Pontifícia Universidade Católica
do Rio de Janeiro

Rafael Augusto Gaseta França

Minerando o processo de um coqueamento retardado através de agrupamento de estados.

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–Graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Hélio Côrtes Vieira Lopes

Rio de Janeiro
setembro de 2021

PUC-Rio - Certificação Digital Nº 1912721/CA

Pontifícia Universidade Católica
do Rio de Janeiro

Rafael Augusto Gaseta França

Minerando o processo de um coqueamento retardado através de agrupamento de estados.

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo:

Prof. Hélio Côrtes Vieira Lopes
Orientador
Departamento de Informática – PUC-Rio

Prof. Marcus Vinicius Soledade Poggi de Aragão
Departamento de Informática – PUC-Rio

Prof. Cassio Freitas Pereira de Almeida
ENCE

Pesquisador Marcelo Lopes de Lima
Cenpes/Petrobras

Rio de Janeiro, 24 de setembro de 2021

PUC-Rio - Certificação Digital Nº 1912721/CA

Todos os direitos reservados. A reprodução, total ou parcial do trabalho, é proibida sem a autorização da universidade, do autor e do orientador.

Rafael Augusto Gaseta França

Graduado em Ciência da Computação pela PUC-Rio em 2019 e Técnico em Mecânica pelo CEFET/RJ em 2007. Trabalhou no Laboratório de Sensores a Fibra Óptica da PUC-Rio, no Instituto Tecgraf e atualmente no ExACTa.

PUC-Rio - Certificação Digital Nº 1912721/CA

Ficha Catalográfica

França, Rafael Augusto Gaseta

Minerando o processo de um coqueamento retardado através de agrupamento de estados. / Rafael Augusto Gaseta França; orientador: Hélio Côrtes Vieira Lopes. – 2021.

95 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2021.

Inclui bibliografia

1. keywordpre – Teses. 2. keywordpre – Teses. 3. mineração de processo. 4. aprendizado de máquina. 5. ciência dos dados. 6. coqueamento retardado. 7. agrupamento hierárquico aglomerativo. I. Lopes, Hélio Côrtes Vieira. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Escrever uma dissertação durante uma pandemia foi uma tarefa desafiadora. A constante sensação de incerteza sobre o futuro próximo dificulta a navegação por perspectivas mais amplas, tão essencial pra o trabalho acadêmico. Por isso, os capítulos que seguem não seriam possíveis sem o apoio de pessoas maravilhosas.

Aos mais pais, Marlete e Gilberto, por todo o sacrifício feito para garantir a chance de uma educação melhor. A minha companheira, Roberta, pela parceria e o apoio diário em todas as áreas da minha vida. Por toda compreensão nos momentos difíceis e por protagonizar tantos dos bons.

Sou imensamente grato ao meu orientador, Hélio Lopes, por todo o incentivo, paciência nas questões acadêmicas e pessoais e confiança no meu trabalho. Ao Professor Marcus Poggi, que me apresentou a área de *process mining* e me incentivou a continuar interessado na área.

A comissão examinadora, Professor Marcus Poggi, Professor Cassio Almeida e ao Pesquisador Marcelo Lopes. Pela disponibilidade e dedicação a avaliação desse trabalho

Agradeço aos meus colegas de trabalho, Thuener Silva, William Fernandes e Georges Spyrides. Pelas discussões sempre enriquecedoras. A Sônia, que na etapa final desse programa de mestrado me ajudou tanto na elaboração desse documento. E ao Marcelo Lopes, pela dedicação a essa pesquisa em conjunto.

Ao meu grande amigo, Thiago Toledo, que por anos tentou me tirar da mecânica e me trazer para computação. E a Débora, que sempre me desafiou a ver a vida por outras perspectivas.

E aos amigos que a computação me trouxe. Em especial a Giulia, que na graduação passamos tanto tempo juntos que deveria ter contado como atividade complementar. A Jordana e a Bianca, pelas conversas maravilhosas e pelos planos que fazemos pra depois dessa pandemia.

Ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

PUC-Rio - Certificação Digital Nº 1912721/CA

Resumo

França, Rafael Augusto Gaseta; Lopes, Hélio Côrtes Vieira. **Minerando o processo de um coqueamento retardado através de agrupamento de estados.** . Rio de Janeiro, 2021. 82p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Procedimentos e processos são essenciais para garantir a qualidade de qualquer operação. Porém, o processo realizado na prática nem sempre está de acordo com o processo idealizado. Além disso, uma análise mais refinada de gargalos e inconsistências só é possível a partir do registro de eventos do processo (*log*). Mineração de processos (*process mining*) é uma área que reúne um conjunto de métodos para reconstruir, monitorar e aprimorar um processo a partir de seu registro de eventos. Mas, ao aplicar as soluções já existentes no *log* de uma unidade de coqueamento retardado, os resultados foram insatisfatórios. O núcleo do problema está na forma como o *log* está estruturado, carecendo de uma identificação de casos, essencial para a mineração do processo. Para contornar esse problema, aplicamos agrupamento hierárquico aglomerativo no *log*, separando as válvulas em grupos que exercem uma função na operação. Desenvolvemos uma ferramenta (`PLANTSTATE`) para avaliar a qualidade desses grupos no contexto da planta e ajustar conforme a necessidade do domínio. Identificando os momentos de ativação desses grupos no *log* chegamos a uma estrutura de sequência e paralelismo entre os grupos. Finalmente, propomos um modelo capaz de representar as relações entre os grupos, resultando em um processo que representa a operações em uma unidade de coqueamento retardado.

PUC-Rio - Certificação Digital Nº 1912721/CA

Palavras-chave

mineração de processo; aprendizado de máquina; ciência dos dados; coqueamento retardado; agrupamento hierárquico aglomerativo.

Abstract

França, Rafael Augusto Gaseta; Lopes, Hélio Côrtes Vieira (Advisor). **Mining the process of a delayed coker using clustered states**. Rio de Janeiro, 2021. 82p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Procedures and processes are essential to guarantee the quality of any operation. However, processes carried out in the real world are not always in accordance with the imagined process. Furthermore, a more refined analysis of obstacles and inconsistencies is only possible from the process events record (log). Process mining is an area that brings together a set of methods to rebuild, monitor and improve processes from their log. Nevertheless, when applying existing solutions to the log of a delayed coker unit, the results were unsatisfactory. The core of the problem is how the log is structured, lacking a case identification, essential for process mining. To deal with this issue, we apply agglomerative hierarchical clustering in the log, separating the valves into groups that perform a task in an operation. We developed a tool (PLANTSTATE) to assess the quality of these groups in the context of the plant and to adjust in accord to the needs of the domain. By identifying the moments of activation of these groups in the log we arrive at a structure of sequence and parallelism between the groups. Finally, we propose a model capable of representing the relationships between groups, resulting in a process that represents the operations in a delayed coker unit.

PUC-Rio - Certificação Digital Nº 1912721/CA

Keywords

process mining; machine learning; data science; delayed coke; agglomerative hierarchical clustering.

Sumário

PUC-Rio - Certificação Digital Nº 1912721/CA

PUC-Rio - Certificação Digital Nº 1912721/CA

Lista de Figuras

PUC-Rio - Certificação Digital Nº 1912721/CA

PUC-Rio - Certificação Digital Nº 1912721/CA

PUC-Rio - Certificação Digital Nº 1912721/CA

Lista de Tabelas

PUC-Rio - Certificação Digital Nº 1912721/CA

Lista de Abreviaturas

PM	Process Mining
BFS	Breadth-first search
AHC	Agglomerative Hierarchical Clustering
RF	Reference Fitness
RP	Reference Precision
DF	Discovered Fitness
DP	Discovered Precision
BMF	Boolean matrix factorization
NMF	Non-negative matrix factorization

PUC-Rio - Certificação Digital Nº 1912721/CA

1
Introdução

Um dos principais desafios das grandes corporações é extrair valor da quantidade massiva de dados gerados pelos seus sistemas informatizados. Parte destes dados descrevem sequências de eventos que fazem parte de processos. Mineração de processos (*process mining*) é uma área relativamente nova, que reúne um conjunto de técnicas para extrair valor desse registro de eventos, como exemplificado na Figura 1.1.

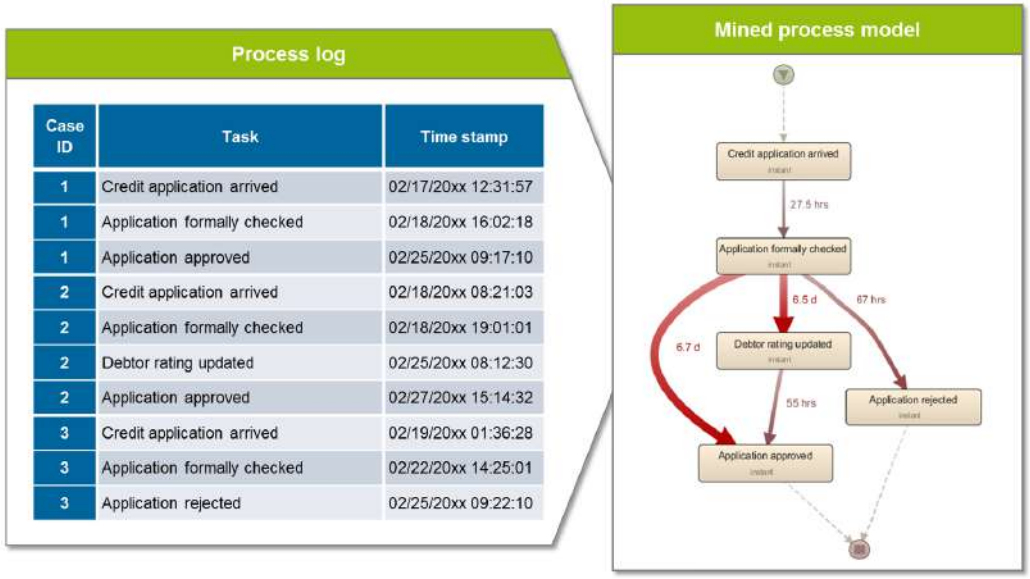


Figura 1.1: Mineração do processo a partir de um registro de eventos.
Fonte: BankingHub (BAN, 2021).

A mineração de processos se posiciona entre aprendizado de máquinas e ciência de dados e tem como objetivo descobrir, verificar e aprimorar processos reais. Normalmente em uma operação existe um processo idealizado, que pode ou não ser executado na prática.

Na primeira etapa, a de descobrimento, o registro de eventos é minerado para construir um modelo do processo real, da forma como ele aconteceu. Essa mineração depende da qualidade do registro e, normalmente, da existência de um identificador de caso. Isso é, uma separação por instância de execução do processo. Essa etapa resulta em um modelo de processo. Dependendo da

PUC-Rio - Certificação Digital Nº 1912721/CA

técnica de mineração utilizada, esse modelo resultante pode assumir diversas estruturas, como um fluxograma, um grafo, uma *Petrinet*, uma arvore, dentre outras. Seja qual for o modelo resultante, ao final dessa etapa já é possível verificar como o processo real aconteceu.

Na etapa de verificação, conhecida também como *conformance checking*, verificamos se o processo real minerado está de acordo com um processo idealizado, e vice versa. Nessa etapa é possível efetuar o *replay*, que consiste em visualizar como cada caso é executado no modelo minerado. Usualmente neste ponto já temos outras informações do registro original aplicadas ao modelo. Um exemplo é a visualização de quais etapas do processo levam mais tempo para serem executadas, auxiliando na identificação de gargalos.

Na etapa de aprimoramento usamos dados encontrados no registro de eventos e o modelo de processo minerado para chegar a um processo melhorado. Técnicas tradicionais de otimização podem ser aplicadas na etapa de aprimoramento.

Este documento propõe um método para realizar a primeira etapa da mineração do processo - descoberta, de uma unidade de coqueamento retardado, a partir de um registro de eventos, que possui como desafio principal a ausência da separação por casos.

PUC-Rio - Certificação Digital Nº 1912721/CA

1.1 Motivação

Em uma unidade de coqueamento retardado, os especialistas de planejamento e automação precisam de ferramentas que auxiliem a visualização do histórico de operações. Essa visualização auxilia na verificação de conformidade com os procedimentos previstos, identificação de mudanças no regime de operação e no treinamento de novos operadores da unidade.

Com o objetivo de gerar um mapa de processo da operação, importamos o registro de atividades da unidade de coqueamento retardado, da forma como eles estão originalmente estruturados, em um *software* para mineração de processos. Como podemos ver nas Figuras 1.2 e 1.3, é difícil extrair informações do resultado obtido. Dessa forma, identificamos a necessidade de pesquisar formas de representar a operação da unidade em um formato legível.

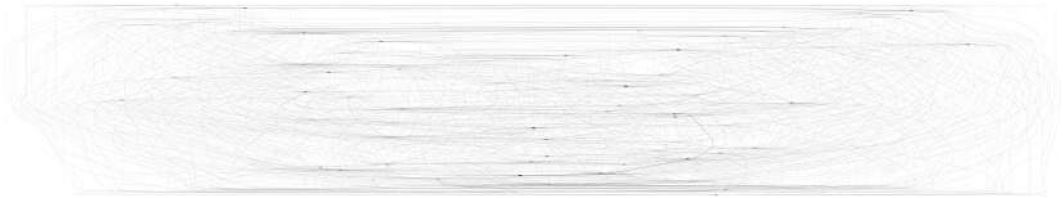


Figura 1.2: Processo gerado pelo software *Disco* utilizando os dados como foram fornecidos.

Figura 1.3: Aproximação da Figura 1.2 em uma região arbitrária.

PUC-Rio - Certificação Digital Nº 1912721/CA

1.2 Problema

Ausência de um método capaz de gerar automaticamente uma visualização legível do processo da operação de coqueamento retardado a partir de um registro de operações da unidade que não possuem separação por casos (*caseid*).

1.3 Objetivo

Extrair valor dos *logs* de eventos de operações de uma unidade de coqueamento retardado através da identificação de estados que representam operações ou sub operações e minerar um processo a partir desses estados. Para direcionar a solução em direção a esse objetivo vamos responder três perguntas de pesquisa:

- **PP1:** Dado um *log* de eventos da planta, é possível descobrir estados relacionados a operação da unidade?
- **PP2:** Dados os estados do processo e um *timestamp* do *log* de eventos da planta, é possível identificar quais estados estão ativos no momento?
- **PP3:** Dado um *log* de eventos da planta sem identificação de casos, é possível minerar a estrutura de um processo?

1.4 Metodologia

Durante revisão bibliográfica não foram encontrados trabalhos relacionados que abordem o problema definido na Seção 1.2. E, devido a falta da identificação de caso no registro, a aplicação dos algoritmos tradicionais de mineração de processos (Weijters and Van der Aalst, 2003) (Günther and Van Der Aalst, 2007) não geram resultados aceitáveis para o domínio. Todavia, as ideias e objetivos propostos nos trabalhos iniciais de mineração de processo (Van Der Aalst, 2012) permanecem aplicáveis ao problema definido. Dessa forma vamos manter os termos definidos e seguir as diretrizes gerais de PM.

Agruparemos os equipamentos de forma que cada grupo tenha o potencial de representar alguma função na operação da unidade. Posteriormente, a mineração do processo será baseada nas atividades desses grupos, e não dos equipamentos individuais.

Os grupos de equipamento serão definidos a partir da similaridade da atividade entre os equipamentos durante o período observado. Ou seja, todos os equipamentos que fazem parte de um mesmo grupo estão ativos simultaneamente em algum momento da operação.

Essa separação de equipamentos em grupos foi realizada utilizando AHC (agrupamento hierárquico aglomerativo). Para preparar a base de dados para a realização do AHC, o *log* base será dividido em segmentos de tempo de tamanho uniforme, gerando o *log* segmentado. Esses segmentos condensam as informações de atividade dos equipamentos durante aquele período.

Baseado nas características do domínio, informadas pelos especialistas de operação, desenvolvemos métricas que auxiliam na avaliação da qualidade dos agrupamentos resultantes do AHC. Essas métricas indicam quais parâmetros do AHC são mais adequados para o objetivo e domínio em questão.

Além da avaliação por métricas, os grupos serão inspecionados pelos especialistas do domínio, de forma a verificar suas funções na unidade. Isso é, se de fato possuem potencial para representar operações e sub operações. Essa verificação será feita através da ferramenta `PLANTSTATE`, desenvolvida

PUC-Rio - Certificação Digital Nº 1912721/CA

durante essa pesquisa, que fornece uma visualização do posicionamento dos grupos na planta.

Para a mineração será gerado o *log* de estados, que é um registro que indica quando cada um dos grupos entrou em atividade durante o período observado. Esse registro é construído através da detecção de quais grupos encontrados estão ativos em cada momento do *log* base.

Finalmente, o processo será minerado através do processamento do *log* de estados. Esse processamento consiste em identificar a relação entre os grupos através da identificação da proximidade dos momentos de ativação e desativação entre eles. A estrutura final do processo será a composição dessas relações identificadas, gerando o grafo do processo.

Uma visão geral da metodologia pode ser conferida na Figura 1.4

PUC-Rio - Certificação Digital Nº 1912721/CA

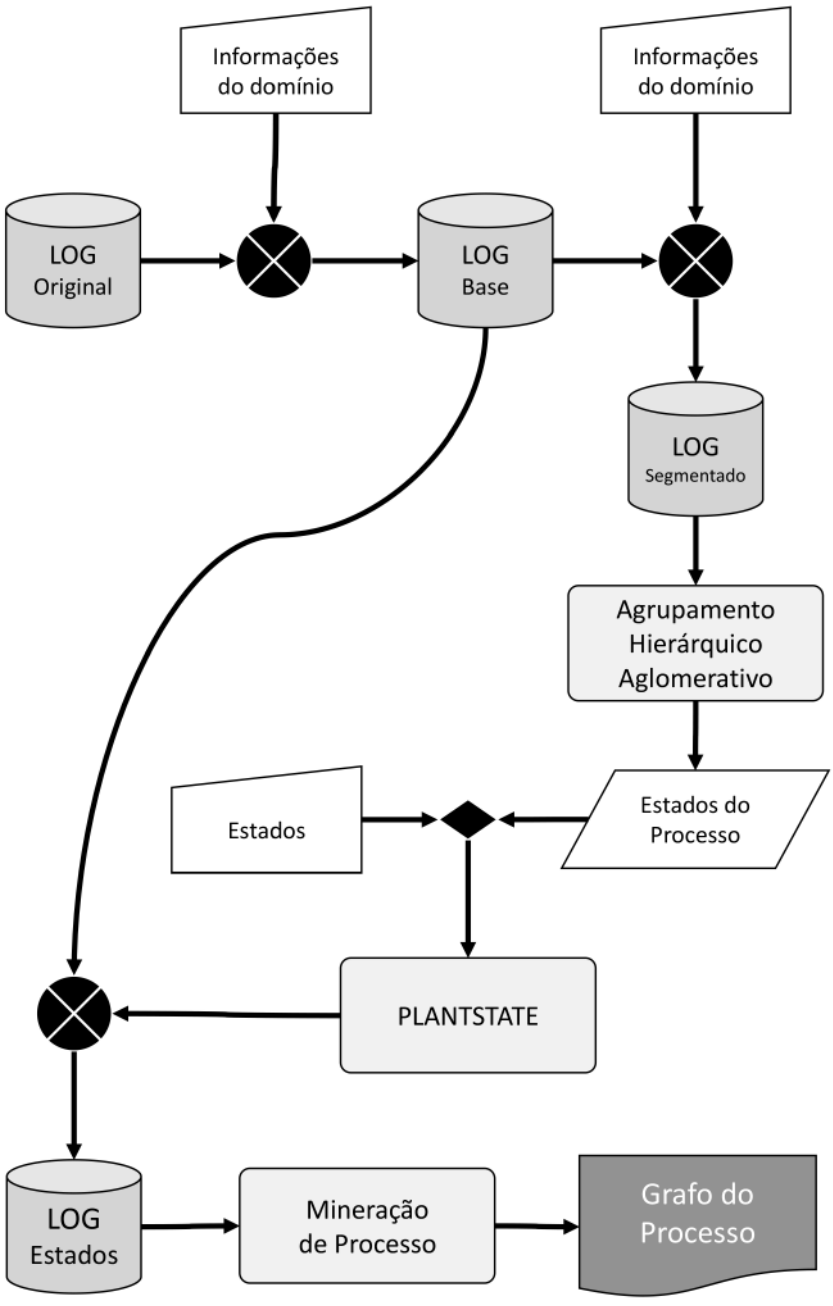


Figura 1.4: Fluxograma da metodologia.

PUC-Rio - Certificação Digital Nº 1912721/CA

1.5 Contribuições

As contribuições dessa pesquisas são referentes a primeira etapa da mineração de um processo, a de descobrimento. E podem ser aplicadas tanto em aplicações práticas quanto em pesquisas futuras. Sendo elas:

- Ferramenta para visualizar os estados das válvulas de uma unidade de coqueamento retardado. Com recursos de criação, edição, importação e exportação de estados. E permitindo a validação deles através da análise do fluxo de fluídos.
- Conjunto de técnicas para viabilizar o descobrimento um processo representado por um registro (*log*) que não contém identificação de casos (*caseid*).
- Modelo flexível de representação de processo, capaz de diferenciar relações de sequência e paralelismo referentes ao inicio, meio e fim de uma atividade.

PUC-Rio - Certificação Digital Nº 1912721/CA

2
Contextualização

Esta pesquisa consiste em extrair informações de dados que descrevem o funcionamento de uma unidade de coqueamento retardado. Neste capítulo vamos falar um pouco dos conceitos principais que envolvem o coqueamento retardado. Além disso, vamos deixar claro como vamos relacionar esses conceitos com as técnicas de agrupamento e mineração de processo utilizadas na pesquisa.

2.1
Coqueamento retardado

O coqueamento retardado consiste no aquecimento gradual de óleo, que é processado e quebrado em outros produtos. O processo é realizado alternadamente entre dois reatores por craqueamento térmico. Por exemplo, quando um reator A está em fase de enchimento - Figura 2.1 - o seu par, reator B, está sendo preparado para o próximo ciclo de enchimento. Ao final do enchimento do reator A o padrão se inverte - Figura 2.2.

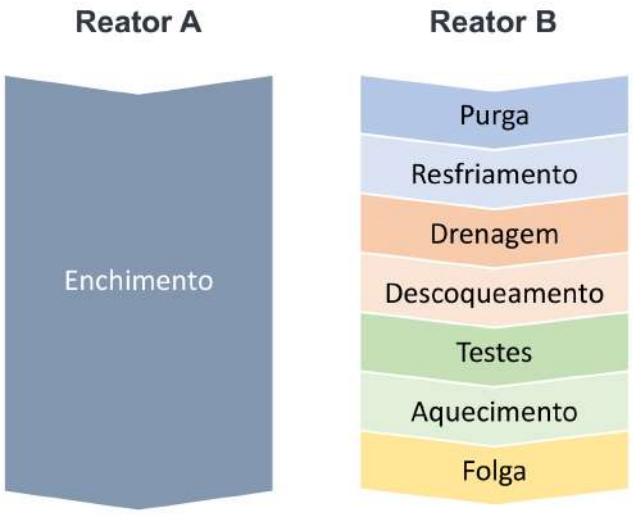


Figura 2.1: Fases do processo de coqueamento retardado durante o enchimento do reator A.

As fases representam funções do processo de coqueamento retardado. Ao final de uma fase o fluxo de fluidos na unidade é redirecionado para atender as demandas da próxima fase. Esse redirecionamento é feito através

PUC-Rio - Certificação Digital Nº 1912721/CA

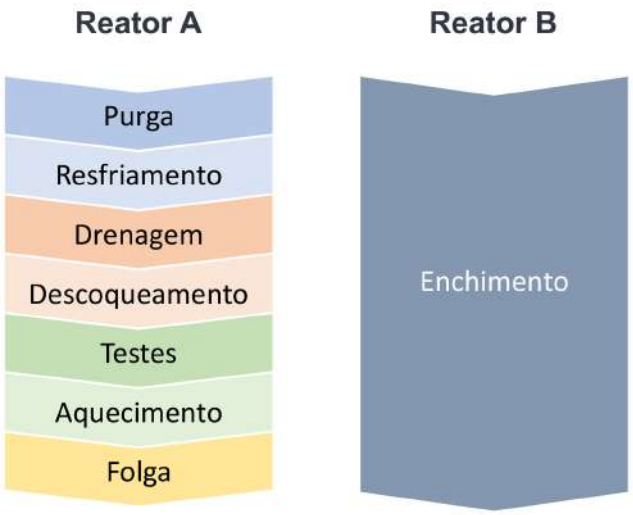


Figura 2.2: Fases do processo de coqueamento retardado durante o enchimento do reator B.

de válvulas e monitorado por sensores. Dessa forma, podemos assumir que as fases do processo são identificáveis pelo conjunto valores em válvulas e sensores específicos, como exemplificado na Figura 2.3.

O arranjo de físico dessas válvulas podem ser divididos em alinhamentos, que permitem a circulação de fluxo de fluidos para paras partes da unidade de produção. Como os alinhamentos também são composto por válvulas, podemos definir as fases do processo em função dos alinhamentos, conforme exemplificado na Figura 2.4.

Mesmo com o apoio de um especialista de operação, comunicar e visualizar os alinhamentos e as fases da operação é uma tarefa difícil. Assim iniciamos nossa abordagem pelo desenvolvimento de uma ferramenta de apoio a visualização de estados da unidade, batizada de `PLANTSTATE`, que detalharemos melhor no Capítulo 3.

PUC-Rio - Certificação Digital Nº 1912721/CA

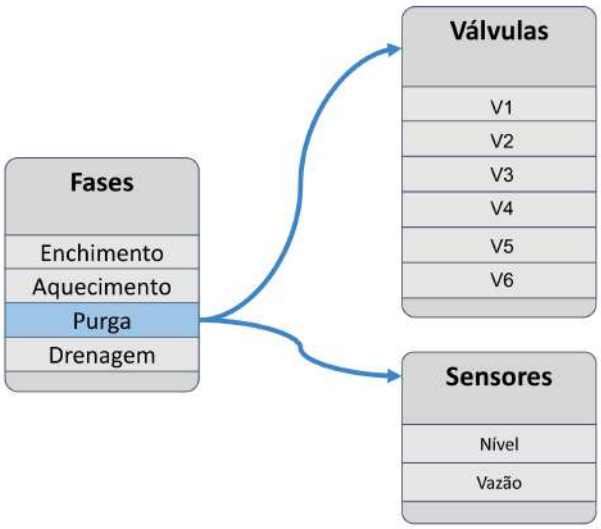


Figura 2.3: Exemplo de relação entre uma fase do processo de operação e suas válvulas e sensores.

PUC-Rio - Certificação Digital Nº 1912721/CA

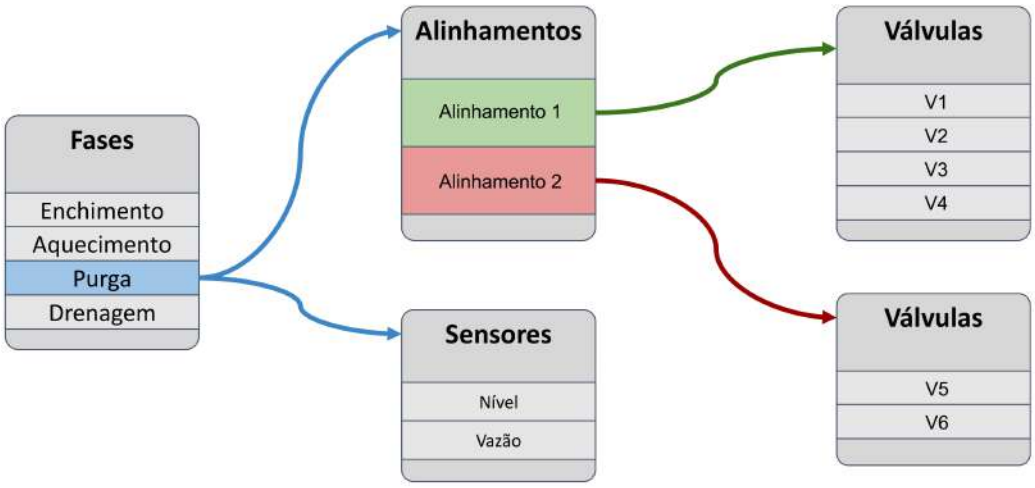


Figura 2.4: Exemplo de relação entre uma fase do processo de operação, seus alinhamentos e sensores. E dos alinhamentos e suas válvulas.

2.2 Equipamentos, grupos, estados, fases e alinhamentos.

Ao longo desse documento repetiremos diversas vezes alguns termos. Nesta seção vamos antecipar como estes termos conectam o domínio com as técnicas utilizadas.

Quando nos falamos dos dados do processo de coqueamento diversas vezes citaremos o termo *equipamentos*. Para o domínio em questão, normalmente estaremos nos referindo a *sensores* ou *válvulas*. Maior parte dos estudos serão realizados nos dados das *válvulas*. O uso do termo *equipamentos* se mantém para deixar claro que o método pode ser expandido para dispositivos além de *válvulas* e *sensores*.

Neste capítulos falamos de *fase* e *alinhamento*. Ambos representam conjuntos de de *equipamentos*, na grande maioria das vezes, *válvulas*. A diferença entre *fase* e *alinhamento* é semântica, depende da função que o conjunto exerce na operação.

Nos capítulos seguintes surgirão os termos *grupo* e *estado*. Um *estado* representa um conjunto de *válvulas*. Dessa forma, um *estado* pode ser um *alinhamento*, uma *fase* ou qualquer outro conjunto de *válvulas*. O termo *grupo* também representa um conjunto de *válvulas*. Mas utilizaremos o termo *grupo* quando esse conjunto de *válvulas* está diretamente relacionado a etapa de agrupamento.

Sendo assim, os termos *fase*, *alinhamento*, *grupo* e *estado* representam conjuntos de *equipamentos*, e na grande maioria das vezes esses *equipamentos* são *válvulas*. E o uso de um termo ou outro é de acordo com o que queremos expressar no contexto que ele se aplica.

PUC-Rio - Certificação Digital Nº 1912721/CA

3
Ferramenta PLANTSTATE

Neste capítulo vamos descrever as funcionalidades da ferramenta PLANTSTATE e comentar sobre a decisão de integra-la na solução proposta. Exceto quando dito o contrário, todas as funcionalidades citadas aqui estão implementadas na versão atual do PLANTSTATE. Por se tratar ainda de uma ferramenta experimental, não vamos entrar em detalhes sobre a implementação de cada funcionalidade.

Na Seção 3.1 vamos comentar um pouco sobre a motivação para o desenvolvimento da ferramenta. Em seguida, na seção 3.2 vamos detalhar a definição do *layout*. Na Seção 3.3 vamos descrever como é feito a simulação do fluxo para o estado atual da planta e sua representação será descrita na Seção 3.4. Terminaremos o capitulo com a Seção 3.5, que fala das funções de edição de grupos no PLANTSTATE.

PUC-Rio - Certificação Digital Nº 1912721/CA

3.1
Motivação

Alguns estados da planta podem ser considerados válidos ou não de acordo com a função resultante da configuração de todas as suas válvulas. Dessa forma a posição física da válvula no *layout* da planta é importante para o entendimento de sua função no sistema. A necessidade de avaliar a posição física de um grande volume de válvulas, de validar os agrupamentos obtidos de forma automática e de incluir conhecimento do domínio foram os principais motivadores para o desenvolvimento da primeira versão do PLANTSTATE.

Em algumas etapas da solução proposta os resultados precisam ser avaliados e editados pelo usuário. A primeira versão do PLANTSTATE já oferecia os recursos de visualização da planta e simulação do fluxo na unidade, então decidimos expandir as funcionalidades da ferramenta para atender as potenciais necessidades dos usuários desta solução. Ao identificar que a ferramenta ajudaria o usuário a aplicar e comunicar os resultados da solução proposta o PLANTSTATE se tornou parte dela. Para facilitar a disponibilização o PLANTSTATE está sendo desenvolvido como uma aplicação web, em HTML e *javascript*.

3.2 *Layout* e equipamentos

Para reproduzir o diagrama da unidade de coqueamento retardado é necessário informações sobre o seu *layout*. Ou seja, quais equipamentos existem naquela planta, suas posições, e suas conexões uns com os outros. Essas informações se encontram disponíveis em desenhos eletrônicos CAD, documentação em PDF e imagens nos *software* de monitoramento e operação da unidade. Porém, essas informações não estão estruturadas com as as informações necessárias para reconstruir o *layout* automaticamente. Dessa forma, até o momento, para cada nova unidade o *layout* deverá ser definido manualmente.

Para garantir alguma flexibilidade e reusabilidade, a definição do *layout* não será programada direto no código fonte do `PLANTSTATE`. O *layout* deve ser definido por um arquivo de configuração utilizando uma lista de declarações. No restante dessa seção vamos apresentar algumas declarações de *layout* e seus desenhos resultantes. E apresentar uma lista de equipamentos, sua declaração e uma breve descrição do seu propósito.

A maior parte das declarações em um *layout* será sobre as válvulas e as conexões de tubulação entre elas. Nosso primeiro conjunto de declarações, na Tabela 3.1, cria duas válvulas e conexão entre elas. A sintaxe de cada declaração é de um *array* em *javascript*. No caso das válvulas, e da maioria dos equipamentos, o formato padrão é `[<string: tipo de equipamento>, <string: id do equipamento>, <float: coordenada X>, <float: coordenada Y>]`. Para as conexões o formato padrão é `["pipe", <string: id do equipamento A>, <string: id do equipamento B>]`. O desenho do *layout* gerado a partir das declarações da Tabela 3.1 é o da Figura 3.1.

Tabela 3.1: Declarações do layout A.

Id	Declaração
1	`["mv","001", 100, 100]`
2	`["mv","002", 200, 200]`
3	`["pipe","001","002"]`

Na Figura 3.1 o `PLANTSTATE` desenhou a tubulação como segmentos de retas ortogonais entre as válvulas MV-001 e MV-002. Para facilitar a criação do *layout* pelo usuário o `PLANTSTATE` tenta encontrar o caminho ortogonal com a menor quantidade de segmentos entre os componentes, considerando as conexões disponíveis nos componentes. Dessa forma, ao aproximarmos os valores da coordenada X das válvulas MV-001 e MV-002, como nas declarações da Tabela 3.2, o `PLANTSTATE` encontrou outra forma de desenhar as tubulações,

PUC-Rio - Certificação Digital Nº 1912721/CA

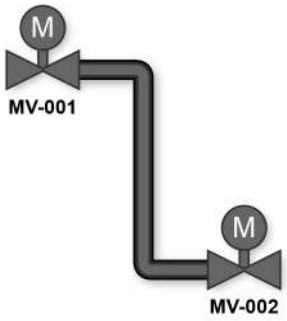


Figura 3.1: *Layout* gerado pelas declarações na Tabela 3.1.

como podemos ver na Figura 3.2. Repare que reduzimos a declaração de criação tubulação apenas para `["pipe"]`, quando declaramos dessa forma o `PLANTSTATE` considera que queremos criar uma conexão entre os dois últimos equipamentos declarados.

Tabela 3.2: Declarações do layout B.

Id	Declaração
1	`["mv","001", 180, 100]`
2	`["mv","002", 200, 200]`
3	`["pipe"]`

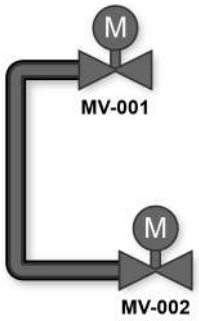


Figura 3.2: *Layout* gerado pelas declarações na Tabela 3.2

Como na criação das tubulações o `PLANTSTATE` considera as conexões disponíveis entre os equipamentos, a ordem de declaração das tubulações afeta o *layout* resultante. Nas Tabelas 3.3 e 3.4 temos declarações idênticas de equipamentos e conexões, a única diferença é a ordem das declarações das conexões. Os *layouts* resultantes são os das Figuras 3.3 e 3.4.

PUC-Rio - Certificação Digital Nº 1912721/CA

Tabela 3.3: Declarações do layout C.

Id	Declaração
1	`["mv","001", 100, 200]`
2	`["mv","002", 300, 250]`
3	`["mv","003", 200, 100]`
4	`["pipe","001","003"]`
5	`["pipe","001","002"]`

Tabela 3.4: Declarações do layout D.

Id	Declaração
1	`["mv","001", 100, 200]`
2	`["mv","002", 300, 250]`
3	`["mv","003", 200, 100]`
4	`["pipe","001","002"]`
5	`["pipe","001","003"]`

PUC-Rio - Certificação Digital Nº 1912721/CA

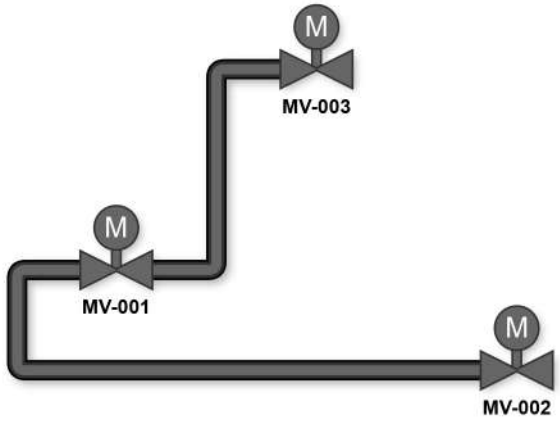


Figura 3.3: *Layout* gerado pelas declarações na Tabela 3.3.

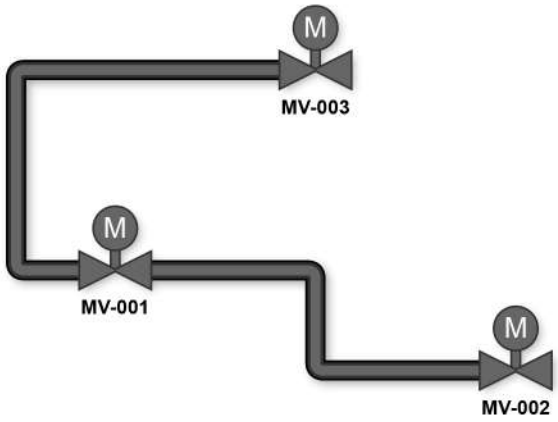


Figura 3.4: *Layout* gerado pelas declarações na Tabela 3.4.

Para finalizar essa bateria de exemplos, na Tabela 3.5 fizemos apenas uma alteração em relação a Tabela 3.4, que foi adicionar mais 1 elemento na declaração do equipamento 001. Esse elemento é opcional e define a rotação do equipamento no *layout*. Como resultado essa rotação afetou como as tubulações foram geradas, como podemos ver na Figura 3.5.

Tabela 3.5: Declarações do layout E.

Id	Declaração
1	`["mv","001", 100, 200, 90]`
2	`["mv","002", 300, 250]`
3	`["mv","003", 200, 100]`
4	`["pipe","001","002"]`
5	`["pipe","001","003"]`

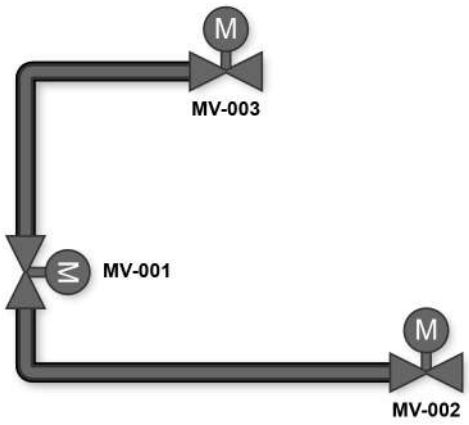
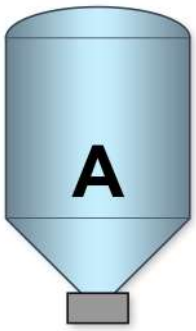


Figura 3.5: *Layout* gerado pelas declarações na Tabela 3.5

Com isso detalhamos os principais tipos de declarações que utilizamos para gerar o *layout* da planta para esta pesquisa, representado na Figura 3.6, que foi toda gerada a partir de declarações como as descritas nessa seção. Na Tabela 3.6 listamos todos os equipamentos disponíveis atualmente no `PLANTSTATE`.

PUC-Rio - Certificação Digital Nº 1912721/CA

Tabela 3.6: Tipos de equipamentos e suas representações e descrições.

Representação	**Tipo**	**Descrição**
	mv	Válvula do tipo MV, possui duas configurações: aberto e fechado.
	xv	Válvula do tipo XV, possui duas configurações: aberto e fechado.
	fv	Válvula do tipo FV, possui duas configurações: aberto e fechado.
	m4	Switch, possui duas configurações para cada uma das 4 conexões: aberto e fechado.
	in	Entrada de fluxo no sistema.
	out	Saída de fluxo do sistema.
	joint	Junta, conecta até 4 tubulações.
	gap	Criado automaticamente quando duas tubulações não conectadas se cruzam.
	reactor	Reator, possui duas configurações: aberto e fechado.

PUC-Rio - Certificação Digital Nº 1912721/CA

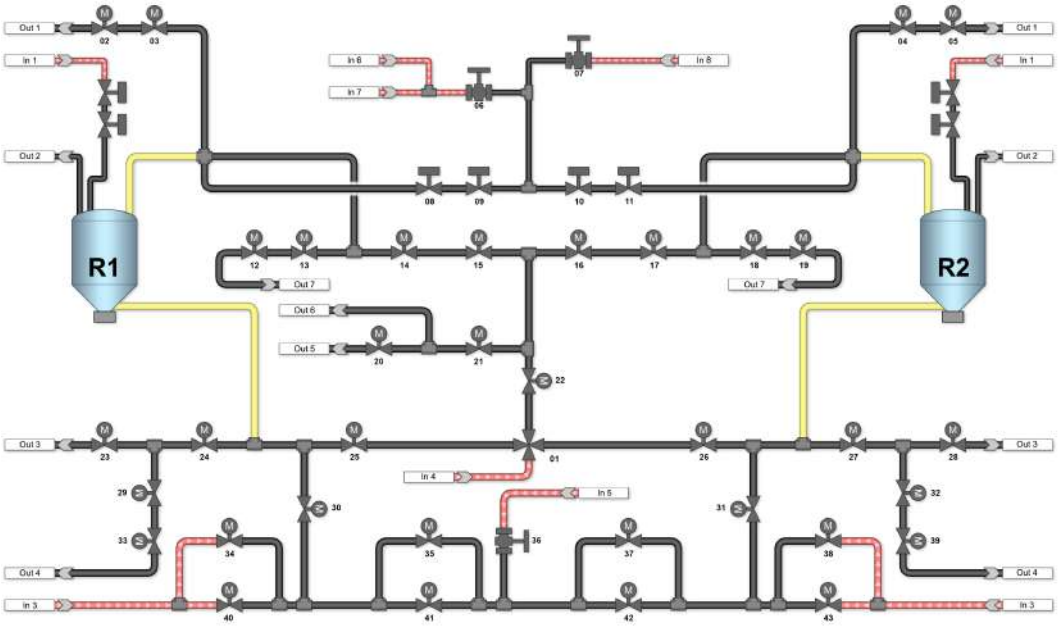


Figura 3.6: *Layout* final da planta usada neste estudo.

PUC-Rio - Certificação Digital Nº 1912721/CA

3.3 Simulação do fluxo na planta

Além da representação gráfica do *layout*, baseado na configuração de cada equipamento o PLANTSTATE é capaz de fazer uma simulação do fluxo de fluidos na planta. Essa não é uma simulação baseada nas propriedades físicas dos fluídos, pois uma simulação desse tipo demandaria muito mais recursos e informações do que temos disponível. O nosso objetivo é definir a direção do fluxo em cada trecho de tubulação de forma todo *inlet* possua um caminho válido, unidirecional, até algum *outlet*.

Para resolver esse problema vamos visualizar o layout da planta como um grafo direcionado, onde os equipamentos são os nós e as tubulações as arestas. A simulação então é feita encontrando o caminho mais curto entre as entradas de fluxo no sistema e suas saídas, em seguida preenchendo os trechos que não fizeram parte desse caminho.

O algoritmo aplicado para encontrar esse caminho mais curto foi o BFS(Bundy and Wallen, 1984). Onde a partir de um nó inicial -entrada de fluido- cada vizinho -conectado por uma aresta- é visitado e marcado como distância 1. Em seguida os vizinhos dos vizinhos são visitados e marcados como distância 2, e assim sucessivamente. Ao final desse processo todas as saídas possíveis para aquele nó de entrada estarão marcadas, e a de menor distância é selecionada. E então basta repetir o processo para cada nó de entrada de fluxo do sistema.

O grafo foi construído considerando que cada equipamento é um nó, e cada conexão por tubulação gera duas arestas, uma em cada direção da conexão, gerando o grafo G_x, que será a base da simulação. A simulação é feita através de um novo grafo G_y, gerado a partir do grafo G_x, considerando as configurações de aberto ou fechado dos equipamentos, como definido no Algoritmo 1:

Algoritmo 1 Simulação, criação do grafo de simulação G_y

```
1: Nós de G_y = Nós de G_x
2: para todo aresta E de G_x faça
3:     C_o = configuração do equipamento do nó de origem da aresta E
4:     se C_o = "aberto"ou C_o = "nenhum" então
5:         Adiciona aresta E a G_y
6:     fim se
7: fim para
```

Com o grafo G_y a simulação é feita em duas etapas, na primeira etapa, definida pelo Algoritmo 3, vamos garantir o fluxo entre os *inlets* e um *outlet*

PUC-Rio - Certificação Digital Nº 1912721/CA

mais próximo. Na segunda etapa, definida pelo Algoritmo 4, vamos usar BFS (Bundy and Wallen, 1984) para preencher os trechos de tubulação que não fizeram parte de nenhum caminho mais curto. No Algoritmo 2 definimos um procedimento que, dado um caminho, fixa a direção possível daquele fluxo no grafo. Ao final do Algoritmo 4 as arestas resultantes em G_y definem o fluxo de fluido na planta.

Algoritmo 2 Simulação, fixar direção do fluxo em um conjunto de arestas

```
1: Procedimento FixarDirecaoDoFluxo(E, G)
2:     para todo aresta e em E faça
3:         i = aresta e com origem e destino invertidos.
4:         se i ∈ G então
5:             Remover i de G
6:         fim se
7:     fim para
8: fim Procedimento
```

Algoritmo 3 Simulação, busca do fluxo principal

```
1: N_i = conjunto de nós de G_y que representam os inlet.
2: N_o = conjunto de nós de G_y que representam os outlet.
3: P_x = conjunto de caminhos principais, inicialmente vazio.
4: para todo nó n em N_i faça
5:     P_i = conjunto de caminhos mais curtos em G_y entre n e todos N_o.
6:     P_m = menor caminho de P_i
7:     Adiciona P_m a P_x
8:     E = arestas de P_m
9:     FixarDirecaoDoFluxo(E, G_y)
10: fim para
```

PUC-Rio - Certificação Digital Nº 1912721/CA

Algoritmo 4 Simulação, busca do fluxo secundário

1: N = conjunto com todos os nós em P_x
2: E_x = conjunto de todas as arestas resultantes do BFS em G_y tendo N como nós iniciais.
3: FixarDirecaoDoFluxo(E_x, G_y)
4: **para todo** par de nós n_1, n_2 de G_y **faça**
5: e_1 = aresta com origem em n_1 e destino n_2
6: e_2 = aresta com origem em n_2 e destino n_1
7: **se** $e_1 \in G_y$ & $e_2 \in G_y$ **então**
8: Remover e_1 ou e_2 de G_y
9: **fim se**
10: **fim para**

3.4 Interatividade e representação

Além de exibir resultados gerados por outros métodos o `PLANTSTATE` permite simular o fluxo na planta de forma interativa, sendo possível alterar a configuração dos equipamentos clicando sobre eles. Construímos um *layout* simples para exemplificar o funcionamento interativo do `PLANTSTATE`. Nesse *layout* vamos explorar 3 configurações, que resultam nos *layouts* das Figuras 3.7, 3.8 e 3.9.

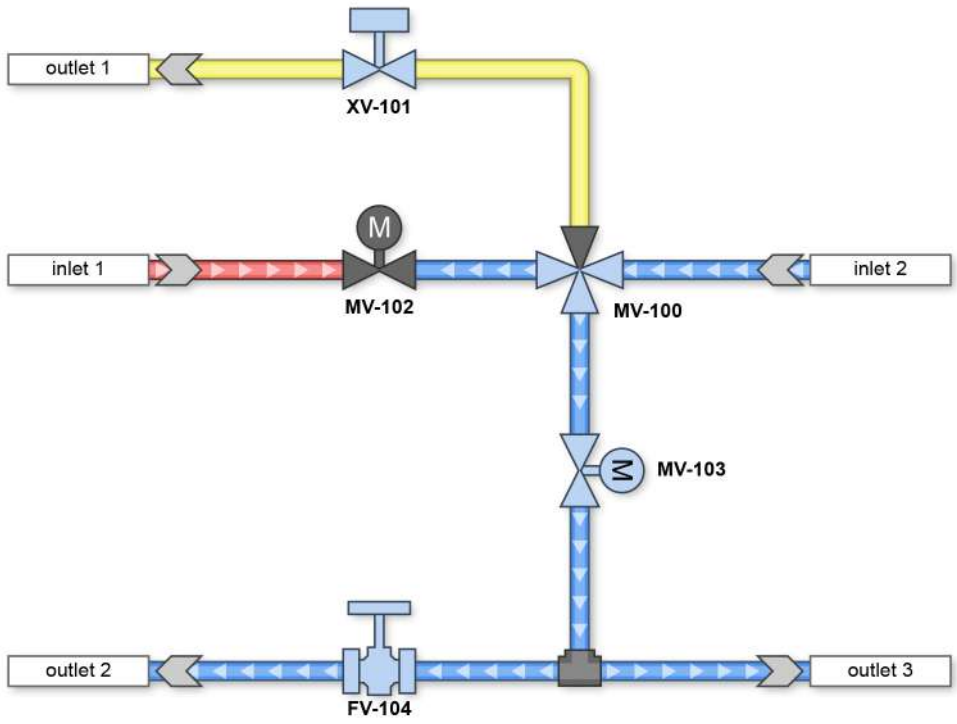


Figura 3.7: Fluxo da planta no *layout* de exemplo na configuração A.

PUC-Rio - Certificação Digital Nº 1912721/CA

Na configuração A – Figura 3.7 - temos a válvula MV-103 na configuração fechada, representada em cinza escuro, e as válvulas XV-101, MV-103 e FV-104 na configuração aberta, representada pela cor azul. A válvula MV-100, o *switch*, possui 3 conexões abertas e uma fechada, também representado pelas mesmas cores.

Ainda na Figura 3.7 temos 3 tipos de representação de fluxo nas tubulações. Tubulações em azul representam fluxos que possuem um caminho entre um *inlet* e um *outlet*. Tubulações em vermelho representam fluxos que entraram no sistema por algum *inlet* mas não conseguem chegar a um *outlet*. Tanto as regiões em vermelho ou em azuis possuem setas que indicam a direção do fluxo. Tubulações em amarelo representam trechos sem fluxo porém conectados a equipamentos com a configuração aberta, indicando que talvez fosse esperado algum fluxo por aquela região. Tubulações em cinza escuro, não presente na Figura 3.7, representam trechos de tubulação sem fluxo.

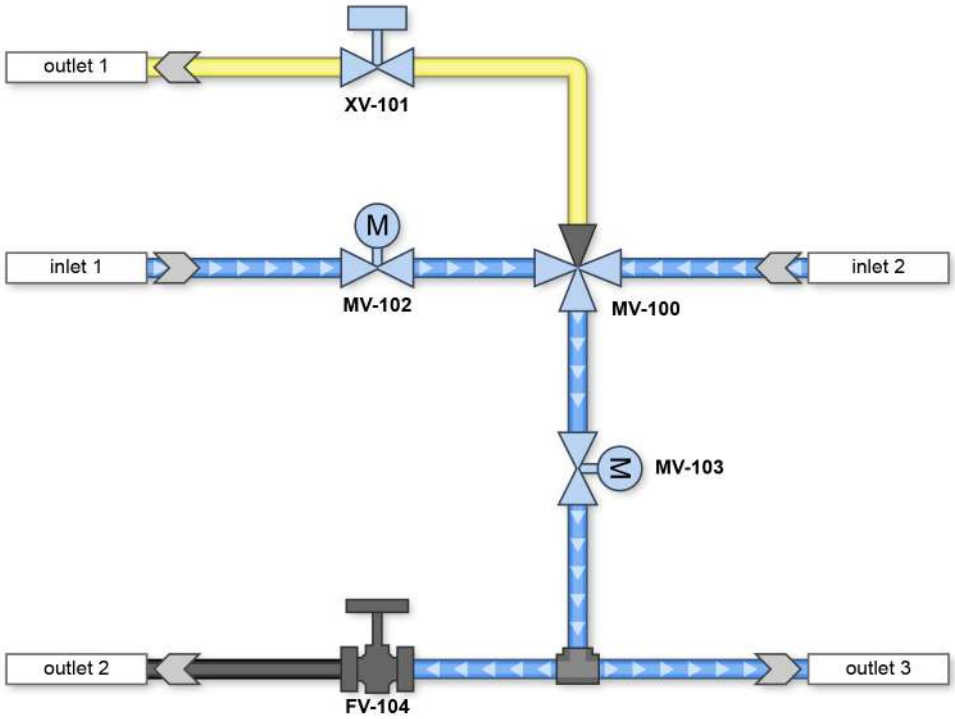


Figura 3.8: Fluxo da planta no *layout* de exemplo na configuração B.

A configuração B tem duas diferenças em relação a configuração A: válvula FV-104 fechada e válvula MV-102 aberta. A abertura da válvula MV-102 eliminou as regiões em vermelho, agora todos os fluxos dos *inlets* chegam a algum *outlet*. Ainda, o fechamento da válvula FV-104 corta o fluxo para o *outlet* 2. Uma característica que reflete o resultado do algoritmo de visualização de fluxo é no trecho de tubulação entre as válvulas MV-102 e MV-100. Na Figura

PUC-Rio - Certificação Digital Nº 1912721/CA

3.7 o fluxo está para a esquerda, em direção a válvula fechada MV-102, na Figura 3.8 o fluxo está para a direita devido a abertura da válvula MV-102.

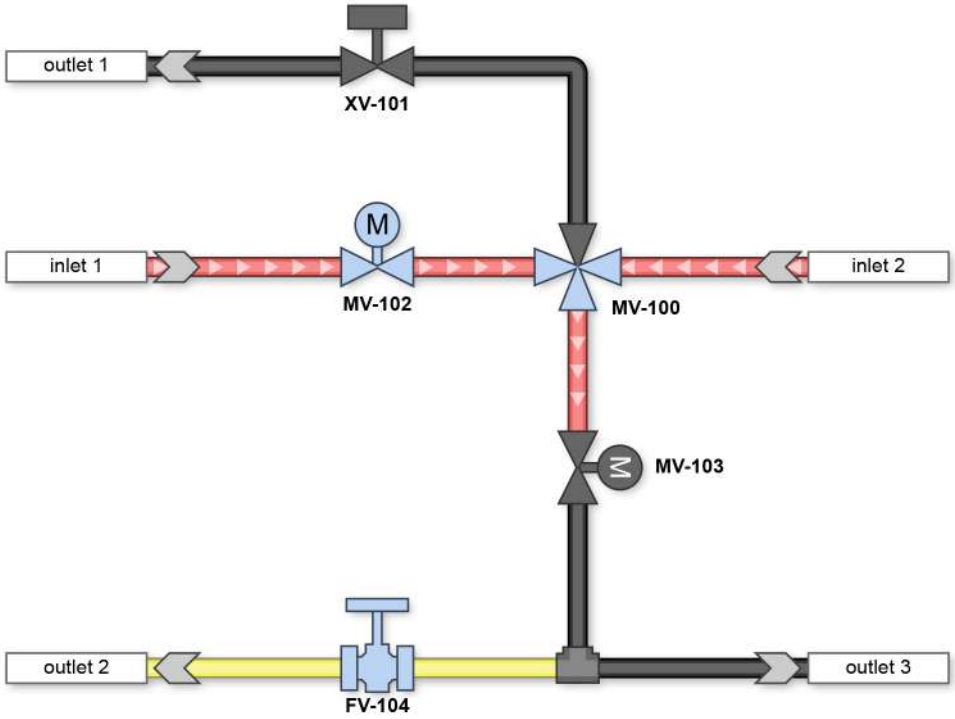


Figura 3.9: Fluxo da planta no *layout* de exemplo na configuração C.

Por último, na configuração C, que gerou o *layout* da Figura 3.9, fechamos as válvulas MV-103 e XV-101. Como indicado pelas tubulações em vermelho, o fechamento da válvula MV-103 cortou completamente a saída de fluxo do sistema.

3.5 Persistência e edição

Em uma das etapas da preparação da base de dados para a mineração do processo precisamos avaliar diversos estados (conjuntos de válvulas). Essa avaliação pode resultar na rejeição ou aceitação completa ou parcial daqueles estados. Implementamos ferramentas para facilitar a navegação entre esses estados, alterar as válvulas pertencentes a um estado, excluir e adicionar novos estados e exportar essas alterações em JSON.

Na Figura 3.10 temos a interface que lista os estados importados e suas opções de edição. Cada linha da lista é clicável, adicionando um efeito de brilho nas válvulas que fazem parte daquele estado, facilitando a comparação com as configurações atuais da planta.

Clicar em algum botão *edit* na lista da Figura 3.10 nos leva a interface de edição daquele estado específico – Figura 3.11. Nesse momento o PLANTSTATE

PUC-Rio - Certificação Digital Nº 1912721/CA

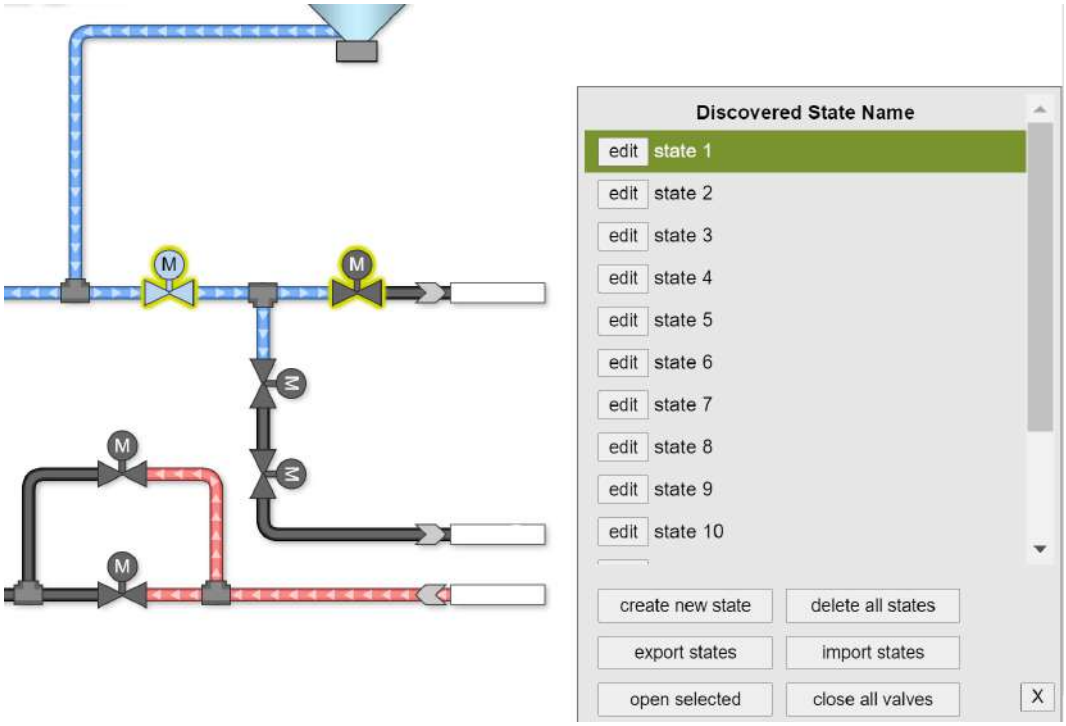


Figura 3.10: PLANTSTATE: Lista de estados e opções de edição.

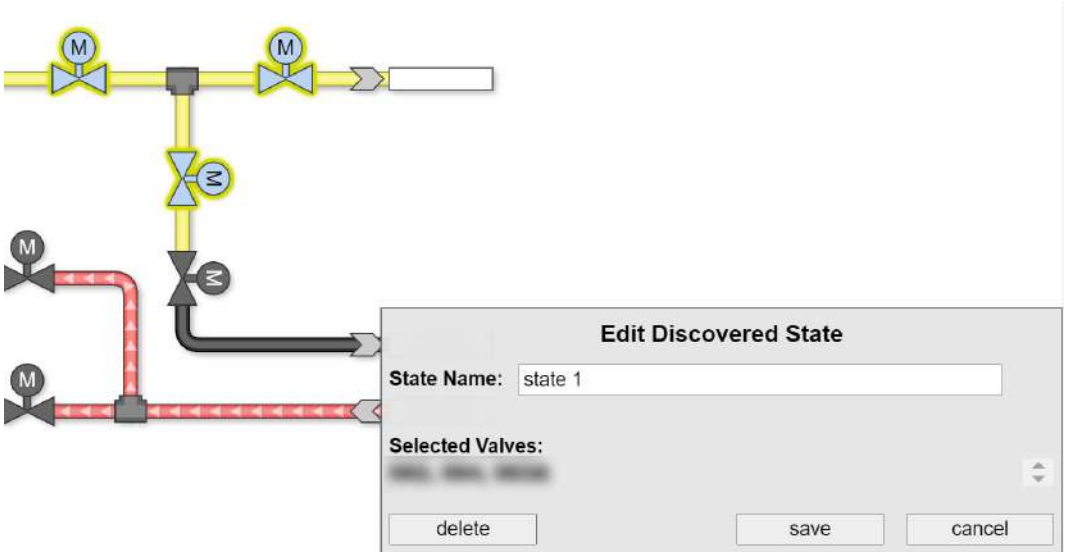


Figura 3.11: PLANTSTATE: Edição de um grupo.

PUC-Rio - Certificação Digital Nº 1912721/CA

fecha todas as válvulas da plana e abre apenas as pertencentes ao grupo em edição. Nesse modo de edição, qualquer válvula clicada será excluída ou editada do estado, dependendo se já faz parte do estado em edição ou não. O usuário pode manter ou descartar as alterações, além de poder alterar o nome do estado ou excluir completamente o estado.

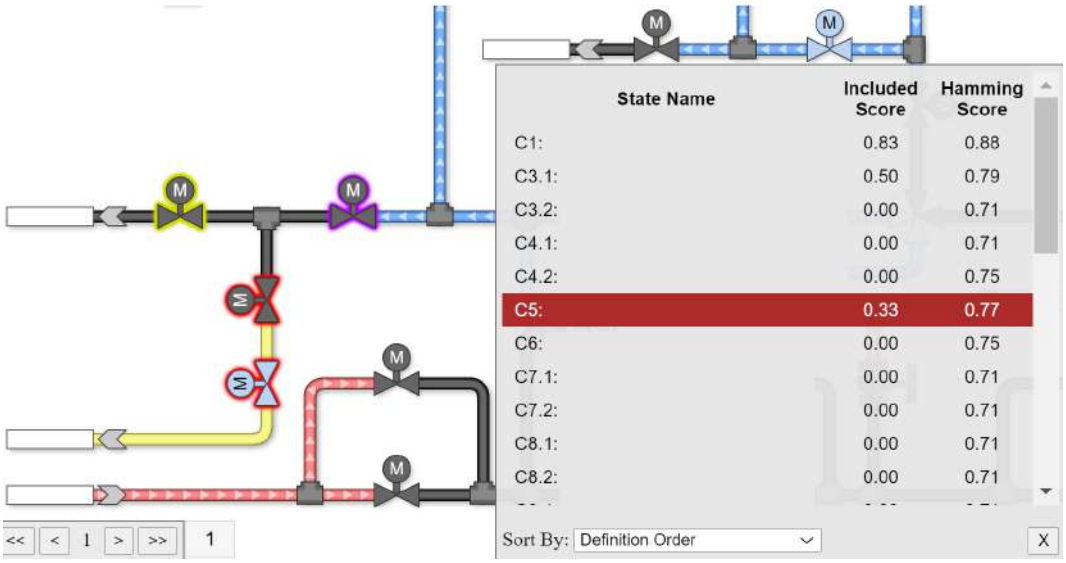


Figura 3.12: PLANTSTATE: Interface de inspeção dos estados de referência.

Na Figura 3.12 temos a lista de estados de referência no PLANTSTATE, que são estados utilizados para comparação com os estados descobertos ou a configuração atual da planta. Nessa lista temos dois *scores*, que auxiliam o avaliador a encontrar rapidamente qual a estado a configuração atual da planta mais se aproxima. Assim como na verificação dos estados para edição, ao clicar em uma linha da lista de estados, o PLANTSTATE realça com um brilho as válvulas pertencentes aquele estado. Esse recurso de realce está ativo para os dois tipos de estados - descoberto e referência. Realçado em verde temos as válvulas que pertencem apenas ao estado descoberto. Em vermelho as válvulas que pertencem apenas ao estado de referência. E em roxo as válvulas que pertencem a ambos os estados.

PUC-Rio - Certificação Digital Nº 1912721/CA

4
Os dados: Estruturação, processamento e enriquecimento

Neste capítulo vamos descrever os passos feitos para limpar, unificar e ajustar a base de dados original de modo a possibilitar a mineração do processo. Iniciaremos com uma visão geral da proposta na Seção 4.1. Na Seção 4.2 vamos detalhar o processamento da base de dados original. Em seguida, na Seção 4.3, detalharemos como os dados foram agrupados para dar suporte ao enriquecimento da base de dados descrito na Seção 4.4.

4.1
Visão geral da proposta de solução

Nesta seção vamos descrever, em etapas, a visão geral da proposta de solução.

1. Filtrar os equipamentos, deixando somente os equipamentos referentes ao coqueamento retardado em um par de reatores.

2. Filtrar as entradas na base de dados das válvulas, deixando apenas os *status* aberto e fechado.

3. Uniformizar as entradas na base de dados das válvulas para que todas possuam apenas valores 0 ou 1, representado respectivamente os *status* de válvula fechada e válvula aberta, gerando o *log* base.

4. Aplicar técnicas de agrupamento para agrupar os dados de forma que válvulas com um padrão de fechamento e abertura similar pertençam a um mesmo grupo.

5. Utilizar a ferramenta `PLANTSTATE` para inspecionar, ajustar e renomear os grupos para que aproximem melhor as funções dentro da operação de coqueamento retardado, criando os estados do processo.

6. Identificar e registrar quando cada um dos estados do processo se manifesta no *log* base, gerando o *log* de estados.

7. Minerar as relações entre os estados do processo a partir de seus momentos de ativação e desativação no *log* de estados, gerando os contextos.

PUC-Rio - Certificação Digital Nº 1912721/CA

8. Simplificar e enriquecer os contextos baseado em conceitos de mineração de processo.

9. Construir o grafo do processo a partir dos contextos finais.

Detalharemos até a etapa 6 nas próximas seções deste capítulo. As etapas seguintes serão exploradas no Capítulo 5.

4.2 Log Base

Nesta seção vamos descrever o processamento na base de dados original que vai nos levar ao *log* base. Na Seção 4.2.1 falamos um pouco sobre as características da base de dados e suas limitações. Na Seção 4.2.2 descrevemos as filtragens e transformações feitas.

4.2.1 Características

A base de dados original consiste em diversos arquivos CSV. Os dados são referentes a atividade de cerca de 400 equipamentos, na sua maioria válvulas e sensores, de uma unidade de coqueamento retardado. As atividades são referentes a três meses consecutivos. O registro de atividades de cada equipamento está separado em um CSV próprio, e a maioria dos equipamentos possui dois arquivos CSV. Um que contém informações de seu *status* por *timestamp*. E outro contém uma leitura de seu valor por *timestamp*. A unidade desse valor varia de acordo com o tipo de equipamento, podendo ser posição/abertura, temperatura, pressão, vazão ou nível. Não há segmentação entre os equipamentos que são relacionados e nem uma identificação clara do que seria um caso no processo.

Em uma exploração inicial dos dados das válvulas identificamos que quatro *status* principais estavam registrados nas suas atividades das válvulas: aberto, fechado, transição e falha. Nesse descartamos os *status* de transição e falha e mantivemos apenas os de aberto e fechado. Em seguida selecionamos algumas válvulas e plotamos em um *heatmap* com os seus momentos de abertura e fechamento em várias sequências de fatias de tempo, como podemos ver na Figura 4.1.

Nesse *heatmap* identificamos que as regiões que representam o período de abertura das válvulas indicam que existe uma lógica de complementação e coatividade entre elas.

Após uma inspeção mais detalhada nos dados identificamos que em diversos momentos essa relação de coatividade e complementação não se

PUC-Rio - Certificação Digital Nº 1912721/CA

Figura 4.1: Análise dos momentos de abertura e fechamento de alguns equipamentos, caso normal.

Figura 4.2: Análise dos momentos de abertura e fechamento de alguns equipamentos, caso discrepante.

PUC-Rio - Certificação Digital Nº 1912721/CA

mantinha, como na Figura 4.2, que o equipamento EP7 -uma válvula- se matem aberto por um período muito longo sem nenhuma outra válvula acompanhar esse padrão. Essa discrepância poderia indicar uma mudança no regime de operação da unidade, como sabemos que existe. Mas essas mudanças na grande maioria das vezes se repetem por mais alguns ciclos. Como informado pelo especialista do domínio, quando a posição de uma válvula chega a um valor muito próximo de 0% ela deve iniciar o *status* fechada. Quando chega próxima a 100% o *status* deve ser aberto. Na maioria dos momentos os dados estão conforme essa regra. Dessa forma reunimos e alinhamos em uma única representação as base de dados de *status* e valor para um mesmo equipamento, como pode ser conferido na Figura 4.3. Essa figura percebemos gradientes na linha *INFO-VALUE*, que representam sequencias em que os valores aumentam ou diminuem gradativamente, indicando o processo de abertura ou fechamento de uma válvula. Logo abaixo, na linha *INFO-CONFIG*, representando os momentos que o *status* está aberto, podemos perceber que, perto das fatias de tempo 158 e 185, existe uma relação entre as duas linhas, com o *status* mudando para aberto quando o valor chega próximo a 1.0 (100%). E quando o valor chega a 0%, perto das fatias de tempo 172 e 200, o *status* muda para fechado. Ainda nesse gráfico plotamos também as atividades para os *status* que descartamos inicialmente, a fim de identificar uma relação com os outros

Figura 4.3: Alinhamento entre os valores e os status de um mesmo equipamento. Caso normal.

valores.

Em muitos trechos nos dados essa consistência entre o valor e *status* não se sustenta, como podemos ver na Figura 4.4, aparentemente o *status* deveria mudar para aberto próximo a fatia 257. Na Figura 4.5 onde o *status* deveria mudar para fechado próximo a fatia 356. Também encontramos regiões onde os dados aparentam estar degenerados, como podemos ver nas Figuras 4.6 e 4.7.

Figura 4.4: Alinhamento entre os valores e os status de um mesmo equipamento. Caso onde há falha na identificação da abertura.

Esses ruídos podem impactar significativamente os resultados dos métodos aplicados a essa base de dados, principalmente os mais sensíveis a ruídos. Até o momento não conseguimos encontrar uma forma automática de filtrar os ruídos, pois a discrepância se manifesta tanto na base de dados de valores quanto na base de dados de *status*.

4.2.2 Filtragens e transformações

Para reduzir o problema a uma base de dados na qual seus equipamentos tenham alguma relação entre si, decidimos por manter apenas os equipamentos de dois reatores. A seleção de quais equipamentos fazem parte desse recorte foi feita manualmente, utilizando informações fornecidas pelos especialistas

PUC-Rio - Certificação Digital Nº 1912721/CA

Figura 4.5: Alinhamento entre os valores e os status de um mesmo equipamento. Caso onde há falha na identificação do fechamento.

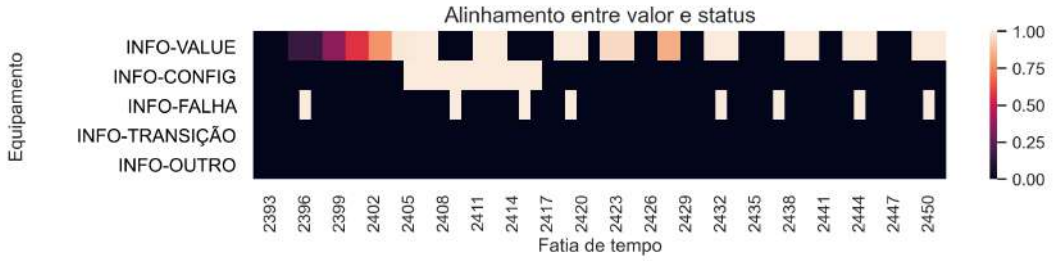


Figura 4.6: Alinhamento entre os valores e os status de um mesmo equipamento. Degeneração 1.

PUC-Rio - Certificação Digital Nº 1912721/CA

do domínio, indicadas nos painéis de controle e monitoramento da unidade e avaliando os diagramas nos manuais de operação.

As informações dos sensores são importantes para resolver ambiguidades e enriquecer a análise da unidade de operação durante as mudanças de estados. Mas, por questões de escopo, neste documento vamos dar foco apenas as válvulas.

Por questões de confidencialidade, renomearemos as válvulas para valores genéricos e ocultaremos informações sensíveis.

Transformamos os valores que representam o *status* de uma válvula de texto para numérico, onde 0 representa o *status* fechado e 1 representa o *status* aberto. Os *status* de falha e transição foram descartados.

O *switch* é uma válvula especial que possui pelo menos 4 *status* diferentes para a configuração aberta, um para cada posição. Cada posição do *switch* direciona o fluxo numa direção diferente. Para manter uma uniformidade nos dados transformamos os dados do *switch* em 4 válvulas distintas, cada uma representando uma posição de abertura do *switch*. Para manter a coerência com a função mecânica do *switch* adicionamos uma regra na construção da base de dados que quando uma das 4 válvulas que representam o *switch* está registrada como aberta as outras 3 devem estar fechadas.

Por fim, unificamos todos os CSV independentes das válvulas em uma

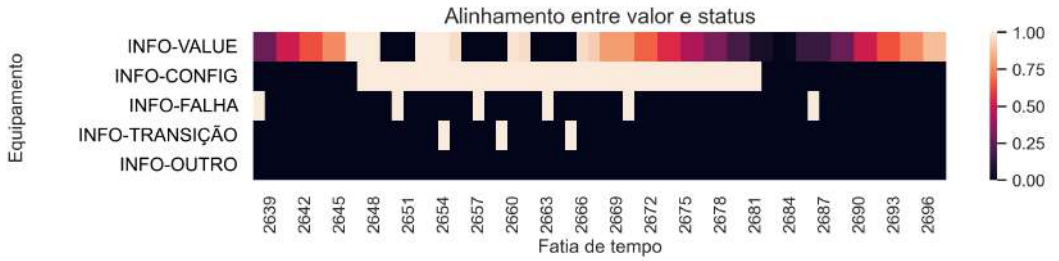


Figura 4.7: Alinhamento entre os valores e os status de um mesmo equipamento. Degeneração 1.

única base de dados, como exemplificado na Figura 4.8. Nessa base de dados cada entrada possui como chave o *timestamp*, e cada variável da entrada representa uma válvula. O valor de cada variável é 0 ou 1, dependendo se a válvula estava aberta ou fechada naquele *timestamp*. Daqui para frente iremos nos referir a essa base de dados unificada como *log* base ou L_b.

timestamp	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
00:00:00	0	1	1	0	1	1	1	0	0	1	1	1	1
00:27:00	0	0	1	1	1	0	0	0	0	1	0	0	0
00:31:00	0	1	0	0	0	0	1	1	1	0	0	1	1
00:34:00	0	1	0	1	0	1	1	0	0	0	0	0	1
00:39:00	1	1	0	1	1	0	0	0	1	1	0	1	0
00:43:00	0	0	1	0	0	0	0	0	1	1	0	1	0
00:51:00	1	0	0	0	0	0	0	1	1	0	1	1	0
00:57:00	1	1	0	0	0	1	1	1	1	0	1	0	1
00:59:00	1	1	0	1	1	1	1	0	0	0	0	0	0
01:22:00	0	1	0	0	1	0	1	0	0	0	1	1	0
01:42:00	1	1	1	1	0	1	0	0	1	0	0	1	1
01:43:00	0	0	1	0	0	0	1	1	1	0	1	0	0
01:48:00	0	0	0	0	1	1	1	0	0	1	1	0	1
01:49:00	1	1	0	0	0	1	0	1	0	1	1	0	1

Figura 4.8: Exemplo de um *log* base.

4.3 Agrupamento

Nesta seção vamos descrever como aplicamos o agrupamento hierárquico aglomerativo (AHC (Day and Edelsbrunner, 1984)) para encontrar os grupos de válvulas que possuem um padrão de funcionamento similar. Vamos discutir duas formas de realizar esses agrupamentos, suas vantagens e limitações e como

PUC-Rio - Certificação Digital Nº 1912721/CA

inspecionar a qualidade dos grupos encontrados. O produto desta seção é um conjunto de grupos de válvulas, os estados do processo, que servirá para a geração de um *log* de estados.

4.3.1 Log segmentado

O *log* base possui uma densidade de registro não uniforme em relação ao tempo. Ou seja, em certos momentos, temos uma grande quantidade de dados registrados em um período curto de tempo. Em outros momentos passamos longos períodos sem registro algum. O AHC não considera as variações de tempo entre as entradas. Então, nesse formato, o algoritmo faz uma interpretação desbalanceada dos dados. Para gerar uma base onde cada entrada possui uma relevância similar, vamos dividir o *log* base em uma sequência de segmentos de 30 minutos de duração.

Dado que em 30 minutos uma válvula pode mudar de aberto para fechado diversas vezes temos que definir como um segmento irá expressar a configuração de uma válvula. Adotamos a definição que o valor de uma válvula no *log* segmentado será um fator entre 0 e 1, que expressa a proporção daqueles 30 minutos do segmento que aquela válvula permaneceu aberta.

Exemplificamos essa transformação nas Figuras 4.9 e 4.10. Na Figura 4.9 temos a segmentação do *log* base, mas ainda mantendo seus valores originais. Na Figura 4.10 substituímos os valores de cada *timestamp* para o fator de abertura para o segmento, finalizando a transformação para o *log* segmentado.

segmento	timestamp	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	00:00:00	0	1	1	0	1	1	1	0	0	1	1	1	1
	00:27:00	0	0	1	1	1	0	0	0	0	1	0	0	0
2	00:31:00	0	1	0	0	0	0	1	1	1	0	0	1	1
	00:34:00	0	1	0	1	0	1	1	0	0	0	0	0	1
	00:39:00	1	1	0	1	1	0	0	0	1	1	0	1	0
	00:43:00	0	0	1	0	0	0	0	0	1	1	0	1	0
	00:51:00	1	0	0	0	0	0	0	1	1	0	1	1	0
	00:57:00	1	1	0	0	0	1	1	1	1	0	1	0	1
	00:59:00	1	1	0	1	1	1	1	0	0	0	0	0	0
3	01:22:00	0	1	0	0	1	0	1	0	0	0	1	1	0
4	01:42:00	1	1	1	1	0	1	0	0	1	0	0	1	1
	01:43:00	0	0	1	0	0	0	1	1	1	0	1	0	0
	01:48:00	0	0	0	0	1	1	1	0	0	1	1	0	1
	01:49:00	1	1	0	0	0	1	0	1	0	1	1	0	1

Figura 4.9: Exemplo da segmentação do um *log* base, ainda com os valores originais.

PUC-Rio - Certificação Digital Nº 1912721/CA

segmento	timestamp	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
1	00:00:00 00:27:00	0,00	0,90	1,00	0,10	1,00	0,90	0,90	0,00	0,00	1,00	0,90	0,90	0,90
2	00:31:00 00:34:00 00:39:00 00:43:00 00:51:00 00:57:00 00:59:00	0,43	0,50	0,27	0,33	0,17	0,27	0,37	0,37	0,77	0,40	0,27	0,70	0,33
3	01:22:00	0,00	0,27	0,00	0,00	0,27	0,00	0,27	0,00	0,00	0,00	0,27	0,27	0,00
4	01:42:00 01:43:00 01:48:00 01:49:00	0,40	0,40	0,20	0,03	0,03	0,43	0,20	0,53	0,20	0,40	0,57	0,03	0,43

Figura 4.10: Exemplo do *log* segmentado, já com seus fatores de abertura de válvula.

Definimos o tamanho do segmento como 30 minutos devido as informações presentes no manual de operação da unidade de coqueamento retardado, que definia como tamanho mínimo de uma fase de operação como 30 minutos. Na prática, segundo as informações do especialista de operação, algumas fases mudam de regime, sendo efetivamente duas ou mais fases distintas. Cada uma dessas sub fases podem ter uma duração menor do que 30 minutos. Como não temos as informações completas dessas sub fases mantivemos o tamanho do segmento em 30 minutos, conforme o manual teórico.

4.3.2 Agrupamentos de válvulas ou segmentos

Experimentamos dois formas de agrupar os dados. A primeira, mais direta, consiste em fazer um agrupamentos de válvulas, onde as variáveis são seus valores nos segmentos de tempo. Dessa forma elas serão agrupadas de acordo com a similaridade entre seus padrões de tempo de abertura. Uma propriedade dessa forma de agrupamento é que cada válvula só pertencerá a um único grupo final. Para o nosso domínio essa é uma limitação considerável, pois se esperamos encontrar grupos que representam etapas processo, deveria ser possível uma válvula pertencer a mais de um grupo. O outro problema é relacionado ao formato da base de dados, pois nesse caso temos muito mais variáveis do que observações, podendo levar ao problema da maldição da dimensionalidade (Köppen, 2000).

Experimentamos também agrupar os dados pelos segmentos, com as

PUC-Rio - Certificação Digital Nº 1912721/CA

válvulas sendo as variáveis. Agrupando dessa forma não temos mais a limitação de apenas uma válvula por grupo, também não temos mais a maldição da dimensionalidade. Mas, diferente de quando agrupamos os dados por válvulas, o resultado do agrupamento não é diretamente aplicável. Dessa forma é necessário um pós processamento, para extrair o conjunto de válvula que representa um grupo de segmentos de tempo.

Como vamos demonstrar nas seções seguintes, não é trivial a transformação de grupos de segmentos em grupos de válvulas. Nos experimentos o agrupamento por válvulas, mesmo com suas limitações, resultou em grupos melhores do que o agrupamento por segmentos. Por isso optamos por usar o agrupamento por válvulas para gerar os estados do processo.

4.3.3 Análise de qualidade

Os métodos de agrupamento devem ser capazes de formar grupos que aproximam estados conhecidos do processo. A primeira análise será feita de forma numérica, através de métricas, onde comparamos os resultados do agrupamento com grupos de referência. A segunda análise será através da ferramenta `PLANTSTATE`, a fim de entender como os grupos descobertos se relacionam com a unidade de coqueamento retardado.

A métrica utilizada, chamada de *score*, tem como objetivo medir se cada um dos grupos descobertos representam bem algum grupo de referência. E, como segundo objetivo, a métrica penaliza grupos descobertos que representem muitos grupos de referência. Com isso o conjunto ideal deve ser composto por grupos que representam bem um, e apenas um, grupo de referência.

O *score* de um resultado de agrupamento é a média aritmética de quatro sub métricas: RF (*reference fitness*), RP (*reference precision*), DF (*discovered fitness*) e DP (*discovered precision*).

O conceito de *fitness* de um grupo g é o quão bem esse grupo é representado por algum grupo de um conjunto grupos G. Já o *precision* de um grupo g é o quão bem esse grupo é capaz de representar apenas um grupo no conjunto de grupos G. Descrevemos como calcular o *fitness* e o *precision* de grupo através dos procedimentos CalcularFitnessDeGrupo e CalcularPrecisionDeGrupo descritos nos Algoritmos 5 e 6 respectivamente.

PUC-Rio - Certificação Digital Nº 1912721/CA

Algoritmo 5 Cálculo do *fitness* de um grupo em relação um conjunto de grupos G

Procedimento CalcularFitnessDeGrupo(g, G)
    $Q \leftarrow$ lista de valores, inicialmente vazia.
    para todo grupo g' em G **faça**
        $q \leftarrow$ quantidade de válvulas de g pertencentes a g'
        adicionar q a Q
    fim para
    $n \leftarrow$ quantidade de válvulas de g
    $m \leftarrow$ maior valor em Q
    retornar m/n
fim Procedimento

Algoritmo 6 Cálculo do *precision* de um grupo g em relação a um conjunto de grupos G.

Procedimento CalcularPrecisionDeGrupo(g, G)
    $Q \leftarrow$ lista de valores, inicialmente vazia.
    para todo grupo g' em G **faça**
        $q \leftarrow$ quantidade de válvulas de g pertencentes a g'
        adicionar q a Q
    fim para
    $c \leftarrow 0.5*$ quantidade de válvulas em G
    $M \leftarrow c$ maiores valores em Q
    remover maior valor de M
    $m \leftarrow$ média de M
    $n \leftarrow$ quantidade de válvulas de g
    retornar 1 - (m/n)
fim Procedimento

Dessa forma, calculamos RF e RP de um grupo de referência g_r em relação aos grupos descobertos G_d através dos procedimentos CalcularFitnessDeGrupo(g_r, G_d) e CalcularPrecisionDeGrupo(g_r, G_d). De forma similar, calculamos DF e DP de um grupo descoberto g_d em relação aos grupos de referência G_r através dos procedimentos CalcularFitnessDeGrupo(g_d, G_r) e CalcularPrecisionDeGrupo(g_d, G_r).

Testaremos as métricas contra dois grupos de referência. O primeiro grupo representa as fases da operação de coqueamento retardado segundo o manual de operação da unidade. O manual de operação é um documento geral que não aborda os detalhes de uma unidade específica. O especialista de

PUC-Rio - Certificação Digital Nº 1912721/CA

operação apontou que existem detalhes além do que estão descritos no manual de operação. O segundo grupo de referência são os alinhamentos. A definição dos alinhamentos é baseada em características do *layout* da planta, sendo assim menos dependente de interpretações.

Dessa forma, dado a quantidade de incógnitas ainda presentes na definição das fases da operação, os testes feitos usando as fases são apenas para efeitos de comparação.

4.3.4 Agrupamento por segmentos

O agrupamento por segmentos consiste em agrupar as linhas do *log* segmentado utilizando AHC. Para transformar agrupamento de segmentos resultante em um conjunto de válvulas criamos o Algoritmo 7. Esse algoritmo consiste em calcular a média do valor de cada válvula v nos segmentos S fornecidos. Consideramos que essa válvula v é um representante do conjunto S se sua média dos valores em S for igual ao seu menor valor no *log* de segmentos L_s, ou maior do que 0.5. O conjunto dos conjuntos de válvulas representantes de cada grupo de segmentos formam os estados do processo.

Algoritmo 7 Extrai o grupo de válvulas representantes de um conjunto S de segmentos.

```
1:  Procedimento ExtrairRepresentantesDosSegmentos(S, L_s)
2:      V ← lista de válvulas, inicialmente vazia.
3:      para todo válvula v faça
4:          c ← menor valor de v em L_s
5:          m ← média dos valores de v em S
6:          se m ≥ c OU m > 0.5 então
7:              adicionar v a V
8:          fim se
9:      fim para
10:     retornar V
11: fim Procedimento
```

Realizamos 24 rodadas de agrupamento, um para cada combinação de métrica de distância e *linkage*. Os *linkages* utilizados foram average, complete e single. As métricas de distância utilizadas foram *euclidean*, *manhattan*, *chebyshev*, *braycurtis*, *canberra*, *hamming*, *jaccard* e *matching*. A quantidade de grupos foi fixada em 14.

Para as métricas , *hamming*, *jaccard* e *matching*, que utilizam dados binários, convertemos os valores do *log* segmentado em 0 quando o valor original

PUC-Rio - Certificação Digital Nº 1912721/CA

estava abaixo de 0.5, e 1 caso contrário.

Nas Tabelas 4.1 e 4.2 temos os *scores* dos agrupamentos com todas as combinações de parâmetros. Em ambas as tabelas os parâmetros do agrupamento são os mesmos, a diferença dos *scores* se dá pela comparação feita com diferentes os grupos de referência. Na Tabela 4.1 temos valores de *score* menores, o que indica o agrupamento, no geral, representou melhor os alinhamentos do processo do que as fases do processo.

PUC-Rio - Certificação Digital Nº 1912721/CA

Tabela 4.1: *Scores* obtidos pelo agrupamento de segmentos com 14 grupos e comparando-os com o grupo de referência das fases do processo.

	Score	RF	RP	DF	DP
braycurtis_average	0.7721	0.7083	0.7429	0.9405	0.6968
jaccard_complete	0.7677	0.7917	0.7024	0.9405	0.6361
braycurtis_complete	0.7637	0.7167	0.7313	0.9226	0.6841
jaccard_average	0.7533	0.7583	0.7115	0.8810	0.6623
euclidean_average	0.7522	0.8000	0.6687	0.9405	0.5996
canberra_average	0.7427	0.7222	0.7325	0.8571	0.6587
braycurtis_single	0.7418	0.5694	0.8452	0.7381	0.8143
matching_complete	0.7414	0.7472	0.7036	0.8929	0.6218
hamming_complete	0.7414	0.7472	0.7036	0.8929	0.6218
manhattan_complete	0.7400	0.7639	0.6897	0.8929	0.6135
euclidean_complete	0.7394	0.8139	0.6675	0.8929	0.5833
chebyshev_single	0.7310	0.8333	0.6552	0.8869	0.5484
manhattan_average	0.7274	0.7972	0.6575	0.8929	0.5619
matching_average	0.7243	0.7889	0.6750	0.8214	0.6119
hamming_average	0.7243	0.7889	0.6750	0.8214	0.6119
manhattan_single	0.7243	0.7139	0.7464	0.7500	0.6869
jaccard_single	0.7222	0.5306	0.8075	0.7679	0.7829
euclidean_single	0.7214	0.6722	0.7079	0.8333	0.6722
chebyshev_average	0.7210	0.8000	0.6762	0.8036	0.6044
chebyshev_complete	0.7200	0.7056	0.6690	0.9286	0.5770
canberra_complete	0.7120	0.6472	0.6778	0.8929	0.6302
canberra_single	0.7008	0.5000	0.7786	0.7857	0.7389
hamming_single	0.6921	0.7139	0.6817	0.7321	0.6405
matching_single	0.6921	0.7139	0.6817	0.7321	0.6405

PUC-Rio - Certificação Digital Nº 1912721/CA

Tabela 4.2: *Scores* obtidos pelo agrupamento de segmentos com 14 grupos e comparando-os com o grupo de referência dos alinhamentos.

	Score	RF	RP	DF	DP
braycurtis_single	0.8069	0.6667	0.8386	0.8810	0.8413
braycurtis_average	0.8036	0.8333	0.7156	0.9762	0.6892
jaccard_complete	0.8009	0.9167	0.6786	0.9762	0.6323
jaccard_average	0.7943	0.8611	0.6944	0.9524	0.6693
braycurtis_complete	0.7877	0.8333	0.6839	0.9762	0.6574
jaccard_single	0.7798	0.5556	0.8452	0.8571	0.8611
manhattan_complete	0.7632	0.8981	0.6455	0.9286	0.5807
euclidean_average	0.7593	0.8333	0.6349	0.9762	0.5926
matching_average	0.7589	0.8426	0.6508	0.9286	0.6138
hamming_average	0.7589	0.8426	0.6508	0.9286	0.6138
euclidean_complete	0.7503	0.8981	0.6217	0.9286	0.5529
canberra_average	0.7503	0.7870	0.6693	0.9286	0.6164
manhattan_single	0.7500	0.7870	0.6944	0.8571	0.6614
euclidean_single	0.7464	0.7593	0.6627	0.9048	0.6587
canberra_complete	0.7444	0.8148	0.6376	0.9286	0.5966
chebyshev_complete	0.7431	0.8426	0.6429	0.9286	0.5582
hamming_complete	0.7421	0.8148	0.6429	0.9286	0.5820
matching_complete	0.7421	0.8148	0.6429	0.9286	0.5820
hamming_single	0.7388	0.7870	0.6680	0.8571	0.6429
matching_single	0.7388	0.7870	0.6680	0.8571	0.6429
canberra_single	0.7328	0.6852	0.7077	0.8571	0.6812
manhattan_average	0.7242	0.8426	0.6032	0.9286	0.5225
chebyshev_average	0.7226	0.7870	0.6455	0.8571	0.6005
chebyshev_single	0.7163	0.8704	0.5886	0.9048	0.5013

PUC-Rio - Certificação Digital Nº 1912721/CA

Selecionamos os resultados com os parâmetros de métrica de distância de *jaccard* e *complete linkage* para fazer uma inspeção mais detalhada. Na Figura 4.11 plotamos um *heatmap* que representa o quanto cada grupo referência está sendo representado pelos grupos descobertos. Onde 1.0 significa que o grupo de referência está completamente contido no grupo descoberto e 0 significa que não está nada contido. Com exceção do caso de alinhamentos que compartilham válvulas, o ideal seria termos apenas valores 0 e 1 nesse mapa.

No *heatmap* conseguimos identificar que os alinhamentos `estado ref 6`, `estado ref 15` e `estado ref 16` não foram bem representados por nenhum dos grupos descobertos. Também é possível identificar que a maioria dos grupos de referência são completamente representados por vários grupos descobertos. O que indica que não temos uma precisão na definição dos grupos.

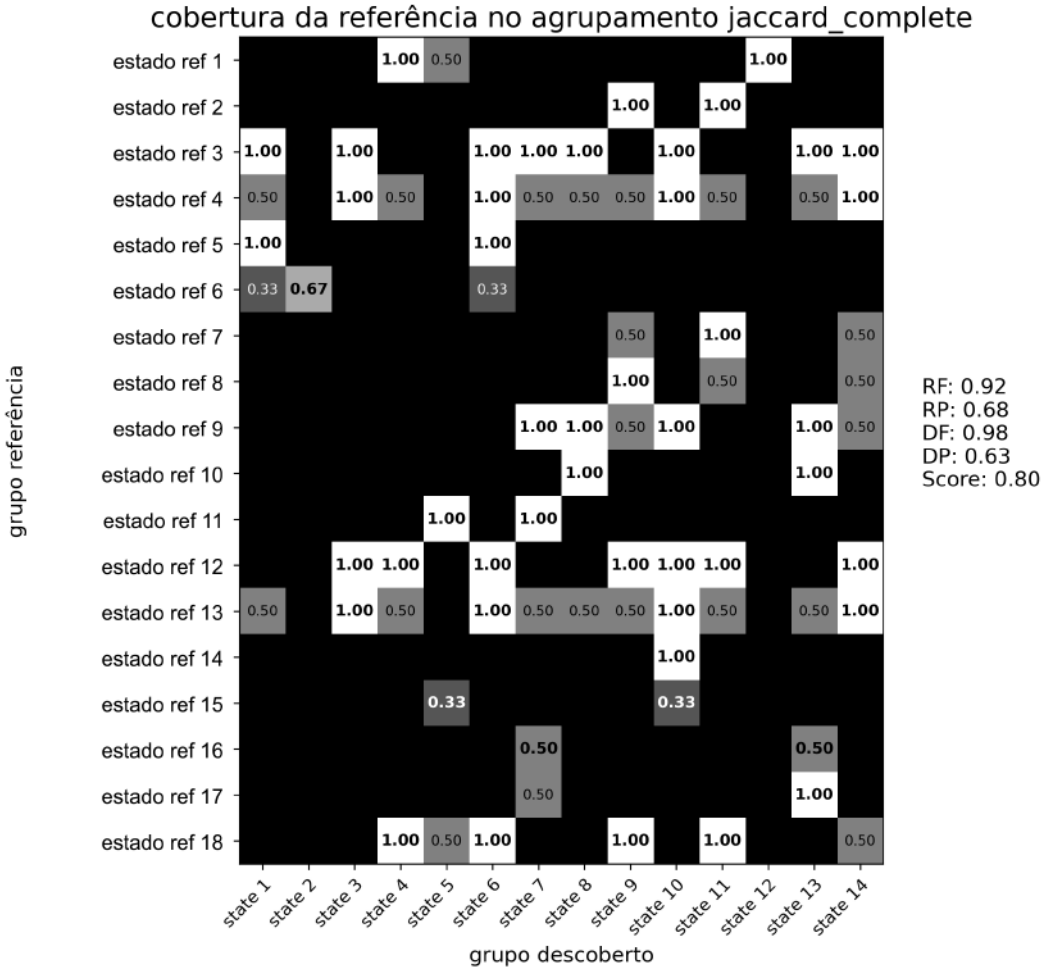


Figura 4.11: *Heatmap* do pertencimento dos grupos de referência nos grupos de descobertos para um resultado de agrupamento.

Para ter uma visão funcional dos grupos descobertos, inspecionamos o

PUC-Rio - Certificação Digital Nº 1912721/CA

resultado na ferramenta `PLANTSTATE`. Nela percebemos que maior parte dos grupos descobertos não possuem uma função clara definida na unidade de coqueamento. Na Figura 4.12 selecionamos um dos estados descobertos para inspecionar e repare que existem válvulas abertas em regiões muito distantes, aparentemente desconexas entre si. Não temos um fluxo na unidade e a maioria dos alinhamentos estão incompletos. Indícios de que o agrupamento de segmentos, da forma como foi feito, é pouco adequado para o problema.

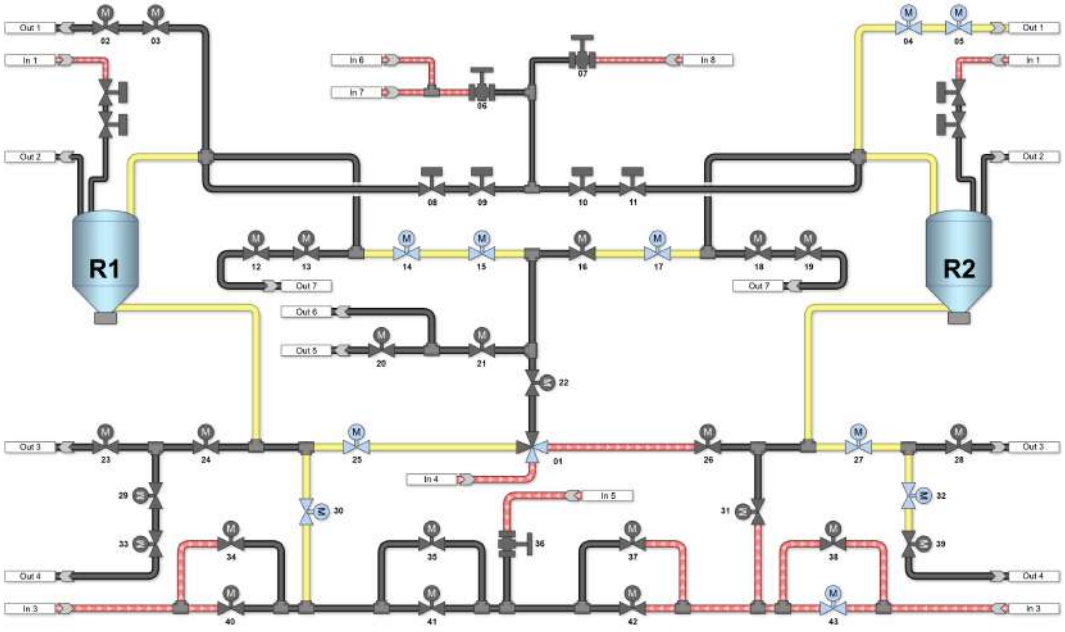


Figura 4.12: `PLANTSTATE`: estado da unidade segundo um dos resultados do agrupamento por segmentos.

PUC-Rio - Certificação Digital Nº 1912721/CA

4.3.5 Agrupamento por válvulas

O agrupamento por válvulas consiste em agrupar as colunas do *log* segmentado utilizando AHC. Diferente do agrupamento por segmentos, os grupos resultantes já são os estados do processo.

Como no agrupamento por segmentos, realizamos 24 rodadas de agrupamento, um para cada combinação de métrica de distância e *linkage*. Os *linkages* utilizados foram *average*, *complete* e *single*. As métricas de distância utilizadas foram *euclidean*, *manhattan*, *chebyshev*, *braycurtis*, *canberra*, *hamming*, *jaccard* e *matching*. A quantidade de grupos foi fixada em 14, baseado na otimização desse parâmetro, como visto na Figura 4.13. Para as métricas com base binária fizemos a mesma conversão do que a descrita no agrupamento por segmentos.

Nas Tabelas 4.3 e 4.4 temos os *scores* dos agrupamentos com todas as combinações de de parâmetros. Em ambas as tabelas os parâmetros do

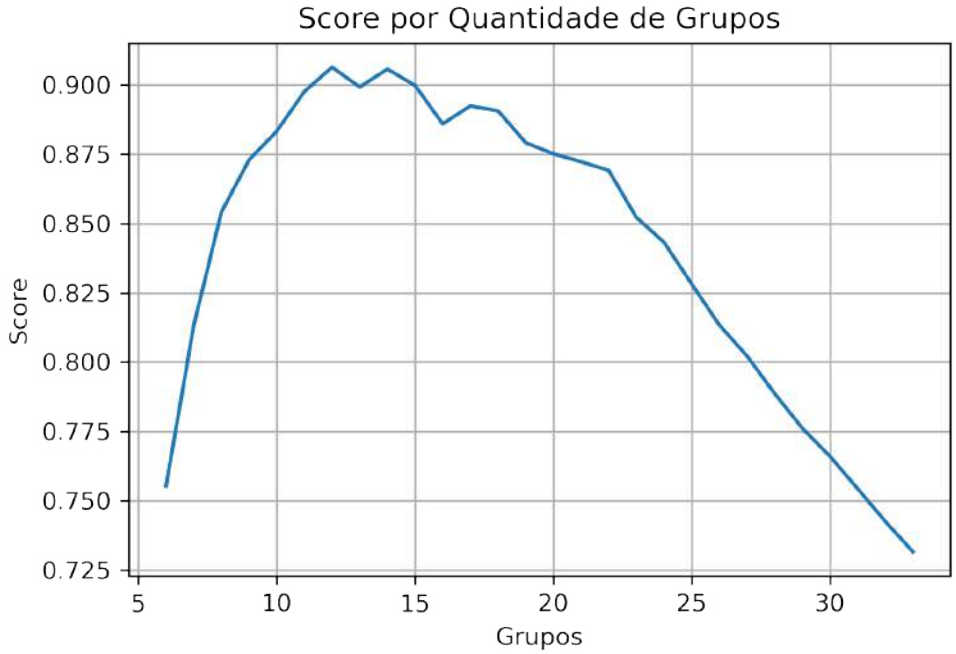


Figura 4.13: Otimização da quantidade de grupos do agrupamento tendo como objetivo *scores* altos.

agrupamento são os mesmos, a diferença dos *scores* se dá pela comparação feita com diferentes grupos de referência. Na Tabela 4.4 temos valores de *score* maiores do que os da Tabela 4.3, indicando que o agrupamento representou melhor os alinhamentos do processo que as fases do processo.

PUC-Rio - Certificação Digital Nº 1912721/CA

Tabela 4.3: *Scores* obtidos pelo agrupamento de válvulas com 14 grupos e comparando-os com o grupo de referência das fases do processo.

	Score	RF	RP	DF	DP
hamming_complete	0.796032	0.669444	0.952778	0.660714	0.901190
matching_complete	0.796032	0.669444	0.952778	0.660714	0.901190
canberra_complete	0.796032	0.669444	0.952778	0.660714	0.901190
manhattan_complete	0.796032	0.669444	0.952778	0.660714	0.901190
euclidean_complete	0.796032	0.669444	0.952778	0.660714	0.901190
braycurtis_complete	0.794444	0.708333	0.958333	0.613095	0.898016
jaccard_complete	0.787302	0.666667	0.952381	0.630952	0.899206
jaccard_average	0.777778	0.650000	0.950000	0.613095	0.898016
braycurtis_average	0.775397	0.708333	0.958333	0.541667	0.893254
matching_average	0.771429	0.627778	0.946825	0.613095	0.898016
manhattan_single	0.771429	0.627778	0.946825	0.613095	0.898016
manhattan_average	0.771429	0.627778	0.946825	0.613095	0.898016
hamming_single	0.771429	0.627778	0.946825	0.613095	0.898016
hamming_average	0.771429	0.627778	0.946825	0.613095	0.898016
euclidean_single	0.771429	0.627778	0.946825	0.613095	0.898016
euclidean_average	0.771429	0.627778	0.946825	0.613095	0.898016
canberra_single	0.771429	0.627778	0.946825	0.613095	0.898016
canberra_average	0.771429	0.627778	0.946825	0.613095	0.898016
matching_single	0.771429	0.627778	0.946825	0.613095	0.898016
jaccard_single	0.769841	0.638889	0.948413	0.595238	0.896825
braycurtis_single	0.763492	0.666667	0.952381	0.541667	0.893254
chebyshev_complete	0.754762	0.638889	0.948413	0.529762	0.901984
chebyshev_average	0.741567	0.583333	0.940476	0.547619	0.894841
chebyshev_single	0.737302	0.669444	0.952778	0.428571	0.898413

PUC-Rio - Certificação Digital Nº 1912721/CA

Tabela 4.4: *Scores* obtidos pelo agrupamento de válvulas com 14 grupos e comparando-os com o grupo de referência dos alinhamentos.

	Score	RF	RP	DF	DP
matching_complete	0.908730	0.824074	0.974868	0.880952	0.955026
canberra_complete	0.908730	0.824074	0.974868	0.880952	0.955026
manhattan_complete	0.908730	0.824074	0.974868	0.880952	0.955026
hamming_complete	0.908730	0.824074	0.974868	0.880952	0.955026
euclidean_complete	0.908730	0.824074	0.974868	0.880952	0.955026
hamming_average	0.897487	0.796296	0.970899	0.869048	0.953704
euclidean_single	0.897487	0.796296	0.970899	0.869048	0.953704
matching_average	0.897487	0.796296	0.970899	0.869048	0.953704
manhattan_single	0.897487	0.796296	0.970899	0.869048	0.953704
manhattan_average	0.897487	0.796296	0.970899	0.869048	0.953704
hamming_single	0.897487	0.796296	0.970899	0.869048	0.953704
matching_single	0.897487	0.796296	0.970899	0.869048	0.953704
euclidean_average	0.897487	0.796296	0.970899	0.869048	0.953704
canberra_single	0.897487	0.796296	0.970899	0.869048	0.953704
canberra_average	0.897487	0.796296	0.970899	0.869048	0.953704
jaccard_complete	0.878968	0.824074	0.974868	0.773810	0.943122
jaccard_average	0.871032	0.796296	0.970899	0.773810	0.943122
braycurtis_complete	0.867063	0.851852	0.978836	0.702381	0.935185
jaccard_single	0.855820	0.777778	0.968254	0.738095	0.939153
braycurtis_single	0.849206	0.824074	0.974868	0.666667	0.931217
braycurtis_average	0.849206	0.824074	0.974868	0.666667	0.931217
chebyshev_average	0.831349	0.703704	0.957672	0.726190	0.937831
chebyshev_complete	0.823743	0.703704	0.957672	0.690476	0.943122
chebyshev_single	0.809193	0.712963	0.958995	0.630952	0.933862

PUC-Rio - Certificação Digital Nº 1912721/CA

Como com os resultados do agrupamento de segmentos, vamos fazer uma inspeção do *heatmap* que indica a coberta dos estados de referência. Selecionamos dois conjuntos de parâmetros, ambos com *complete linkage*, um com distância *euclidean* e outro distância de *hamming*. As métricas foram selecionadas devido ao seu alto *score* nas Tabelas 4.3 e 4.4.

A relação mais evidente entre as 4.14 e 4.15 é mesmo que suas métricas utilizem um *log* segmentado diferente - binarizado e não binarizado, seus valores são os mesmos. Essa similaridade explica os valores de *score* idênticos nos primeiros nas Tabelas 4.3 e 4.4. O mesmo ocorre entre os resultados nas Figuras 4.16 e 4.17.

Uma vantagem em relação ao agrupamento por segmentos é que os grupos descobertos são mais específicos, por isso seu *score* mais alto. Mesmo assim, se considerarmos o grupo de referência como as fases do processo - Figuras 4.14 e 4.15, ainda não temos representação completa para a maioria dos grupos de referência.

Já quando usamos como referência de comparação os alinhamentos do processo - Figuras 4.16 e 4.17 temos valor de *score* acima de 0.9 e uma representação mais completa dos grupos de referência, indicando que o método é mais adequado para encontrar os alinhamentos da unidade.

PUC-Rio - Certificação Digital Nº 1912721/CA

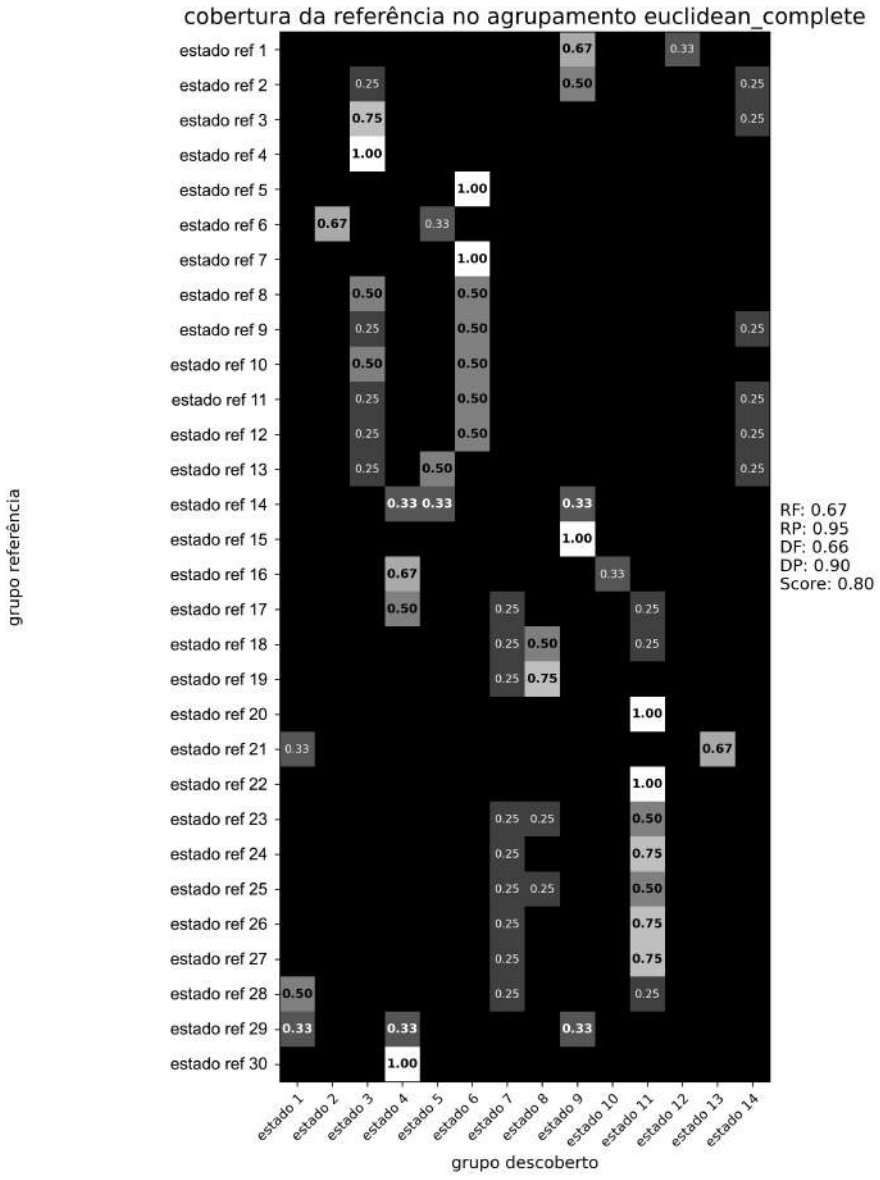


Figura 4.14: *Heatmap* do pertencimento dos grupos de referência das fases do processo nos grupos de descobertos para um resultado de agrupamento.

PUC-Rio - Certificação Digital Nº 1912721/CA

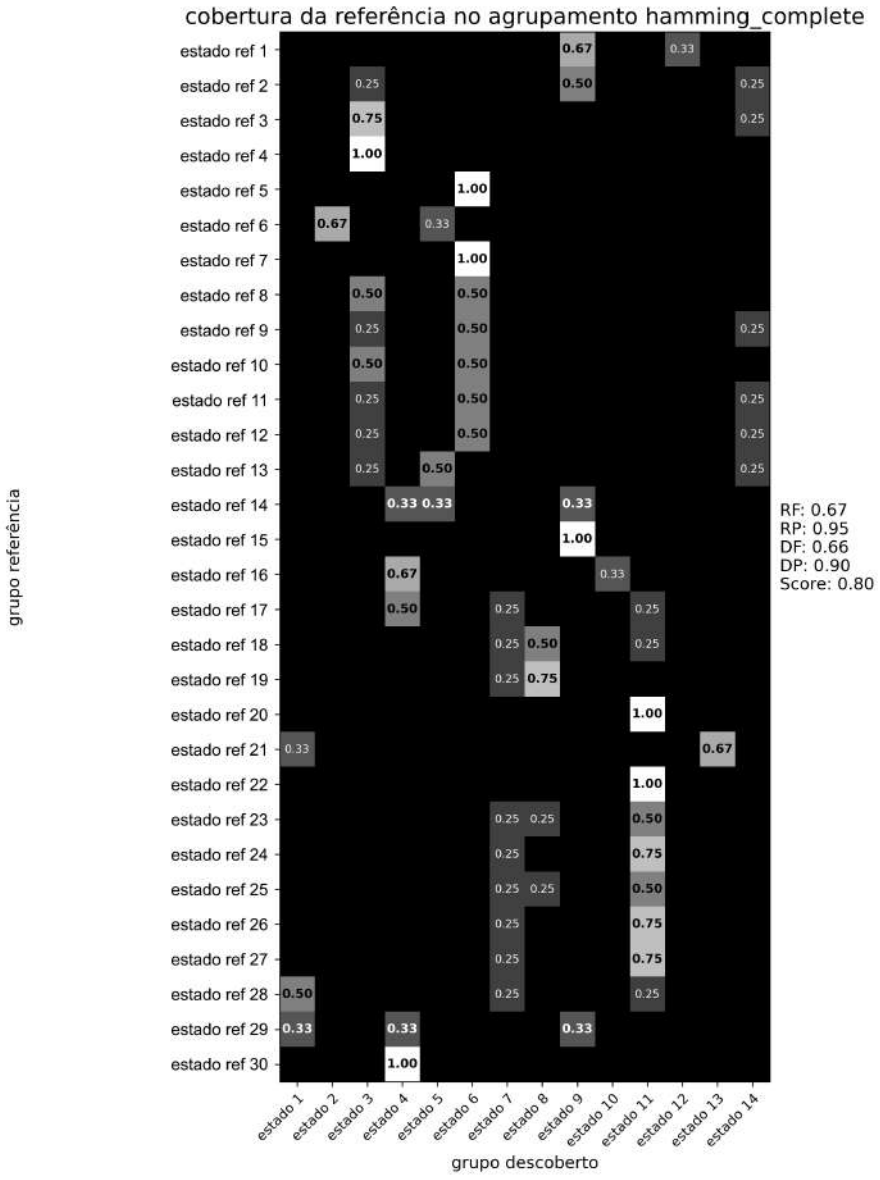


Figura 4.15: *Heatmap* do pertencimento dos grupos de referência das fases do processo nos grupos de descobertos para um resultado de agrupamento.

PUC-Rio - Certificação Digital Nº 1912721/CA

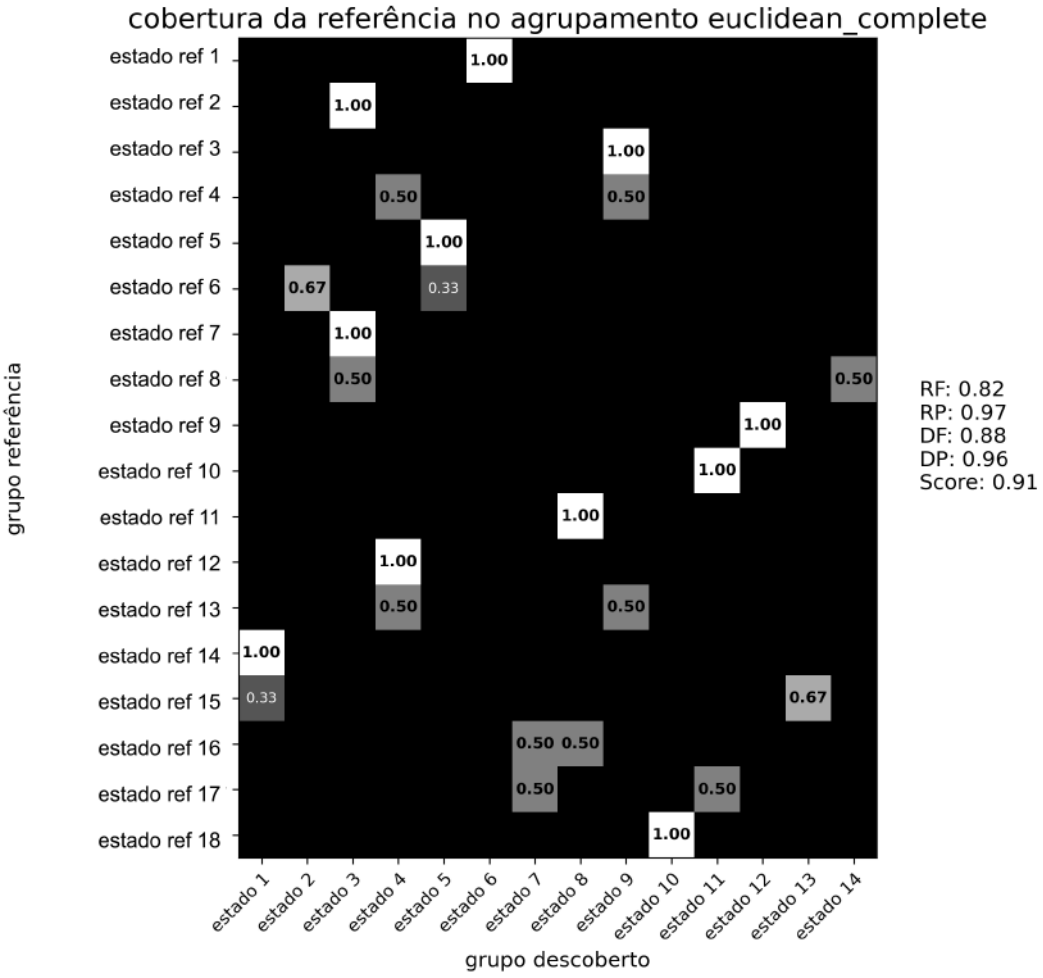


Figura 4.16: *Heatmap* do pertencimento dos grupos de referência dos alimentos da unidade nos grupos de descobertos para um resultado de agrupamento.

PUC-Rio - Certificação Digital Nº 1912721/CA

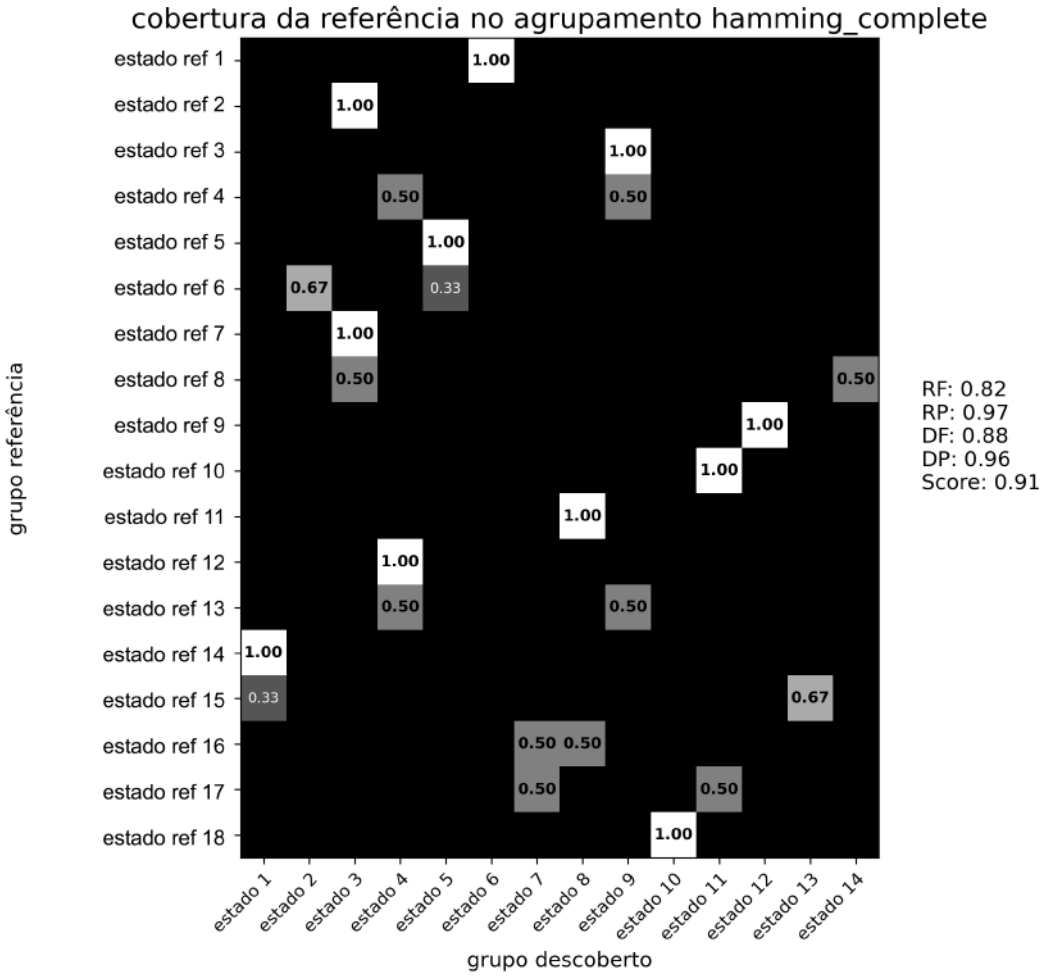


Figura 4.17: *Heatmap* do pertencimento dos grupos de referência dos alimentos da unidade nos grupos de descobertos para um resultado de agrupamento.

PUC-Rio - Certificação Digital Nº 1912721/CA

Por fim, para uma visão funcional dos grupos descobertos, vamos inspeciona-los na ferramenta `PLANTSTATE`. Nela percebemos que a maioria dos grupos possuem uma função bem definida. As Figuras 4.18 e 4.19 temos dois grupos distintos, onde um representa perfeitamente o alinhamento `fundo R2 - Out 3` e o outro `fundo R1 - Out 3`. Já as Figuras 4.20 e 4.21 representam dois alinhamentos em cada uma. Isso indica que esses alinhamentos provavelmente se manifestam em conjunto no *log* base. Os dois alinhamentos formam uma fase completa do processo, identificado pelos segmentos de tubulação em azul, indicando que há um fluxo de fluídos na planta.

Essa identificação de grupos distintos e com uma simetria indica que o agrupamento por válvulas com distância *euclidean* e *complete linkage* resulta em uma boa solução para o problema. Existirem dois alinhamentos distintos em um único grupo não é um problema, pois como os alinhamentos são claros esse grupo seria identificado e nomeado pelo usuário na ferramenta `PLANTSTATE`.

Dessa forma respondemos a **PP1:** Dado um *log* de eventos da planta, é possível descobrir estados relacionados a operação da unidade? Sim, demonstramos que com a aplicação de técnicas de agrupamento é possível descobrir estados que representam alinhamentos e fases do processo de coqueamento retardado.

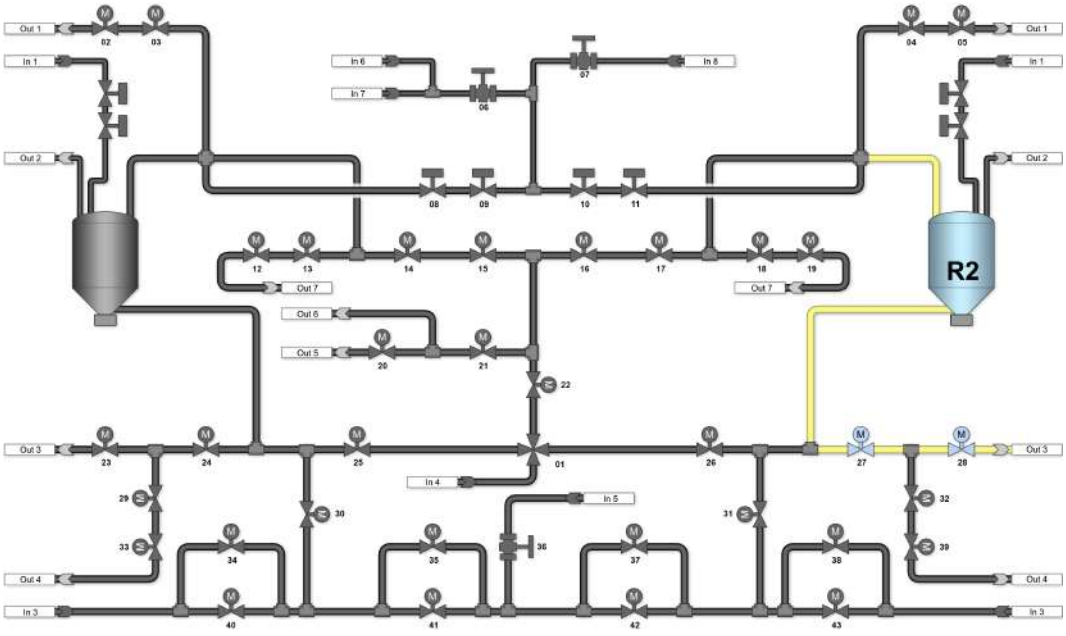


Figura 4.18: `PLANTSTATE`: estado da unidade segundo um dos grupos do agrupamento por válvulas.

PUC-Rio - Certificação Digital Nº 1912721/CA

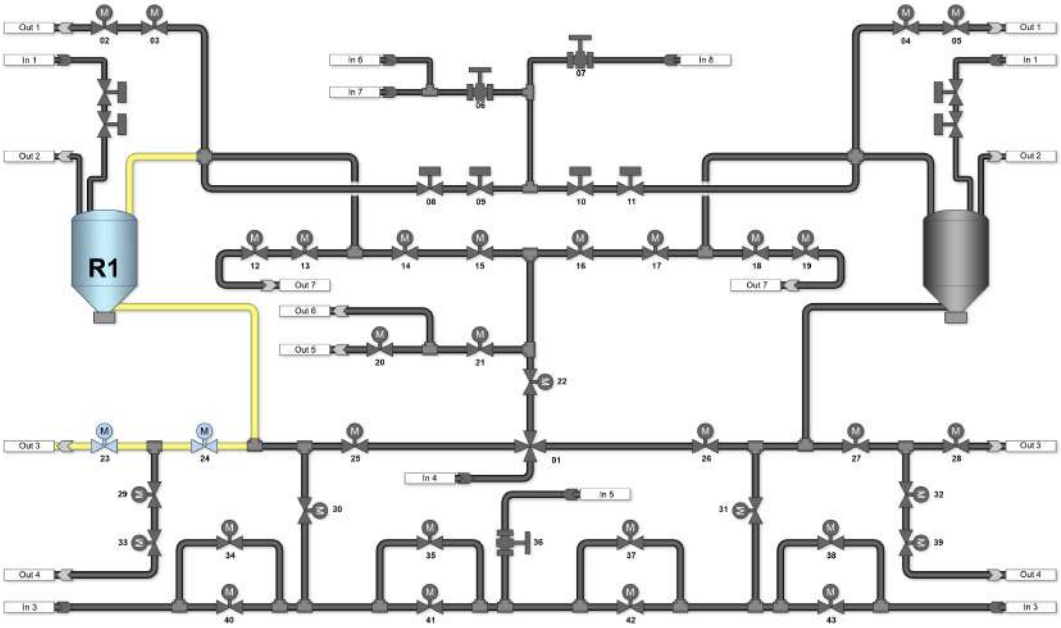


Figura 4.19: `PLANTSTATE`: estado da unidade segundo um dos grupos do agrupamento por válvulas.

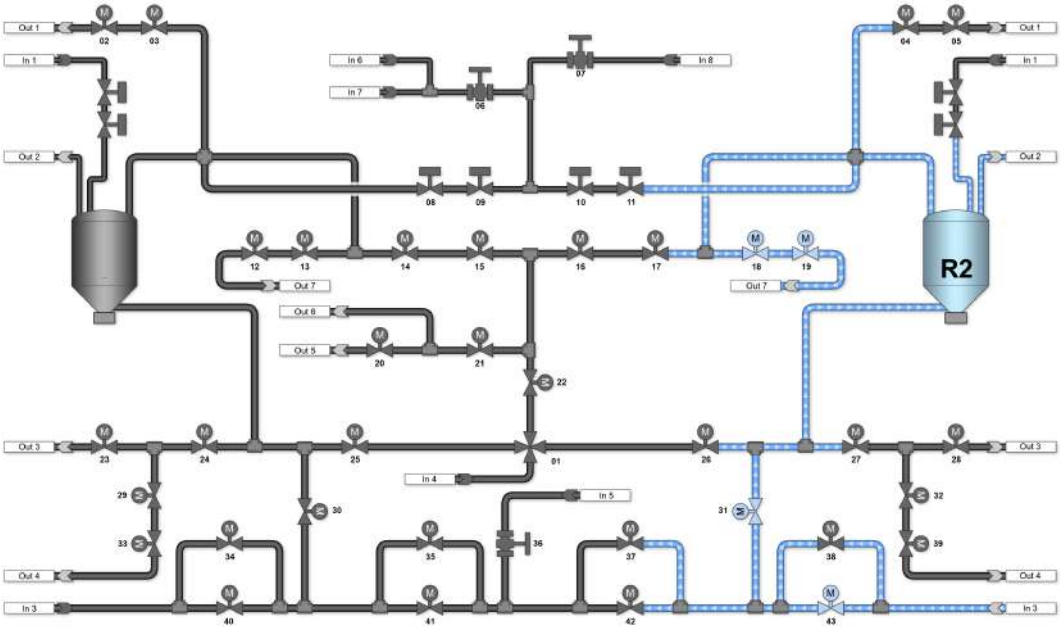


Figura 4.20: `PLANTSTATE`: estado da unidade segundo um dos grupos do agrupamento por válvulas.

PUC-Rio - Certificação Digital Nº 1912721/CA

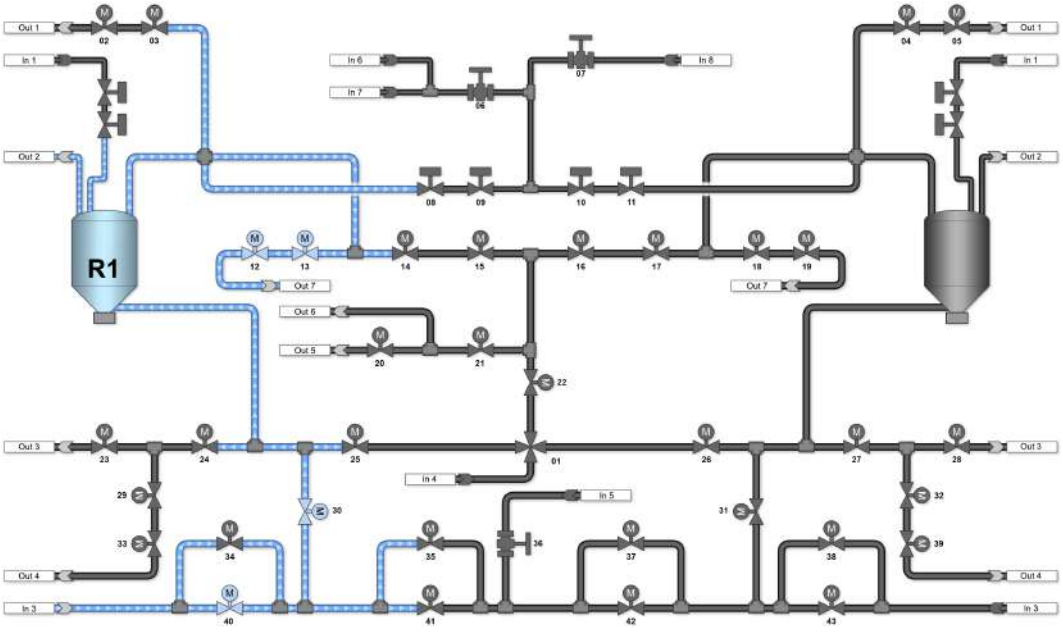


Figura 4.21: `PLANTSTATE`: estado da unidade segundo um dos grupos do agrupamento por válvulas.

PUC-Rio - Certificação Digital Nº 1912721/CA

4.4 Log de Estados

Com um conjunto de estados do processo definido podemos minerar a relação entre esses estados no *log* base. Os estados do processo podem ser encontrados utilizando os métodos descritos na Seção 4.3, por outros métodos de agrupamento, construídos na ferramenta `PLANTSTATE` ou um mistura desses recursos. O *log* base descreve um registro de válvulas e por isso vamos descrever como fazer a identificação dos estados do processo no *log* base e discutir o efeito dos parâmetros envolvidos.

4.4.1 Algoritmo

Cada estado do conjunto de estados do processo é definido por quais válvulas devem estar abertas para que aquele estado seja considerado ativo. Para encontrar quais estados estão ativos em cada momento do *log* base, iremos comparar, para cada estado do processo, qual a proporção de suas válvulas que estão abertas naquele momento do *log*. O estado será considerado ativo se essa proporção ultrapassar um valor definido. Ao final do processo, registrando quais estados estão ativos em cada momento do *log* base, teremos um *log* de estados.

Para facilitar o entendimento dessa relação entre a representação dos estados do processo e os dados do *log*, geramos a Figura 4.22. Através dela podemos ver que podemos representar ambos como vetores, o que facilita a definição de critérios que calculem a relação entre os dois, conforme vamos descrever no restante desta seção.

Estados do processo

Estados	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
E_1	0	1	1	0	1	1	1	0	0	1	1	1	1
E_2	0	0	1	1	1	0	0	0	0	1	0	0	0
E_3	0	1	0	0	0	0	1	1	1	0	0	1	1
E_4	0	1	0	1	0	1	1	0	0	0	0	0	1
E_5	1	1	0	1	1	0	0	0	1	1	0	1	0

***Log* Base**

timestamp	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
00:00:00	0	1	1	0	1	1	1	0	0	1	1	1	1
00:27:00	0	0	1	1	1	0	0	0	0	1	0	0	0
00:31:00	0	1	0	0	0	0	1	1	1	0	0	1	1
00:34:00	0	1	0	1	0	1	1	0	0	0	0	0	1
00:39:00	1	1	0	1	1	0	0	0	1	1	0	1	0
00:43:00	0	0	1	0	0	0	0	0	1	1	0	1	0

Figura 4.22: Estados do processo e *log* base representados como vetores.

PUC-Rio - Certificação Digital Nº 1912721/CA

Seja E_x o vetor que representa o estado x do conjunto de estados do processo E. Cada índice deste vetor representa uma válvula, e seus valores representam se aquela válvula deve estar aberta naquele estado (1) ou se tanto faz (0).

Seja ϵ o vetor que representa o estado atual de uma entrada do *log*. Assim como no vetor E_x, cada índice representa uma válvula. Mas seus valores representam se uma válvula está aberta (1) ou fechada (0).

Cada estado E_x possui um *score* S_x em relação ao estado atual ϵ, dado por:

$$
\begin{aligned}
Y &= E_x \cdot E_x \\
X &= E_x \cdot \epsilon \\
S_x &= \begin{cases} 0 & Y = 0 \\ X/Y & \text{c.c.} \end{cases}
\end{aligned}
\tag{4-1}
$$

Para efeitos de clareza, representamos na Figura 4.23 um exemplo do que seria o E_x e o ϵ em relação aos estados do processo e ao *log* base respectivamente.

Estados do processo

Estados	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
E1	0	1	1	0	1	1	1	0	0	1	1	1	1
E2	0	0	1	1	1	0	0	0	0	1	0	0	0
E3	0	1	0	0	0	0	1	1	1	0	0	1	1
E4	0	1	0	1	0	1	1	0	0	0	0	0	1
E5	1	1	0	1	1	0	0	0	1	1	0	1	0

E_x

***Log* Base**

timestamp	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
00:00:00	0	1	1	0	1	1	1	0	0	1	1	1	1
00:27:00	0	0	1	1	1	0	0	0	0	1	0	0	0
00:31:00	0	1	0	0	0	0	1	1	1	0	0	1	1
00:34:00	0	1	0	1	0	1	1	0	0	0	0	0	1
00:39:00	1	1	0	1	1	0	0	0	1	1	0	1	0
00:43:00	0	0	1	0	0	0	0	0	1	1	0	1	0

ϵ

Figura 4.23: Estados do processo e *log* base representados como vetores, destacando um exemplo de E_x e ϵ.

Então se $S_x \geq AH$ o estado E_x é considerado ativo. Onde AH é um parâmetro definido arbitrariamente entre 0 e 1.

Adotamos também um segundo critério, menos rigoroso, para decisão da ativação de um estado: seja S o conjunto dos valores de S_x, o valor SM é igual ao menor valor entre os AQ maiores valores de S. Um estado E_x é

PUC-Rio - Certificação Digital Nº 1912721/CA

considerado ativo se $S_x \geq SM$ e $S_x \geq AS$. Onde AQ e AS são parâmetros definidos arbitrariamente com AQ maior ou igual a 0 e AS entre 0 e 1.

4.4.2 Ajuste de parâmetros

Os parâmetros AH, AS e AQ controlam a quantidade de estados ativos a cada entrada do *log*. AH representa um valor mínimo de *score* para que um estado seja considerado ativo em um momento do *log*.

Para *logs* que representam processos com muitas variações no regime de operação, ruídos e imprecisões temos um critério mais flexível. O parâmetro AQ determina o mínimo de estados que serão considerados ativos por entrada do *log* desde que seus *scores* sejam maiores do que AQ.

O valor AH deve ser ajustado para o valor mínimo de *score* para que um estado seja considerado ativo. Caso o *score* de um estado não atenda o valor mínimo de AH, mas ainda assim seja o melhor que temos no momento, podemos decidir aproveita-lo caso seu *score* seja pelo menos AS. O parâmetro AQ controla a quantidade de estados com o *score* abaixo de AH devem ser considerados.

Realizamos experimentos em cima do *log* base com diferentes configurações de AH, AS e AQ que expressam a varição dos resultados em função a varição dos parâmetros.

O primeiro conjunto de parâmetros foi definido para ser rigoroso, apenas os estados que se manifestarem por completo serão considerados ativos.
Conjunto de parâmetros 1:

$$\begin{aligned} AH &= 1.0 \\ AS &= 1.0 \\ AQ &= 0 \end{aligned} \tag{4-2}$$

O segundo conjunto de parâmetros é um pouco mais permissivo do que o primeiro, porém ainda não utiliza o relaxamento proporcionado pelos parâmetros AQ e AS, que permite *scores* abaixo de AH.
Conjunto de parâmetros 2:

$$\begin{aligned} AH &= 0.8 \\ AS &= 1.0 \\ AQ &= 0 \end{aligned} \tag{4-3}$$

Dessa vez vamos flexibilizar a seleção. Em cada passo, caso nenhum

PUC-Rio - Certificação Digital Nº 1912721/CA

estado tenha um *score* acima do valor de AH, vamos considerar como ativo os AQ estados de maior *score* caso seus *score* sejam maior do que AS. Ou seja, caso o valor de AH não seja atendido vamos permitir alguns estados desde que atendam pelo menos um *score* mínimo de AS.

Conjunto de parâmetros 3:

$$\begin{aligned} AH &= 0.8 \\ AS &= 0.5 \\ AQ &= 1 \end{aligned} \tag{4-4}$$

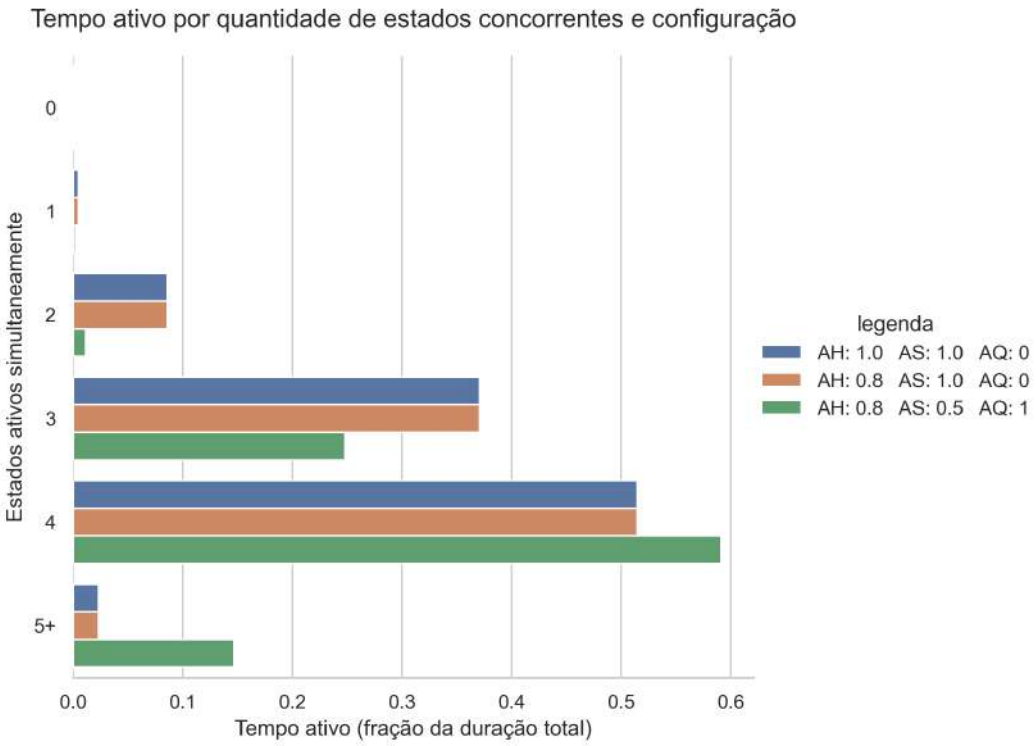


Figura 4.24: Tempo ativo por quantidade de estados concorrentes e por conjunto de parâmetros.

Na Figura 4.24 identificamos que apenas quando relaxamos os parâmetros AQ e AS tivemos alterações na identificação dos estados ativos. Esse relaxamento aumentou a proporção de tempo que 4 ou mais estados ficaram ativos simultaneamente durante o *log*.

Na Figura 4.25 percebemos que apenas quando relaxamos os parâmetros AQe AS tivemos alterações na identificação dos estados ativos. Esse relaxamento aumentou o tempo que a maioria dos estados foram identificados como ativos no *log*. Um destaque em especial vai para o estado 11, que mais do que duplicou seu tempo de atividade. Esse comportamento pode indicar que o es-

PUC-Rio - Certificação Digital Nº 1912721/CA

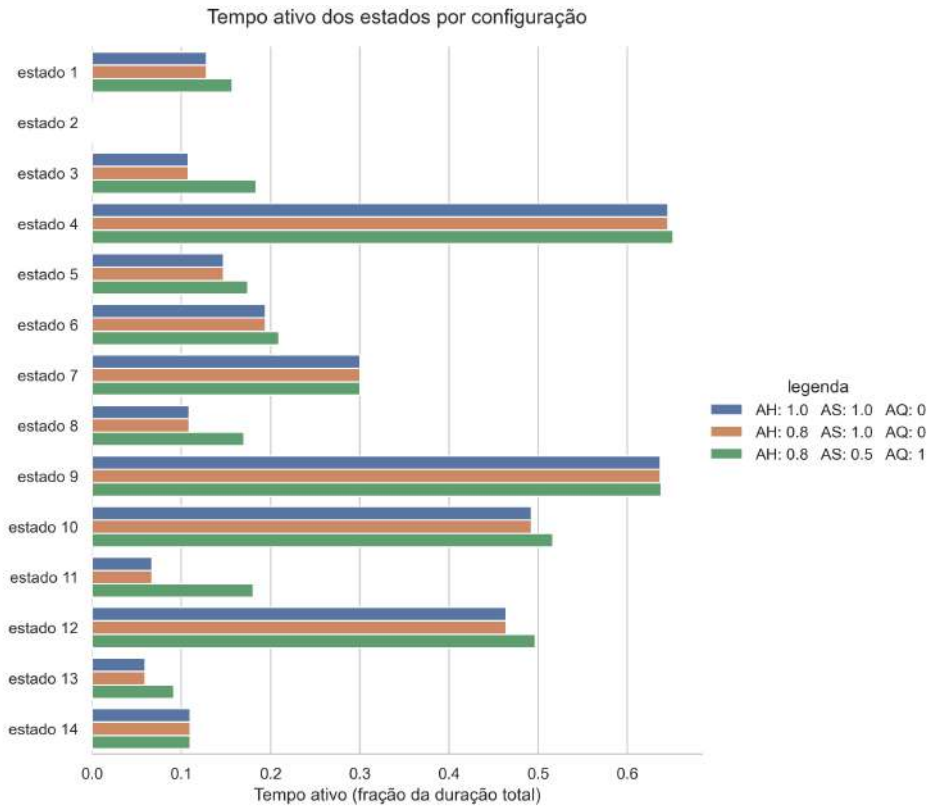


Figura 4.25: Tempo ativo de cada estado por conjunto de parâmetros.

tado 11 precise ser revisado, pois pode conter válvulas que deveriam pertencer a estados separados.

Ao investigar o estado 11 usando o `PLANTSTATE`, percebemos, na Figura 4.26, que suas válvulas estão posicionadas em locais opostos da planta de operação. No nosso cenário esse fato indica que as válvulas deveriam estar em estados separados. É possível que melhorias no método de agrupamento resolvesse essa questão automaticamente. Caso não resolva, o correto para o nosso cenário seria separar o estado 11 em dois grupos. Para este documento não iremos fazer essa separação e vamos prosseguir com os estados obtidos automaticamente. Finalmente, para a operação de coqueamento em questão, uma média de 4 estados ativos simultaneamente é o suficiente para o funcionamento do sistema de coqueamento. O relaxamento de AQ e AS não parece necessário.

Ao gerar o *log* de estados respondemos a pergunta **PP2:** Dados os estados do processo e um *timestamp* do *log* de eventos da planta, é possível identificar quais estados estão ativos no momento? Sim, com *log* base, que mantém configuração das válvulas até o momento, e uma métrica de distância entre a configuração das válvulas no *timestamp* e cada um dos estados do processo.

PUC-Rio - Certificação Digital Nº 1912721/CA

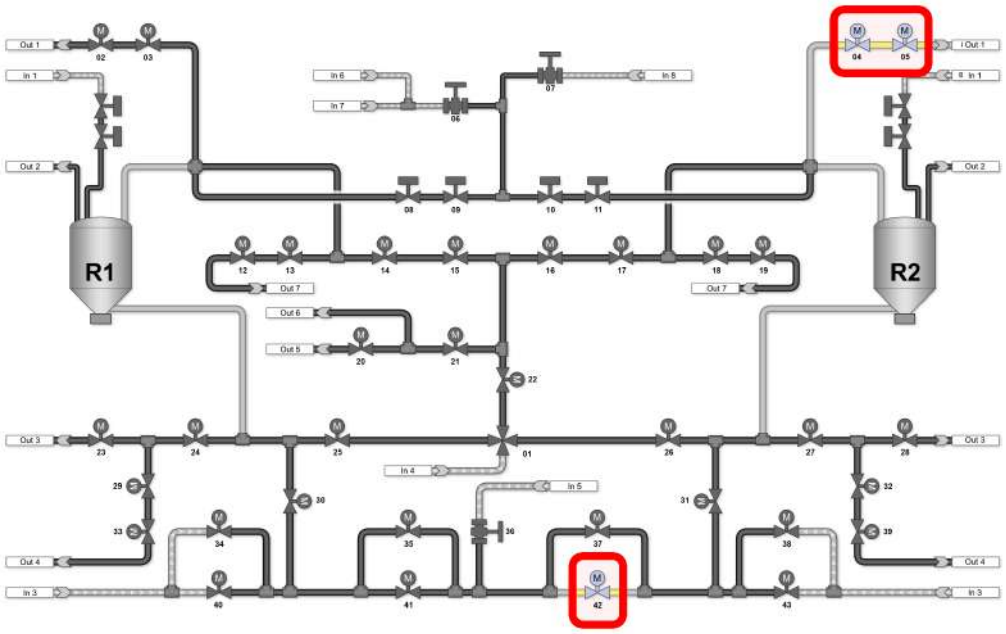


Figura 4.26: `PLANTSTATE` com destaque para o posicionamento das válvulas do estado 11.

PUC-Rio - Certificação Digital Nº 1912721/CA

5
Minerando o processo dos estados

Neste capítulo vamos descrever duas formas de prosseguir com a mineração do processo. No primeiro método vamos transformar o *log* de estados em um *log* de casos, que nos permite utilizar as ferramentas disponíveis de *process mining*. No segundo vamos pular a etapa de identificação dos casos e minerar a estrutura do processo. Em ambos os métodos vamos discutir a influência dos parâmetros que definimos e apresentar resultados de experimentos para diferentes configurações desses parâmetros.

Cada entrada do *log* de estados contém a informação do *timestamp* e quais estados estão ativos naquele *timestamp*. Com essa informação vamos encontrar os momentos que cada um dos estados foi ativado e por quanto tempo ficaram ativados. Em ambos os métodos vamos partir deste principio.

PUC-Rio - Certificação Digital Nº 1912721/CA

5.1
Separação por casos

Como já citado no Capítulo 2, na operação de coqueamento retardado, quando um dos reatores encerra seu ciclo de enchimento o outro reator inicia um novo ciclo de enchimento. O equipamento que redireciona o fluxo entre os reatores, é o *switch*. Vamos utilizar a atividade do *switch* no *log* para delimitar os casos do processo, onde uma mudança no *switch* marca o fim de um caso e o início de um novo.

Nas Figuras 5.1 e 5.2 temos os *switches* em duas posições diferentes, representando o enchimento de cada um dos reatores. Identificamos os estados do processo que contem as válvulas que representam essas posições e os definiremos como os estados que determinam a mudança de caso (*caseid*).

Com um *log* de estados L_e, estados do processo E, e o conjunto de estados C que determinam a mudança de *caseid* podemos definir o Algoritmo 8, que irá construir o *log* de casos.

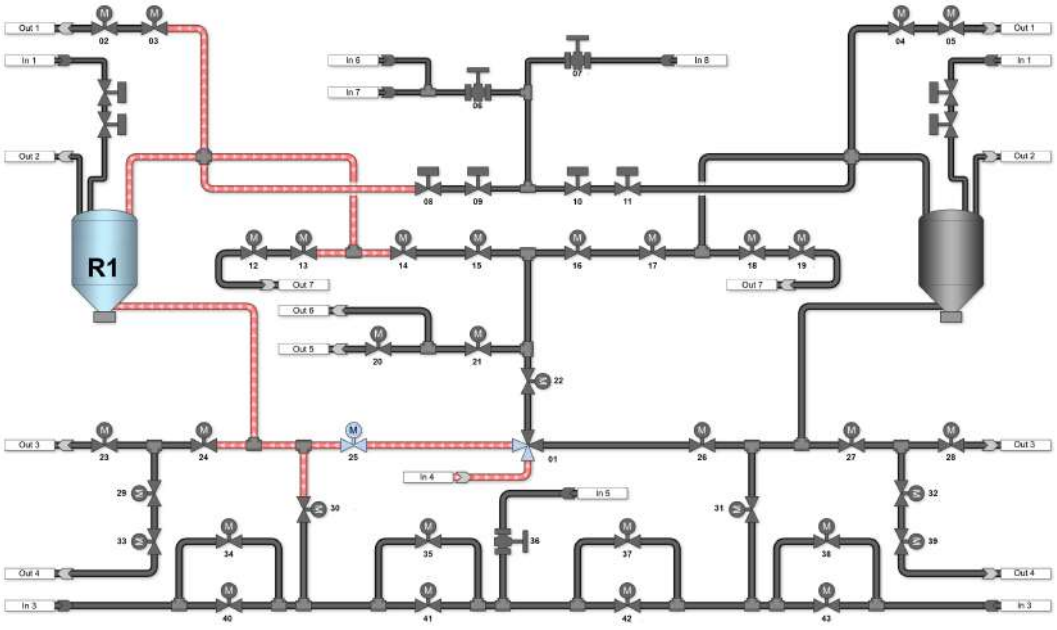


Figura 5.1: Switch posicionado para o enchimento do reator R1.

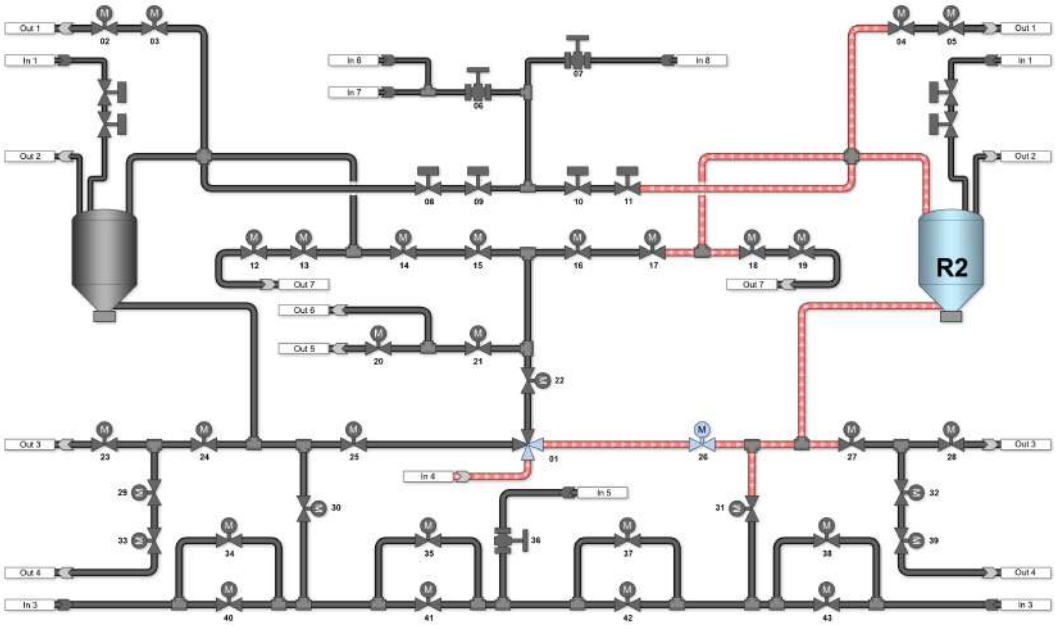


Figura 5.2: Switch posicionado para o enchimento do reator R2.

PUC-Rio - Certificação Digital Nº 1912721/CA

Algoritmo 8 Transformação do *log* de estados em *log* de casos

1: $L_c \leftarrow$ *log* de casos, inicialmente vazio
2: $c \leftarrow 0$
3: $\mathcal{A} \leftarrow$ registro de estados ativos na entrada anterior, inicialmente vazio
4: **para todo** entrada ℓ em L_e **faça**
5: $t \leftarrow$ *timestamp* da entrada ℓ
6: $\mathcal{E} \leftarrow$ estados ativos na entrada ℓ
7: **para todo** $\epsilon \in C$ **faça**
8: **se** $\epsilon \in \mathcal{E}$ **então**
9: incremente c
10: **fim se**
11: **fim para**
12: **para todo** $\epsilon \in E$ **faça**
13: **se** $\epsilon \in \mathcal{E}$ & $\epsilon \notin \mathcal{A}$ **então**
14: Registre ϵ em $\mathcal{A}$ com $\epsilon_i = t$ e $\epsilon_c = c$
15: **senão se** $\epsilon \notin \mathcal{E}$ & $\epsilon \in \mathcal{A}$ **então**
16: Registre uma nova entrada em L_c com os dados: *caseid* $= \epsilon_c$; *eventid* $= \epsilon$; *timestamp_start* $= \epsilon_i$; *timestamp_finish* $= t$
17: Remova ϵ de $\mathcal{A}$
18: **fim se**
19: **fim para**
20: **fim para**

Executamos o Algoritmo 8 com $C =$ [estado 10, estado 12] e exportamos o *log* de casos retornado em CSV com as colunas de *caseid*, *eventid*, *timestamp_start*, *timestamp_finish*. Em seguida importamos esse *log* no *software Disco* (Günther and Rozinat, 2012) utilizando a coluna *caseid* como *case*, *eventid* como *activity*, *timestamp_start* e *timestamp_finish* como os *timestamps* (Figura 5.3).

O processo resultante, sem filtros (*slider* de *activity* e *paths* em 100%) é o representado na Figura 5.4. Apesar da quantidade arestas, o processo é legível, principalmente quando comparado ao primeiro resultado (Figura 1.2). Com o filtro máximo de arestas (*paths* em 0%) o processo se torna ainda mais legível (Figura 5.5).

PUC-Rio - Certificação Digital Nº 1912721/CA

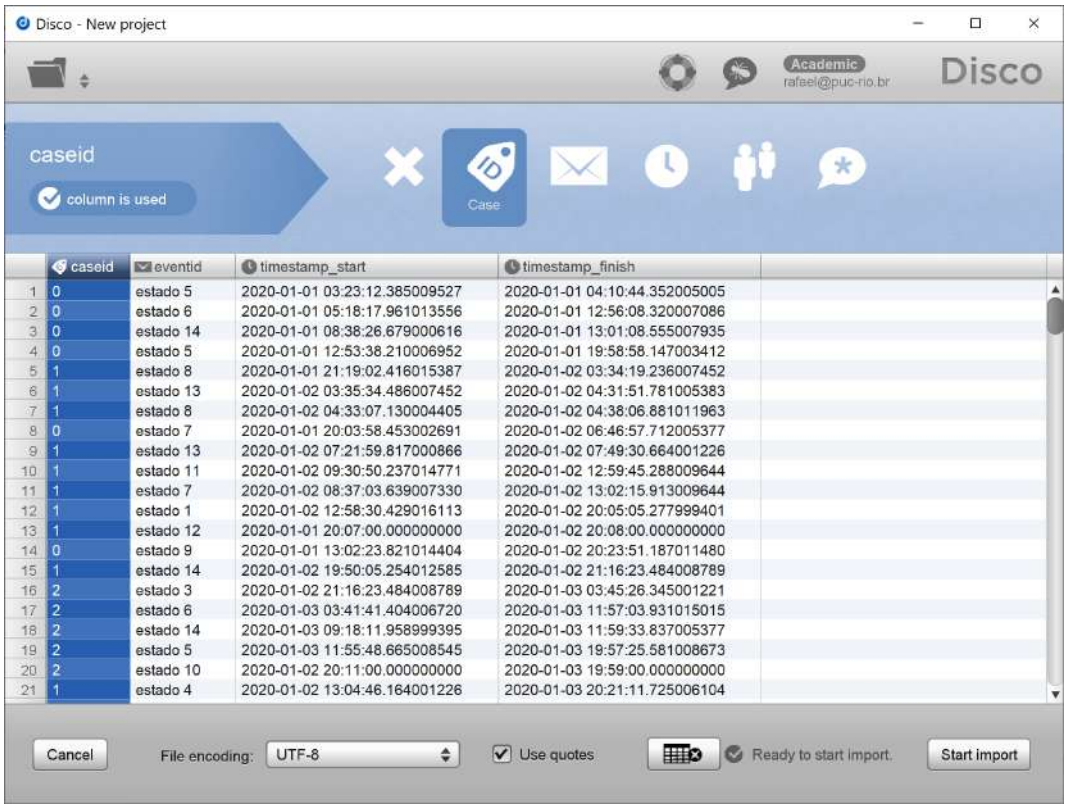


Figura 5.3: Etapa de importação do *log* de casos no *software* Disco.

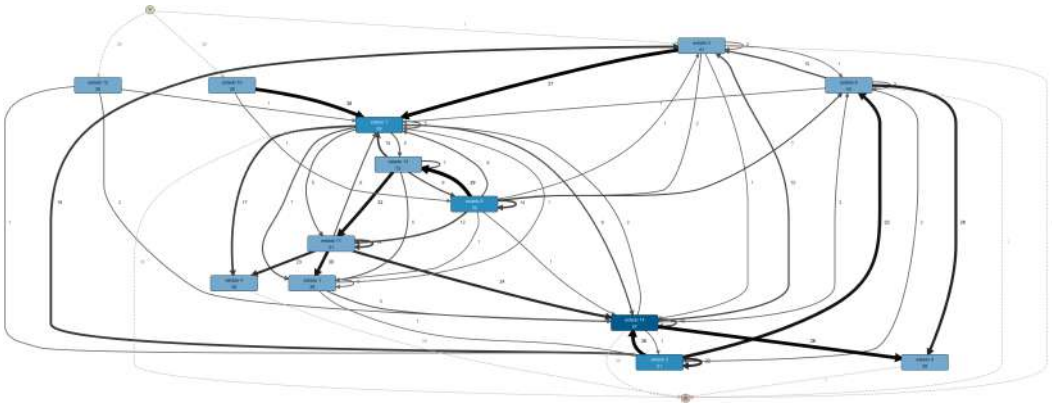


Figura 5.4: Processo gerado pelo Disco, *sliders* de *activity* e *path* 100% (sem filtros).

PUC-Rio - Certificação Digital Nº 1912721/CA

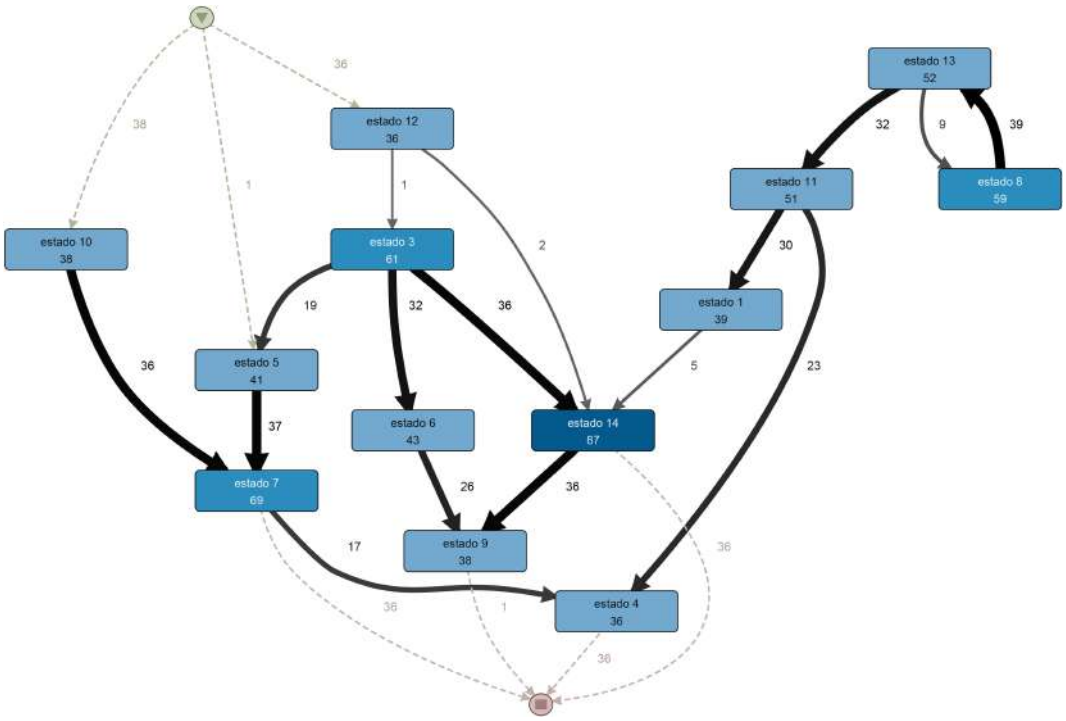


Figura 5.5: Processo gerado pelo Disco, *slider* de *activity* em 100% e de *paths* em 0%.

A principal vantagem desse método é a possibilidade de utilizar os *softwares* e pacotes de mineração já desenvolvidos para a geração e análise do processo. Mas o que de fato fizemos foi reduzir o problema original em segmentos menores, através do agrupamento de válvulas e definição semi arbitrária de casos. Além disso, as relações de paralelismo, muito importantes para o domínio em questão, não são claramente explicitadas na representação de processo utilizada por essas ferramentas. Dessa forma ainda há indícios de que podemos melhorar a estratégia de mineração e representar melhor as relações importantes para o nosso domínio.

5.2 Mineração da estrutura

Para minerar o processo em um *log* sem identificação de caso vamos utilizar uma estrategia que minera diretamente a estrutura do processo. Vamos iterar pelo *log* de estados e cada vez que um estado ativar ou desativar vamos registrar o contexto desse evento, isso é, quais outros estados ativaram ou desativaram recentemente. A proximidade, em tempo, entre os eventos determinará a força da relação entre os estados.

Na Figura 5.6 damos um exemplo de como vamos representar o processo. Os nós dos grafos do processo possuem 3 áreas distintas. Na área central

PUC-Rio - Certificação Digital Nº 1912721/CA

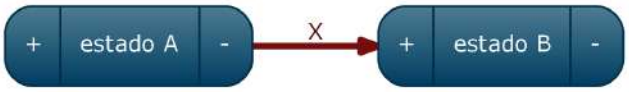
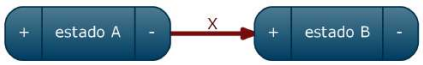
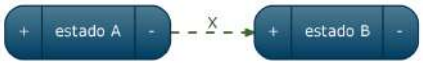
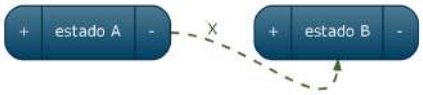
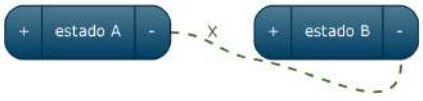


Figura 5.6: Exemplo de representação do nó do grafo do processo.

temos o nome da atividade, aqui representando estados do processo. Arestas que conectam com essa região central representam relações que acontecem enquanto aquele estado estava ativado. A esquerda temos uma região com o símbolo "+", arestas ligadas nesta região representam relações que ocorrem antes da ativação do estado. Por fim, temos uma região com um símbolo "-", arestas conectadas a essa região representam relações que ocorreram depois da desativação do estado.

Na Tabela 5.1 listamos os contextos que queremos capturar, suas descrições, representação em texto e representação no grafo do mapa do processo. Estes contextos serão minerados no *log* de estados, e quando exibidos em conjunto formam o grafo do processo.

Tabela 5.1: Tipos de contexto e suas representações.

Representação em grafo	**Notação**	**Descrição**
	`A->B`	Sequência direta: B começou um pouco depois que A terminou
	`A[]B`	Sequência sobreposta: B terminou um pouco depois que A começou
	`A[[B`	Ativação sequencial: B começou um pouco depois que A começou
	`A]]B`	Desativação sequencial: B terminou um pouco depois que A terminou
	`A[=B`	Ativação próxima: A e B começaram em momentos parecidos
	`A]=B`	Desativação próxima: A e B terminam em momentos parecidos

PUC-Rio - Certificação Digital Nº 1912721/CA

Consideramos como primários os contextos `A->B`, `A[]B`, `A[[B` e `A]]B`, pois são minerados diretamente do *log* de estados. Os contextos `A[=B` e `A]=B` como secundários, minerados a partir dos contextos primários.

5.2.1 Mineração dos contextos primários

A distância máxima de relação entre os estados para a formação dos contextos, assim como a penalidade aplicada a essa distância, serão definidos pelos parâmetros descritos na Tabela 5.2.

Tabela 5.2: Parâmetros que controlam a captura de contextos.

Parâmetro	Descrição
MDO	Distância máxima, em segundos, entre a desativação de um estado A e a ativação de um estado B para gerar o contexto `A->B`.
MDI	Distância máxima, em segundos, entre a ativação de um estado A e a desativação de um estado B para gerar o contexto `A[]B`.
MDP	Distância máxima, em segundos, entre a ativação de um estado A e a ativação de um estado B para gerar o contexto `A[[B`. Também controla a distância máxima entre a desativação de um estado A e a desativação de um estado B para gerar o contexto `A]]B`.
MW	Peso mínimo para um contexto que está na distância máxima permitida para a sua geração. Entre 0 e 1.

A mineração dos contextos primários é feita através de uma passagem pelo *log* de estados. Similar a geração do *log* de casos, para identificar o momento de ativação de um estado do processo basta checar se na entrada atual ele está ativado e na entrada anterior desativado. Simetricamente, a identificação da desativação se dá quando o estado está desativado na entrada atual do *log* e ativado na anterior.

Para gerar um contexto primário válido, toda vez que uma ativação ou desativação for detectada precisamos checar se ela ocorreu dentro do limite de tempo definido de outras ativações e desativações.

Deve ser registrado, junto com o contexto, o peso ω da relação. Esse peso é calculado por uma interpolação baseada na distância d entre os eventos (ativação e desativação) e o valor mínimo definido pelo parâmetro MW. Onde quanto menor a distância entre os eventos, maior o peso da relação, calculado

PUC-Rio - Certificação Digital Nº 1912721/CA

conforme a fórmula 5-1.

$$\omega = \begin{cases} 0 & d = 0 \\ 1 + (d * (MW - 1)/M) & \text{c.c.} \end{cases} \quad (5\text{-}1)$$

Com M podendo assumir os valores dos parâmetros MDO, MDI ou MDP dependendo do tipo do contexto.

5.2.2 Simplificação dos contextos primários

Os contextos encontrados representam relações entre os estados e eventualmente se tornarão arestas no grafo do processo. Para não sobrecarregar o processo final com informações pouco relevantes vamos fazer algumas filtragens nos contextos encontrados. Antes de remover os contextos pouco frequentes vamos verificar se é possível mesclar contextos. Sendo esses os que representam relações parecidas e que possuem um desequilíbrio muito grande de frequência entre si.

O primeiro caso dessa redução envolve dois contextos primários: sequência direta e sequência sobreposta. Ambos refletem uma relação de sequência entre estados, mas quando dois estados possuem os dois tipos de relação entre si, como exemplificado na Figura 5.7, podemos considerar mesclar as duas relações em uma só – Figura 5.8.

Figura 5.7: Relações de sequência direta (`A->B`) e sequência sobreposta (`A[]B`) entre os mesmos estados.

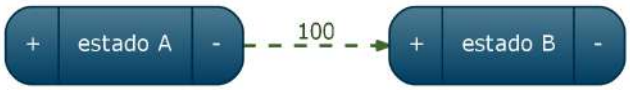


Figura 5.8: Relação de sequência sobreposta (`A[]B`).

Definimos que, quando o peso total de uma das duas relações é muito maior do que a outra, apenas a maior relação se mantém. A frequência da relação mantida agora é a soma da frequência das duas relações originais. O critério utilizado para definir quando mesclar os dois tipos de relação é controlado pelo parâmetro ERS e implementado segundo o Algoritmo 9.

PUC-Rio - Certificação Digital Nº 1912721/CA

Algoritmo 9 Simplificação de contextos de sequência

para todo $A, B \in E$ **faça**
  $F_x \leftarrow$ Frequência de `A->B`
  $F_y \leftarrow$ Frequência de `A[]B`
  $P_x \leftarrow$ Peso total de `A->B`
  $P_y \leftarrow$ Peso total de `A[]B`
  $R \leftarrow |P_x - P_y|/(P_x + P_y)$
  se $R \geq ERS$ **então**
    se $P_x \geq P_y$ **então**
      Frequência de `A->B` $= F_x + F_y$
      Frequência de `A[]B` $= 0$
      Peso total de `A->B` $= P_x + P_y$
      Peso total de `A[]B` $= 0$
    senão
      Frequência de `A->B` $= 0$
      Frequência de `A[]B` $= F_x + F_y$
      Peso total de `A->B` $= 0$
      Peso total de `A[]B` $= P_x + P_y$
    fim se
  fim se
fim para

Dessa forma, o parâmetro ERS controla a diferença mínima necessária entre os pesos dos contextos para mesclar os dois. Quanto menor o ERS, mais contextos serão mesclados.

5.2.3 Geração dos contextos secundários

Enquanto os contextos primários coletam informações do *log* entrada-a-entrada, os contextos secundários identificam novos contextos baseado nos contextos primários. Para encontrar contexto secundário de ativação próxima, `A[=B`, vamos procurar por casos onde o peso total entre ativações sequenciais de `A[[B` e `B[[A` é muito próximo. Transformando duas relações em uma única relação `A[=B`, como exemplificado nas Figuras 5.9 e 5.10.

O grau de proximidade necessário entre os pesos para a transformação é controlado pelo parâmetro ERP (entre 0 e 1). Dessa forma, as relações `A[=B` podem ser encontradas segundo o Algoritmo 10.

PUC-Rio - Certificação Digital Nº 1912721/CA

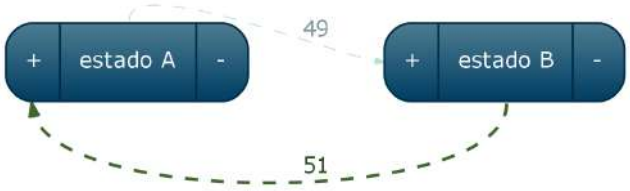


Figura 5.9: Relações de ativação sequencial `A[[B` e `B[[A`.

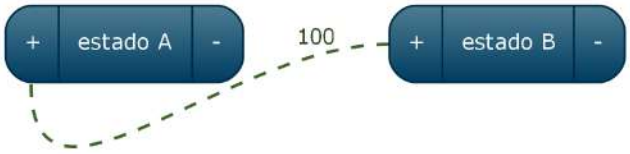


Figura 5.10: Relação de ativação próxima `A[=B` ou `B[=A`.

Algoritmo 10 encontrando contextos do tipo ativação próxima

para todo $A, B \in E$ **faça**
  $F_x \leftarrow$ Frequência de `A[[B`
  $F_y \leftarrow$ Frequência de `B[[A`
  $P_x \leftarrow$ Peso total de `A[[B`
  $P_y \leftarrow$ Peso total de `B[[A`
  $R \leftarrow |P_x - P_y|/(P_x + P_y)$
  se $R < (1 - ERP)$ **então**
    Frequência de `A[=B` $= F_x + F_y$
    Frequência de `A[[B` $= 0$
    Frequência de `B[[A` $= 0$
    Peso total de `A[=B` $= P_x + P_y$
    Peso total de `A[[B` $= 0$
    Peso total de `B[[A` $= 0$
  fim se
fim para

A mineração do contexto de desativação próxima, `A]=B`, é feita de forma semelhante, mas analisando as sequências de desativação `A]]B` e `B]]A`, como descrito no Algoritmo 11.

PUC-Rio - Certificação Digital Nº 1912721/CA

Algoritmo 11 encontrando contextos do tipo desativação próxima

para todo $A, B \in E$ **faça**
    $F_x \leftarrow$ Frequência de `A]]B`
    $F_y \leftarrow$ Frequência de `B]]A`
    $P_x \leftarrow$ Peso total de `A]]B`
    $P_y \leftarrow$ Peso total de `B]]A`
    $R \leftarrow |P_x - P_y|/(P_x + P_y)$
    se $R < (1 - ERP)$ **então**
        Frequência de `A]=B` $= F_x + F_y$
        Frequência de `A]]B` $= 0$
        Frequência de `B]]A` $= 0$
        Peso total de `A]=B` $= P_x + P_y$
        Peso total de `A]]B` $= 0$
        Peso total de `B]]A` $= 0$
    fim se
fim para

PUC-Rio - Certificação Digital Nº 1912721/CA

Por definição, as relações `A[=B` e `B[=A` são equivalentes, assim como as relações `A]=B` e `B]=A`.

Os contextos `A[=B` e `A]=B` estão ligados as relações de paralelismo encontradas na literatura de *process mining* (Leemans et al., 2014), dado que podemos interpretar que se `A[[B` e `B[[A` acontecem com uma frequência muito parecidas, então temos poucos indícios para afirmar que A acontece antes de B, nem o contrário. Só podemos afirmar que A e B possuem alguma correlação. Mas, os métodos de *process mining* normalmente representam esse paralelismo como nenhuma aresta entre A e B.

5.2.4 Corte de contextos

Por fim, o recurso que utilizaremos para simplificar o processo é a redução de contextos por corte. Todo contexto entre dois estados A e B quaisquer será descartado se seu peso total for menor do que uma fração da frequência de A e de B. Essa fração minima é controlada pelo parâmetro ERC (entre 0 e 1). Sua implementação é segundo o Algoritmo 12.

Algoritmo 12 corte de contextos de baixo peso

```
1: para todo A, B ∈ E faça
2:     para todo %% ∈ contextos existentes faça
3:         P ← Peso total de A%%B
4:         F_x ← Ocorrências de A
5:         F_y ← Ocorrências de B
6:         se P < F_x * (1 − ERC) & P < F_y * (1 − ERC) então
7:             Frequência de A%%B = 0
8:             Peso de A%%B = 0
9:         fim se
10:    fim para
11: fim para
```

Com isso respondamos integralmente **PP3:** Dado um *log* de eventos da planta sem identificação de casos, é possível minerar a estrutura de um processo? Sim, o agrupamento das válvulas em estados permitiu que encontrássemos relações de sequência e paralelismo entre os estados através da busca pelos momentos de ativação e desativação de cada estado.

5.3 Outros tipos de paralelismo

Nos contextos tratamos apenas das relações de paralelismo entre as extremidades dos estados - o início (ativação) e o final (desativação). Decidimos por não representar paralelismo dos períodos de interseção entre os estados em um grafo de processo.

A representação obtida em um gráfico de Gantt (Wilson, 2003) seria capaz de expressar os momentos de intersecção entre os estados ao longo do *log* de estados. O desafio é como sobrepor as ocorrências de um mesmo estado para gerar uma visão geral do estado, de forma que o gráfico de Gantt não se torne muito extenso e difícil de analisar.

No problema do coqueamento retardado, a visualização de tempo de intersecção entre os estados é um recurso a mais para analisar quais estados fazem parte de qual processo de produção. Sabemos que os estados que representam o enchimento dos reatores são longos, contínuos e alternados. Dessa forma, os estados que possuem uma grande intersecção de tempo com um estado que representa um enchimento de um reator têm alta probabilidade de representar uma etapa do processo de produção do outro reator.

Uma alternativa compacta para ter uma visão geral do tempo de intersecção de atividade entre os estados é através de um *heatmap* (Wilkinson and

PUC-Rio - Certificação Digital Nº 1912721/CA

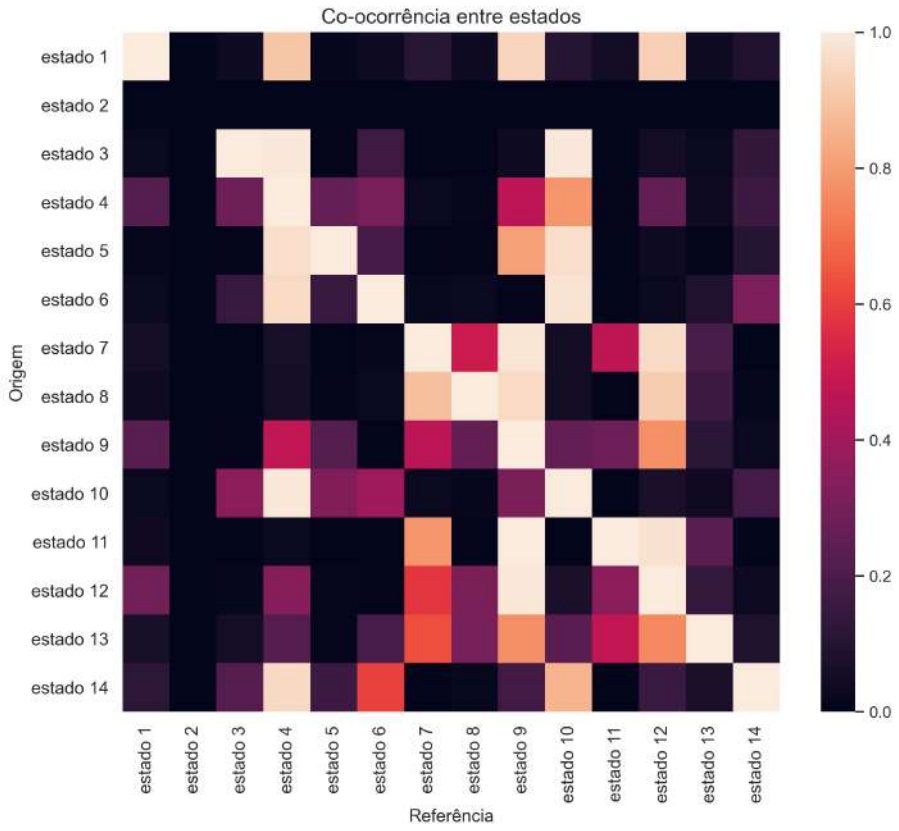


Figura 5.11: *Heatmap* da intersecção de tempo de ocorrência entre os estados.

Friendly, 2009), como fizemos na Figura 5.11. Nela, cada célula representa o quanto o estado do eixo Origem ficou ativo junto com o estado do eixo Referência em relação ao seu tempo total de atividade. Onde um valor próximo a 0 representa que o estado de referência ficou muito pouco tempo ativo enquanto o estado de origem estava ativo.

Como já citado, no conjunto de estados do processo E, os estados 10 e 12 representam o enchimento dos reatores. Na Figura 5.11 é possível ver que, como esperado, as colunas do estado 10 e estado 12 se complementam no sentido que, ao somarmos as duas, quase todas as suas células ficam com valores próximos a 1. Isso reflete o comportamento alternado entre o enchimento dos reatores.

Relembrando, enquanto um reator está em enchimento o outro passa por diversas fases de preparação. Ainda na Figura 5.11, podemos perceber que na coluna do estado 10, os valores dos estados 3, 5, 6 e 14 são altos. Isso indica que eles ocorrem com maior frequência enquanto o estado 10 está ativo e por isso provavelmente representam operações relacionadas ao reator do estado 12. O mesmo ocorre para os estados 1, 7, 8 e 9 em relação ao estado 12. Com valores próximos a 0.5, distribuídos entre as colunas dos estados 10 e 12, os estados 4, 9

PUC-Rio - Certificação Digital Nº 1912721/CA

e 13 ocorrem com uma frequência considerável durante o enchimento de ambos reatores, representando um processo comum. Finalmente, o estado 2 reúne as válvulas que nunca abrem ou não tiveram um padrão claro o suficiente para o algoritmo de agrupamento.

Essas conclusões indicam que ainda existem relações a serem exploradas no *log* de estados. Utilizar as informações presentes no *heatmap* da Figura 5.11 na mineração de contextos pode fornecer mais ferramentas para trabalhar em conjunto com os outros contextos já existentes. Porém, não identificamos uma forma direta de utilizar essas informações.

5.4 Resultados e ajustes

Temos um total de 7 parâmetros controlando a mineração do processo. Os parâmetros MDO, MDI, MDP e MW controlam a distância máxima entre os contextos e são dependentes do domínio da aplicação. Os parâmetros ERS, ERP e ERC controlam como os contextos – que se tornarão arestas - deverão ser reduzidos e transformados. A quantidade de arestas no grafo do processo é proporcional aos valores dos parâmetros, isso é, ao aumentar o valor de qualquer um dos parâmetros, é esperado que a quantidade de arestas no processo final aumente. Nesta seção vamos apresentar e discutir resultados da mineração do *log* de estados L_e com diferentes parâmetros.

Começando do pior caso, com todos os parâmetros relaxados ao máximo e uma distância de captura de contexto de 30 minutos. O resultado pode ser conferido na Figura 5.12. Apesar de muito mais legível do que o resultado inicial, Figura 1.4, a quantidade de arestas ainda pode ser grande demais para uma análise visual.

A seguir apenas reduzimos um pouco o valor do parâmetro ERC, que faz um corte indiscriminado de arestas pouco frequentes. Decidimos por apresentar a variação deste parâmetro primeiro devido a sua eficiência em simplificar o processo. O resultado, na Figura 5.13, é mais compressível o anterior.

Para quantificar a perda de expressividade que essa simplificação causou comparamos a redução da quantidade de arestas únicas com a redução da frequência total das arestas. A quantidade de arestas únicas representa a variação no padrão de sequencia de ativações, a frequência das arestas representa a consistência de uma sequencia. Dessa forma, consideramos boas as simplificações que reduzem o numero de arestas sem reduzir muito a frequência total das arestas.

Comparando o grafo do processo da Figura 5.12 com o da 5.13 percebemos que tivemos uma redução da quantidade de arestas de mais de 50%, mas

PUC-Rio - Certificação Digital Nº 1912721/CA

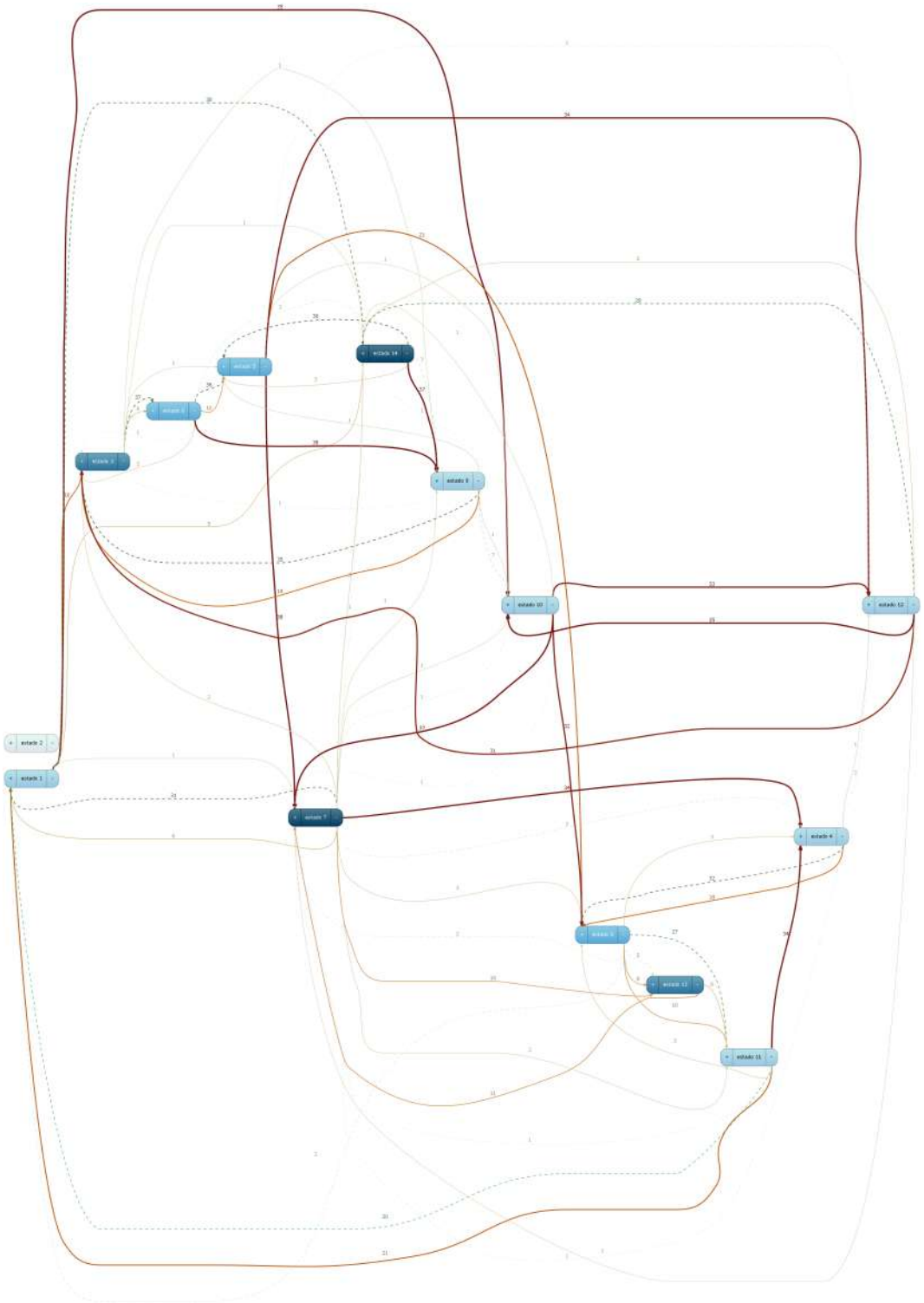


Figura 5.12: Grafo do processo. 74 arestas com frequência total 977
Parâmetros: MDO=30min, MDI=15min, MDP=30min, MW=0.1
ERS=1, ERP=1, ERC=1

PUC-Rio - Certificação Digital Nº 1912721/CA

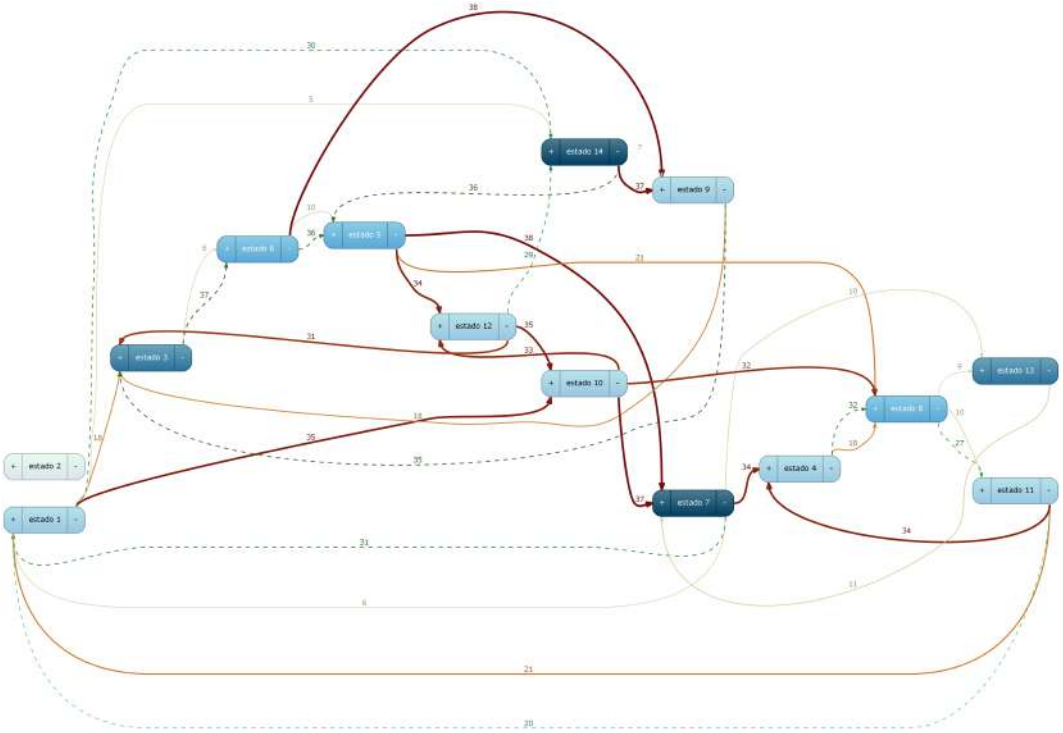


Figura 5.13: Grafo do processo. 36 arestas com frequência total 901
Parâmetros: MDO=30min, MDI=15min, MDP=30min, MW=0.1
ERS=1, ERP=1, ERC=0.9

a frequência total das arestas caiu menos de 10%, indicando um bom custo benefício para uma redução suave do valor do parâmetro ERC.

Aplicando a fusão de contextos, reduzimos os valores dos parâmetros ERS e ERP para 0.5 e o resultado está na Figura 5.14. Como consequência a quantidade de arestas únicas caiu em aproximadamente 20%, mas a frequência total de arestas subiu ligeiramente. Esse resultado é esperado, pois após mesclar dois contextos o peso do contexto resultante subiu, deixando seu valor acima do limiar de remoção definido pelo parâmetro de corte ERC. Apesar de uma redução de arestas menor do que o causado diretamente pelo parâmetro ERC, o uso dos parâmetros ERS e ERP permite uma redução significava na quantidade de arestas únicas sem comprometer a frequência total de arestas. Podemos pensar no método que utiliza parâmetros ERS e ERP como uma ferramenta para agrupar contextos parecidos.

No próximo passo fizemos uma redução nos valores dos parâmetros MDO, MDI e MDP, que representam a distância de captura de contexto. Como podemos ver na Figura 5.15, a redução de arestas únicas foi na casa dos 20% e da frequência total perto dos 15%. Esse é um parâmetro que deve ser utilizado considerando sempre as características do domínio. Pois uma redução excessiva pode implicar na falha na captura de relações frequentes importantes. Por outro lado, valores muito altos podem capturar contextos

PUC-Rio - Certificação Digital Nº 1912721/CA

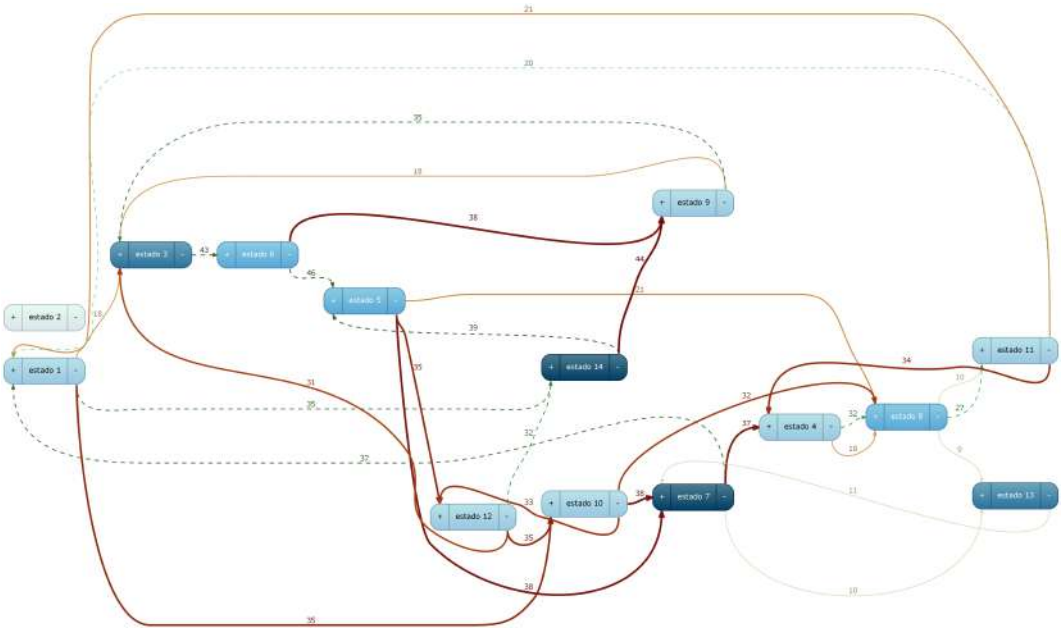


Figura 5.14: Grafo do processo. 31 arestas com frequência total 912
Parâmetros: MDO=30min, MDI=15min, MDP=30min, MW=0.1
ERS=0.5, ERP=0.5, ERC=0.9

não relacionados. Por isso deve-se ponderar, dado o domínio, qual é o tempo máximo entre eventos para considerar que eles podem estar relacionados.

Na Figura 5.16 temos o resultado aumentando o valor do parâmetro MW. Esse parâmetro não controla a distância máxima de captura, mas sim o peso mínimo para eventos muito distantes dentro dessa distância máxima definida por MDO, MDI e MDP. No caso do *log* de estados L_e, essa alteração teve um reflexo pequeno, mas conseguimos ver aplicação em casos mais ruidosos. Para os próximos experimentos manteremos o valor anterior de MW.

Até momento os resultados exibem apenas as arestas geradas pelos contextos de sequência direta e sequência sobreposta. Na Figura 5.17 temos um grafo de processo arestas representando todas as relações descritas na Tabela 5.1.

Por último, sem mudar nenhum parâmetro, na Figura 5.18 temos um processo que exibe apenas as relações de ativação e desativação entre os estados. Esse tipo de visualização é importante em casos onde queremos buscar por algum tipo de correlação entre ativações e desativações entre estados. Nesse caso percebemos a formação de alguns agrupamentos, sugerindo que alguns estados possuem alguma relação de início e fim concorrente. Como por exemplo os estados 7 e 11, que no processo representado na Figura 5.15 verificamos que não há uma relação de sequencia entre eles, mas agora, pela Figura 5.18, percebemos que existe uma relação entre suas desativações.

PUC-Rio - Certificação Digital Nº 1912721/CA

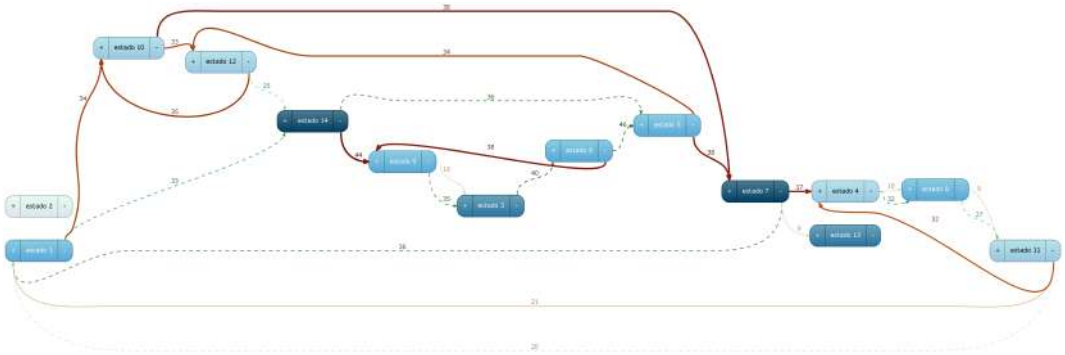


Figura 5.15: Grafo do processo. 25 arestas com frequência total 768
Parâmetros: MDO=15min, MDI=9min, MDP=15min, MW=0.1
ERS=0.5, ERP=0.5, ERC=0.9

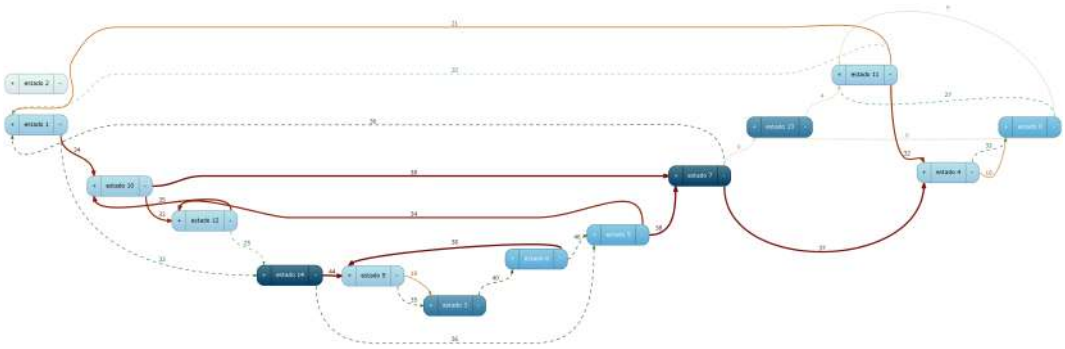


Figura 5.16: Grafo do processo. 27 arestas com frequência total 778
Parâmetros: MDO=15min, MDI=9min, MDP=15min, MW=1.0
ERS=0.5, ERP=0.5, ERC=0.9

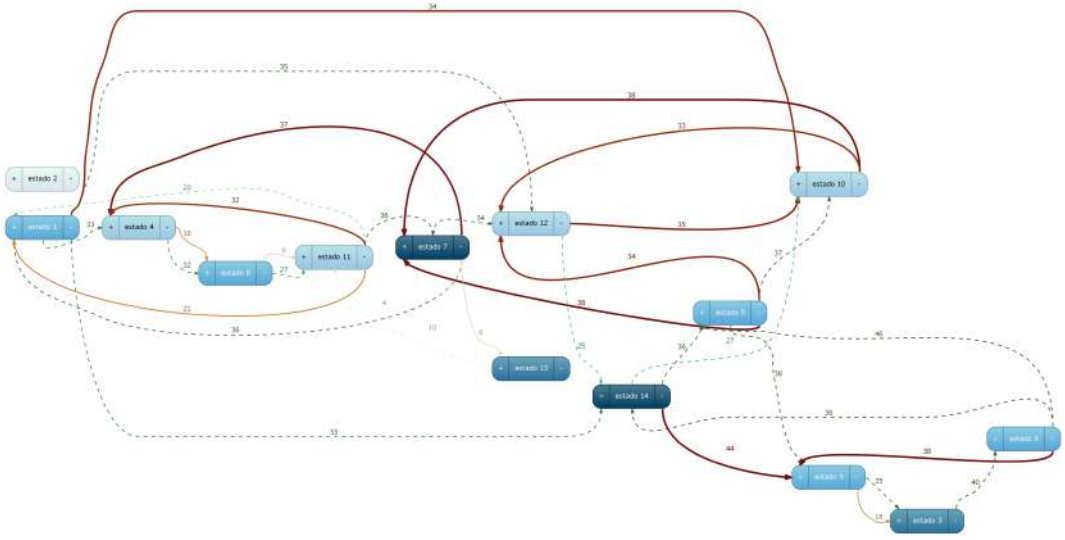


Figura 5.17: Grafo do processo. 35 arestas com frequência total 1060
Parâmetros: MDO=15min, MDI=9min, MDP=15min, MW=0.1
ERS=0.5, ERP=0.5, ERC=0.9

PUC-Rio - Certificação Digital Nº 1912721/CA

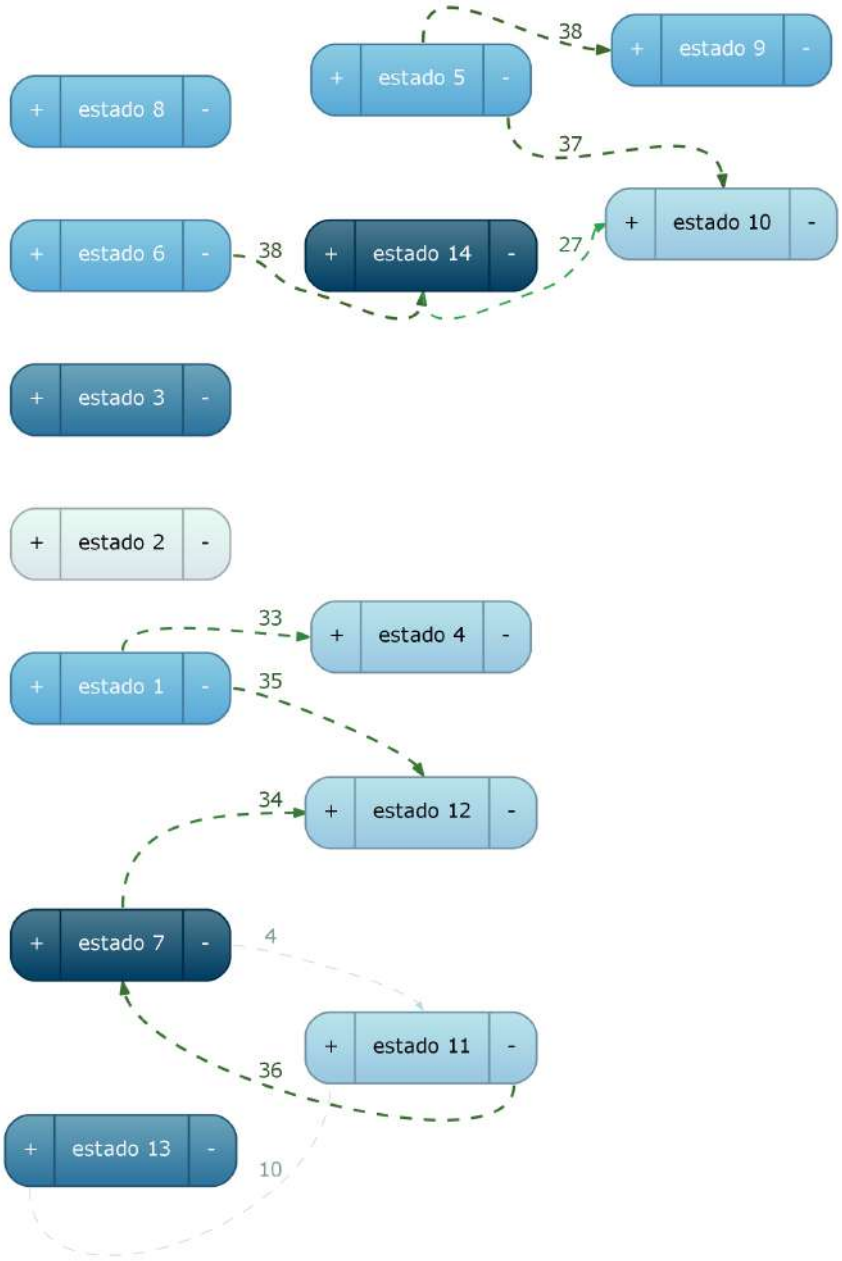


Figura 5.18: Grafo do processo. 10 arestas com frequência total 292
Parâmetros: MDO=15min, MDI=9min, MDP=15min, MW=0.1
ERS=0.5, ERP=0.5, ERC=0.9

PUC-Rio - Certificação Digital Nº 1912721/CA

6
Conclusões

Neste trabalho propomos uma solução para minerar um processo a partir de um registro sem identificação de casos. Essa proposta foi desenvolvida a partir de estudos para responder três perguntas de pesquisa:

- **PP1:** Dado um *log* de eventos da planta, é possível descobrir estados relacionados a operação da unidade?
- **PP2:** Dados os estados do processo e um *timestamp* do *log* de eventos da planta, é possível identificar quais estados estão ativos no momento?
- **PP3:** Dado um *log* de eventos da planta sem identificação de casos, é possível minerar a estrutura de um processo?

Demonstramos que técnicas de agrupamento, mais especificamente o AHC, podem ser aplicadas ao registro de eventos da unidade para encontrar grupos de válvulas que possuem um padrão de funcionamento semelhante. Mesmo com a limitações impostas pelas características do AHC, a avaliação dos grupos resultantes indicou uma alta capacidade de representarem funções da operação de coqueamento retardado, respondendo assim a primeira pergunta de pesquisa na Seção 4.3.

A avaliação adequada dos grupos resultantes do AHC se revelou de suma importância, pois é a partir desses grupos que o descobrimento do processo é realizado. Por isso foi necessário a aplicação de métricas de comparação entre os grupos encontrados e grupos de referência. Além das métricas ficou evidente o ganho em utilizar ferramentas como o `PLANTSTATE`, que contextualizam os grupos descobertos na planta, para avaliar os resultados.

O resultado da etapa de agrupamento são grupos de válvulas, então é possível utilizar técnicas além das exploradas nessa pesquisa para encontrar esses grupos, reforçando a modularidade da solução proposta. E dado que os grupos resultantes possuem uma formatação clara, eles podem ser facilmente corrigidos e redefinidos pelo usuário.

Encontrar os momentos de ativação dos estados do processo dependeu apenas de um critério. Esse critério define se as válvulas ativas em um dado momento do *log* original atendem as válvulas definidas em um estado do processo. O critério proposto gerou resultados satisfatórios para o domínio

PUC-Rio - Certificação Digital Nº 1912721/CA

explorado, respondendo, na Seção 4.4, a segunda pergunta de pesquisa. Como o único requisito desta etapa é identificar os momentos que estados do processo iniciam ou terminam, o critério de ativação pode ser estendido para que a solução proposta atenda outros domínios.

Utilizando as informações dos momentos de inicio e fim do período de atividade dos estados do processo é possível encontrar relações entre eles. Posteriormente essas relações podem ser processadas para construir a estrutura do processo. Dessa forma, no Capítulo 5, respondemos a última pergunta de pesquisa.

Para simplificar a estrutura encontra encontrada o tradicional corte de relações pouco frequentes se mostrou eficiente. Porém, vimos que também é possível simplificar o processo agrupando relações similares, reduzindo a quantidade final de arestas únicas no processo sem reduzir a contagem total de arestas, o que resulta em relações mais bem definidas.

Vimos também que uma representação de modelo de processo mais flexível pode ajudar na extração de informações do processo. No modelo proposto podemos escolher quais tipo de relações precisamos que sejam plotadas no grafo de processo, permitindo concentrar a análise nas relações de interesse no momento.

Alguns dos parâmetros utilizados na solução proposta dependem de algum conhecimento do domínio. Na maioria das vezes o impacto desses parâmetros é diretamente identificável no processo gerado e facilmente ajustáveis. Porém, a definição do tempo máximo que indica se duas operações podem estar relacionadas é altamente depende das particularidades de cada operação.

A avaliação dos resultados obtidos nessa pesquisa foi relacionada a interpretabilidade do processo minerado com o método proposto em comparação com a mineração obtida com os *softwares* comerciais. Ainda é necessário fazer uma avaliação se o resultado obtido representa as operações. Essa validação só é possível em conjunto com um especialista da operação. Até a data de entrega deste documento, não conseguimos a disponibilidade de um especialista para essa validação de domínio.

Por fim, a solução proposta possui propriedades modulares, o que possibilita o aprimoramento de etapas de forma independente e a reutilização parcial ou completa da solução em outras aplicações. Na próxima seção vamos finalizar este documento com algumas possibilidades de trabalhos futuros.

PUC-Rio - Certificação Digital Nº 1912721/CA

6.1 Trabalhos futuros

Uma expansão natural na etapa de descoberta dos estados do processo é pesquisar sobre como transformar os grupos obtidos pelo AHC dos segmentos de tempo em grupos de válvulas. Uma solução para esse problema resultaria em um AHC capaz de gerar grupos mais próximos das operações da unidade de coqueamento retardado e mais generalizável para outros domínios.

Outra alternativa para o descobrimento dos estados do processo é utilizar outras técnicas, descartando completamente o AHC. Considerando o *log* de segmentos como uma matriz, técnicas de fatoração de matrizes, como o BMF (Miettinen and Neumann, 2020) e NMF (Dhillon and Sra, 2005), tem potencial de identificação de padrões nessas matrizes, que podem representar funções da unidade de operação. Da mesma forma, ao interpretar os dados como uma sequencia de padrões que se repetem, é possível aplicar técnicas de identificação de comunidades (Clauset et al., 2004).

Assim como na evolução das técnicas de PM, ainda há espaço para novos processamentos de contextos. Informações como o tempo de coativação entre dois estados podem trazer novas formas de agrupar contextos. Além disso, analisamos apenas as relações entre dois estados, pode ser que analisando relações entre grupos maiores de estados proporcione uma novas perspectivas, assim como aconteceu com o agrupamento de válvulas em estados.

Sobre a expansão do método, neste projeto nos concentramos apenas nas informações das válvulas. Segundo os manuais de operação, alguns estados do processo são diferenciáveis apenas pelos valores de leitura em sensores como vazão e temperatura. Como considerar os dados dos sensores na definição dos estados do processo ainda é uma pergunta em aberto.

Por fim, nesta solução tratamos apenas da primeira etapa da mineração de processos. Como realizar a etapa de *conformance checking* e como visualizar os dados originais no grafo do processo são assuntos ainda inexplorados nesse domínio.

PUC-Rio - Certificação Digital Nº 1912721/CA

Referências Bibliográficas

(2021). Mining-driven process controlling, or: What does the process look like? `https://www.bankinghub.eu/banking/operations/mining-driven-process-controlling`. Accessed: 2021-05-10.

Bundy, A. and Wallen, L. (1984). Breadth-first search. In *Catalogue of artificial intelligence tools*, pages 13–13. Springer.

Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111.

Day, W. H. and Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of classification*, 1(1):7–24.

Dhillon, I. S. and Sra, S. (2005). Generalized nonnegative matrix approximations with bregman divergences. In *NIPS*, volume 18. Citeseer.

Günther, C. W. and Rozinat, A. (2012). Disco: Discover your processes. *BPM (Demos)*, 940:40–44.

Günther, C. W. and Van Der Aalst, W. M. (2007). Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *International conference on business process management*, pages 328–343. Springer.

Köppen, M. (2000). The curse of dimensionality. In *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*, volume 1, pages 4–8.

Leemans, S. J., Fahland, D., and Van Der Aalst, W. M. (2014). Process and deviation exploration with inductive visual miner. *BPM (demos)*, 1295(8).

Miettinen, P. and Neumann, S. (2020). Recent developments in boolean matrix factorization. *arXiv preprint arXiv:2012.03127*.

Van Der Aalst, W. (2012). Process mining. *Communications of the ACM*, 55(8):76–83.

Weijters, A. J. and Van der Aalst, W. M. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162.

PUC-Rio - Certificação Digital Nº 1912721/CA

Wilkinson, L. and Friendly, M. (2009). The history of the cluster heat map. *The American Statistician*, 63(2):179–184.

Wilson, J. M. (2003). Gantt charts: A centenary appreciation. *European Journal of Operational Research*, 149(2):430–437.

PUC-Rio - Certificação Digital Nº 1912721/CA