Checagem de conformidade baseada em alinhamento para uma rede de Petri estocástica

Matheus Pereira de Almeida

Dissertação apresentada ao
Instituto de Matemática e Estatística
da Universidade de São Paulo
para obtenção do título de
Mestre em Ciências

Programa: Ciência da Computação
Orientadora: Profª. Drª. Karina Valdivia Delgado

São Paulo
Outubro de 2023

Checagem de conformidade baseada em alinhamento para uma rede de Petri estocástica

Matheus Pereira de Almeida

Esta é a versão original da dissertação
elaborada pelo candidato Matheus
Pereira de Almeida, tal como
submetida à Comissão Julgadora.

Dedico essa dissertação aos meus pais, os dois maiores incentivadores das realizações dos meus sonhos.

Agradecimentos

A Deus, por ter permitido que eu tivesse saúde e determinação para não desanimar durante a realização deste trabalho. A minha orientadora, que sempre me auxiliou no desenvolvimento desse projeto, dedicando incontáveis horas. A todos que participaram, direta ou indiretamente do desenvolvimento deste trabalho de pesquisa, enriquecendo o meu processo de aprendizado.

Resumo

Matheus Pereira de Almeida. **Checagem de conformidade baseada em alinhamento para uma rede de Petri estocástica**. Dissertação (Mestrado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

O alinhamento oferece um método confiável para realizar a checagem de conformidade, detectando desvios que levam à não conformidade. A maioria dos métodos de alinhamento utiliza o conceito de *movimentos de alinhamento* em conjunto com uma rede de Petri determinística e um log de eventos como entrada, com o objetivo de encontrar um alinhamento com o menor custo. Lidar com informações probabilísticas no modelo de processo e no log de eventos é um dos grandes desafios na área de checagem de conformidade. No entanto, há poucos trabalhos na literatura sobre checagem de conformidade que exploram informações estocásticas no modelo de processo. Abordagens probabilísticas existentes não levam em consideração o conceito de *movimentos de alinhamento*, especificamente os *movimentos de log*, pressupondo que o modelo de processo fornecido como entrada permanece inalterado. Baseando-se nos fundamentos existentes da área de checagem de conformidade que utiliza a técnica de alinhamento, foi definida formalmente neste trabalho a tarefa de alinhamento que tem como entrada uma rede de Petri estocástica e um log de eventos, emprega o conceito de movimentos de alinhamento e tem como objetivo produzir um ranking ótimo de alinhamentos considerando uma métrica de otimização que combina a probabilidade de disparo das transições da rede de Petri estocástica e o custo dos alinhamentos. Como resultado, obtemos um ranking de alinhamentos que reflete a realidade das ocorrências dos traces presentes no log de eventos e a suposição de que o modelo de processo em uso pode estar desatualizado. Também propomos o *ProbPlanAlign*, uma abordagem que modela essa tarefa de alinhamento como um problema de Caminho Mais Curto Estocástico, codifica-o usando a Linguagem de Definição de Domínio de Planejamento Probabilístico (PPDDL) e encontra o ranking ótimo de alinhamentos usando planejadores probabilísticos existentes na literatura. Experimentos com o *ProbPlanAlign* utilizando logs de eventos sintéticos e modelos de processos com um número crescente de transições foi realizada para analisar a escalabilidade da proposta. O consumo de tempo do *ProbPlanAlign* para encontrar um ranking ótimo de alinhamentos cresce linearmente à medida que o tamanho das redes de Petri probabilísticas cresce, em termos do número de transições. Especificamente, o consumo de tempo foi de 21,4 segundos para o modelo de processo maior avaliado, que possui 237 transições.

Palavras-chave: Mineração de processo. Checagem de conformidade. Alinhamento. Planejamento automático. Planejamento probabilístico.

Abstract

Matheus Pereira de Almeida. **Alignment-based conformance checking for a stochastic Petri net**. Thesis (Master's). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

Alignment offers a reliable method for conformance checking by detecting deviations that lead to nonconformity. Most alignment methods use the concept of *alignment moves*, with a deterministic Petri net and an event log trace to be aligned as input, and aim to find an alignment with the lowest cost. Dealing with probabilistic information from the process model and event log is one of the great challenges in the area of conformance checking. However, there are few works in the literature on conformance checking that explore stochastic information from the process model. Existing probabilistic approaches do not take into account the concept of *alignment moves*, specifically the *log moves*, assuming that the process model provided as input remains unchanged. Building on the existing alignment-based conformance-checking fundamentals, we formally define in this paper the alignment task that takes as input a stochastic Petri net and an event log trace to be aligned, employs the concept of alignment moves, and aims to produce an optimal ranking of alignments considering an optimization metric that combines both the firing probability of transitions in the stochastic Petri net and the cost of the alignments. Consequently, a ranking of alignments is obtained that meets the reality of trace occurrences in the event log and the assumption that the process model in use may be outdated. We also propose *ProbPlanAlign*, an approach that models this alignment task as a Stochastic Shortest Path problem, encodes it using the Probabilistic Planning Domain Definition Language (PPDDL), and finds the optimal ranking of alignments using off-the-shelf probabilistic planners. An experimental evaluation of *ProbPlanAlign* with synthetic event logs and process models with a different number of transitions was performed to analyze its scalability. The time consumption of *ProbPlanAlign* to find an optimal ranking of alignments grows linearly when the size of the probabilistic Petri nets grows in terms of transitions. Specifically, the time consumption was 21.4 seconds for the larger process model evaluated which has 237 transitions.

Keywords: Process mining. Conformance checking. Alignment. Automated planning. Probabilistic planning.

Lista de Abreviaturas

LD	Distância de Levenshtein (*Levenshtein distance*).
EMD	Distância de Movimento da Terra (Earth's Move Distance).
BPM	Gerenciamento de processos de negócio (*Business process management*).
SSP	Caminho Mais Curto Estocástico (Stochastic Shortest Path).
PNML	Linguagem de marcação de rede de Petri (*Petri Net Markup Language*).
BPMN	Modelo e notação de processos de negócio (*Business Process Model and Notation*).
PDDL	Linguagem de definição de domínio de planejamento.
PPDDL	Linguagem de definição de domínio de planejamento probabilístico.
MDP	Processo de Decisão de Markov.

Lista de Símbolos

ℓ	Função que associa um rótulo à uma transição.
m_i	*Marking* inicial.
m_f	*Marking* final.
ε	O universo de eventos.
σ_L	Um *trace*.
(s_L, s_M)	Representa um movimento legal do alinhamento.
$\gg$	Ausência de movimento.
τ	Rótulo vazio.
λ_N	Função que mapeia cada evento a seu respectivo rótulo.
ℓ	Função que associa um rótulo com uma transição em T.
$\emptyset$	Conjunto vazio.
s_0	O estado inicial de alinhamento.
evEND	Evento artificial.
$\mathcal{G}$	Conjunto de estados de alinhamento final.
$K(\gamma)$	Função do custo de um alinhamento.
γ_*	Alinhamento ótimo.
π	Política.
π_*	Política ótima.
$\mathscr{Y}$	Conjunto de todos os pares (s_L, s_M).
$\mathcal{S}$	Conjunto de estados de alinhamento.
L	Log de eventos.
N	Rede de Petri.
P	Conjunto finito de lugares.
T	Conjunto finito de transições.
A	Conjunto que denota o nome das atividades.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

O conceito de alinhamento apresenta uma abordagem para a realização da checagem de conformidade, permitindo a detecção de desvios que levam à não conformidade. Quando as ações registradas no log de eventos estão em concordância com às ações permitidas pelo modelo de processo para uma instância desse processo, isso indica que ambos estão alinhados.

A maioria dos métodos de alinhamento utilizam o conceito de movimentos de alinhamento, dado como entrada um modelo de processo representado por uma rede de Petri determinística e um trace a ser alinhado. Existem três tipos de movimentos de alinhamento [6]: movimentos *síncronos*, movimentos *do modelo* e movimentos *de log*.

Um movimento *síncrono* ocorre quando a execução da atividade registrada no trace do log de eventos se alinha com uma transição permitida no modelo de processo. Um movimento *do modelo* acontece quando há uma movimentação exclusivamente dentro do modelo de processo, e por fim, um movimento *de log* ocorre quando há um movimento exclusivamente dentro do trace do log de eventos.

Cada movimento de alinhamento tem um custo associado. O mais utilizado na literatura associa um custo 0 a um movimento *síncrono* e 1 aos demais. Por fim, o objetivo é encontrar um alinhamento ótimo, ou seja, um alinhamento com o menor custo total.

Lidar com informações probabilísticas pertencentes ao modelo de processo ou ao log de eventos é um dos grandes desafios na área de checagem de conformidade [1]. No entanto, há poucos trabalhos na literatura [12, 31, 9, 18, 20] que exploram informações estocásticas no modelo de processo. Para representar as informações estocásticas de um modelo de processo, geralmente são utilizadas redes de Petri estocásticas; este é um tipo de rede de Petri que contém probabilidades de disparo para as transições. Essa rede de Petri estocástica pode ser obtida através do conhecimento de um analista de negócios, que avalia o processo e atribui pesos para as transições com base em sua experiência, ou através de uma análise automatizada do log de eventos [12, 31]. A Figura 1.1 mostra um exemplo de uma rede de Petri estocástica que possui cinco transições ($t1$, $t2$, $t3$, $t4$ e $t5$) que são representadas por retângulos, quatro lugares ($p1$, $p2$, $p3$ e $p4$) representados por círculos, um token em $p1$ e rótulos *(Receive order, Analyze order, Accept, Reject* e τ). Observe que as letras entre parênteses na Figura 1.1 foram usadas para encurtar os rótulos.

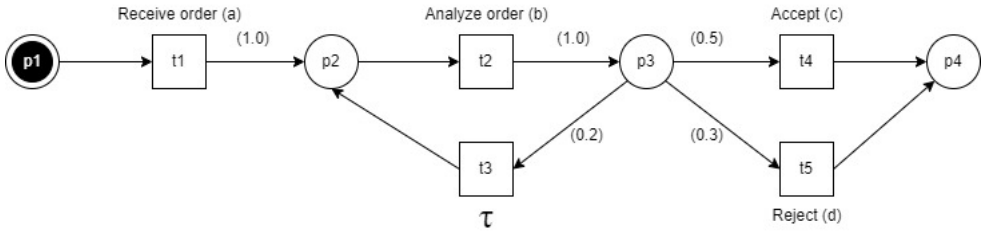

Figura 1.1: *Exemplo de uma rede de Petri estocástica adaptada de Koorneef et al. [16].*

O modelo de processo apresentado na Figura 1.1 é um exemplo de um processo de compra. Ele começa recebendo um pedido de compra, em seguida, a empresa analisa o pedido, pode repetir a análise, se necessário, e, finalmente, aceitar ou rejeitar o pedido. *Receive Order* e *Analyze order* têm probabilidade de disparo 1, e *Accept*, *Reject* e τ têm probabilidades de disparo de 0,5, 0,3 e 0,2, respectivamente. Além disso, é possível que uma atividade não apresentada no modelo possa ocorrer no log de eventos; nesse caso, é possível supor que o modelo está desatualizado. Para esse cenário, considere a atividade f (que não está presente no modelo) como a ação de enviar um e-mail ao cliente confirmando o recebimento do pedido.

Existem alguns trabalhos na literatura que utilizam modelos de processo estocásticos, como abordagens para realizar a descoberta de processos estocásticos [12, 31, 27] e checagem de conformidade estocástica [20, 21, 9, 18]. Leemans, Syring e Aalst [20] e Leemans et al. [21] usam uma métrica baseada no método chamado *distância de movimento da terra* (EMD) para medir o esforço necessário para transformar o log de eventos em traces permitidos pelo modelo de processo estocástico. Leemans, Maggi e Montali [18] (i) calculam a probabilidade de um trace atingir um marking final, (ii) calculam a probabilidade de que a rede gere um trace do modelo que satisfaça propriedades temporais e (iii) verificam se a rede está em conformidade com um conjunto de restrições temporais probabilísticas. A abordagem de Bergami et al. [9] realiza a verificação de conformidade e tem como entrada um modelo de rede de fluxo *(workflow net)* estocástico e um trace a ser alinhado e produz como saída um ranking de alinhamentos, contendo apenas alinhamentos permitidos pelo modelo. Para a geração do ranking, são considerados (i) a similaridade do trace sendo alinhado com o modelo e (ii) a probabilidade do trace ocorrer no modelo.

De acordo com Bergami et al. [9], este é o primeiro artigo que encontra alinhamentos usando um modelo estocástico. No entanto, essa abordagem não utiliza o conceito de movimento de alinhamento, pressupondo que o modelo de processo fornecido como entrada permanece inalterado. Por exemplo, usando a rede de Petri apresentada na Figura 1.1 e o trace $\langle a, f \rangle$ a ser alinhado, Bergami et al. [9] gera um ranking que inclui os seguintes alinhamentos permitidos pelo modelo: $\langle a, b, c \rangle$, $\langle a, b, d \rangle$, $\langle a, b, b, c \rangle$, $\langle a, b, b, d \rangle$, $\langle a, b, b, b, c \rangle$ e $\langle a, b, b, b, d \rangle$.

1.1 Objetivos

O objetivo deste trabalho é formalizar a tarefa de alinhamento para redes de Petri estocásticas e propor uma primeira abordagem que encontra um ranking de alinhamentos

a partir de uma política ótima que reflete à realidade das ocorrências no log de eventos e tem como pressuposto que o modelo de processo em uso pode estar desatualizado. Por exemplo, usando a rede de Petri estocástica apresentada na Figura 1.1 e o trace $\langle a, f, c \rangle$ para ser alinhado, nossa abordagem gera um ranking com alinhamentos que permitem atividades não apresentadas no modelo, por exemplo, $\langle a, f, b, c \rangle$ que inclui a atividade f.

1.2 Contribuições

As contribuições deste trabalho podem ser resumidas da seguinte forma:

- Uma formalização da tarefa de alinhamento que tem como entrada uma rede de Petri estocástica e um trace a ser alinhado, emprega o conceito de movimentos de alinhamento e visa produzir um ranking de alinhamentos utilizando uma política ótima considerando uma métrica de otimização que combina a probabilidade de disparo de transições na rede de Petri estocástica e o custo dos alinhamentos.
- *ProbPlanAlign*, uma abordagem que (i) modela a tarefa de alinhamento (considerando uma rede de Petri estocástica e um trace a ser alinhado) como um problema do *caminho mais curto estocástico* [10], (ii) a codifica usando a *linguagem probabilística de definição de domínio de planejamento* (PPDDL) [33], e (iii) encontra um ranking de alinhamentos a partir de uma política ótima e usando planejadores probabilísticos disponíveis na literatura.
- Uma avaliação experimental abrangente do *ProbPlanAlign* com logs de eventos sintéticos e modelos de processos com um número diferente de transições foi realizado para analisar a escalabilidade da proposta quando o tamanho das redes de Petri probabilísticas aumenta.

1.3 Estrutura do trabalho

O restante deste trabalho está organizado da seguinte forma: a Seção 2 apresenta os fundamentos teóricos; a Seção 3 apresenta a formalização proposta da tarefa de alinhamento para uma rede de Petri estocástica e um trace a ser alinhado pertencente a um log de eventos; a Seção 4 descreve como a tarefa de alinhamento é representada como um problema de caminho mais curto estocástico e codificada usando planejamento probabilístico; a Seção 5 apresenta nossa estratégia para realizar a tarefa de alinhamento e gerar um ranking; a Seção 6 apresenta experimentos com o *ProbPlanAlign* e a abordagem de Bergami et al. [9] em log de eventos sintéticos e modelos de processo com diferentes números de transições; e a Seção 7 apresenta as considerações finais.

Capítulo 2

Fundamentos teóricos

Nesse capitulo os fundamentos teóricos utilizados nesse trabalho são introduzidos.

2.1 Redes de Petri e Log de Eventos

Uma das notações usadas para representar processos de negócios é a rede de Petri. Elas são simples e adequadas o suficiente para modelar aspectos cruciais dos processos de negócios [2]. Primeiramente, são definidas redes de Petri rotuladas determinísticas (simplesmente chamadas de redes de Petri) e log de eventos, em seguida é definido o conceito de rede de Petri estocástica.

Definição 1 (Rede de Petri rotulada[18])**.** *Uma rede de Petri rotulada é definida como uma tupla* (P, T, F, A, ℓ) *tal que:*

- P *é um conjunto finito de lugares.*
- T *é um conjunto finito de transições tal que* $T \cap P = \emptyset$.
- $F \subseteq (P \times T) \cup (T \times P)$ *é uma relação que conecta lugares às transições e transições aos lugares.*
- A *é um conjunto que denota o nome das atividades e o rótulo silencioso* τ.
- $\ell : T \to A$ *é uma função que associa um rótulo com uma transição em* T.

Um lugar $p \in P$ serve como um lugar de entrada para uma transição $t \in T$ se e somente se $(p, t) \in F$. O conjunto de *lugares de entrada* para uma transição t é simbolizado como $\bullet t$. Da mesma forma, um lugar $p \in P$ é um lugar de saída para uma transição t se e somente se $(t, p) \in F$. O conjunto de *lugares de saída* para uma transição t é representado como $t\bullet$.

O estado de execução de uma rede de Petri está associado à distribuição de *tokens* em seus lugares e é referido como um *marking* .

Definição 2 (Marking [18]). *Um marking m de uma rede de Petri N é um multiconjunto sobre os lugares de N, associando a cada lugar $p \in P$ um número $m(p)$ de tokens em p. Dado um marking m de N e uma transição $t \in T$, pode-se dizer que:*

- *t está habilitada no marking m se e somente se para cada lugar de entrada há pelo menos um token, i.e, para cada lugar $p \in {}^\bullet t$, $m(p) \geq 1$.*
- *$E(m)$ é o conjunto de transições habilitidadas em um marking m.*
- *Assumindo que t está habilitada em m, t dispara em m para a rede de Petri N, produzindo um novo marking m'.*

A definição de marking está intimamente ligada ao estado do modelo de processo, fornecendo uma descrição e dando um sentido de continuidade para o modelo de processo. A ativação de uma transição habilitada resulta na criação de um novo *marking*, na qual um *token* é consumido de cada lugar de entrada, enquanto um *token* é gerado a cada lugar de saída.

Definição 3 (Execução [18]). *Uma execução de uma rede de Petri N a partir de um marking inicial m_s até um marking final m_f é uma sequencia finita de transições $t_0, t_1, ..., t_n$ tal que existem markings $m_0, m_1, ..., m_{n+1}$ para a rede de Petri N com (i) $m_0 = m_s$ e (ii) $m_{n+1} = m_f$.*

A execução de uma instância de um modelo de processo está diretamente relacionada ao conceito de evento, trace e log de eventos. Um evento é a execução de uma atividade registrada em um log de eventos e serve como dado fundamental usado no contexto de mineração de processos. Um evento contém certos atributos essenciais, como *ID*, atividade, *timestamp* e outros atributos que podem ser opcionais, como recurso e custo. O *timestamp* de um evento é um atributo que registra o momento exato em que o evento ocorreu. Considerando o modelo de processo apresentado na Figura 1.1, um exemplo de evento pode ser (1, *Receive order*, 1/1/2017 9:13:00, (Recurso David, Custo 10)). *Por simplicidade, foram* abstraídos os atributos adicionais e utilizadas apenas as informações sobre os rótulos das atividades registradas. Para fazer isso, uma função de mapeamento chamada *Classificador* é definida, que é responsável por realizar essa tarefa.

Definição 4 (Classificador [5]). *Um classificador λ_N é uma função que mapeia cada evento a seu respectivo rótulo (i.e., $\lambda_N : \varepsilon \rightarrow A$).*

Usando essa abstração é possível definir formalmente trace e log de eventos.

Definição 5 (Trace e log de eventos [5]). *Um trace é uma sequência de atividades e um log de eventos L é um multiconjunto de traces.*

Nesse trabalho, é feita a suposição de que um log de eventos contém dados exclusivamente associados a um processo. Além disto, para maior clareza, embora um trace seja definido como uma sequência de atividades, a partir deste ponto, será usado o termo *evento* para se referir a um componente de um trace em um log de eventos.

A Tabela 2.1 mostra um exemplo de um log de eventos com apenas os rótulos abreviados e suas respectivas frequências de ocorrência no log de eventos. Dois tipos de traces que

Trace	Frequency
$\langle a, b, c \rangle$	40
$\langle a, b, d \rangle$	10
$\langle a, b, b, c \rangle$	80
$\langle a, b, b, d \rangle$	20
$\langle a, b, b, b, c \rangle$	40
$\langle a, b, b, b, d \rangle$	10
$\langle a, b, b, b, b, b, b, b, c \rangle$	1
$\langle a, c \rangle$	1

Tabela 2.1: *Exemplo de um log de eventos, adaptado de Koorneef et al. [16].*

podem fazer parte de um log de eventos são traces permitidos pelo modelo e traces com ruido, definidos da seguinte forma.

Definição 6 (Trace de modelo [18])**.** *Dado uma rede de Petri rotulada N, um trace σ é um trace de modelo de N se existir uma execução de transições $\eta = t_0, t_1, ..., t_n$, que tem uma sequência correspondente de rótulos $\ell(t_0), \ell(t_1), ..., \ell(t_n)$ de tal forma que esta sequência corresponda ao trace σ uma vez que todos rótulos silenciosos τ sejam removidos.*

De acordo com a Definição 6, um trace de modelo está relacionado a uma execução que começa a partir de uma transição inicial e progride satisfatoriamente até uma transição final do modelo de processo. Na prática, isso significa que o trace não contém ruído, que é a ocorrência de um desvio na execução de uma instância do modelo de processo. A seguir, é definido o conceito de trace com ruído.

Definição 7 (Trace com ruído [13])**.** *Dado uma rede de Petri rotulada N, um trace com ruído é um trace que não está em conformidade com um modelo de processo N por alguma razão, e geralmente tem uma ocorrência pouco frequente no log de eventos. Esse trace tem um desvio do que é esperado pelo modelo de processo N. Como resultado, existem duas possíveis explicações: ou o modelo N precisa ser ajustado para acomodar de forma correta o trace, ou houveram problemas durante a execução da instância do processo, o que é refletido no log de eventos.*

Por exemplo, em referência ao modelo de processo apresentado na Figura 1.1 e ao log de eventos mostrado na Tabela 2.1, o trace $\langle a, c \rangle$ é classificado como um trace com ruído, enquanto os outros são classificados como traces do modelo.

Redes de Petri rotuladas (Definição 1) podem ser estendidas com recursos estocásticos introduzindo pesos nas transições. Esses pesos podem ser gerados por um estimador [12] ou fornecidos por um especialista no campo. A probabilidade de disparar uma transição habilitada é determinada pelo peso dessa transição em proporção à soma dos pesos de todas as transições habilitadas. Além disso, nessa definição, o foco está nas probabilidades de disparo das transições, portanto, as transições não são divididas em transições temporizadas e imediatas, como na definição de *rede de Petri estocástica generalizada* [28].

Definição 8 (Rede de Petri Rotulada Estocástica [18])**.** *Uma Rede de Petri Rotulada Esto-*

cástica é definida como uma tupla (N, m_s, m_f, W) *que contém:*

- *Uma rede de Petri rotulada* N.
- *Um marking inicial* m_s *e um marking final* m_f
- *Uma função* $W : T \to \mathbb{R}$ *que atribui um peso para cada transição. A probabilidade de disparo de uma transição* t *em um marking* m *corresponde a* $\mathbb{P}(m \xrightarrow{t} m') = \frac{W(t)}{\sum_{t' \in E(m)} W(t')}$.

Um exemplo de uma rede de Petri estocástica é mostrado na Figura 1.1, tal que $P = \{p1, p2, p3, p4\}$, $T = \{t1, t2, t3, t4, t5\}$ e *A={Receive Order, Analyze Order, Accept, Reject,* τ*}* (as letras a, b, c, d foram usadas para encurtar os rótulos, respectivamente). Nesta figura, a probabilidade de disparo é mostrada entre parênteses para cada transição. Além disso, esta figura mostra o *marking* inicial m_s, tal que $m_s(p1) = 1$ e 0 para os outros lugares. Observe que esta rede de Petri estocástica não contém atividades paralelas.

A seguir, é definida uma rede de Petri estocástica k-bound e um *marking* do tipo *deadlock*.

Definição 9 (k-boundedness [3])**.** *Uma rede de Petri estocástica* M *é* k*-bounded se e somente se para cada marking alcançável* m *e para cada lugar* p*,* $m(p) \leq k$*, tal que* $k \in \mathbb{N}$.

Uma rede de Petri é 1-bounded, também chamada de segura, se um lugar nunca contém mais de um *token* no momento.

Definição 10 (Deadlock marking [18])**.** *Um marking* m *de uma rede de Petri estocástica* M *é um deadlock se não existe nenhuma transição habilitada, i.e.,* $E(m) = \emptyset$.

No restante deste trabalho, é feita a suposição de que a rede de Petri estocástica utilizada é 1-bounded. Essa suposição é a mesma utilizada por Leoni e Marrella [22] e Lanciano [17]. Restringir o foco da pesquisa a redes de Petri 1-bounded não impõe restrições significativas, uma vez que a maioria dos processos de negócios podem ser efetivamente representados usando tais redes de Petri, como demonstrado em trabalhos anteriores [22, 17].

Além disso, é assumido que a rede de Petri estocástica não possui *deadlock*, ou seja, a rede sempre é capaz de atingir um *marking* final m_f.

2.2 Tarefa de alinhamento considerando uma rede de Petri determinística

O objetivo da checagem de conformidade é basicamente quantificar os desvios entre um modelo de processo e um log de eventos [20].

Especificamente, a checagem de conformidade baseada em alinhamento alinha um log de eventos com um modelo de processo. A partir desse alinhamento é possível identificar desvios que levam à não conformidade. A criação desse alinhamento é complexa, uma vez que o log de eventos pode desviar do modelo de processo em vários pontos [25].

A seguir, é descrita a formalização da tarefa de alinhamento considerando uma rede de Petri determinística e um trace a ser alinhado pertencente a um log de eventos. Primeiro,

o estado de alinhamento é definido. Esse estado representa o ponto que a rede de Petri e o trace do log de eventos estão, durante a execução da tarefa de alinhamento.

Definição 11 (Estado de Alinhamento[1])**.** *Dado uma rede de Petri N e um trace $\sigma_L \in L$, um estado de alinhamento s é definido pela posição tal que o marking se encontra em N e o próximo evento do trace $e_i \in \sigma_L$ a ser alinhado. Defini-se$\mathcal{S}$ como sendo o conjunto de estados de alinhamento.*

O estado inicial de alinhamento, $s_0 \in \mathcal{S}$, é aquele em que a execução da tarefa de alinhamento começa. Em contraste, um estado de alinhamento final, $s_g \in \mathcal{S}$, é aquele em que a execução da tarefa de alinhamento termina, ou seja, um estado em que tanto o fim do modelo de processo N quanto o evento artificial *evEND* são alcançados. O evento *evEND* representa o próximo evento após o último evento do trace σ_L. Defini-se $\mathcal{G}$ como sendo o conjunto de estados de alinhamento final, de modo que $\mathcal{G} \subset \mathcal{S}$.

Definição 12 (Ação de Movimentação de Alinhamento[2])**.** *Uma ação de movimentação de alinhamento é um movimento de alinhamento que pode ser escolhido durante a execução da tarefa de alinhamento. Existem três tipos de movimentos de alinhamento: síncronos, de log e de modelo.*

Considere que s_L representa uma atividade no trace do log de eventos associada a um evento, s_M representa uma atividade no modelo de processo associada a uma transição, e $\gg$ denota que nenhum movimento ocorreu. Dada uma rede de Petri N e um trace $\sigma_L \in L$, uma ação de movimentação de alinhamento a_t é nomeada de acordo com o tipo de movimento e informações adicionais, da seguinte forma:

- *Para a ação de movimentação síncrona, a informação adicional é o lugar onde o token está em M e o próximo evento $e_i \in \sigma_L$ para alinhar. O resultado da execução dessa ação pode ser representado pelo par (s_L, s_M) tal que $s_L \neq \gg$ e $s_M \in T$ para a mesma atividade.*
- *Para a ação de movimentação de log, a informação adicional é o próximo evento $e_i \in \sigma_L$ para alinhar e o evento $e_{i+1} \in \sigma_L$. O resultado da execução dessa ação pode ser representado pelo par (s_L, s_M) tal que $s_L \neq \gg$ e $s_M = \gg$.*
- *Para a ação de movimentação de modelo, a informação adicional é o lugar tal que o token está em M e a atividade s_M. O resultado da execução dessa ação pode ser representada pelo par (s_L, s_M) tal que $s_L = \gg$ e $s_M \in T$.*

Esses movimentos ocorrem durante a execução da tarefa de alinhamento e são os resposáveis por checar a conformidade entre o modelo de processo e o trace do log de eventos.

Por exemplo, considere a rede de Petri estocástica na Figura 1.1 (sem considerar as probabilidades de disparo) e o trace do log de eventos a ser alinhado $\sigma_L = \langle a, f \rangle$. A Figura

[1] A definição de estado de alinhamento é semelhante aos estados no sistema de transições criados a partir da rede de Petri do produto definida por Adriansyah, Sidorova e Dongen [4] e à definição de estados na abordagem proposta por Leoni e Marrella [22].

[2] Essa definição se baseia na definição de movimento de alinhamento dada por Adriansyah, Sidorova e Dongen [4] e Aalst [2].

2.1 mostra um estado de alinhamento chamado s_4, sendo que o *marking* está em $p3$ na rede de Petri e o próximo evento do trace a ser alinhado é f. Para esse estado de alinhamento, quatro ações de movimento de alinhamento podem ser executadas: moveLog-f-evEND, moveModel-p3-c, moveModel-p3-d e moveModel-p3-τ. Por exemplo, se moveLog-f-evEND for escolhido, o próximo estado será s_7, e se moveModel-p3-c for escolhido, o próximo estado será s_8.

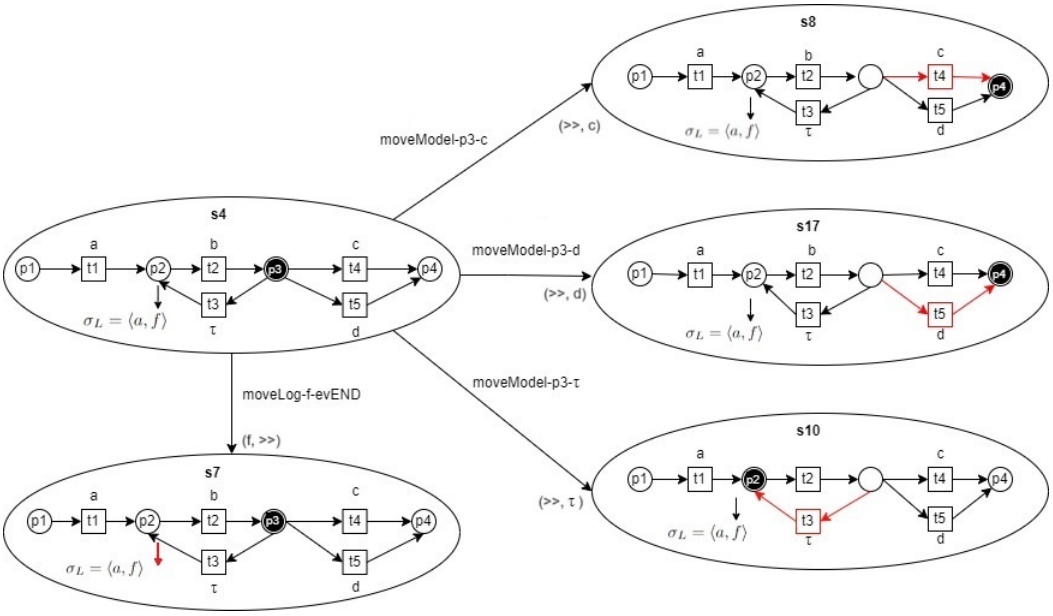

Figura 2.1: *Exemplos de estados de alinhamento e ações de movimento de alinhamento determinístico no processo de alinhamento da Rede de Petri rotulada apresentada na Figura 1.1 (sem considerar as probabilidades de disparo) e o trace do log de eventos $\sigma_L = \langle a, f \rangle$.*

Cada par (s_L, s_M), que representa o resultado da execução da ação de movimento de alinhamento, possui um custo associado a ele.

Definição 13 (Função de custo do par (s_L, s_M) [17])**.** *Dada uma rede de Petri N, um trace $\sigma_L \in L$ e $\mathscr{Y}$ o conjunto de todos os pares (s_L, s_M), uma função de custo c atribui um custo não negativo a cada par: $\mathscr{Y} \rightarrow \mathbb{N}_0^+$.*

Existem diferentes estratégias para determinar a função de custo. Ela pode ser gerada automaticamente com base em um log de eventos [16, 14, 6], pode ser fornecida por um especialista no campo ou pode simplesmente utilizar custos de 0 ou 1. Essa última é chamado de função de custo padrão e é amplamente utilizado na literatura. Na função de custo padrão, o par que representa o resultado de um movimento síncrono e um movimento de modelo de uma transição invisível tem custo 0, caso contrário, o par tem custo 1, ou seja:

$$c(s_L, s_M) = \begin{cases} 0 & s_L = s_M \\ 0 & s_L = \gg \text{ and } s_M = \tau \\ 1 & s_L \neq \gg \text{ and } s_M = \gg \\ 1 & s_L = \gg \text{ and } s_M \in T \end{cases} \tag{2.1}$$

Definição 14 (Processo de alinhamento considerando uma rede de Petri)**.** *No processo de alinhamento de uma rede de Petri determinística N e um trace $\sigma_L \in L$, a cada passo de tempo $t \in \{0, 1, ...\}$, ele se encontra em um estado de alinhamento s_t, uma ação de movimento de alinhamento a_t é escolhida, a qual tem um custo c_t, e ele faz a transição para um estado s_{t+1} até que o processo alcance um estado final s_g.*

Um alinhamento completo (para simplificação, chamado de alinhamento) é composto por uma sequência de pares (s_L, s_M) que representa o resultado da aplicação das ações de movimento de alinhamento e que permite atingir um estado final s_g.

Definição 15 (Alinhamento [22])**.** *Dada uma rede de Petri N, um trace $\sigma_L \in L$, e $\mathscr{Y}$ o conjunto de todos os pares (s_L, s_M), uma sequência $\gamma \in \mathscr{Y}^*$ é um alinhamento de N e σ_L se, ignorando todas as ocorrências de $\gg$, a projeção no primeiro elemento dos pares resulta em σ_L e a projeção no segundo elemento dos pares resulta em uma sequência que pode ser executada na rede de Petri N.*

Por exemplo, considerando a rede de Petri na Figura 1.1 (sem considerar as probabilidades de disparo) e o trace a ser alinhado $\sigma_L = \langle a, f \rangle$, três alinhamentos possíveis (γ_1, γ_2 e γ_3) são apresentados na Figura 2.2. Para cada alinhamento, a primeira linha representa o trace σ_L e a segunda linha representa a execução completa do caminho no modelo de processo dado.

Por exemplo, o primeiro alinhamento é $\gamma_1 = \langle (a, a), (f, \gg), (\gg, b), (\gg, c) \rangle$ e representa o resultado da aplicação de quatro movimentos de alinhamento: syncMove-p1-a, moveLog-f-evEND, moveModel-p2-b e moveModel-p3-c, respectivamente. Esse alinhamento é representado de forma abreviada como $\langle a, f, b, c \rangle$.

Um dos possíveis objetivos da tarefa de alinhamento é encontrar um alinhamento com o menor custo final, chamado de alinhamento ótimo. Portanto, primeiro é definido o custo de um alinhamento com base na função de custo dos pares (s_L, s_M) e, em seguida, um alinhamento ótimo.

Definição 16 (Custo de um alinhamento) [17])**.** *O custo de um alinhamento γ entre o trace σ_L e N é computado como a soma dos custos de todos os pares constituintes (s_L, s_M), i.e., $K(\gamma) = \sum_{(s_L, s_M) \in \gamma} c(s_L, s_M)$.*

Definição 17 (Alinhamento ótimo)**.** *Dado uma rede de Petri N e um trace a ser alinhado $\sigma_L \in L$, um alinhamento ótimo γ_* é aquele que possui o menor custo K.*

Dois exemplos de alinhamentos ótimos para a rede de Petri apresentada na Figura 1.1 (sem considerar as probabilidades de disparo) e o trace a ser alinhado $\sigma_L = \langle a, f \rangle$ são $\gamma_1 = \langle a, f, b, c \rangle$ e $\gamma_2 = \langle a, b, f, c \rangle$ (primeiro e segundo alinhamentos na Figura 2.2). Ambos os alinhamentos têm custo 3, considerando a função de custo padrão dada na Equação 2.1.

$$\gamma_1 = \begin{array}{|c|c|c|c|}\hline \text{a} & \text{f} & \gg & \gg \\ \hline \text{a} & \gg & \text{b} & \text{c} \\ \hline \end{array} \quad \gamma_2 = \begin{array}{|c|c|c|c|}\hline \text{a} & \gg & \text{f} & \gg \\ \hline \text{a} & \text{b} & \gg & \text{c} \\ \hline \end{array} \quad \gamma_3 = \begin{array}{|c|c|c|c|c|}\hline \text{a} & \text{f} & \gg & \gg & \gg \\ \hline \text{a} & \gg & \text{b} & \text{b} & \text{d} \\ \hline \end{array}$$

Figura 2.2: *Exemplo de alinhamentos possíveis dado o trace $\sigma_L = \langle a, f \rangle$ e a rede de Petri N apresentada na Figura 1.1 (sem considerar as probabilidades de disparo). Em cada alinhamento, a primeira linha representa o trace σ_L e a segunda linha representa uma execução completa no modelo de processo N.*

2.3 Abordagens para resolver a tarefa de alinhamento considerando uma rede de Petri determinística

Nesta seção são discutidos trabalhos relacionados à nossa proposta. Existem diferentes abordagens para resolver a tarefa de alinhamentos para redes de Petri determinísticas. No contexto desse trabalho, essas abordagens foram agrupadas em soluções que são baseadas em grafos, sendo essa abordagem utilizada como base para a maioria dos trabalhos da literatura e o grupo dos trabalhos que utilizam planejamento automático.

2.3.1 Abordagem baseada em grafos

A abordagem baseada em grafo do Adriansyah [5] é usada como base para a maioria dos trabalhos na literatura para resolver a tarefa de alinhamento para redes de Petri determinísticas. Esta abordagem utiliza o produto de duas redes de Petri como espaço de busca para o problema de alinhamento. A primeira rede de Petri é a rede de Petri de eventos (que representa o trace a ser alinhado) e a segunda é a rede de Petri que corresponde ao modelo de processo. A rede de Petri produto inclui essas duas redes de Petri com transições síncronas adicionais que são criadas pelo emparelhamento de transições em ambas as redes de Petri com o mesmo rótulo. A abordagem de alinhamento de Adriansyah [5] usa o conceito de movimento de alinhamento junto com uma função de custo padrão. Na rede de produto, os movimentos síncronos são modelados por transições síncronas, os movimentos de log por transições na rede de eventos e os movimentos de modelo por transições na rede de processos. Em seguida, é criado o sistema de transição da rede de Petri do produto. Este sistema de transição também é chamado de grafo de alcançabilidade (*reachability graph*)[5].

2.3.2 Abordagem baseada em planejamento automático

As abordagens baseadas em planejamento [22, 26, 25] para a tarefa de alinhamento de redes de Petri determinísticas e um trace são descritas nesta subseção. Essas abordagens modelam a tarefa como um problema de planejamento determinístico e a codificam usando a linguagem *Linguagem de definição de domínio de planejamento* (PDDL). Utilizando planejamento automático é possível aproveitar uma série de planejadores determinísticos existentes na literatura e utilizar aquele que melhor se adapta ao problema [22, 26, 25].

A primeira abordagem baseada em planejamento para esta tarefa foi proposta por Leoni e Marrella [22]. Leoni e Marrella [22] assumiram que os traces são totalmente ordenados [3].

[3] Um trace totalmente ordenado é composto de eventos que possuem *timestamps* diferentes.

Nos experimentos, eles demonstraram que os métodos existentes não conseguem concluir suas tarefas quando confrontados com modelos de processos excessivamente grandes e traces de log de eventos excepcionalmente longos devido a restrições de memória. Em contraste, a abordagem proposta por Leoni e Marrella [22] realiza com sucesso a tarefa de alinhamento sob essas condições.

M. Leoni e Marrella [25] removeram a suposição de que os traces são totalmente ordenados e nesse caso, trabalham com traces parcialmente ordenados [4]. Uma ferramenta foi desenvolvida por M. Leoni e Marrella [26] que considera traces totalmente ordenados e parcialmente ordenados.

2.4 Checagem de conformidade para redes de Petri estocásticas

Na literatura sobre checagem de conformidade há trabalhos que exploram os modelos de processos estocásticos. No âmbito deste estudo, nosso foco está nas redes de Petri estocásticas. Alguns trabalhos foram desenvolvidos que utilizam modelos estocásticos, como abordagens em descoberta de processos estocásticos [12, 31, 27] e checagem de conformidade estocástica [20, 9, 18, 21].

Nesta subseção, primeiro, são descritas as abordagens para checagem de conformidade que usam redes de Petri estocásticas, mas não tratam da tarefa de alinhamento [20, 18, 21]. Essas abordagens são descritas para demonstrar como as redes de Petri estocásticas estão sendo utilizadas na literatura de Mineração de Processos. Por fim, o único trabalho da literatura que resolve a tarefa de alinhamento para redes de Petri estocásticas [9] é descrito.

2.4.1 Checagem de conformidade

Leemans, Syring e Aalst [20] introduz uma medida de conformidade que leva em consideração os atributos estocásticos do log de eventos e do modelo de processo. A métrica proposta por Leemans, Syring e Aalst [20] baseia-se na métrica de distância *Earth's move* (EMD) e mede o esforço para transformar as distribuições de traces do log de eventos na distribuição de traces do modelo de processo. O trabalho de Leemans, Syring e Aalst [20] foi estendido por Leemans et al. [21] com suporte para atividades silenciosas e duplicadas.

Leemans, Maggi e Montali [18] delineiam três objetivos principais em seu trabalho. A primeira é determinar, dados uma rede de Petri estocástica e um conjunto de *marking* finais, a probabilidade de um trace da rede terminar em um desses *markings*. O segundo objetivo envolve calcular a probabilidade da rede gerar um trace de modelo que satisfaça uma propriedade temporal especificada. O terceiro objetivo é avaliar se uma rede de Petri estocástica rotulada está em conformidade com um conjunto de restrições temporais probabilísticas. Leemans, Maggi e Montali [18] trabalha no conceito temporal dentro

[4] Um trace parcialmente ordenado é composto de eventos que podem ter o mesmo *timestamp*. Desta forma, não é possível saber a ordem correta da execução dos eventos

de redes de Petri estocásticas, reconhecendo a possibilidade de atrasos no disparo das transições.

2.4.2 Tarefa de alinhamento

Bergami et al. [9] realiza a checagem de conformidade considerando uma *workflow net* estocástica e um trace $\sigma_L \in L$ como entrada; e produz como resultado um ranking de alinhamentos. Duas abordagens foram propostas para gerar os rankings: uma abordagem de força bruta e uma abordagem otimizada em termos de tempo.

A abordagem de força bruta gera todos os traces de modelo possíveis $\sigma_M \in \mathcal{M}$, levando em consideração um limite $0 \leq n \leq 1$, o que evita a ocorrência de *loop* infinito. Além disso, é calculado um valor de similaridade entre cada trace do modelo $\sigma_M \in \mathcal{M}$ e o trace que está sendo alinhado $\sigma_L \in \mathcal{L}$. Esta métrica de similaridade é:

$$sim = \frac{1}{\frac{1}{c} + dist(\sigma_L, \sigma_M)}$$

,

Em que *dist* é a métrica de distância chamada distância de Levenshtein (LD) [24] e c é uma constante diferente de zero. Um valor de probabilidade também é atribuído a cada trace do modelo $\sigma_M \in \mathcal{M}$ obtido pela multiplicação de todas as probabilidades de disparo de cada transição que compõem o trace do modelo. Finalmente, estes dois valores são multiplicados, produzindo o valor de probabilidade final exibido no ranking.

Na Tabela 2.2 é mostrado o ranking gerado pela abordagem de força bruta para a rede de Petri estocástica da Figura 1.1 e para o *trace* $\langle a, f \rangle$. Note que nesse exemplo, foram incluídos apenas *traces* de tamanho igual ou menor a 5 permitidos pelo modelo. Considere c = 1. A primeira linha da tabela tem o *trace* $\langle a, b, c \rangle$. A probabilidade dele é $1.0 * 1.0 * 0.5 = 0.5$ e a similaridade é igual a $\frac{1}{4} \approx 0.25$. Multiplicando esses 2 valores, obtemos 0.125, sendo esse o maior valor do ranking.

Trace	Probabilidade	Similaridade	Probabilidade do ranking
$\langle a, b, c \rangle$	0.5	0.25	0.125
$\langle a, b, d \rangle$	0.3	0.25	0.075
$\langle a, b, b, c \rangle$	0.1	0.2	0.020
$\langle a, b, b, d \rangle$	0,06	0.2	0,012
$\langle a, b, b, b, c \rangle$	0.02	0.17	0.003
$\langle a, b, b, b, d \rangle$	0.012	0.17	0.002

Tabela 2.2: *Ranking de alinhamentos gerado pela abordagem de força bruta proposta por Bergami et al. [9]*

A abordagem otimizada em termos de tempo produz um ranking aproximado representando traces como vetores numéricos usando *embedding*. O algoritmo k-Nearest Neighbors (kNN) é então utilizado para encontrar os pontos mais próximos do trace que está sendo alinhado a partir de um conjunto de pontos, usando uma função de distância.

2.5 Problema do Caminho Mais Curto Estocástico e Planejamento Probabilístico

O principal arcabouço teórico usado em planejamento probabilístico é o *Processo de Decisão de Markov* (MDP) [29]. Em MDPs, um agente interage com um ambiente tomando ações com resultados incertos. O *Problema do Caminho Mais Curto Estocástico* (SSP) [10], um tipo de MDP, é empregado para representar situações em que o objetivo de um agente é minimizar o custo acumulado esperado necessário para alcançar um objetivo específico.

Em SSPs, em qualquer momento dado, denotado como t, o agente está no estado s_t e deve selecionar uma ação a_t. Essa ação resulta em uma transição para o estado s_{t+1} com uma distribuição de probabilidade sobre s_t e a_t, incorrendo em um custo de c_t. Esse processo é repetido até que um estado meta seja alcançado.

Definição 18 (Caminho Mais Curto Estocástico.)**.** *Um SSP [10] é uma tupla $\mathcal{M} = \langle \mathcal{S}, s_0, \mathcal{A}, P, c, \mathcal{G} \rangle$, tal que: $\mathcal{S}$ é um conjunto finito de estados; $s_0 \in \mathcal{S}$ é o estado inicial; $\mathcal{A}$ é um conjunto finito de ações; $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ é a função de transição, tal que $P(s, a, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$; $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ é a função de custo representando o custo de executar uma ação a no estado s e que leva a um estado s'; $\mathcal{G} \subset \mathcal{S}$ é o conjunto de estados meta, i.e., $c(g, a) = 0$ e $P(g, a, g) = 1, \forall g \in \mathcal{G}$ e $a \in \mathcal{A}$.*

O objetivo de um SSP é encontrar uma política π, um mapeamento de estados para ações, que minimiza o custo acumulado esperado para chegar na meta $V^{\pi}(s) = \lim_{T \to \infty} \mathbb{E}\left[\sum_{t=0}^{T-1} c_t \mid \pi, s_0 = s\right]$.

A partir dessa equação, é possível obter o seguinte sistema de equações lineares para avaliar a política π [29]:

$$V^{\pi}(s) = \begin{cases} 0, & \text{se } s \in \mathcal{G} \\ \sum_{s' \in \mathcal{S}} \left[c(s, \pi(s), s') + P(s, \pi(s), s') V^{\pi}(s')\right], & \text{Caso contrário.} \end{cases} \tag{2.2}$$

π_* é uma política ótima se:

$$V^{\pi_*}(s) \leq V^{\pi}(s) \quad \forall \pi \in \Pi \text{ e } \forall s \in \mathcal{S}.$$

Seja V^* a função valor associada com a política ótima π_*. V^* é chamada de função valor **ótima** e pode ser definida em termos do seguinte sistema de equaçôes de Bellman. [8]:

$$V^*(s) = \begin{cases} 0, & \text{se } s \in \mathcal{G} \\ \min_{a \in \mathcal{A}} \left\{ \sum_{s' \in \mathcal{S}} \left[c(s, a, s') + P(s, a, s') V^*(s')\right] \right\}, & \text{Caso contrário.} \end{cases} \tag{2.3}$$

Alternativamente, em vez de avaliar um estado s, é possível considerar um par estado-ação (s, a) usando a função valor Q. Dessa forma, $Q^*(s, a)$ é denominado como a função

valor Q ótima, representando a escolha ótima de tomar a ação a no estado s:

$$Q^*(s,a) = \begin{cases} 0, & \text{se } s \in \mathcal{G} \\ \sum_{s' \in S} \left[c(s,a,s') + P(s,a,s') V^*(s') \right], & \text{Caso contrário.} \end{cases} \tag{2.4}$$

Portanto, a política ótima π_* pode ser obtida para um estado s selecionando a ação que minimiza $Q^*(s,a)$:

$$\begin{aligned} \pi_*(s) &= \arg\min_{a \in \mathcal{A}} \left\{ \sum_{s' \in S} \left[c(s,a,s') + P(s,a,s') V^*(s') \right] \right\} \\ \pi_*(s) &= \arg\min_{a \in \mathcal{A}} \{ Q^*(s,a) \}. \end{aligned} \tag{2.5}$$

Existem dois algoritmos na literatura amplamente utilizados para encontrar uma política ótima, que são o algoritmo de iteração de valor e o algoritmo de iteração de política [10].

Em planejamento probabilístico, domínios e problemas podem ser representados fazendo uso da *Linguagem de Definição de Domínio de Planejamento Probabilístico* (PPDDL) [33]. O PPDDL utiliza dois arquivos, o arquivo de descrição do problema e o arquivo de descrição do domínio.

Definição 19 (Descrição do problema.)**.** *Considere $\mathcal{P}_D = (s_0, \mathcal{G}, \mathcal{O})$, tal que s_0 é a descrição do estado inicial, $\mathcal{G}$ é a descrição dos estados meta desejados e $\mathcal{O}$ são os objetos que compõem o problema.*

Definição 20 (Descrição do domínio.)**.** *Considere $\mathcal{D}_D = (\mathcal{A}, \mathcal{P}, \mathcal{T})$ como uma descrição de domínio, em que $\mathcal{A}$ representa as ações com efeitos probabilísticos, $\mathcal{P}$ é um conjunto de proposições que descrevem os estados do problema e $\mathcal{T}$ representa os tipos de objetos.*

Uma ação $a \in \mathcal{A}$ pode ser representada como $a = (Par_a, Pre_a, Eff_a)$, em que Par_a é a lista de parâmetros de entrada, Pre_a são as condições prévias para executar essa ação e Eff_a são os efeitos probabilísticos de executar a ação a. Tanto as condições prévias quanto os efeitos são declarados em termos das proposições em $\mathcal{P}$ em $\mathcal{D}_D$.

Capítulo 3

Tarefa de alinhamento considerando uma rede de Petri estocástica

Neste capítulo, é proposta uma formalização da tarefa de alinhamento que recebe como entrada uma rede de Petri estocástica e um trace a ser alinhado, emprega o conceito de movimento de alinhamento e tem como objetivo produzir um ranking de alinhamentos a partir de uma política ótima e que explique as não conformidades. Para obter esse ranking, é usada uma métrica de otimização que combina tanto a probabilidade de disparo das transições na rede de Petri estocástica quanto o custo dos alinhamentos. Essa formalização é parcialmente inspirada na formalização de Adriansyah [5], que serve como base para a maioria dos trabalhos na literatura para resolver a tarefa de alinhamento em redes de Petri determinísticas.

Ao trabalhar com redes de Petri estocásticas, a noção de estados de alinhamento é a mesma que a apresentada na Seção 2.2. No entanto, o processo de alinhamento precisa ser modificado para levar em consideração as probabilidades de disparo, especificamente, a definição da ação de movimentação do modelo precisa ser modificada.

Definição 21 (Ação de movimento de alinhamento estocástico)**.** *Uma ação de movimento de alinhamento estocástico é um movimento de alinhamento que pode ser escolhido durante a execução da tarefa de alinhamento. Existem três tipos de movimentos de alinhamento: síncrono, de log e de modelo.*

Considerando que s_L representa uma atividade no trace do log de eventos associada a um evento, s_M representa uma atividade no modelo de processo associada a uma transição e $\gg$ denota que nenhum movimento ocorreu, dada uma rede de Petri estocástica M e um trace $\sigma_L \in L$, uma ação de movimento de alinhamento estocástico a_t é nomeada pelo tipo do movimento e informações adicionais, da seguinte forma:

- *Para as ações de movimentação síncrona e de log, as informações adicionais e a representação do resultado da execução dessa ação são as mesmas conforme definidas na Definição 12.*

- *Para a ação de movimentação de modelo estocástico, as informações adicionais se referem ao local onde o token está em M. O resultado da execução dessa ação depende da probabilidade de disparo da transição habilitada e pode ser representado pelo par (s_L, s_M), onde $s_L =\gg$ e $s_M \in T$.*

Seja $\mathcal{A}$ o conjunto de todas as ações de movimentação de alinhamento estocástico.

O processo de alinhamento considerando uma rede de Petri estocástica é semelhante ao processo de alinhamento considerando uma rede de Petri determinística, exceto que precisamos levar em consideração as probabilidades de disparo da ação de movimentação de modelo estocástico.

Definição 22 (Processo de Alinhamento considerando uma Rede de Petri Estocástica)**.** *No processo de alinhamento, em qualquer etapa de tempo $t \in \{0, 1, ...\}$, ele se encontra em um estado de alinhamento s_t, escolhe-se um movimento de alinhamento a_t que tem um custo c_t e faz a transição para um estado s_{t+1} considerando as probabilidades de disparo (se o movimento de alinhamento a_t escolhido for um movimento de modelo) e de acordo com uma estratégia ad hoc, até que o alinhamento seja concluído.*

Por exemplo, considere a rede de Petri estocástica na Figura 1.1 e o *trace* a ser alinhado $\sigma_L = \langle a, f \rangle$.

A Figura 3.1 mostra um estado de alinhamento chamado s_4, onde o *token* está em $p3$ na rede de Petri estocástica e o próximo evento do trace a ser alinhado é f. Para esse estado de alinhamento, duas ações de movimento de alinhamento estocástico podem ser executadas: moveLog-f-evEND e moveModel-p3. Se moveLog-f-evEND for escolhido, o próximo estado será s_7. Se moveModel-p3 for escolhido, o próximo estado pode ser s_8, s_{17} ou s_{10} com probabilidades de 0.5, 0.3 e 0.2, respectivamente.

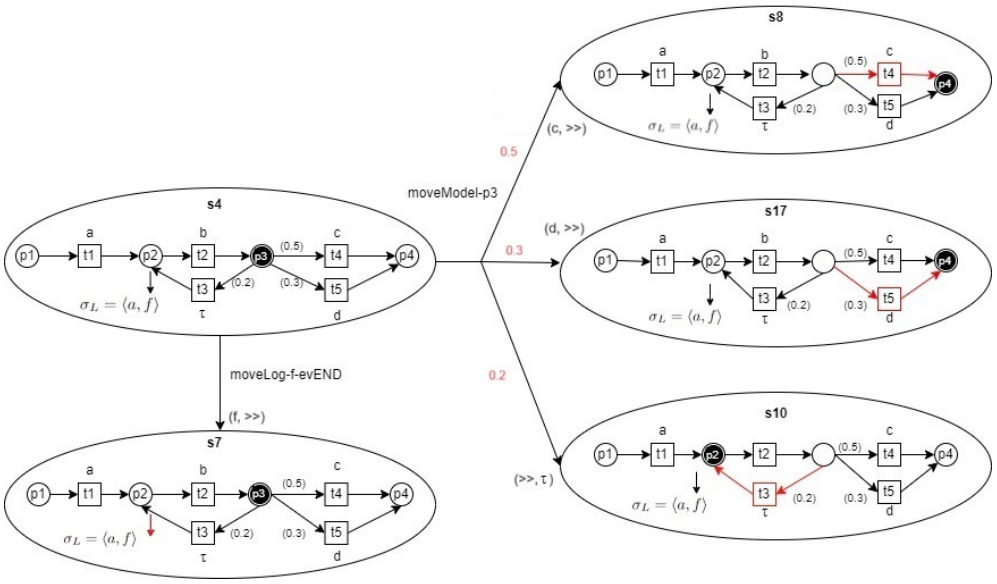

Figura 3.1: *Exemplos de estados de alinhamento e ações de movimento de alinhamento estocástico no processo de alinhamento da rede de Petri estocástica apresentada na Figura 1.1 e o trace $\sigma_L = \langle a, f \rangle$.*

Em seguida, uma política de alinhamento é definida. Para cada possível estado de

alinhamento na tarefa de alinhamento, uma política escolhe uma ação de alinhamento estocástico.

Definição 23 (Política de alinhamento)**.** *Uma política de alinhamento* $\pi : \mathcal{S} \to \mathcal{A}$ *é um mapeamento entre cada estado* s *e uma ação de movimentação de alinhamento estocástico* $a = \pi(s)$*. Seja* Π *o conjunto de políticas de alinhamento.*

Por exemplo, considere a rede de Petri estocástica na Figura 1.1 e o *trace* a ser alinhado $\sigma_L = \langle a, f \rangle$. O número de estados de alinhamento possíveis é 23. Uma política para esse problema de alinhamento deve escolher uma ação de movimentação de alinhamento estocástico para cada um desses estados de alinhamento. Por exemplo, para o estado de alinhamento s_4 apresentado na Figura 3.1, uma política π_1 poderia escolher a ação de alinhamento moveLog-f-evEND, ou seja, moveLog-f-evEND= $\pi_1(s_4)$.

Para comparar diferentes políticas de alinhamento, é definida uma função de valor para elas. Note que uma política π pode gerar diversos alinhamentos devido ao fato de que vários estados de alinhamento são possíveis quando uma ação de movimentação de modelo estocástico é escolhida, dependendo de uma probabilidade de disparo. Portanto, para calcular o valor $V^{\pi}(s)$, as probabilidades de disparo e o custo dos pares (s_L, s_M) dos alinhamentos devem ser considerados.

Definição 24 (Valor de uma política de alinhamento)**.** *Dado um estado de alinhamento* $s \in \mathcal{S}$*, o valor de uma política de alinhamento* π *para o estado de alinhamento* s *é o custo acumulado esperado para concluir o alinhamento a partir de* s*, ou seja:*

$$V^{\pi}(s) = \lim_{T \to \infty} \mathbb{E}\left[\sum_{t=0}^{T-1} c_t \mid \pi, s_0 = s\right] \tag{3.1}$$

$$= \mathbb{E}\left[\sum_{(s_L, s_M) \in \gamma} c(s_L, s_M) \mid \pi, s_0 = s\right] = \mathbb{E}\left[K(\gamma) \mid \pi, s_0 = s\right] \tag{3.2}$$

Observe que na Definição 24, a soma do custo de todos os pares constituintes do alinhamento γ é representada por $K(\gamma)$ (Definição 16). O objetivo é encontrar uma política de alinhamento ótima.

Definição 25 (Política de alinhamento ótima.)**.** π_* *é uma política de alinhamento ótima se:*

$$V^{\pi_*}(s) \leq V^{\pi}(s) \quad \forall \pi \in \Pi \; e \; \forall s \in \mathcal{S}.$$

Seja V^* a função de valor associada à política de alinhamento ótima π_*. V^* é chamada de função de valor **ótima** e corresponde ao menor custo acumulado esperado para concluir o alinhamento a partir do estado de alinhamento s.

Por exemplo, considere a rede de Petri estocástica na Figura 1.1, o *trace* a ser alinhado $\sigma_L = \langle a, f \rangle$ e o estado s_4 apresentado na Figura 3.1. Uma política ótima π_* escolhe moveLog-f-evEND para o estado s_4, e o valor ótimo para este estado é $V^*(s_4) = 2.5$, o que corresponde

ao menor custo acumulado esperado para concluir o alinhamento a partir do estado s_4. No estado s_7, uma política ótima π_* escolhe moveModel-p3 e o valor ótimo para este estado é $V^*(s_7) = 1.5$, o que corresponde ao menor custo acumulado esperado para concluir o alinhamento a partir do estado s_7.

Uma vez que π_*, como qualquer política, pode gerar vários alinhamentos, é possível estabelecer um ranking entre eles.

Definição 26 (Ranking de alinhamentos ótimos)**.** *A partir de uma política de alinhamento ótima π_*, é possível gerar um conjunto de alinhamentos simulando essa política n vezes. Pode-se ordenar esses alinhamentos a partir da frequência de ocorrência de cada um. Esse ranking de alinhamentos é chamado de ranking de alinhamentos ótimos.*

Observação 1. *Se as probabilidades de disparo na rede de Petri estocástica forem todas iguais a 1, ou seja, se for uma rede de Petri determinística, o ranking de alinhamentos ótimos terá apenas um alinhamento. Este alinhamento corresponde a um alinhamento ótimo apresentado na Definição 17 para redes de Petri determinísticas.*

Capítulo 4

Representando a tarefa de alinhamento como um problema de caminho mais curto estocástico e codificando-o em PPDDL

A tarefa de alinhamento considerando uma rede de Petri estocástica formalizada no capítulo 3 pode ser modelada como um SSP $\langle \mathcal{S}, s_0, \mathcal{A}, P, c, \mathcal{G} \rangle$, em que:

- $\mathcal{S}$ é um conjunto finito de estados de alinhamento, dado na Definição 11;
- $s_0 \in \mathcal{S}$ é o estado de alinhamento inicial;
- $\mathcal{A}$ é um conjunto finito de ações de movimento de alinhamento estocástico, dado na Definição 21;
- $P(s, a, s')$ é igual a um para ações síncronas e de movimentação de log. Para as ações de movimento do modelo estocástico, esta probabilidade é igual às probabilidades de disparo das transições na rede de Petri estocástica;
- $c(s, a, s')$ é definido com base na função de custo dos pares (s_L, s_M) dada na Definição 13;
- $\mathcal{G} \subset \mathcal{S}$ é o conjunto de estados meta.

A seguir, é descrito como o problema de encontrar um ranking de alinhamentos pode ser codificado como um problema de planejamento e domínio utilizando a linguagem PPDDL, dado um trace σ_L a ser alinhado e uma rede de Petri estocástica M. Assim, este problema pode ser resolvido usando qualquer planejador probabilístico disponível na literatura.

A codificação proposta é bastante semelhante à proposta por Leoni e Marrella [22] que usa uma rede de Petri determinística como entrada e retorna um alinhamento ótimo. Em [22] os movimentos de alinhamento foram definidos como ações de planejamento no arquivo de domínio. No *ProbPlanAlign*, as principais mudanças são a forma como os

movimentos do modelo estocástico estão sendo representados, uma vez que o modelo de processo agora é uma rede de Petri estocástica e portanto é necessário modelar essas ações utilizando efeitos probabilísticos. As movimentações de log e síncrona também foram alteradas para facilitar a extração da atividade que foi executada pela ação.

4.1 Descrição do Problema

No arquivo do problema $\mathcal{P}_D$, existem três tipos de objetos:

- **place** representa os lugares na rede de Petri estocástica.
- **execActivity** representa qual foi a atividade executada para uma determinada ação. As atividades possíveis estão no conjunto composto pela união das atividades apresentadas no modelo (incluindo as transições invisíveis) e as atividades apresentadas no trace σ_L.
- **event** representa o evento que está sendo alinhado para o trace σ_L.

Por exemplo, para o trace $\sigma_L = \langle a, f \rangle$ e a rede de Petri estocástica na Figura 1.1, os seguintes objetos são introduzidos no problema de planejamento probabilístico:

```
{: objects p1 p2 p3 p4 - place
           a b c d f inv -  execActivity
           ev1 ev2 evEND - event
}
```

Três objetos do tipo *event* foram definidos porque o *trace* $\sigma_L = \langle a, f \rangle$ possui dois eventos e é necessário incluir o `evEND` para indicar o fim do trace. O número de eventos depende do número de atividades em σ_L, portanto para cada trace diferente a ser alinhado é criado um número diferente de eventos. Isto é necessário para acompanhar qual atividade está sendo alinhada no momento. Observe que a transição invisível (`inv`) e a atividade f (que não é apresentada no modelo) foram explicitamente introduzidas como *execActivity*.

Os predicados booleanos são:

- **tracePointer.** O predicado `(tracePointer ?e - evento)` é válido para o evento $e \in \sigma_L$ quando e é o próximo evento a ser alinhado.
- **token**. Para cada lugar $p \in P$, `(token ?p - lugar)` vale se e somente se p contém um *token* no *marking* atual.
- **executedActivity**. Para cada atividade $u \in T \cup \sigma_L$, `(executedActivity ?u - execActivity)` vale se e somente se a atividade executada for aquela relacionada ao efeito da ação que está sendo executada.

Para o *trace* $\sigma_L = \langle a, f \rangle$ e a rede de Petri estocástica na Figura 1.1, o estado inicial s_0 é representado como:

```
(:init
    (tracePointer ev1)
    (token p1)
)
```

Quando um alinhamento completo é encontrado para o trace σ_L, o `token` deve estar no lugar `p4` e o `tracePointer` deve estar em `evEND`. Assim, os estados meta $\mathcal{G}$ são representados como:

```
(:goal
    (and
    (not (token p1))
    (not (token p2))
    (not (token p3))
    (token p4)
    (tracePointer evEND)
))
```

Ao lidar com PPDDL, minimizar o custo é o mesmo que maximizar a recompensa. Portanto, a função de custo padrão c (Equação 2.1) foi adaptada para usar uma recompensa r, como segue:

$$r(s_L, s_M) = \begin{cases} 0 & s_L = s_M \\ 0 & s_L =\gg \text{ and } s_M = \tau \\ -1 & s_L \neq\gg \text{ and } s_M =\gg \\ -1 & s_L =\gg \text{ and } s_M \in T \end{cases} \tag{4.1}$$

Observe que diminuir a recompensa em 1 é o mesmo que aumentar a recompensa em -1. Desta forma, a métrica pode ser definida como `(:metric maximize (reward))` em PPDDL.

4.2 Descrição do domínio

A descrição do domínio $\mathcal{D}_D$ é composta pelas ações, pelo conjunto de preposições e pelos tipos de objetos. Os objetos foram introduzidos anteriormente e são `place`, `execActivity` e `event`. Existem três tipos de ações em $\mathcal{D}_D$, movimentos síncronos, movimentos de modelo e movimentos de log. Observe que todas as ações descritas a seguir compartilham o mesmo efeito, ou seja, configurando todas as `executedActivity` para false e então definindo a atividade atual `executedActivity` para verdadeiro. Definir todas as atividades como falsas é necessário para capturar a atividade que está sendo executada atualmente e não aquelas que foram executadas anteriormente. Além disso, todas as ações não possuem nenhuma `executedActivity` como uma precondição.

- **Movimento síncrono.** Uma ação separada desse tipo é criado para cada par $t \in T$ e evento $e \in \sigma_L$ tal que $\ell(t) = \lambda_N(e)$.

  Por exemplo, considere a rede de Petri estocástica apresentada na Figura 1.1 e o trace $\sigma_L = \langle a, f \rangle$. Um movimento síncrono quando o `token` da rede de Petri estocástica está no `place p1` e o próximo evento a ser alinhado do trace σ_L é `ev1` (que corresponde a atividade a) é representada pela seguinte ação:

  ```
  (:action moveSync-p1-ev1
  :precondition (and  (token p1)
  ```

```
(tracePointer ev1))
:effect (and (not (executedActivity a))
        (not (executedActivity b))
        (not (executedActivity c))
        (not (executedActivity d))
        (not (executedActivity inv))
        (not (executedActivity f))
        (not (tracePointer ev1))
(tracePointer ev2)
(probabilistic 1 (and (not (token p1))
(token p2) (executedActivity a)))))
```

As precondições dessa ação são: (i) existe um token no lugar `p1`; (ii) `ev1` é o próximo evento de σ_L a ser alinhado. Depois de aplicar essa ação, os efeitos são: (i) todas as fluentes do tipo `executedActivity` são ajustadas para false; (ii) apenas a atividade que foi executada com essa ação (atividade `a`) é ajustada para true; (iii) o tracePointer se move de `ev1` para `ev2`; e (iv) o token se move de `p1` para `p2`.

- **Movimento de modelo.** Uma ação desse tipo é criada para cada lugar $p \in \mathcal{P}$. Um movimento de modelo faz uma transição habilitada `t` ser executada sem ocorrer nenhuma movimentação no trace σ_L.

  Por exemplo, considere a ação de mover `p3` no modelo de processo da Figura 1.1. Esse movimento de modelo é representado pela seguinte ação:

```
(:action moveModel-p3
:precondition (token p3)
:effect (and (not (executedActivity a))
        (not (executedActivity b))
        (not (executedActivity c))
        (not (executedActivity d))
        (not (executedActivity inv))
        (not (executedActivity f))
(probabilistic
    0.5 (and (not (token p3))
        (token p4) (executedActivity c)
        (decrease (reward) 1))
    0.3 (and (not (token p3))
        (token p4) (executedActivity d)
        (decrease (reward) 1))
    0.2 (and (not (token p3))
        (token p2) (executedActivity inv))
)))
```

As precondições dessa ação são: (i) existe um token em `p3`. Depois de aplicar essa ação, os efeitos são: (i) todas as fluentes do tipo `executedActivity` são ajustadas para false; (ii) com probabilidade de 0.5 ajusta a fluente `executedActivity c` para true, o token se move de `p3` para `p4` e a recompensa é decrescida em 1; (iii) com

probabilidade de 0.3 ajusta a fluente `executedActivity d` para true, o token move-se de `p3` para `p4` e a recompensa é decrescida em 1; (iv) com probabilidade de 0.2 ajusta a fluente `executedActivity inv` para true e o token move-se de `p3` para `p2`.

- **Movimento de log.** Uma ação desse tipo é criada para cada evento $e \in \sigma_L$.

  Por exemplo, considere o evento a do trace $\sigma_L = \langle a, f \rangle$:

```
(:action moveLog-a-ev1-ev2
:precondition (tracePointer ev1)
:effect (and (not (executedActivity a))
        (not (executedActivity b))
        (not (executedActivity c))
        (not (executedActivity d))
        (not (executedActivity inv))
        (not (executedActivity f))
        (not (tracePointer ev1))
(tracePointer ev2) (executedActivity a)
(decrease (reward) 1) ))
```

  A precondição é que `ev1` seja o evento atual de σ_L. Os efeitos são: (i) o *tracePointer* move-se de *ev1* para *ev2*; (ii) todas as fluentes do tipo `executedActivity` são ajustadas para false; (iii) somente a atividade que foi executada com essa ação (atividade `a` e que corresponde ao `ev1`) é ajustada como true; e (iv) a recompensa é decrescida em 1.

4.3 Hiper-grafo de transição de estado da tarefa de alinhamento

Dado um modelo de processo estocástico M e o trace a ser alinhado σ_L, a tarefa de alinhamento codificada em PPDDL, assim como qualquer problema de planejamento probabilístico, pode ser representado como um hiper-grafo de transição de estado. Nesse grafo, os nós são os estados definidos pelos valores dos predicados `token`, `tracePointer` e `executedActivity`, esses estados são chamados de **estados de alinhamento**. As hiper-arestas são ações de movimentação estocástica que podem ser movimentações síncronas, de modelo ou de log.

A Figura 4.1 mostra o hiper-grafo de transição de estado para o modelo de processo estocástico apresentado na Figura 1.1 e o trace a ser alinhado $\sigma_L = \langle a, f \rangle$. Existem 23 estados de alinhamento. Observe que os predicados negados são omitidos nos estados de alinhamento neste grafo. No caso de ações com efeitos probabilísticos, cada ramo é rotulado pela probabilidade dada no modelo de processo estocástico. Caso contrário, esta probabilidade é omitida. O estado inicial é s_0 e os estados meta são s_{20}, s_{21} e s_{22}.

Por exemplo, considere que o estado de alinhamento atual é s_{16} em que `(token p3)`, `(tracePointer evEND)` e `(executedActivity b)` são verdadeiros. Isto significa que neste estado de alinhamento, existe um token no lugar `p3` na rede de Petri estocástica, e o `tracePointer` está em `evenEND` que corresponde ao evento final no trace σ_L.

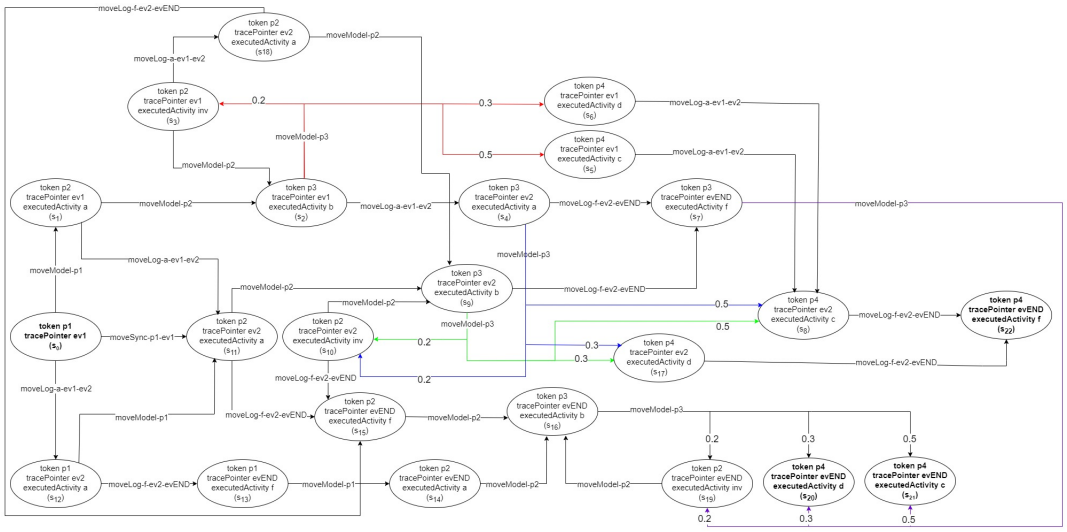

Figura 4.1: *Hiper-grafo de transição de estado criado codificando a tarefa de alinhamento em PPDDL para o modelo de processo estocástico apresentado na Figura 1.1 e o trace a ser alinhado* $\sigma_L = \langle a, f \rangle$

Como as precondições da ação `moveModel-p3` são cumpridas, a execução dessa ação produz os seguintes efeitos: (i) o token no lugar `p3` é consumido e um token é produzido no lugar `p4` ou em `p2`; (ii) a fluente `executedActivity` é definida como falso para todas as atividades possíveis; (iii) a fluente `executedActivity` é definida como verdadeiro para a atividade executada que pode ser `c`, `d` ou `inv`; (iv) a recompensa total é a mesma quando a atividade executada é `inv` ou decresce em 1 nos outros casos. Assim, de acordo com a descrição da ação `moveModel-p3`, com uma probabilidade de 0,2, o próximo estado de alinhamento é s_{19}. Com uma probabilidade de 0,3 o próximo estado é s_{20} e com uma probabilidade de 0,5 é s_{21}. Nos estados s_{20} e s_{21}, `(token p4)` e `(tracePointer evEND)` são verdadeiros, portanto, ambos são estados meta.

4.4 Modelando atividades que ocorrem em paralelo

Vale ressaltar também que esta abordagem probabilística consegue lidar satisfatoriamente com atividades que ocorrem paralelamente no modelo, assim como a abordagem que usa planejamento determinístico proposta por Leoni e Marrella [22].

Uma instância de rede de Petri estocástica com atividades paralelas é mostrada na Figura 4.2. Para este caso, se o modelo possui um token em p3, a transição é habilitada e pode ser disparada com probabilidade 0,6, resultando no consumo do token em p3 e na geração de dois tokens, um no local p4 e outro em p5 . Observe que neste caso não se trata de uma escolha exclusiva entre p4 ou p5; assim, ambas as transições c e d tornam-se habilitadas.

Por exemplo, considerando o modelo de processo na Figura 4.2 e o trace $\sigma_L = \langle a, f \rangle$, a ação de alinhamento de `moveModel-p3` é modelada da seguinte forma:

```
(:action moveModel-p3
:precondition (token p3)
:effect (and (not (executedActivity a))
```

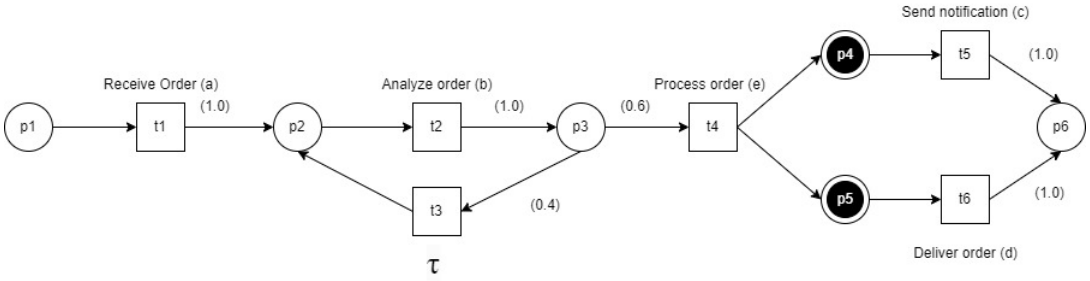

Figura 4.2: *Exemplo de rede de Petri estocástica com atividades paralelas.*

```
        (not (executedActivity b))
        (not (executedActivity c))
        (not (executedActivity d))
        (not (executedActivity e))
        (not (executedActivity inv))
        (not (executedActivity f))
(probabilistic
    0.6 (and (not (token p3))
        (token p4) (token p5) (executedActivity e)
        (decrease (reward) 1))
    0.4 (and (not (token p3))
        (token p2) (executedActivity inv))
)))
```

Capítulo 5

Estratégia de alinhamento

Esse capítulo descreve o *ProbPlanAlign*, a primeira abordagem que usa uma rede de Petri estocástica e um trace de um log de eventos como entrada, emprega o conceito de movimento de alinhamento estocástico e produz um ranking de alinhamentos utilizando uma política ótima. Esta abordagem modela a tarefa de alinhamento como um problema do *caminho mais curto estocástico* e o codifica usando PPDDL. A Figura 5.1 mostra uma visão geral desta abordagem. *ProbPlanAlign* recebe como entrada uma rede de Petri estocástica e um *trace* para alinhar. Depois disso, a tarefa de alinhamento é codificada em PPDDL conforme descrito no Capítulo 4. Assim, são gerados os arquivos de domínio e de problema. Um par diferente de arquivos de domínio e de problema é criado para cada *trace*. Em seguida, esses arquivos são usados como entrada para um planejador probabilístico. É importante notar que *ProbPlanAlign* pode utilizar qualquer planejador probabilístico disponível na literatura.

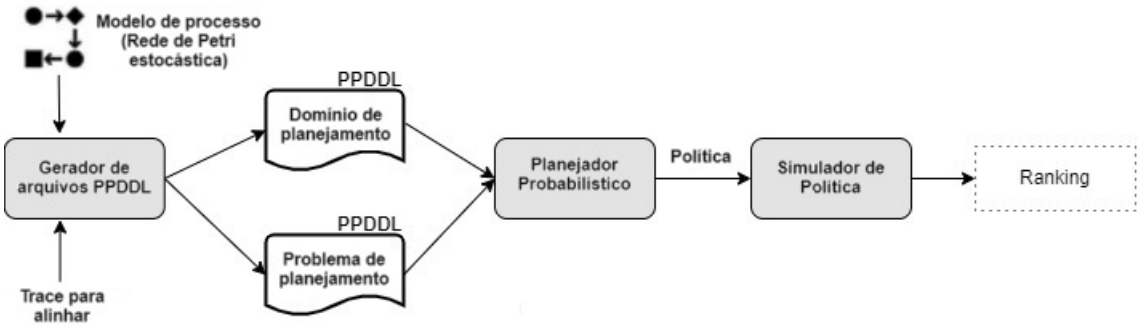

Figura 5.1: *Uma visão geral da abordagem* ProbPlanAlign *que gera um ranking de alinhamentos.*

A saída do planejador probabilístico é uma política de alinhamento ótima, ou seja, para cada estado de alinhamento s ele retorna uma ação de alinhamento com o menor custo acumulado esperado para completar o alinhamento de s. Lembre-se que os estados do problema de planejamento probabilístico correspondem a estados de alinhamento que são definidos pelos valores dos predicados `token`, `tracePointer` e `executedActivity`.

Um alinhamento pode ser encontrado simulando a política de alinhamento π desde o estado inicial $s_0 \in \mathcal{S}$ até atingir um estado meta $g \in \mathcal{G}$. Para gerar um ranking para

um trace σ_L, o *ProbPlanAlign* simula a política *n* vezes, assim é possível obter múltiplos alinhamentos possíveis para o mesmo trace σ_L. Após simular a política, os alinhamentos são ordenados do mais frequente para o menos frequente.

5.1 Exemplo e simulação de uma política ótima

Considere o hiper-grafo de transição de estado dado na Figura 4.1 criado pela codificação da tarefa de alinhamento para o modelo de processo estocástico apresentado na Figura 1.1 e o trace a ser alinhado $\sigma_L = \langle a, f \rangle$. A função Q^*-valor ótima (Equação 2.4) para cada estado de alinhamento do par e ação de movimento de alinhamento é mostrada na terceira coluna da Tabela 5.1. Por exemplo, para o estado de alinhamento s_4 onde temos os predicados `(token p3)`, `(tracePointer ev2)` e `(executedActivity a)`, as ações que podem ser aplicadas são `moveModel-p3` e `moveLog-f-ev2-evEND`. O planejador, que encontra a política que minimiza o custo acumulado esperado para a meta, escolhe `moveLog-f-ev2-evEND` como a melhor ação para este estado, pois possui o melhor valor Q^*, ou seja, $\pi_*(s_4)$ =`moveLog-f-ev2-evEND`. Observe que na Tabela 5.1, para os estados $s_1, s_2, s_3, s_9, s_{10}, s_{11}, s_{12}, s_{18}$, existem várias ações ótimas. Por exemplo, no estado s_1, existem duas ações possíveis: `moveModel-p2` e `moveLog-a-ev1-ev2`. Ambas as ações têm o mesmo custo acumulado esperado de 4,5. Isso significa que existe mais de uma política ótima para este problema (2^8=256 políticas ótimas, onde 8 é o número de estados que possuem duas ações com o mesmo custo). Observe que em geral as implementações de algoritmos de planejamento probabilístico escolhem uma dessas políticas para ser retornada.

Uma possível política completa π_* que poderia ser retornada por um planejador probabilístico é apresentada na quarta coluna da Tabela 5.1. Considerando esta política π_* e n=100, o simulador de política poderia retornar o alinhamento $\langle a, f, b, c \rangle$ como o mais frequente e $\langle a, f, b, b, b, b, b, c \rangle$ como o menos frequente . Por exemplo, o trace $\langle a, f, b, c \rangle$, que é a opção mais bem classificada, pode ser obtido da seguinte forma usando esta política π_*:

- A partir do estado inicial s_0, `moveSync-p1-ev1` é executado, levando ao estado s_{11}.
- Do estado s_{11}, `moveLog-f-ev2-evEND` é executado, levando ao estado s_{15}.
- Do estado s_{15}, `moveModel-p2` é executado, levando ao estado s_{16}.
- Do estado s_{16} `moveModel-p3` é executado, levando ao estado final s_{21}.

O trace $\langle a, f, b, d \rangle$, também pode ser obtido simulando esta política π_* seguindo a mesma sequência mostrada anteriormente, exceto o último passo, que é substituído por:

- Do estado s_{16} `moveModel-p3` é executado, levando ao estado final s_{20}.

Observe que, a partir do mesmo estado de alinhamento s_{16}, é possível ir para diferentes estados de alinhamento (s_{20} e s_{21}) devido às probabilidades de disparo quando a ação de movimento estocástico `moverModelo-p3` é escolhida.

Já que em π_*, `moveModel-p3` é a melhor ação para o estado de alinhamento s_{11} onde `(token p2)`, `(tracePointer ev2)` e `executedActivity a` são verdadeiros, o alinhamento $\langle a, b, f, c \rangle$ não será gerado pelo simulador de política usando esta política, mas pode

Estado de alinhamento	Ação de movimento de alinhamento	$Q^*(s,a)$	$\pi_*(s)$
s_0	moveSync-p1-ev1	3.5	
	moveLog-a-ev1-ev2	5.5	moveSync-p1-ev1
	moveModel-p1	5.5	
s_1	moveModel-p2	4.5	*moveModel-p2*
	moveLog-a-ev1-ev2	4.5	
s_2	moveModel-p3	3.5	*moveModel-p3*
	moveLog-a-ev1-ev2	3.5	
s_3	moveModel-p2	4.5	*moveModel-p2*
	moveLog-a-ev1-ev2	4.5	
s_4	moveModel-p3	3.25	moveLog-f-ev2-evEND
	moveLog-f-ev2-evEND	2.5	
s_5	moveLog-a-ev1-ev2	2	moveLog-a-ev1-ev2
s_6	moveLog-a-ev1-ev2	2	moveLog-a-ev1-ev2
s_7	moveModel-p3	1.5	moveModel-p3
s_8	moveLog-f-ev2-evEND	1	moveLog-f-ev2-evEND
s_9	moveModel-p3	2.5	*moveModel-p3*
	moveLog-f-ev2-evEND	2.5	
s_{10}	moveLog-f-ev2-evEND	3.5	*moveModel-p2*
	moveModel-p2	3.5	
s_{11}	moveModel-p2	3.5	*moveLog-f-ev2-evEND*
	moveLog-f-ev2-evEND	3.5	
s_{12}	moveModel-p1	4.5	*moveModel-p1*
	moveLog-f-ev2-evEND	4.5	
s_{13}	moveModel-p1	3.5	moveModel-p1
s_{14}	moveModel-p2	2.5	moveModel-p2
s_{15}	moveModel-p2	2.5	moveModel-p2
s_{16}	moveModel-p3	1.5	moveModel-p3
s_{17}	moveLog-f-ev2-evEND	1	moveLog-f-ev2-evEND
s_{18}	moveModel-p2	3.5	*moveModel-p2*
	moveLog-f-ev2-evEND	3.5	
s_{19}	moveModel-p2	2.5	moveModel-p2
s_{20}		0	
s_{21}		0	
s_{22}		0	

Tabela 5.1: *Ações de movimento de alinhamento aplicáveis para cada estado de alinhamento com seu Q^*-valor correspondente e uma política completa π_* que pode ser retornada por um planejador probabilístico*

ser gerado por outra política ótima na qual a melhor ação retornada pelo planejador para o estado s_{11} é `moveModel-p2`.

5.2 Similaridades e diferenças com os trabalhos existentes na literatura

Nesta seção, são discutidas as similaridades e diferenças entre nossa proposta e os estudos previamente discutidos nas Seções 2.3 e 2.4.

A formulação proposta nesse trabalho para a tarefa de alinhamento para redes de Petri estocásticas é inspirada na formulação de Adriansyah [5] para a tarefa de alinhamento, uma vez que a definição de movimentos de alinhamento estocásticos é baseada na definição dos movimentos de alinhamento de Adriansyah [5]. Além disso, nossa proposta traduz a tarefa de alinhamento para o problema do caminho mais curto, porém no nosso caso este problema é o *problema de caminho mais curto estocástico*. Diferente da abordagem do Adriansyah [5] que tem como objetivo encontrar um alinhamento ótimo (Definição 17), nosso objetivo é encontrar um ranking de alinhamentos a partir de uma política ótima

(Definição 26) .

Igual a [22, 26, 25], nossa proposta também é uma abordagem baseada em planejamento, porém *ProbPlanAlign* modela a tarefa de alinhamento como um problema de planejamento probabilístico em vez de um problema de planejamento determinístico. Semelhante a [22, 26, 25], *ProbPlanAlign* codifica o problema usando uma linguagem de planejamento. No caso dos três primeiros, a linguagem utilizada é a PDDL, enquanto, em nossa proposta, utilizamos sua versão probabilística, que é chamada de PPDDL. Além disso, assim como a abordagem de Leoni e Marrella [22], em nossa proposta assumimos que os traces estão totalmente ordenados.

Semelhante à abordagem proposta por Bergami et al. [9], *ProbPlanAlign* gera um ranking de alinhamentos. Porém, as métricas utilizadas para gerar o ranking são diferentes. As métricas usadas na abordagem de Bergami et al. [9] são uma função de distância e a probabilidade do trace do modelo, essas duas métricas são então multiplicadas para obter a probabilidade final do trace do log de eventos. *ProbPlanAlign* encontra a política com o melhor custo acumulado esperado para completar o alinhamento. Esta métrica (valor esperado) combina a probabilidade de disparo das transições na rede de Petri estocástica e o custo dos alinhamentos. Finalmente, *ProbPlanAlign* usa o conceito de ações de movimento de alinhamento estocástico no processo de alinhamento. Portanto, o *ProbPlanAlign* considere que o modelo pode ser atualizado, enquanto Bergami et al. [9] considera que o modelo fornecido como entrada continua sem alteração.

A Tabela 5.2 mostra as diferenças e semelhanças entre a abordagem do *ProbPlanAlign* e Bergami et al. [9].

	Entrada	Saída	Métrica	Conceito de Movimentação de Alinhamento	Modelo fornecido como entrada
Abordagem de Bergami et al. [9]	Rede de Petri estocástica e trace do log de eventos	Ranking de alinhamentos	Função de distância e probabilidade do trace do modelo	Não usa esse conceito	Continua sem alteração
ProbPlanAlign	Rede de Petri estocástica e trace do log de eventos	Ranking de alinhamentos	Custo do alinhamento e probabilidades de disparo das transições *(Função valor)*	Utiliza este conceito	Pode ser atualizado

Tabela 5.2: *Comparação entre a abordagem de Bergami et al. [9] e o ProbPlanAlign.*

Capítulo 6

Avaliação

Nossa proposta está principalmente relacionada ao trabalho de Bergami et al. [9], que é o único trabalho na literatura que também realiza alinhamento em uma rede de Petri estocástica. Por esta razão, neste capítulo, realizamos experimentos comparando o *ProbPlanAlign* com o trabalho de Bergami et al. [9].

As questões de pesquisa que os experimentos pretendem responder são:

- **RQ1**: Qual é a diferença entre o ranking de alinhamentos gerados pela abordagem de Bergami et al. [9] e pelo *ProbPlanAlign*?
- **RQ2**: Como o tamanho do modelo de processo (em termos de transições) impacta o consumo de tempo da abordagem de Bergami et al. [9] e do *ProbPlanAlign*?
- **RQ3**: Como o número de simulações *n* impacta o consumo de tempo e o tamanho do conjunto de alinhamentos retornados por *ProbPlanAlign*?
- **RQ4**: O *ProbPlanAlign* consegue competir com a abordagem do Bergami et al. [9] em termos de consumo de tempo?
- **RQ5**: Como os diferentes tipos de ruído impactam o consumo de tempo da abordagem de Bergami et al. [9] e do *ProbPlanAlign*?

Para responder a RQ1, a rede de Petri estocástica mostrada na Figura 1.1 é usada. Para responder RQ2, RQ3 e RQ4, usamos logs de eventos sintéticos e modelos de processos com um número variado de transições. Para gerar os logs de eventos sintéticos, primeiro foram gerados traces em conformidade a partir da rede de Petri sintética e depois incluímos diferentes tipos de ruído. O ruído pode ser uma troca, inserção ou remoção de um evento com chance de Y%. O objetivo de incluir ruído nos traces em conformidade é ter traces que não estejam em conformidade com o modelo de processo e, portanto, sejam candidatos a serem alinhados. Assim, o percentual de ruído está diretamente relacionado ao percentual de não conformidade do trace do log de eventos. Este método de injetar ruído para gerar logs de eventos sintéticos também é utilizado em outros trabalhos [22, 25, 26, 17, 5, 4]. Para responder a RQ5, usamos um log de eventos sintético e um modelo de processo de tamanho 96, cada trace no log de eventos sintético contém apenas um tipo de ruído.

6.1 Ferramenta e configurações

Desenvolvemos uma ferramenta [1] que implementa a estratégia de alinhamento descrita no Capítulo 5. Caso uma rede de Petri estocástica não esteja disponível, esta ferramenta, adicionalmente, permite ao usuário importar uma rede de Petri determinística e um log de eventos de treinamento para geração de uma rede de Petri estocástica, de acordo com os estimadores descritos por Burke, Leemans e Wynn [12]. Os estimadores implementados são os estimadores de frequência, par de atividades, e par de atividades com escala média [12]. Adicionalmente, a ferramenta implementada é capaz de estimar a frequência de atividades invisíveis para obter uma rede de Petri estocástica. Quando uma rede de Petri determinística que inclui atividades invisíveis é utilizada para obter uma rede de Petri estocástica, existe um desafio na determinação da frequência real de atividades invisíveis, uma vez que elas não são registradas no log de eventos de treinamento. Para resolver esse problema, optamos por ler o log de eventos e atribuir um rótulo às atividades invisíveis em cada trace, tratando-as como atividades rotuladas e assim contando sua frequência nos estimadores.

Após obter a rede de Petri estocástica, o usuário da ferramenta deve inserir um *trace* para alinhar. Então a ferramenta gera os arquivos de domínio e problema para este *trace*. Por fim, a ferramenta utiliza um planejador probabilístico para consumir esses arquivos e gerar um ranking.

Nestes experimentos, o *ppddl-planner*[2] é usado. O *ppddl-planner* é baseado no planejador *mdpsim2*, apresentado na primeira trilha probabilística da competição internacional de planejamento [32]. O *ppddl-planner* trabalha com um cliente e um servidor, enquanto o cliente envia o estado inicial do problema, o servidor responde com qual deve ser a ação a ser executada dado este estado. Após o cliente executar esta ação, ele retorna um novo estado. O planejador então permanece indo e voltando até atingir um estado meta. Em outras palavras, o *ppddl-planner* intercala planejamento e simulação. É possível obter da execução do *ppddl-planner* uma política parcial, ou seja, uma política que inclui apenas ações para os estados que foram visitados durante a simulação.

No *ppddl-planner* é possível escolher entre um conjunto de algoritmos e heurísticas para resolver o problema de planejamento. Nestes experimentos, foi escolhido o algoritmo denominado *LAO* [15] e a heurística *hmax*. Para os experimentos, foi utilizada uma máquina com CPU Intel(R) Core(TM) i5-2430M 2,80GHz e 4GB de RAM.

6.2 Rankings obtidos a partir de uma rede de Petri estocástica

Nesta seção, respondemos à primeira questão de pesquisa:

RQ1: Qual é a diferença entre o ranking de alinhamentos gerados pela abordagem de Bergami et al. [9] e pelo *ProbPlanAlign*?

[1] Disponível em https://github.com/matheusPereiraAlmeida/conf_checker_ppddl_gen

[2] https://github.com/fteicht/ppddl-planner

Para responder a esta pergunta, usamos a rede de Petri estocástica mostrada na Figura 1.1 e os traces $\langle a, f\rangle$ $\langle a, b\rangle$, $\langle a, f, c\rangle$, $\langle a, f, b, b, c\rangle$ e $\langle a, b, c\rangle$. Dentre esses traces, existem aqueles com uma atividade não apresentada no modelo de processo, traces com atividades invisíveis e *loops*. Os rankings são mostrados na Tabela 6.1.

A política gerada pelo planejador probabilístico foi simulada $n = 100$ vezes para obter o ranking de cada *trace*. Vale ressaltar que o valor de n influência o tamanho do conjunto de alinhamentos retornado pelo *ProbPlanAlign*. Por exemplo, para o trace $\langle a, f\rangle$ com $n = 100$ o tamanho deste conjunto é 6 (ver Tabela 6.1). Para o mesmo trace e $n = 25$, o tamanho desse conjunto é 4 e o ranking retornado inclui $\langle a, f, b, c\rangle$, $\langle a, f, b, d\rangle$, $\langle a, f, b, b, c\rangle$ e $\langle a, f, b, b, d\rangle$

Nos traces $\langle a, f\rangle$, $\langle a, f, c\rangle$ e $\langle a, f, b, b, c\rangle$, há uma atividade que ocorre e não é permitida pelo modelo, que é a atividade f, que representa a ação de enviar um e-mail ao cliente confirmando o recebimento do pedido. A tabela 6.1 mostra que as abordagens Bergami et al. [9] e *ProbPlanAlign* geram alinhamentos completamente diferentes, pois nossa abordagem é capaz de lidar com a atividade inexistente no modelo. Observe também que todos os rankings encontrados pela abordagem de Bergami et al. [9] contêm o mesmo conjunto de alinhamentos, variando apenas na probabilidade.

$\langle a, f\rangle$				$\langle a, b\rangle$				$\langle a, f, c\rangle$				$\langle a, f, b, b, c\rangle$				$\langle a, b, c\rangle$			
Bergami et al.		*ProbPlanAlign*		*Bergami et al.*		*ProbPlanAlign*		*Bergami et al.*		*ProbPlanAlign*		*Bergami et al.*		*ProbPlanAlign*		*Bergami et al.*		*ProbPlanAlign*	
trace	**prob**	**trace**	**freq**	**trace**	**prob**	**trace**	**freq**	**trace**	**prob**	**trace**	**freq**	**trace**	**prob**	**trace**	**freq**	**trace**	**prob**	**trace**	**freq**
abc	0.166	afbc	53	abc	0.25	abc	56	abc	0.25	afbc	100	abc	0.166	afbbc	100	abc	0.5	abc	100
abd	0.099	afbd	25	abd	0.15	abd	28	abd	0.099			abd	0.075			abd	0.15		
abbc	0.014	afbbc	11	abbc	0.016	abbc	10	abbc	0.016			abbc	0.02			abbc	0.02		
abbd	0.008	afbbd	6	abbd	0.009	abbd	2	abbd	0.008			abbd	0.009			abbd	0.009		
abbbc	0.001	afbbbc	4	abbbc	0.002	abbbc	1	abbbc	0.002			abbbc	0.002			abbbc	0.002		
abbbd	0.001	afbbbbc	1	abbbd	0.001	abbbbc	1	abbbd	0.001			abbbd	0.001			abbbd	0.001		

Tabela 6.1: *Rankings gerados pela abordagem de Bergami et al. [9] e* ProbPlanAlign *(com $n = 100$) para a rede de Petri estocástica mostrada na Figura 1.1 e diferentes traces.*

Para os traces $\langle a, f, c\rangle$, $\langle a, f, b, b, c\rangle$ e $\langle a, b, c\rangle$ apenas um alinhamento é encontrado pelo *ProbPlanAlign*. A razão pela qual nossa abordagem gera apenas um alinhamento para esses traces específicos é que o único lugar na rede de Petri estocástica mostrada na Figura 1.1 que tem probabilidades diferentes de 1 é o lugar `p3`. A partir desse ponto é possível disparar *c*, *d* ou *inv*. Além disso, observe que o movimento síncrono é sempre priorizado para esses traces, pois tem custo 0. Assim, dado o estado de alinhamento onde o token está no lugar p3 na rede de Petri estocástica e a próxima atividade a ser alinhada é a atividade *c*, que é o caso desses traces, um movimento síncrono será necessariamente escolhido pelo planejador como a melhor ação de movimento. Assim, a ação `moveModel-p3` (que tem efeitos probabilísticos) não é escolhida pelo planejador, consequentemente, um ranking com mais de 1 alinhamento não será gerado pelo *ProbPlanAlign*. Além disso, observe que o trace $\langle a, b, c\rangle$ é totalmente permitido pelo modelo. Para este trace, o *ProbPlanAlign* retorna apenas $\langle a, b, c\rangle$ porque as melhores ações de movimentação são as síncronas que têm probabilidade um e custo zero. A abordagem de Bergami et al. [9] retorna $\langle a, b, c\rangle$ como o primeiro alinhamento do ranking que inclui outros alinhamentos.

Para os traces $\langle a, f\rangle$ e $\langle a, b\rangle$ vários alinhamentos são encontrados pelo *ProbPlanAlign*. Dado o estado de alinhamento no qual o token está no lugar *p3* na rede de Petri estocástica e a próxima atividade a ser alinhada é `evEND` (significando que o trace a ser alinhado já

terminou), a ação `moveModel-p3` é selecionada pelo planejador para chegar ao final da rede de Petri estocástica. Como esta ação tem efeitos probabilísticos, o simulador de política escolherá qual atividade (c, d ou inv) deverá ser executada de acordo com as probabilidades de disparo da rede de Petri estocástica. Para o trace $\langle a, b \rangle$, nossa abordagem resulta no mesmo ranking gerado por Bergami et al. [9], porém quando o trace $\langle a, f \rangle$ é alinhado, nosso ranking é capaz de definir qual deve ser a melhor posição para a atividade f e então retornar um ranking de alinhamento correspondente.

Resumindo, se o trace a ser alinhado é inteiramente permitido pelo modelo, *ProbPlanAlign* retorna apenas um alinhamento porque escolhe ações síncronas com custo 0 para minimizar o custo acumulado esperado, enquanto a abordagem de Bergami et al. [9] retorna um ranking com mais de um alinhamento. Observe que retornar um ranking com mais de um alinhamento possível pode não ser muito útil neste caso, uma vez que o trace já está em conformidade com o modelo de processo.

Caso o trace a ser alinhado não seja totalmente permitido pelo modelo, A abordagem de Bergami et al. [9] retorna apenas alinhamentos permitidos pelo modelo, assumindo que o modelo de processo fornecido como entrada permanece inalterado. Em contraste, *ProbPlanAlign* assume que o modelo de processo em uso pode estar desatualizado. Especificamente, se houver uma atividade que ocorre no trace a ser alinhado e não é permitida pelo modelo, *ProbPlanAlign* escolhe uma ação de movimentação de log para lidar com ela.

6.3 Análise de escalabilidade

Esta seção tem como objetivo analisar a escalabilidade da abordagem de Bergami et al. [9] e do *ProbPlanAlign* quando o tamanho das redes de Petri probabilísticas (em termos do número de transições) cresce. Especificamente, nesta seção, as questões de pesquisa RQ2, RQ3 e RQ4 são respondidas.

Para responder a essas perguntas, foram usadas redes de Petri e log de eventos sintéticos. Primeiro, o PLG2 [11] é utilizado para gerar seis redes de Petri determinísticas, com tamanhos crescentes: 22, 45, 96, 128, 187 e 237 transições. As redes de Petri com 128, 187 e 237 contêm transições paralelas.

Para cada modelo de processo, criamos um log de eventos L_M com 1000 traces. Somente traces em conformidade foram incluídos em cada log de eventos gerado. Em seguida, cada rede de Petri determinística gerada foi importada para gerar uma rede de Petri estocástica usando o estimador de frequência descrito em [12] considerando 900 traces do log de eventos L_M. Em seguida, foi injetado ruído nos 100 traces restantes de L_M. O ruído pode ser uma troca, inserção ou remoção de um evento com chance de Y%. Nestes experimentos, cada trace pode ter vários tipos de ruído e Y é definido como 10%, 20% e 30%. Esses traces restantes são chamados de traces com ruído. Conforme afirmado anteriormente, o objetivo de incluir ruído nos traces é ter traces que não estejam em conformidade com o modelo do processo.

Para cada trace com ruído, o *ppddl-planner* (que inclui o planejador e o simulador) foi executado 25 e 100 vezes para gerar 25 e 100 alinhamentos possíveis, respectivamente (ou

seja, $n = 25$ e $n = 100$). As figuras 6.1 e 6.2 mostram o consumo de tempo do *ProbPlanAlign* para gerar um ranking de alinhamentos para $n = 100$ e $n = 25$, respectivamente. Como dito antes, *ProbPlanAlign* emprega o conceito de ações de movimento de alinhamento e encontra uma política ótima que minimiza o custo acumulado esperado, diferente da abordagem proposta por Bergami et al. [9] que usa métricas simples para gerar o ranking. Também foi incluído o tempo da abordagem de força bruta proposta por Bergami et al. [9] para ter uma base de comparação nas Figuras 6.1 e 6.2.

RQ2: Como o tamanho do modelo de processo (em termos de transições) impacta o consumo de tempo da abordagem de Bergami et al. [9] e do *ProbPlanAlign*?

O gargalo da abordagem de Bergami et al. [9] está no modelo de processo, uma vez que esta abordagem gera todos os traces possíveis do modelo, o cálculo do ranking é realizado simplesmente multiplicando as métricas de probabilidade e similaridade. Consequentemente, quanto maior o modelo de processo, mais traces de modelo ele terá, levando a um aumento no tempo de processamento. Podemos ver esse comportamento da abordagem de Bergami et al. [9] nas Figuras 6.1 e 6.2.

No caso do *ProbPlanAlign*, à medida que o tamanho do modelo de processo cresce em termos de transições, o número de ações aumenta e o espaço de busca do planejador probabilístico também será maior, consequentemente este algoritmo requer mais tempo. Podemos ver este comportamento do *ProbPlanAlign* nas Figuras 6.1 e 6.2.

O consumo de tempo do *ProbPlanAlign* com $n = 100$ (Figura 6.1) e $n = 25$ (Figura 6.2) parece crescer linearmente com inclinação 0,377 e 0,095 em média, respectivamente. Além disso, vale ressaltar que o consumo de tempo do *ProbPlanAlign* com $n = 25$ é inferior a 21,4 segundos para todos os tamanhos do modelo de processo e níveis de ruído. Assim, o *ProbPlanAlign* com $n = 25$ apresenta um bom desempenho em termos de consumo de tempo para gerar um ranking de alinhamentos, especialmente para modelos de processos com um grande número de transições.

RQ3: Como o número de simulações n impacta o consumo de tempo e o tamanho do conjunto de alinhamentos retornados pelo *ProbPlanAlign*?

Como o *ppddl-planner* (o planejador utilizado nos experimentos) intercala planejamento e simulação, o valor de n influencia (i) o tempo consumido tanto pelo planejador probabilístico quanto pelo simulador de políticas; e (ii) o tamanho do conjunto de alinhamentos retornados pelo *ProbPlanAlign*.

Enquanto com $n = 100$, *ProbPlanAlign* consegue encontrar um maior número de alinhamentos no ranking ao custo de um tempo de processamento maior, com $n = 25$ o *ProbPlanAlign* pode encontrar um número menor de alinhamentos para formar o ranking, porém com um tempo de execução bem menor. Assim, há uma *trade-off* entre o número de alinhamentos encontrados e o tempo de processamento. Vale ressaltar que não há perda muito significativa em termos de qualidade de rankings encontrados, porque os alinhamentos mais frequentes aparecem no ranking resultante, e os alinhamentos menos frequentes ficarão de fora.

Por exemplo, para a rede de Petri estocástica com 237 transições e 30% de ruído, com

$n = 25$ o tempo necessário para o *ProbPlanAlign* gerar um ranking de alinhamentos foi reduzido aproximadamente 4 vezes quando comparado a $n = 100$. Além disso, o tamanho do conjunto de alinhamentos foi reduzido de 11 para 5.

RQ4: O *ProbPlanAlign* consegue competir com a abordagem do Bergami et al. [9] em termos de consumo de tempo?

A Figura 6.1 mostra que permitir a movimentação do log tornou o *ProbPlanAlign* mais caro do que a abordagem de Bergami et al. [9] na maioria dos casos para $n = 100$. Para reduzir o consumo de tempo de *ProbPlanAlign*, é possível escolher um valor menor de n.

A Figura 6.2 mostra que o *ProbPlanAlign* com $n = 25$ é mais rápido que a abordagem de Bergami et al. [9] para todos os níveis de ruído quando o tamanho do modelo de processo é igual a 187 e 237. Para o tamanho de 128 com 30% de ruído *ProbPlanAlign* também é mais rápido. Para outros tamanhos, o consumo de tempo da abordagem de *ProbPlanAlign* e Bergami et al. [9] é semelhante.

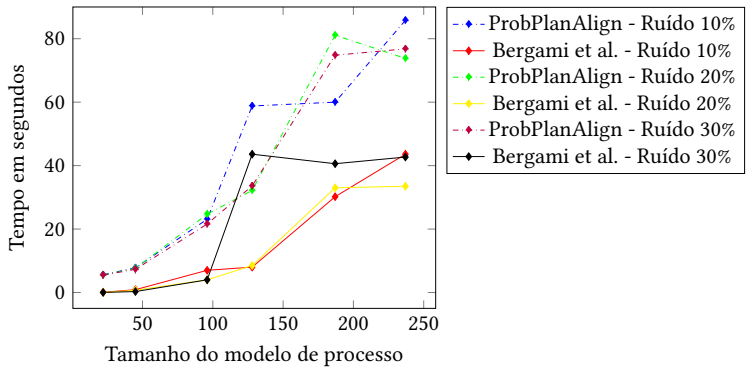

Figura 6.1: *Consumo de tempo das abordagens* ProbPlanAlign *(com $n = 100$) e Bergami et al. [9] para gerar um ranking de alinhamentos.*

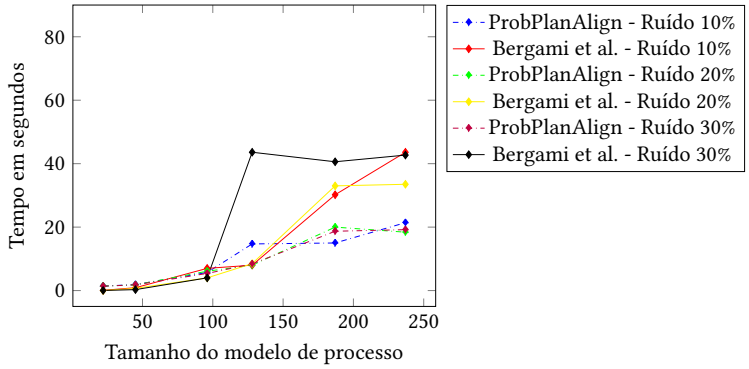

Figura 6.2: *Consumo de tempo das abordagens* ProbPlanAlign *(com $n = 25$) e Bergami et al. [9] para gerar um ranking de alinhamentos.*

6.4 Análise para diferentes tipos de ruído

Esta seção tem como objetivo responder à questão de pesquisa RQ5:

RQ5: Como os diferentes tipos de ruído impactam o consumo de tempo da abordagem de Bergami et al. [9] e do *ProbPlanAlign*?

Para responder a esta questão uma rede de Petri determinística de tamanho 96 e um log de eventos com 1000 traces são utilizados, sendo 900 traces empregados para transformar a rede de Petri determinística em uma rede de Petri estocástica. Nos 100 traces restantes, foi injetado ruído. Assim como na seção anterior, são empregados três tipos de ruído: inserção, remoção e troca de um evento com chance de Y%. Y é definido como 10%, 15%, 20%, 25% e 30%. Nesta análise, apenas um tipo de ruído é incluído em cada trace. Neste experimento, o *ppddl-planner* foi executado 25 vezes, ou seja, $n = 25$.

A Figura 6.3 mostra que a inserção, remoção e troca de um evento têm um impacto relativamente insignificante na abordagem do Bergami et al. [9]. Isso acontece porque o gargalo dessa abordagem está no próprio modelo de processo, especificamente com o número de transições, e nesses experimentos esse valor é fixado em 96.

No caso da abordagem do *ProbPlanAlign*, quanto mais ações presentes no problema de alinhamento codificado como PPDDL, mais tempo o planejador levará para gerar uma política ótima. Observe que a injeção de ruído pode diminuir ou aumentar o comprimento do trace a ser alinhado, afetando o número de ações criadas em PPDDL do tipo `log-move` e `move-sync`.

Para o ruído de inserção, mais atividades são adicionadas aos traces para serem alinhados, resultando em um número maior de ações do tipo `log-move` e `move-sync` e portanto aumentando o consumo de tempo do *ProbPlanAlign*, à medida que o espaço de busca do algoritmo se torna maior. Na Figura 6.4, no caso do ruído de inserção, o tempo de processamento do *ProbPlanAlign* aumenta linearmente à medida que a porcentagem de ruído aumenta.

Para o ruído de remoção, o trace a ser alinhado é reduzido, levando a um menor número de ações geradas e consequentemente a um menor tempo necessário para o *ProbPlanAlign* encontrar um ranking. A figura 6.4 mostra que quando a porcentagem de ruído de remoção aumenta, o tempo de processamento do *ProbPlanAlign* diminui linearmente.

Por outro lado, o ruído de troca não afetou o tempo de processamento para as abordagens *ProbPlanAlign* e Bergami et al. [9].

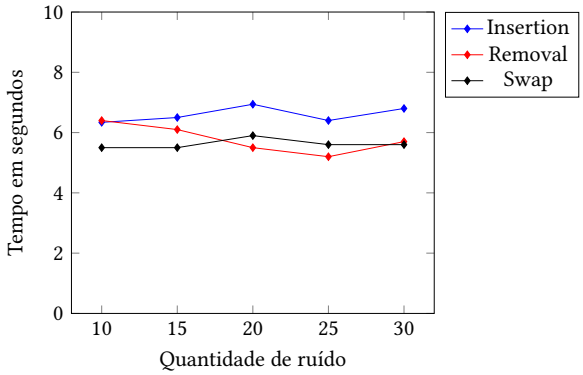

Figura 6.3: *Consumo de tempo da abordagem de Bergami et al. [9] para cada tipo de ruído.*

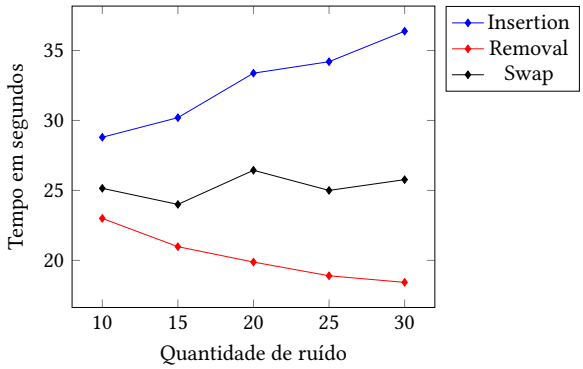

Figura 6.4: *Consumo de tempo do* ProbPlanAlign *para cada tipo de ruído.*

Capítulo 7

Considerações Finais

7.1 Contribuições

Uma das contribuições deste trabalho é o desenvolvimento da formalização da tarefa de alinhamento de uma rede de Petri estocástica M e um trace $\sigma \in L$ utilizando o conceito de ação de movimento de alinhamento estocástico. Especificamente, dada a utilização de uma rede de Petri estocástica, definimos a ação de movimento do modelo estocástico que inclui as probabilidades de disparo das transições da rede de Petri estocástica. Também definimos matematicamente uma política ótima de alinhamento, ou seja, um mapeamento entre estados de alinhamento e ações de movimento de alinhamento com o menor custo acumulado esperado para completar o alinhamento. Observe que este valor esperado combina tanto a probabilidade de disparo das transições na rede de Petri estocástica quanto o custo dos alinhamentos. Assim, a tarefa de alinhamento visa produzir um ranking de alinhamentos simulando n vezes esta política ótima de alinhamento.

Para resolver efetivamente esta tarefa de alinhamento, também propomos uma abordagem chamada *ProbPlanAlign* que modela nossa proposta como um problema de *caminho mais curto estocástico*. Descrevemos detalhadamente como esse problema de alinhamento pode ser codificado usando PPDDL, especialmente como representar ações de movimentação de modelo usando esta linguagem. Além disso, incorporamos o predicado booleano `executedActivity` para facilitar a extração da atividade executada associada a cada ação. Por fim, incorporamos uma métrica que visa minimizar o custo acumulado esperado considerando a função de custo padrão.

Foi realizada uma comparação entre o *ProbPlanAlign* e a abordagem proposta por Bergami et al. [9], destacando as distinções entre eles quanto aos alinhamentos retornados e ao consumo de tempo. Como esperado, à medida que o tamanho do modelo de processo aumenta, o consumo de tempo de ambas as abordagens aumenta. Adicionalmente, foi analisado como o tipo de ruído afeta o consumo de tempo de ambas as abordagens. Observamos que o tipo de ruído não impacta significativamente o tempo de consumo da abordagem de Bergami et al. [9]; entretanto, isso afeta o tempo do *ProbPlanAlign*. Enquanto o ruído de inserção tende a aumentar o consumo de tempo, o ruído de remoção tende a diminuir o consumo de tempo do *ProbPlanAlign*.

7.2 Limitações

As limitações da proposta são:

- Assumir a Petri net estocástica como sendo 1-bounded.
- Trabalhar apenas com traces totalmente ordenados.
- Não fornecer suporte para trabalhar com rótulos duplicados do modelo de processo.

7.3 Publicações

7.3.1 Using automatic planning to find the most probable alignment: A history-based approach

O estudo intitulado *Using automatic planning to find the most probable alignment: A history-based approach* [7] propõe a identificação do alinhamento ótimo por meio da aplicação de uma função de custo, a qual se fundamenta na análise do histórico de execução do processo e utiliza uma rede de Petri de natureza determinística.

A determinação automática da função de custo é efetuada a partir do log de eventos, no qual é atribuída uma frequência a cada atividade presente no log de eventos. A partir dessas frequências, é aplicada uma função de perfil de custo, conforme delineada por Alizadeh, Leoni e Zannone [6], a fim de converter essas frequências em custos adequados para integração na função de custo. Essa proposta foi incorporada à ferramenta originalmente desenvolvida por Leoni e Marrella [22].

Além disso, a validação dessa abordagem foi conduzida por meio da análise de dados de log de eventos sintéticos e de um log de eventos reais. Os resultados obtidos indicam que a utilização de uma função de custo fundamentada no histórico pode proporcionar alinhamentos que se aproximam mais do log de eventos, ou seja, alinhamentos que têm maior probabilidade de ocorrência, em comparação com aqueles obtidos por meio da função de custo convencional.

Esse trabalho foi publicado no *Encontro Nacional de Inteligência Artificial e Computacional* (ENIAC) [7].

7.3.2 Alignment-based conformance checking for a stochastic Petri net

Os resultados e contribuições finais desta dissertação foram submetidos para um periódico internacional.

7.4 Trabalhos Futuros

Como perspectiva para trabalhos futuros, é possível utilizar uma função de custo baseada no histórico, similar àquela empregada em Almeida et al. [7] que é calculada para

uma rede de Petri determinística, porém, no nosso caso, seria utilizada uma rede de Petri estocástica. A partir disso, é possível fazer uma análise da influência da incorporação dessa função de custo à solução. Essa análise pode abranger tanto a qualidade do alinhamento quanto o tempo necessário para realizá-lo.

Além disso, outra extensão possível desta pesquisa seria considerar não apenas o fluxo de controle, mas também outros atributos do log de eventos que possam estar relacionados a não conformidade como por exemplo aspectos temporais, de recursos ou de dados, conforme discutido em Leoni e Aalst [23]. Outra direção a ser explorada é a ampliação do suporte para atividades com rótulos duplicados, semelhante ao que é feito no trabalho de Leemans, Maggi e Montali [19].

Ademais, há um campo crescente de estudos na literatura que se concentram no alinhamento de processos em um ambiente online, isto é, com instâncias de processos que ainda não foram concluídas, como exemplificado no trabalho de Raun, Tommasini e Awad [30]. Um novo algoritmo aproximado é proposto por Raun, Tommasini e Awad [30] para realizar alinhamento considerando um tipo de rede de Petri determinística chamada *workflow net*. Um possível trabalho futuro pode ser aplicar planejamento probabilístico no contexto de alinhamento online considerando redes de Petri estocásticas.

Apêndice A

Modelos de processos sintéticos

Neste apêndice, fornecemos os diagramas das redes de Petri sintéticas usadas para avaliar a escalabilidade de nossa abordagem baseada em planejamento.

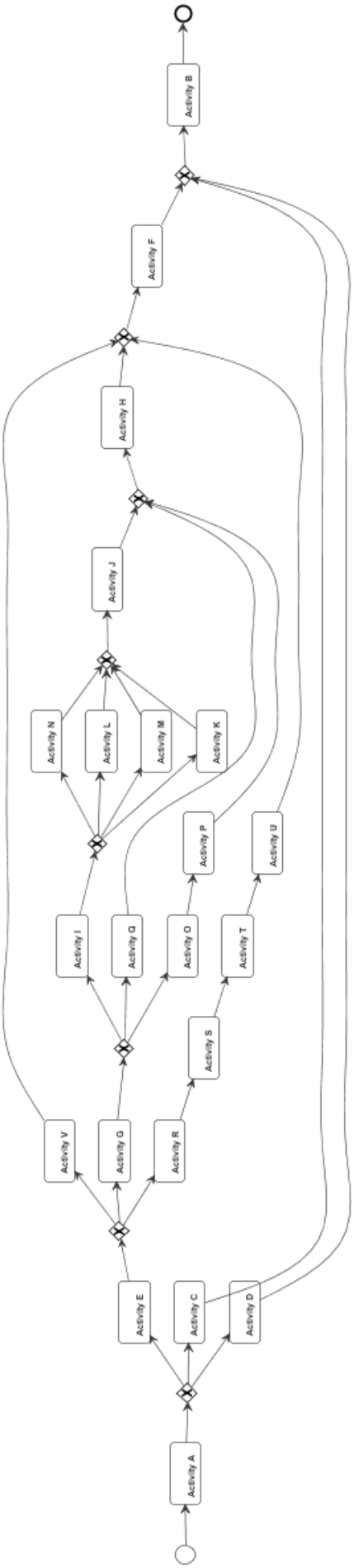

Figura A.1: *Uma rede de Petri sintética com 22 transições.*

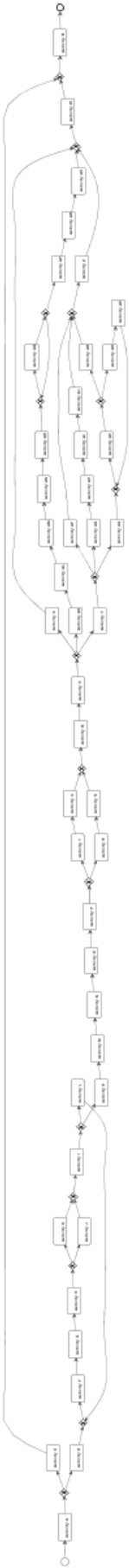

Figura A.2: *Uma rede de Petri sintética com 45 transições.*

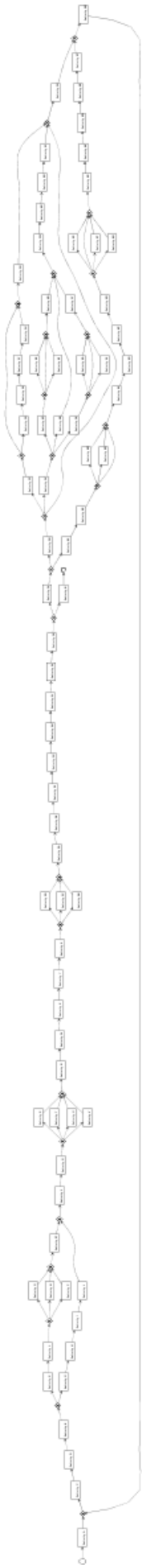

Figura A.3: *Uma rede de Petri sintética com 77 transições.*

Figura A.4: *Uma rede de Petri sintética com 96 transições.*

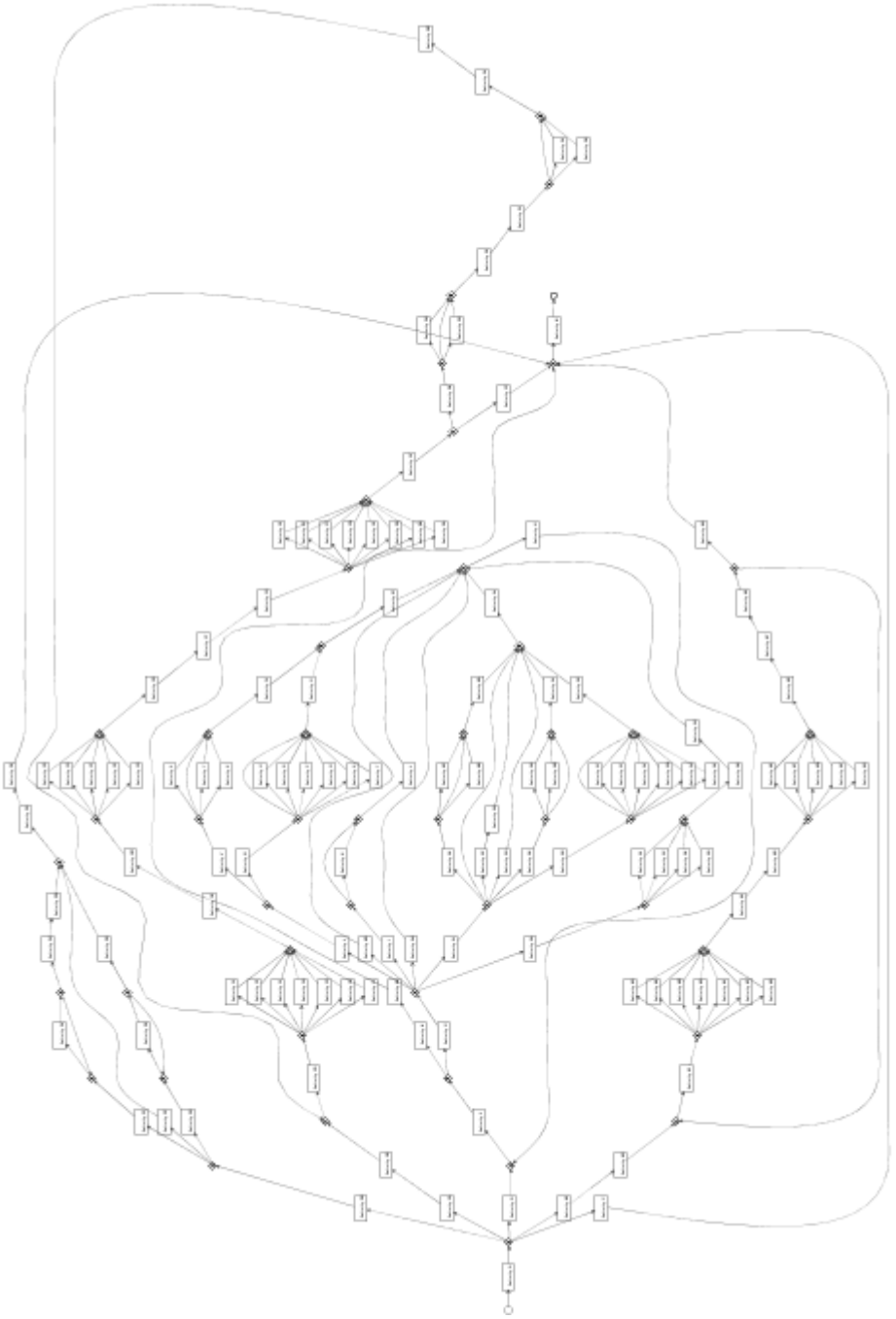

Figura A.5: *Uma rede de Petri sintética com 128 transições.*

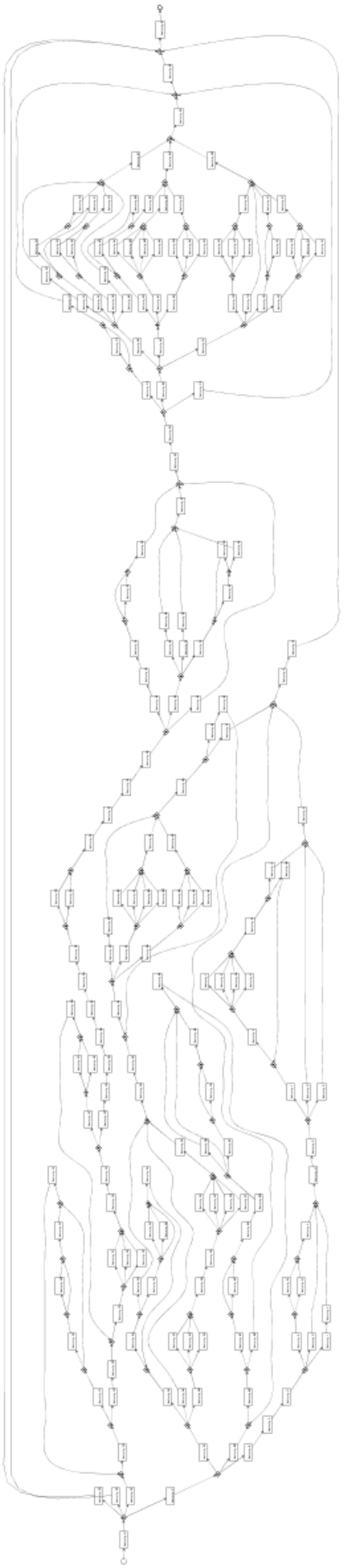

Figura A.6: *Uma rede de Petri sintética com 187 transições.*

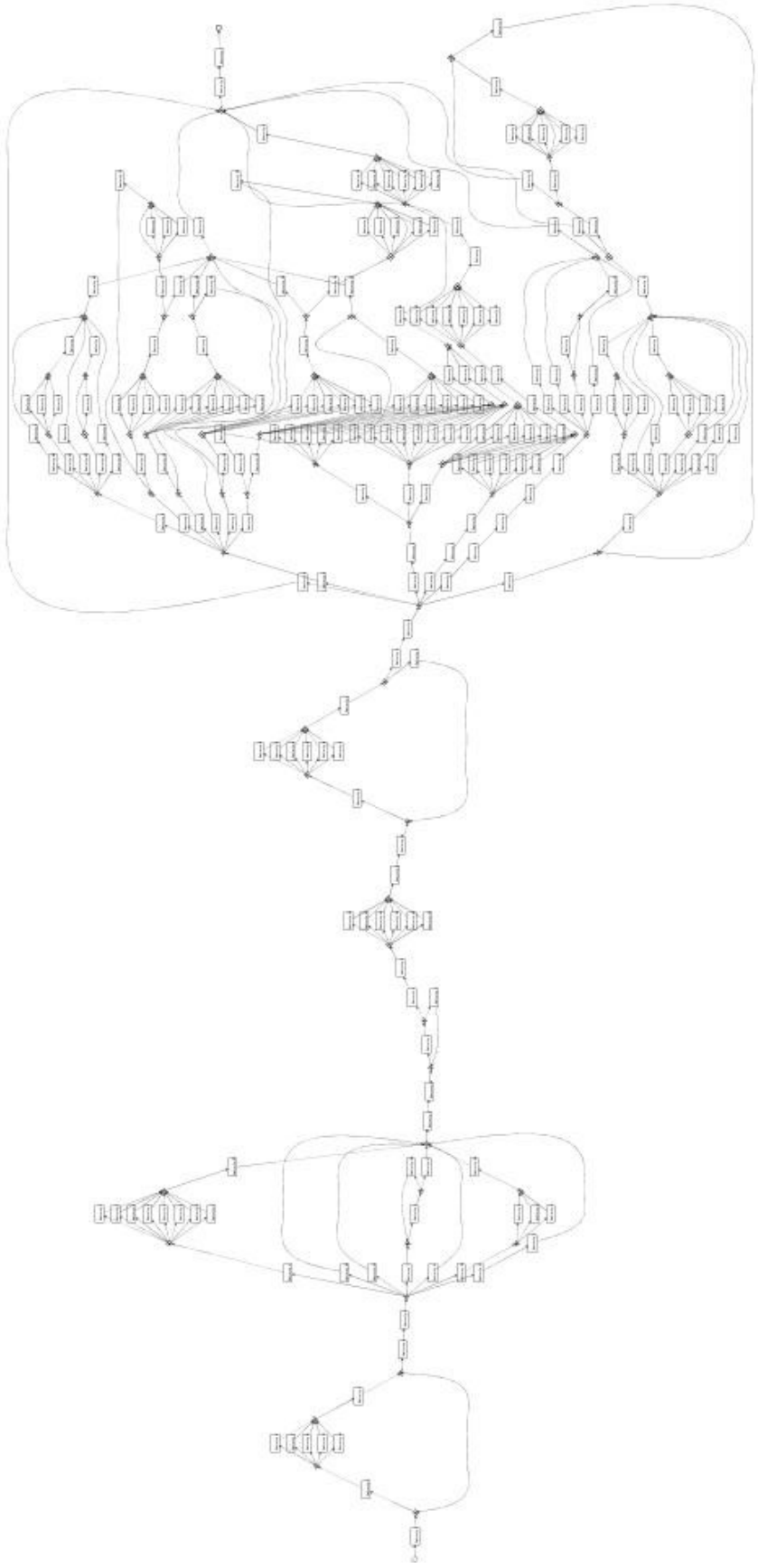

Figura A.7: *Uma rede de Petri sintética com 237 transições.*

Referências

[1] Wil van der Aalst. "Academic View: Development of the Process Mining Discipline". Em: *Process Mining in Action: Principles, Use Cases and Outlook*. Ed. por Lars Reinkemeyer. Cham: Springer International Publishing, 2020, pp. 181–196 (ver p. 1).

[2] Wil M. P. van der Aalst. *Process Mining: Data Science in Action*. 2ª ed. Heidelberg: Springer, 2016 (ver pp. 4, 8).

[3] Wil M.P. van der Aalst. "The Application of Petri Nets to Workflow Management". Em: *J. Circuits Syst. Comput.* 8 (1998), pp. 21–66 (ver p. 7).

[4] A. Adriansyah, N. Sidorova e B. F. van Dongen. "Cost-based fitness in conformance checking". Em: *International Conference on on Application of Concurrency to System Design* (2011), pp. 57–66 (ver pp. 8, 31).

[5] Arya Adriansyah. "Aligning observed and modeled behavior". Tese de dout. Technical University Eindhoven, 2014 (ver pp. 5, 11, 16, 29, 31).

[6] Mahdi Alizadeh, Massimiliano de Leoni e Nicola Zannone. "History-Based Construction of Alignments for Conformance Checking: Formalization and Implementation". Em: *Data-Driven Process Discovery and Analysis*. Ed. por Paolo Ceravolo, Barbara Russo e Rafael Accorsi. Cham: Springer International Publishing, 2015, pp. 58–78 (ver pp. 1, 9, 40).

[7] Matheus Almeida et al. "Using automatic planning to find the most probable alignment: A history-based approach". Em: nov. de 2022 (ver p. 40).

[8] Richard Bellman. *Dynamic Programming*. 1ª ed. Princeton, NJ, USA: Princeton University Press, 1957 (ver p. 14).

[9] Giacomo Bergami et al. "Probabilistic Trace Alignment". Em: *2021 3rd International Conference on Process Mining (ICPM)*. 2021, pp. 9–16 (ver pp. 11, 13, 1–3, 12, 30–37, 39).

[10] Dimitri Bertsekas. *Dynamic Programming and Optimal Control*. Belmont, Mass: Athena Scientific, 1995 (ver pp. 3, 14, 15).

[11] A. Burattin. "PLG2: Multiperspective processes randomization and simulation for online and offline settings". Em: *preprint arXiv:1506.08415* (2015) (ver p. 34).

[12] Adam Burke, Sander J. J. Leemans e Moe Thandar Wynn. "Stochastic Process Discovery by Weight Estimation". Em: *Process Mining Workshops*. Cham: Springer International Publishing, 2021, pp. 260–272 (ver pp. 1, 2, 6, 12, 32, 34).

[13] J. Wainer F. Bezerra. "Algorithms for anomaly detection of traces in logs of process aware information systems". Em: *J on Data Semantics* 38 (2013), pp. 33–44 (ver p. 6).

[14] H. Reijers F. Mannhardt M. Leoni e W. Van Der Aalst. "Balanced multi-perspective checking of process conformance". Em: *Computing* 98.4 (2016), pp. 407–437 (ver p. 9).

[15] Eric A Hansen e Shlomo Zilberstein. "LAO*: A heuristic search algorithm that finds solutions with loops". Em: *Artificial Intelligence* 129.1-2 (2001), pp. 35–62 (ver p. 32).

[16] Marie Koorneef et al. "Automatic root cause identification using most probable alignments". Em: *13th International Workshop on Business Process Intelligence* (2018), pp. 204–215 (ver pp. 13, 2, 6, 9).

[17] Giacomo Lanciano. "Alignment-Based Conformance Checking of Partially-Ordered Traces and Process Models Using Automated Planning". Tese de dout. Jan. de 2018 (ver pp. 7, 9, 10, 31).

[18] Sander J. J. Leemans, Fabrizio Maria Maggi e Marco Montali. "Reasoning on Labelled Petri Nets and Their Dynamics in a Stochastic Setting". Em: *Business Process Management*. Ed. por Claudio Di Ciccio et al. Cham: Springer International Publishing, 2022, pp. 324–342 (ver pp. 1, 2, 4–7, 12).

[19] Sander J. J. Leemans, Fabrizio Maria Maggi e Marco Montali. "Reasoning on Labelled Petri Nets and Their Dynamics in a Stochastic Setting". Em: *Business Process Management*. Ed. por Claudio Di Ciccio et al. Cham: Springer International Publishing, 2022, pp. 324–342 (ver p. 41).

[20] Sander J. J. Leemans, Anja F. Syring e Wil M. P. van der Aalst. "Earth Movers' Stochastic Conformance Checking". Em: *Business Process Management Forum*. Ed. por Thomas Hildebrandt et al. Cham: Springer International Publishing, 2019, pp. 127–143. ISBN: 978-3-030-26643-1 (ver pp. 1, 2, 7, 12).

[21] Sander J.J. Leemans et al. "Stochastic process mining: Earth movers' stochastic conformance". Em: *Information Systems* 102 (2021), p. 101724. ISSN: 0306-4379 (ver pp. 2, 12).

[22] M. Leoni e A. Marrella. "Aligning Real Process Executions and Prescriptive Process Models through Automated Planning". Em: *Expert Syst Appl* 82 (2017), pp. 162–183 (ver pp. 7, 8, 10–12, 20, 25, 30, 31, 40).

[23] Massimiliano de Leoni e Wil M. P. van der Aalst. "Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming". Em: *Business Process Management*. Ed. por Florian Daniel, Jianmin Wang e Barbara Weber. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 113–129 (ver p. 41).

[24] Vladimir I Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". Em: *Soviet Physics Doklady*. Vol. 10. 8. 1966, pp. 707–710 (ver p. 13).

[25] G. Lanciano M. Leoni e A. Marrella. "Aligning Partially-Ordered Process-Execution Traces and Models Using Automated Planning". Em: *International Conference on Automated Planning and Scheduling* 28.1 (2018), pp. 321–329 (ver pp. 7, 11, 12, 30, 31).

[26] L. Giacomo M. Leoni e A. Marrella. "A Tool for Aligning Event Logs and Prescriptive Process Models through Automated Planning." Em: *BPM (Demos)*. 2017 (ver pp. 11, 12, 30, 31).

[27] Felix Mannhardt et al. "Modelling Data-Aware Stochastic Processes - Discovery and Conformance Checking". Em: *Application and Theory of Petri Nets and Concurrency*. Ed. por Luis Gomes e Robert Lorenz. Cham: Springer Nature Switzerland, 2023, pp. 77–98. ISBN: 978-3-031-33620-1 (ver pp. 2, 12).

[28] M Ajmone Marsan et al. "Modelling with generalized stochastic Petri nets". Em: *ACM SIGMETRICS performance evaluation review* 26.2 (1998), p. 2 (ver p. 6).

[29] Martin Puterman. *Markov decision processes: discrete stochastic dynamic programming*. New York: Wiley, 1994 (ver p. 14).

[30] Kristo Raun, Riccardo Tommasini e Ahmed Awad. "I Will Survive: An Event-driven Conformance Checking Approach Over Process Streams". Em: jun. de 2023, pp. 49–60 (ver p. 41).

[31] Andreas Rogge-Solti, Wil M. P. van der Aalst e Mathias Weske. "Discovering Stochastic Petri Nets with Arbitrary Delay Distributions from Event Logs". Em: *Business Process Management Workshops*. Ed. por Niels Lohmann, Minseok Song e Petia Wohed. Cham: Springer International Publishing, 2014, pp. 15–27. ISBN: 978-3-319-06257-0 (ver pp. 1, 2, 12).

[32] Håkan Younes et al. "The First Probabilistic Track of the International Planning Competition". Em: *J. Artif. Intell. Res. (JAIR)* 24 (jul. de 2005), pp. 851–887 (ver p. 32).

[33] Håkan L. S. Younes e Michael L. Littman. "PPDDL 1 . 0 : An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects". Em: 2004 (ver pp. 3, 15).