHAMMAD ZAKIN NADA RAYA
NIM. 210605110135

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

ANALISIS PERBANDINGAN ALGORITMA *HEURISTIC MINER* DAN *INDUCTIVE MINER* DALAM *PROCESS MINING*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
MUHAMMAD ZAKIN NADA RAYA
NIM. 210605110135

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

HALAMAN PERSETUJUAN

ANALISIS PERBANDINGAN ALGORITMA *HEURISTIC MINER* DAN *INDUCTIVE MINER* DALAM *PROCESS MINING*

SKRIPSI

Oleh :
MUHAMMAD ZAKIN NADA RAYA
NIM. 210605110135

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 05 Juni 2025

Pembimbing I,	Pembimbing II,
Dr. M. Ainul Yaqin, M.Kom	Ashri Shabrina Afrah, M.Kom
NIP. 19761013 200604 1 004	NIP. 19900430 202012 1 003

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPU
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

ANALISIS PERBANDINGAN ALGORITMA *HEURITIC MINER* DAN *INDUCTIVE MINER* DALAM *PROCESS MINING*

SKRIPSI

Oleh :
MUHAMMAD ZAKIN NADA RAYA
NIM. 210605110135

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 16 Juni 2025

Susunan Dewan Penguji

Ketua Penguji	: Syahiduz Zaman, M.Kom NIP. 19700502 200501 1 005	( )
Anggota Penguji I	: Dr. Totok Chamidy, M.Kom NIP. 19691222 200604 1 001	( )
Anggota Penguji II	: Dr. M. Ainul Yaqin, M.Kom NIP. 19761013 200604 1 004	( )
Anggota Penguji III	: Ashri Shabrina Afrah, M.Kom NIP. 19900430 202012 1 003	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Ir. Fachrul Kurniawan, M.MT, IPU
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	:	Muhammad Zakin Nada Raya
NIM	:	210605110135
Fakultas / Program Studi	:	Sains dan Teknologi / Teknik Informatika
Judul Skripsi	:	Analisis Perbandingan Algoritma *Heuristic Miner* dan *Inductive Miner* dalam *Proses Mining*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 16 Juni 2025
Yang membuat pernyataan

Muhamad Zakin Nada Raya
NIM. 210605110135

MOTTO

... Berikanlah yang Terbaik, Berikanlah yang Terhebat ...

HALAMAN PERSEMBAHAN

Dengan hati yang penuh syukur ke hadirat Sang Maha Pencipta, atas limpahan rahmat, kesehatan, serta kekuatan yang tak terhingga, sehingga karya sederhana ini dapat terselesaikan dengan baik. Dengan segala kerendahan hati, penulis mempersembahkan skripsi ini kepada:

Ayah dan Ibu tercinta,

Mokh. Imam Dardiri dan Endang Puji Astutik

Cahaya yang tak pernah padam dalam setiap langkah, penopang doa di setiap rintangan. Terima kasih atas cinta tanpa syarat, ketulusan tanpa pamrih, dan pengorbanan yang tak mungkin terbalas oleh apa pun di dunia ini.

Kakakku tersayang,

Muhammad Wafi Ruafi Ar-raya

Sumber semangat dan inspirasi, tempat berpulang di tengah hiruk-pikuk dunia. Terima kasih atas dukungan dan kebersamaan yang menguatkan di setiap waktu.

Teman-teman seperjuangan,

Keluarga besar Teknik Informatika Angkatan 2021 “ASTER”

Terima kasih atas tawa yang menenangkan, diskusi yang mencerahkan, dan perjuangan yang kita lalui bersama. Semoga tali silaturahmi yang terjalin tetap kuat, dan kesuksesan senantiasa menyertai langkah kita ke depan.

KATA PENGANTAR

Bismillahirrahmaanirrahiim, Assalamu'alaikum wr. wb.

Segala puji dan syukur penulis panjatkan ke hadirat Allah Subhanahu wa ta'ala atas limpahan rahmat, taufik, dan hidayah-Nya sehingga penulisan skripsi yang berjudul "Analisis Perbandingan Algoritma *Heuristic Miner* dan *Inductive Miner* dalam *Proses Mining*" ini dapat diselesaikan dengan baik. Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Penulis menyadari bahwa pencapaian ini tidak lepas dari bantuan, bimbingan, doa, serta dukungan dari berbagai pihak. Oleh karena itu, dengan segala kerendahan hati, penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Prof. Dr. M. Zainuddin, M.A., selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang, atas kepemimpinan dan kebijakan yang terus mendorong peningkatan mutu pendidikan dan riset di lingkungan universitas.
2. Prof. Dr. Hj. Sri Harini, M.Si., selaku Dekan Fakultas Sains dan Teknologi, yang telah memberikan dukungan penuh dalam proses akademik dan non-akademik mahasiswa.
3. Dr. Fachrul Kurniawan, M.MT., IPM., selaku Ketua Program Studi Teknik Informatika, atas arahan dan semangatnya dalam membina mahasiswa menjadi insan akademik yang kompeten dan berakhlak.

4. Dr. M. Ainul Yaqin, M.Kom., selaku Dosen Pembimbing I, atas segala bimbingan, kesabaran, dan ilmu yang telah diberikan selama proses penyusunan skripsi ini.
5. Ashri Shabrina Afrah, M. T., selaku Dosen Pembimbing II, atas masukan berharga dan dukungan yang sangat membantu dalam penyelesaian karya tulis ini.
6. Syahiduz Zaman, M.Kom. dan Dr. Totok Chamidy, M.Kom., selaku dosen penguji, atas waktu, perhatian, dan evaluasi objektif yang sangat membantu penulis dalam melihat kelemahan serta kelebihan penelitian ini secara jernih.
7. Seluruh staf dan dosen Program Studi Teknik Informatika, yang telah menanamkan ilmu, semangat, dan nilai-nilai akademik selama proses perkuliahan.
8. Keluarga tercinta: Bapak Imam Dardiri dan Ibu Endang Puji Astutik yang selalu memberikan doa, dukungan, dan kasih sayang yang tiada henti. Terima kasih atas pengorbanan, kesabaran, dan semangat yang menjadi sumber kekuatan dalam menjalani setiap proses perkuliahan hingga penyusunan skripsi ini. Segala pencapaian ini tidak akan berarti tanpa restu dan cinta kalian.
9. Teman-teman seperjuangan di Kelas E Teknik Informatika, yang telah menjadi bagian dari perjalanan akademik ini. Terima kasih atas kebersamaan, kerja sama, canda tawa, serta semangat yang terus menyertai selama masa perkuliahan. Dukungan kalian menjadi salah satu penyemangat dalam menyelesaikan setiap tantangan di bangku kuliah.

10. Keluarga besar Teknik Informatika angkatan 2021 “ASTER”, atas solidaritas, semangat kolaborasi, dan kebersamaan yang tak akan pernah terlupakan. Terima kasih telah menjadi bagian dari perjalanan ini, menghadirkan warna dan kenangan indah selama masa perkuliahan.
11. Teman terbaik penulis, Gigih Agung Prasetyo, yang telah menjadi teman diskusi dan revisi. Terima kasih atas waktu, pemikiran, dan bantuan yang tak ternilai dalam membantu penulis menghadapi berbagai tantangan, terutama saat mencari solusi dari permasalahan dalam penelitian ini.
12. Teman terdekat penulis, Aninda Rizky Hartanti, yang telah hadir dalam banyak momen penting selama proses penyusunan skripsi. Terima kasih atas kebersamaan, ajakan nugas, dan waktu yang diluangkan, semua itu membantu penulis untuk tetap konsisten menyelesaikan tugas hingga akhir

Penulis menyadari bahwa karya ini masih jauh dari kata sempurna. Oleh karena itu, segala bentuk kritik dan saran yang membangun sangat penulis harapkan demi peningkatan kualitas di masa yang akan datang. Semoga skripsi ini dapat memberikan manfaat nyata, menjadi referensi bagi penelitian selanjutnya, dan berkontribusi dalam pengembangan teknologi berbasis nilai-nilai keislaman.

Malang, 16 Juni 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Raya, Muhammad Zakin Nada. 2025. **Analisis Perbandingan Algoritma Heuristic Miner dan Inductive Miner Dalam Process Mining.** Thesis. Skripsi. Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. M. Ainul Yaqin, M.Kom (II) Ashri Shabrina Afrah, M.T.

.

Kata kunci: Process Mining, Inductive Miner, Heuristic Miner.

Pertumbuhan pesat data digital dalam beberapa tahun terakhir telah mendorong pentingnya penemuan pengetahuan dan penambangan data, terutama dalam konteks manajemen proses bisnis. *Process mining* muncul sebagai bidang yang menjembatani kesenjangan antara *data mining* dan manajemen proses, memberikan wawasan untuk perbaikan efisiensi dan identifikasi pola dalam proses bisnis. Penelitian ini menganalisis perbandingan antara *heuristic miner* dan *inductive miner* dalam *process* mining untuk memodelkan proses bisnis dengan data yang memiliki karakteristik yang berbeda. Fokus utama dari penelitian ini adalah untuk mengeksplorasi pengaruh kompleksitas model terhadap akurasi model hasil *process mining* yang dihasilkan oleh kedua metode tersebut. Analisis dilakukan dengan membandingkan tingkat kesamaan antara model acuan dan model hasil *process mining*. Hasil penelitian menunjukkan bahwa *inductive miner infrequent & all operators* memberikan hasil rata rata tertinggi, menunjukkan kesesuaian model yang lebih baik dan nilai *similarity* lebih tinggi yaitu 98.62% dan untuk *heuristic miner* yang terbaik adalah parameter *dependency threshold* lebih dari sama dengan 90 dengan presentase *similarity* sebesar 94,63%. Kompleksitas model tidak menunjukkan pengaruh signifikan terhadap akurasi kecocokan antara model acuan dan model *process mining* yang dihasilkan.

ABSTRACT

Raya, Muhammad Zakin Nada. 2025. **Comparison Analysis of Heuristic and Inductive Algorithms in Process Mining.** Thesis. Informatics Engineering Study Program, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University Malang. Supervisor: : (I) Dr. M. Ainul Yaqin, M.Kom (II) Ashri Shabrina Afrah, M.T.

Key words: Process Mining, Inductive Miner, Heuristic Miner.

The rapid growth of digital data in recent years has driven the importance of knowledge discovery and data mining, especially in the context of business process management. Process mining is emerging as a field that bridges the gap between data mining and process management, providing insights for efficiency improvement and pattern identification in business processes. This research analyzes the comparison between heuristic miner and inductive miner in process mining to model business processes with data having different characteristics. The main focus of this research is to explore the effect of model complexity on the accuracy of process mining models produced by both methods. The analysis is done by comparing the similarity rate between the reference model and the result of process mining. The results show that the inductive miner infrequent & all operators gives the highest average result, showing better model fit and higher similarity value of 98.62% and for the heuristic miner the best is the dependency threshold parameter more than equal to 90 with a similarity percentage of 94.63%. Model complexity does not show a significant effect on the accuracy of the match between the reference model and the resulting process mining model.

مستخلص البحث

ر يا، محمد زكين ندا. **2025**. **تحليل مقارن للخوارزميات الاستدلالية والاستقرائية في عملية التعدين**. الأطروحة. دراسة هندسة المعلوماتية، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف (أولاً) د. م. عين .اليقين، م. كوم (ثانيًا) عشري شبرينة أفراح، م. ت

الكلمات الرئيسية: عملية التعدين، عامل التعدين الاستقرائي، عامل التعدين الاستقرائي، عامل التعدين الاستدلالي

أدى النمو السريع للبيانات الرقمية في السنوات الأخيرة إلى تعزيز أهمية اكتشاف المعرفة والتنقيب في البيانات، خاصةً في سياق إدارة العمليات التجارية. يبرز التنقيب في العمليات كمجال يسد الفجوة بين التنقيب في البيانات وإدارة العمليات، مما يوفر رؤى لتحسين الكفاءة وتحديد الأنماط في العمليات التجارية. ويحلل هذا البحث المقارنة بين المنقّب الاستدلالي والمنقّب الاستقرائي في التنقيب عن العمليات لنمذجة العمليات التجارية ذات البيانات ذات الخصائص المختلفة. ينصب التركيز الرئيسي لهذا البحث على استكشاف تأثير تعقيد النموذج على دقة نماذج التنقيب عن العمليات التي تنتجها كلتا الطريقتين. يتم إجراء التحليل من خلال مقارنة معدل التشابه بين النموذج المرجعي ونتيجة التنقيب عن العمليات. تُظهر النتائج أن عامل التعدين الاستقرائي غير المتكرر وجميع المشغلين يعطي أعلى متوسط نتيجة، ويظهر ملاءمة أفضل للنموذج وقيمة تشابه أعلى بنسبة 98.62% وبالنسبة لعامل التعدين الاستدلالي فإن أفضلها هو معلمة عتبة التبعية أكثر من 90 مع نسبة تشابه 94.63.% لا يُظهر تعقيد النموذج تأثيرًا كبيرًا على دقة .التطابق بين النموذج المرجعي ونموذج التنقيب عن العمليات الناتج

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pesatnya pertumbuhan data digital dalam beberapa tahun terakhir telah meningkatkan perhatian terhadap penemuan pengetahuan dan penambangan data, seiring dengan kebutuhan yang mendesak untuk mengubah data tersebut menjadi informasi dan pengetahuan yang bermanfaat (Rani, 2021). Saat ini, organisasi menghadapi tantangan dalam menganalisis data kompleks yang dihasilkan dari berbagai aktivitas bisnis dan sistem digital (Parimala et al., 2017). *Process mining* muncul sebagai bidang baru yang menjembatani kesenjangan antara *data mining* dan manajemen proses bisnis, membantu mengidentifikasi pola, mengukur efisiensi, dan memberikan wawasan untuk perbaikan proses (Kamala, 2019).

Process mining adalah metode analisis data yang bertujuan untuk mengevaluasi dan menganalisis proses bisnis berdasarkan data yang direkam dalam sistem informasi. Metode ini mengekstraksi pengetahuan dari log kejadian (*event logs*) untuk membangun model proses bisnis yang sebenarnya, mengidentifikasi inefisiensi, dan menemukan area yang dapat ditingkatkan (Aalst & Dustdar, 2012).

Konsep *process mining* ini memiliki kesamaan dengan prinsip pencatatan amal dalam Islam, sebagaimana yang dijelaskan dalam Surah Al-Qamar ayat 52-53:

وَكُلُّ شَيْءٍ فَعَلُوْهُ فِى الزُّبُرِ وَ كُلُّ صَغِيْرٍ وَّكَبِيْرٍ مُّسْتَطَرٌ

"*Segala sesuatu yang telah mereka perbuat (tertulis) dalam buku-buku catatan (amal). Dan setiap hal yang kecil maupun besar adalah tertulis." (QS. Al-Qamar: 52-53)*

Dalam *Tafsir Al-Mishbah*, Quraish Shihab (Shihab, 2005) menjelaskan bahwa ayat ini menunjukkan bahwa segala tindakan manusia, baik yang kecil maupun yang besar, semuanya tercatat dengan rapi dan sempurna dalam buku catatan amal. Tidak ada satu pun perbuatan yang luput dari pencatatan, sekecil apapun itu. Prinsip ini sangat relevan dengan konsep *process mining* dalam konteks bisnis, setiap aktivitas yang tercatat dalam *event logs* dianalisis secara menyeluruh untuk memantau kinerja, mengidentifikasi kesalahan, dan meningkatkan efisiensi. Sama seperti pencatatan amal yang menekankan keadilan dan akurasi, *process mining* memastikan bahwa setiap tindakan bisnis terdokumentasi dengan tujuan evaluasi dan perbaikan. Prinsip ini mengingatkan kita bahwa setiap tindakan akan diperhitungkan dan memiliki dampak di masa depan, baik di dunia bisnis maupun dalam kehidupan akhirat menurut ajaran Islam.

Process mining terdiri dari tiga kegiatan utama: *discovery*, *conformance*, dan *enhancement* (Aalst, 2016). *Discovery* adalah kegiatan paling mendasar yang bertujuan untuk membangun model proses berdasarkan data *event logs* tanpa memerlukan pengetahuan sebelumnya tentang struktur proses tersebut. Beberapa algoritma *discovery* yang sering digunakan antara lain *alpha miner* (mengidentifikasi pola urutan aktivitas secara sederhana), *heuristic miner* (memperhitungkan frekuensi dan hubungan aktivitas), dan *inductive miner* (menggunakan pendekatan rekursif untuk menghasilkan model yang valid). selain

itu, ada *genetic miner* yang berbasis algoritma genetika untuk menemukan model optimal, dan *region-based mining* yang menggunakan teori *region* untuk menghasilkan model dengan akurasi matematis. Sementara itu, *conformance* bertujuan untuk mengevaluasi kesesuaian model dengan proses nyata, dan *enhancement* digunakan untuk memperbaiki dan memperkaya model dengan informasi tambahan. *Discovery* menjadi kunci dalam *process mining* karena menentukan dasar analisis dan model yang akan digunakan.

Di antara berbagai algoritma *discovery* yang telah disebutkan, *heuristic miner* dan *inductive miner* merupakan dua pendekatan yang paling populer dalam praktik *process mining*. Kedua algoritma ini memiliki karakteristik yang berbeda dalam cara mereka memodelkan dan menyederhanakan proses yang ada. Menurut (W. V. D. Aalst, 2016) dalam bukunya *"Process Mining: Data Science in Action," heuristic miner* adalah algoritma yang memperhitungkan frekuensi kejadian dan urutannya saat membangun model proses. Algoritma ini lebih fokus pada jalur yang sering terjadi dan mengabaikan perilaku dengan frekuensi rendah atau yang dianggap sebagai *noise*. Dengan menggunakan grafik ketergantungan yang mirip dengan relasi "langsung mengikuti," *heuristic miner* melengkapi model dengan pengukuran frekuensi untuk mempelajari *causal nets* (C-nets), yang secara efektif merepresentasikan alur proses. Di sisi lain, *inductive miner* seperti dijelaskan oleh Van Der Aalst, adalah algoritma yang menggunakan pendekatan rekursif untuk memecah *event logs* menjadi *sublog* yang lebih kecil. Pendekatan ini bertujuan untuk menemukan model proses secara sistematis, dengan memastikan bahwa model yang dihasilkan adalah *sound* (valid secara formal) dan mampu

merepresentasikan perilaku dalam *event logs. Inductive miner* dikenal karena fleksibilitas, skalabilitas, dan kemampuannya untuk menangani kompleksitas proses dunia nyata.

Meskipun kedua algoritma ini memiliki tujuan yang sama, yaitu menemukan model proses yang merepresentasikan data *event logs*, namun keduanya memiliki perbedaan pendekatan terhadap kompleksitas model. *Heuristic miner*, menggunakan pendekatan berbasis frekuensi yang lebih fleksibel, memungkinkan untuk menangani *loop,* tugas duplikat, dan aliran kontrol yang lebih kompleks. Akan tetapi, akibat pendekatannya lebih heuristik dan kurang terstruktur, akurasi modelnya dapat menurun dalam situasi dengan banyak *noise* atau data yang tidak lengkap (Peng et al., 2021). *Inductive miner* memiliki keuntungan dalam menghasilkan model yang lebih sederhana dan akurat dalam situasi yang membutuhkan presisi tinggi, khususnya pada *log* dengan aliran kompleks (Andreswari et al., 2021). Algoritma yang baik dalam *process mining* dirancang untuk mempertahankan akurasi tinggi bahkan ketika menangani model yang kompleks (Augusto et al., 2022). Algoritma *process mining* sangat memengaruhi hasil secara langsung yang pada gilirannya juga memiliki pengaruh signifikan terhadap para pembuat kebijakan di perusahaan. Di era *big data* saat ini, organisasi menghasilkan data proses dalam jumlah besar, dan mereka memerlukan metode yang efektif untuk mengekstraksi informasi berharga dari data ini guna mendukung pengambilan keputusan yang lebih baik (Wu et al., 2008). Oleh karena itu penting untuk menentukan algoritma mana yang lebih efektif dan efisien, dan salah satu

aspek penting dalam memilih metode *process mining* adalah mempertimbangkan tingkat akurasi yang dihasilkan dalam menangani data kompleks.

Sebelum melakukan penelitian, penting untuk memahami studi terdahulu di bidang ini. Hasyim et al. (2020) membahas penggunaan algoritma *inductive miner* dalam *process mining* untuk 9 skenario proses bisnis, menemukan bahwa meskipun mencapai tingkat kesamaan tertinggi 60% pada salah satu model, algoritma ini memiliki keterbatasan dalam representasi akurat model proses. (Bakhshi et al., 2023) membandingkan *heuristics miner* dan *inductive miner* dalam memodelkan jalur perawatan pasien sepsis. Hasilnya menunjukkan kedua metode memiliki keterbatasan, dengan *heuristics miner* memberikan *fitness* 77.4% dan kesederhanaan 47%, sementara *inductive miner* mencapai *fitness* 84.8% dan kesederhanaan 62.2%. Meski lebih sederhana, *inductive miner* masih gagal merepresentasikan struktur proses secara jelas, sehingga model berbasis pengetahuan ahli menjadi alternatif yang lebih efektif. Sahlabadi et al. (2023) membandingkan lima algoritma *process miner* untuk evaluasi arsitektur perangkat lunak, menemukan bahwa *heuristic miner* unggul dengan nilai *fitness* tinggi (0.6 - 0.812) dan model yang lebih sederhana (30-50 *arc*), dibandingkan algoritma lain yang menghasilkan model kompleks dan sulit dianalisis. Nuritha & Mahendrawathi (2017) membahas penggunaan *inductive miner* dan *heuristic miner* untuk memodelkan proses bisnis yang mengandung *noise* pada *event logs*. *Inductive miner* menunjukkan kinerja lebih baik dengan *similarity* 1 pada sebagian besar model, sementara *heuristic miner* lebih bervariasi (nilai tertinggi 0,75). *Inductive miner* memiliki rata-rata *similarity* lebih tinggi (0,736 untuk industri semen dan 1

untuk industri farmasi) dibandingkan dengan *heuristic miner* (0,325 untuk semen dan 0,390625 untuk farmasi). Kurniati et al. (2016) menguji *heuristic miner* pada berbagai *event logs* dan menemukan kinerjanya baik dengan *fitness* lebih dari 0,8, meskipun sangat bergantung pada pengaturan parameter, dengan *positive observation threshold* tetap signifikan pada *event logs* besar.

Dengan perbedaan pendekatan antara *heuristic miner* dan *inductive miner*, penting untuk melakukan analisis perbandingan antara keduanya untuk mengetahui bagaimana kedua algoritma tersebut menangani kompleksitas model dan bagaimana hal itu memengaruhi akurasi dalam representasi proses. Penelitian ini diharapkan dapat memberikan wawasan lebih dalam mengenai metode mana yang lebih optimal untuk digunakan dalam berbagai skenario proses bisnis serta bagaimana kompleksitas model mempengaruhi hasil yang diharapkan, dengan cara membandingkan tingkat kesamaan antara proses bisnis rujukan dan hasil proses bisnis yang dihasilkan oleh kedua metode.

Melalui analisis ini, diharapkan diperoleh pemahaman yang lebih baik mengenai pengaruh kompleksitas model terhadap akurasi dalam *process mining* serta keunggulan dan keterbatasan masing-masing metode dalam menghasilkan model yang representatif.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah yang diambil adalah sebagai berikut:

1. Bagaimana Pengaruh tingkat kompleksitas model proses bisnis terhadap akurasi *process mining* yang dihasilkan oleh metode *inductive miner* dan *heuristic miner*?
2. Sejauh mana kompleksitas model proses bisnis memengaruhi kinerja metode *heuristic miner* dan *inductive miner* dalam menghasilkan model proses yang akurat?

1.3 Batasan Masalah

Data yang digunakan adalah model proses bisnis yang memiliki 10 karakteristik yang berbeda.

1.4 Tujuan Penelitian

1. Menganalisis pengaruh tingkat kompleksitas model proses bisnis terhadap akurasi *process mining* yang dihasilkan oleh metode *inductive miner* dan *heuristic miner*.
2. Menilai sejauh mana kompleksitas model proses bisnis memengaruhi kinerja metode *heuristic miner* dan *inductive miner* dalam menghasilkan model proses yang akuat.

1.5 Manfaat Penelitian

1. Memberikan pemahaman tentang pengaruh kompleksitas model proses bisnis terhadap akurasi *process mining*, sehingga mempermudah analisis pemilihan metode yang tepat.

2. Meningkatkan keakuratan model proses bisnis dengan menilai kinerja metode *heuristic miner* dan *inductive miner* dalam konteks kompleksitas model.
3. Menyediakan referensi bagi penelitian lanjutan di bidang *process mining* dan pengembangan model proses bisnis yang lebih efisien.

BAB II

STUDI PUSTAKA

2.1 Penelitian Terkait

Dalam upaya memahami efektivitas algoritma dalam *process mining*, berbagai penelitian terdahulu telah membandingkan kinerja algoritma, termasuk *heuristic miner* dan *inductive miner*. Penelitian-penelitian ini memberikan gambaran tentang keunggulan dan keterbatasan masing-masing algoritma dalam merepresentasikan proses dan alur kerja dalam suatu sistem. Penelitian oleh (Hasyim et al., 2020) yang berjudul "Analisis Perbandingan Metode *Alpha Miner, Inductive Miner* dan *Causal-Net Mining* dalam *Proses Mining"* membahas penggunaan algoritma *inductive miner* dalam *process mining* untuk 9 skenario model proses bisnis. *inductive miner* menunjukkan tingkat kesamaan (*similarity*) tertinggi sebesar 60% pada salah satu model, namun secara keseluruhan, hasil studi menunjukkan bahwa *inductive miner* memiliki keterbatasan dalam merepresentasikan model proses secara akurat. Hal ini menunjukkan bahwa algoritma *inductive miner* mungkin tidak selalu efektif untuk semua jenis skenario proses bisnis.

Penelitian yang dilakukan oleh (Bakhshi et al., 2023) membandingkan dua metode *process mining*, yaitu *heuristics miner* dan *inductive miner*, dalam memodelkan jalur perawatan pasien sepsis. Hasil perbandingan menunjukkan bahwa kedua metode memiliki keterbatasan dalam representasi proses yang akurat. *heuristics miner* menghasilkan model dengan tingkat *fitness* sebesar 77.4% dan kesederhanaan (*simplicity*) 47%, menunjukkan bahwa model ini memiliki banyak

hubungan antara aktivitas yang kompleks dan sulit dipahami. Di sisi lain, *inductive miner* memberikan *fitness* yang lebih tinggi, yaitu 84.8%, dan kesederhanaan sebesar 62.2%, membuat model ini sedikit lebih mudah dipahami. Meskipun demikian, *inductive miner* masih gagal merepresentasikan struktur proses secara jelas dan akurat. Hasil studi ini menyimpulkan bahwa baik *heuristic miner* maupun *inductive miner* tidak cukup efektif untuk menggambarkan proses perawatan sepsis secara lengkap. Oleh karena itu, peneliti mengembangkan model proses berbasis pengetahuan ahli sebagai alternatif yang lebih efektif dalam merepresentasikan jalur perawatan pasien.

Penelitian oleh (Sahlabadi et al., 2023) membandingkan lima algoritma *process mining* yaitu: *Flower, alpha, ILP (Integer Linear Programming), heuristic,* dan *inductive* untuk evaluasi arsitektur perangkat lunak. Perbandingan dilakukan berdasarkan kriteria kualitas seperti *fitness*, *precision*, *generalization*, dan *simplicity*. Hasil penelitian menunjukkan bahwa algoritma *heuristic miner* memiliki performa terbaik karena mampu menghasilkan model dengan nilai *fitness* tinggi (0.6 - 0.812) dan kesederhanaan yang baik (jumlah *arc* hanya sekitar 30-50). Selain itu, *heuristic miner* dapat menangani *noise* dan menjaga keseimbangan antara ketepatan (*precision*) dan generalisasi (*generalization*). Sebaliknya, algoritma lain seperti *alpha* dan ILP menghasilkan model yang sangat kompleks (sekitar 3000 *arc*), membuatnya sulit untuk dianalisis. Dengan demikian, *heuristic miner* diidentifikasi sebagai metode paling efektif untuk evaluasi arsitektur perangkat lunak.

Penelitian oleh (Nuritha & Mahendrawathi, 2017) yang berjudul "*Structural Similarity Measurement of Business Process Model to Compare Heuristic and Inductive Miner Algorithms Performance in Dealing with Noise*" membahas penggunaan algoritma *inductive miner* dan *heuristic miner* dalam *process mining* untuk memodelkan proses bisnis yang mengandung *noise* pada *event logs*. Hasil penelitian menunjukkan bahwa *inductive miner* memiliki kinerja yang lebih baik dalam menghadapi *noise*, dengan nilai *structural similarity* yang lebih tinggi, terutama pada dua studi kasus industri: proses pengadaan bahan baku pada industri semen dan proses produksi di industri farmasi**.** *Inductive miner* menghasilkan nilai *similarity* 1 pada sebagian besar model, sementara *Heuristic miner* menghasilkan nilai *similarity* yang lebih bervariasi, dengan nilai tertinggi mencapai 0,75 dan terendah pada beberapa model mencapai 0,25. Meskipun demikian, *heuristic miner* juga menunjukkan performa yang cukup baik pada alur proses yang melibatkan percabangan. Secara keseluruhan, penelitian ini menyarankan bahwa *inductive miner* lebih efektif untuk menangani *event logs* dengan *noise*, namun *heuristic miner* lebih baik pada proses dengan percabangan kompleks. Secara keseluruhan, *inductive miner* memiliki rata-rata *similarity* yang lebih tinggi (0,736 untuk industri semen dan 1 untuk industri farmasi) dibandingkan dengan *heuristic miner* (0,325 untuk industri semen dan 0,390625 untuk industri farmasi).

Tabel 2. 1 Penelitian Terkait

Judul	Input	Metode	Output	Perbedaan
Analisis Perbandingan Metode *Alpha Miner, Inductive Miner* dan *Causal-Net Mining dalam Proses Mining*	Dataset yang terdiri dari beberapa model proses bisnis	*Alpha Miner*, *Inductive Miner*, dan *Casual-Net*	Hasil penelitian menunjukkan algoritma *alpha miner* memiliki tingkat *similarity* tertinggi 0.89 dan terendah 0.12, serta	Tidak melibatkan kompleksitas

Judul	Input	Metode	Output	Perbedaan
(Hasyim et al., 2020)			0.75 pada model lain. *inductive miner* paling cocok dengan nilai *similarity* 0.6 dan 0.35. Tingkat similarity untuk *Causal Net* (C-Net) tidak disebutkan.	
Optimizing Sepsis Care through Heuristics Methods in Process Mining: A Trajectory Analysis (Bakhshi et al., 2023)	Data perawatan pasien sepsis	*Inductive miner* dan *heuristic miner*	*Heuristics miner Simplicity*: 47% *Inductive miner Simplicity*: 62.2%	Tidak melibatkan kompleksitas
Process Mining Discovery Techniques for Software Architecture Lightweight Evaluation Framework (Sahlabadi et al., 2023)	*Event logs* diperoleh dari simulasi yang mencatat urutan aktivitas dalam proses bisnis atau arsitektur perangkat lunak.	*Flower model, alpha algorithm, Integer Linear Programming* (ILP), *heuristic miner inductive miner*	*Heuristic Miner:Fitness*: 0.6 - 0.812, *Arc*: 30-50 *Inductive miner*: *Fitness*: 0.5 - 0.6, *Arc*: 100-200 *Flower model*: *Fitness*: 1.0, *Arc*: ~3000 *Alpha algorithm*: *Fitness*: 0.4 - 0.5, *Arc*: ~3000 ILP: *Fitness*: 0.3 - 0.4, *Arc:* ~3000	Tidak melibatkan kompleksitas
Structural Similarity Measurement of Business Process Model to Compare Heuristic and Inductive Miner Algorithms Performance in Dealing with Noise (Nuritha & Mahendrawathi, 2017)	Dataset proses pengadaan bahan baku pada industri semen dan proses produksi di industri farmasi	*Inductive miner* dan *heuristic miner*	*Inductive miner* memiliki rata-rata *similarity* yang lebih tinggi (0,736 untuk industri semen dan 1 untuk industri farmasi) h*euristic miner* (0,325 untuk industri semen dan 0,390625 untuk industri farmasi).	Tidak melibatkan kompleksitas
Analisis Perbandingan Algoritma *Heuristic* dan *Inductive* dalam *Process Mining* (Zakin)	Data adalah *event logs* yang telah disimulasikan, yang mencakup berbagai skenario proses bisnis, mulai	*Inductive miner* dan h*euristic miner*	Hasil perhirungan kompleksitas dan *similarity*	Melibatkan kompleksitas

Judul	Input	Metode	Output	Perbedaan
	dari proses yang sederhana hingga sangat kompleks.			

Keterbaruan dalam penelitian ini terletak pada pelibatan kompleksitas model proses bisnis dalam *process mining* dan penelitian sebelumnya tidak melibatkan kompleksitas.

2.2 *Process Mining*

Process mining adalah satu bidang ilmu baru yang berkembang dari berbagai bidang ilmu lain yang terkait. Dari sisi metodologi, *Process mining* berkaitan erat dengan *data mining*, dengan pembeda utama pada orientasinya ke proses bisnis. Selain itu, *process mining* sering dikaitkan dengan pemodelan proses bisnis, namun *process mining* menghasilkan model proses secara otomatis dari data-data proses bisnis yang tersimpan. *Process mining* juga terkait dengan sains data, karena merupakan metode untuk menganalisis data dan menjawab pertanyaan-pertanyaan bisnis yang berkaitan dengan data (Kurniati et al., 2023). Dalam pengembangannya, *process mining* telah dapat digunakan dengan berbagai macam teknik atau algoritma dalam proses mencari informasi atau proses *discovery*. Yang menjadi ciri dari suatu *process mining* adalah dapat divisualisasikannya bisnis proses yang diidentifikasi tersebut schingga dapat dengan mudah untuk mengekstrak informasi yang ada di dalamnya (Rahmawati, 2016). Ada tiga macam tipe dari *process mining*, yaitu: *discovery*, *conformance*, dan *enhancement* (Vossen, 2012):

1. *Discovery (*Penemuan)

   Discovery adalah proses model proses dibuat secara otomatis dari data *log* kejadian (*event logs*). Model ini dibuat tanpa menggunakan pengetahuan atau model sebelumnya, sehingga hanya berdasarkan data yang ada. Dengan *discovery*, kita bisa mendapatkan wawasan tentang bagaimana proses sebenarnya berjalan dalam organisasi. Teknik seperti *alpha algorithm* dan *heuristic miner* sering digunakan dalam tahapan ini untuk membangun model dari *log* kejadian.

2. *Conformance Checking* (Pemeriksaan Kesesuaian)

   Conformance checking bertujuan untuk membandingkan model proses yang telah ditemukan atau model yang sudah ada sebelumnya dengan data *log* kejadian aktual untuk mengetahui apakah proses berjalan sesuai dengan yang diharapkan. Kegiatan ini berguna untuk mengidentifikasi deviasi atau ketidaksesuaian antara proses yang diinginkan dan proses yang sebenarnya terjadi, membantu organisasi untuk memahami apakah ada pelanggaran kebijakan atau prosedur yang perlu diperbaiki proses nyata.

3. *Enhancement* (Peningkatan)

   Pada tahap *enhancement*, model proses yang ada diperbaiki atau ditingkatkan berdasarkan data dari *log* kejadian. Tujuan dari kegiatan ini adalah untuk meningkatkan kinerja dan efisiensi proses dengan menambahkan informasi baru, seperti waktu proses atau titik kemacetan

yang ditemukan selama analisis. Dengan melakukan *enhancement*, organisasi bisa memperbaiki proses bisnis mereka berdasarkan data faktual.

2.3 *Event Logs*

Event logs adalah catatan yang berisi informasi tentang aktivitas atau kejadian yang terjadi dalam suatu sistem, aplikasi perangkat lunak, atau jaringan komputer. Setiap *event* (kejadian) yang terjadi di dalam sistem dicatat dalam *log*, yang berfungsi untuk melacak berbagai operasi, seperti aktivitas pengguna, kesalahan sistem, kegagalan, atau peristiwa lain yang relevan. *Event logs* biasanya digunakan untuk keperluan seperti *debugging*, pemantauan operasional, analisis kinerja, audit keamanan, dan deteksi ancaman keamanan.

Event logs sangat penting dalam administrasi sistem, karena dengan menganalisis *log* ini, administrator dapat memahami status sistem, mendiagnosis masalah, dan memastikan sistem berjalan dengan benar. *Event logs* juga berguna membantu mendeteksi aktivitas yang tidak biasa atau mencurigakan, sehingga memungkinkan respons cepat terhadap ancaman keamanan (Hutahean et al., 2022).

Tabel 2. 2 Contoh *Event Logs*

Case ID	***Activity***	***Timestamp***
4	A	2024-09-10 10:07:00
4	B	2024-09-10 10:08:00
4	D	2024-09-10 10:09:00
4	C	2024-09-10 10:10:00
5	A	2024-09-10 10:11:00
5	D	2024-09-10 10:12:00
5	B	2024-09-10 10:13:00
5	C	2024-09-10 10:14:00

2.4 Proses Bisnis

Proses bisnis merupakan serangkaian aktivitas yang saling terkait dan terstruktur dalam suatu organisasi untuk mencapai tujuan tertentu. Proses bisnis terdiri dari serangkaian aktivitas atau tugas yang dilakukan dalam urutan tertentu dengan menggunakan sumber daya organisasi untuk mencapai tujuan atau misi bisnisnya. (Vincek, 2018). Proses bisnis terdiri dari aktivitas terstruktur yang menciptakan nilai dalam sebuah organisasi (Blattmeier, 2023).

Menurut *Business Process Management* oleh (Weske, 2012), tujuan utama proses bisnis adalah untuk meningkatkan pemahaman dan analisis terhadap operasi perusahaan serta hubungan antar aktivitasnya. Proses bisnis juga bertujuan untuk meningkatkan fleksibilitas agar dapat beradaptasi dengan perubahan organisasi dan teknologi tanpa mengubah keseluruhan sistem. Selain itu, manajemen proses bisnis mendukung peningkatan berkelanjutan dengan mengidentifikasi dan memperbaiki area yang kurang efisien. Proses bisnis juga berfungsi sebagai repositori pengetahuan perusahaan, mendokumentasikan bagaimana operasi berjalan, serta menjembatani kesenjangan antara strategi bisnis dan teknologi informasi agar implementasi sistem lebih efisien.

Penggambaran proses bisnis dapat dilakukan dengan *menggunakan Business Process Model and Notation* (BPMN). BPMN adalah standar yang digunakan secara luas untuk mendefinisikan alur kerja dan proses bisnis (Kräuter et al., 2023). BPMN berfungsi sebagai bahasa umum bagi para pemangku kepentingan dalam pengembangan perangkat lunak, menjembatani kesenjangan antara analis bisnis, pelanggan, dan pemrogram(Lam et al., 2020). BPMN

merupakan standar grafis yang digunakan untuk memvisualisasikan proses bisnis dengan elemen-elemen seperti *events* (peristiwa), *activities* (aktivitas), *gateways* (pengambilan keputusan), dan *flows* (alur proses) (Sumarsono et al., 2023).

Tujuan dari BPMN adalah untuk mendukung pemodelan proses bisnis untuk pengguna teknis dan pengguna bisnis, dengan menyediakan notasi yang intuitif bagi pengguna bisnis, namun mampu merepresentasikan semantik proses yang kompleks (Von Rosing et al., 2015). BPMN dirancang agar mudah dimengerti oleh semua pemangku kepentingan bisnis. Ini termasuk analis bisnis yang membuat dan menyempurnakan proses, pengembang teknis yang bertanggung jawab untuk mengimplementasikannya, dan manajer bisnis yang memantau dan mengelolanya. Oleh karena itu, BPMN berfungsi sebagai bahasa yang umum, menjembatani kesenjangan komunikasi yang sering terjadi antara proses bisnis desain dan implementasi.

BPMN membantu organisasi dalam mendokumentasikan, menganalisis, serta mengoptimalkan alur kerja secara lebih sistematis dan mudah dipahami oleh berbagai pemangku kepentingan (Baihaki & Djamaluddin, 2022). Dengan BPMN, proses bisnis dapat divisualisasikan dalam bentuk diagram yang menunjukkan urutan aktivitas, aliran data, serta interaksi antara berbagai departemen atau sistem dalam organisasi (Tampubolon & Situmorang, 2023). Hal ini memungkinkan perusahaan untuk mengidentifikasi inefisiensi, mengoptimalkan sumber daya, serta meningkatkan efektivitas operasionalnya.

2.5 *Heuristic Miner*

Heuristic miner adalah teknik penemuan proses yang fokus pada pembuatan model proses dengan mempertimbangkan frekuensi kejadian dan urutannya. Ide dasar dari *heuristic miner* adalah mengabaikan jalur yang jarang terjadi dan fokus pada perilaku yang paling umum dalam suatu proses. Pendekatan ini membuat *heuristic miner* lebih kuat dibandingkan dengan metode lainnya, karena menyaring *noise* dan hanya berfokus pada aliran utama dari proses.

Langkah-langkah dalam *heuristic miner* :

1. Membangun *Directly-Follows Graph*: Identifikasi hubungan "langsung mengikuti" dalam *log* kejadian, artinya untuk setiap pasangan aktivitas ketika satu aktivitas secara langsung mengikuti aktivitas lain, hubungan tersebut dicatat.
2. Menghitung Ukuran Ketergantungan: Untuk setiap pasangan aktivitas a dan b yang secara langsung diikuti, hitung ukuran ketergantungannya:

   Ukuran ketergantungan |a⇒Lb| antara aktivitas a dan b didefinisikan sebagai:

   $$|a \Rightarrow Lb| = \frac{|a>Lb| - |b>La|}{|a>Lb| + |b>La| + 1} \text{ jika, } a \neq b \quad (2.1)$$

   |a>Lb| adalah jumlah kejadian ketika a secara langsung diikuti oleh b dalam *log*.

   Untuk *self-loop* (misalnya, a diikuti oleh a), rumusnya adalah:

   $$|a \Rightarrow La| = \frac{|a>La|}{|a>La| + 1} \quad (2.2)$$

Nilai ketergantungan ini berkisar antara -1 hingga 1. Nilai yang mendekati 1 menunjukkan ketergantungan positif yang kuat, artinya a sering mengarah ke b dan sebaliknya .

3. Membangun *Dependency Graph*: Membuat grafik node mewakili aktivitas dan busur mewakili ketergantungan yang dihitung pada langkah sebelumnya. Hanya busur yang memenuhi ambang batas tertentu untuk fokus pada jalur yang paling signifikan dan mengabaikan *noise*.
4. Memahami *Splits* dan *Joins*: Tujuannya adalah untuk mengekstrak *Causal Net* (C-net) dari *log* kejadian. *Dependency graph* membantu mengidentifikasi aktivitas awal dan akhir serta struktur inti dari proses. Kemudian, input dan *output bindings* untuk setiap aktivitas ditentukan untuk melengkapi *C-net*.
5. Visualisasi dan Interpretasi

   Setelah model proses dibuat, model tersebut dapat divisualisasikan, biasanya dengan ketebalan busur yang menggambarkan frekuensi jalur. Ini membantu menyoroti aliran utama dalam model. Pendekatan ini fleksibel dan dapat disesuaikan dengan berbagai representasi, seperti *C-nets*, yang menyediakan cara intuitif dan ekspresif untuk memodelkan perilaku kompleks dalam proses.

2.6 *Inductive Miner*

Inductive Miner adalah sebuah algoritma penemuan proses yang digunakan dalam *process mining*. Tujuan utamanya adalah untuk mengekstraksi model proses dari *event logs*. *Inductive miner* membangun pohon proses (*process tree*) yang

selalu konsisten (*sound*), menghindari masalah seperti *deadlock* dan *livelock*. Algoritma ini menggunakan pendekatan "*divide-and-conquer*" (bagi dan taklukkan) untuk membagi *log* kejadian menjadi *sublog* yang lebih kecil dan mengidentifikasi pola seperti urutan (*sequences*), pilihan (*choices*), paralelisme (*parallelisms*), dan *loop*. Pendekatan rekursif ini memungkinkan penemuan model proses terstruktur yang dapat menangani berbagai pola perilaku dalam proses nyata. Langkah-Langkah Dasar *Inductive Miner*:

1. Mulai dengan seluruh *event log*s: Bangun graf "*directly-follows*" untuk *log* kejadian, yang menangkap hubungan langsung antara aktivitas.
2. Identifikasi *Cut*: Tentukan jenis *cut* (misalnya, urutan, paralel, pilihan, atau *loop*) yang terbaik untuk membagi aktivitas dalam *log*. Langkah ini melibatkan identifikasi bagaimana aktivitas berhubungan satu sama lain.
3. Bagi *Log*: Berdasarkan *cut* yang telah diidentifikasi, bagi *log* kejadian menjadi *sublog*. Setiap *sublog* akan menjadi cabang dalam pohon proses.
4. Penerapan Rekursif: Terapkan *inductive miner* secara rekursif pada setiap *sublog* untuk menghasilkan model subproses sampai mencapai kasus dasar ketika *sublog* hanya berisi satu aktivitas.
5. Gabungkan Submodel: Bangun pohon proses dengan menggabungkan model subproses menggunakan *cut* yang telah diidentifikasi.

2.7 *Similarity*

Similarity measurement atau pengukuran kesamaan proses bisnis mengacu pada metode untuk menghitung kesamaan antara dua atau lebih model proses bisnis. Proses ini penting dalam konteks manajemen proses bisnis untuk memahami

kemiripan dan perbedaan di antara proses, yang membantu dalam optimalisasi dan pengembangan lebih lanjut dari proses tersebut (Zeng et al., 2020). Terdapat tiga metode utama dalam pengukuran kesamaan proses bisnis, yaitu *structural similarity*, *behavioral similarity* , dan *semantic similarity*. Ketiga metode ini berfokus pada aspek yang berbeda dari model proses bisnis dan menggunakan pendekatan yang spesifik dalam menghitung kesamaan.

Structural similarity dalam *process mining* mengacu pada tingkat kemiripan antara model-model proses berdasarkan elemen-elemen struktural (Kang et al., 2008). *Structural similarity* digunakan untuk membandingkan bagaimana dua model proses diatur secara topologis, tanpa mempertimbangkan perilaku eksekusi atau makna semantis dari aktivitas dalam model tersebut. Menurut Remco Djikman (Dijkman et al., 2009) untuk menghitung kesamaan dua proses, harus ditemukan pemetaan yang menginduksi kesamaan maksimal. Untuk menghitung nilai *structural similarity* dengan menggunakan persamaan *jaccard coefficient similarity*:

$$Sim\ (A, B) = \frac{\mathrm{A} \cap \mathrm{B}}{\mathrm{A} \cup \mathrm{B}} \tag{2.3}$$

Keterangan:
A : model proses 1
A $\cap B$: jumlah yang sama
B : model proses 2
$A \cup B$: jumlah yang sama + jumlah yang berbeda

Behavioral similarity didapatkan dengan menghitung jarak kedua proses bisnis dalam ruang vektor yang dibangun dari jejak kaki kausal (*causal footprint*) kedua proses bisnis tersebut. Dari jejak-jejak *causal footprint* yang terbentuk akan membentuk *node-node* yang akan saling terhubung. *Node-node* tersebut terbentuk

dari *task* yang terdapat pada model proses yang sesuai dengan alur yang berjalan. Dari hasil penentuan *causal footprint* tersebut dapat dihitung nilai kemiripannya menggunakan persamaan *jaccard coefficient similarity* seperti pada perhitungan *structural similarity*.

Semantic similarity adalah tugas untuk mengukur tingkat kemiripan atau kesamaan antara kalimat, kata, atau teks berdasarkan maknanya. *Semantic similarity* berfokus pada memahami dan membandingkan makna yang terkandung dalam teks (Alian & Awajan, 2020). Perhitungan kemiripan semantik dilakukan dengan mempertimbangkan kemiripan label atau *string*, serta kemiripan kontekstual yang merujuk pada kesamaan makna antar kalimat (Abriani, 2020). Untuk kemiripan kontekstual, digunakan algoritma Wu Palmer untuk menghitung kemiripan antar kata dalam kalimat. Setelah itu, nilai kemiripan antar kalimat ditentukan dengan mengakumulasikan matriks kemiripan kata, yang dibobot berdasarkan nilai kelas kata menggunakan metode AHP. Kriteria yang digunakan adalah *noun* (kata benda) dan *verb* (kata kerja), karena kedua kelas kata ini dihitung nilai kemiripannya menggunakan WS4J. Persamaan untuk menghitung kemiripan kalimat berdasarkan konteksnya adalah:

$$Sentence\ similarity = (nmax \times bn) + (vmax + bn) \quad (2.4)$$

Dalam persamaan ini, $nmax$ adalah rata-rata nilai dari kriteria *noun*, sedangkan $vmax$ adalah rata-rata nilai dari kriteria *verb*. Bobot nilai untuk *noun* (bn) dan *verb* (bv) ditentukan melalui AHP. Hasil perhitungan AHP memberikan bobot sebesar 0.25 untuk *noun* dan 0.75 untuk *verb*.

2.8 Kompleksitas

Kompleksitas dalam konteks BPMN mengacu pada kerumitan struktural dari model proses bisnis, yang dapat berdampak pada pemahaman dan pemeliharaan (Rolón et al., 2009). Kompleksitas model proses bisnis diukur melalui berbagai metrik yang disusun berdasarkan struktur *Basic Control Structure* (BCS). Untuk menghitung kompleksitas, terdapat beberapa langkah dan komponen yang digunakan, salah satunya adalah rumus *Yaqin complexity formula* yang dirancang untuk lebih sensitif terhadap perubahan kecil pada struktur proses bisnis.

Untuk menghitung kompleksitas menggunakan *Yaqin complexity formula* yang mencakup beberapa parameter seperti jumlah aktivitas, percabangan (AND, OR, XOR), *loop*, dan kedalaman, kita akan menggunakan definisi dari *Yaqin complexity formula* sebagai berikut:

1. *Nodes* (Ns): Jumlah aktivitas (*nodes*) dalam proses.
2. *Arcs* (As): Jumlah penghubung (*arcs*) antar *node* dalam proses.
3. AND *Branching Complexity* (CAND): Mengukur kompleksitas dari percabangan AND berdasarkan jumlah percabangan.
4. XOR *Branching Complexity* (CXOR): Mengukur kompleksitas dari percabangan XOR.
5. OR *Branching Complexity* (COR): Mengukur kompleksitas dari percabangan OR.
6. *Cyclic Complexity* ($Ccyc$): Mengukur kompleksitas yang dihasilkan oleh *loop* atau siklus dalam proses.
7. *Depth Complexity* (D): Mengukur kedalaman rata-rata dari setiap aktivitas.

Yaqin complexity formula ini menghitung kompleksitas berdasarkan beberapa komponen utama, yaitu jumlah aktivitas (Ns), jumlah *arc* (As), kompleksitas percabangan AND (CAND), XOR (CXOR), OR (COR), kompleksitas siklus (*Cyclic complexity*), dan kompleksitas kedalaman (*Depth complexity*). Rumus *Yaqin complexity formula* terdapat pada persamaan 2.5.

$$YC = Ns + As + CAND + CXOR + COR + CCYC + CD \quad (2.5)$$

Dalam rumus tersebut, Ns adalah jumlah simpul yang merupakan gabungan dari awal, akhir, aktivitas, dan percabangan; As adalah jumlah *arc*; CAND, CXOR, dan COR adalah kompleksitas dari percabangan logika AND, XOR, dan OR; *Cyclic complexity* menghitung kompleksitas dari *loop* dalam proses bisnis, dan *Depth complexity* mengukur kedalaman rata-rata dari tiap aktivitas dalam proses.

Untuk mengukur kompleksitas dalam *Yaqin complexity formula*, setiap komponen dalam *Basic Control Structure* (BCS) diberi nilai *cognitive weight* (CW), yang mengindikasikan tingkat kesulitan atau beban kognitif yang diperlukan untuk memahami elemen-elemen dalam model proses bisnis. Bobot ini berdasarkan pada teori kognitif yang mengukur kompleksitas dalam hal pemahaman manusia terhadap struktur alur kerja. Berikut adalah nilai *cognitive weight* yang digunakan untuk masing-masing elemen dalam BCS:

Tabel 2. 3 Nilai *Cognitive Weight*

Workflow Pattern	**Weight**
Sequence	1
XOR	3
AND	4
OR	7
Cyclic	3
Depth	14

Setiap elemen dalam model BPMN ini memiliki pengaruh langsung terhadap kompleksitas keseluruhan, dan bobot kognitif ini digunakan untuk menghitung dampak relatif dari masing-masing elemen terhadap pemahaman dan kesulitan dalam model tersebut. Perhitungan kompleksitas berdasarkan *cognitive weight* memungkinkan kita untuk menilai seberapa rumit atau mudahnya suatu model untuk dipahami, yang sangat penting untuk pengelolaan dan pemeliharaan proses bisnis secara efisien.

Perhitungan untuk setiap komponen kompleksitas dalam rumus *Yaqin Complexity Formula*, yaitu C_{AND}, C_{XOR}, C_{OR}, C_{CYC}, dan CD. Berikut adalah penjelasan detail mengenai cara menghitung masing-masing komponen kompleksitas tersebut:

2.8.1 Kompleksitas Percabangan AND (C_{AND})

Percabangan AND mengharuskan semua cabang untuk dieksekusi secara paralel, dan kompleksitasnya tergantung pada jumlah cabang yang terlibat. Perhitungan kompleksitas AND dilakukan berdasarkan jumlah jalur yang mungkin untuk setiap percabangan AND. Rumus perhitungan kompleksitas percabangan AND terdapat pada persamaan 2.6.

$$C_{AND} = W_{AND} . \sum (n!)_i \tag{2.6}$$

Keterangan:
W_{AND} : *cognitive weight* untuk percabangan AND (dalam hal ini, $W_{AND} = 4$)
n : jumlah cabang dalam percabangan AND yang terlibat dalam model proses,
i : indeks untuk masing-masing percabangan AND dalam model tersebut.

2.8.2 Kompleksitas Percabangan XOR (C_{XOR})

Percabangan XOR memungkinkan hanya satu jalur yang dipilih untuk dieksekusi. Oleh karena itu, kompleksitas percabangan XOR dihitung berdasarkan jumlah cabang yang ada dan jalur yang bisa dipilih. Rumus perhitungan kompleksitas percabangan XOR terdapat pada persamaan 2.7.

$$C_{XOR} = W_{XOR} . \sum (n)_i \tag{2.7}$$

Keterangan:
W_{XOR} : *cognitive weight* untuk percabangan XOR (dalam hal ini, W_{XOR}=3
n : jumlah cabang dalam percabangan XOR yang terlibat dalam model,
i : indeks untuk percabangan XOR tersebut.

2.8.3 Kompleksitas Percabangan OR (C_{OR})

Percabangan OR memungkinkan lebih dari satu jalur untuk dipilih, dan kompleksitasnya bergantung pada jumlah cabang dan kemungkinan kombinasi jalur yang dapat dipilih. Rumus perhitungan kompleksitas percabangan OR terdapat pada persamaan 2.8.

$$C_{OR} = W_{OR} . \sum (\frac{n!}{(n-k)!})_i \tag{2.8}$$

Keterangan:
W_{OR} : *cognitive weight* untuk percabangan OR (dalam hal ini, W_{OR}=7)
n : jumlah cabang yang terlibat dalam percabangan OR,
k : jumlah cabang yang dapat dipilih pada percabangan tersebut.

2.8.4 Kompleksitas Siklus (*Cyclic Complexity*, C_{CYC})

Cyclic Complexity mengukur kompleksitas yang dihasilkan oleh *loop* atau siklus dalam proses. Perhitungan *Cyclic Complexity* dihitung berdasarkan jumlah aktivitas yang terlibat dalam *loop*, serta diameter dari siklus tersebut. Rumus perhitungan kompleksitas siklus terdapat pada persamaan 2.9.

$$C_{cyc} = W_{cyc}.\frac{Acs_{cyc}}{D_m} \quad (2.9)$$

Keterangan:
W_{CYC} : *cognitive weight* untuk siklus (CYC), bobot yang diberikan bernilai 3,
A_{CYC} : jumlah aktivitas yang terlibat dalam siklus atau *loop*,
D_m : diameter siklus, yaitu panjang jalur yang dibentuk oleh *loop*.

2.8.5 Kompleksitas Kedalaman (*Depth Complexity*, CD)

Depth Complexity mengukur kedalaman rata-rata dari setiap aktivitas dalam model proses bisnis. Kedalaman mengacu pada berapa banyak langkah yang perlu dilalui untuk mencapai suatu aktivitas dari titik awal proses. Perhitungan *Depth Complexity* dihitung berdasarkan kedalaman masing-masing aktivitas menggunakan rumus dalam persamaan 2.11 sebelum itu kita perlu menghitung *depth average* menggunakan persaqamaan 2.10.

$$D_{avg} = \frac{\sum D_j}{A_{cs}} \quad (2.10)$$

Keterangan:
D_j : kedalaman dari aktivitas ke-j,
A_{cs} : jumlah total aktivitas dalam proses.

Setelah kedalaman rata-rata dihitung, *Depth Complexity* dihitung sebagai:

$$C_D = W_D.D_{avg} \quad (2.11)$$

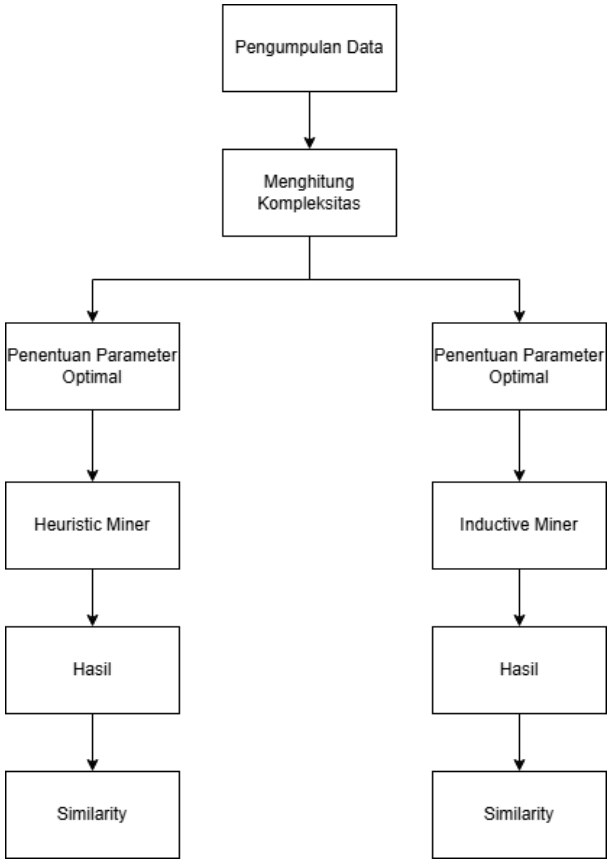
Keterangan:
W_D : *cognitive weight* untuk kedalaman, diberi bobot 14.
D_{avg} : kedalaman rata rata.

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Prosedur Penelitian

Untuk mengatasi masalah yang telah dibahas sebelumnya prosedur penelitian ini meliputi alur dari input data hingga output yang sesuai dengan tujuan penelitian. Berikut adalah prosedur penelitian yang akan diterapkan dalam penelitian ini yang disajikan pada Gambar 3.1:



Gambar 3. 1 Prosedur Penelitian

3.1.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini diperoleh dari simulasi yang dilakukan berdasarkan model proses bisnis yang dijelaskan dalam paper "*Measuring Scalable Business Process Model Complexity Based on Basic Control Structure "* oleh (Yaqin et al., 2020) . Data ini menggambarkan aktivitas-aktivitas yang terjadi dalam suatu proses bisnis. Aktivitas-aktivitas tersebut dicatat dan disusun dalam format file.csv. Data yang terkandung dalam *file* ini terdiri dari beberapa kolom, yaitu:

1. Case_ID : Berisi ID dari kasus.
2. Activity : Menunjukkan nama aktivitas.
3. Timestamp : Menampilkan tanggal dan waktu saat aktivitas tersebut dilakukan.

3.1.2 Pengukuran Kompleksitas

Untuk menghitung kompleksitas menggunakan *Yaqin complexity formula* yang mencakup beberapa parameter seperti jumlah aktivitas, percabangan (AND, OR, XOR), *loop*, dan kedalaman sesuai persamaan 2.5. Sekarang kita akan menghitung kompleksitas model proses bisnis Gambar 4.2b, diketahui:

Nodes (Ns): 9

Arcs (As): 10 (karena ada 10 penghubung antara node)

AND *Branch Complexity* ($CAND$): 2 AND gateways dengan 3 percabangan, sesuai persamaan 2.6 maka:

$$C_{AND} = 4.(3!) = 24$$

XOR *Branch Complexity* ($CXOR$), dihitung berdasarkan persamaan 2.7. Karena Gambar 4.2b tidak memiliki XOR maka:

$$C_{XOR} = 3.0 = 0$$

OR *Branch Complexity* (COR) dihitung berdasarkan persamaan 2.8. Karena Gambar 4.2b tidak memiliki OR maka:

$$C_{OR} = 7.0 = 0$$

Cyclic Complexity ($Ccyc$) dihitung berdasarkan persamaan 2.9. Karena Gambar 4.2b tidak memiliki *loop* maka.

$$C_{cyc} = 3.\frac{0}{7} = 0$$

Depth Complexity (C*D*) dihitung berdasarkan persamaan 2.10, D_j adalah kedalaman masing masing aktivitas dan Acs adalah aktivitas, Gambar 4.2b memiliki 5 aktivitas dengan kedalaman masing-masing aktivitas seperti berikut ini:

A	=	1	C	=	1	E	=	2
B	=	2	D	=	2			

$$D_{avg} = \frac{8}{5} = 1{,}6$$

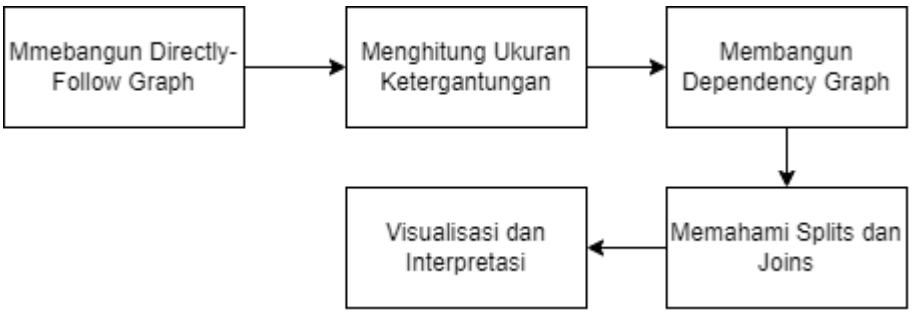
Setelah didapatkan hasil tersebut, lanjut perhitungan berdasarkan persamaan 2.11

$$C_D = 14.1{,}6 = 22{,}4$$

Setelah mendapatkan semua parameter perhitungan *yaqin complexity formula,* maka bisa dilakukan perhitungan sesuai persamaan 2.5:

$$YC = 9 + 10 + 24 + 0 + 0 + 0 + 22{,}4 = 65{,}4$$

3.1.3 *Heuristic Miner*



Gambar 3. 2 Proses Algoritma *Heuristic Miner*

Gambar 3.2 menggambarkan proses algoritma *heuristic miner* yang terdiri dari beberapa langkah. Langkah pertama adalah membangun *directly-follow graph* dari *event logs*, di sini kita akan menggunakan dataset 4.2a. 4.2a memiliki urutan aktivitas ABDC dan ADBC, berikut adalah *direct follows*-nya:

A diikuti oleh B A diikuti oleh D

B diikuti oleh D D diikuti oleh B

D diikuti oleh C B diikuti oleh C

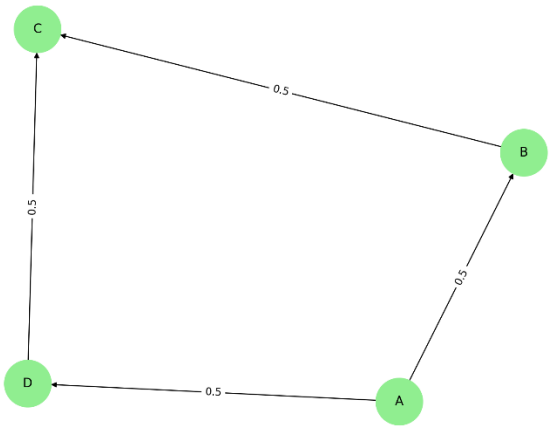
Setiap pasangan aktivitas yang diikuti langsung satu sama lain *adalah direct follows* dalam alur proses ini. Selanjutnya, ukuran ketergantungan antar aktivitas dihitung. Ukuran ini menunjukkan seberapa sering satu aktivitas diikuti oleh aktivitas lain dan seberapa kuat hubungan antar aktivitas tersebut. Hasil perhitungan ini kemudian digunakan untuk menyaring hubungan yang lemah dan menonjolkan hubungan yang lebih kuat dalam proses yang dihitung menggunakan persamaan 2.1.

$$\text{A-B} = \frac{1-0}{1+0+1} = \frac{1}{2} = 0.5 \qquad \text{B-C} = \frac{1-0}{1+0+1} = \frac{1}{2} = 0.5$$

$$\text{A-D} = \frac{1-0}{1+0+1} = \frac{1}{2} = 0.5 \qquad \text{D-B} = \frac{1-1}{1+1+1} = \frac{0}{3} = 0$$

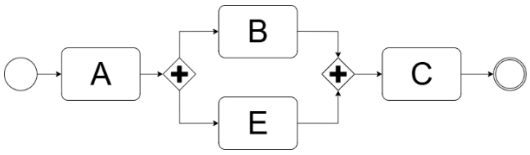
$$\text{B-D} = \frac{1-1}{1+1+1} = \frac{0}{3} = 0 \qquad \text{D-C} = \frac{1-0}{1+0+1} = \frac{1}{2} = 0.5$$

Setelah ukuran ketergantungan dihitung, langkah berikutnya adalah membangun *dependency graph*.



Gambar 3. 3 *Depedency Graph*

Gambar 3.3 ini menampilkan aliran proses yang lebih jelas dengan hanya mempertimbangkan hubungan yang memiliki ketergantungan kuat. Kemudian, graf yang telah dibangun dianalisis untuk mengidentifikasi pola *splits* (percabangan) dan *joins* (penggabungan). *Splits* terjadi ketika satu aktivitas diikuti oleh beberapa aktivitas lainnya, sementara *joins* terjadi ketika beberapa aktivitas bergabung sebelum melanjutkan ke aktivitas berikutnya. Pemahaman tentang *splits* dan *joins* ini penting untuk menggambarkan struktur proses yang kompleks.

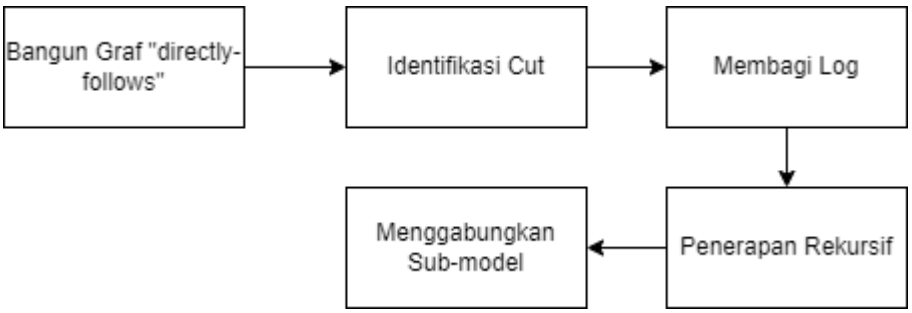


Gambar 3. 4 Proses Bisnis Hasil *Process Mining*

Langkah terakhir adalah melakukan visualisasi dan interpretasi dari graf yang telah dibuat. Visualisasi ini mempermudah pemahaman alur proses secara

keseluruhan dan memungkinkan analisis lebih lanjut terkait performa, efisiensi, dan peluang perbaikan proses yang telah diekstraksi.

3.1.4 *Inductive Miner*



Gambar 3. 5 Proses Algoritma *Inductive Miner*

Proses algoritma *heuristic miner* dalam Gambar 3.5 dimulai dengan membangun graf "*directly-follows*" yang merepresentasikan hubungan langsung antara aktivitas-aktivitas dalam *log*. Graf ini menggambarkan bagaimana aktivitas-aktivitas saling berkaitan secara berurutan dalam proses bisnis, memungkinkan identifikasi pola-pola dasar dalam aliran proses. Di sini kita akan menggunakan dataset 4.2a yang memiliki urutan aktivitas ABDC dan ADBC, berikut adalah *direct follows*-nya:

A diikuti oleh B A diikuti oleh D

B diikuti oleh D D diikuti oleh B

D diikuti oleh C B diikuti oleh C

Langkah berikutnya adalah mengidentifikasi *cut*, yaitu titik pemisahan dalam alur proses yang menunjukkan proses bercabang atau bergabung. Identifikasi *cut* ini penting untuk memahami struktur proses yang lebih kompleks, seperti percabangan (*splits*) dan penggabungan (*joins*).

Dari dua trace [A, B, D, C] dan [A, D, B, C], terlihat bahwa aktivitas A selalu di awal dan C selalu di akhir, menunjukkan pola urutan tetap. Aktivitas B dan D muncul di tengah dengan urutan yang saling bertukar, yang menunjukkan bahwa keduanya dapat dijalankan secara paralel. Karena tidak ada aktivitas yang eksklusif (XOR), struktur proses yang tepat adalah:

Sequence: A → AND(B, D) → C.

Setelah *cut* diidentifikasi, proses *log* kemudian dibagi menjadi bagian-bagian yang lebih kecil berdasarkan pola-pola tersebut, memungkinkan analisis yang lebih terperinci.

Kita buat 3 *sublog*:

Sublog 1 (untuk A): [[A], [A]]

Sublog 2 (untuk B dan D): [[B, D], [D, B]]

Sublog 3 (untuk C): [[C], [C]]

Setelah pembagian *log* dilakukan, langkah selanjutnya adalah penerapan rekursif, algoritma diterapkan kembali pada bagian-bagian *log* yang telah dipisahkan untuk menemukan sub-proses dalam setiap bagian. Proses ini dilakukan berulang kali hingga struktur proses yang lengkap dan mendetail dapat diperoleh.

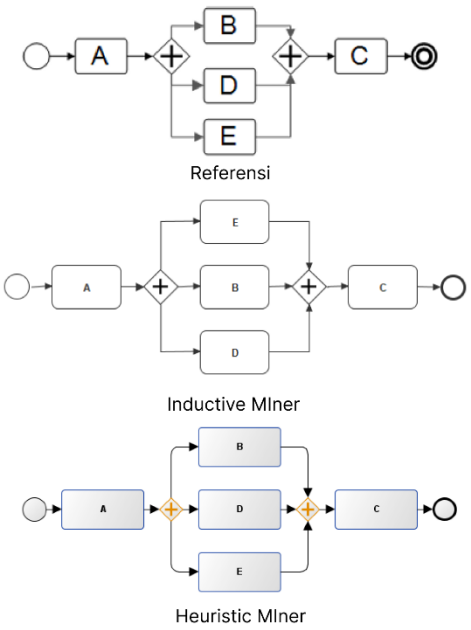
Langkah terakhir adalah menggabungkan sub-model yang telah ditemukan. Sub-model ini kemudian digabungkan menjadi satu model proses yang utuh, memberikan gambaran lengkap tentang alur dan struktur proses yang dianalisis.

→ (A, ∧(B, D), C)

Dengan melalui langkah-langkah ini, algoritma *inductive miner* mampu mengungkap model proses dari data *log* yang kompleks dan menghasilkan representasi yang lebih akurat dari proses bisnis.

3.1.5 Pengukuran Similarity

Langkah selanjutnya adalah menghitung *similarity* (kesamaan) antara model yang dihasilkan algortima dengan model referensi (ideal). Penghitungan *similarity* bertujuan untuk mengetahui seberapa mirip model proses yang dihasilkan dengan model referensi, atau untuk membandingkan proses-proses yang berbeda dalam konteks kesamaan perilaku. Salah satu metode yang dapat digunakan untuk menghitung similarity adalah *jaccard similarity*. Gambar 3.6 memperlihatkan model referensi dan dua model hasil *process mining* yang dihasilkan oleh *inductive miner* dan *heuristic miner*:



Gambar 3. 6 Model Referensi, *Inductive,* dan *Heuristics Miner*

Setelah mengidentifikasi set aktivitas dari masing-masing model (seperti yang terlihat pada gambar), langkah selanjutnya adalah menghitung *jaccard similarity* menggunakan persamaan 2.1:

Notasi BPMN terdiri dari *activity*, *arcs*, dan *gateaway* sehingga dari perbandingan model tersebut didapatkan elemen irisan sebagai berikut:

Activity : 4

Arcs : 9

Gateaway : 2

Sedangkan, gabungan dari kedua model, yaitu:

Activity : 4

Arcs : 9

Gateaway : 2

Dengan menggunakan rumus *Jaccard*:

$$Sim\ (A, B) = \frac{4 + 9 + 2}{4 + 9 + 2} = \frac{15}{15} = 1$$

Hasil ini berlaku untuk kedua model *mining*, baik *inductive miner* maupun *heuristic miner*, karena aktivitas dalam kedua model tersebut sama dengan model referensi. Jadi :

Jaccard similarity antara model referensi dan *inductive miner* = 1.0.

Jaccard similarity antara model referensi dan *heuristic miner* = 1.0.

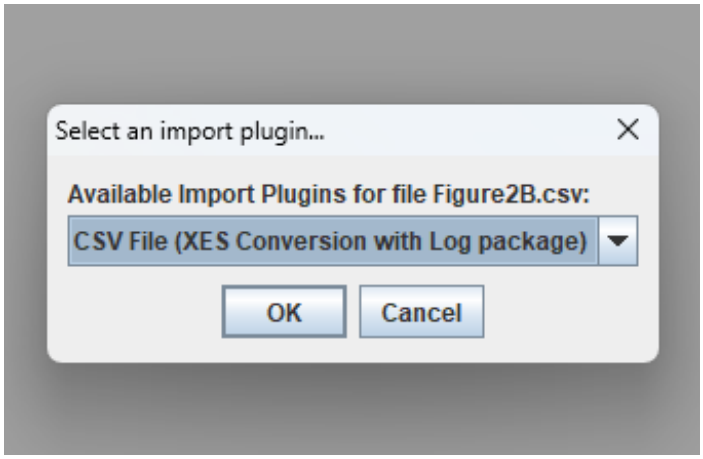
3.2 Desain Percobaan

Terdapat beberapa pengujian yang dilakukan. Berikut adalah alur dari pengujian dari penelitian ini :

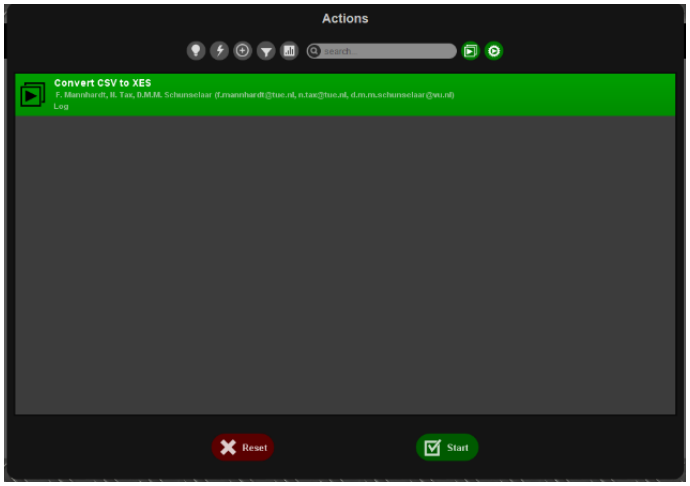
3.2.1 Input Data

Algoritma *heuristic miner* digunakan dalam penelitian ini karena algoritma *heuristic miner* memiliki keunggulan dengan mengabaikan jalur yang jarang terjadi dan fokus pada perilaku yang paling umum dalam suatu proses. Berikut merupakan contoh impentasi algoritma *heuristic miner* dalam penelitian ini menggunakan aplikasi ProM:

Langkah pertama adalah memilih *plugin* impor pada aplikasi ProM untuk file csv. *Plugin* yang dipilih adalah "CSV File (XES Conversion with Log package)" seperti pada Gambar 3.7, yang berfungsi untuk mengonversi *file* CSV menjadi format XES seperti pada Gambar 3.8. Format XES sering digunakan dalam analisis *process mining* untuk mengelola *event logs*. Langkah ini bertujuan agar data dalam *file* CSV dapat dibaca dan diolah oleh ProM dalam konteks analisis *process mining*.

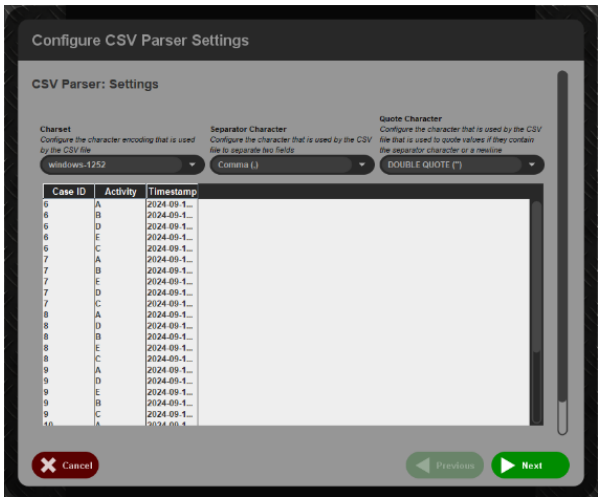


Gambar 3. 7 *Import File* CSV



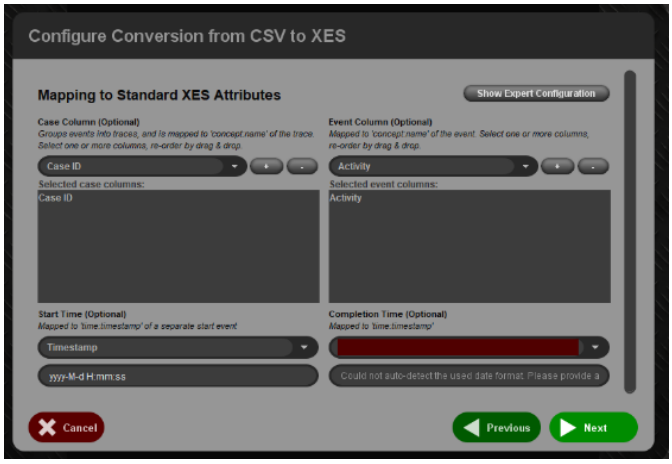
Gambar 3. 8 Menjalankan *Plugin*

Pengaturan CSV *Parser Settings* di aplikasi ProM. Pengguna memilih *encoding windows*-1252, menggunakan koma (,) sebagai pemisah kolom, dan tanda kutip ganda (") sebagai karakter kutipan. Pratinjau data menampilkan tiga kolom: "Case ID", "Activity", dan "Timestamp". Setelah memastikan pengaturan sesuai, pengguna dapat melanjutkan dengan menekan tombol "Next" seperti pada Gambar 3.9.

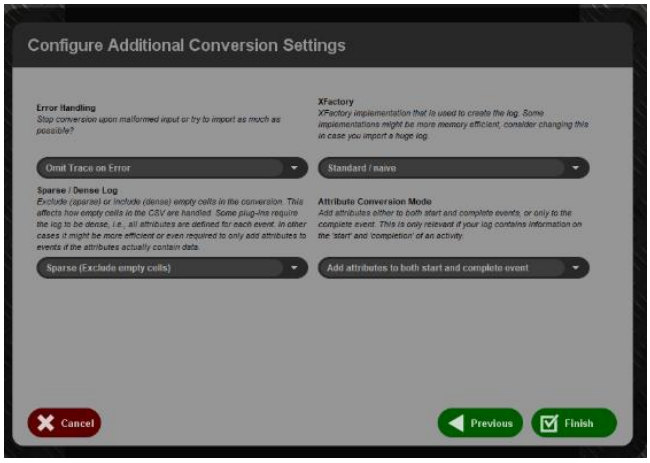


Gambar 3. 9 Konfigurasi *File* CSV

Konfigurasi di ProM dengan menambahkan beberapa kolom ke atribut XES yang sesuai. Seperti pada Gambar 3.10 Menambahkan "Case ID" ke case kolom, "Activity" ke *event* kolom, "Timestamp" ke *start time*, dan mengosongkan *Completion Time* lalu pencet *finish* seperti pada Gambar 3.11.

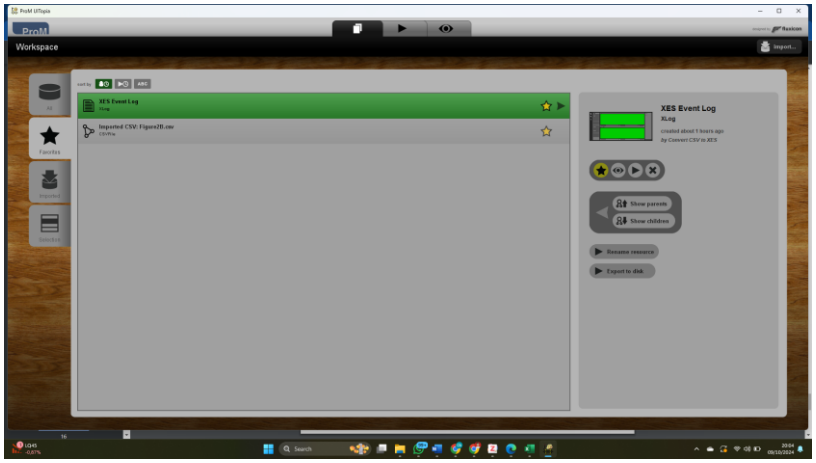


Gambar 3. 10 Konfigurasi *File* CSV ke *File* XES



Gambar 3. 11 Verifikasi Konfigurasi *File*

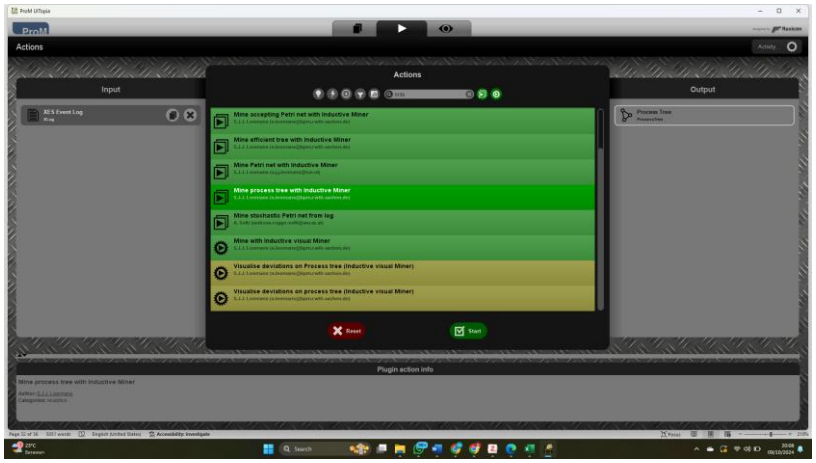
Ini adalah hasil konversi dari *file* CSV menjadi format XES pada Gambar 3.12.



Gambar 3. 12 XES *Event Logs* yang Telah Dikonversi

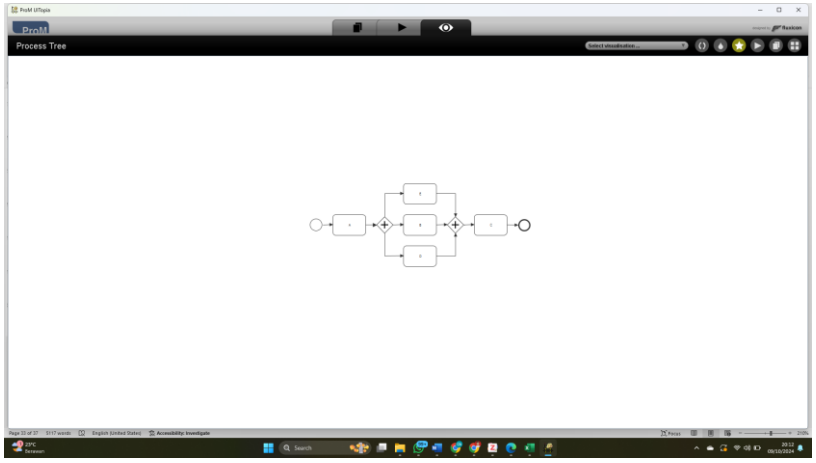
3.2.2 Pengujian Dalam *Inductive Miner*

File XES *event logs* yang telah dikonversi digunakan sebagai input dalam aplikasi ProM. Pada bagian *Actions,* pengguna memilih untuk melakukan *Mine process tree with Inductive Miner*, yang merupakan salah satu metode dalam *process mining* untuk mengekstraksi pohon proses dari *event logs*. Opsi ini membantu memvisualisasikan bagaimana proses berjalan berdasarkan data yang ada. Setelah opsi ini dipilih, pengguna bisa melanjutkan dengan menekan tombol Start untuk memulai proses penambangan (*mining*) pohon proses dari *event logs* yang telah dikonversi, sehingga menghasilkan model proses berdasarkan *log* yang tersedia seperti pada Gambar 3.13.



Gambar 3. 13 Memilih Metode *Inductive Miner*

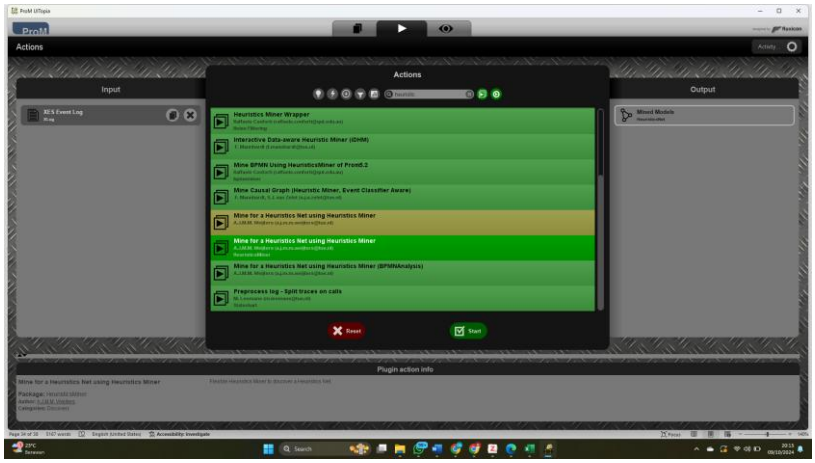
Hasil yang diperoleh divisualisasikan dalam bentuk BPMN (*Business Process Model and Notation*). Diagram BPMN ini menampilkan alur aktivitas yang telah diidentifikasi dari *log* proses, dengan simbol-simbol yang mewakili aktivitas, percabangan, dan alur proses. Hasil ini menggambarkan struktur proses secara detail, menunjukkan urutan aktivitas serta interaksi di antara mereka, dan digunakan untuk memvisualisasikan dan memahami proses bisnis yang telah dilakukan *process mining* dari data *log* seperti pada Gambar 3.14.



Gambar 3. 14 Model yang Dihasilkan *Inductive Miner*

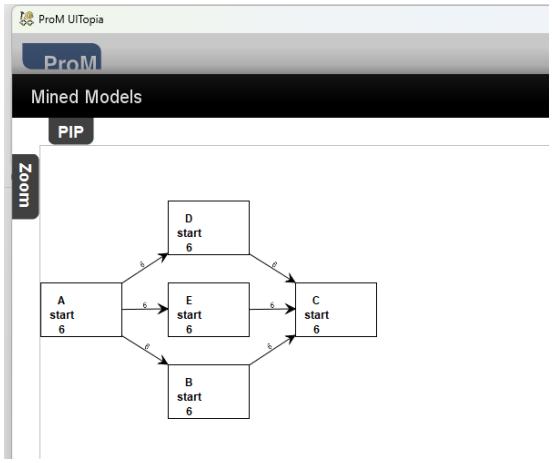
3.2.3 Pengujian Dalam *Heuristic Miner*

File XES *Event Logs* yang telah dikonversi digunakan sebagai input dalam aplikasi ProM. Pada bagian *Actions*, pengguna memilih untuk melakukan *Mine for a Heuristics Net using Heuristics Miner* dari daftar tindakan yang tersedia seperti pada Gambar 3.15. *Heuristics miner* membantu dalam mengidentifikasi pola-pola umum dalam *log*, termasuk dependensi antar aktivitas, frekuensi, dan hubungan kausal.



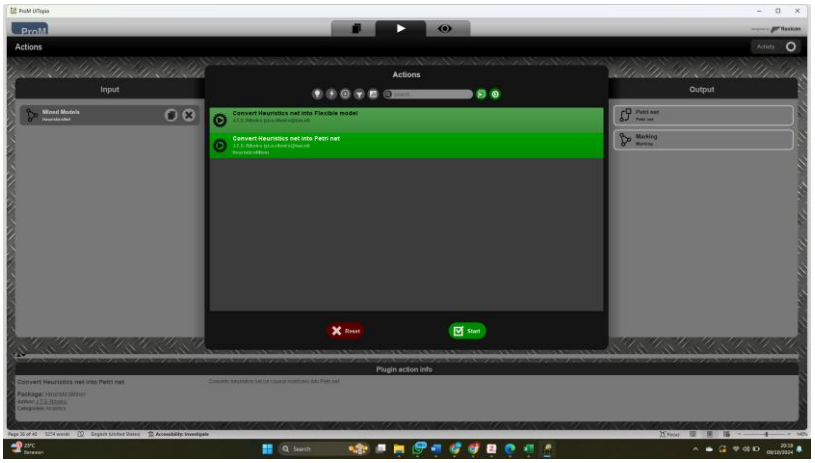
Gambar 3. 15 Memilih Metode *Heuristics Miner*

Hasil dari langkah *heuristics miner* ini tidak berbentuk BPMN, melainkan dalam bentuk *heuristics net* yang menggambarkan hubungan kausal antara aktivitas berdasarkan frekuensi dan dependensi dalam *event logs* seperti pada Gambar 3.16.

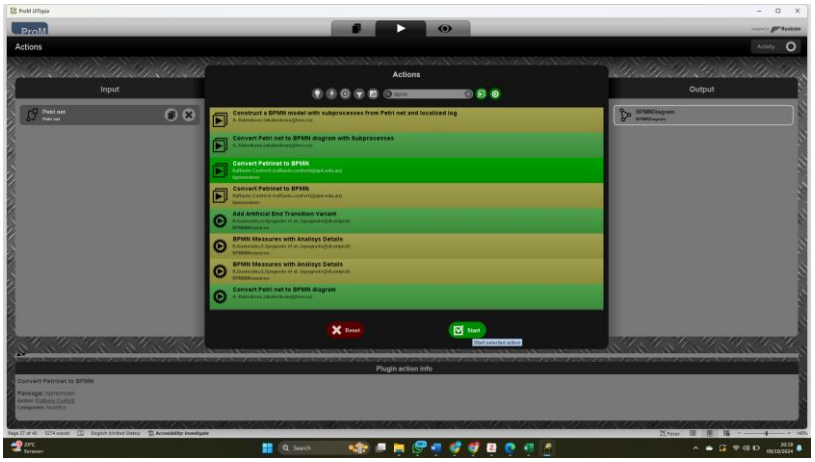


Gambar 3. 16 Model yang dihasilkan oleh *Heuristics Miner* dalam Bentuk *Heuristics Net*

Karena hasil ini bukan dalam bentuk BPMN, untuk mendapatkan model BPMN, hasil ini harus dikonversi terlebih dahulu menjadi Petri Net seperti pada Gambar 3.17. Proses konversi dari *heuristics net* ke *petri net* memungkinkan visualisasi yang lebih formal dan analitis dari proses tersebut. Setelah *petri net* diperoleh, langkah selanjutnya adalah mengonversinya menjadi BPMN seperti pada Gambar 3.18, yang merupakan model standar untuk menggambarkan proses bisnis. BPMN lebih mudah dibaca dan digunakan untuk komunikasi bisnis, sementara *Petri Net* memberikan representasi yang lebih mendalam dari segi perilaku sistem dan analisis proses.

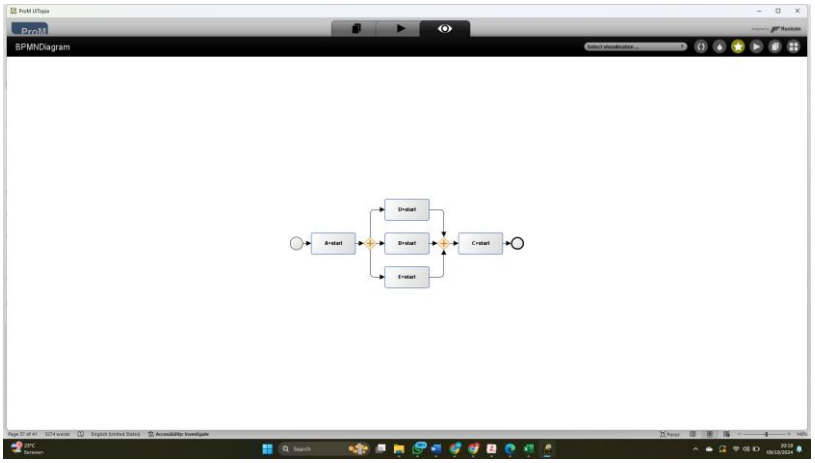


Gambar 3. 17 Mengkonversi *Heuristics net* Menjadi *Petri Net*



Gambar 3. 18 Mengkonversi *Petri Net* Menjadi BPMN

Dengan demikian, langkah konversi *Heuristics Net* ke *Petri Net*, lalu *Petri Net* ke BPMN, akan memberikan hasil akhir berupa diagram BPMN yang bisa digunakan untuk analisis bisnis lebih lanjut seperti pada Gambar 3.19.



Gambar 3. 19 Hasil yang Dihasilkan *Heuristics Miner* dalam Bentuk BPMN

3.3 Skenario Eksperimen

Eksperimen ini menggunakan 39 dataset *event logs* yang telah disimulasikan, yang mencakup berbagai skenario proses bisnis, mulai dari proses yang sederhana hingga sangat kompleks. Masing-masing *event logs* memiliki 9 karakteristik yang berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND, jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi cabang AND, kedalaman percabangan AND, variasi *gateway* XOR dan OR, variasi siklus (*loop*), kombinasi logika percabangan dengan *looping* dan *nested loops*. Untuk setiap dataset, algoritma *heursitc miner* dan *inductive miner* akan diterapkan, dan hasil model yang dihasilkan akan dibandingkan. Eksperimen ini akan mencakup tiga skenario untuk mengevaluasi kinerja algoritma *heuristic miner* dan *inductive miner* pada berbagai jenis *event logs*:

1. Pengujian kinerja kedua algoritma berdasarkan kompleksitas proses dalam *event logs*. Dalam skenario ini, berbagai *event log* dengan 9 karakteristik berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND,

jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi cabang AND, kedalaman percabangan AND, variasi *gateway* XOR dan OR, variasi siklus (*loop*), dan kombinasi logika percabangan dengan *looping*. Proses yang bercabang kompleks akan diujikan untuk melihat seberapa baik kedua algoritma menangkap logika percabangan yang benar dan menghasilkan model yang sesuai dengan kenyataan proses. Hasil dari kedua algoritma akan dievaluasi untuk mengukur kemampuan mereka dalam menangani kompleksitas logika percabangan. Pengujian ini akan menggunakan parameter default yang ada pada aplikasi ProM untuk algoritma *heuristic miner* dan *inductive miner*.

2. Mengeksplorasi varian dari *inductive miner*, *inductive miner*, seperti *inductive miner infrequent & lifecycle*, *inductive miner lifecycle, inductive miner exhaustive k-sucsessor, inductive miner incompleteness, inductive miner infrequent & all operators, inductive miner all operators,* dan *inductive miner infrequent* dengan tambahan pengujian *noise threshold*. Parameter *noise threshold* akan diuji dengan berbagai nilai untuk melihat bagaimana algoritma menangani *noise* dalam *event logs*. Setiap varian akan diterapkan tidak hanya pada dataset yang sama tetapi juga pada berbagai jenis dataset 9 karakteristik berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND, jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi cabang AND, kedalaman percabangan AND, variasi *gateway* XOR dan OR, variasi siklus (*loop*), dan kombinasi logika percabangan dengan *looping*.

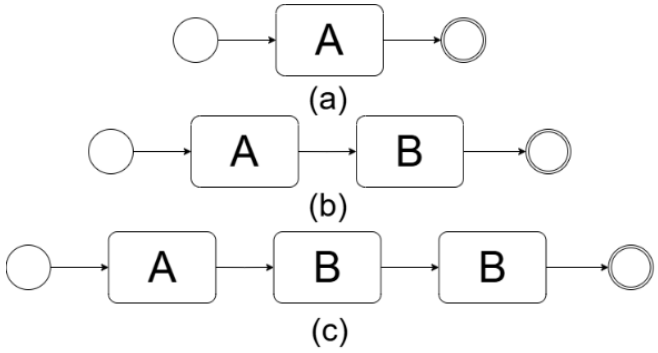
3. Skenario ketiga akan menguji pengaruh berbagai parameter yang digunakan dalam algoritma *heuristic miner* terhadap kinerja model yang dihasilkan. Parameter yang diuji meliputi *dependency threshold*, *relative-to-best threshold*, *length-one-loops*, *length-two-loops*, dan *long-distance dependency*. Variasi nilai *threshold* ini akan diterapkan untuk memahami bagaimana setiap parameter model proses yang dihasilkan. Setiap variasi akan diterapkan tidak hanya pada dataset yang sama tetapi juga pada berbagai jenis dataset 9 karakteristik berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND, jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi cabang AND, kedalaman percabangan AND, variasi *gateway* XOR dan OR, variasi siklus (*loop*), dan kombinasi logika percabangan dengan *looping*.
4. Skenario keempat akan menguji kinerja kedua algoritma dengan mengeksplorasi parameter dan varian algoritma. Dalam skenario ini, *event logs* yang digunakan adalah *event logs* dengan karakteristik *nested loops* yang berarti sebuah *loop* dapat memiliki satu atau lebih *loop* yang terdapat di dalamnya, yang masing-masing akan dieksekusi secara berulang untuk setiap iterasi dari *loop* luar. skenario ini diperlukan untuk memastikan bahwa dapat menangani kompleksitas dunia nyata yang sering kali mengandung *nested loops* dan pola pengulangan yang saling terkait.

BAB IV

HASIL DAN PEMBAHASAN

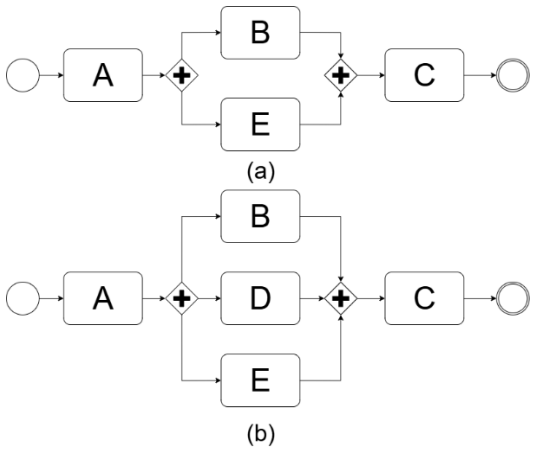
4.1 Hasil Pengumpulan Data

Eksperimen ini menggunakan 29 dataset *event logs* yang telah disimulasikan, yang mencakup berbagai skenario proses bisnis, mulai dari proses yang sederhana hingga sangat kompleks. Masing-masing *event logs* memiliki 9 karakteristik yang berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND, jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi cabang AND, kedalaman percabangan AND, variasi gateway XOR dan OR, variasi siklus (*loop*), dan kombinasi logika percabangan dengan *looping*. Berikut adalah beberapa model proses bisnis tersebut:



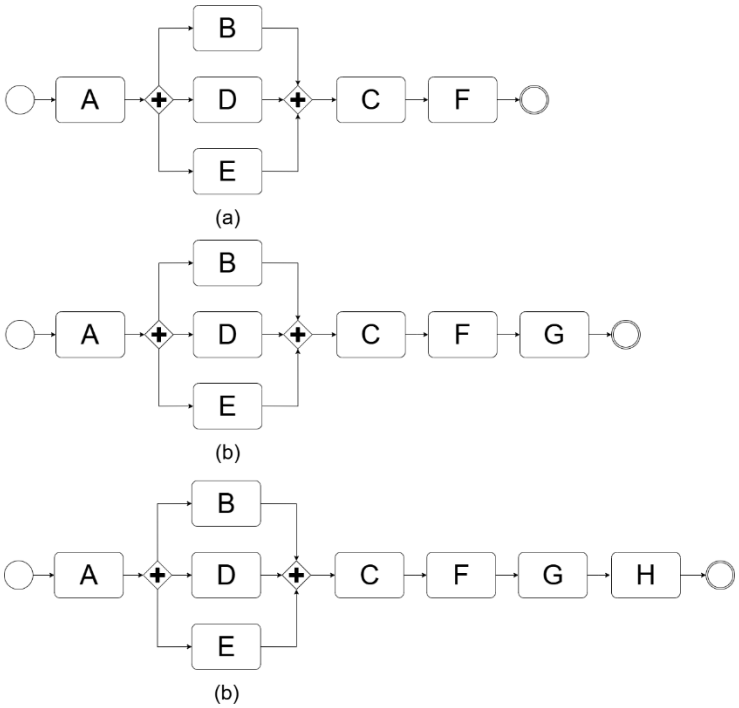
Gambar 4. 1 Model Proses Bisnis Sekuensial

Gambar 4.1 adalah model proses bisnis berurutan dengan jumlah aktivitas yang berbeda



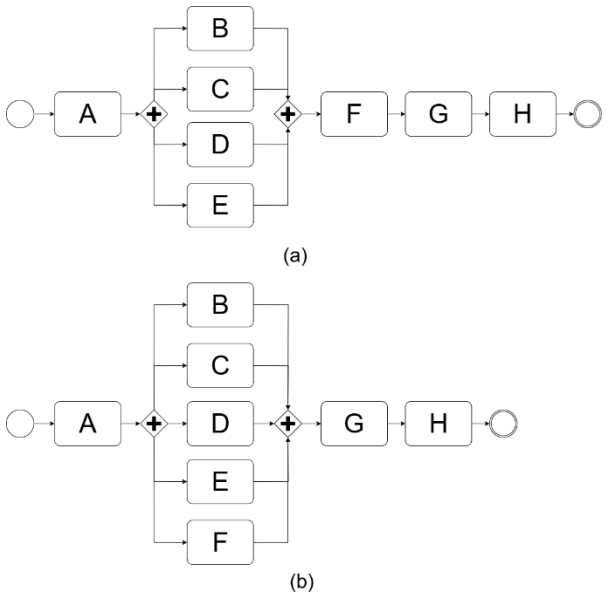
Gambar 4. 2 Model Proses Bisnis Percabangan AND

Gambar 4.2 adalah model proses bisnis dengan cabang AND yang memiliki jumlah cabang berbeda dan jumlah aktivitas yang berbeda.



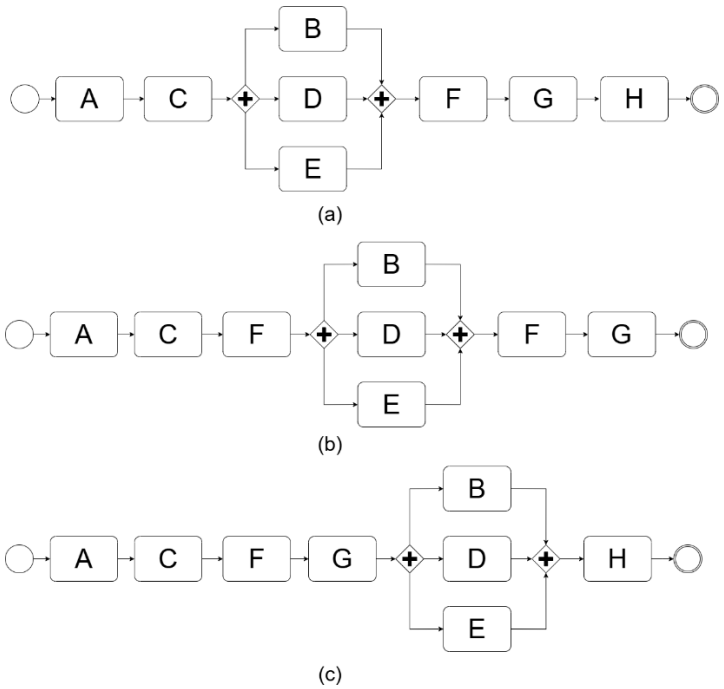
Gambar 4. 3 Model Proses Bisnis AND dengan Penambahan Sekuensial

Gambar 4.3 adalah model proses bisnis dengan cabang AND yang memiliki jumlah aktivitas berurutan berbeda.



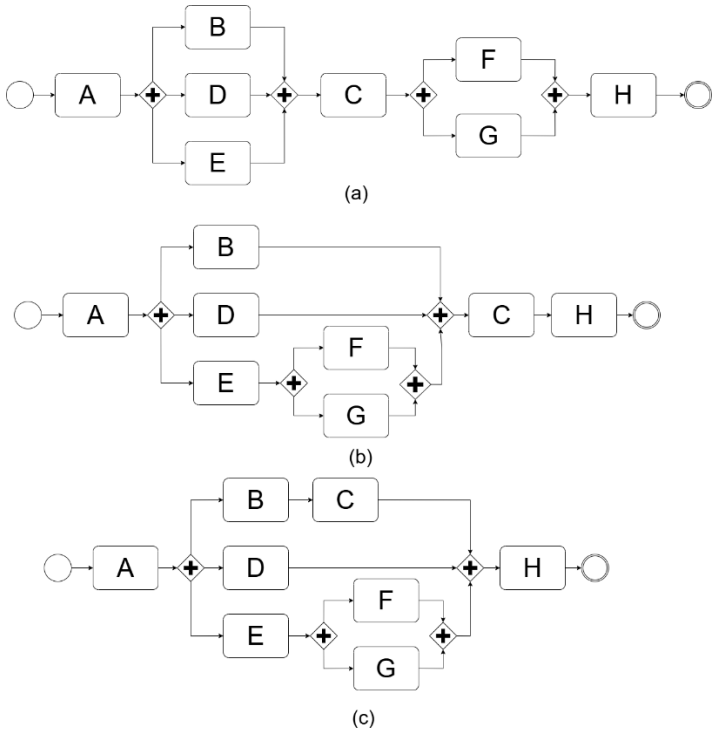
Gambar 4. 4 Model Proses Bisnis dengan Variasi Jumlah Percabangan

Gambar 4.4 adalah Model proses bisnis dengan cabang AND yang memiliki jumlah cabang berbeda dengan penambahan jumlah aktivitas.



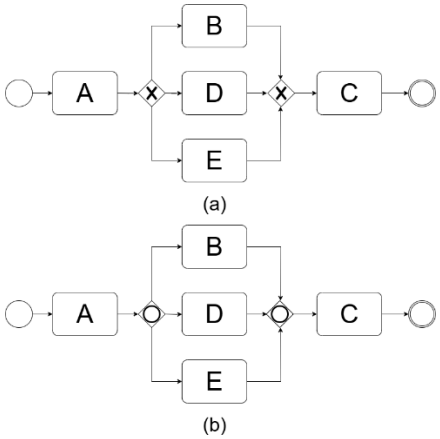
Gambar 4. 5 Model Proses Bisnis dengan Variasi Posisi Percabangan

Gambar 4.5 adalah model proses bisnis dengan cabang AND yang memiliki jumlah cabang berbeda dengan penambahan jumlah aktivitas.



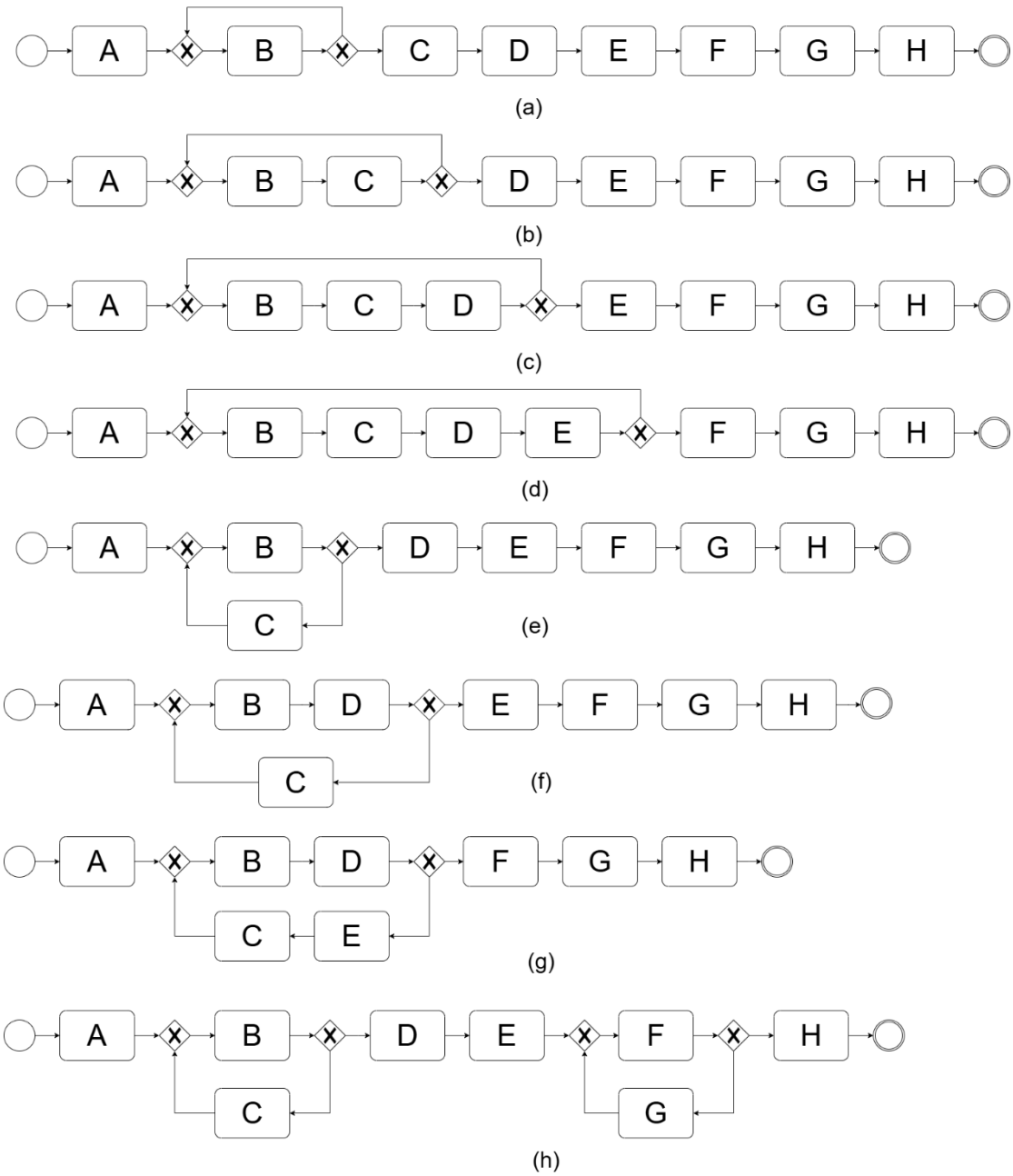
Gambar 4. 6 Model Proses Bisnis dengan Variasi Kedalaman Percabangan

Gambar 4.6 adalah model proses bisnis dengan cabang AND yang memiliki variasi lokasi cabang yang berbeda.



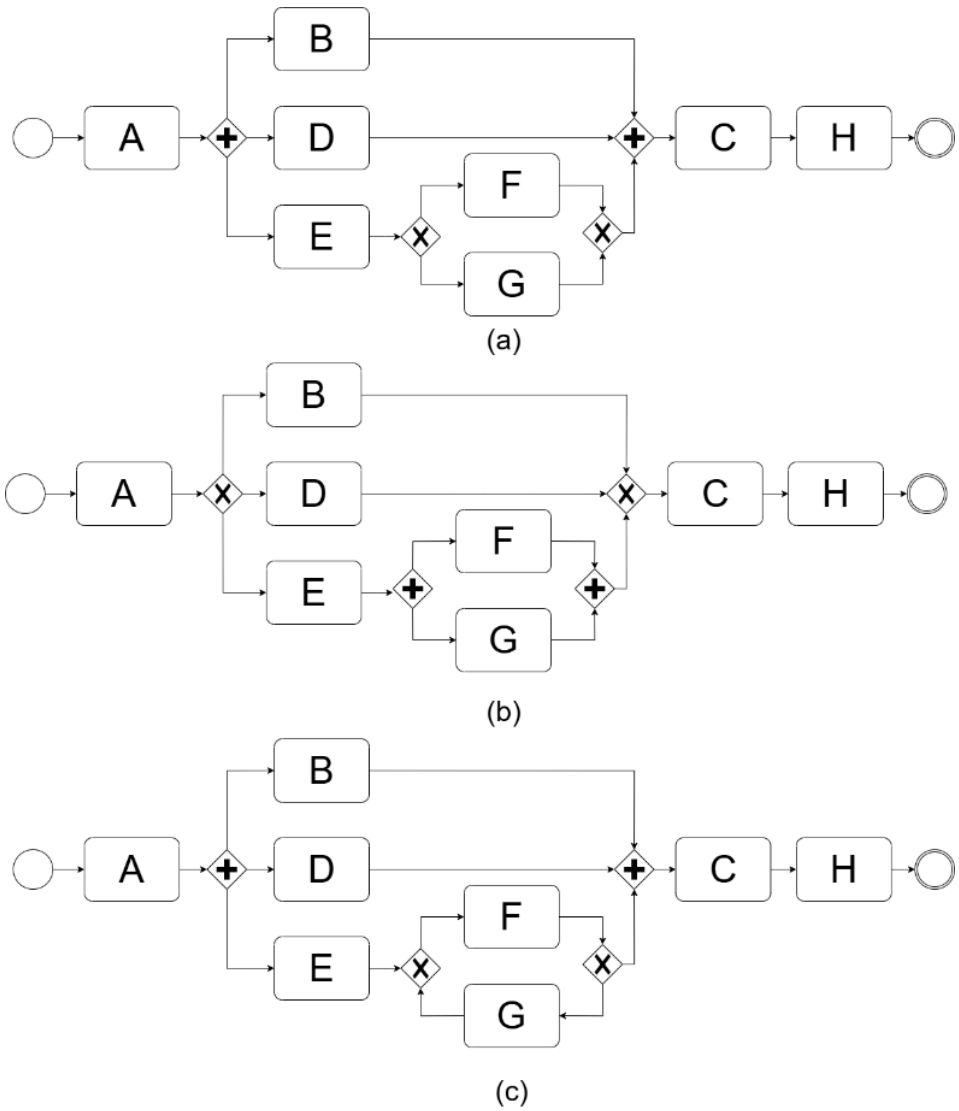
Gambar 4. 7 Model Proses Bisnis dengan Variasi Tipe Percabangan

Gambar 4.7 adalah model proses bisnis dengan variasi tipe percabangan yaitu XOR dan OR.



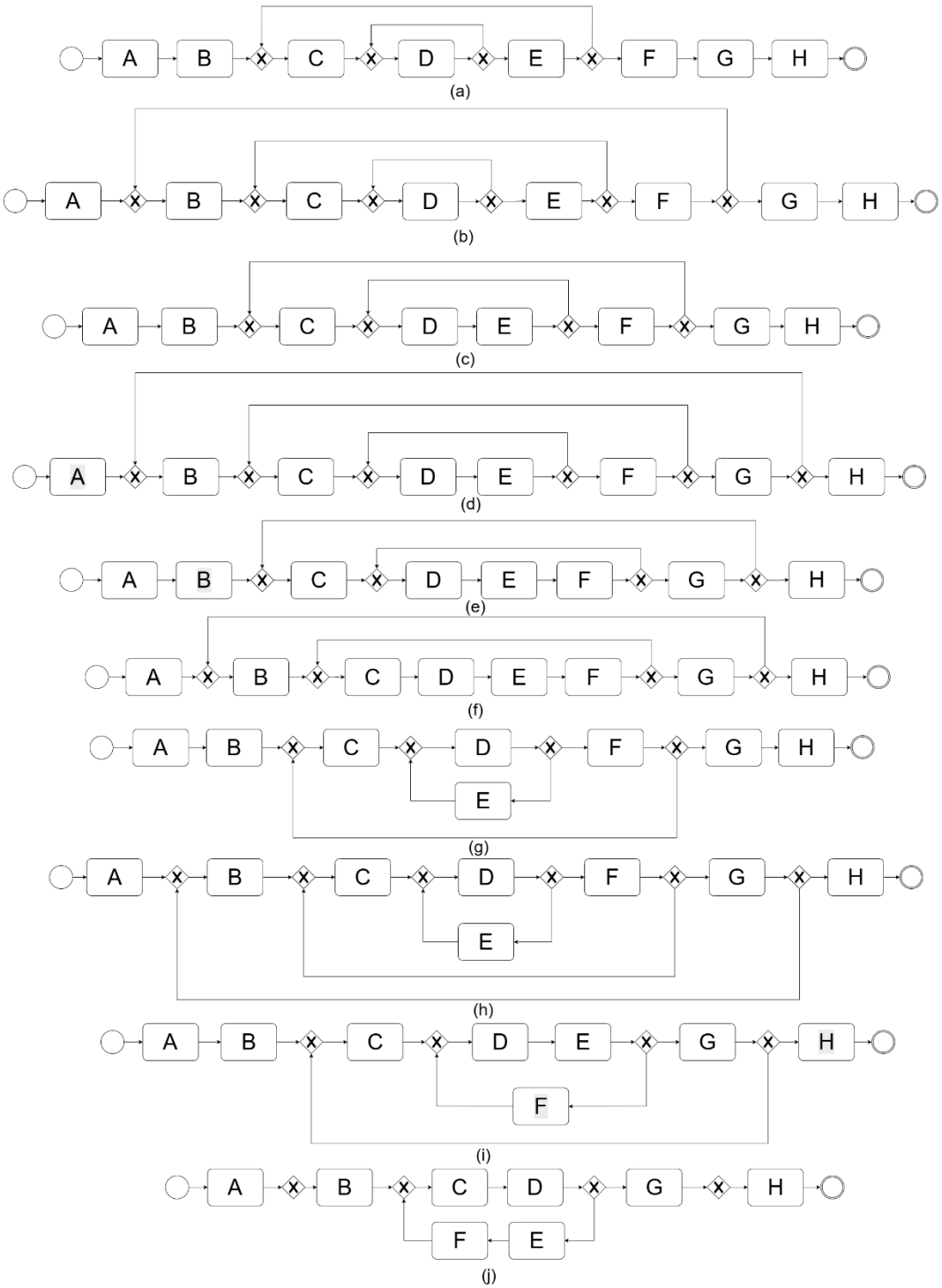
Gambar 4. 8 Model Proses Bisnis dengan Variasi Perulangan

Gambar 4.8 adalah model proses bisnis dengan variasi perulangan, terdapat model proses bisnis yang memiliki satu, dua, tiga , dan empat aktivitas yang berada dalam perulangan dan variasi perulangan yang lainya.



Gambar 4. 9 Model Proses Bisnis dengan Kombinasi Percabangan

Gambar 4.9 adalah model proses bisnis dengan variasi kombinasi gerbang logika dan perulangan.



Gambar 4. 10 Model Proses Bisnis dengan Kombinasi Percabangan

Gambar 4.10 model proses bisnis dengan kombinasi perulangan dalam perulangan atau *nested loops*.

Tabel 4. 1 Karakteristik BPMN

Figure	**Tujuan**
Gambar 4.1	Model proses bisnis berurutan dengan jumlah aktivitas yang berbeda. Menunjukkan pengaruh penambahan aktivitas berurutan terhadap kompleksitas proses bisnis.
Gambar 4.2	Model proses bisnis dengan cabang AND yang memiliki jumlah cabang berbeda. Menunjukkan pengaruh peningkatan jumlah cabang terhadap kompleksitas proses bisnis.
Gambar 4.3	Model proses bisnis dengan cabang AND yang memiliki jumlah aktivitas berurutan berbeda. Menunjukkan pengaruh penambahan aktivitas berurutan terhadap kompleksitas proses bisnis.
Gambar 4.4	Model proses bisnis dengan cabang AND yang memiliki jumlah cabang berbeda. Menunjukkan pengaruh peningkatan jumlah cabang terhadap kompleksitas proses bisnis.
Gambar 4.5	Model proses bisnis dengan cabang AND yang memiliki variasi lokasi cabang yang berbeda. Menunjukkan pengaruh variasi lokasi percabangan terhadap kompleksitas proses bisnis.
Gambar 4.6	Model proses bisnis dengan cabang AND yang memiliki variasi kedalaman yang berbeda. Menunjukkan pengaruh variasi kedalaman terhadap kompleksitas proses bisnis.
Gambar 4.7	Model proses bisnis dengan variasi percabangan XOR dan OR. Menunjukkan pengaruh setiap gateway terhadap kompleksitas proses bisnis.
Gambar 4.8	Model proses bisnis dengan variasi perulangan. Menunjukkan pengaruh berbagai variasi perulangan terhadap kompleksitas proses bisnis.
Gambar 4.9	Model proses bisnis dengan kombinasi gerbang logika dan perulangan. Menunjukkan pengaruh kombinasi gerbang logika dan perulangan terhadap kompleksitas proses bisnis.
Gambar 4.10	Model proses bisnis dengan kombinasi perulangan dalam perulangan atau *nested loops*. Menunjukkan pengaruh kombinasi kombinasi perulangan dalam perulangan atau *nested loops* terhadap kompleksitas proses bisnis.

Tabel 4.1 menjelaskan karakteristik model proses bisnis dan tujuanya. Berdasarkan dataset pada Gambar 4.1 sampai 4.10, didapatkan *event logs* yang ditunjukan pada Tabel. 4.2 berikut:

Tabel 4. 2 *Eventlogs* Model Proses Bisnis

Gambar	***Event Logs***
4.1a	A
4.1b	AB
4.1c	ABC
4.2a	ABDC, ADBC
4.2b	ABDEC, ABEDC, ADBEC, ADEBC, AEBDC, AEDBC
4.3a	ABDECF, ABEDCF, ADBECF, ADEBCF, AEBDCF, AEDBCF
4.3b	ABDECFG, ABEDCFG, ADBECFG, ADEBCFG, AEBDCFG, AEDBCFG

Gambar	*Event Logs*
4.3c	ABDECFGH, ABEDCFGH, ADBECFGH, ADEBCFGH, AEBDCFGH, AEDBCFGH
4.4a	ABCDEFGH, ABCEDFGH, ABDCEFGH, ABDECFGH, ABECDFGH, ABEDCFGH, ACBDEFGH, ACBEDFGH, ACDBEFGH, ACDEBFGH, ACEBDFGH, ACEDBFGH, ADBCEFGH, ADBECFGH, ADCBEFGH, ADCEBFGH, ADEBCFGH, ADECBFGH, AEBCDFGH, AEBDCFGH, AECBDFGH, AECDBFGH, AEDBCFGH, AEDCBFGH
4.4b	ABCDEFGH, ABCDFEGH, ABCEDFGH, ABCEFDGH, ABCFDEGH, ABCFEDGH, ABDCEFGH, ABDCFEGH, ABDECFGH, ABDEFCGH, ABDFCEGH, ABDFECGH, ABECDFGH, ABECFDGH, ABEDCFGH, ABEDFCGH, ABEFCDGH, ABEFDCGH, ABFCDEGH, ABFCEDGH, ABFDCEGH, ABFDECGH, ABFECDGH, ABFEDCGH, ACBDEFGH, ACBDFEGH, ACBEDFGH, ACBEFDGH, ACBFDEGH, ACBFEDGH, ACDBEFGH, ACDBFEGH, ACDEBFGH, ACDEFBGH, ACDFBEGH, ACDFEBGH, ACEBDFGH, ACEBFDGH, ACEDBFGH, ACEDFBGH, ACEFBDGH, ACEFDBGH, ACFBDEGH, ACFBEDGH, ACFDBEGH, ACFDEBGH, ACFEBDGH, ACFEDBGH, ADBCEFGH, ADBCFEGH, ADBECFGH, ADBEFCGH, ADBFCEGH, ADBFECGH, ADCBEFGH, ADCBFEGH, ADCEBFGH, ADCEFBGH, ADCFBEGH, ADCFEBGH, ADEBCFGH, ADEBFCGH, ADECBFGH, ADECFBGH, ADEFBCGH, ADEFCBGH, ADFBCEGH, ADFBECGH, ADFCBEGH, ADFCEBGH, ADFEBCGH, ADFECBGH, AEBCDFGH, AEBCFDGH, AEBDCFGH, AEBDFCGH, AEBFCDGH, AEBFDCGH, AECBDFGH, AECBFDGH, AECDBFGH, AECDFBGH, AECFBDGH, AECFDBGH, AEDBCFGH, AEDBFCGH, AEDCBFGH, AEDCFBGH, AEDFBCGH, AEDFCBGH, AEFBCDGH, AEFBDCGH, AEFCBDGH, AEFCDBGH, AEFDBCGH, AEFDCBGH, AFBCDEGH, AFBCEDGH, AFBDCEGH, AFBDECGH, AFBECDGH, AFBEDCGH, AFCBDEGH, AFCBEDGH, AFCDBEGH, AFCDEBGH, AFCEBDGH, AFCEDBGH, AFDBCEGH, AFDBECGH, AFDCBEGH, AFDCEBGH, AFDEBCGH, AFDECBGH, AFEBCDGH, AFEBDCGH, AFECBDGH, AFECDBGH, AFEDBCGH, AFEDCBGH
4.5a	ACBDEGH, ACBEDGH, ACDBEGH, ACDEBGH, ACEBDGH, ACEDBGH
4.5b	ACFBDEH, ACFBEDH, ACFDBEH, ACFDEBH, ACFEBDH, ACFEDBH
4.5c	ACFGBDEH, ACFGBEDH, ACFGDBEH, ACFGDEBH, ACFGEBDH, ACFGEDBH
4.6a	ABDECFGH, ABDECGFH, ABEDCFGH, ABEDCGFH, ADBECFGH, ADBECGFH, ADEBCFGH, ADEBCGFH, AEBDCFGH, AEBDCGFH, AEDBCFGH, AEDBCGFH
4.6b	ABDEFGCH, ABDEGFCH, ABEFGDCH, ABEGFDCH, ADBEFGCH, ADBEGFCH, ADEFGBCH, ADEGFBCH, AEFGDBCH, AEGFDBCH, AEFGBDCH, AEGFBDCH
4.6c	ABCDEFGH, ABCDEGFH, ABCEFGDH, ABCEGFDH, ADBCEFGH, ADBCEGFH, ADEFGBCH, ADEGFBCH, AEFGBCDH, AEGFBCDH, AEFGDBCH, AEGFDBCH
4.7a	ABC, ADC, AEC
4.7b	ABC, ADC, AEC, ABDC, ABEC, ADBC, ADEC, AEBC, AEDC, ABDEC, ABEDC, ADBEC, ADEBC, AEBDC, AEDBC
4.8a	ABBBCDEFGH
4.8b	ABCBCDEFGH
4.8c	ABCDBCDEFGH
4.8d	ABCDEBCDEFGH
4.8e	ABCBDEFGH
4.8f	ABDCBDEFGH
4.8g	ABDECBDFGH

Gambar	*Event Logs*
4.8h	ABCBDEFGFH
4.9a	ABDEFCH, ABDEGCH, ABEFDCH, ABEGDCH, ADBEFCH, ADBEGCH, ADEFBCH, ADEGBCH, AEFBDCH, AEGBDCH, AEFDBCH, AEGDBCH
4.9b	ABCH, ADCH, AEFGCH, AEGFCH
4.9c	ABDEFGFCH, ABEFGFDCH, ADBEFGFCH, ADEFGFBCH, AEFGFBDCH, AEFGFDBCH
4.10a	ABCDDECDEFGH
4.10b	ABCDDECDEFBCDEFGH
4.10c	ABCDEDEFCDEFGH
4.10d	ABCDEDEFCDEFGBCDEFGH
4.10e	ABCDEFDEFGCDEFGH
4.10f	ABCDEFCDEFGBCDEFGH
4.10g	ABCDEDFCDFGH
4.10h	ABCDEDFCDFGBCDFGH
4.10i	ABCDEFDEGCDEGH
4.10j	ABCDEFGBCDGH

Dengan menggunakan *event logs* pada Tabel 4.2, dilakukan *process mining* menggunakan algoritma *heuristic miner* dan *inductive miner*.

4.2 Hasil Perhitungan Kompleksitas Data

Pada penelitian ini, perhitungan kompleksitas menggunakan *Yaqin Complexity Formula*. Perhitungan ini bertujuan untuk mengetahui nilai kompleksitas masing-masing data. Perhitungan kompleksitas dilakukan dengan mengidentifikasi elemen-elemen utama dalam model yang tersaji dalam Tabel 4.3.

Tabel 4. 3 Hasil Pengukuran Metrik Model Proses Bisnis

BPM	Ns	Acs	As	C_{AND}	C_{XOR}	C_{OR}	Dm	C_{Cyc}	D_{Avg}	D_{max}	CD
4.1a	3	1	2	0	0	0	3	0	1.000	1	14.000
4.1b	4	2	3	0	0	0	4	0	1.000	1	14.000
4.1c	5	3	4	0	0	0	5	0	1.000	1	14.000
4.2a	8	4	8	8	0	0	6	0	1.500	2	21.000
4.2b	9	5	10	24	0	0	7	0	1.600	2	22.400
4.3a	10	6	11	24	0	0	8	0	1.500	2	21.000
4.3b	11	7	12	24	0	0	9	0	1.429	2	20.006
4.3c	12	8	13	24	0	0	10	0	1.375	2	19.250
4.4a	12	8	14	96	0	0	10	0	1.500	2	21.000
4.4b	12	8	15	480	0	0	10	0	1.625	2	22.750
4.5a	12	8	13	24	0	0	10	0	1.375	2	19.250
4.5b	12	8	13	24	0	0	10	0	1.375	2	19.250
4.5c	12	8	13	24	0	0	10	0	1.375	2	19.250
4.6a	14	8	16	32	0	0	10	0	1.625	2	22.750
4.6b	14	8	16	32	0	0	10	0	1.875	3	26.250

BPM	Ns	Acs	As	C_{AND}	C_{XOR}	C_{OR}	Dm	C_{Cyc}	D_{Avg}	D_{max}	CD
4.6c	14	8	16	32	0	0	10	0	2.000	3	28.000
4.7a	9	5	10	0	9	0	5	0	1.600	2	22.400
4.7b	9	5	10	0	0	105	7	0	1.600	2	22.400
4.8a	12	8	12	0	0	0	11	0.273	1.125	2	14.750
4.8b	12	8	12	0	0	0	12	0.500	1.250	2	17.400
4.8c	12	8	12	0	0	0	13	0.692	1.375	2	19.250
4.8d	12	8	12	0	0	0	14	0.857	1.500	2	21.000
4.8e	12	8	12	0	0	0	11	0.545	1.250	2	17.500
4.8f	12	8	12	0	0	0	12	0.750	1.375	2	19.250
4.8g	12	8	12	0	0	0	12	1.000	1.500	2	21.000
4.8h	14	8	15	0	0	0	12	1.000	1.500	2	21.000
4.9a	14	8	16	24	6	0	9	0	1.875	3	25.250
4.9b	14	8	16	8	9	0	8	0	1.875	3	26.250
4.9c	14	8	16	24	0	0	9	0.600	1.875	3	26.250
4.10a	14	8	15	0	0	0	14	0.643	1.250	3	17.500
4.10b	16	8	18	0	0	0	19	0.790	2.250	4	31.500
4.10c	14	8	15	0	0	0	16	0.750	1.875	4	26.250
4.10d	16	8	18	0	0	0	21	0.858	2.250	4	31.500
4.10e	14	8	15	0	0	0	17	0.883	2.750	4	38.500
4.10f	14	8	15	0	0	0	19	0.948	2.375	4	33.250
4.10g	14	8	15	0	0	0	13	0.923	2.625	4	36.750
4.10h	16	8	18	0	0	0	18	1.000	1.625	4	22.750
4.10i	14	8	15	0	0	0	15	1.000	1.750	4	24.500
4.10j	14	8	14	0	0	0	12	1.000	2.000	3	28.000

Setelah dilakukan perhitungan menggunakan *Yaqin Complexity Formula*, maka didapatkan hasil kompleksitas masing-masing data seperti yang tercantum pada Tabel 4.4.

Tabel 4. 4 Hasil Pengukuran Kompleksitas Model Proses Bisnis

Model Proses Bisnis	Hasil Perhitungan *Yaqin Complexity* Formula
4.1a	19.000
4.1b	21.000
4.1c	23.000
4.2a	45.000
4.2b	65.400
4.3a	66.000
4.3b	67.000
4.3c	68.250
4.4a	143.000
4.4b	529.750
4.5a	68.250
4.5b	68.250
4.5c	68.250
4.6a	84.750
4.6b	88.250
4.6c	90.000
4.7a	50.400

Model Proses Bisnis	Hasil Perhitungan *Yaqin Complexity* Formula
4.7b	146.400
4.8a	40.023
4.8b	42.000
4.8c	43.942
4.8d	45.857
4.8e	42.045
4.8f	44.000
4.8g	46.000
4.8h	51.000
4.9a	86.250
4.9b	73.250
4.9c	80.917
4.10a	46.500
4.10b	66.290
4.10c	56.000
4.10d	66.358
4.10e	66.383
4.10f	63.198
4.10g	66.673
4.10h	57.750
4.10i	54.500
4.10j	57.000

Tabel 4.4 adalah hasil perhitungan kompleksitas menggunakan *Yaqin Complexity Formula*. Perhitungan kompleksitas bertujuan untuk mengetahui tingkat kompleksitas masing-masing model proses bisnis serta sejauh mana metode *heuristic miner* dan *inductive miner* dapat mengukur serta menangani proses bisnis dengan kompleksitas yang berbeda-beda. Nilai kompleksitas yang bervariasi menunjukkan adanya perbedaan dalam struktur dan tingkat kesulitan proses bisnis, yang dapat mempengaruhi efektivitas metode penambangan proses dalam merekonstruksi model bisnis yang optimal.

4.3 Hasil Eksperimen Skenario 1

Dalam skenario ini, berbagai *event logs* dengan 9 karakteristik berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND, jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi

cabang AND, kedalaman percabangan AND, variasi gateway XOR dan OR, variasi siklus (*loop*), dan kombinasi logika percabangan dengan *looping*. Proses yang bercabang kompleks akan diujikan untuk melihat seberapa baik kedua algoritma menangkap logika percabangan yang benar dan menghasilkan model yang sesuai dengan kenyataan proses. Hasil dari kedua algoritma akan dievaluasi untuk mengukur kemampuan mereka dalam menangani kompleksitas logika percabangan. Pengujian ini akan menggunakan parameter default yang ada pada aplikasi ProM untuk algoritma *heuristic miner* dan *inductive miner*. Setelah melakukan proses *discovery* menggunakan metode *heuristic miner* dan *inductive miner*, hasil BPMN akan dibandingkan d0engan BPMN referensi menggunakan *jaccard similarity*, berikut adalah hasil perhitungan *similarity* antara dua model proses bisnis hasil *discovery* dan model proses bisnis dataset yang ditunjukkan pada Tabel 4.5.

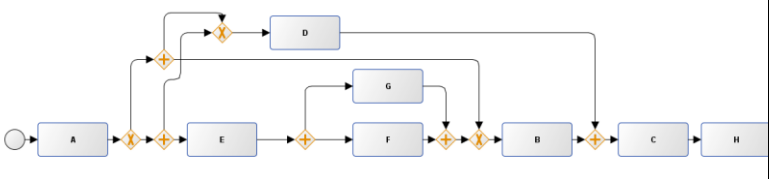
Tabel 4. 5 Hasil Pengukuran *Similarity* Skenario 1

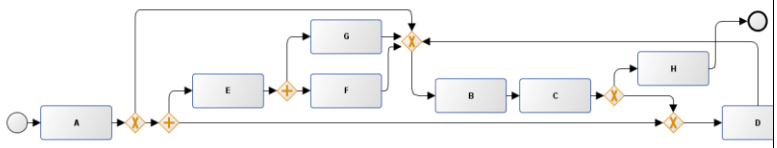
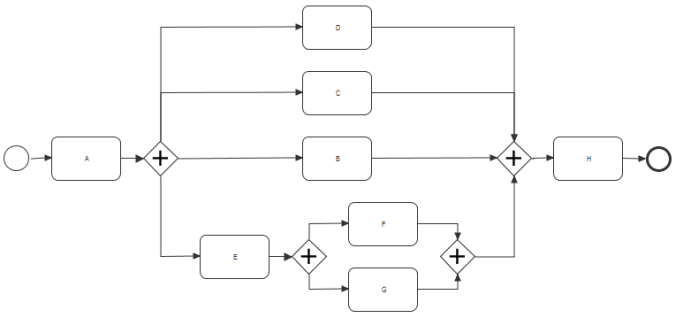
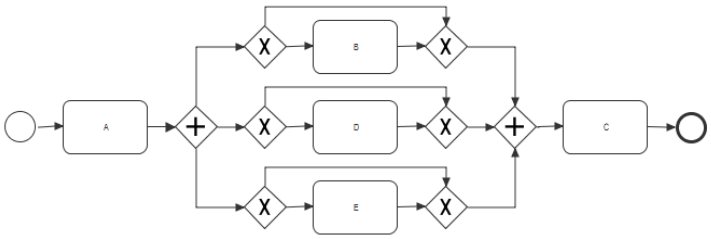
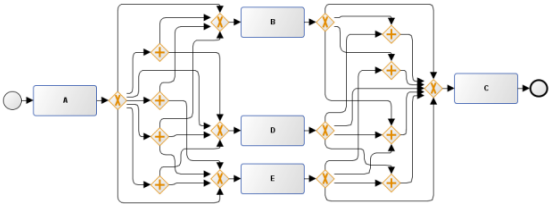
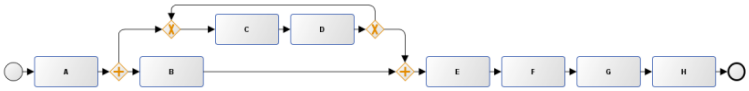
Dataset	***Similarity***	
	Inductive Miner	***Heuristic Miner***
4.1a	1	1
4.1b	1	1
4.1c	1	1
4.2a	1	1
4.2b	1	1
4.3a	1	1
4.3b	1	1
4.3c	1	1
4.4a	1	1
4.4b	1	1
4.5a	1	1
4.5b	1	1
4.5c	1	1
4.6a	1	1
4.6b	1	0.750
4.6c	0.965	0.848
4.7a	1	1
4.7b	0.424	0.269
4.8a	1	1
4.8b	1	1

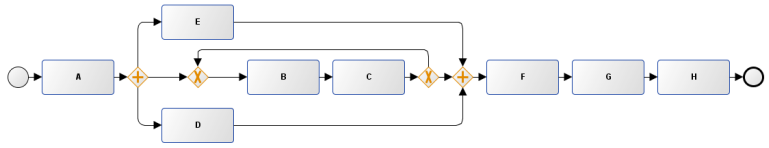
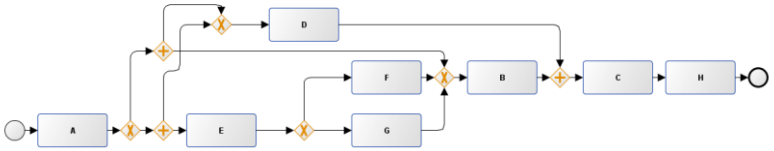
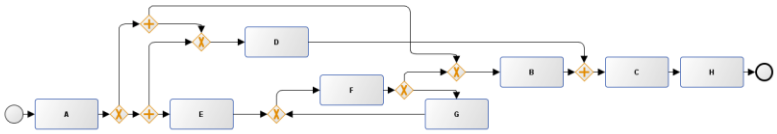
Dataset	*Similarity*	
	Inductive Miner	***Heuristic Miner***
4.8c	0.815	1
4.8d	0.785	1
4.8e	1	1
4.8f	1	1
4.8g	1	1
4.8h	1	1
4.9a	1	0.800
4.9b	1	1
4.9c	1	0.777

Pada Tabel 4.5 diketahui bahwa terdapat perbedaan *similarity* pada setiap data. setelah melakukan perhitungan *similarity*, didapatkan bahwa nilai *similarity* antara dua model proses mendekati atau sama dengan 1, maka semakin mirip pula kedua model proses tersebut. Sebaliknya, ketika nilai *similarity* antara dua model proses dibawah 1, maka semakin rendah tingkat kemiripan kedua model proses. Data dengan nilai *similarity* 1 merupakan model proses dari hasil *discovery* yang cocok dengan model proses pada dataset. Sedangkan, model proses dengan nilai *similarity* dibawah 1 merupakan model proses dari hasil discovery yang tidak cocok dengan dataset. Berikut adalah model proses bisnis hasil *discovery* yang memiliki nilai *similarity* di bawah 1 pada eksperimen skenario 1, berikut adalah model proses bisnis tersebut

Tabel 4. 6 Model Proses Bisnis yang Memiliki Nilai *Similarity* < 1

Dataset	**Hasil**	**Similarity**
4.6b	 *Heuristic Miner*	0.750

Dataset	Hasil	Similarity
4.6c	 *Heuristic Miner*	0.848
4.6c	 *Inductive Miner*	0.965
4.7b	 *Inductive Miner*	0.424
4.7b	 *Heuristic Miner*	0.269
4.8c	 *Inductive Miner*	0.815

Dataset	Hasil	Similarity
4.8d	 *Inductive Miner*	0.785
4.9a	 *Heuristic Miner*	0.800
4.9c	 *Heuristic Miner*	0.777

Dari Tabel 4.6 dapat dilihat bahwa *inductive miner* tidak mampu menghasilkan proses bisnis pada BPMN Gambar 4.6c, 4.7b, 4.8c, dan 4.8d. *Inductive* kurang mampu dalam menangani dataset dengan karakteristik variasi kedalaman cabang AND yang berbeda yaitu Gambar 4.6c dengan nilai kompleksitas 90.000. Gambar 4.7b melibatkan variasi percabangan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.8c dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 43.942 dan 45.857. Hal ini terjadi karena kode algoritma *inductive miner* dalam aplikasi ProM tidak mampu menangani dataset tersebut. Ketidakmampuan ini terlihat pada kode berikut.

```
Start

Initialize cutFinders with:

CutFinderIMExclusiveChoice

CutFinderIMSequence

CutFinderIMConcurrentWithMinimumSelfDistance

CutFinderIMLoop

CutFinderIMConcurrent

For each cutFinder in cutFinders do

    c := call cutFinder.findCut with log, logInfo, and
minerState

    If c is valid then

        Return c

Return null

End
```

Inductive miner gagal dalam menangani dataset dengan variasi kedalaman yang berbeda karena *cut finders* yang digunakan memiliki keterbatasan dalam menangani struktur proses yang sangat dalam atau bercabang. *Cut finders* berfungsi untuk membagi *log* menjadi *sublog* berdasarkan pola aktivitas. Namun, pada struktur yang sangat dalam, bercabang, atau kompleks, seperti *loop* yang dalam atau percabangan XOR, algoritma ini kesulitan menemukan pemotongan yang valid.

Selain itu, *inductive miner* gagal dalam menangani dataset dengan pola OR karena tidak memiliki *cutfinder* OR yang secara eksplisit menangani pola OR dalam proses. Pola OR memungkinkan aktivitas yang dapat terjadi dalam beberapa jalur berbeda, dan tanpa *cutfinder* yang dapat menangani kasus ini. *Inductive miner* akan

mencoba menangani pola tersebut dengan XOR atau AND, yang tidak cocok untuk percabangan alternatif. Dalam hal ini, *inductive miner* menganggap aktivitas sebagai percabangan eksklusif (XOR) atau paralel (AND), yang mengarah pada pemodelan yang salah dan kesalahan dalam memisahkan *log*. Hal ini menyebabkan struktur proses yang tidak akurat, jalur yang seharusnya bersifat alternatif dianggap sebagai bagian dari percabangan yang tidak saling tergantung atau paralel.

```
for each edgeIndex in dfg.getDirectlyFollowsEdges() do
    source := dfg.getDirectlyFollowsEdgeSourceIndex(edgeIndex)
    target := dfg.getDirectlyFollowsEdgeTargetIndex(edgeIndex)
    if not dfg.isStartActivity(source) then
        if not dfg.isEndActivity(source) then
            if not dfg.isStartActivity(target) then
                components.mergeComponentsOf(source, target)
            end;
        end;
    else
        if not dfg.isEndActivity(source) then
            components.mergeComponentsOf(source, target)
        end;
    end;
end;
```

Inductive miner gagal menangani perulangan yang melibatkan lebih dari dua aktivitas karena pendekatan penggabungan komponen yang digunakan hanya efektif untuk *loop* yang melibatkan dua aktivitas saja. Pada kode tersebut, *Inductive*

Miner menggabungkan aktivitas yang terhubung langsung menggunakan components.mergeComponentsOf(source, target), tetapi tidak memperhitungkan bahwa dalam struktur perulangan lebih dari dua aktivitas, hubungan antar aktivitas harus dipahami sebagai bagian dari sebuah siklus berulang.

Dari Tabel 4.6 dapat dilihat bahwa *heuristic miner* tidak mampu menghasilkan proses bisnis pada BPMN Gambar 4.6b, 4.6c, 4.7b, 4.9a, dan 4.9c. *heuristic miner* kurang mampu dalam menangani dataset dengan karakteristik variasi kedalaman cabang AND yang berbeda yaitu Gambar 4.6b dan 4.6c dengan nilai kompleksitas 88.250 dan 90.000. Gambar 4.7b melibatkan variasi percabangan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.9a dan 4.9c menunjukkan model proses bisnis dengan kombinasi gerbang logika dan perulangan dengan nilai kompleksitas 86.250 dan 80.917. Hal ini terjadi karena kode algoritma *heuristic miner* dalam aplikasi ProM tidak mampu menangani dataset tersebut. Ketidakmampuan ini terlihat pada kode berikut.

```
START

For each event in log:

    Calculate direct and long-range dependencies

    If there are complex OR branches, check XOR conditions
carefully

    If dependencies exceed threshold, add to relationship sets
(input/output sets)

For each pair of events (i, j):

    If distance between them is large (long-distance
dependency), calculate the L2L dependency
```

```
If complex OR or deep branches are encountered:

    Apply XOR logic carefully to ensure no connections are
missed

END
```

Heuristics miner mengalami kesulitan dalam menangani variasi cabang OR dan kedalaman karena keterbatasan dalam mengelola ketergantungan jarak jauh dan penerapan logika XOR dalam membangun hubungan antar aktivitas. Dalam kode yang ada, *heuristics miner* mengandalkan perhitungan ketergantungan langsung dan jarak jauh antara aktivitas-aktivitas yang ada dalam *log*, menggunakan fungsi-fungsi seperti calculateLongDistanceDependencyMeasure untuk menghitung ketergantungan yang lebih jauh. Namun, ketika dihadapkan dengan percabangan OR yang memiliki banyak jalur eksekusi, hubungan antar aktivitas menjadi lebih kompleks dan sering kali saling bertumpang tindih. Kode ini mengandalkan logika XOR pada fungsi seperti xorInWelcome dan xorOutWelcome untuk memproses hubungan antar aktivitas yang terhubung melalui logika OR, namun, dalam percabangan yang memiliki kedalaman berbeda atau banyak jalur eksekusi, logika ini tidak selalu mampu menangani dengan tepat semua hubungan yang mungkin terjadi. Ketergantungan yang lebih panjang dan kedalaman cabang yang bervariasi dapat menyebabkan beberapa koneksi antar aktivitas terlewat atau teridentifikasi dengan tidak akurat, sehingga *heuristics miner* kesulitan dalam menangani variasi OR dan kedalaman cabang yang tinggi.

Berdasarkan hasil eksperimen skenario 1, model proses bisnis yang dihasilkan melalui *discovery* menggunakan *Inductive Miner* menghasilkan 4 model proses bisnis dengan tingkat *similarity* di bawah 1, sementara model proses bisnis

yang dihasilkan menggunakan *Heuristic Miner* menghasilkan 5 model proses bisnis dengan *similarity* di bawah 1. Hal ini menunjukkan bahwa *Inductive Miner* lebih unggul dibandingkan *Heuristic Miner* dalam menangani dataset yang memiliki tingkat kompleksitas yang bervariasi. *Inductive Miner* mampu menghasilkan model yang lebih sesuai dengan data yang ada, dengan tingkat kesesuaian yang lebih tinggi, meskipun terdapat beberapa model yang memiliki *similarity* di bawah 1. Sebaliknya, *Heuristic Miner* menghasilkan lebih banyak model dengan ketidakcocokan yang lebih besar terhadap dataset tersebut.

4.4 Hasil Eksperimen Skenario 2

Pada skenario kedua dilakukan pengujian menggunakan varian dari *Inductive Miner*, seperti *inductive miner infrequent & lifecycle*, *inductive miner lifecycle, inductive miner exhaustive k-sucsessor, inductive miner incompleteness, inductive miner infrequent & all operators, inductive miner all operators,* dan *inductive miner infrequent* dengan tambahan pengujian *noise threshold*. Setiap varian akan diterapkan tidak hanya pada dataset yang sama tetapi juga pada berbagai jenis dataset 9 karakteristik berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND, jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi cabang AND, kedalaman percabangan AND, variasi *gateway* XOR dan OR, variasi siklus (*loop*), dan kombinasi logika percabangan dengan *looping*. Pengujian dengan parameter *noise threshold* hanya dapat dilakukan pada varian *inductive miner infrequent*, *inductive miner infrequent & all operators*, dan *inductive miner infrequent & lifecycle,* karena hanya ketiga varian tersebut yang mendukung penyesuaian terhadap tingkat *noise* dalam proses

ekstraksi model. Berikut adalah perhitungan hasil similarity masing-masing varian *inductive miner*.

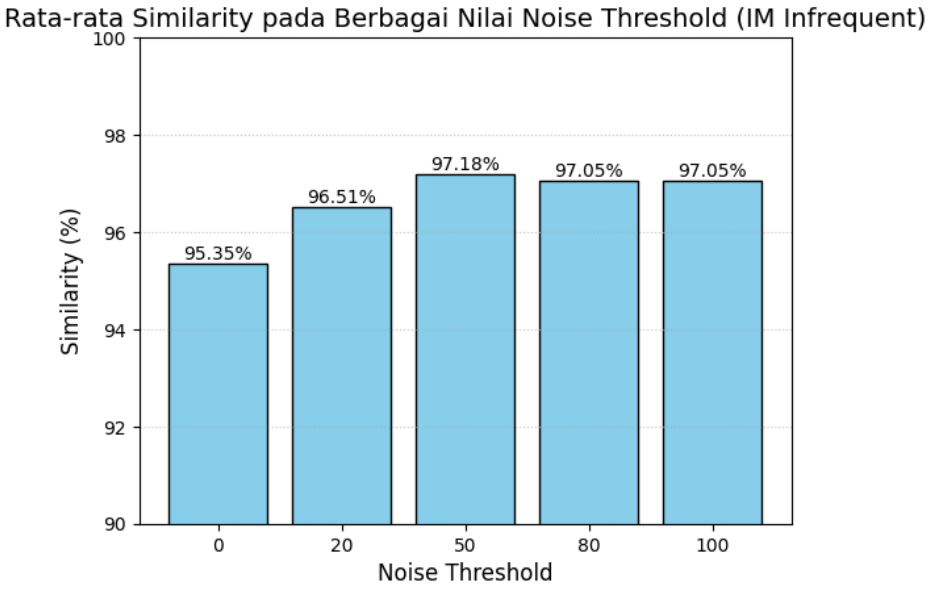
Hasil perhitungan similarity *inductive miner infrequent*:

Tabel 4. 7 Hasil Pengukuran *Similarity Inductive Miner Infrequent*

Dataset	***Similarity***				
	IM Infrequent (0)	***IM Infrequent (20)***	***IM Infrequent (50)***	***IM Infrequent (80)***	***IM Infrequent (100)***
4.1a	1	1	1	1	1
4.1b	1	1	1	1	1
4.1c	1	1	1	1	1
4.2a	1	1	1	1	1
4.2b	1	1	1	1	1
4.3a	1	1	1	1	1
4.3b	1	1	1	1	1
4.3c	1	1	1	1	1
4.4a	1	1	1	1	1
4.4b	1	1	1	1	1
4.5a	1	1	1	1	1
4.5b	1	1	1	1	1
4.5c	1	1	1	1	1
4.6a	1	1	1	1	1
4.6b	1	1	0.964	0.964	0.964
4.6c	0.965	0.965	0.965	0.965	0.965
4.7a	1	1	1	1	1
4.7b	0.424	0.424	0.882	0.882	0.882
4.8a	1	1	0.772	0.772	0.772
4.8b	1	1	1	1	1
4.8c	0.687	0.815	0.815	0.815	0.815
4.8d	0.578	0.785	0.785	0.785	0.785
4.8e	1	1	1	1	1
4.8f	1	1	1	1	1
4.8g	1	1	1	1	1
4.8h	1	1	1	1	1
4.9a	1	1	1	1	1
4.9b	1	1	1	1	1
4.9c	1	1	1	0.964	0.964

Tabel 4.7 menyajikan pengujian terhadap 29 dataset menggunakan algoritma *inductive miner infrequent* dengan lima konfigurasi nilai *noise threshold*: 0, 20, 50, 80, dan 100. Setiap kolom menunjukkan nilai *similarity* antara model hasil proses *mining* dan proses bisnis referensi untuk setiap dataset pada nilai *threshold* tertentu. Setelah melakukan perhitungan *similarity*, didapatkan bahwa

nilai *similarity* antara dua model proses mendekati atau sama dengan 1, maka semakin mirip pula kedua model proses tersebut. Sebaliknya, ketika nilai *similarity* antara dua model proses dibawah 1, maka semakin rendah tingkat kemiripan kedua model proses. Perbedaan hasil *similarity* dari masing-masing nilai *threshold* menunjukkan bahwa tingkat toleransi terhadap *noise* dapat memengaruhi akurasi model proses yang dihasilkan. Berikut adalah presentase rata-rata *similarity*:



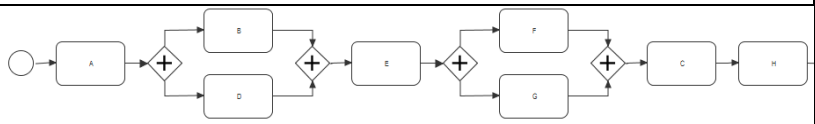
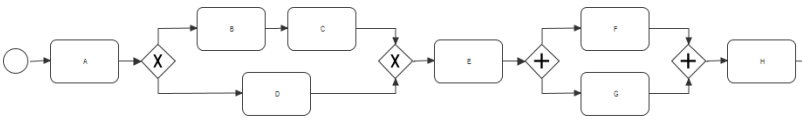
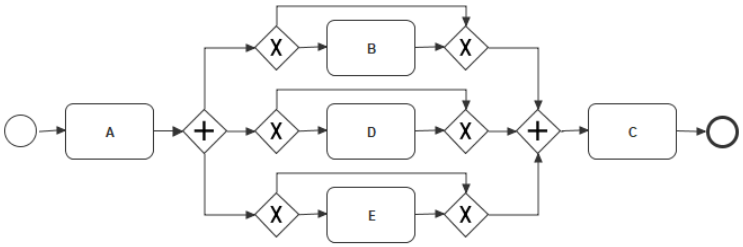
Gambar 4. 11 Grafik Presentase *Similarity Inductive Miner Infrequent*

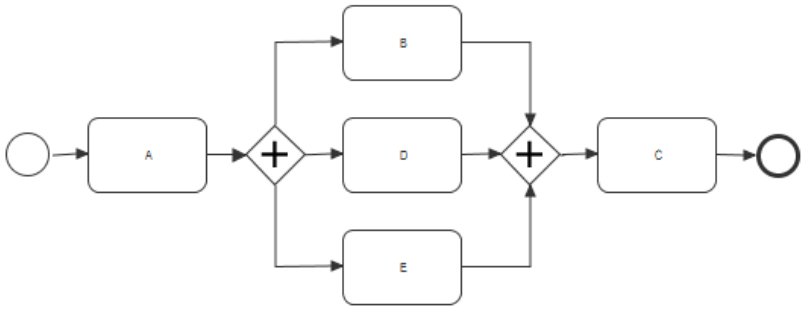
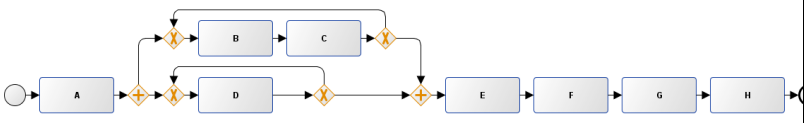
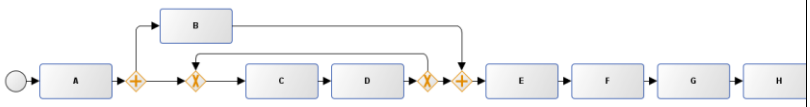
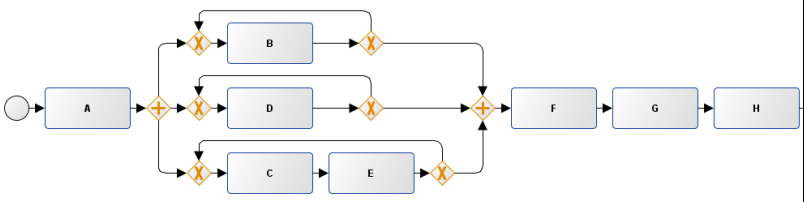
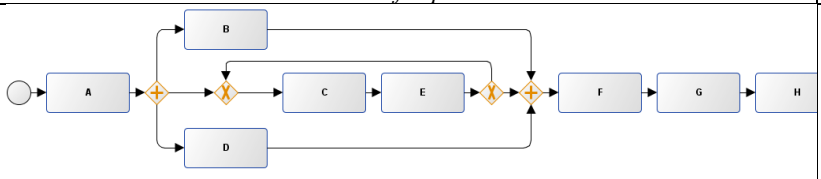
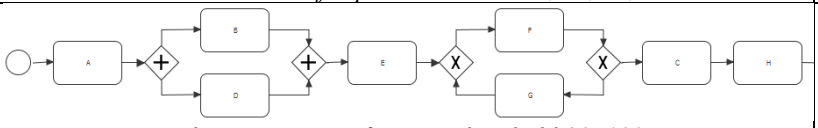
Threshold 0 dan 20 menghasilkan jumlah nilai *similarity* sebesar 1 yang paling banyak dibandingkan *threshold* lainnya. Artinya, pada nilai *threshold* ini, model proses yang dihasilkan sepenuhnya cocok dengan proses bisnis referensi pada lebih banyak dataset. Meskipun demikian, secara rata-rata, *threshold* 50 memiliki nilai *similarity* tertinggi, yakni sebesar 97,18%. Ini menunjukkan bahwa meskipun jumlah nilai 1 sedikit lebih rendah, *similarity* pada dataset yang tidak menghasilkan nilai 1 tetap tinggi. Artinya, *threshold* 50 memberikan kompromi terbaik antara generalisasi dan akurasi. Pada *threshold* 80 dan 100, menghasilkan

rata-rata *similarity* yang tinggi, yakni masing-masing sebesar 97,05%. Meskipun keduanya memiliki rata-rata yang lebih tinggi dibanding *threshold* 0 (95,35%) dan 20 (96,51%), jumlah dataset dengan nilai *similarity* 1 lebih sedikit.

Dengan demikian, *threshold* 0 dan 20 menghasilkan lebih banyak nilai *similarity* sempurna (1), namun memiliki rata-rata *similarity* yang lebih rendah. *Threshold* 50 memberikan rata-rata *similarity* tertinggi secara keseluruhan, menunjukkan keseimbangan antara akurasi dan generalisasi. Sementara itu, *threshold* 80 dan 100 juga memiliki rata-rata *similarity* tinggi, meskipun jumlah nilai 1 lebih sedikit. Berikut adalah beberapa model proses bisnis hasil *discovery Inductive Miner Infrequent* dengan beberapa pengujian *threshold* yang memiliki nilai *similarity* di bawah 1:

Tabel 4. 8 Model Proses Bisnis yang Memiliki *Nilai Similarity* < 1

Dataset	**Hasil**	***Similarity***
4.6b	 *Inductive Miner Infrequent threshold 50, 80, 100*	0.964
4.6c	 *Inductive Miner Infrequent threshold 0, 20, 50, 80, 100*	0.965
4.7b	 *Inductive Miner Infrequent threshold 0, 20,*	0.424

Dataset	Hasil	*Similarity*
4.7b	 *Inductive Miner Infrequent threshold 50, 80, 100*	0.882
4.8a	 *Inductive Miner Infrequent threshold 50, 80, 100*	0.772
4.8c	 *Inductive Miner Infrequent threshold 0*	0.687
4.8c	 *Inductive Miner Infrequent threshold 20, 50, 80, 100*	0.815
4.8d	 *Inductive Miner Infrequent threshold 0*	0.578
4.8d	 *Inductive Miner Infrequent threshold 20, 50, 80, 100*	0.785
4.9c	 *Inductive Miner Infrequent threshold 80, 100*	0.964

Dari Tabel 4.8 dapat dilihat bahwa *inductive miner infrequent* 0-20 tidak mampu menghasilkan proses bisnis yang sama pada BPMN Gambar 4.6c, 4.7b, 4.8a, 4.8c, dan 4.8d. Untuk *threshold* 50 adalah 4.6b, 4.6c, 4.7b, 4.8a, 4.8c, dan 4.8d

sedangkan 80 dan 100 adalah 4.6b, 4.6c, 4.7b, 4.8a, 4.8c, 4.8d dan 4.9c. *Inductive Miner Infrequent* kurang mampu dalam menangani dataset dengan karakteristik 4.7b, 4.8a, 4.8c, 4.8d, dan 4.9c. Gambar 4.6b dan 4.6c, yang memiliki karakteistik variasi kedalaman cabang AND yang berbeda dengan nilai kompleksitas 88.250 dan 90.000. Gambar 4.7b melibatkan variasi percabangan XOR dan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.8a, 4.8c, dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 42.000, 43.942, dan 45.857. Sementara itu, Gambar 4.9c menggambarkan kombinasi gerbang logika dan perulangan, yang menciptakan interaksi antara cabang-cabang logika dan siklus dengan nilai kompleksitas 80.917.

Inductive miner dan *inductive miner infrequent* menghadapi masalah yang sama dalam menangani dataset dengan perulangan lebih dari dua aktivitas, pola OR, dan percabangan AND dengan variasi kedalaman yang berbeda. Kedua metode ini kesulitan dalam memodelkan perulangan, percabangan alternatif (OR), dan cabang AND yang memiliki kedalaman yang bervariasi, yang menyebabkan kesalahan dalam representasi proses. Namun, *inductive miner infrequent* menghasilkan hasil yang lebih baik dibandingkan *inductive miner* karena penerapan *log filtering* dan pengurangan kompleksitas pada pohon proses. IMf memfilter aktivitas yang jarang terjadi (*infrequent activities*) dengan menggunakan *noise threshold* untuk mengurangi pengaruh data yang tidak relevan, yang dapat menyebabkan model yang lebih rumit dan tidak representatif. Selain itu, IMf menggunakan teknik pengurangan pohon (*tree reduction*) untuk menyederhanakan model, menjaga

hanya informasi yang penting dan relevan, sehingga menghasilkan pohon proses yang lebih sederhana dan efisien.

Dari pengujian threshold pada algoritma *inductive miner infrequent,* dilihat bahwa persentase yang lebih tinggi dihasilkan ketika nilai *threshold* dinaikkan, menunjukkan bahwa algoritma mampu menangkap kompleksitas yang lebih besar dalam model proses bisnis seiring dengan peningkatan *threshold*. Namun, meskipun terjadi peningkatan persentase pada beberapa dataset, terdapat juga dataset yang malah mengalami penurunan ketika *threshold*-nya dinaikkan, yang mengindikasikan bahwa pada beberapa kasus, peningkatan *threshold* justru menyebabkan ketidakcocokan yang lebih besar antara model yang dihasilkan dan dataset yang diuji. Banyak aktivitas *infrequent* (jarang terjadi) yang dianggap sebagai *noise* dan dihapus. Meskipun aktivitas tersebut jarang terjadi, mereka mungkin membawa informasi penting terkait dengan variasi dalam proses atau situasi yang lebih spesifik

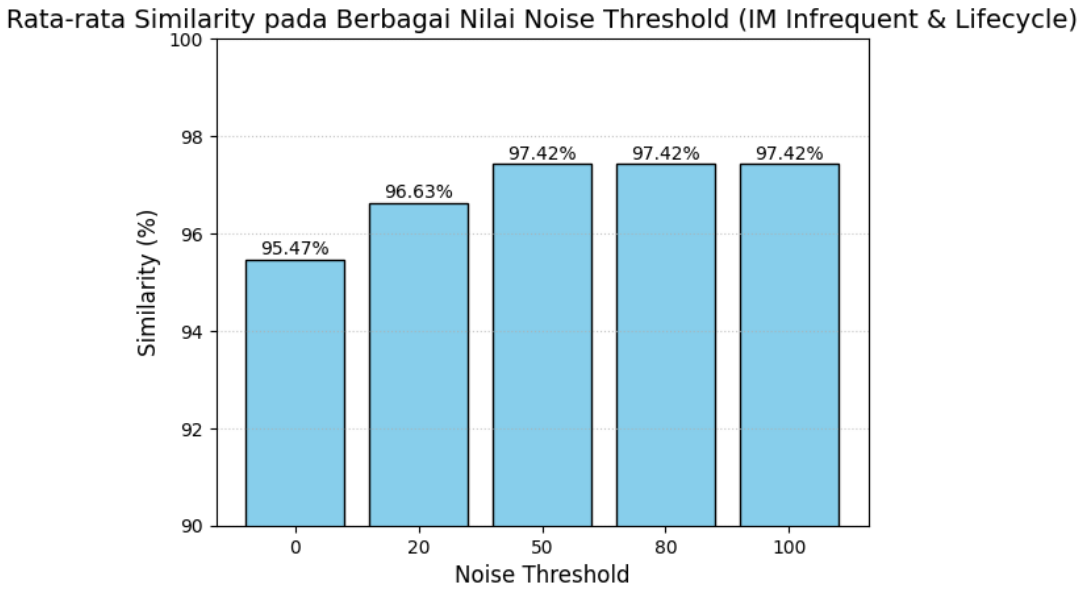
Pada *threshold* 0-20, *inductive miner infrequent* mampu menghasilkan model yang memiliki nilai similarity 1 pada dataset 4.6b dan 4.9c. Namun, pada *threshold* 50, 80, dan 100, nilai similarity justru mengalami penurunan, mengindikasikan bahwa peningkatan *threshold* menyebabkan ketidakcocokan yang lebih besar antara model yang dihasilkan dan dataset pada gambar tersebut. Hal ini mungkin disebabkan oleh keterbukaan yang lebih besar dalam mengidentifikasi ketergantungan dan struktur yang lebih rumit pada *threshold* yang lebih tinggi, yang tidak selalu sesuai dengan pola dalam dataset yang sederhana.

Berikut adalah hasil perhitungan similarity *inductive miner infrequent & lifecycle*:

Tabel 4. 9 Hasil Pengukuran *Similarity Inductive Miner Infrequent & Lifecycle*

Dataset	***Similarity***				
	IM Infrequent & Lifecycle (0)	***IM Infrequent & Lifecycle (20)***	***IM Infrequent & Lifecycle (50)***	***IM Infrequent & Lifecycle (80)***	***IM Infrequent & Lifecycle (100)***
4.1a	1	1	1	1	1
4.1b	1	1	1	1	1
4.1c	1	1	1	1	1
4.2a	1	1	1	1	1
4.2b	1	1	1	1	1
4.3a	1	1	1	1	1
4.3b	1	1	1	1	1
4.3c	1	1	1	1	1
4.4a	1	1	1	1	1
4.4b	1	1	1	1	1
4.5a	1	1	1	1	1
4.5b	1	1	1	1	1
4.5c	1	1	1	1	1
4.6a	1	1	1	1	1
4.6b	1	1	1	1	1
4.6c	1	1	1	1	1
4.7a	1	1	1	1	1
4.7b	0.424	0.424	0.882	0.882	0.882
4.8a	1	1	0.772	0.772	0.772
4.8b	1	1	1	1	1
4.8c	0.687	0.815	0.815	0.815	0.815
4.8d	0.578	0.785	0.785	0.785	0.785
4.8e	1	1	1	1	1
4.8f	1	1	1	1	1
4.8g	1	1	1	1	1
4.8h	1	1	1	1	1
4.9a	1	1	1	1	1
4.9b	1	1	1	1	1
4.9c	1	1	1	1	1

Tabel 4.9 menunjukkan hasil pengukuran *similarity* menggunakan varian algoritma *inductive miner infrequent & lifecycle* terhadap sejumlah dataset dengan pengujian pada berbagai nilai *noise threshold* (0, 20, 50, 80, dan 100). Berikut adalah presentasenya:

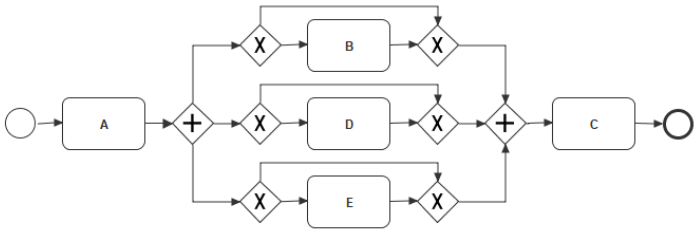
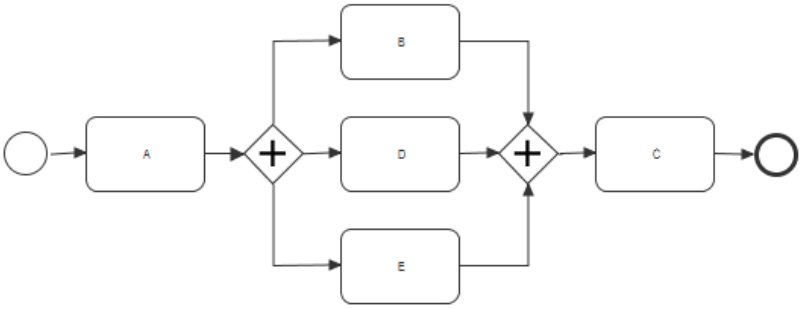
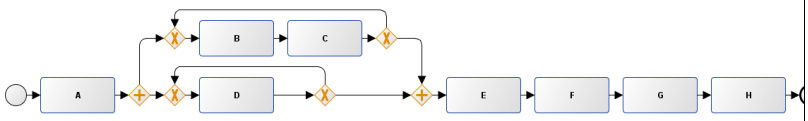
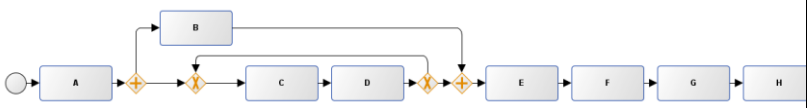
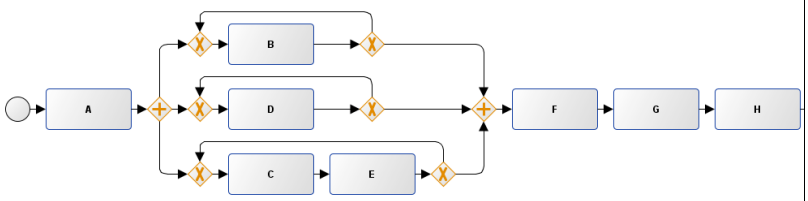
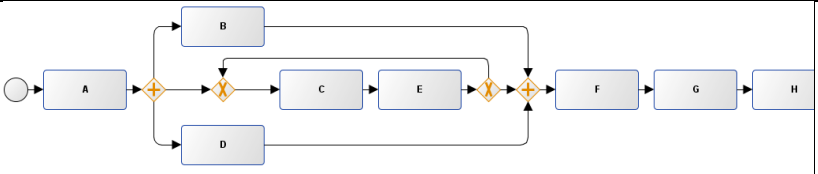


Gambar 4. 12 Grafik Presentase *Similarity IM Infrequent & Lifecycle*

Dari data terlihat bahwa *noise threshold* 0 dan 20 menghasilkan jumlah kecocokan sempurna (nilai *similarity* = 1) yang paling banyak. Namun, secara rata-rata, *threshold* 20 memberikan *nilai similarity* yang lebih tinggi (96,63%) dibanding *threshold* 0 (95,47%). Sementara itu, *threshold* 50, 80, dan 100 menghasilkan rata-rata *similarity* tertinggi secara keseluruhan, yaitu 97,42%. Meskipun jumlah nilai *similarity* = 1 pada ketiga *threshold* tersebut lebih sedikit dibandingkan threshold 0 dan 20, tetapi secara umum mereka menunjukkan kestabilan dan konsistensi hasil yang tinggi.

Dengan demikian, dapat disimpulkan bahwa *threshold* 50, 80, dan 100 lebih optimal dalam menjaga kualitas model secara keseluruhan, sedangkan *threshold* 0 dan 20 cenderung memberikan lebih banyak hasil sempurna namun dengan rata-rata *similarity* yang sedikit lebih rendah. Berikut adalah beberapa model proses bisnis hasil *discovery inductive miner infrequent & lifecycle* dengan beberapa pengujian *threshold* yang memiliki nilai *similarity* di bawah 1:

Tabel 4. 10 Model Proses Bisnis yang Memiliki Nilai *Similarity* < 1

Dataset	**Hasil**	***Similarity***
4.7b	 *Inductive Miner Infrequent & Life Cycle threshold 0, 20,*	0.424
4.7b	 *Inductive Miner Infrequent & Life Cycle threshold 50, 80, 100*	0.882
4.8a	 *Inductive Miner Infrequent & Lifecycle threshold 50, 80, 100*	0.772
4.8c	 *Inductive Miner Infrequent & Lifecycle threshold 0*	0.687
4.8c	 *Inductive Miner Infrequent & Lifecycle threshold 20, 50, 80, 100*	0.815
4.8d	 *Inductive Miner Infrequent & Life Cycle threshold 0*	0.578
4.8d	 *Inductive Miner Infrequent & Life Cycle threshold 20, 50, 80, 100*	0.785

Dari Tabel 4.10 dapat dilihat bahwa *inductive miner infrequent & lifecycle* tidak mampu menghasilkan proses bisnis yang sama pada BPMN Gambar 4.7b, 4.8c, dan 4.8d dan pada *threshold* yang lebih tinggi, seperti 50, 80, dan 100, Gambar 4.8a justru mengalami penurunan dalam nilai s*imilarity*, menunjukkan bahwa meskipun *threshold* yang lebih tinggi memungkinkan algoritma untuk menangkap lebih banyak kompleksitas, pada beberapa kasus, peningkatan *threshold* malah menyebabkan ketidakcocokan yang lebih besar antara model yang dihasilkan dan dataset yang diuji. Gambar 4.7b melibatkan variasi percabangan XOR dan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.8a, 4.8c, dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 42.000, 43.942, dan 45.857.

Inductive miner infrequent & lifecycle menghadapi masalah yang sama dengan varian *inductive miner & inductive miner infrequent* dalam menangani dataset dengan perulangan lebih dari dua aktivitas, pola OR, namun berbeda dengan yang lain, metode ini mampu menangani dataset dengan karakteristik AND dengan variasi kedalaman. *inductive miner infrequent* dan *inductive miner lifecycle* memiliki kemampuan lebih baik dalam menangani percabangan AND dengan variasi kedalaman dibandingkan varian lain karena perbedaan dalam cara mereka menangani pemisahan *log* dan pengolahan aktivitas *infrequent*. Hal ini disebabkan karena varian ini memiliki LogSplitterIMlc dan *CutFinder* yang memadai.berikut adalah *pseudocode* LogSplitterIMlc*:*

```
class LogSplitterIMi implements LogSplitter:

private logSplitterIMi: LogSplitter = new LogSplitterIMi()

function split(log: IMLLog, logInfo: IMLLogInfo, cut: Cut,
minerState: MinerState): LogSplitResult:

    result = logSplitterIMi.split(log, logInfo, cut, minerState)

    if minerState.parameters.isRepairLifeCycle() AND
(cut.operator != Operator.xor AND cut.operator !=
Operator.concurrent):

        newSublogs = empty list of IMLlog

        for each sublog in result.sublogs:

newSublogs.append(LifeCycle(minerState.parameters.isDebug()).pre
ProcessLog(sublog))

        result.sublogs = newSublogs
```

LogSplitterIMlc yang digunakan di *inductive miner infrequent lifecycle* menyediakan metode untuk memperbaiki *log* dengan siklus kehidupan yang rusak atau tidak konsisten, meningkatkan keakuratan dalam pemisahan *sublogs* pada percabangan AND dengan variasi kedalaman. Ketika ada percabangan AND yang lebih dalam, proses reparasi siklus ini memperbaiki *log* sehingga aktivitas-aktivitas yang saling terhubung dapat dipisahkan dengan benar.

Dari pengujian threshold pada algoritma *inductive miner infrequent & lifecycle,* dilihat bahwa persentase yang lebih tinggi dihasilkan ketika nilai *threshold* dinaikkan, menunjukkan bahwa algoritma mampu menangkap kompleksitas yang lebih besar dalam model proses bisnis seiring dengan peningkatan *threshold*. Namun, meskipun terjadi peningkatan persentase pada beberapa dataset, terdapat juga dataset yang malah mengalami penurunan ketika *threshold*-nya dinaikkan, yang mengindikasikan bahwa pada beberapa kasus,

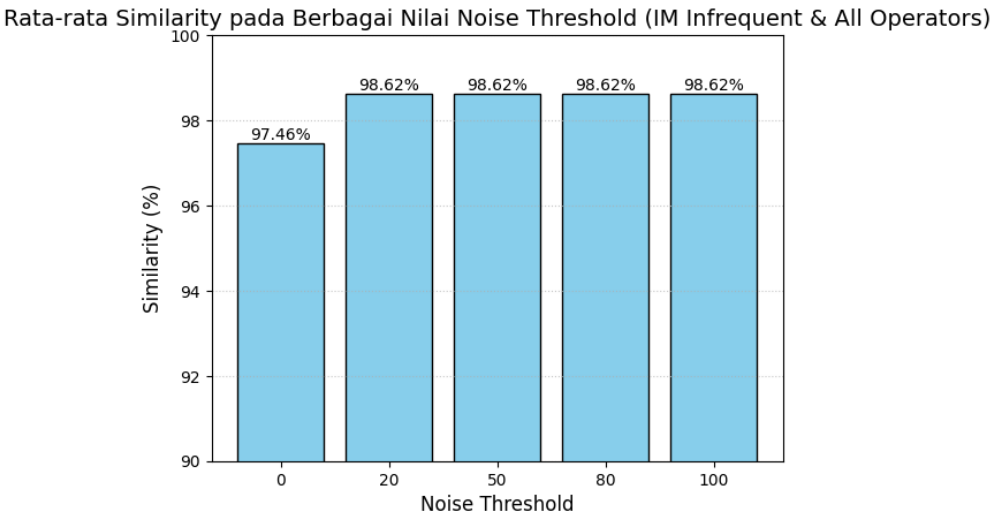
peningkatan threshold justru menyebabkan ketidakcocokan yang lebih besar antara model yang dihasilkan dan dataset yang diuji. Banyak aktivitas *infrequent* (jarang terjadi) yang dianggap sebagai *noise* dan dihapus. Meskipun aktivitas tersebut jarang terjadi, mereka mungkin membawa informasi penting terkait dengan variasi dalam proses atau situasi yang lebih spesifik

Berikut adalah hasil perhitungan *similarity* dari *inductive miner infrequent & all operators*:

Tabel 4. 11 Hasil Pengukuran *Similarity Inductive Miner Infrequent & All Operators*

Dataset	***Similarity***				
	IM Infrequent & All Operators (0)	***IM Infrequent & All Operators (20)***	***IM Infrequent & All Operators (50)***	***IM Infrequent & All Operators (80)***	***IM Infrequent & All Operators (100)***
4.1a	1	1	1	1	1
4.1b	1	1	1	1	1
4.1c	1	1	1	1	1
4.2a	1	1	1	1	1
4.2b	1	1	1	1	1
4.3a	1	1	1	1	1
4.3b	1	1	1	1	1
4.3c	1	1	1	1	1
4.4a	1	1	1	1	1
4.4b	1	1	1	1	1
4.5a	1	1	1	1	1
4.5b	1	1	1	1	1
4.5c	1	1	1	1	1
4.6a	1	1	1	1	1
4.6b	1	1	1	1	1
4.6c	1	1	1	1	1
4.7a	1	1	1	1	1
4.7b	1	1	1	1	1
4.8a	1	1	1	1	1
4.8b	1	1	1	1	1
4.8c	0.687	0.815	0.815	0.815	0.815
4.8d	0.578	0.785	0.785	0.785	0.785
4.8e	1	1	1	1	1
4.8f	1	1	1	1	1
4.8g	1	1	1	1	1
4.8h	1	1	1	1	1
4.9a	1	1	1	1	1
4.9b	1	1	1	1	1
4.9c	1	1	1	1	1

Tabel 4.11 menunjukkan hasil pengukuran *similarity* untuk algoritma *inductive miner infrequent & all operators* dengan berbagai nilai threshold (0, 20, 50, 80, 100). Dari data terlihat bahwa semua *threshold* menghasilkan jumlah nilai *similarity* sempurna (nilai = 1) yang sama, menunjukkan bahwa algoritma ini cukup stabil dalam menjaga struktur model yang serupa di berbagai kondisi *threshold*.



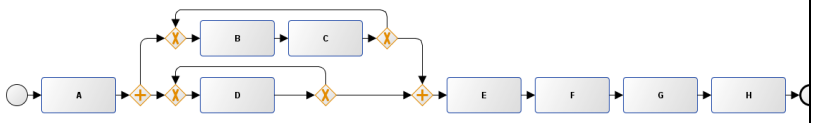
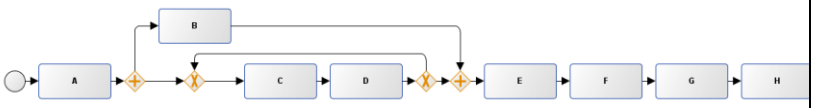
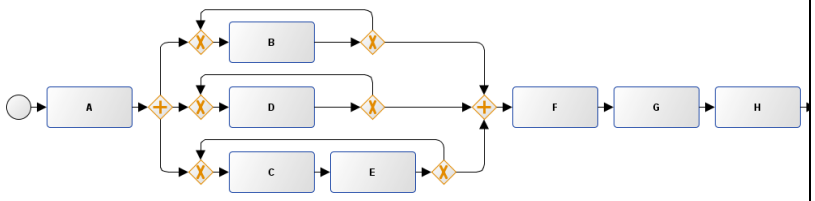
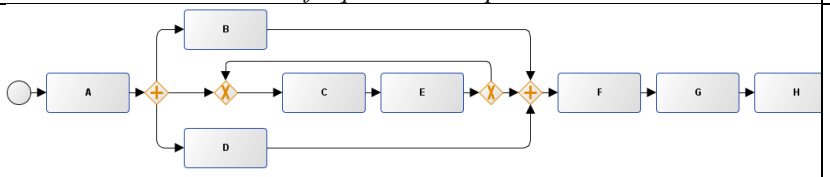
Gambar 4. 13 Grafik Presentase *Similarity IM Infrequent & All Operators*

Namun, jika dilihat dari rata-rata *similarity*, *threshold* 20, 50, 80, dan 100 memberikan hasil yang lebih tinggi, yaitu sebesar 98,62%, dibandingkan dengan *threshold* 0 yang memiliki rata-rata *similarity* 97,46%. Hal ini menunjukkan bahwa meskipun jumlah nilai sempurna sama, kualitas keseluruhan model lebih terjaga pada *threshold* 20 ke atas.

Dengan demikian, dapat disimpulkan bahwa meskipun semua *threshold* menghasilkan jumlah *similarity* = 1 yang sama, *threshold* 20, 50, 80, dan 100 lebih optimal dari sisi konsistensi dan kualitas model secara keseluruhan, dibandingkan *threshold* 0 yang sedikit lebih rendah dalam hal rata-rata *similarity*. Berikut adalah beberapa model proses bisnis hasil *discovery inductive miner infrequent & all*

operators dengan beberapa pengujian *threshold* yang memiliki nilai *similarity* di bawah 1:

Tabel 4. 12 Model Proses Bisnis yang Memiliki Nilai *Similarity* < 1

Dataset	Hasil	Similarity
4.8c	 *Inductive Miner Infrequent & All Operators threshold 0*	0.687
4.8c	 *Inductive Miner Infrequent & All Operators threshold 20, 50, 80, 100*	0.815
4.8d	 *Inductive Miner Infrequent & All Operators threshold 0*	0.578
4.8d	 *Inductive Miner Infrequent & All Operators threshold 20, 50, 80, 100*	0.785

Dari Tabel 4.12 dapat dilihat bahwa *inductive miner infrequent & all operators* tidak mampu menghasilkan proses bisnis yang sama pada BPMN Gambar 4.8c, dan 4.8d Gambar 4.8c, dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 43.942 dan 45.857.Dari pengujian *threshold* pada algoritma *inductive miner infrequent & all operators,* dilihat bahwa persentase yang lebih tinggi dihasilkan ketika nilai *threshold* dinaikkan, menunjukkan bahwa algoritma mampu

menangkap kompleksitas yang lebih besar dalam model proses bisnis seiring dengan peningkatan *threshold*.

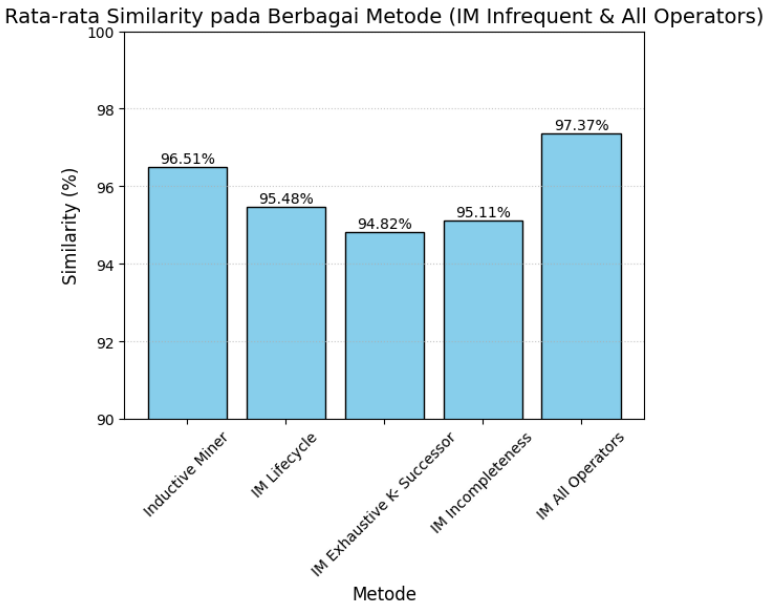
Varian *Inductive Miner Infrequent & All Operator* memiliki kemampuan untuk menangani pola OR karena dilengkapi dengan *cutfinder* yang memadai, seperti *CutFinderIMfa*, yang secara khusus dirancang untuk menangani aktivitas yang dapat terjadi di jalur alternatif secara bersamaan. *Cutfinder* ini memungkinkan pemisahan aktivitas berdasarkan pola OR, yang memungkinkan algoritma untuk membagi *log* dengan tepat saat ada pilihan jalur yang saling bersifat opsional. Meskipun demikian, seperti varian lainnya, *inductive miner infrequent all operators* masih mengalami kesulitan dalam menangani pola *loop* karena keterbatasan dalam mengelola siklus berulang. Berikut adalah hasil *similarity* dari beberapa varian *inductive miner*.

Tabel 4. 13 Hasil Pengukuran *Similarity* Skenario 2

Dataset	***Similarity***				
	Inductive Miner	***IM Lifecycle***	***IM Exhaustive K-Sucsessor***	***IM Incompletness***	***IM All Operators***
4.1a	1	1	1	1	1
4.1b	1	1	1	1	1
4.1c	1	1	1	1	1
4.2a	1	1	1	1	1
4.2b	1	1	1	1	1
4.3a	1	1	1	1	1
4.3b	1	1	1	1	1
4.3c	1	1	1	1	1
4.4a	1	1	1	1	1
4.4b	1	1	1	1	1
4.5a	1	1	1	1	1
4.5b	1	1	1	1	1
4.5c	1	1	1	1	1
4.6a	1	1	1	1	1
4.6b	1	1	1	0.857	1
4.6c	0.965	1	0.857	0.928	1
4.7a	1	1	1	1	1
4.7b	0.424	0.424	0.424	0.424	1
4.8a	1	1	0.814	1	1
4.8b	1	1	0.833	1	1
4.8c	0.815	0.687	0.800	0.687	0.687
4.8d	0.785	0.578	0.769	0.687	0.550

4.8e	1	1	1	1	1
4.8f	1	1	1	1	1
4.8g	1	1	1	1	1
4.8h	1	1	1	1	1
4.9a	1	1	1	1	1
4.9b	1	1	1	1	1
4.9c	1	1	1	1	1

Pada Tabel 4.13 diketahui bahwa terdapat perbedaan *similarity* pada setiap data. setelah melakukan perhitungan *similarity*, didapatkan bahwa nilai *similarity* antara dua model proses mendekati atau sama dengan 1, maka semakin mirip pula kedua model proses tersebut. Sebaliknya, ketika nilai *similarity* antara dua model proses dibawah 1, maka semakin rendah tingkat kemiripan kedua model proses. Data dengan nilai *similarity* 1 merupakan model proses dari hasil *discovery* yang cocok dengan model proses pada dataset. Sedangkan, model proses dengan nilai *similarity* dibawah 1 merupakan model proses dari hasil *discovery* yang tidak cocok dengan dataset. Berikut adalah presentase nilai *similarity* pada varian *inductive miner*.



Gambar 4. 14 Grafik Presentase *Similarity Varian Inductive Miner*

Berdasarkan grafik yang diberikan, *inductive miner* memiliki nilai *similarity* 96.51%, menunjukkan bahwa model proses bisnis yang dihasilkan lebih sesuai dengan dataset yang diuji. Meskipun memiliki *similarity* yang tinggi, *inductive miner* tetap menghasilkan 4 model proses bisnis dengan nilai *similarity* di bawah 1, yang menunjukkan adanya beberapa ketidakcocokan antara model yang dihasilkan dan dataset.

Inductive miner tidak mampu menghasilkan proses bisnis pada BPMN Gambar 4.6c, 4.7b, 4.8c, dan 4.8d. *Inductive miner* kurang mampu dalam menangani dataset dengan karakteristik variasi kedalaman cabang AND yang berbeda yaitu Gambar 4.6c dengan nilai kompleksitas 90.000. Gambar 4.7b melibatkan variasi percabangan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.8c dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 43.942 dan 45.857. Hal ini terjadi karena kode algoritma *inductive Miner* dalam aplikasi ProM tidak mampu menangani dataset tersebut.

Inductive Miner gagal dalam menangani dataset dengan variasi kedalaman yang berbeda karena *CutFinders* yang digunakan memiliki keterbatasan dalam menangani struktur proses yang sangat dalam atau bercabang. *CutFinders* berfungsi untuk membagi *log* menjadi *sublog* berdasarkan pola aktivitas. Namun, pada struktur yang sangat dalam, bercabang, atau kompleks, seperti *loop* yang dalam atau percabangan XOR, algoritma ini kesulitan menemukan pemotongan yang valid.

Selain itu, *inductive miner* gagal dalam menangani dataset dengan pola OR karena tidak memiliki *CutFinder* OR yang secara eksplisit menangani pola OR dalam proses. Pola OR memungkinkan aktivitas yang dapat terjadi dalam beberapa jalur berbeda, dan tanpa CutFinder yang dapat menangani kasus ini. *inductive miner* akan mencoba menangani pola tersebut dengan XOR atau AND, yang tidak cocok untuk percabangan alternatif. Dalam hal ini, *inductive miner* menganggap aktivitas sebagai percabangan eksklusif (XOR) atau paralel (AND), yang mengarah pada pemodelan yang salah dan kesalahan dalam memisahkan *log*. Hal ini menyebabkan struktur proses yang tidak akurat, jalur yang seharusnya bersifat alternatif dianggap sebagai bagian dari percabangan yang tidak saling tergantung atau paralel.

Inductive Miner Lifecycle memiliki nilai *similarity* 95.48%, sedikit lebih rendah dibandingkan dengan *inductive miner. Inductive miner lifecycle* menghasilkan 3 model dengan nilai *similarity* di bawah 1, yang menunjukkan bahwa meskipun algoritma ini juga menghasilkan model yang cukup akurat, ada beberapa model yang tidak sepenuhnya cocok dengan dataset. *Inductive miner lifecycle* lebih fokus pada pemodelan siklus hidup suatu proses, dengan fokus pada transisi antar tahap dalam setiap kasus individu.

Inductive miner lifecycle tidak mampu menghasilkan proses bisnis pada BPMN Gambar 4.7b, 4.8c, dan 4.8d. *Inductive miner* kurang mampu dalam menangani dataset dengan karakteristik melibatkan variasi percabangan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400, yaitu Gambar 4.7b. Gambar 4.8c dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai

kompleksitas berturut-turut 43.942 dan 45.857. Hal ini terjadi karena kode algoritma *inductive miner lifecycle* dalam aplikasi ProM tidak mampu menangani dataset tersebut.

Inductive miner lifecycle memiliki kemampuan lebih baik dalam menangani percabangan AND dengan variasi kedalaman dibandingkan varian lain karena perbedaan dalam cara mereka menangani pemisahan *log* dan pengolahan aktivitas *infrequent*. Hal ini disebabkan karena varian ini memiliki LogSplitterIMlc dan *CutFinder* yang memadai.berikut adalah *pseudocode* LogSplitterIMlc*:*

```
class LogSplitterIMi implements LogSplitter:

private logSplitterIMi: LogSplitter = new LogSplitterIMi()

function split(log: IMLLog, logInfo: IMLLogInfo, cut: Cut,
minerState: MinerState): LogSplitResult:

    result = logSplitterIMi.split(log, logInfo, cut, minerState)

    if minerState.parameters.isRepairLifeCycle() AND
(cut.operator != Operator.xor AND cut.operator !=
Operator.concurrent):

        newSublogs = empty list of IMLlog

        for each sublog in result.sublogs:

newSublogs.append(LifeCycle(minerState.parameters.isDebug()).pre
ProcessLog(sublog))

        result.sublogs = newSublogs
```

LogSplitterIMlc yang digunakan di *inductive miner lifecycle* menyediakan metode untuk memperbaiki *log* dengan siklus kehidupan yang rusak atau tidak konsisten, meningkatkan keakuratan dalam pemisahan *sublogs* pada percabangan AND dengan variasi kedalaman. Ketika ada percabangan AND yang lebih dalam,

proses reparasi siklus ini memperbaiki *log* sehingga aktivitas-aktivitas yang saling terhubung dapat dipisahkan dengan benar.

Inductive miner exhaustive k-successor memiliki nilai *similarity* 94.82%, yang lebih rendah dari kedua metode sebelumnya. *Inductive miner exhaustive k-successor* menghasilkan 6 model dengan nilai *similarity* di bawah 1, yang lebih banyak dibandingkan dengan *inductive miner* dan *inductive miner lifecycle*. Ini menunjukkan bahwa algoritma ini kurang efektif dalam menangani dataset tertentu. *Inductive miner* tidak mampu menghasilkan proses bisnis pada BPMN Gambar 4.6c, 4.7b, 4.8c, dan 4.8d. *Inductive miner exhaustive k-successor* kurang mampu dalam menangani dataset dengan karakteristik variasi kedalaman cabang AND yang berbeda yaitu Gambar 4.6c dengan nilai kompleksitas 90.000. Gambar 4.7b melibatkan variasi percabangan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.8a, 4.8b, 4.8c, dan4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 40.023, 42.000, 43.942 dan 45.857. Hal ini terjadi karena kode algoritma *inductive miner exhaustive k-successor* dalam aplikasi ProM tidak mampu menangani dataset tersebut.

```
function FindCut(log: IMLog; logInfo: IMLLogInfo; minerState:
MinerState): Cut;
var
  kSuccessor: UpToKSuccessorMatrix;
  exhaustive: Exhaustive;
  r: Result;
begin
  { Membuat k-Successor Matrix dari log dan informasi log }
  kSuccessor := UpToKSuccessor.fromLog(log, logInfo);

  { Membuat objek Exhaustive dengan log, informasi log, k-
Successor Matrix, dan minerState }
  exhaustive := new Exhaustive(log, logInfo, kSuccessor,
minerState);
  { Melakukan pencarian exhaustif untuk menemukan cut }
```

```
  r := exhaustive.tryAll();
  { Mengembalikan cut yang ditemukan }
  result := r.cut;
end;
```

Inductive miner exhaustive k-successor menghasilkan model yang berbeda dari varian lainnya karena menggunakan pendekatan CutFinder yang berbeda. *Inductive miner exhaustive k-successor* mengandalkan pencarian *exhaustif* melalui *k-Successor matrix*, yang mencoba semua kemungkinan hubungan antar aktivitas untuk menemukan *cut* yang optimal, sementara varian lainnya, seperti *inductive miner*, menggunakan strategi pemisahan yang lebih sederhana dan langsung.

Inductive miner incompleteness memiliki nilai similarity 95.11%, sedikit lebih rendah dari *inductive miner lifecycle*. Algoritma ini menghasilkan 5 model dengan nilai *similarity* di bawah 1, yang menunjukkan adanya ketidakcocokan yang cukup signifikan pada beberapa model yang dihasilkan, meskipun secara keseluruhan masih menghasilkan hasil yang cukup baik.

Inductive miner incompleteness tidak mampu menghasilkan proses bisnis pada BPMN Gambar 4.6b, 4.6c, 4.7b, 4.8c, dan 4.8d. *Inductive miner incompleteness* kurang mampu dalam menangani dataset dengan karakteristik variasi kedalaman cabang AND yang berbeda yaitu Gambar 4.6b dan 4.6c dengan nilai kompleksitas 88.250 dan 90.000. Gambar 4.7b melibatkan variasi percabangan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.8c dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 43.942 dan 45.857. Hal ini terjadi karena kode algoritma *inductive*

miner incompleteness dalam aplikasi ProM tidak mampu menangani dataset tersebut.

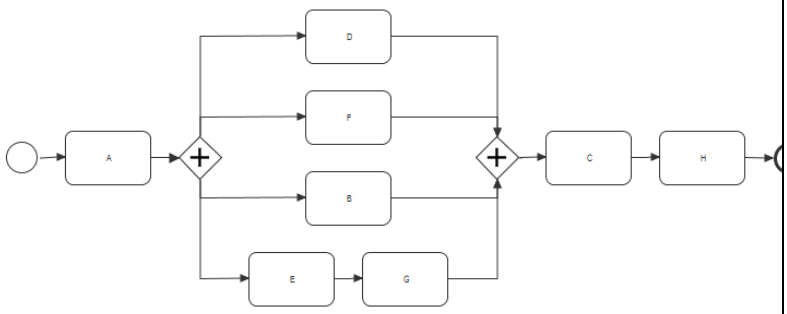
Perbedaan hasil yang terjadi pada *inductive miner incompleteness* disebabkan oleh penggunaan *CutFinder* yang berbeda. *CutFinder* yang digunakan dalam IMc menggabungkan beberapa teknik canggih, termasuk SAT *solvers* seperti SATSolveXor, SATSolveParallel, SATSolveSequence, dan SATSolveLoop untuk menangani ketidaklengkapan data dalam *log*. Pendekatan ini memungkinkan *inductive miner incompleteness* untuk menangani data yang tidak lengkap atau hilang dengan lebih efektif, serta mengidentifikasi *cut* yang optimal meskipun ada ketergantungan kompleks antar aktivitas. Berbeda dengan varian *inductive miner* lainnya, yang menggunakan pendekatan lebih sederhana dalam pemisahan *cut*, *inductive miner incompleteness* memperkenalkan algoritma yang lebih kompleks untuk mengatasi ketidaklengkapan dan masalah ketergantungan dalam dataset.

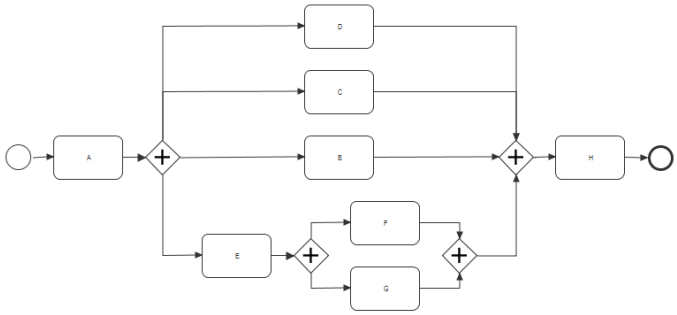
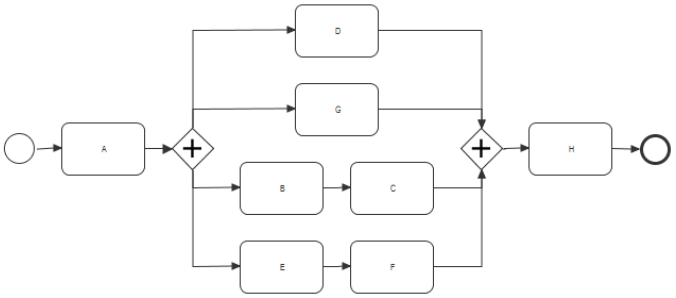
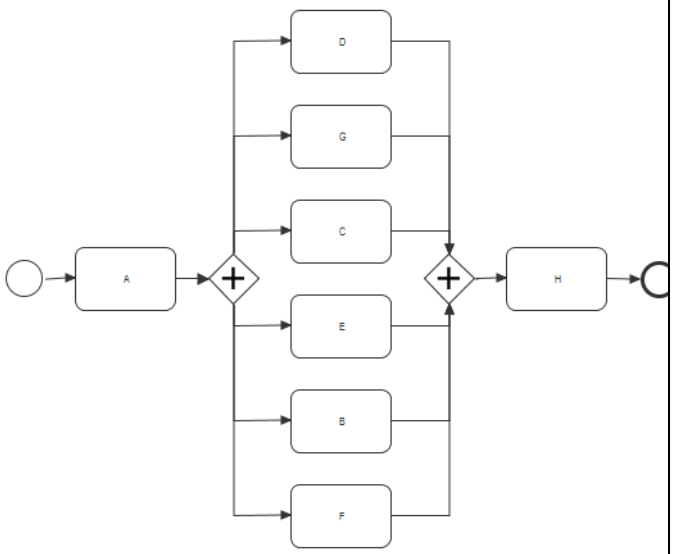
Terakhir, *Inductive Miner All Operators* memiliki nilai similarity 97.37%, yang merupakan yang tertinggi di antara semua metode. Hal ini menunjukkan *inductive miner all operators* menghasilkan model yang paling sesuai dengan dataset, namun masih menghasilkan 2 model dengan nilai *similarity* di bawah 1, meskipun jumlahnya paling sedikit di antara semua varian. Varian ini lebih baik dari pada varian yang lain, namun masih memiliki kesamaan yaitu tidak mampu menangani dataset seperti Gambar 4.8c dan 4.8d menunjukkan variasi perulangan, yang menambahkan siklus berulang dalam proses dengan nilai kompleksitas berturut-turut 43.942 dan 45.857. *Inductive miner all operators* mampu menangani OR dan kedalaman cabang AND karena kombinasi dari CutFinder yang lebih

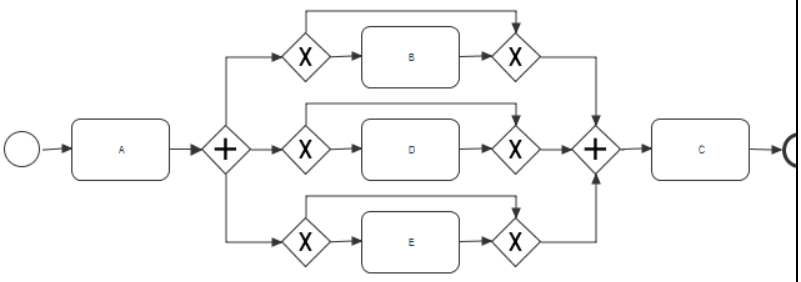
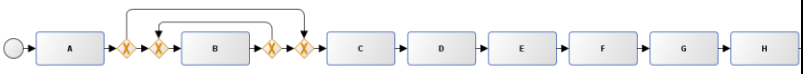
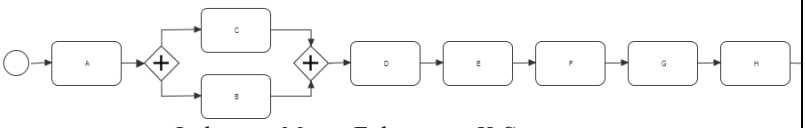
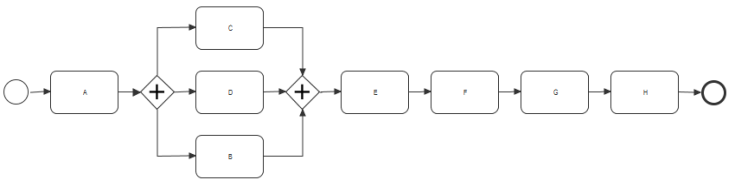
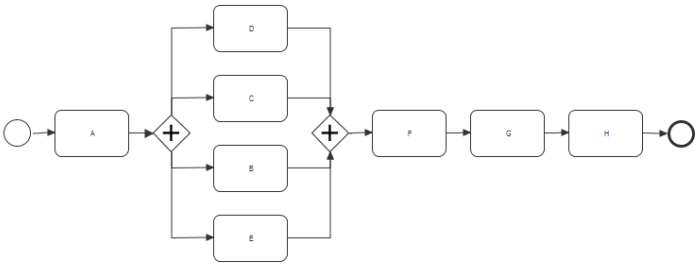
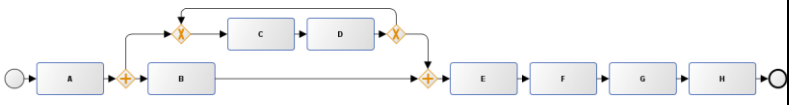
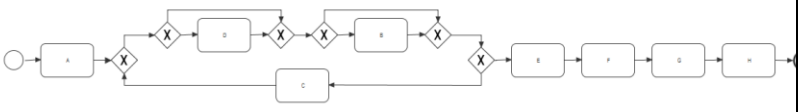
canggih dan LogSplitter yang lebih fleksibel, yang memungkinkan pemisahan *log* yang lebih dinamis dan menangani berbagai jenis struktur cabang dalam proses secara lebih efektif. Pendekatan ini memungkinkan *inductive miner all operators* untuk menangani *log* yang lebih kompleks dengan percabangan dan kedalaman yang lebih tinggi, memberikan kemampuan yang lebih baik dalam memodelkan proses yang memiliki struktur lebih rumit.

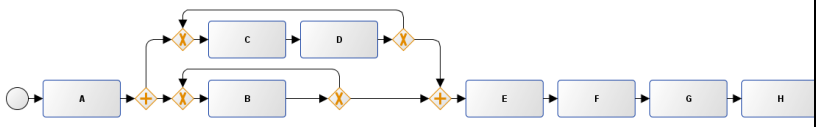
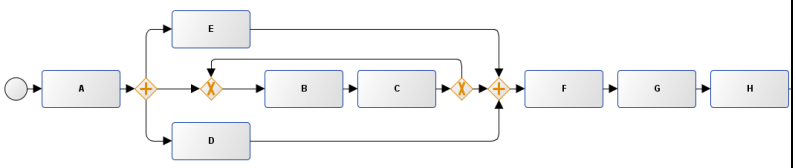
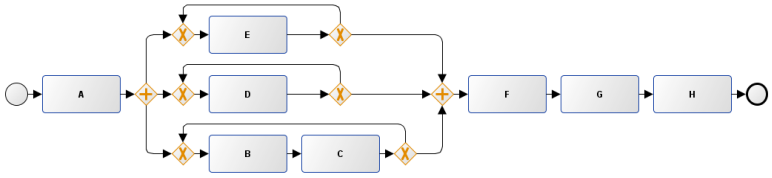
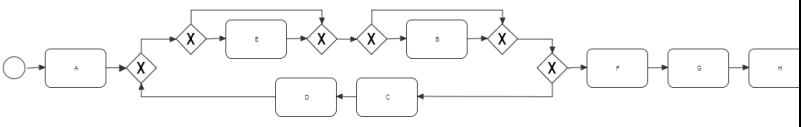
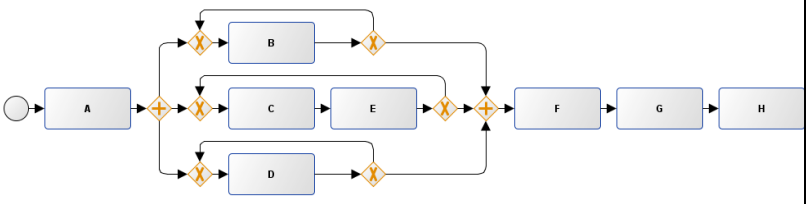
Secara keseluruhan, meskipun nilai *similarity* pada setiap metode menunjukkan hasil yang baik, masih terdapat beberapa model yang tidak sepenuhnya sesuai dengan dataset yang diuji, dengan *inductive miner exhaustive k-successor* menghasilkan model yang paling banyak dengan nilai *similarity* di bawah 1. Berikut adalah beberapa model proses bisnis hasil *discovery* yang memiliki nilai *similarity* di bawah 1, berikut adalah model proses bisnis tersebut:

Tabel 4. 14 Model Proses Bisnis yang Memiliki Nilai *Similarity* < 1

Dataset	**Hasil**	***Similarity***
4.6b	 *Inductive Miner Incompletness*	0.857

Dataset	Hasil	*Similarity*
4.6c	 *Inductive Miner*	0.965
4.6c	 *Inductive Miner Exhaustive K-Sucsessor*	0.857
4.6c	 *Inductive Miner Incompletness*	0.928

Dataset	Hasil	*Similarity*
4.7b	 *Inductive Miner Exhaustive K-Sucsessor, Inductive Miner Incompleteness*	0.424
4.8a	 *Inductive Miner Exhaustive K-Sucsessor*	0.814
4.8b	 *Inductive Miner Exhaustive K-Sucsessor*	0.833
4.8c	 *Inductive Miner Exhaustive K-Sucsessor*	0.800
4.8d	 *Inductive Miner Exhaustive K-Sucsessor*	0.769
4.8c	 *Inductive Miner*	0.815
4.8c	 *Inductive Miner Incompletness*	0.687

Dataset	Hasil	*Similarity*
4.8c	 *Inductive Miner All Operators*	0.687
4.8d	 *Inductive Miner*	0.785
4.8d	 *Inductive Miner Lifecycle*	0.578
4.8d	 *Inductive Miner Incompletness*	0.687
4.8d	 *Inductive Miner All Operators*	0.550

Berdasarkan hasil eksperimen skenario 2, model proses bisnis hasil *discovery* menggunakan 8 varian *inductive miner*, terlihat bahwa *inductive miner infrequent & all operators* menghasilkan model proses bisnis yang memiliki nilai *similarity* dengan presentase tertinggi, yaitu 98,62%. Hal ini menunjukkan bahwa varian *inductive miner infrequent & all operators* tersebut lebih mampu menangani

berbagai model proses bisnis dengan kompleksitas yang beragam. *inductive miner infrequent & all operators* memberikan hasil yang lebih optimal dalam hal kecocokan model dengan data, yang tercermin dari *nilai* similarity yang lebih tinggi daripada varian *inductive miner yang lain*.

4.5 Hasil Eksperimen Skenario 3

Skenario ketiga akan menguji pengaruh *relative-to-best threshold*, *length-one-loops*, *length-two-loops*, *long-distance,* dan *dependency threshold* yang digunakan dalam algoritma *heuristic miner* terhadap kinerja model yang dihasilkan. Setiap variasi akan diterapkan tidak hanya pada dataset yang sama tetapi juga pada berbagai jenis dataset 9 karakteristik berbeda, seperti jumlah aktivitas sekuensial, jumlah percabangan AND, jumlah aktivitas sekuensial dalam percabangan, jumlah cabang AND yang berbeda, variasi lokasi cabang AND, kedalaman percabangan AND, variasi *gateway* XOR dan OR, variasi siklus (*loop*), dan kombinasi logika percabangan dengan *looping*.

Tabel 4. 15 Hasil Pengukuran *Similarity* Uji *Relative To Best Threshold*

Dataset	***Similarity***				
	Heuristic Miner (relative to best threshold 0)	***Heuristic Miner (relative to best threshold 5)***	***Heuristic Miner (relative to best threshold 50)***	***Heuristic Miner (relative to best threshold 75)***	***Heuristic Miner (relative to best threshold 100)***
4.1a	1	1	1	1	1
4.1b	1	1	1	1	1
4.1c	1	1	1	1	1
4.2a	1	1	1	1	1
4.2b	1	1	1	1	1
4.3a	1	1	1	1	1
4.3b	1	1	1	1	1
4.3c	1	1	1	1	1
4.4a	1	1	1	1	1
4.4b	1	1	1	1	1
4.5a	1	1	1	1	1
4.5b	1	1	1	1	1
4.5c	1	1	1	1	1

4.6a	1	1	1	1	1
4.6b	0.750	0.750	0.750	0.750	0.750
4.6c	0.848	0.848	0.848	0.848	0.848
4.7a	1	1	1	1	1
4.7b	0.269	0.269	0.269	0.269	0.269
4.8a	1	1	1	1	1
4.8b	1	1	1	1	1
4.8c	1	1	1	1	1
4.8d	1	1	1	1	1
4.8e	1	1	1	1	1
4.8f	1	1	1	1	1
4.8g	1	1	1	1	1
4.8h	1	1	1	1	1
4.9a	0.800	0.800	0.800	0.800	0.800
4.9b	1	1	1	1	1
4.9c	0.777	0.777	0.777	0.777	0.777

Tabel 4.15 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *relative to best threshold*. Pada Tabel 4.15 menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Tabel 4. 16 Hasil Pengukuran *Similarity* Uji *Length One Loop*

Dataset	***Similarity***			
	Heuristic Miner (length one loop 0)	***Heuristic Miner (length one loop 50)***	***Heuristic Miner (length one loop 90)***	***Heuristic Miner (length one loop 100)***
4.1a	1	1	1	1
4.1b	1	1	1	1
4.1c	1	1	1	1
4.2a	1	1	1	1
4.2b	1	1	1	1
4.3a	1	1	1	1
4.3b	1	1	1	1
4.3c	1	1	1	1
4.4a	1	1	1	1
4.4b	1	1	1	1
4.5a	1	1	1	1
4.5b	1	1	1	1
4.5c	1	1	1	1
4.6a	1	1	1	1
4.6b	0.750	0.750	0.750	0.750
4.6c	0.848	0.848	0.848	0.848
4.7a	1	1	1	1
4.7b	0.269	0.269	0.269	0.269
4.8a	1	1	1	1
4.8b	1	1	1	1
4.8c	1	1	1	1

4.8d	1	1	1	1
4.8e	1	1	1	1
4.8f	1	1	1	1
4.8g	1	1	1	1
4.8h	1	1	1	1
4.9a	0.800	0.800	0.800	0.800
4.9b	1	1	1	1
4.9c	0.777	0.777	0.777	0.777

Tabel 4.16 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *length one loop*. Pada Tabel 4.16 menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Tabel 4. 17 Hasil Pengukuran *Similarity* Uji *Length Two Loop*

Dataset	***Similarity***			
	Heuristic Miner (length two loop 0)	***Heuristic Miner (length two loop 50)***	***Heuristic Miner (length two loop 90)***	***Heuristic Miner (length two loop 100)***
4.1a	1	1	1	1
4.1b	1	1	1	1
4.1c	1	1	1	1
4.2a	1	1	1	1
4.2b	1	1	1	1
4.3a	1	1	1	1
4.3b	1	1	1	1
4.3c	1	1	1	1
4.4a	1	1	1	1
4.4b	1	1	1	1
4.5a	1	1	1	1
4.5b	1	1	1	1
4.5c	1	1	1	1
4.6a	1	1	1	1
4.6b	0.750	0.750	0.750	0.750
4.6c	0.848	0.848	0.848	0.848
4.7a	1	1	1	1
4.7b	0.269	0.269	0.269	0.269
4.8a	1	1	1	1
4.8b	1	1	1	1
4.8c	1	1	1	1
4.8d	1	1	1	1
4.8e	1	1	1	1
4.8f	1	1	1	1
4.8g	1	1	1	1
4.8h	1	1	1	1
4.9a	0.800	0.800	0.800	0.800
4.9b	1	1	1	1
4.9c	0.777	0.777	0.777	0.777

Tabel 4.17 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *length two loop*. Pada Tabel 4.17 menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Tabel 4. 18 Hasil Pengukuran *Similarity* Uji *Long Distance*

Dataset	***Similarity***				
	Heuristic Miner (long distance 0)	***Heuristic Miner long distance 25)***	***Heuristic Miner (long distance 50)***	***Heuristic Miner (long distance 90)***	***Heuristic Miner (long distance 100)***
4.1a	1	1	1	1	1
4.1b	1	1	1	1	1
4.1c	1	1	1	1	1
4.2a	1	1	1	1	1
4.2b	1	1	1	1	1
4.3a	1	1	1	1	1
4.3b	1	1	1	1	1
4.3c	1	1	1	1	1
4.4a	1	1	1	1	1
4.4b	1	1	1	1	1
4.5a	1	1	1	1	1
4.5b	1	1	1	1	1
4.5c	1	1	1	1	1
4.6a	1	1	1	1	1
4.6b	0.750	0.750	0.750	0.750	0.750
4.6c	0.848	0.848	0.848	0.848	0.848
4.7a	1	1	1	1	1
4.7b	0.269	0.269	0.269	0.269	0.269
4.8a	1	1	1	1	1
4.8b	1	1	1	1	1
4.8c	1	1	1	1	1
4.8d	1	1	1	1	1
4.8e	1	1	1	1	1
4.8f	1	1	1	1	1
4.8g	1	1	1	1	1
4.8h	1	1	1	1	1
4.9a	0.800	0.800	0.800	0.800	0.800
4.9b	1	1	1	1	1
4.9c	0.777	0.777	0.777	0.777	0.777

Tabel 4.18 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *long distance*. Pada Tabel 4.18

menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Tabel 4. 19 Hasil Pengukuran *Similarity* Uji *Depedency*

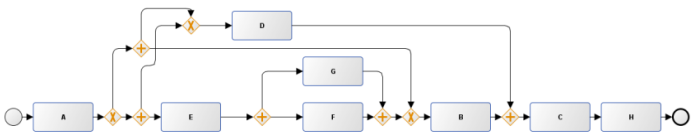
Dataset	***Similarity***				
	Heuristic Miner (Depedency 0)	***Heuristic Miner (Depedency 10)***	***Heuristic Miner (Depedency 50)***	***Heuristic Miner (Depedency 90)***	***Heuristic Miner (Depedency 100)***
4.1a	1	1	1	1	1
4.1b	1	1	1	1	1
4.1c	1	1	1	1	1
4.2a	1	1	1	1	1
4.2b	1	1	1	1	1
4.3a	1	1	1	1	1
4.3b	1	1	1	1	1
4.3c	1	1	1	1	1
4.4a	1	1	1	1	1
4.4b	1	1	1	1	1
4.5a	1	1	1	1	1
4.5b	1	1	1	1	1
4.5c	1	1	1	1	1
4.6a	1	1	1	1	1
4.6b	0.521	0.521	0.521	0.750	0.750
4.6c	0.500	0.500	0.500	0.848	0.848
4.7a	1	1	1	1	1
4.7b	0.269	0.269	0.269	0.269	0.269
4.8a	1	1	1	1	1
4.8b	1	1	1	1	1
4.8c	1	1	1	1	1
4.8d	1	1	1	1	1
4.8e	1	1	1	1	1
4.8f	1	1	1	1	1
4.8g	1	1	1	1	1
4.8h	1	1	1	1	1
4.9a	0.565	0.565	0.565	0.800	0.800
4.9b	1	1	1	1	1
4.9c	0.571	0.571	0.571	0.777	0.777

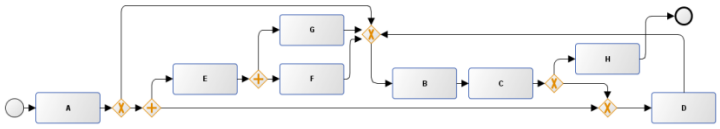
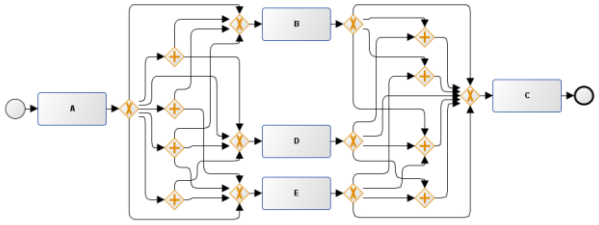
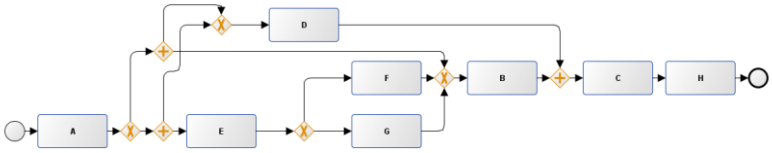
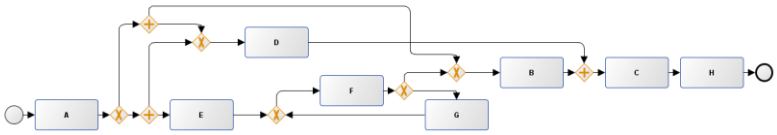
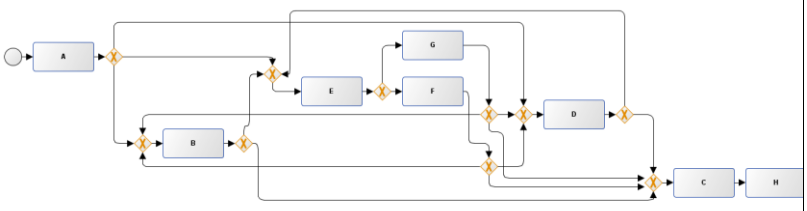
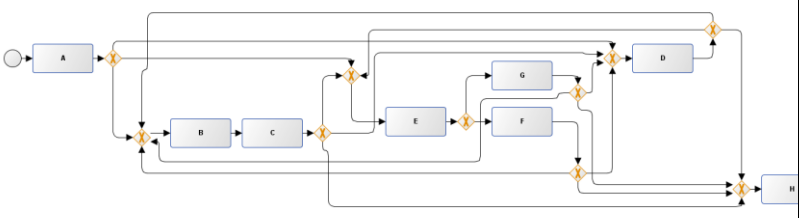
Tabel 4.19 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *long distance*. Pada Tabel 4.19 menunjukkan adanya perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini berpengaruh.

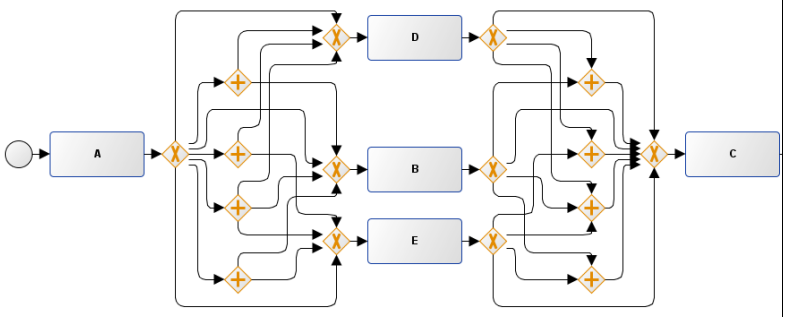
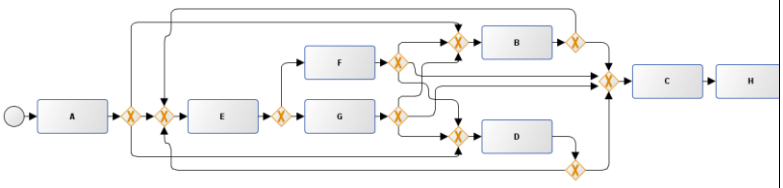
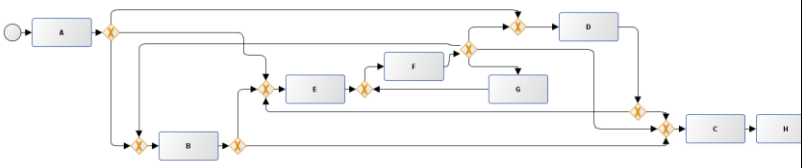
Pengujian pada parameter *dependency threshold* menunjukkan hasil yang lebih baik ketika nilai *threshold* berada di atas 90. Pada rentang nilai tersebut, tingkat kemiripan model proses yang dihasilkan dengan dataset meningkat, mengindikasikan bahwa pengaturan *dependency threshold* yang lebih tinggi memberikan hasil *discovery* yang lebih sesuai dan akurat dengan model proses pada dataset.

Dari pengujian diketahui bahwa terdapat perbedaan *similarity* pada setiap data. Setelah melakukan perhitungan *similarity*, didapatkan bahwa nilai *similarity* antara dua model proses mendekati atau sama dengan 1, maka semakin mirip pula kedua model proses tersebut. Sebaliknya, ketika nilai *similarity* antara dua model proses dibawah 1, maka semakin rendah tingkat kemiripan kedua model proses. Data dengan nilai *similarity* 1 merupakan model proses dari hasil *discovery* yang cocok dengan model proses pada dataset. Sedangkan, model proses dengan nilai *similarity* dibawah 1 merupakan model proses dari hasil *discovery* yang tidak cocok dengan dataset. Berikut adalah model proses bisnis yang memiliki nilai *similarity* di bawah 1 :

Tabel 4. 20 Model Proses Bisnis yang Memiliki Nilai *Similarity* < 1

Dataset	**Hasil**	**Similarity**
4.6b	 *Heuristic Miner* (*Depedency Treshold* 90, 100)	0.750

Dataset	Hasil	Similarity
4.6c	 *Heuristic Miner* (*Depedency Treshold* 90, 100)	0.848
4.7b	 *Heuristic Miner* (*Depedency Treshold* 90, 100)	0.269
4.9a	 *Heuristic Miner* (*Depedency Treshold* 90, 100)	0.800
4.9c	 *Heuristic Miner* (*Depedency Treshold* 90, 100)	0.777
4.6b	 *Heuristic Miner* (*Depedency Treshold* 10, 50)	0.521
4.6c	 *Heuristic Miner* (*Depedency Treshold* 10, 50)	0.500

Dataset	Hasil	Similarity
4.7b	 *Heuristic Miner* (*Depedency Treshold* 10, 50)	0.269
4.9a	 *Heuristic Miner* (*Depedency Treshold* 10, 50)	0.565
4.9c	 *Heuristic Miner* (*Depedency Treshold* 10, 50)	0.571

Berdasarkan hasil eksperimen skenario 3, dengan variasi *dependency threshold* yang berbeda, terlihat bahwa algoritma *heuristic miner* menghasilkan empat model proses bisnis yang memiliki nilai *similarity* di bawah 1 pada dataset yang sama, yaitu 4.6b, 4.6c, 4.7b, 4.9a, dan 4.9c. *Heuristic miner* kurang mampu menangani dataset dengan karakteristik AND dengan variasi kedalaman seperti 4.6b dan 4.6c dengan nilai kompleksitas 88.250 dan 90.000. Gambar 4.7b melibatkan variasi percabangan OR, yang memperkenalkan banyak jalur eksekusi yang berbeda dengan nilai kompleksitas 146.400. Gambar 4.9a dan 4.9c menunjukkan variasi dengan kombinasi gerbang logika dan perulangan dengan nilai kompleksitas berturut-turut 86.250 dan 80.917. Hal ini terjadi karena

heuristic miner kesulitan menangani ketergantungan paralel yang mendalam karena modelnya berfokus pada ketergantungan yang terjadi dalam urutan tertentu, bukan pada eksekusi paralel. Sedangkan untuk percabangan OR, yang memperkenalkan banyak jalur alternatif, *heuristic miner* lebih sulit menangkap ketergantungan antar jalur tersebut. Selain itu, kombinasi gerbang logika dan perulangan menambah kompleksitas yang tidak dapat dijelaskan hanya dengan frekuensi kejadian atau ketergantungan linear. Dengan kompleksitas yang tinggi, *heuristic miner* tidak dapat menangani hubungan yang lebih rumit yang melibatkan pilihan alternatif dan *looping* secara efektif, karena lebih berfokus pada pola ketergantungan langsung dan sederhana.

Namun, ketika dibandingkan antara variasi *dependency threshold*, *heuristic miner* dengan *dependency threshold* di atas 90 menunjukkan nilai *similarity* yang paling mendekati 1 pada sebagian besar model proses bisnis, dibandingkan dengan *threshold* di bawah 90. Hal ini menunjukkan bahwa *heuristic miner* dengan *dependency threshold* di atas 90 menghasilkan model yang lebih akurat dan lebih cocok dengan dataset yang digunakan.

Nilai *similarity* yang lebih tinggi mendekati 1 mengindikasikan bahwa model proses yang dihasilkan lebih sesuai dengan alur proses yang ada pada data, sehingga semakin mendekati 1, semakin baik kesesuaian model proses tersebut dengan data yang ada. Sebaliknya, nilai *similarity* yang lebih rendah menunjukkan ketidakcocokan antara model yang dihasilkan dengan dataset yang digunakan, yang berarti model tersebut kurang representatif atau kurang optimal dalam merepresentasikan alur proses yang sebenarnya.

Dengan demikian, dapat disimpulkan bahwa penggunaan *dependency threshold* di atas 90 memberikan hasil yang lebih baik dalam hal kesesuaian model proses bisnis, sehingga dapat dikatakan bahwa *threshold* ini memberikan kinerja yang lebih optimal dalam eksperimen ini dibandingkan dengan *threshold* yang lebih rendah.

4.6 Hasil Eksperimen Skenario 4

Skenario keempat akan menguji kinerja kedua algoritma dengan mengeksplorasi parameter dan varian algoritma. Dalam skenario ini, *event logs* yang digunakan adalah *event logs* dengan karakteristik *nested loops* yang berarti sebuah *loop* dapat memiliki satu atau lebih *loop* yang terdapat di dalamnya, yang masing-masing akan dieksekusi secara berulang untuk setiap iterasi dari *loop* luar. skenario ini diperlukan untuk memastikan bahwa dapat menangani kompleksitas dunia nyata yang sering kali mengandung *nested loops* dan pola pengulangan yang saling terkait. Dataset yang digunakan adalah 4.10a sampai 4.10j. Berikut adalah hasil eksperimen tersebut.

Tabel 4. 21 Hasil Pengukuran *Similarity* Berbagai Varian *Inductive Miner*

Dataset	***Similarity***				
	Inductive Miner	***IM Lifecycle***	***IM Exhaustive K-Sucsessor***	***IM Incompletness***	***IM All Operators***
4.10a	0,520	0,781	0,781	0,923	0,520
4.10b	0,625	0,625	0,625	0,576	0,625
4.10c	0,480	0,657	0,657	0,694	0,480
4.10d	0,555	0,555	0,555	0,568	0,555
4.10e	0,520	0,520	0,625	0,625	0,520
4.10f	0,462	0,462	0,462	0,531	0,462
4.10g	0,625	0,625	0,675	0,675	0,625
4.10h	0,613	0,613	0,666	0,576	0,571
4.10i	0,581	0,581	0,581	0,531	0,581
4.10j	0,613	0,613	0,613	0,729	0,675

Tabel 4.21 adalah hasil *similarity* dari beberapa varian *inductive miner,* terlihat dari Tabel 4.21 bahwa *inductive miner* tidak mampu menangani *nested loops.* Hal ini terlihat dari hasil *similarity* semua varian *inductive miner* tidak ada yang memiliki hasil 1 sama sekali yang menunjukkan proses bisnis hasil *process mining* menggunakan *inductive miner* tidak memliki kesamaan dengan proses bisnis acuan.

Inductive miner tidak mampu menangani *nested loops* secara efektif karena keterbatasan dalam mengelola dependensi berlapis, struktur perulangan bersyarat, dan modeling yang terlalu sederhana untuk menangani kompleksitas yang terjadi dalam perulangan bersarang. Sementara algoritma ini sangat kuat dalam menangani *log* dengan struktur sekuensial dan percabangan OR yang lebih sederhana, kompleksitas *nested loops* yang melibatkan ketergantungan dinamis antar lapisan *loop* membuatnya kesulitan untuk menghasilkan model yang akurat. Hal ini terlihat dari kode berikut.

```
iterator := splitResult.sublogs.iterator;
firstSublog := iterator.next();
begin
    firstChild := mineNode(firstSublog, tree, minerState);
    addChild(newNode, firstChild, minerState);
end;
if (splitResult.sublogs.size > 2) then
begin
    redoXor := new Xor("");
    addNode(tree, redoXor);
    addChild(newNode, redoXor, minerState);
end
else
begin
    redoXor := newNode;
end;
while (iterator.hasNext()) do
begin
    sublog := iterator.next();
    child := mineNode(sublog, tree, minerState);
    addChild(redoXor, child, minerState);
end;
```

```
begin
    tau := new AbstractTask.Automatic("tau");
    addNode(tree, tau);
    addChild(newNode, tau, minerState);
end;
```

Pada bagian kode ini, terlihat bagaimana *loop* diidentifikasi dan diproses dengan cara tertentu. Ketika sebuah *cut* (pemotongan) ditemukan, *inductive miner* akan menangani *loop* dengan cara yang berbeda dibandingkan dengan jenis cut lainnya (misalnya, sekuens atau OR). Dalam kode, ketika operator *cut* adalah *loop*, penanganannya dilakukan dengan cara terpisah. Ada bagian khusus untuk menangani *loop*, ketika *sublog* pertama kali diekstrak, dan kemudian "redo" dilakukan dengan menggunakan XOR. *Nested loops* memerlukan pemahaman yang lebih kompleks tentang ketergantungan antara *loop* dalam dan *loop* luar. *Inductive miner* hanya menangani *loop* tunggal dan menerapkan penanganan khusus untuk *operator loop*. Namun, untuk *nested loops*, satu *loop* berada di dalam *loop* lainnya, algoritma ini tidak dapat mengelola ketergantungan yang kompleks antara lapisan-lapisan perulangan tersebut.

Tabel 4. 22 Hasil *Similarity Inductive Miner Infrequent*

Dataset	***Similarity***				
	IM Infrequent (0)	***IM Infrequent (20)***	***IM Infrequent (50)***	***IM Infrequent (80)***	***IM Infrequent (100)***
4.10a	0,520	0,520	0,925	0,925	0,925
4.10b	0,625	0,625	0.903	0.903	0.903
4.10c	0,480	0,781	0,961	0,961	0,961
4.10d	0,555	0,666	0,875	0,875	0,875
4.10e	0,520	0,641	0,925	0,925	0,925
4.10f	0,462	0,555	0,892	0,892	0,892
4.10g	0,675	0,757	0,961	0,923	0,923
4.10h	0,612	0,750	0,933	0,933	0,933
4.10i	0,581	0,735	0,925	0,888	0,888
4.10j	0,613	0,675	0,900	0,888	0,888

Tabel 4.22 menunjukkan hasil pengukuran *similarity* hasil *process mining* menggunakan *inductive miner infrequent.* Dari Tabel 4.22 terlihat bahwa nilai *similarity* mengalami peningkatan nilai ketika pengaturan *threshold* semakin tinggi.

Tabel 4. 23 Hasil *Similarity Inductive Miner Infrequent & Lifecycle*

Dataset	***Similarity***				
	IM Infrequent & Lifecycle (0)	***IM Infrequent & Lifecycle (20)***	***IM Infrequent & Lifecycle (50)***	***IM Infrequent & Lifecycle (80)***	***IM Infrequent & Lifecycle (100)***
4.10a	0,520	0,520	0,925	0,925	0,925
4.10b	0,625	0,717	0.903	0.903	0.903
4.10c	0,480	0,781	0,961	0,961	0,961
4.10d	0,555	0,666	0,875	0,875	0,875
4.10e	0,520	0,641	0,925	0,925	0,925
4.10f	0,462	0,555	0,892	0,892	0,892
4.10g	0,675	0,892	0,961	0,923	0,923
4.10h	0,612	0,750	0,933	0,933	0,933
4.10i	0,581	0,735	0,925	0,888	0,888
4.10j	0,613	0,675	0,900	0,888	0,888

Tabel 4.23 menunjukkan hasil pengukuran *similarity* hasil *process mining* menggunakan *inductive miner infrequent & lifecycle.* Dari Tabel 4.23 terlihat bahwa nilai *similarity* mengalami peningkatan nilai ketika pengaturan *threshold* semakin tinggi.

Tabel 4. 24 Hasil *Similarity Inductive Miner Infrequent* dan *All Operators*

Dataset	***Similarity***				
	IM Infrequent & All Operators (0)	***IM Infrequent & All Operators (20)***	***IM Infrequent & All Operators (50)***	***IM Infrequent & All Operators (80)***	***IM Infrequent & All Operators (100)***
4.10a	0,625	0,925	0,925	0,925	0,925
4.10b	0,625	0,717	0.903	0.903	0.903
4.10c	0,480	0,781	0,961	0,961	0,961
4.10d	0,555	0,666	0,875	0,875	0,875
4.10e	0,520	0,641	0,925	0,925	0,925
4.10f	0,462	0,555	0,892	0,892	0,892
4.10g	0,757	0,892	0,961	0,923	0,923
4.10h	0,613	0,750	0,933	0,933	0,933
4.10i	0,581	0,735	0,925	0,888	0,888
4.10j	0,675	0,613	0,900	0,888	0,888

Tabel 4.24 menunjukkan hasil pengukuran *similarity* hasil *process mining* menggunakan *inductive miner infrequent & all operators.* Dari Tabel 4.24 terlihat bahwa nilai *similarity* mengalami peningkatan nilai ketika pengaturan *threshold* semakin tinggi walaupun belum ada yang mencapai nilai 1. Ketika *threshold* dinaikkan dalam algoritma *inductive miner*, model yang dihasilkan cenderung menjadi lebih selektif dalam memilih pola-pola atau hubungan yang dianggap signifikan dalam data. *Threshold* berfungsi sebagai batas minimum seberapa sering suatu hubungan atau pola harus terjadi agar bisa dimasukkan ke dalam model. Dengan menaikkan *threshold*, algoritma akan mengabaikan hubungan yang jarang atau tidak cukup kuat, sementara lebih fokus pada pola-pola yang lebih sering muncul dalam data. Hal ini menyebabkan model yang dihasilkan lebih terstruktur dan stabil, dengan lebih banyak fokus pada pola yang dominan. Akibatnya, model akan lebih mirip dengan data asli, karena pola-pola yang lebih sering teridentifikasi dan lebih relevan, sehingga meningkatkan *similarity* antara model yang dihasilkan dan dataset. Oleh karena itu, semakin tinggi threshold, semakin tinggi pula similarity yang tercapai, karena model hanya mencakup hubungan yang lebih sering dan lebih mencerminkan pola utama dalam data seperti pada kode berikut.

```
public class LogSplitterIMlc implements LogSplitter {
    private static final LogSplitter logSplitterIMi = new
LogSplitterIMi();
    public LogSplitResult split(IMLog log, IMLogInfo logInfo,
Cut cut, MinerState minerState) {

        LogSplitResult result = logSplitterIMi.split(log,
logInfo, cut, minerState);
        if (minerState.parameters.isRepairLifeCycle() &&
cut.getOperator() != Operator.xor
                && cut.getOperator() != Operator.concurrent) {
            List<IMLog> newSublogs = new ArrayList<>();
            for (IMLog sublog : result.sublogs) {
```

```
                newSublogs.add(new
LifeCycles(minerState.parameters.isDebug()).preProcessLog(sublog
));
            }
            result.sublogs = newSublogs;
        }
        return result;
    }
}
```

LogSplitterIMlc berfungsi untuk membagi *log* berdasarkan *cut* yang ditemukan, yang kemudian dipengaruhi oleh *threshold*. Ketika *threshold* dinaikkan, algoritma akan lebih selektif dengan mengabaikan hubungan yang jarang atau tidak konsisten. Dalam hal ini, LogSplitterIMlc akan memfilter *log* dan memperbaiki *trace* yang tidak konsisten, khususnya dalam kasus *sequence* dan *loop*. Dengan demikian, model yang dihasilkan hanya akan menyertakan pola-pola yang lebih sering terjadi dan lebih stabil. Peningkatan *threshold* ini menyebabkan model lebih fokus pada hubungan-hubungan yang dominan dan lebih terlihat dalam data, sehingga meningkatkan *similarity* antara model yang dihasilkan dan dataset asli.

Berikut adalah hasil *process mining* menggunakan *heuristic miner:*

Tabel 4. 25 Hasil Pengukuran *Similarity* Uji *Relative To Best Threshold*

Dataset	***Similarity***				
	Heuristic Miner (relative to best threshold 0)	***Heuristic Miner (relative to best threshold 5)***	***Heuristic Miner (relative to best threshold 50)***	***Heuristic Miner (relative to best threshold 75)***	***Heuristic Miner (relative to best threshold 100)***
4.10a	0,880	0,880	0,880	0,880	0,880
4.10b	0,733	0,733	0,733	0,733	0,733
4.10c	1	1	1	1	1
4.10d	0,900	0,900	0,900	0,900	0,900
4.10e	0,880	0,880	0,880	0,880	0,880
4.10f	0,880	0,880	0,880	0,880	0,880
4.10g	1	1	1	1	1
4.10h	0,900	0,900	0,900	0,900	0,900
4.10i	1	1	1	1	1
4.10j	1	1	1	1	1

Tabel 4.25 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *relative to best*. Pada Tabel 4.25 menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Tabel 4. 26 Hasil Pengukuran *Similarity* Uji *Length One Loop*

Dataset	***Similarity***			
	Heuristic Miner (length one loop 0)	***Heuristic Miner (length one loop 50)***	***Heuristic Miner (length one loop 90)***	***Heuristic Miner (length one loop 100)***
4.10a	1	1	0,880	0,880
4.10b	0,900	0,900	0,733	0,733
4.10c	1	1	1	1
4.10d	0,900	0,900	0,900	0,900
4.10e	0,880	0,880	0,880	0,880
4.10f	0,880	0,880	0,880	0,880
4.10g	1	1	1	1
4.10h	0,900	0,900	0,900	0,900
4.10i	1	1	1	1
4.10j	1	1	1	1

Tabel 4.26 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *length one loop*. Pada Tabel 4.26 menunjukkan adanya perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini berpengaruh.

Tabel 4. 27 Hasil Pengukuran *Similarity* Uji *Length Two Loop*

Dataset	***Similarity***			
	Heuristic Miner (length two loop 0)	***Heuristic Miner (length two loop 50)***	***Heuristic Miner (length two loop 90)***	***Heuristic Miner (length two loop 100)***
4.10a	0,880	0,880	0,880	0,880
4.10b	0,733	0,733	0,733	0,733
4.10c	1	1	1	1
4.10d	0,900	0,900	0,900	0,900
4.10e	0,880	0,880	0,880	0,880
4.10f	0,880	0,880	0,880	0,880
4.10g	1	1	1	1
4.10h	0,900	0,900	0,900	0,900
4.10i	1	1	1	1
4.10j	1	1	1	1

Tabel 4.27 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *length two loop*. Pada Tabel 4.27 menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Tabel 4. 28 Hasil Pengukuran *Similarity* Uji *Long Distance*

Dataset	***Similarity***				
	Heuristic Miner (long distance 0)	***Heuristic Miner long distance 25)***	***Heuristic Miner (long distance 50)***	***Heuristic Miner (long distance 90)***	***Heuristic Miner (long distance 100)***
4.10a	0,880	0,880	0,880	0,880	0,880
4.10b	0,733	0,733	0,733	0,733	0,733
4.10c	1	1	1	1	1
4.10d	0,900	0,900	0,900	0,900	0,900
4.10e	0,880	0,880	0,880	0,880	0,880
4.10f	0,880	0,880	0,880	0,880	0,880
4.10g	1	1	1	1	1
4.10h	0,900	0,900	0,900	0,900	0,900
4.10i	1	1	1	1	1
4.10j	1	1	1	1	1

Tabel 4.28 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *uji long distance*. Pada Tabel 4.28 menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Tabel 4. 29 Hasil Pengukuran *Similarity* Uji *Depedency*

Dataset	***Similarity***				
	Heuristic Miner (Depedency 0)	***Heuristic Miner (Depedency 10)***	***Heuristic Miner (Depedency 50)***	***Heuristic Miner (Depedency 90)***	***Heuristic Miner (Depedency 100)***
4.10a	0,880	0,880	0,880	0,880	0,880
4.10b	0,733	0,733	0,733	0,733	0,733
4.10c	1	1	1	1	1
4.10d	0,900	0,900	0,900	0,900	0,900
4.10e	0,880	0,880	0,880	0,880	0,880
4.10f	0,880	0,880	0,880	0,880	0,880
4.10g	1	1	1	1	1
4.10h	0,900	0,900	0,900	0,900	0,900
4.10i	1	1	1	1	1
4.10j	1	1	1	1	1

Tabel 4.29 menunjukkan hasil *similarity* hasil *process mining* menggunakan *heuristic miner* dengan pengujian parameter *depedency*. Pada Tabel 4.29 menunjukkan tidak ada perubahan nilai *similarity* pada beberapa pengaturan *threshold,* yang berarti parameter ini tidak berpengaruh.

Dari hasil eksperimen nilai *similarity* mengalami kenaikan ketika parameter *one length loop* diatur di 50 ke bawah. *Heuristic miner* memiliki hasil yang lebih baik daripada *inductive miner, heuristic miner* memiliki hasil *process mining* yang memiliki nilai *similarity* 1. Hal ini menunjukkan *heuristic miner* lebih mampu menangani *nested loops* daripada *inductive miner*. Hal ini dikarenakan pendekatan yang digunakan oleh *heuristics miner* lebih fleksibel dalam mendeteksi pola hubungan yang lebih kompleks dalam data *log*, termasuk pola yang berulang dalam bentuk *loop* yang bersarang. Dalam *heuristics miner*, hubungan antar *event* diukur dengan menggunakan *dependency measures* yang mempertimbangkan baik hubungan langsung maupun hubungan jangka Panjang. Ketika *log* berisi *nested loops*, model ini secara efektif dapat mengenali dan menangani dependensi yang saling terkait, termasuk hubungan berulang yang seringkali sulit dideteksi dengan metode lain seperti pada kode berikut.

```
START
  Set L1LThreshold = 0.8
  Set L2LThreshold = 0.5
  For each event pair (i, j):
    L1L = CalculateL1L(i, j)
    If L1L >= L1LThreshold:
      Accept dependency between i and j

    L2L = CalculateL2L(i, j)
    If L2L >= L2LThreshold:
      Accept long-range dependency between i and j

  For each loop:
    If loop detected:
      Process loop body and redo part under XOR
```

```
  If dependency measure < threshold:
    Remove noise and update connections

END
Function CalculateL1L(i, j):
  Return (SuccessionCount(i, j) + SuccessionCount(j, i)) /
(TotalEvents(i) + TotalEvents(j))

Function CalculateL2L(i, j):
  Return (LongRangeSuccession(i, j) + LongRangeSuccession(j, i))
/ (TotalEvents(i) + TotalEvents(j))
```

Pseudocode tersebut menggambarkan bagaimana *heuristics miner* menangani dependensi antara *event* dalam *log* dan mengelola *nested loops* serta *noise*. Fungsi utama dari *pseudocode* ini adalah untuk menghitung dan memverifikasi dependensi antara setiap pasangan *event* (L1L dan L2L), L1L mencerminkan dependensi langsung antara *event*, sementara L2L menggambarkan dependensi jangka panjang. Dependensi tersebut hanya diterima jika nilai L1L atau L2L melebihi ambang batas (*threshold*) yang ditentukan. Selain itu, *pseudocode* ini juga menangani *nested loops* dengan memisahkan proses *loop body* dan *redo* bagian dalam *nested loop*, serta menanganinya secara terpisah dengan XOR untuk menjaga integritas model. Terakhir, *pseudocode* ini memperhatikan *noise* dalam *log*, yaitu dependensi yang tidak memenuhi ambang batas, dan mengabaikannya untuk menghasilkan model yang lebih bersih dan relevan. Dengan menurunkan *threshold*, lebih banyak dependensi yang diterima, yang pada gilirannya menghasilkan model dengan hubungan yang lebih kompleks dan meningkatkan *similarity* antara model yang dihasilkan dan *log* asli.

4.7 Penentuan Parameter Optimal

Eksperimen dengan berbagai skenario dilakukan untuk menentukan parameter optimal bagi algoritma *process mining*, yaitu *heuristic miner* dan *inductive miner*. Berdasarkan hasil eksperimen skenario 1, model proses bisnis yang dihasilkan melalui *discovery* menggunakan *inductive miner* menghasilkan empat model proses bisnis dengan tingkat similarity di bawah 1, sementara *heuristic miner* menghasilkan lima model dengan *similarity* di bawah 1. Hal ini menunjukkan bahwa *inductive miner* lebih unggul dibandingkan *heuristic miner* dalam menangani dataset dengan tingkat kompleksitas yang bervariasi. *Inductive miner* mampu menghasilkan model yang lebih sesuai dengan data yang ada, meskipun terdapat beberapa model yang memiliki *similarity* di bawah 1. Sebaliknya, *heuristic miner* menghasilkan lebih banyak model dengan ketidakcocokan yang lebih besar terhadap dataset tersebut.

Pada eksperimen skenario 2, model proses bisnis yang dihasilkan menggunakan 8 varian *inductive miner*. Varian *inductive miner* dengan pilihan *infrequent & all operators* menghasilkan model proses bisnis dengan *similarity* di bawah 1 paling sedikit dan memiliki presentase *similarity* paling tinggi yaitu 97,46%. Hal ini menunjukkan bahwa kedua varian tersebut lebih mampu menangani berbagai model proses bisnis dengan kompleksitas yang beragam. *inductive miner infrequent & all operators* menghasilkan model dengan tingkat kesesuaian yang cukup baik. varian *infrequent & all operators* memberikan hasil yang lebih optimal dalam hal kecocokan model dengan data, yang tercermin dari nilai *similarity* yang lebih tinggi, jika *noise threshold*-nya di atas atau sama dengan

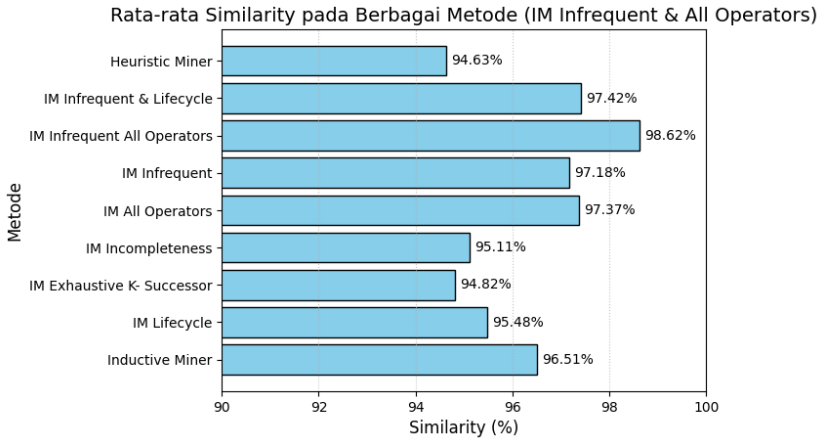
20, memungkinkan algoritma untuk menangkap lebih banyak kompleksitas dalam data dan menghasilkan model yang lebih akurat.

Pada eksperimen skenario 3, dengan variasi *relative-to-best threshold*, *length-one-loops*, *length-two-loops*, *long-distance,* dan *dependency threshold* yang berbeda, terlihat bahwa *heuristic miner* menghasilkan empat model proses bisnis yang memiliki nilai *similarity* di bawah 1 pada dataset yang sama, yaitu 4.6b, 4.6c, 4.7b, 4.9a, dan 4.9c. Namun, ketika dibandingkan antara variasi *dependency threshold*, *heuristic miner* dengan *threshold* lebih dari sama dengan 90 menunjukkan nilai *similarity* yang paling mendekati 1 pada sebagian besar model proses bisnis dibandingkan dengan *threshold* di bawah 90. Hal ini menunjukkan bahwa *Heuristic Miner* dengan *dependency threshold* lebih dari sama dengan 90 menghasilkan model yang lebih akurat dan lebih cocok dengan dataset yang digunakan. Nilai *similarity* yang lebih tinggi mendekati 1 mengindikasikan bahwa model proses yang dihasilkan lebih sesuai dengan alur proses yang ada pada data. Sebaliknya, nilai *similarity* yang lebih rendah menunjukkan ketidakcocokan antara model yang dihasilkan dengan dataset yang digunakan, yang berarti model tersebut kurang representatif atau kurang optimal dalam merepresentasikan alur proses yang sebenarnya. Oleh karena itu, dapat disimpulkan bahwa penggunaan *dependency threshold* 90 memberikan hasil yang lebih baik dalam hal kesesuaian model proses bisnis, sehingga memberikan kinerja yang lebih optimal dalam eksperimen ini dibandingkan dengan *threshold* yang lebih rendah.

Pada eksperimen skenario 4, eksperimen ini dilakukan untuk menguji kemampuan algoritma *heuristic miner* dan *inductive miner* menggunakan dataset

berkarakteristik *nested loops. Inductive miner* mengalami ketidakmampuan dalam menangani dataset berkarakteristik *nested loops* dibuktikan dengan hasil *similarity* tidak ada yang mencapai 1. Namun, *inductive miner infrequent, inductive miner infrequent & lifecycle, dan inductive miner infrequent & all operators* memiliki nilai *similarity* yang lebih baik dari varian *inductive miner* lainya dan mengalami peningkatan di pengaturan *threshold* lebih dari sama dengan 80. Sementara itu, *heuristic miner* memiliki nilai yang lebih baik daripada *inductive miner*. Parameter terbaik dalam menangani dataset berkarakteristik *nested loops* pada algoritma *heuristic miner* adalah pengaturan parameter *one length loop* kurang dari sama dengan 50.

Dari keempat eksperimen yang dilakukan, dapat disimpulkan bahwa parameter terbaik untuk *inductive miner* adalah varian *inductive miner infrequent & all operators*, karena menghasilkan model dengan tingkat *similarity* yang lebih tinggi dan lebih sedikit model dengan similarity di bawah 1. Sementara itu, untuk *heuristic miner*, parameter terbaik adalah *dependency threshold* lebih dari sama dengan 90, yang menghasilkan model dengan kesesuaian paling baik dengan dataset, dengan *similarity* yang paling mendekati 1. Gambar 4.15 menunjukkan tingkat keakuratan masing-masing metode (*inductive miner* dan *heuristic miner*) dengan pengaturan parameter terbaik dari uji skenario, berdasarkan rata-rata nilai *similarity* yang dihasilkan.



Gambar 4. 15 Rata-Rata *Similarity Inductive Miner* & *Heuristic Miner*

Gambar 4.15 menggambarkan seberapa baik setiap metode dalam mencocokkan model dengan data yang ada. Berdasarkan Gambar 4.15 terlihat bahwa *inductive miner infrequent & all operators* memberikan hasil rata-rata tertinggi, menunjukkan kesesuaian model yang lebih baik dengan data.

Sementara itu untuk dataset dengan karakteristik *nested loops* dapat disimpulkan parameter terbaik *inductive miner* adalah *inductive miner infrequent, inductive miner infrequent & lifecycle, dan inductive miner infrequent & all operators* dengan pengaturan lebih dari sama dengan 80. Sementara itu, untuk *heuristic miner*, parameter terbaik adalah *one length loop* kurang dari sama dengan 50.

Jika dibandingkan antara *heuristic miner* dan *inductive miner*, hasil eksperimen menunjukkan bahwa *inductive miner* lebih unggul. Hal ini dikarenakan *inductive miner* menghasilkan lebih sedikit model dengan *similarity* di bawah 1 dan presentase *similarity* lebih tinggi, yang menunjukkan bahwa model yang dihasilkan lebih sesuai dengan alur proses yang ada pada data. Dengan demikian, *inductive*

miner dapat dianggap sebagai pilihan yang lebih baik dibandingkan dengan *heuristic miner* dalam menangani berbagai karakteristik model proses bisnis.

4.8 Integrasi Islam

Penelitian ini tidak hanya bertujuan untuk mengeksplorasi efektivitas algoritma dalam *process mining*, seperti *heuristic miner* dan *inductive miner*, dalam membangun model proses bisnis yang lebih akurat dan efisien, tetapi juga berupaya untuk mengintegrasikan nilai-nilai Islam dalam setiap tahapannya. *Process mining* adalah teknik analisis data yang bertujuan untuk mengevaluasi dan menganalisis proses bisnis berdasarkan data yang tercatat dalam sistem informasi, melalui *event logs* yang mencatat setiap aktivitas yang terjadi. Dalam *process mining*, data yang telah tercatat ini dianalisis untuk membangun model proses yang menggambarkan bagaimana proses tersebut berjalan, mengidentifikasi inefisiensi, dan memberikan wawasan untuk perbaikan. Prinsip yang terkandung dalam *process mining* sangat mirip dengan ajaran Islam mengenai pencatatan amal. Dalam Islam, segala tindakan, baik yang besar maupun kecil, tercatat dengan akurat dan rinci untuk pertanggungjawaban di dunia dan akhirat, sebagaimana yang tercantum dalam Surah Al-Qamar ayat 52-53:

وَكُلُّ شَيْءٍ فَعَلُوْهُ فِى الزُّبُرِ وَ كُلُّ صَغِيْرٍ وَّكَبِيْرٍ مُّسْتَطَرٌ

"*Segala sesuatu yang telah mereka perbuat (tertulis) dalam buku-buku catatan (amal). Dan setiap hal yang kecil maupun besar adalah tertulis." (QS. Al-Qamar :52-53)*

Dalam *Tafsir Al-Mishbah*, Quraish Shihab (Shihab, 2005) menjelaskan bahwa ayat ini menunjukkan bahwa segala tindakan manusia, baik yang kecil

maupun yang besar, semuanya tercatat dengan rapi dan sempurna dalam buku catatan amal. Tidak ada satu pun perbuatan yang luput dari pencatatan, sekecil apapun itu. Prinsip ini sangat relevan dengan konsep *process mining* dalam konteks bisnis, setiap aktivitas yang tercatat dalam *event logs* dianalisis secara menyeluruh untuk memantau kinerja, mengidentifikasi kesalahan, dan meningkatkan efisiensi. Sama seperti pencatatan amal yang menekankan keadilan dan akurasi, proses *mining* memastikan bahwa setiap tindakan bisnis terdokumentasi dengan tujuan evaluasi dan perbaikan. Prinsip ini mengingatkan kita bahwa setiap tindakan akan diperhitungkan dan memiliki dampak di masa depan, baik di dunia bisnis maupun dalam kehidupan akhirat menurut ajaran Islam.

Prinsip pencatatan amal ini juga tercermin dalam Surah Al-Zalzalah ayat:7-8, yang menegaskan bahwa segala amal, sekecil apapun, akan dihitung dan dilihat. Allah SWT berfirman:

فَمَن يَعْمَلْ مِثْقَالَ ذَرَّةٍ خَيْرًا يَرَهُ وَمَن يَعْمَلْ مِثْقَالَ ذَرَّةٍ شَرًّا يَرَهُ

"*Barang siapa yang mengerjakan kebaikan seberat dzarrah, pasti dia akan melihatnya, dan barang siapa yang mengerjakan keburukan seberat dzarrah, pasti dia akan melihatnya*." *(QS. Al-Zalzalah :7-8)*

Dalam tafsirnya pada Surah Al-Zalzalah (Dahri, 2022), KH. Moechjar Dahri menjelaskan bahwa setiap amal perbuatan, sekecil apapun, akan dihitung dan diperlihatkan pada hari kiamat. Ini mencerminkan prinsip akuntabilitas yang sangat kuat dalam Islam, segala sesuatu yang kita lakukan, baik yang baik maupun buruk, akan dicatat dan tidak ada yang terlewatkan. Dalam konteks ini, bumi sebagai saksi akan memperlihatkan segala peristiwa yang terjadi, seiring dengan datangnya hari

kiamat yang akan mengguncangkan seluruh alam semesta. Hal ini menjadi pengingat bagi setiap individu bahwa tidak ada satu pun perbuatan yang luput dari perhatian Allah, baik itu sebesar atom (*dzarrah*) maupun lebih besar dari itu.

Tafsir KH. Moechjar Dahri juga menunjukkan bahwa hari kiamat merupakan saat pertanggungjawaban semua amal yang telah dilakukan oleh manusia, segala catatan akan dipertimbangkan dengan adil. Hal ini bisa dianalogikan dengan prinsip yang terkandung dalam *process mining*, setiap langkah dalam sebuah proses tercatat dan dianalisis secara cermat, memastikan bahwa tidak ada yang terlewat dalam evaluasi dan perbaikan model

.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dapat disimpulkan bahwa:

1. Tingkat kompleksitas model proses bisnis tidak selalu berpengaruh langsung terhadap akurasi *process mining* yang dihasilkan oleh metode *inductive miner* dan *heuristic miner*. Sebaliknya, faktor yang lebih berpengaruh dalam menentukan akurasi hasil *process mining* adalah struktur dari model BPMN khususnya elemen-elemen seperti *looping*, kedalaman percabangan, dan percabangan OR.
2. Kompleksitas model proses bisnis tidak secara langsung mempengaruhi kinerja metode *heuristic miner* dan *inductive miner* dalam menghasilkan model yang akurat. yang lebih memengaruhi adalah struktur dari BPMN itu sendiri, seperti adanya *looping*, kedalaman percabangan, dan percabangan OR. Kompleksitas model proses bisnis tidak secara langsung mempengaruhi kinerja metode *heuristic miner* dan *inductive miner* dalam menghasilkan model yang akurat. Hal yang lebih berpengaruh adalah struktur dari BPMN itu sendiri, seperti adanya *looping*, kedalaman percabangan, dan percabangan OR. Struktur-struktur ini dapat memengaruhi bagaimana kedua algoritma ini memproses data dan menghasilkan model. Hasil *process mining* dengan algoritma *inductive*

miner menunjukkan tingkat *similarity* sebesar 98,62%, sementara *heuristic miner* menghasilkan nilai *similarity* 94,63%.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, beberapa rekomendasi dan saran untuk penelitian selanjutnya dapat dipertimbangkan. Saran-saran ini bertujuan untuk mengeksplorasi pendekatan yang lebih beragam dan memperluas cakupan penggunaan data serta algoritma yang lebih canggih. Berikut adalah beberapa saran yang dapat dijadikan referensi untuk penelitian lanjutan:

1. Penelitian selanjutnya disarankan untuk mengeksplorasi algoritma *process mining* yang lain untuk memperkaya pemahaman tentang efektivitas berbagai metode dalam *process mining*, disarankan untuk mengeksplorasi algoritma *process mining* lainnya selain *heuristic miner* dan *inductive miner*. algoritma seperti *genetic algorithm*, *alpha miner*, atau *fuzzy miner* dapat memberikan perspektif yang berbeda dalam hal penanganan data yang kompleks dan tidak terstruktur. menguji berbagai algoritma ini dalam konteks yang sama akan memungkinkan perbandingan kinerja, terutama dalam hal akurasi, efisiensi, dan kemampuan algoritma untuk menangani variasi dalam struktur proses bisnis.
2. Menguji coba data yang memiliki *noise*, salah satu tantangan yang sering dihadapi dalam *process mining* adalah keberadaan *noise* dalam data *log.* untuk penelitian selanjutnya, disarankan untuk menguji algoritma dengan data yang mengandung *noise*, seperti data yang tidak lengkap, kesalahan pencatatan, atau aktivitas yang tidak relevan. Penanganan *noise* dalam data

akan menguji ketahanan dan kemampuan algoritma dalam menghasilkan model yang akurat meskipun data yang digunakan tidak sempurna. Hal ini penting untuk memahami sejauh mana algoritma dapat beradaptasi dengan kondisi data yang tidak ideal dan untuk meningkatkan robustitas model yang dihasilkan.

DAFTAR PUSTAKA

Aalst, W. V. D. (2016). *Process Mining*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49851-4

Aalst, W. V. D., & Dustdar, S. (2012). Process Mining Put into Context. *IEEE Internet Computing, 16*(1), 82–86. https://doi.org/10.1109/MIC.2012.12

Abriani, G. U. (2020). Pengukuran Kemiripan Model Proses Bisnis untuk Menentukan Common Fragment. *MATICS*, *12*(2), 117–123. https://doi.org/10.18860/mat.v12i2.8355

Alian, M., & Awajan, A. (2020). Semantic Similarity for English and Arabic Texts: A Review. *Journal of Information & Knowledge Management*, *19*(04), 2050033. https://doi.org/10.1142/S0219649220500331

Andreswari, R., Syahputra, I., & Lubis, M. (2021). Performance Analysis of Heuristic Miner and Genetics Algorithm in Process Cube: A Case Study. *International Journal on Advanced Science, Engineering and Information Technology, 11*(1), 393–399. https://doi.org/10.18517/ijaseit.11.1.11544

Augusto, A., Mendling, J., Vidgof, M., & Wurm, B. (2022). The Connection Between Process Complexity of Event Sequences and Models Discovered by Process Mining. *Information Sciences*, 598, 196–215. https://doi.org/10.1016/j.ins.2022.03.072

Baihaki, M. & Djamaluddin. (2022). Perancangan Proses Bisnis Sistem Informasi Pendaftaran Tugas Akhir Menggunakan Business Prosess Modelling Notation (BPMN). *Bandung Conference Series*: Industrial Engineering Science, 2(1). https://doi.org/10.29313/bcsies.v2i1.2447

Bakhshi, A., Hassannayebi, E., & Sadeghi, A. H. (2023). Optimizing Sepsis Care Through Heuristics Methods in Process Mining: A Trajectory Analysis. *Healthcare Analytics*, 3, 100187. https://doi.org/10.1016/j.health.2023.100187

Blattmeier, M. (2023). The Aestheticization of Business Processes: Visualizing Their Gestalt for Collective Thinking. *Journal of Information Technology*, *38*(4), 459–486. https://doi.org/10.1177/02683962231166438

Dahri, M. (2022). *Tafsir Juz Amma*. Sahabat Pustaka.

Dijkman, R., Dumas, M., & García-Bañuelos, L. (2009). Graph Matching Algorithms for Business Process Model Similarity Search. In U. Dayal, J. Eder, J. Koehler, & H. A. Reijers (Eds.), *Business Process Management* (Vol. 5701, pp. 48–63). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03848-8_5

Hasyim, R. A., Yaqin, M. A., & Utomo, A. H. (2020). Analisis Perbandingan Metode Alpha Miner, Inductive Miner dan Causal-Net Mining dalam Proses Mining. *Jurnal Teknologi Informasi dan Terapan*, *7*(2), 77–85. https://doi.org/10.25047/jtit.v7i2.165

Hutahean, N., Sholeha, D., & Simanullang, S. (2022). Penerapan Perangkat EvenLog Analyzer pada Digital Forensik di Lingkungan Fakultas Teknik Universitas Darma Agung. Impression : *Jurnal Teknologi dan Informasi*, *2*(3), 112–118. https://doi.org/10.59086/jti.v2i3.538

Kamala, B. (2019). BAGH – Comparative Study. *2019 3rd International Conference on Computing and Communications Technologies (ICCCT)*, 272–277. https://doi.org/10.1109/ICCCT2.2019.8824878

Kang, D., Song, I.-G., Park, S., Bae, D.-H., Kim, H.-K., & Lee, N. (2008). A Case Retrieval Method for Knowledge-Based Software Process Tailoring Using Structural Similarity. *2008 15th Asia-Pacific Software Engineering Conference*, 51–58. https://doi.org/10.1109/APSEC.2008.15

Kräuter, T., Rutle, A., König, H., & Lamo, Y. (2023). Formalization and Analysis of BPMN Using Graph Transformation Systems. In M. Fernández & C. M. Poskitt (Eds.), *Graph Transformation* (Vol. 13961, pp. 204–222). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-36709-0_11

Kurniati, A. P., Kusuma, G. P., & Suyanto. (2023). *Process Mining*. Informatika Bandung.

Lam, B. H., Nguyen, V. T. H., Phan, C. H., & Truong, T. T. T. (2020). An Approach for Application Generation Based on BPMN. *2020 12th International Conference on Knowledge and Systems Engineering (KSE)*, 115–119. https://doi.org/10.1109/KSE50997.2020.9287849

Nuritha, I., & Mahendrawathi, E. R. (2017). Structural Similarity Measurement of Business Process Model to Compare Heuristic and Inductive Miner Algorithms Performance in Dealing with Noise. *Procedia Computer Science*, 124, 255–263. https://doi.org/10.1016/j.procs.2017.12.154

Parimala, K., Rajkumar, G., Ruba, A., & Vijayalakshmi, S. (2017). Challenges and Opportunities with Big Data. *International Journal of Scientific Research in Computer Science and Engineering, 5*(5), 16–20.

Peng, W., Zhang, Z., Hildebrant, R., & Ren, S. (2021). Empirical Studies of Three Commonly Used Process Mining Algorithms. *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2492–2499. https://doi.org/10.1109/SMC52423.2021.9658861

Rahmawati, D. (2016). *Aplikasi Pendeteksi Fraud pada Event Log Proses Bisnis Pengadaan Barang dan Jasa menggunakan Algoritma Heuristic Miner*. Universitas Islam Negeri Malang Maulana Malik Ibrahim Malang.

Rani, S. (2021). Data Mining. *International Journal of Advanced Research in*

Science, Communication and Technology, 50–55. https://doi.org/10.48175/IJARSCT-V2-I3-309

Rolón, E., Cardoso, J., García, F., Ruiz, F., & Piattini, M. (2009). Analysis and Validation of Control-Flow Complexity Measures with BPMN Process Models. In T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, & R. Ukor (Eds.), *Enterprise, Business-Process and Information Systems Modeling* (Vol. 29, pp. 58–70). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-01862-6_6

Sahlabadi, M., Chandren Muniyandi, R., Shukur, Z., Qamar, F., & Hussain Ali Kazmi, S. (2023). Process Mining Discovery Techniques for Software Architecture Lightweight Evaluation Framework. *Computers, Materials & Continua, 74*(3), 5777–5797. https://doi.org/10.32604/cmc.2023.032504

Shihab, M. Q. (2005). *Tafsir al-Mishbāh: Pesan, kesan, dan keserasian al-Qur'an* (Cet. 6). Lentera Hati.

Sumarsono, S., Saputro, D., & Rifai, A. F. (2023). Pemodelan Proses Bisnis Kuliah Online MOOCs menggunakan BPMN (Studi Kasus alison.com). *JISKA (Jurnal Informatika Sunan Kalijaga), 8*(3), 199–209. https://doi.org/10.14421/jiska.2023.8.3.199-209

Tampubolon, M. M., & Situmorang, P. N. C. (2023). Pembuatan Model Bisnis Proses Aplikasi Tebaran Nusira dengan Pendekatan BPMN. *Data Sciences Indonesia (DSI), 3*(1), 12–22. https://doi.org/10.47709/dsi.v3i1.2269

Vincek, I. (2018). Business Processes as Business Systems. *Tehnički Glasnik, 12*(1), 55–61. https://doi.org/10.31803/tg-20170808183458

Von Rosing, M., White, S., Cummins, F., & De Man, H. (2015). Business Process Model and Notation—BPMN. In *The Complete Business Process Handbook* (pp. 433–457). Elsevier. https://doi.org/10.1016/B978-0-12-799959-3.00021-5

Vossen, G. (2012). The Process Mining Manifesto—An interview with Wil van der Aalst. *Information Systems, 37*(3), 288–290. https://doi.org/10.1016/j.is.2011.10.006

Weske, M. (2012). *Business Process Management*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-28616-2

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z.-H., Steinbach, M., Hand, D. J., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1–37. https://doi.org/10.1007/s10115-007-0114-2

Yaqin, M., Sarno, R., Institut Teknologi Sepuluh Nopember, Rochimah, S., & Institut Teknologi Sepuluh Nopember. (2020). Measuring Scalable Business Process Model Complexity Based on Basic Control Structure. *International*

Journal of Intelligent Engineering and Systems, 13(6), 52–65. https://doi.org/10.22266/ijies2020.1231.06

Zeng, Q., Liu, J., Zhou, C., Liu, C., & Duan, H. (2020). A Novel Approach for Business Process Similarity Measure Based on Role Relation Network Mining. *IEEE Access, 8*, 60918–60928. https://doi.org/10.1109/ACCESS.2020.2983114