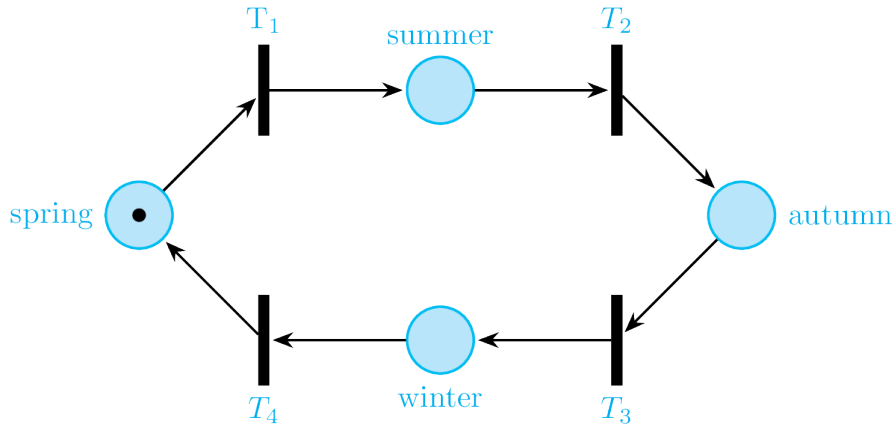
Uma Estratégia de Simulação CLEMATIS-SimPy

Uma breve abordagem

Aluno: Pedro Paulo Araújo
Orientador (a): Felipe Verri
Co-orientador (a): Paulo Victor

22 de abril de 2023



Sumário

Seção atual

Introdução

Contexto

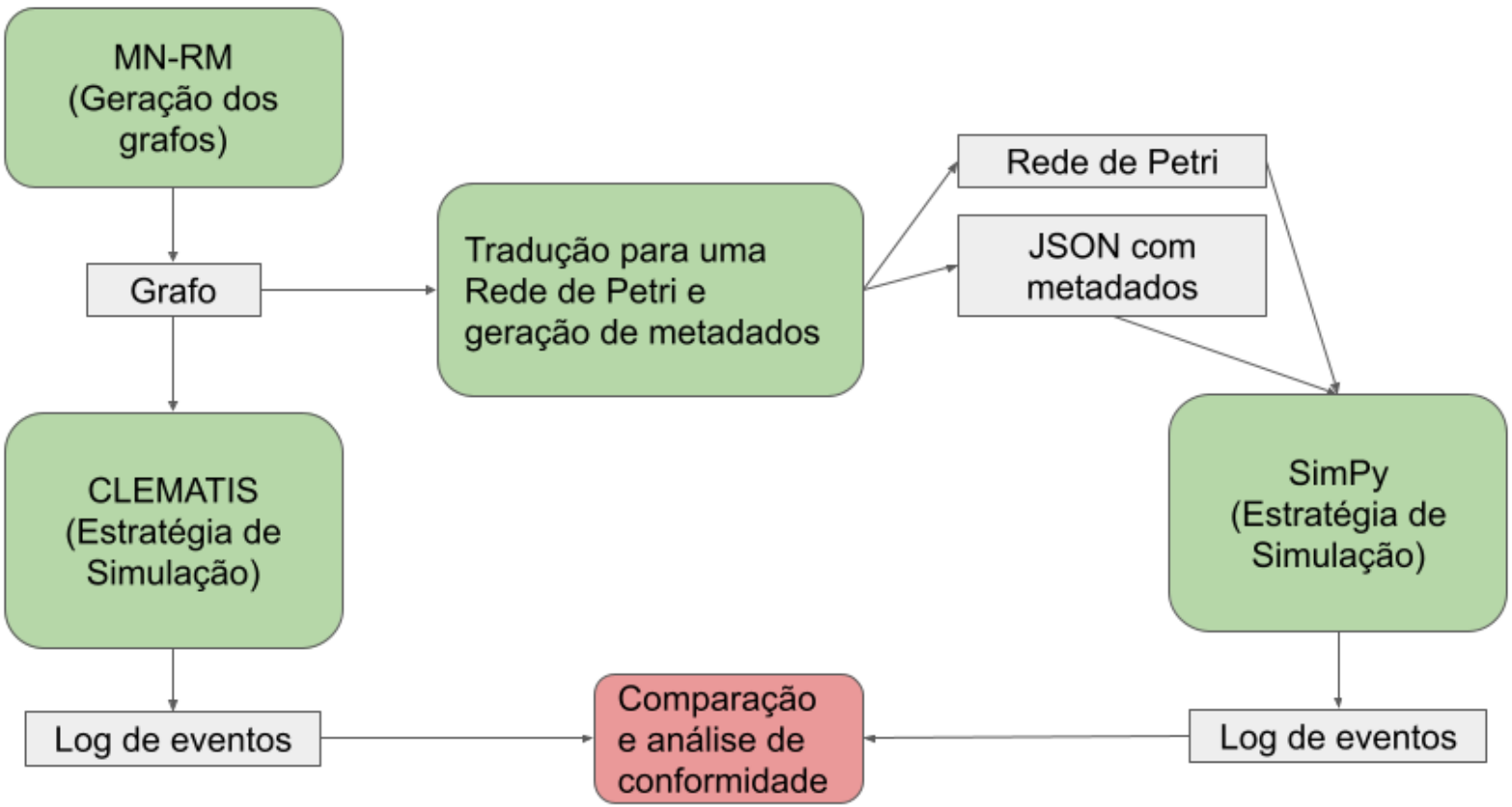
Contexto

Tendo em vista as estratégias simulação existentes, a existência de um modelo de fácil utilização e que permitisse a inclusão de mais variáveis como *production rate* e *buffer size*, se fez relevante.

Objetivo

O objetivo deste projeto foi o desenvolvimento de um modelo de simulação que permitisse a inclusão de variáveis diferentes, e a comparação do modelo produzido com os atuais.

Diagrama


MN-RM
(Geração dos
grafos)
Grafo
Tradução para uma
Rede de Petri e
geração de metadados
Rede de Petri
JSON com
metadados
CLEMATIS
(Estratégia de
Simulação)
SimPy
(Estratégia de
Simulação)
Log de eventos
Comparação
e análise de
conformidade
Log de eventos

Seção atual

O que foi feito?

Tradução

Tradução

A tradução do grafo foi feita utilizando uma biblioteca chama PnTools, desenvolvida para a manipulação de petriNets, ela tem o potêncial de criar arquivos **pnml** *(Petri Net Markup Language)*, que basicamente é uma extensão do formato **xml**, esse formato de arquivo é vantajoso pois permite a comunicação com a biblioteca de mineração de processos **PM4PY**.

JSON para metadados

Como um dos objetivos foi que esta estratégia de simulação considerasse determinado buffer para cada um dos nós, bem como tempo de atividade para cada um deles, no momento da tradução, um arquivo **JSON** é gerado contendo cada uma das informções dos respectivos nós.

A escolha do formato **JSON** se deve a sua interoperabilidade e facilidade na manipulação dos dados.

O que foi feito?

Simulação

Sobre a Simulação

A simulação consiste em ler os dados do arquivo **pnml** e **JSON** e modelá-los na classe criada *Node*, classe essa que possui todos os atributos relacionados ao nó necessários para a simulação.

Modelagem ao SimPy

Para que fosse possível se utilizar do máximo de ferramentas dipo-níveis no SimPy, foi necessário modelar de acordo com o funciona-mento do mesmo.

Como cada nó representaria em tese uma máquina no chão de fá-brica, cada nó possui um método **operate** que permite que a máquina possa processar os tokens.

Para modelar o buffer de cada máquina, foi utilizada uma das es-trutras de **resources** do SimPy, a **Store** que permite o armazenamento de objetos do tipo de python e transferência de objetos entre **buffers**

Saida

Ao fim da simulação os dados são armazenados em um arquivo **txt** que pode posteriormente ser análisado atráves da análise de conformidade.

Seção atual

Comparação e Análise Superficial

Modelo Original x Modelo Descoberto

Exemplo 1

Neste exemplo um grafo de 3 nós, 2 etapas de produção e fator randômico 2, foi analisado.

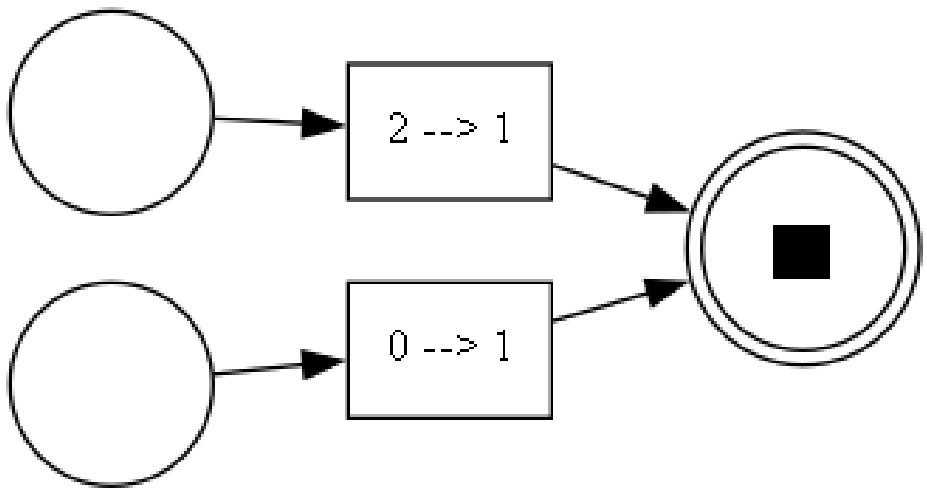


Figura 1: Modelo Original MN-RM

Exemplo 1

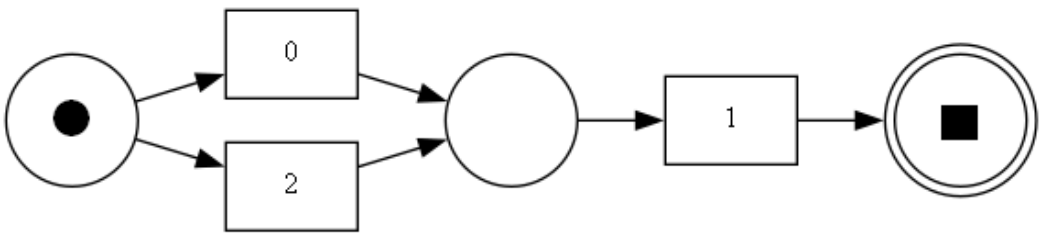


Figura 2: Modelo Descoberto Alpha Miner

Exemplo 2

Neste outro exemplo um grafo com $\alpha = 0.7$ de 10 nós, 7 etapas de produção e fator randômico 2, foi analisado.

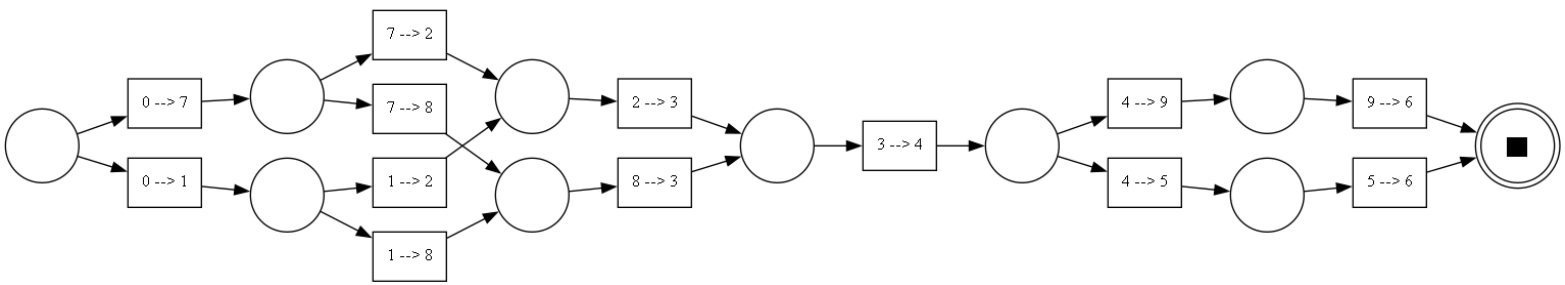


Figura 3: Modelo Original MN-RM

Exemplo 2

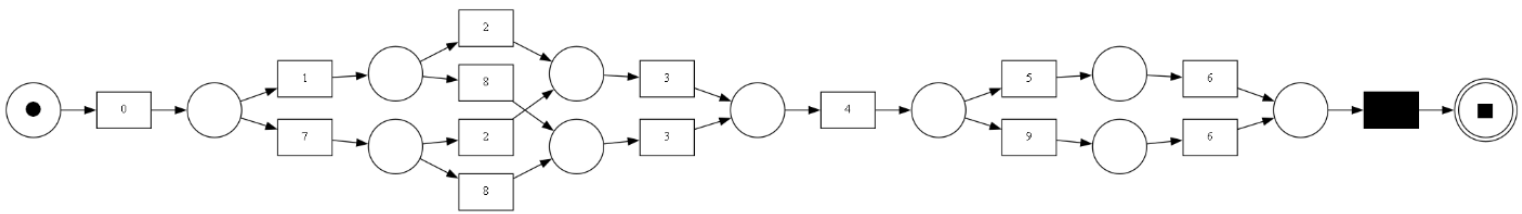


Figura 4: Discovered Workflow Net

Seção atual

Considerações Finais

Algumas melhorias a serem feitas seria a distribuição probalística dos tempos de produção e dos buffers, bem como análise de sua eficácia.

Outra possivel alteração seria a mudança do critério de parada da simulação, atualmente o critério de parada é o tempo de simulação, mas poderia ser alterado para a quantidade de tokens processados.

Alem disso, poderia ser implementado mais um fator, sendo este o *failure rate* de cada máquina, possibilitando assim um cenário mais próximo da realidade.

Repositório

Código Disponível no GitHub

<https://github.com/PedroOSilv/CLEMATIS-SIMPY-Simulation>


Gratidão!
Perguntas
e
Comentários