Trabajo Fin de Máster

Optimización dun modelo estatístico para a simulación de procesos

Xaime Suárez Blanco

Máster en Técnicas Estadísticas

Curso 2018-2019

Propuesta de Trabajo Fin de Máster

Título en galego: Optimización dun modelo estatístico para a simulación de procesos
Título en español: Optimización de un modelo estadístico para la simulación de procesos
English title: Optimization of a statistical model to simulate processes.
Modalidad: Modalidad B
Autor/a: Xaime Suárez Blanco, USC
Director/a: Javier Roca Pardiñas, UVigo;
Tutor/a: Moisés Vilar Vidal, Gradiant;
Breve resumen del trabajo: La minería de procesos permite generar, a partir de un registro de eventos, el modelo del proceso subyacente a esos datos de manera automática. En Gradiant hemos aumentado las capacidades de uno de los algoritmos clsicos de minería de procesos para obtener un modelo de simulación. El objetivo de esta práctica es mejorar este modelo aplicando los conocimientos estadísticos, de teoría de colas, teoría de juegos, modelos de regresión etc. que posea el candidato..
Recomendaciones: Regresión y clasificación. Simulación de eventos discretos.
Otras observaciones: Recibiremos las candidaturas interesadas en el siguiente correo electrónico: rrhh@gradiant.org Realizaremos entrevistas personales a los candidatos presentados. La realización del TFM llevará aparejada una compensación económica

Índice xeral

Resumo

Resumo en galego

No presente documento presentarese a ferramenta EasyProcess, un simulador do sistema de urxencias desenvolto por Gradiant que busca mellorar a calidade do mesmo. Para comprender o funcionamento deste simulador precisaremos realizar unha pequena introducción a minería de procesos para comprender o modelo e as representacións do mesmo. Tras esto realizaremos un achegamento os algoritmos de simulación que desenvolvín na estancia de prácticas que realizarán a simulación de entradas de doentes e os tempos de servizo. Por outra banda tamén presentaremos os estatísticos que desenvolvín e que no indicarán se un cambio no simulador deriva nunha mellora ou nun empeoramento do mesmo. Finalmente realizaremos algunhas conclusión sobre cales son as liñas de actuación no futuro para mellorar o simulador dende o meu punto de vista.

English abstract

In the present document, we will review the EasyProcess tool, a simulator of the emergency system developed by Gradiant, that try to improve the quality of this system. To understand the operation of this simulator, we need to make a small introduction to process mining to understand the model and its representations. After this, we will provide an explanation of the simulation algorithms that I developed as intern, so I will try to explain the simulation of patient entrances and service times. On the other hand I will also present the statistics that I have developed and which will indicate if a change in the simulator perform in an improvement or a deterioration of the same. Finally, we will make some conclusions about the lines of action in the future to improve the simulator.

Prefacio

Na actualidade a estatística convertiuse nunha poderosa ferramenta que nos axuda a comprender o mundo que nos rodea en todos os ámbitos da vida. Se ben no caso da saúde parece que a incorporación de estas técnicas está a ser máis difícil pola complexidade dos problemas que se plantean, cada vez máis profesionais danse conta da necesidade da utilización da estatística combinada coas ferramentas TIC para mellorar os servicios no día a día. Así nestes últimos anos o número de diferentes aplicación que combinan as técnicas de Big Data, minería de datos, minería de procesos cos métodos clásicos de estatística non paran de medrar, o que posibilita unha mellora na calidade dos servicios e na toma de decisións.

En concreto neste traballo se estudiará a aplicación EasyProcess, a cal é unha ferramenta que a partir de un conxunto de datos clínicos, optimizará un proceso clínico basándose na minería de procesos e na simulación de eventos discretos. A mellora da calidade dos servizos clínicos é un tema clásico, e existen traballos previos que utilizan exclusivamente a minería de procesos con este fin, entre os que podemos destacar os traballos de *Mans et al(2008)* ou *Kim et al(2013)*. No noso caso engadimos a dificultade da simulación, que axudará os profesionais a coñecer a calidade que se pode acadar cos recursos cos que contan ou a cuantificar esta calidade ao engadir certos recursos que poden ser modelados.

Máis precisamente EasyProcess céntrase no proceso de urxencias, un dos departamentos con maior importancia nos hospitais e que conta cunha gran complexidade pola gran cantidade de variables e parámetros que existen, ademais de pola autonomía que poseen os profesionais nos seus procederes. Polo tanto, neste traballo será necesario levar acabo unha pequena familiarización coa terminoloxía utilizada en urxencias e cos algoritmos de minería de procesos dos cales obteremos un proceso a partir do cal realizaremos as simulacións. Tras esta familiarización, trataremos de presentar o seu funcionamento e os indicadores que nos dirán se o facer algún cambio no modelo, este mellora ou empeora.

Desta maneira o proxecto dividirase en 5 capítulos, así comezaremos realizando unha pequena introdución explicando que obxectivos ten a implantación de esta ferramenta ademais de introducir os conceptos necesarios para entender os nosos conxuntos de datos cos que nos esncontramos en urxencias. No segundo capítulo explicaremos en que consiste a minería de datos de forma que introduciremos brevemente os principais conceptos e o algoritmo que utilizará a ferramenta para obter modelos que utilizaremos máis tarde para simular. No terceiro capítulo veremos cales son os algoritmo que segue a ferramenta para realziar a simulación de forma que nos centraremos en tres aspectos como é o algoritmo do fluxo, a simulación de entrada de doentes e a simulación dos tempos de espera de cada doente. No cuarto capítulo faremos unha enumeración e xustificación dos estatísticos que van a ser utilizados para evaluar a calidade do modelo e a nivel interno para saber se un cambio no simulador redunda en unha mellora na calidade do mesmo. Finalmente no último capítulo realizaremos algunhas conclusión sobre cales son as liñas de actuación no futuro para mellorar o simulador dende o meu punto de vista.

Capítulo 1

Introdución ao problema

Esta sección dividirémola en dúas subseccións onde nun primer apartado realizaremos un pequeno resumo sobre a ferramenta na que colaberei no seu desenvolvemento durante o período de prácticas en Gradiant, e en un segundo apartado realizaremos un achegamento a terminoloxía médica necesaria para entender os datos que manexaremos durante o traballo.

1.1. EasyProcess

EassyProcess é unha ferramenta de optimización de procesos centrada no ámbito clínico que non require de coñecementos técnicos para o seu uso. Esta ferramenta é accesible a través do navegador, aínda que neste momento ao estar en desenvolvemento só podemos utilizar unha demo da mesma con permiso de Gradiant.

O obxectivo primordial de EasyProcess será mellorar a calidade do proceso de urxencias a cal non é unha tarefa sinxela. Con tal premisa en mente realízase unha búsqueda dos principais problemas que se encontrarán durante o desenvolvemento da ferramenta, chegando a catro principais:

1. Cando un servizo non funciona todo o ben que esperamos, adoitamos negar que o problema está en que non traballamos da forma axeitada.
2. A pregunta crave non é cantos recursos debemos engadir ou reducir para mellorar o servizo, senón canta calidade podemos alcanzar cos recursos que temos.
3. Se queremos pensar desta maneira, debemos non centrarnos tanto nos datos (tempos de espera, número de recursos...) e empezar a fixarnos máis no proceso: como estamos a traballar?
4. Os procesos asistenciais son dos máis complicados que existen porque os profesionais posúen unha gran autonomía na súa proceder e existen infinidade de variables e parámetros a ter en conta.

Se ben os dous primeiros problemas son quizais alleos a aplicación, o tercer problema é un claro motivo para a utilización da minería de datos para o análisis de escenarios en vez de realizar un simple análisis de datos que se considera insuficiente neste caso. Por outro lado o cuarto problema seguramente sexa o máis difícil de solucionar, polo que este será o motivo de moitas simplicacións que se irán xustificando no desenvolvemento dos algoritmos no Capítulo 3. Co obxectivo optimizar o proceso de urxencias e tendo en conta as problemáticas anteriores, decidíronse seguir os seguintes pasos,

1. Entrevistar a unha serie de expertos sobre o proceso co obxectivo de obter todos os datos posibles sobre o mesmo.

2. Con eses datos, construír un modelo matemático que podamos analizar, compartir e do que sacar conclusións.

3. Utilizar algún software de simulación para probar o modelo matemático e comparar diferentes escenarios hipotéticos sobre o mesmo.

A continuación podemos ver un exemplo dos pasos do a seguir para utilizar esta ferramenta. En concreto podemos resumilos nos seguintes puntos,

1. **Cargar un arquivo**: Carga un simple arquivo Excel con eventos sobre o proceso. Un arquivo que calquera pode construír.

2. **Excutar a ferramenta**: Pulsar un botón para que EasyProcess realice todos os cálculos necesarios sobre o ficheiro Excel.

3. **Obter un modelo**: EasyProcess descubrirá de maneira automática o modelo por nós, con toda a información necesaria: tempos de servizo, puntos de decisión, distribucións...

4. **Analizar os resultados**: Podemos executar unha simulación ou realizar unha análise de inmediato, sen demoras, co consecuente aforro de esforzo, tempo e diñeiro.

1.2. Proceso de urxencias

Nesta sección introduciremos os principais conceptos necesarios para entender como funciona o sistema de urxencias e en definitiva os nosos conxuntos de datos. Temos que cada novo doente que entra no sistema de urxencias pasa por algunha destas actividades:

- **Admisión**: Todos os doentes deben pasar por admisión exceptuando os doentes que entran como críticos. Aquí se lles toman os datos ós doentes, se lles asigna un número e pasan as salas de espera de triaxe no caso dos adultos e a sala de pediatría no caso dos nenos.

- **Triaxe**: Esta consulta actúa como distribuidor de maneira que clasifica os doentes según o seu grado de urxencia, nelas as enfermeiras aplican un algoritmo que según os datos recollidos en admisión asignan un color ao doente que reflexa a gravidade deste e por tanto del dependerá o tempo de espera. Ademais deste color en triaxe tamén se lle asigna os doentes un dos destinos principais: COTO(Cirugía, Oftalmoloxía, Traumatoloxía, Otorrinolaringoloxía), Consultas ou Boxes.

- **Destinos principais**: Os destinos principais son ciruxía, oftalmoloxía, traumatoloxía, otorrinolaringoloxía, consulta e boxes. Neles os doentes comenzan a ser tratados según a necesidade que precisen. A principal diferencia entre Consultas e Boxes, radica en que os Boxes están diseñados para as persoas que se encontran en un estado máis grave. A partir de aquí os médicos poden dar a alta ou derivalos a outras zonas como poden ser os raios X ou realizar algunha proba de laboratorio.

- **Alta**: Xeralmente soe ser o último paso despois de facer todas as probas. Os médicos deciden se o doente debe ser ingresado no hospital ou pode volver a casa, ámbolos dous casos aparecen reflexados en urxencias como alta. É decir, se un médico decide ingresar a un doente no hospital este aínda segue non sistema hospitalario, sen embargo aparecerá como dado de alta no sistema de urxencias.

Comentar que existen diferentes tipos de triaxes, en *Bermejo et al(2013)* vemos que xeralmente os máis utlizados no estado Español son os Sistema de Triaxe Español(STE) e o Sistema de Triaxe Manchester(STM). No noso caso traballaremos co triaxe Manchester onde se clasifíca os doentes en cinco colores (Azul, Verde, Amarelo, Naranxa e Vermello), ordeado de menor a maior urxencia. Xeralmente

en urxencias tratase de que o reparto sea equitativo entre os equipos, é decir, se existen varios boxes e a un lle cae un doente triado como naranxa, a este boxes non lle vai a caer outro doente naranxa ata a cada un dos boxes restante lle caiga un. Tamén decir que si un doente permanece moito tempo na sala de espera ou empeoran os seus síntomas este pode ser retriado, cambiando a un color que indique máis gravidade. Para máis información sobre este tema podemos consultar *Souto-Ramos(2008)*.

A cuestión que nos tratará agora será como se ve reflexado este percorrido nos nosos datos. No noso caso traballaremos cunha estructura de datos comunmente chamada rexistro de eventos que definiremos matemáticamente no seguinte capítulo. A grande rasgos un rexistro de eventos é unha tabla formada cada fila por:

- Unha actividade, a tarefa que se executou no devandito evento (a admisión do paciente, o triaxe, ou alta etc.)
- Un identificador do caso ou instancia do proceso, é dicir, a quen ou a que se lle realizou certa tarefa (o paciente, o episodio etc.)
- Un selo temporal que indica cando se realizou dita tarefa.

Na táboa 1.1 podemos ver un exemplo dunha de estes conxunto de datos,

Caso	Actividade	Selo Temporal
1	a	2018-01-01 00:03:43
1	b	2018-01-01 00:11:16
1	c	2018-01-01 02:49:15
2	a	2018-01-01 00:04:48
2	d	2018-01-01 02:03:58
2	c	2018-01-01 03:27:55
3	a	2018-01-01 19:48:26
3	b	2018-01-02 12:10:09
3	c	2018-01-02 12:11:00

Cadro 1.1: Exemplo de rexistro de eventos de datos simulados

Para rematar esta sección realizaremos unha enumeración dos posibles nomes de actividades que nos podremos encontrar nos nosos conxuntos de datos. Decir que nestos datos non aparece o destino principal ao que vai cada doente nin as probas de laboratorio que se lle practican, perda de información que provocará un empeoramento no simulador e que está motivada pola falta de medios por parte do hospital na recollida de datos. As nosas variables serán:

- **Admisión**: Denota que un novo doente acaba de entrar no sistema de urxencias.
- **Triaxe**: Denota que doente foi triado según a gravidade. Nos nosos conxuntos de datos é habitual que apareza o color que indica a gravidade do mesmo.

- **Interconsulta**: Denota que un doente acaba de entrar en consultas, sen concretar en cal delas.
- **Resposta**: Denota que o doente obtuvo unha resposta da consulta anterior. Polo tanto ambas variables están ligadas.
- **RX Exploración**: Denota que a un doente está na sala de Raios X.
- **RX Informe**: Denota que un doente acábaselle de redactar o informe sobre a lesión detectada nos raios X.
- **RX Solicitude**: Denota que se realizou unha petición para que o doente entre na sala de raios X.
- **Alta**: Denota que o doente saíu do proceso de urxencias, ben por que se lle da a alta ou ben porque se ingresa no hospital.

En xeral esta nomenclatura non é xeralizables a todos os conxuntos de datos médicos pero si os que aparecerán no desenvolvemento de este traballo de fin de máster.

Capítulo 2

Minería de procesos

En este capítulo trataremos de facer una pequena introdución a minería de procesos necesaria para comprender o noso simulador. Por tanto trataremos de resumir os tipos de minería de procesos que existen, estableceremos as relacións que existen entre os modelos e a realidade, introduciremos a principal representación dos procesos e veremos que tipos de datos necesitamos para o noso fin. Para esta introdución seguiremos como libro de referencia *Van der Aalst(2011)*, a partir do cal explicaremos os conceptos necesarios para unha comprensión do problemas ao que nos enfrentamos.

Según *Van der Aalst(2011)* a minería de procesos é un novo medio que nos permite mellorar todo tipo de aplicacións. Estos métodos non paran de crecer nos úlitmo anos motivado polo aumento da capacidade de recolección de datos sobre eventos e a gran cantidade de detalles que obtemos deles. Se ben a minería de procesos surxiu na década de inicio de siglo, existiron certos precedentes como poden ser os traballos de *Khoussainov et al(1958)*, o cal presentou unha primeira pequena aproximación dos lenguaxes que sintetizaba as máquinas de estados finitos, tras esto *Petri(1962)* introduce o primer lenguaxe que é capaz de capturar as relacións entre actividades e *Gold(1967)* será o primeiro que comece a investigar sistemáticamente outros tipos de lenguaxe.

O obxectivo da minería de procesos será usar os datos de eventos para extraer información sobre certo proceso. Xeralmente vamos a poder distinguir tres tipos de minería de procesos distintas según o obxectivo que teñamos. A primeira sería a minería de datos orientada ó descubrimento, esta consiste en coller uns datos producido por un proceso sen ter ningunha información previa e tratar de obter unha representación do modelo a través de algún algoritmo. O segundo tipo de minería de proceso sería o que busca a conformidade, nela comparamos un proceso existente coñecido con un conxunto de datos do mesmo proceso. Esto pode ser utilizado para comparar se o que ocorre na realidade é veraz co que nos di o modelo, así esta parte da minería de datos soe estar orientada hacia o descubrimento de fraudes. Por útlimo, o terceiro tipo de minería de datos está orientada hacia a mellora, aquí a idea é tratar de mellorar unha aproximación do proceso utilizando a información recollida nun conxunto de datos. A diferenza principal entre a anterior e esta é que aquí o obxectivo é cambiar o modelo e non comprobar se este funciona correctamente.

Na minería de procesos é clave tratar de enfatizar a relación que existe entre a realidade e o modelo. Neste contexto surxen os conceptos de Play-in, Play-out e Replay que reflexan estas relacións. Play-out é a concepción máis clásica da minería de datos, así o Play-out estará basease en a partir de un modelo obter un conxunto de datos que nos pode servir por exemplo como predicción do futuro ou para analizar o presente. Por outro lado Play-in é a concepción inversa, trataremos de obter un modelo a partir de un conxunto de datos, soe estar referenciado por exemplo a inferencia. A idea do Play-in é que aplicar un algoritmo a un conxunto de datos a partir do cal podamos obter unha representación do modelo que nos axude a comprender que ocorre na realidade. Finalmente referímonos a Replays

ao feito de tendo un modelo e un conxunto de datos coñecidos, trataremos de executar estos datos a través do modelo con diferentes obxectivos:

1. Comprobar o modelo: O obxectivo é ver como de parecido é o modelo a realidade e ver por exemplo si as trazas que xenera o modelo son parecidas as obtidas nos datos.

2. Engadir o modelo frecuencias ou información temporal: Utilizando o replay podremos tratar ver con que frecuencia aparece cada traza ou ver en que parte do modelo se perde máis tempo detectando así os coñecidos como pezcozos de botella.

3. Axuda operacional: A idea aquí sería tratar de identificar trazas que sean desviacións do que normalmente ocorre, xerandose así unha alerta de que algo pode ir indo mal.

Como vemos para calquera dos tres tipos de minería de datos vamos a necesitar ou ben unha representación de algún tipo do modelo ou uns datos a partir dos cales aplicaremos as técnicas. Se ben en canto as representacións dos modelos existen unha gran variedade según a disciplina onde nos encontremos, no noso caso centrarémonos nas Petri Nets. Estas serán as que darán como resultado o algoritmo que presentaremos máis adiante. Por outro lado os datos utilizados en esta modalidade son coñecidas como rexistros de eventos, os cales requeriran de un formato mínimo para que nos sean útiles no noso propósito.

Como mencionamos no capítulo anterior os datos que nos recollen como se levou a cabo o proceso coñécense como rexistros de eventos. Nel aparecerán como mínimo recollidos unha actividades ou tarefa, un identificador do caso e un selo temporal. Para definir este concepto formalmente será preciso definir previamente distintos conceptos. Para esto comezaremos definindo a traza a cal simplemente é o conxunto de actividades ordeadas que teñen o mesmo identificador.

Definición 2.1 *(Traza) Sexa A o conxunto de todas as actividades, unha traza $t \in A^*$ é unha secuencia de actividades. Notaremos por $T = A^*$ e $|t|$ a lonxitude da traza.*

Por exemplo, no Cadro 1.1 podemos ver para o primeiro identificador teríamos a traza $t =< a, b, c > \in T$ que está formada polas actividades $\{a, b, c\}$ e ademais terase que $|t| = 3$ pois ten 3 actividades. Estas actividades aparecerán reflexadas no rexistro de eventos onde ademais cada actividade ten asociado un tempo.

Definición 2.2 *(Rexistro de Evento) Sexa $L = B(T)$ o conxunto dos rexistros de eventos. Un rexistro de evento $l \in L$ é un multiconxunto de trazas observadas. Ademais tense que $\hat{l} = \{t \in l\}$ é o conxunto de trazas que aparecen en $l \in L$ e denotaremos por $|l|$ o número de trazas que hai no rexistro de eventos e por $l(t)$ o número de trazas t que hai no rexistro de eventos.*

Podemos exemplificar esto a partir do rexistro de eventos recollido no Cadro 1.1, onde se tería que as trazas serían $l = | < a, b, c >^2, < a, d, c >^1 |$ polo que $|l| = 3$, e de forma que si $t_1 =< a, b, c >$ se tería que $l(t_1) = l(< a, b, c >) = 2$.

A partir de estes tipo de datos deberemos construír modelos a partir dos cales tratemos de explicar que proceso se encontra detrás deles, de ahí a importancia destos rexistros de eventos.

Na actualidade a aparición dunha gran variedade de distitas notacións para representar os procesos é unha boa ilustración da importancia que está a coller a minería de procesos. Así para esta parte necesitaremos asumir que existe un conxunto de actividades A, e o obxectivo será decidir que actividades necesitan ser executadas e en que orde. No noso caso vamos a centrarnos nas Petri Nets as cales son o mellor e o máis vello lenguaxe para representar procesos. A pesar de que a notación das Petri Nets é sinxela e intuitiva, estas son moi útiles para analizar a realidade.

Unha Petri Net é un gráfico bipartito que consta de lugares (representada por cículos) e transicións (representada por cuadrados) que están unidas por arcos que van de un lugar a unha transición ou vice versa como podemos ver na Figura 2.1. Matemáticamente esto pode formalizarse como,

Definición 2.3 *(Petri Net) Unha Petri Net é unha tupla $N = (P, T, F)$ onde P é un conxunto finito de lugares, T é un conxunto finito de transicións de forma que $P \cap T = \emptyset$, e $F \subseteq (P \times T) \cup (T \times P)$ é o conxunto dos arcos directos chamados relación de transición.*

As Petri Nets son gráficos estáticos, non van variando ao largo do tempo, polo que necesitarán añadirlle algo para que sexan útiles para representar o fluxo dun proceso. Con esta intención as Petri Nets engádenselle un número discreto de "puntos"chamados *marcas*, de forma que se realizará unha transición na Petri Net cando se execute (se dispara) unha actividade e unha destas marcas estexa dispoñibles na posición de inicio. É decir, se hai suficiente marcas nun lugar de saída entón terá lugar a transición consumindo unha marca no lugar de partida e creando unha marca no lugar de chegada. A distribución de estas marcas sobre a Petri Net representará unha configuración desta. A este tipo de rede nótase por Petri Net marcada cuxa definición formal podemos ver a continuación.

Definición 2.4 *Unha Petri Net marcada é un par (N, M), onde $N = (P, T, F)$ é unha Petri Net e onde $M \in B(P)$ é un conxunto múltiple sobre P denotando as marcas sobre a rede. O conxunto de marcas sobre a Petri Net denótase por $\aleph$.*

Na Figura 2.1 podemos ver un exemplo de representación do proceso do rexistro de evento do Cadro 1.1,

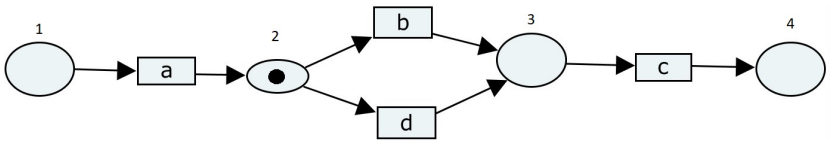

Figura 2.1: Exemplo de Petri Net dun proceso simulado

Seguindo a notación introducida nas definición teríamos os seguintes conxuntos para o proceso da Figura 2.1,

1. P={1,2,3,4}
2. T={a,b,c,d}
3. F={(1,a),(a,2), (2,b), (2,d), (b,3), (d,3), (3,c),(c,4)}

Valéndonos da Figura 2.1 podemos ver tamén como funcionan as marcas. Estas marcas como comentamos antes son as que permiten o funcionamento dinámico da Petri Net, de forma de que unha transición estará dispoñible sempre que exista unha marca na casilla de saída. Así por exemplo no noso caso temos que a transición b estaría dispoñible. Así se por exemplo se activase a transición b, comunmente denótase por disparar a transición, mandaríase a marca do lugar 2 ao lugar 3. Para formalizar estas reglas de transición deberemos introducir certa notación para os lugares (transicións) de entradas (saídas).

Definición 2.5 *Sexa $N = (P, T, F)$ unha Petri Net, chamaremos nodos ao conxunto $P \cup T$. Un nodo x será un nodo de entrada de outro nodo y se e solo se existe un arco directo de x a y, é decir, $(x, y) \in F$. O nodo x será un nodo de saída de y se e solo se $(y, x) \in F$. Para todo $x \in P \cup T$, tense que $\bullet x = \{y | (y, x) \in F\}$ e $x\bullet = \{y | (x, y) \in F\}$.*

Neste punto xa podemos introducir formalmente o que se coñece popurlamente como disparar unha marca.

Definición 2.6 *(Reglas de transición) Sexa $(N.M)$ unha marked Petri Net con $N = (P, T, F)$ e $M \in B(P)$. Unha transición $t \in T$ está permitida e denotaremola por $(N, M)[t]$, se e solo se $\bullet t \leq M$.*

Como podemo ver na Definición 2.6 utilizaremos a notación $(N, M)[t]$ que denota que t está permitida en M, por exmplo no noso caso teríamos que $(N, 2)[b]$.

Finalmente introduciremos a modificación da Petri Net máis utilizada que será a que nos de como resultado o noso algoritmo que se coñece como Causal Petri Net. En definitiva a idea desta C-Net é transformar a representación gráfica que é a Petri Net nunha tabla a partir da cal podramos calcular a Petri Net asociada e vice versa. Ademáis introduciremos o concepto de Gráfico de Dependencia, asociado a estas C-nets e que tamén precisaremos para o noso algoritmo.

Definición 2.7 *(Causal Petri Nets ou C-net) Unha Causal Petri Net é unha tupla $N = (T, I, O)$ onde,*

1. *T é un conxunto finito de actividades.*
2. *$I : T \to P(P(T))$ é a función de entradas.*
3. *$O : T \to P(P(T))$ é a función de saídas.*

Se $e \in T$ entón $\bullet e = \cup I(e)$ denota as actividades de entrada en e $e\bullet = \cup O(e)$ denota as actividades de saída de e. Temos ademáis que $P(X)$ denota o conxunto por partes de X.

Definición 2.8 *(Gráfico de Dependencia(DG)) Se (T, I, O) é unha C-net entón o gráfico de depencia correspondente DG é a relación en $T(DG \subseteq T \times T)$ con,*

$$DG = \{(a, b) | (a \in T \wedge b \in a\bullet) \vee (b \in T \wedge a \in \bullet b)\}$$

Como poidemos comprobar unha representación tan sinxela como as Petri Nets ten unha base teórica moi forte por detrás. Na literatura é habitual encontrarse con variacións destas Petri Nets capaces de captar aspectos relativos os datos ou as marcas temporais das actvidades. Así por exemplo en ocasión aparecen Petri Nets onde cada transición ten asociado un selo temporal que consiste no mínimo tempo para que se consuma unha marca nesa transición. Para o noso simulador simplemente nos quedaremos cunha Causal Petri Net, que será suficiente para o noso propósito.

Finalmente, para rematar esta sección trataremos de profundizar un pouco máis nas transicións para reforzar algunha das nomenclatura poden aparecer máis adiante. Se ben esta nomenclatura non sería necesaria xa que as propias Petri Nets son capaces de disntinguir os distintos tipos de transicións, a súa utilización é xeneralizada pola súa comodidade. Esta nomenclatura aparece por primeira vez no lenguaxe YAWL que significa "Yet Another Workflow Language", neste lenguaxe temos que a chegada ou a saída para cada transición pode ser de distintos de tipos e según este tipo utilizaremos unha ou outra nomenclatura. Para diferenciar a entrada e a saída de cada un dos lugares utilizanse os termos join e split respectivamente, así por exemplo na Figura 2.1 teremos que a transición c é unha transición join e a transición a é unha transición split.

Ademais según como sean consumidas as marcas nos encontramos ante varios tipos de transicións. Así por exemplo na Figura 2.2 temos que para que se leve a cabo transición c deberá haber necesariamente unha marca nos lugares 4 e 5, este tipo de transicións coñecense como transicións AND-join. Esto podémolo ver como que para o AND-join fai falla unha sincronización que precisa que todos as marcas cheguen aos lugares anteriores. En cambio na Figura 2.1 vemos que para que se execute c só se necesita que se consuma unha marca no lugar 3 polo que nos encontramos entón ante un XOR-join.

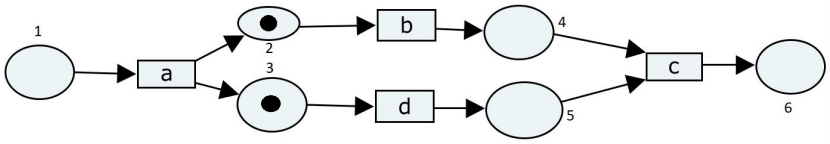

Figura 2.2: Exemplo de Petri Net dun proceso simulado

Análogamente esta distinción faise para o caso dos splits, así por exemplo a transición a da Figura 2.2 unha vez disparada creará unha marca nas posición 2 e 3, polo que despois podranse executar ambas actividades e polo tanto estamos ante un AND-split. Sen embargo a transición a da Figura 2.1 só crea unha única marca polo que despois só se podrá executar unha das dúas actividades e polo tanto estamos ante un XOR-split.

2.1. Algoritmo de minado

Nesta sección presentaremos o algortimo de minado utilizado na construción de EasyProcess. Na actualidade existen gran variedade de algoritmos de minado que evolucionaron a partir do primeiro proposto por *Agrawal et al(1998)*. Esta gran variedade podémola contrastar por exemplo *Yue et al(2011)*, o cal realiza un interesante recopilación dos algoritmos actuales. En el se comenza cunha enumeración dos principais algoritmos coas súas características para acabar enumerando as melloras levadas a cabo nestes algoritmos nos últimos anos. Por outro lado en artículos como *Weber et al(2013)*, *Wang et al(2012)*, *Tiwari el al(2008)* ou en *De Weerdt et al(2012)* se realizan diversos estudios de comparación de algoritmos e se propoñen además algúns índices para levar a cabo estás comparacións.

Nestos artículos podemos ver as comparacións entre os principais algoritmos como son α algoritmo, $\alpha^{\#}$ algoritmo, α^{++} algoritmo, algoritmo genético, algoritmo Heuristic, algoritmo Fuzzy, algoritmo Inductivo ou o algoritmo Flexible Heuristic. No noso caso EasyProcess está basado no algoritmo Flexible Heuristic Miner(FHM), este algoritmo aparece por primeira vez en *Weijters e Ribeiro(2011)* e a súa explicación estará basada neste artigo.

En primeiro lugar, para a contrución do modelo a través do algoritmo necesitaremos analizar cales son as dependencias causales, é decir, analizar se unha actividade X é moi probable que lle siga unha actividade Y entón diremos que existe unha relación de dependecia entre estas dúas tarefas. Para obter estas relacións axudaremonos dos gráficos de dependencia, que para a súa comprensión precisaremos antes introducir notación e uns conceptos previos.

Definición 2.9 *Sexa T un conxunto de actividades, $\delta \in T^*$ unha traza do proceso, $W : T \to N$ é un rexistro de eventos do proceso e $a, b \in T$:*

1. *$a >_W b$ se e só se existe unha traza $\delta = t_1 t_2 ... t_n$, e un $i \in \{1, .., n-1\}$ tal que $\delta \in W$ e $t_i = a$ e $t_{i+1} = b$ (sucesor directo).*

2. *$a >>_W b$ se e só se existe unha traza $\delta = t_1 t_2 ... t_n$, e un $i \in \{1, .., n-2\}$ tal que $\delta \in W$ e $t_i = t_{i+2} = a$ e $t_{i+1} = b$ e $a \neq b$ (bucles de lonxitude dous).*

3. *$a >>>_W b$ se e só se existe unha traza $\delta = t_1 t_2 ... t_n$, e $i < j$ un $i, j \in 1, .., n$ tal que $\delta \in W$ e $t_i = a$ e $t_j = b$ (succesor directo ou indirecto).*

Para a construción do gráfico de dependencia, necesitaremos introducir unha métrica basada en frecuencias que nos mida a relación entre dous eventos. Esta métrica están basadas no conteo das relacións presentadas na definición anterior.

Definición 2.10 *(Medidas de dependencia) Sexa W un rexistro de eventos sobre T, $a, b \in T$, $|a >_W b|$ o número de veces que ocorre que $a >_W b$ en W, e $|a >>_W b|$ o número de veces que ocorre que $a >>_W b$ en W.*

$$a \Rightarrow_W b = \left(\frac{|a >_W b| - |b >_W a|}{|a >_W b| + |b >_W a| + 1} \right) \quad \textbf{se} \quad a \neq b$$

$$a \Rightarrow_W a = \left(\frac{|a >_W a|}{|a >_W a| + 1} \right)$$

$$a \Rightarrow^2_W b = \left(\frac{|a >>_W b| - |b >>_W a|}{|a >>_W b| + |b >>_W a| + 1} \right) \quad \textbf{se} \quad a \neq b$$

En primeiro lugar vemos que esta medida toma valores entre -1 e 1. Por outro lado vemos que estas medidas indicarán unha relación máis forte de dependencia canto máis cernano estexa o valor de 1. Estas medidas podémolas utilizar para decidir se existe unha relación entre dúas actividades, así podemos fixar uns límites de forma que se as medidas as superan concluímos que existe relación entre ambas. Esta é unha das ideas fundamentais para a construcción do flexible heuristic miner, onde as relacións de dependencia virán dadas sempre que se superen tres límites coñecidos como: Dependecy threshold, Length-one loop threshold e Length-two loop threshold. Polo tanto si nos interesa que non haxa bucles de lonxitude un, simplemente debemos fixar o Length-one loop threshold=1.

Sen embargo estos límites tamén poden ser contraproducentes se eliminan relacións que deberían de existir, así veremos que este algoritmo é capaz de detectar relacións sin estar totalmente influenciado polos límites. A idea consiste en que exceptuando a tarefa inicial e a final todas van a como mínimo un precedente e unha posterior. Ademais utilizaremos outro límite de forma que non só se aceptarán relacións entre actividades que superan os límites anteriores, se non tamén que teremos en conta as actividades cuxa diferencia entre a súa medida e a medida de dependencia mellor sea inferior a este outro límite. Finalmente debemos comentar que por razóns prácticas o algoritmo creará dúas actividades ficticias para o inicio e o final para evitar problemas cando hai "ruído"no rexistro de eventos, que formalizamos a continuación.

Definición 2.11 *(Extensión no inicio e fin) Sexa W un rexistro de eventos sobre T, definimos o rexistro de evento artificial W^+ conocido como start/end-extesion sobre T^+ como*

1. $T^+ = T \cup \{start, end\}$
2. $W^+ = \{start\delta end \mid \delta \in W\}$

En este instante xa temos todo o necesario para a introducir a primeira parte do algoritmo basada na construción do gráfico de dependencia.

Paso 1. Construción do Gráfico de Dependencias(DG)

Definición 2.12 *(Dependency Graph-algorithm): Sexa W un rexistro de eventos sobre T, W^+ un rexistro de eventos sobre T^+, σ_α o límite de dependencia, σ_{L1L} o límite para bucles de tamaño un, σ_{L2L} o límite para bucles de tamaño 2 e σ_r o límite Relative-to-best. Se define o gráfico de dependencia $DG(W^+)$.*

1. $T = \{t \mid \exists \sigma \in W^+ \setminus t \in \sigma\}$ *(conxuntos de actividades que aparecen no rexistro)*
2. $C_1 = \{(a, a) \in T \times T \mid a \Rightarrow_W a \geq \sigma_{L1L}\}$ *(bucles de lonxitude un)*
3. $C_2 = \{(a, b) \in T \times T \mid (a, a) \notin C_1 \wedge (b, b) \notin C_1 \wedge a \Rightarrow^2_W b \geq \sigma_{L2L}\}$ *(bucles de lonxitude dous)*
4. $C_{out} = \{(a, b) \in T \times T \mid b \neq End \wedge a \neq b \wedge (\forall y \in T \ (a \Rightarrow_W b) \geq (a \Rightarrow_W y))\}$ *(seguidor máis forte)*

5. $C_{in} = \{(a,b) \in T \times T \mid\ a \neq Start \wedge a \neq b \wedge (\forall x \in T\ \ (a \Rightarrow_W b) \geq (x \Rightarrow_W b))\}$ *(causa máis forte)*
6. $C'_{out} = \{(a,x) \in C_{out} \mid\ (a \Rightarrow_W x) < \sigma_\alpha \wedge \big(\exists (b,y) \in C_{out} \setminus \big[(a,b) \in C_2 \wedge (b \Rightarrow_W y) - (a \Rightarrow_W x) \geq \sigma_r\big]\big)\}$ *(eliminamos as saídas máis débiles dos bucles de lonxitude dous)*
7. $C_{out} = C_{out} - C'_{out}$ *(eliminamos as conexións máis débiles)*
8. $C'_{in} = \{(x,a) \in C_{in} \mid\ (x \Rightarrow_W a) < \sigma_\alpha \wedge \big(\exists (y,b) \in C_{in} \setminus \big[(a,b) \in C_2 \wedge (y \Rightarrow_W b) - (x \Rightarrow_W a) \geq \sigma_r\big]\big)\}$ *(eliminamos as entradas máis débiles para os bucles de lonxitude dous)*
9. $C_{in} = C_{in} - C'_{in}$ *(eliminamos as conexións máis débiles)*
10. $C''_{out} = \{(a,b) \in T \times T \mid\ (a \Rightarrow_W b) \geq \sigma_\alpha \vee \exists (a,b) \in C_{out} \setminus (a \Rightarrow_W c) - (a \Rightarrow_W b) < \sigma_r\}$
11. $C''_{in} = \{(b,a) \in T \times T \mid\ (b \Rightarrow_W a) \geq \sigma_\alpha \vee \exists (b,c) \in C_{in} \setminus (b \Rightarrow_W c) - (b \Rightarrow_W a) < \sigma_r\}$
12. $DG = C_1 \cup C_2 \cup C''_{out} \cup C''_{in}$

Con todos estos pares temos toda a información necesaria sobre as relacións de dependencia entre as actividades. Xeralmente unha vez calculados todos estos pares podémolos presentar nunha tabla onde para cada actividade dispóñense as causas e seguidores según os pares que temos en DG. Vexamos unha ilustración do cálculo seguindo o exemplo presentado en *Weijters e Ribeiro(2011)*. Para elo supoñamos que temos un rexistro de eventos onde as frecuencias de dependencia directa sexan as do Cadro 2.1 e as frecuencias dos bucles de orden dous estexan no Cadro 2.2, estos cadros de exemplo tamén aparecen en *Weijters e Ribeiro(2011)*.

	Start	A	B	C	D	E	F	G	H	I	J	K	L	End
Start	0	1000	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	520	480	0	0	0	0	0	0	0	0	0	0
B	0	0	0	360	182	198	0	0	0	323	27	0	0	0
C	0	0	338	0	125	128	40	48	8	349	0	0	0	0
D	0	0	0	63	0	0	586	0	0	193	68	5	6	0
E	0	0	0	73	0	0	0	619	0	236	67	0	3	0
F	0	0	0	16	124	134	0	0	327	212	88	0	7	0
G	0	0	0	16	143	145	0	0	359	220	105	0	10	0
H	0	0	0	11	0	0	0	0	0	252	105	614	5	0
I	0	0	119	0	209	236	179	210	166	315	576	0	0	0
J	0	0	23	0	135	155	102	117	118	0	0	381	5	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	1000
L	0	0	0	17	3	2	1	4	9	0	0	0	0	0
End	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	1000	10000	1000	1036	921	998	987	2010	1036	1000	1036	1000	36	1000

Cadro 2.1: Frecuencia dos sucesores directos

	Start	A	B	C	D	E	F	G	H	I	J	K	L	End
Start	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	89	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	104	0	0	0	0	0	0
F	0	0	0	0	110	0	0	0	0	0	0	0	0	0
G	0	0	0	133	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	19	0	40	63	59	57	97	116	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0		0	0	0	0	0	0

Cadro 2.2: Frecuencia dos búcles de orden 2

Por tanto os pasos a seguir para a construcción do grafo de dependencia serían:

1. Primeiro constrímos o conxunto coas actvidades. No noso caso $T = \{A, B, C, D, E, F, G, H, I, J, K, L\}$.

2. Construímos o conxunto cos búcles de tamaño un. Neste caso o único candidato é I que como vemos no Cadro 2.1 está seguido 315 veces por el mesmo. Polo que $C_1 = \{(I, I)\}$.

3. Para este paso usaremos o Cadro 2.2, nel vemos que DFD aparece 89 veces e que FDF aparece 110 veces entón $D \Rightarrow_W^2 F = (89 + 100)/(89 + 110 + 1) = 0{,}995$ e como $D \notin C_1$ e $F \notin C_1$ polo que $(D, F), (F, D) \in C_2$. Cos mesmos razoamentos chegamos a que $C_2 = \{(F, D), (D, F), (E, G), (G, E)\}$.

4. Para construír C_{out} calculamos a paratir de Cadro 2.1 os seguidores máis fortes fixándonos por fila. Por exemplo para a tarea C o valor máis alto sería 0.997 correspondente a I polo que $(C, I) \in C_{out}$.

5. De forma análoga ahora fixándonos por columnas calculamos C_{in}. Por exemplo para a actividade K o valor maior será 0.998 entón $(H, K) \in C_{in}$.

6,7. Tomemos como exemplo a actividade D e F. Temos que $(D, F) \in C_2$. A saída máis forte de D é K(0.833) por enriba de F e a saída de F é H(0.997). Por esta razón $(D, K) \in C'_{out}$ e por eso eliminámolo de C_{out}.

8,9. Son análogos a 6,7.

10,11. Engadimos novas conexión se se superan o límite Relative-to-best. Por exemplo sería o caso de J a L onde se aceptaría pois a diferencia supera o límite.

12. Xuntamos todos os grupos creados antes e obtemos o gráfico de dependencia que pode ver no Cadro 2.3.

Entrada	Actividade	Saída
{}	A	{B,C}
{A}	B	{D,E}
{A,L}	C	{I}
{B,F,G}	D	{F}
{B,F,G}	E	{G}
{D}	F	{D,E,H}
{E}	G	{D,E,H}
{F,G}	H	{K}
{C,I}	I	{I,J}
{I}	J	{K,L}
{J,H,}	K	{}
{J}	L	{C}

Cadro 2.3: Diagrama del algoritmo de fluxo real

Como vemos nesta tabla aínda non sabemos de que tipo son os splits/joins polo que será a seguinte parte do argumento.

Paso 2. Minado dos Splits/Joins

Esta parte do algortimo tratará de buscar de que tipo son as conexións joins/splits. Para continuar con esta parte do algoritmo necesitaremos definir o que é unha augmented-C-net, que nos permitirá indicar o número de veces que aparece cada tipo de split/joins.

Definición 2.13 *(Augmented C-net): An augmented-C-net é unha tupla (T,I,O) onde*

1. *T é un conxunto finito de actividades.*
2. *$I : T \to P(P(T) \to N)$ é a función de frecuencias de saída.*
3. *$O : T \to P(P(T) \to N)$ é a función de frecuencias de entradas.*

A partir do Cadro 2.3 veremos un exemplo de cal é a idea básica, collamos por exemplo A e vemos que as posibles saídas son $\{B, C\}$. Sen embargo agora queremos saber se A sempre é seguido por C e B (AND-split), sólo por A ou C (XOR-split) ou a maioría de veces só por unha pero a veces por ambas (OR-split). Así por exemplo se no noso rexistro de eventos aparece que a actividade A sempre está seguida por B e C, digamos 1000 veces, entonces coa nosa augmented-C-net aparecería indicado como $O(A) = \{\{B, C\}^{1000}\}$. Se por exemplo estuvese seguido por un deles cada vez teríamos por

exemplo $O(A) = \{\{B\}^{520}\{C\}^{480}\}$, esto sería por tanto un exemplo de XOR-split.

Expliquemos esto en máis profundidade, collamos a actividade A e unha traza de exemplo ABDCIFIJEGHK. Valéndonos do Cadro 2.3 temos que B e C poideron ser activados por A ao ser antecedente de ambos, e como nesta traza A é o candidato a activalos máis cercano entón actualizamos a información de A co patrón $\{B, C\}$, é decir, $O(A) = O(A) \cup \left[\{B, C\}\right]$ co que aumentaría unha unidade a frecuencia co que aparce $\{B, C\}$. Loxicamente nos podemos encontrar con situacións máis complexas, por exemplo tomando a mesma traza a partir do Cadro 2.3 vemos que os seguidores de $B = \{D, E\}$ e os antecedentes destes son $E = D = \{B, F, G\}$. Para D nesta traza o único candidato posible é B, sen embargo para E temos a F e a B como candidatos, nestes casos elixiremos como o activador de E o cándidato máis cercano na traza F. Polo tanto agora podemos actualizar a información e obteríamos $O(B) = O(B) \cup \left[\{D\}\right]$, aumentando unha unidade a frecuencia de $\{D\}$ despois de B.

Entrada	Actividade	Saída
$\{\}$	A	$\{\{B, C\}^{1000}\}$
$\{\{A\}^{1000}\}$	B	$\{D\}^{467}, \{E\}^{533}$
$\{\{A\}^{1000}, \{L\}^{36}\}$	C	$\{\{I\}^{1036}\}$
$\{\{B\}^{467}, \{F\}^{222}, \{G\}^{232}\}$	D	$\{\{F\}^{908}\}$
$\{\{B\}^{533}, \{F\}^{215}, \{G\}^{250}\}$	E	$\{\{G\}^{998}\}$
$\{\{D\}^{908}\}$	F	$\{\{D\}^{222}, \{E\}^{215}, \{H\}^{471}\}$
$\{\{E\}^{998}\}$	G	$\{\{D\}^{232}, \{E\}^{250}, \{H\}^{516}\}$
$\{\{F\}^{471}, \{G\}^{516}\}$	H	$\{\{K\}^{987}\}$
$\{\{C\}^{1036}\{, I\}^{974}\}$	I	$\{\{I\}^{974}, \{J\}^{1036}\}$
$\{\{I\}^{1036}\}$	J	$\{\{K\}^{1000}, \{L\}^{36}\}$
$\{\{J, H\}987, \{J, D\}^{13}\}$	K	$\{\}$
$\{\{J\}^{36}\}$	L	$\{\{C\}^{36}\}$

Cadro 2.4: Diagrama del algoritmo de fluxo real

Paso 3. Minado da dependencia en distancia longas

Este é o paso final do algortimo que está diseñado para identificar as relacións na que unha actividade X depende indirectamente doutra Y. É decir, neste paso o algoritmo detectará que para execución de X é necesario que aparezca a actividade Y en algún momento non necesariamente cercano. Para ter en conta estas dependencias é necesario definir unha nova métrica que teña en conta estas relacións. A idea principal consiste en buscar pares de actividades con frecuencias similares nos cales a primeira actividade estexa directa ou indirectamente seguida pola segunda.

Definición 2.14 *(Long distance dependency measure): Sexa W un rexistro de eventos sobre T, $a, b \in T$, $|a >>>_W b|$ o número de veces que ocorre $a >>>_W b$ en W, e $|a|$ o número de veces que ocorre a en W.*

$$a \Rightarrow^{l}_{W} b = \left(\frac{2(|a >>>_W b|)}{|a| + |b| + 1}\right) - \left(\frac{2Abs(|a| - |b|)}{|a| + |b| + 1}\right)$$

Como vemos un valor cercano a 1 da primeira parte da expresión indícanos que a tarefa a está sempre seguida por b. Un valor cercano a 0 da segunda expresión indícanos que a frecuencia coa que aparece a e b é parecida. É decir, un valor total cercano a 1 significa que a tarefa a sempre está seguida por b e que a frecuencia coa que aparecen ambas é parecida. De forma análoga ao feito con outras medidas, iremos engandindo a DG estas dependencias sempre que se verifique $a \Rightarrow^{l}_{W} b \geq \sigma_l$ onde σ_l se

denomina o límite de distancia larga. Ao final deste paso necesitaremos volver a calcular a información relacionada cos split/join.

Como vemos ó final destes pasos obteríamos unha C-net, que poderíamos representala en forma de Petri Net e que podríamos utilizar como simulador mediante o sistema de marcas explicada na anterior sección. Sen embargo solamente esto non é suficiente pois utilizando as C-net non sabemos cada canto tempo entra unha marca nova no sistema ou cada canto tempo actívase unha das transicións se hai as marcas necesarias. Para suplir estas dificultades diseñouse un sistema de simulación que explicaremos no seguinte capítulo.

Capítulo 3

Modelo de simulación

Na capítulo anterior introducimos os conceptos necesarios para obter unha Causal Petri Net a partir de un rexistro de eventos utilizando o algoritmo de minado Flexible Heuristic Miner, además gracias aos conceptos introducidos sobre minería de datos somos capaces de comprender esta Causal Petri Net que transformaremos en unha Petri Net para levar a cabo a simulación utilizando o sistema de marcas. Sen embargo este sistema por si sólo no vai a ser suficiente para realizar as simulación pois teremos que xerar as marcas nos distintos sitios por algún método e dos mesmo xeito as actividades deben ser activadas según algún criterio e non de forma aleatoria. Para resolver todas estas cuestión dividiremos este capítulos en tres partes onde no primeiro nos centraremos en explicar como se simulará o fluxo a través da Petri Net, o segundo se basará na simulación das entradas no sistema e finalmente aproximarémonos aos métodos polos que simulamos as activacións das distintas actividades.

Antes de decantarse por este tipo de modelo de simulación en Gradiant se estudiaron outros métodos cuxo estado do arte está máis avanzado como a teoría de colas. Sen embargo os datos cos que habitualmente se vai a contar nos van a impedir utilizar estas técnicas, esto é debido a que nos rexistros de eventos non aparece reflexado o tempo de espera que nos serviría para modelar as colas, se non que para cada actividade temos acumulado o tempo de espera e o de consulta. Este non é o único problema xa que o sistema de urxencias comparte certos recursos como os raios X co hospital, polo que non podríamos modelar as entradas en esta actividade por que non contamos cos datos de entrada a través do hospital.

3.1. Algoritmo de fluxo

Nesta sección estudiaremos como actúa o simulador, é decir, nos centraremos en como o simulador xerará os rexistros de eventos a partir da Petri Net calculada seguindo o algoritmo da sección anterior. Para a explicación deste algoritmo nos apoiaremos na Figura 3.1. Como vemos nela aparecen as palabras CE e RE, o CE é a cola de eventos nela almacenaremos os eventos coa correspondente marca de tempo e número de caso para os eventos que aínda non se executaron. Por outro lado en RE é o rexistro de eventos e nela están gardados os eventos de novo coa marca de tempo e o número de caso que xa foron executados. Polo tanto temos que CE actúa como unha cola de eventos futuros e RE actuará como un rexistro do executado.

Comecemos por tanto a facer unha pequena descrición do algoritmo de fluxo a partir da enumeración dos pasos a seguir. Debemos reseñar que cando comezamos tanto CE como RE están baldeiros e o tempo do simulador t está fixado en 0. Como veremos neste algoritmo non se especificará como son simuladas as entradas ou os distintos tempos que aparecen no algoritmo, estos serán especificados nas seguintes seccións. Por tanto os pasos a seguir son os seguintes:

1. Se o tempo do simulador t é maior que o tempo fixado de simulación T entón rematamos a simulación, en caso contrario existirán eventos por simular polo que pasamos o seguinte punto de decisión.

2. Se en CE non hai almacenado ningunha entrada entón simularíase unha nova entrada que terá asociado un tempo e un número de caso que se engadirá en CE ordeado polo tempo. En caso de que teñamos xa unha entrada en CE pasaríamos o seguinte punto.

3. Eleximos o primer evento por tempo de CE, actualizamos o tempo do simulador ao tempo asociado a esa actividades e executámolo, é decir, eliminámolo do CE e engadímolo a RE.

4. Se a actividade que se acaba de executar é a última da traza engadímola ao RE e volvemos o inicio. En caso contrario xeramos un novo evento a partir do último evento seguindo o modelo da Petri Net co número de caso asociado e un tempo simulado. Esto se engade na CE e volvemos o primer punto.

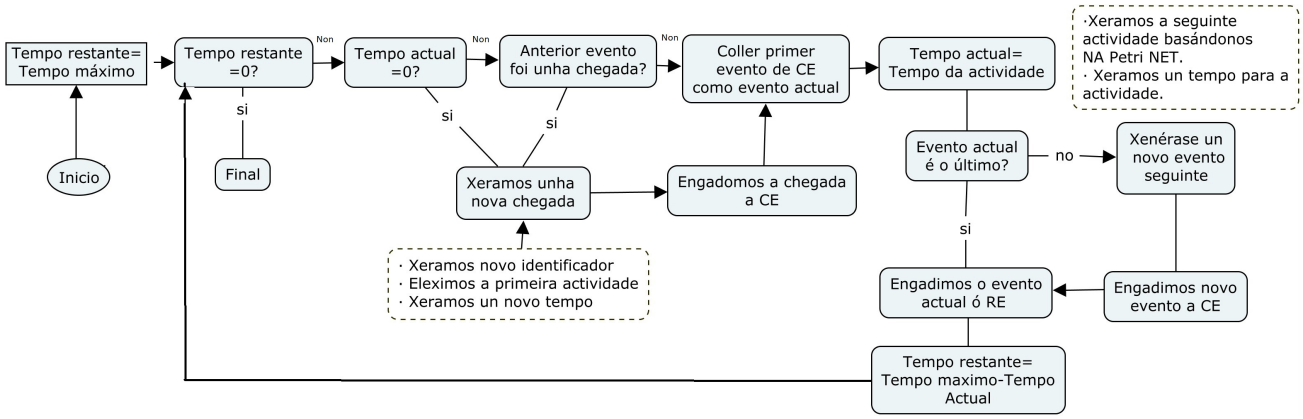

Figura 3.1: Diagrama del algoritmo de fluxo

Axudarémonos dun exemplo para adquirir unha maior compresión do algoritmo, para cada evento utilizaremos a notación $(CaseId|Actividade|Tempo)$ e ademais supondremos que sólo temos dúas actividades posibles A, B sendo A a actividade incial e B a final. Neste caso vamos a partir de que o simulador non ten ningún evento tanto en CE como en RE, e vamos a fixar o tempo de simulación en T=10. Para axudarnos a seguir o algoritmo anterior utilizaremos o mesmo esquema durante o exemplo.

- Como acabamos de inicializar a simulación o tempo do simulador será cero polo que aínda quedaríanos eventos por simular.

- Da mesma forma non temos ningunha entrada en CE, polo que simularíamos unha entrada que vamos a denotar por A co número de caso asociado e o tempo $(1|A|2)$.

- Na actualidade o único evento que temos en CE é $(1|A|2)$ polo que actualizaríamos o tempo do simulador a t=2 e seguindo o punto tres eliminanríamolo de CE e engadiríamolo a RE.

- Agora o tempo do simulador segue sendo menor que o tempo fixado $(t = 2 < T = 10)$ polo que podremos crear unha nova actividade $(1|B|5)$, que engadiremos a CE e volveríamos o punto un.

Unha vez acabado este primer ciclo obtivemos que o tempo do simulador é t=2, en CE temos $(1|B|5)$ e en RE temos $(1|A|2)$. Continuando con este mesmo exemplo:

- O tempo do simulador está fixado en $t = 2 < T = 10$ polo que aínda nos quedan eventos por simular.

- Do mesmo xeito volvemos a no ter entradas en CE polo que deberemos de simular unha nova entrada $(2|A|11)$ e engadíla a CE.
- Neste momento teríamos en CE as actividades $(1|B|5)$ e $(2|A|11)$, polo que nos quedaríamos coa de menor tempo que sería $(1|B|5)$, polo que actulizamos o tempo a t=5, eliminámola de CE e engadímola a RE.
- Para o seguinte punto vemos que non sería o momento de executar un novo evento pois como comentamos antes B denota un evento final polo que volvería de novo o punto un.

Polo tanto rematada esta etapa o tempo do simulador é t=5, en CE temos $(2|A|11)$ e en RE temos $(1|B|5)$ e $(1|A|2)$. Finalmente continuemos un paso máis do algoritmo.

- O tempo do simulador está fixado en $t = 5 < T = 10$ polo que aínda nos quedan eventos por simular.
- Actualmente en CE temos unha entrada polo que non sería necesario crear unha nova.
- Ademáis actualmente é o único evento que temos almacenado en CE, polo que collemos a $(2|A|11)$ actualizaríamos o tempo a t=11, eliminaríamolo de CE e engadiríamolo a RE.
- Finalmente como o tempo do simulador agora é $t = 11 > T = 10$, finalizaríamos a nosa simulación.

O resultado da simulación sería o rexistro de eventos, que denotamos por RE, formado por $(1|A|2)$, $(1|B|5)$ e $(2|A|11)$. Como vemos diferentes incógnitas nos quedan presentes como por exemplo como se simula o tempo entre entradas ou o tempo asociado a cada unha das actividades. As seguintes seccións centraranse na presentación dos algoritmos que simularán estos tempos.

3.2. Simulación de entrada dos doentes

Na presente sección se precisará o algoritmo co que simularemos as entradas dos doentes no servicio de urxencias no hospital a partir dun rexistro de eventos de referencia. En concreto inicialmente elaboraremos un pequeno análisis que xustificará a elección dos métodos presentados posteriormente na descrición do algoritmo.

3.2.1. Introdución ó problema

Para a realización dunha simulación do proceso de urxencias será necesario en primeiro lugar tratar de conseguir que o número de entradas dos doentes do simulador sexa coherente coas entradas na realidade. Buscaremos esta coherencia sen medo ao sobreaxuste pois o número de entradas é un parámetro externo ao hospital, xa que un hospital non pode controlar as entradas que haberá no servizo de urxencias. No noso caso, a única información coa que contamos sobre as entradas no rexistro de referencias son os primeiros eventos, coa súa correspondente marca temporal, de cada unha das trazas.

De todos os xeitos debemos ter en conta que tampouco buscamos unha precisión absoluta no número de entradas senón que queremos engadirlle certa variabilidade ao noso simulador. Isto xustifícase partindo de que esta variabilidade tamén existe na realidade, pois se collésemos o mesmo día en dous anos diferentes seguramente o número de entradas non sexa o mesmo. Para a inclusión desta variabilidade no modelo de simulación basearémonos na utilización de modelos de Poisson, utilizados amplamente en estatística para a xeración de chegadas en simuladores de eventos discretos. No noso caso, o modelado das chegadas como un proceso de Poisson está xustificado, ademais, cunha análise exploratorio dos datos onde se aplicou o test χ^2 de Pearson e concluíuse que a distribución de Poisson

é a máis adecuada para o devandito proceso de chegadas.

Outro punto a ter en conta é que existe estacionalidade neste proceso de chegadas. En efecto, intuitivamente podemos concluír que o número de chegadas non será o mesmo un día de xaneiro que un día de xuño, ou un luns con respecto a un domingo, ou no intervalo das 10.00 ás 11.00 que das 02.00 ás 03.00. Detectar todas as dependencias estacionais, sexan cales sexan, a partir do rexistro de eventos de referencia é un problema de difícil solución. Por iso, optamos pola seguinte simplificación. Como o noso caso de uso céntrase exclusivamente no proceso do servizo de urxencias, sabemos que as principais dependencias estacionais serán:

1. Por mes do ano
2. Por día da semana
3. Por intervalo horario do día

É máis, non é desatinado estender estas tres dependencias estacionais principais a calquera proceso asistencial dentro do ámbito sanitario. Efectivamente, noutro tipo de procesos (industriais, operativos, administrativos...) noutros sectores, isto non será certo e deberase abordar o problema desde outra perspectiva.

En resumo, o algoritmo proposto tratará de axustarse o máximo posible ao proceso de entradas subxacente ao rexistro de referencia, pero mantendo un grao de xeneralización, e ademais deberá ter en conta as dependencias estacionais principais introducidas arriba, parametrizándoas en función do rexistro de referencia.

3.2.2. Introdución ó algoritmo de entrada de doentes

Nesta sección presentaremos o algoritmo baseado na análise feita na introdución ao problema. Por tanto este algoritmo deberá:

1. Contar coa variabilidade que existe na realidade e que lle dotará o proceso de Poisson.
2. Ter en conta as diferenzas que se observaron nas entradas nos distintos meses, días e horas.

O obxectivo principal é obter os parámetros que caracterizan aos distintos procesos de Poisson que existirán para cada dependencia estacional. Desta maneira, o xerador de chegadas do futuro simulador, optará por un proceso de Poisson ou outro en función do mes, do día da semana e do intervalo horario en que se atope o tempo de simulación en cada iteración da mesma.

Para esto, optouse por un enfoque top-down:

1. Primeiro, detectaranse que meses do ano son parecidos entre si (o que denominamos agrupar por meses).
2. Despois, dentro de cada grupo mensual, detectaranse que días da semana son parecidos entre si (o que denominamos agrupar por días).
3. Por último, dentro de cada grupo diario, detectaranse que intervalos horarios son parecidos entre si (o que denominamos agrupar por horas).

Estes agrupamentos faranse utilizando o algoritmo k-means, en concreto o algoritmo implementado foi o proposto en *Hartingan e Wong (1979)* elixindo 20 centros distintos de forma arbitraria para todos os casos. Por último, tras obter estes agrupamentos, calcularemos os parámetros do proceso Poisson para cada un dos grupos. O formato de entrada de datos que soporta este algoritmo é o mesmo que o do Cadro 3.1. Nela aparecen o número de entradas por día e hora,

Día/Hora	00	01	02	03	04	05	06	07	...
01/01/18	12	17	12	10	8	8	8	12	...
02/01/18	12	5	5	8	3	6	9	7	...
03/01/18	18	15	10	5	6	7	4	4	...
04/01/18	18	7	8	7	7	4	11	10	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	...

Cadro 3.1: Exemplo de entrada de datos do algortitmo que xera entradas.

É dicir, calculamos o número de entradas/chegadas (o número de primeiros eventos de cada traza no rexistro de referencia) presentes en cada intervalo horario para cada día.

3.2.3. Descrición do algoritmo de entrada de doentes

A idea básica do algoritmo como comentamos será agrupar as unidades de tempo cuxo número de entradas sexa parecido para estimar os parámetros necesarios para simular utilizando o proceso Poisson. Para realizar estas agrupacións comezaremos coas unidades de tempo maiores (anos) e iremos agrupando cara as máis pequenas (horas), realizando estes agrupacións co algoritmo k- means. Decidimos facer este tipo de agrupamentos tratando de buscar o punto intermedio entre o resultado que se produciría ao coller un lambda específico para cada mes, día e hora e que se produce ao aplicar k- means á matriz de datos inicial. Deste xeito podemos afirmar que con respecto ao primeiro caso evitaríamos que algoritmo non se vexa afectado por días "raros"como un festivo ou respecto ao segundo caso evitariamos que aparecesen grupos por succesos "raros"continuados nun período de tempo máis ou menos longo como pode ser unha semana de greve. Por tanto, dividiremos o algoritmo en distintos pasos segundo na unidade temporal na que nos atopemos que decidiremos en primeiro lugar.

Detección das unidades de tempo.

O primeiro paso consistirá en detectar a cantidade de datos que temos, para iso simplemente será necesario contar o número de filas que temos na matriz anterior. A partir deste cálculo decidimos en que paso comezamos a agrupación. Doutra banda supoñeremos que como mínimo no noso rexistro de eventos aparecerán recolleitos dous días. Os pasos para seguir serán os recollidos na Figura 3.2.

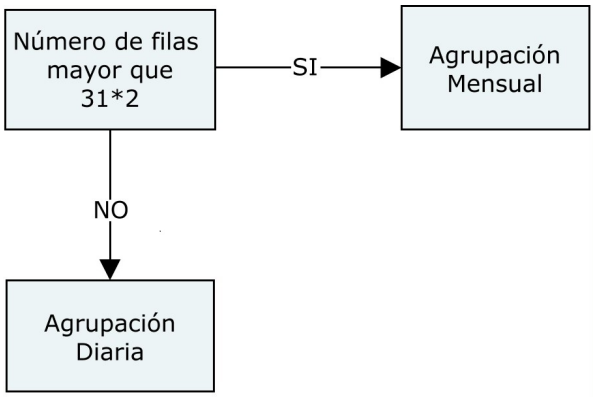

Figura 3.2: Diagrama de fluxo para detección de unidades de tempo.

Seguindo o diagrama decidiremos para cada conxunto de datos o primeiro agrupamento que realizaremos.

Agrupacións Mensuais.

En caso de cumprir as condicións necesarias do diagrama anterior para os meses, os pasos a seguir serán,

1. Contabilizar o número medio de entradas durante os distintos meses, estean completos ou non. É dicir debemos de ter un vector que para cada mes conteña o número medio de entradas cada mes.

2. Utilizar o algoritmo k-means para agrupar sistematicamente os meses segundo o número medio de entradas. Para iso comezaremos cun único grupo e iremos aumentando un grupo ata que haxa tantos grupos como meses. Á vez para cada un destes agrupamentos calculamos o erro de agrupamento asociado, este calcúlase como a suma ao cadrado das diferenzas entre os datos dun grupo e os centros de cada grupo, sendo o erro total a suma dos erros de todos os grupos. Desta forma para cada k denotamos por x_{ij} con j=1,...,k e i=1,...,n_j os i datos que pertencen ao grupo j onde n_j é o número de datos que pertencen á grupo j e por $\bar{x}_j$ os centros asociados a cada grupo. Tense que o erro asociado a cada número k de grupos será:

$$Error_k = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2$$

3. Unha vez teñamos os erros totais asociados para cada número de grupos, calculamos as diferenzas entre os erros ao engadir un grupo. É dicir, imos calculando as diferenzas entre os erros totais entre ter un só grupo a ter dous, de ter dous a ter tres e así ata chegar ao total de grupos. Seguindo a notación anterior para cada k terase que,

$$Pend_k = Error_k - Error_{k+1} \qquad k = 1, ..., 11$$

4. Cando teñamos estas diferenzas calculadas, dividimos todas as diferenzas pola primeira obtendo así unhas diferenzas "normalizadas". Tratamos así de obter un erro que nos sirva de forma global sen importar o caso. Estas diferenzas normalizadas poden ser calculadas como,

$$Pend_k = Pend_k / Pend_1 \qquad k = 1, ..., 11$$

5. A partir destas diferenzas normalizadas consideramos que a mellora en canto ao erro que provoca engadir un grupo non será suficiente se a diferenza normalizada é menor a 0.01. É dicir, quedamos co número de grupos cuxa diferencia normalizada sexa menor que 0.01. Se este criterio non se cumpre nunca quedamos con que cada mes formará un grupo.

6. Finalmente gardamos neste paso os gruops cos meses.

Agrupacións Diarias

Neste punto podémonos atopar con dúas situacións:

- O número de datos non foi o suficientemente grande como para facer unha agrupación mensual, por tanto debemos calcular o número de entradas medio para cada un dos días da semana. É dicir, o número de entradas medio para os luns, martes, etc.

- Fixemos a agrupación mensual anterior polo que calcularemos a entrada media para cada un dos días diferenciando polos grupos creados anteriormente.

Unha vez calculadas estas entradas medias para algunha das opcións anteriores volvemos aplicar o algoritmo neste caso tomando un criterio de erro máis laxo,

1. Contabilizar o número medio de entradas durante os distintos días estean completos ou non. É dicir debemos de ter un vector que para cada mes conteña o número medio de entradas cada día.

2. Utilizar o algoritmo k- means para agrupar sistematicamente os meses segundo o número medio de entradas. Para iso comezaremos cun único grupo e iremos aumentando un grupo ata que haxa tantos grupos como días. Á vez para cada un destes agrupamentos calculamos o erro de agrupamento asociado, este calcúlase como a suma ao cadrado das diferenzas entre os datos dun grupo e os centros de cada grupo, sendo o erro total a suma dos erros de todos os grupos. Desta forma para cada k denotamos por x_{ij} con j=1,...,k e i=1,...,n_j os i datos que pertencen ao grupo j onde n_j é o número de datos que pertencen á grupo j e por $\bar{x}_j$ os centros asociados a cada grupo. Tense que o erro asociado a cada número k de grupos será:

$$Error_k = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2$$

3. Unha vez teñamos os erros totais asociados para cada número de grupos, calculamos as diferenzas entre os erros ao engadir un grupo. É dicir, imos calculando as diferenzas entre os erros totais entre ter un só grupo a ter dous, de ter dous a ter tres e así ata chegar ao total de grupos. Seguindo a notación anterior para cada k terase que,

$$Pend_k = Error_k - Error_{k+1} \qquad k = 1, ..., 6$$

4. Cando teñamos estas diferenzas calculadas, dividimos todas as diferenzas pola primeira obtendo así unhas diferenzas "normalizadas". Tratamos así de obter un erro que nos sirva de forma global sen importar o caso. Estas diferenzas normalizadas poden ser calculadas como,

$$Pend_k = Pend_k / Pend_1 \qquad k = 1, ..., 6$$

5. A partir destas diferenzas normalizadas consideramos que a mellora en canto ao erro que provoca engadir un grupo non será suficiente se a diferenza normalizada é menor a 0.01. É dicir, quedamos co número de grupos cuxa diferencia normalizada sexa menor que 0.01. Se este criterio non se cumpre nunca quedamos con que cada mes formará un grupo.

6. Finalmente gardamos neste paso os días que pertencen a cada grupo dentro dos grupos mensuais formados no paso anterior.

Agrupacións por Horas

Neste punto non teremos que facer a diferenza pois supoñemos que como mínimo no noso rexistro de eventos contamos cun día rexistrado, por tanto neste punto imos ter sempre unha agrupación por días aínda que esta sexa que cada día este en un grupo individual. Como sempre neste paso utilizaremos a información que obtivemos nos pasos anteriores.

1. Contabilizar o número medio de entradas durante as distintas horas que deben estar agrupadas segundo os pasos anteriores. É dicir debemos calcular as entradas medias para as mesmas horas dentro dos grupos de días e meses que obtivemos nos pasos anteriores. Partimos por tanto dun vector que para cada hora do día contén o número de entradas medio para cada un dos grupos que se foron formando nos pasos anteriores.

2. Utilizar o algoritmo k-means para agrupar sistematicamente os meses segundo o número medio de entradas. Para iso comezaremos cun único grupo e iremos aumentando un grupo ata que haxa tantos grupos como horas. Desta forma para cada k denotamos por x_{ij} con j=1,...,k e i=1,...,n_j os i datos que pertencen ao grupo j onde n_j é o número de datos que pertencen á grupo j e por $\bar{x}_j$ os centros asociados a cada grupo. Tense que o erro asociado a cada número k de grupos será:

$$Error_k = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2$$

3. Unha vez teñamos os erros totais asociados para cada número de grupos, calculamos as diferenzas entre os erros ao engadir un grupo. É dicir, imos calculando as diferenzas entre os erros totais entre ter un só grupo a ter dous, de ter dous a ter tres e así ata chegar ao total de grupos. Seguindo a notación anterior para cada k terase que,

$$Pend_k = Error_k - Error_{k+1} \qquad k = 1, ..., 23$$

4. Cando teñamos estas diferenzas calculadas, dividimos todas as diferenzas pola primeira obtendo así unhas diferenzas "normalizadas". Tratamos así de obter un erro que nos sirva de forma global sen importar o caso. Estas diferenzas normalizadas poden ser calculadas como,

$$Pend_k = Pend_k / Pend_1 \qquad k = 1, ..., 23$$

5. A partir destas diferenzas normalizadas consideramos que a mellora en canto ao erro que provoca engadir un grupo non será suficiente se a diferenza normalizada é menor a 0.1. É dicir, quedamos co número de grupos cuxa diferencia normalizada sexa menor que 0.1. Se este criterio non se cumpre nunca quedamos con que cada mes formará un grupo.

6. Finalmente gardamos neste paso as horas que pertencen a cada grupo dentro dos grupos diaiors e mensuais formados nos pasos anteriores.

Simulación

Unha vez chegados aquí xa clasificamos toda unidade de tempo nun dos grupos, polo que para cada franxa horaria na que estea o simulador terá un parámetro asociado a partir de cal poderá xerase as entradas. A idea será que o usuario fixe o espazo de tempo que quere xerar, por exemplo se quere xerar 48 horas dun luns de Xuño a partir da 1 da noite. A partir desta entrada de datos o simulador deberá escoller o parámetro asociado a esa franxa horaria e simular os as entradas seguindo un proceso Poisson que explicaremos a continuación.

Definición 3.1 *Unha variable aleatoria seguirá unha distribución exponencial con parámetro* $\beta > 0$ *, se a súa función de densidade de probabilidade ven dada por:*

$$f_X(t) = \begin{cases} \beta e^{-\beta t} & se \quad t \geq 0 \\ 0 & se \quad t < 0 \end{cases}$$

Definición 3.2 *Unha variable aleatoria X seguirá unha distribución Poisson de parámetro* $\lambda > 0$ *se toma valores no conxunto* $\{0, 1, 2, ...\}$ *con probabilidade,*

$$P(X = k) = e^{-\lambda} \frac{\lambda^k}{k!} \qquad k = 1, 2, 3, ..$$

Estas dúas variables aleatorias gardan unha estreita relación entre elas que veremos grazas ao proceso Poisson. Para definirlo formalmente precisaremos antes introducir certa notación apoiándonos na Figura 3.3,

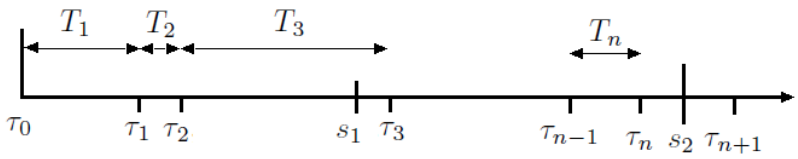

Figura 3.3: Diagrama de tempos e eventos dun proceso Poisson

Na figura 3.3 denotamos por τ_0 o instante inicial, ademais denotaremos por $T_1, ..., T_n$ os tempos entre dúas chegadas consecutivas. Así por exemplo o tempo entre a segunda e a primeira chegada será T_2.

Por outro lado, denotamos por τ_n con n=1,2,... os tempos dende o instante inicial τ_0 ata a chegada n-ésima. Así si queremos calcular o tempo τ_n simplemente teremos que facer $\tau_n = \sum_{i=1}^{n} T_i$.

Por último se seleccionamos un tempo arbitrario s, o número de chegadas dende τ_0 ata s será igual o primeiro dos índices de τ_n que verifique que $\tau_n < s$. Por exemplo tomando o s_1 da Figura 3.3, teríase que o evento 2 sería o primeiro que cumple esa condición. Agora xa contamos con todos os ingredientes para definir o proceso de Poisson.

Definición 3.3 *Sexa* $T_1, ..., T_n$ *un conxunto de variables aleatorias independentes que seguen unha distribución exponencial de parámetro* λ*. Sexa* τ_0 *e* $\tau_n = \sum_{i=1}^{n} T_i$ *para* $n > 0$ *. Definimos o proceso Poisson de parámetro* λ *por:*

$$\{N(s) = \text{máx}\{n : t_n \leq s\}, \quad s \geq 0\}$$

De forma equivalente podemos definir o proceso poisson tamén como,

Definición 3.4 *Sexa* $\{N_t, t > 0\}$ *é un proceso de Poisson de parámetro* λ *se:*

1. $N_0 = 0$
2. *Para calquer* $t_0 < t_1 < ... < t_n$ *as variable* $X_{t_1} - X_{t_0}$,$X_{t_2} - X_{t_1}$,...,$X_{t_n} - X_{t_{n-1}}$ *son independentes.*
3. $N_{s+t} - N_s \sim Poisson(\lambda t)$

Por tanto neste punto temos para cada intervalo de tempo concreto un parámetro e proceso Poisson será a ferramenta a partir da cal realizaremos as simulacións. Polo tanto se estamos nun intervalo de tempo concreto escolleremos o parámetro asociado e xeraremos tempo coa función de distribución exponencial seguindo o proceso Poisson nese intervalo de tempo, cando cambiemos de intervalo xeraremos outro proceso Poisson nese intervalo.

3.3. Algoritmo de búsqueda da función da probabilidade dos tempos entre servizos

Nesta sección detallaremos o algoritmo utilizado para obter a función de distribución a partir da cal xeraremos os tempos de servizo no noso simulador. Previamente á introdución do algoritmo, faremos un achegamento á metodoloxía,a cal a de ser o máis manexable posible para o usuario.

3.3.1. Introdución ó problema da simulación dos tempos entre servizos

Na simulación do proceso de urxencias é de vital importancia xerar o tempos entre eventos con distintos métodos que sexan comprensibles para o usuario e á vez deben ser coherentes coa realidade. A procura da coherencia radica en que se unha simulación quere ser útil esta ha de ser como mínimo lóxica. Un bo exemplo é precisamente o caso dos tempos, xa que un bo simulador nunca debe crear tempos negativos, imposibles na realidade, ou tempos que se afasten moito en media dos que pasan na realidade.

Doutra banda debemos de buscar métodos cuxo resultado sexa facilmente manexable por usuarios que poden non ter coñecementos estatísticos, así cremos que o resultado final debe ser unha distribución cuxos parámetros sexan facilmente interpretables como a media. Comentar que na literatura estatística existen distintos métodos a partir dos cales poderíamos recrear os tempos de forma moi precisa, con todo descartáronse métodos de estatística non paramétrica ou métodos de mixtura de normais combinados co algoritmo EM, cuxos resultados non concordaba co noso axioma de sinxeleza.

Finalmente a través da análise realizada sobre os datos podemos concluír:

1. As estacionalidades, que estaban presentes no número de entradas, nos que tiñamos en conta o mes e o día da semana non parecen influír nos tempos entre servizos.
2. A única estacionalidade plausible parece o efecto das horas do día, con todo estas estacionalidades parecen afectar só as actividades que adoitan desenvolverse ao principio no proceso de urxencias. Polo que tampouco estudiaremos estacionalidade no tempo.

Á vista dos puntos anteriores decidiuse non ter en conta ningún efecto de estacionalidade no noso algoritmo.

3.3.2. Introdución ó algoritmo de búsqueda da función de distribución máis razoable

A idea xeral do simulador é que a partir dun rexistro de eventos de referencia, selecciónese a función de distribución que mellor se axusta a el para xerar os tempos. Como comentamos traballaremos con funcións de distribución sinxelas, que poderán ser modificadas de forma intuitiva mediante algún dos seus parámetros. Por exemplo moitas destas variables contarán cun parámetro media, de forma que é fácil intuír que se reducimos á metade este parámetro obteremos que os tempo que se xeren serán en media a metade. Entre as funcións de distribución guiámonos/guiámosnos pola análise realizada en *Kim(2013)* e no *Natrella(2010)*, do que concluímos que utilizaremos as distribucións Uniforme, Exponencial, Weibull, Gamma e Lognormal.

Por tanto agora a incógnita que nos queda por resolver é que distribución é máis compatible cos nosos datos. Neste contexto toman relevancia os tests de bondade de axuste como o tests de Kolmogorov-Smirnov, o test de Anderson- Darlingn o test de Cramér-Von Mises ou o test Chi Cadrado entre outros. Estes testean a hipótese de que os nosos datos seguen unha distribución paramétrica cos parámetros totalmente especificados. No noso caso traballaremos co test de Kolmogorov- Smirnov por ser os tempos unha variable continua e por ser o de uso máis estendido, pois en definitiva os outros son modificacións del exceptuando Chi-Cadrado. A grandes liñas co test de Kolmogorov- Smirnov calculase a distancia máxima entre a distribución que queremos testear e a función de distribución empírica, que é unha estimación non paramétrica construída a partir dos datos.

En concreto o test de Kolmogorov-Smirnov trátase de un tests non paramétrico de bondade de axuste para distribucións de probabilidades unidimensionais que nos serve para comparar unha distribución de probabilidade con outra de referencia. A distribución do estadístico foi calculada en *Smirnov(1939)* e en *Kolmogorov(1933)*. O estatítico de Kolmogorov-Smirnov calculase como,

$$D_n = \sup_x |F_n(x) - F(x)|$$

onde a F_n é a función de distribución empírica que se calcula como,

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I(X_i) \quad \text{onde} \quad I(X_i) = \begin{cases} 1 & se \quad X_i \leq x \\ 0 & se \quad X_i > x \end{cases}$$

A problemática final recaerá en que como comentamos o tests necesita que a distribución deba estar totalmente especificada, isto quere dicir que os parámetros das distribucións teñen que ser os reais. Se estes parámetros son estimados o test será conservador, é dicir, será máis fácil que se acepte que unha distribución é compatible cos nosos datos que se os parámetros fosen os reais como se pode ver en *Gihman(1961)* ou *Massey(1951)*. Para solucionar esta problemática na literatura estatística se propoñeron distintas correccións como en *Lilliefors(1967)*, *Lillieflors(1969)*, *Pearson e Wirching(1982)*, *Durbin(1976)* ou en *Babu e Rao(2004)*. No noso caso decidímonos por utilizar unha metodoloxía baseada na simulación análoga á proposta en *Lillieflors(1967)* y *Lillieflors(1969)* adaptándoa a todas as distribucións propostas anteriormente. A grande rasgos crearemos conxuntos de datos coa distribución que queremos testear e calcularemos para cada un deles o valor do estatístico de Kolmogorov- Smirnov que logo compararemos co valor do estatístico de Kolmogorov- Smirnov dos datos reais. A idea é que se os nosos datos seguen a distribución que conxeturamos o valor do estatístico para os nosos datos será "parecida"ó estatístico para os conxuntos de datos que simulamos.

3.3.3. Descrición do algoritmo de búsqueda da función de distribución máis razoable

A idea básica como viñemos comentando será a partir dos tempos entre dúas actividades, calcularemos os valores do test obtidos coa simulación para as distribucións que propuxemos e quedarémonos coa que mellor se axuste aos nosos datos. Por tanto o noso algoritmo só necesitará como entrada os datos dos tempos entre as actividades para a cal queremos calcular a función de distribución. A partir dos cales aplicaremos os seguinte pasos.

1. Calculamos as estimacións dos parámetros das distribucións Uniforme, Exponencial, Weibull, Gamma e Lognormal. No caso das distribucións distribución Gamma e Weibull utilizaremos a compostas polos parámetros de escala e forma. Para o cálculo da súa estimación utilizaremos o método de máxima verosimilitude. En ocasións os parámetros obtido por máxima verosimilitud non terá unha fórmula implícita e teremos que resolver a ecuación de máxima verosimilitude por un método iterativo, no noso caso será Newton- Raphson.

2. Calcular o estatístico de Kolmogorov- Smirnov dos datos reais supoñendo unha distribución con eses parámetros estimados. Na sección 3.3.2 explicítanse este test e os cálculos para obter o estatístico.

3. Xeramos B conxuntos de datos coa distribución que queremos testear e cos parámetros calculado no primeiro punto. Un número B=5000 pode ser indicado.

4. Para cada un dos B conxuntos de datos calculamos o estatístico de Kolmogorov- Smirnov de forma análoga a como o faciamos no punto 2 para os datos reais.

5. Construímos agora a función de distribución empírica dos estatísticos de Kolmogorov- Smirnov, a función de distribución empírica está indicada como calculala na sección 3.3.2.

6. A nova probabilidade calcularémola substituíndo na función de distribución empírica o valor do estatístico de Kolmogorov- Smirnov para os datos reais calculado no punto 2. Novamente esta información está ampliada na sección 3.3.2.

7. Finalmente repetiremos do punto 3 ao 6 anterior para todas as distribucións: Uniforme, Exponencial, Weibull, Gamma e Lognormal. Ao final teremos unha probabilidade asociada para cada unha das distribucións e quedarémonos coa que teñan o valor máis alto.

Ao final deste algoritmo por tanto teremos unha función de distribución coa que xeraremos os tempos entre cada par de actividades enlazadas no simulador. Unha vez obtidos as distribucións podremos realizar as simulación mediante distintos métodos por exemplo no caso da exponencial podemos utilizar o método da transformada inversa.

Capítulo 4

Calidade do modelo

Durante o desenvolvemento do modelo de simulación, necesitamos un valor cuantitativo que nos diga se as sucesivas melloras implementadas en iteracións sucesivas realmente melloran o resultado final da simulación e en que medida. Para conseguir este obxectivo, eliximos unha serie de estatísticos que nos servirán para avaliar a calidade do modelo de simulación en función do rexistro de eventos que xera.

Nesta sección trataremos de describir o contexto de avaliación no que nos encontramos e expoñeremos os devanditos estatísticos. Para cada un deles facilitaremos unha explicación do marco teórico que o sustenta e un resumo do seu cálculo.

4.1. Contexto de avaliación

A avaliación da calidade dun proceso é un elemento crucial para identificar se un cambio no modelo simulación supuxo unha mellora ou un empeoramento nel. Para identificar que medidas son as máis adecuadas para avaliar esta calidade é necesario reflexionar sobre cales son os elementos necesarios para levar a cabo a simulación e cales son os resultados que obtemos a partir dela. Por tanto será importante analizar previamente en que se fundamenta o simulador.

A obtención do simulador pode dividirse en dúas fases.

1. Constrúese, utilizando técnicas de minería de procesos e algoritmos de cálculo de indicadores estatísticos sobre distintos aspectos do proceso, un modelo de simulación que sexa o máis próximo posible ao proceso real (que nos é descoñecido) a partir dun rexistro de eventos de referencia (que habitualmente será un rexistro de eventos de datos reais).
2. Execútase un algoritmo de simulación (baseándose nun simulador de eventos discretos, por exemplo) usando o modelo de simulación construído na fase previa. Desta maneira, o modelo de simulación ditaranos as pautas que este algoritmo debe seguir á hora de xerar chegadas de novos casos ao proceso, xerar tempos de espera e de servizo, decidir que opción tómase nun punto de decisión, onde comeza ou finaliza un caso no proceso etc.

O resultado final, por tanto, de executar o simulador será un rexistro de eventos simulados coa mesma estructura que o rexistro de eventos de entrada. Tendo en conta isto, decidimos que unha boa forma de avaliar o simulador é comparar o seu resultado coa realidade, é dicir, comparar ambos os rexistros de eventos: o simulado e o de referencia. Concretamente, centrámonos en dous enfoques á hora de realizar esta comparativa:

1. Avaliar como de bo é o simulador xerando **trazas que manteñan a estrutura xeral** das observadas no rexistro de referencias. É dicir, avaliar que o control do fluxo que realiza o simulador

axústase en maior ou menor medida á realidade. Isto involucra o propio modelo do proceso descuberto, así como outros parámetros que definan o camiño que un caso xerado segue a través do proceso durante a simulación:

   a) Se se observan os mesmos patróns de comportamento na chegada de casos ao proceso (por exemplo, tendo en conta dependencias na taxa de chegadas con respecto á hora do día en que se producen),

   b) En que punto do proceso comeza cada caso e con que distribución

   c) Que opcións tómanse en cada punto de derivación do proceso e con que distribución,

   d) Se se cumpren as relacións causales que se observan na realidade,

   e) En que punto do proceso finaliza cada caso

2. Avaliar a bondade do simulador desde un punto de vista **temporal**. É dicir, como de semellantes son os selos temporais (as mostras de tempo) xeradas coas observadas na realidade e como de semellante é o seu comportamento conxunto con respecto ao observado no rexistro de referencia: tempo medio de cada caso, tempo medio de estancia total no proceso, tempo medio entre cada par de actividades etc.

Por tanto a seguinte sección dividirase en dúas seccións principais que responderán aos dous enfoques. Na primeira sección introduciremos os estatísticos que nos medirán a similitude desde un punto de vista temporal. Na segunda, os estatísticos que nos medirán a similitude da estrutura de trazas. Á súa vez esta última sección será dividida en tres subseccións:

1. Centrándonos na similitude do proceso de chegadas simuladas e as observadas.
2. Centrándonos na similitude entre o número de actividades simuladas e as observadas.
3. Centrándonos na similitude entre os patróns das trazas (ou variantes) simuladas e as observadas.

4.2. Estadísticos para o tempo

Nesta primeira sección explicaremos o estatístico que tratará de comprobar a similitude entre os tempos do rexistro de referencia e os tempos do rexistro simulado. Segundo o interese que teñamos, podemos utilizalo con distintos obxectivos como pode ser comprobar a similitude dos tempos que está cada paciente no sistema de urxencias ou comparar a similitude dos tempos entre dúas actividades adxacentes (o que denominamos, tempos de transición).

4.2.1. Estadístico de Wilcoxon

No noso contexto temos dúas mostras aleatoria independentes como son os tempos do rexistro de referencia e os tempos do rexistro simulado e queremos testear a hipótesis nula de que ambas mostras proveñen da mesma poboación. Á hora de comparar dúas mostras de dúas poboacións, a media é unha boa referencia cando ambas as mostras posúen aproximadamente o mesmo número de observacións por encima e por baixo do devandito valor, é dicir, cando ambas as mostras son simétricas respecto a súa media. Entón é melloor utilizar estadísticos que estén basados na mediana ou, no seu defecto, nas posicións dos elementos dentro da mostra.

O estatístico de Wilcoxon permitiranos comprobar se dúas mostras pertencen á mesma poboación. Intuitivamente o test de Wilcoxon consiste en combinar ambas as mostras, asignar rangos ás observacións xuntas e definir o estatístico como a suma dos rangos das observacións nunha das dúas poboacións. Se a suma é demasiado pequena ou demasiado grande, ésto sería evidencia de que os valores dunha das poboacións tenden a ser máis pequeno ou máis grandes que os da outra e rexeitaríase

a hipótese nula de igualdade entre as mostras.

Formalmente necesitmos dúas mostras aleatorias simples $X_1, ..., X_n$ con función de distribución F(x) e $Y_1, ..., Y_n$ con función de distribución G(x) independiente da primeira. Podemos encontrarnos en algunha das tres situacións:

1. $H_0 : X = Y$ e $H_1 : X \neq Y$
2. $H_0 : X \geq Y$ e $H_1 : X < Y$
3. $H_0 : X \leq Y$ e $H_1 : X > Y$

No noso caso só nos interesará o primer caso xa que só queremos saber se as mostras proveñen ou non da mesma poboación. Finalmente para o seu cálculo procederemos da seguinte forma:

1. Ordéanse de forma ascendente ambas mostras fixándose si proceden da mostra X e Y.
2. Asígnase a cada dato o rango na mostra combinada de tamaño n+m
3. Calcúlase $R_x = \sum_{j=1}^{n+m} j \cdot I\{\text{A observación j-esima pertence a mostra X}\}$

En *Wilcoxon et al.(1970)* podemos ver o seguinte teorema que nos indica a distribución.

Teorema 4.1 *Se o número de datos é superior a 20 en ambas mostras a distribución de Wilcoxon pódese aproximar por unha* $Normal(\mu, \sigma^2)$ *onde,*

$$\mu = \frac{n(n+m+1)}{2} \qquad \sigma^2 = \frac{nm(n+m+1)}{12}$$

Grazas a este criterio e fixando niveis de significación do 5 % determinaremos se as mostras son iguais ou non. Por outro lado no noso caso o interese recaerá en comparar dous valores numéricos para determinar entre dous simuladores cal funciona mellor. Neste caso o ideal sería realizar unha simulación parecida con ambos simuladores e obter os rexistros de eventos correspondentes. Tras esto deberíamos calcular os estatísticos de wilcoxon en ámbolos dous casos comparando os rexistros dos dous simuladores co real e ver cal dos dous é mellor. Coa intención de evitar a arbitrariedade de só simular un rexistro, trataremos de realizar varias simulacións dependendo o número da lonxitude das mesmas. Así podríamos enumerar os seguintes pasos,

1. Realizar B simulación de distinta lonxitude de tempo para os dous simuladores.
2. Calcular para cada unha destas o coeficiente indicado anteriormente.
3. Calcular a media de cada simulador.
4. Quedarse co simulador cuxo coeficiente sexa máis cercano a 0.

Obtendo de esta forma uns estadísticos que nos sirven para compara que simulador nos da mellor resultado en canto aos tempos entre actividades.

4.3. Estadístico para a estructura das trazas

Nesta sección presentaremos os estatísticos que avalían cuán bo é o simulador xerando patróns de comportamento similares aos observados. En concreto dividiremos esta parte en tres subseccións:

1. Na primeira, presentaremos un estatístico que nos, mida a similitude entre o número de eventos de cada actividade observados e xerados.

2. Na segunda, presentaremos un estatístico que nos mida a similitude entre a cantidade de chegadas observadas e xeradas.

3. Finalmente, presentaremos un estatístico que nos mida a similitude entre os patróns de trazas ou variantes observadas e xeradas.

4.3.1. Similitude entre o número de eventos de cada actividade

Nesta sección introduciremos os conceptos necesarios para medir a bondade do simulador xerando eventos coa mesma distribución de actividades que a observada no rexistro de referencia. Para iso introduciremos unha serie de conceptos que iremos intercalando con algún exemplo útil. Con este obxectivo presente decantámonos por utilizar o coeficiente de continxencia que tratará de cuantificar a relación entre o número de eventos de cada actividade que teñen lugar en ambos os rexistros: de referencia e simulado.

Como paso previo á obtención deste estatístico, debemos construír as chamadas táboas de continxencia que no noso caso terán a seguinte información:

- En xeral, cada fila correspóndese con cada unha das diferentes categorías observadas no rexistro de referencia. No noso caso usaremos as actividades para definir cada categoría.
- Para cada fila, calcúlanse dous valores,
  - O total de ocorrencias da devandita actividade no rexistro de referencia
  - O total de ocorrencias da devandita actividade no rexistro simulado

Un exemplo de táboa de continxencia para o proceso de urxencias dun hospital podería ser o Cadro 4.3.1.

Actividades	**Referencia**	**Simulado**
Admisión	4756	5030
Triaxe	4756	4943
Interconsulta	30	1
Respuesta	30	1
RX Exploración	2715	1520
RX Informe	322	945
RX Solicitud	2932	2407
Alta	4459	5027
Total	**20000**	**19874**

Cadro 4.1: Número de actividades de dous rexistros simulados.

A partir do Cadro 4.3.1 calcularemos un estatístico que nos indique como de homoxéneos son estes valores, o chamado Coeficiente de Continxencia. Este está baseado no estatístico χ^2 de Fisher. O estatístico χ^2 dinos o grao de similitude entre as frecuencias de ocorrencia dunha actividade no rexistro de referencia con respecto ao rexistro simulado. O seu cálculo divídese en tres pasos:

- Primeiro calculamos a táboa de continxencia coas frecuencias observadas no rexistro de eventos.
- Despois, calculamos a táboa de continxencia coas frecuencias esperadas baixo a suposición de homoxeneidade, tal e como veremos máis adiante.
- Por último, comprobamos se hai diferenzas significativas entre a táboa de frecuencias observadas e a táboa de frecuencias esperadas. Se non as hai, entendemos que ambas as mostras son homoxéneas.

O cálculo da táboa de continxencia coas frecuencias observadas é trivial. Simplemente contabilizamos a observación de cada categoría no rexistro de eventos. Por outro lado calcular a segunda das táboas debemos obter en primeiro lugar as estimacións das probabilidades de cada categoría se houbese homoxeneidade. Matemáticamente se denotamos por a_{ij} o número de ocurrencias na fila i e na columna j, por $a_{i\cdot}$ a suma dos elementos da fila i e $a_{\cdot j}$ a suma dos da columna j e por A o número total de datos, temos que para a categoría da fila i a probabilidade virá dada por:

$$P(\text{Categoría i}) = \frac{a_{i1} + a_{i2}}{A}$$

Para o cálculo das frecuencias esperadas realizamos unha repartición do número total de observacións en partes proporcionais á probabilidade observada de cada categoría, que calculamos coa fórmula anterior e denotaremos por e_{ij}.

$$e_{ij} = a_{\cdot j} \cdot P(\text{Categoría i}) = a_{\cdot j} \cdot (a_{i\cdot}/A) = (a_{\cdot j} \cdot a_{i\cdot})/A$$

Finalmente unha vez teñamos as frecuencias esperadas calculadas para toda fila i e columna j, xa podemos calcular o estatístico χ^2 de Fisher como as diferenzas entre o número de eventos observados e os esperados. Así se temos k actividades distintas teremos que o estadístico calcularase como,

$$\chi^2 = \sum_{i=1}^{k} \sum_{j=1}^{2} \frac{(a_{ij} - e_{ij})^2}{e_{ij}}$$

Sen embargo este estatístico ten o problema de non estar limitado, polo que podemos utilizar o coñecido como coeficiente de continxencia. Para o seu calculo utilizaremos o χ^2 de Fisher e o número total de observacións , da seguinte forma:

$$C = \sqrt{\frac{\chi^2}{\chi^2 + n}}$$

Este estatístico tomará valores entre cero e un, de forma que canto máis próximo a cero estea maior será a homoxeneidade na táboa, sendo o cero indicativo de homoxeneidade total. Por tanto no noso caso cada vez que realicemos unha modificación nalgunha parte do algoritmo que poida afectar as actividades podemos de comparar se o resultado que obtemos é mellor que antes da modificación mediante este estatístico. Como no caso anterior un bo procedemento de comparación sería,

1. Realizar B simulación de distinta lonxitude de tempo para os dous simuladores.
2. Calcular para cada unha destas o coeficiente indicado anteriormente.
3. Calcular a media de cada simulador.
4. Quedarse co simulador cuxo coeficiente sexa máis cercano a 0.

Igual que no caso anterior canto maior sexa o número de simulacións obteremos resultados máis fiables. No noso caso estivemos realizando 1000 simualción para guiarnos.

4.3.2. Similitude no proceso de chegada de doentes

Tal e como se explicou na Sección 3.2, o simulador tenta xerar as chegadas ao proceso detectando a estacionalidade que puidese existir no rexistro de referencia. En efecto, se supoñemos que estamos a traballar co proceso do servizo de urxencias dun hospital, parece lóxico pensar que a taxa de pacientes dependerá:

- Do mes no que nos atopemos: non é o mesmo agosto, cando moita xente está de vacacións, que un mes con gran influencia da gripe como decembro,
- Do día da semana: debería haber diferenzas entre a fin de semana e o luns, por exemplo.
- Da hora do día: como mínimo, deberían de detectarse as diferenzas entre as horas nocturnas e as diúrnas..

Por todo iso, é importante que os estatísticos sexan capaces de captar estas dependencias para modificar o proceso de chegadas do simulador. Con esta intención realizaremos dous tipos de tablas de continxencia de forma que aplicaremos o estadístico anterior de forma totalmente análoga. As novas tablas terán categorías distintas a anterior de forma que na primeira tabla as categorías pasarán a ser os días da semana como podemos ver na Cadro 4.3.2.

Día	**Referencia**	**Simulado**
Día 1	501	467
Día 2	577	487
Día 3	548	455
Día 4	523	447
Día 5	428	465
Día 6	462	449
Día 7	499	461
Total	**3538**	**3231**

Cadro 4.2: Número de entradas por días de dous rexistro de eventos simulados.

E na segunda as categorías pasarán a ser as horas do día como podemos ver na Táboa 4.3.2 onde para simplificar eliminamos as últimas horas do día.

Franxa Horaria	Referencia	Simulado
00-01	187	280
01-02	146	301
02-03	128	320
03-04	95	322
04-05	100	277
05-06	78	271
06-07	86	297
⋮	⋮	⋮

Cadro 4.3: Número de entradas por hora e dous rexistro de eventos simulados.

Novamente a partir de estas dúas táboas podemos calcular o coeficiente de contixencia como describimos antes,

$$\chi^2 = \sum_{i=1}^{k}\sum_{j=1}^{2}\frac{(a_{ij}-e_{ij})^2}{e_{ij}} \quad \text{e} \quad C = \sqrt{\frac{\chi^2}{\chi^2+n}}$$

De forma que para evitar a aleatoriadade de só comprabar unha simulación podemos seguir o método proposto anteriormente para comprobar que simulación era mellor,

1. Realizar B simulación de distinta lonxitude de tempo para os dous simuladores.
2. Calcular para cada unha destas os coeficientes indicados para cada unha das tablas.
3. Calcular a media de cada simulador.
4. Quedarse co simulador cuxo coeficiente sexa máis cercano a 0.

Novamente a maior número de simulacións obteremos resultados máis fiables, nas probas en Gradiant e este traballo de fin de máster realizamos 1000 simulacións.

4.3.3. Similitude entre os patróns de trazas

Previamente a que nos adentremos no estudo destes indicadores necesitaremos entender o concepto de variantes ou patróns de traza. Tal e como se explica no Capítulo 2, cada dato dentro dun rexistro de eventos está formado por:

1. Unha actividade, a tarefa que se executou no devandito evento (a admisión do paciente, o triaxe, ou alta etc.)
2. Un identificador do caso ou instancia do proceso, é dicir, a quen ou a que se lle realizou certa tarefa (o doente, o episodio etc.)

3. Un selo temporal que indica cando se realizou dita tarefa.

Teremos por tanto que unha traza non é máis que unha agrupación de eventos para un mesmo caso. O obxectivo exposto nesta sección é identificar se as variantes do rexistro de referencia e do rexistro simulado son razoablemente parecidas. Para os indicadores explicados nesta sección non só se terá en conta que as variantes da simulación sexan iguais ás do rexistro de referencia, senón tamén que a frecuencia de cada patrón sexa parecido en ambos os casos.

Os indicadores utilizados aquí serán o recall, precision e F- score propostos en *van der Aalst(2018)*. Estes indicadores proveñen do campo da recuperación de información, onde a precision é a fracción de casos relevantes entre os casos recuperados e o recall é a fracción de casos relevantes que foron recuperados entre todos os casos realmente relevantes.

En *van der Aalst(2018)*, adáptase esta definición á minería de procesos onde o **recall** será a fracción de variantes permitidas polo modelo que foron observadas no rexistro de referencia e a **precision** a fracción de variantes observadas no rexistro simulado que tamén foron observadas no rexistro de referencia.

Por exemplo se todas as variantes que xera un modelo de simulación foron observadas no rexistro de referencia na mesma medida, a precisión será 1. Doutra banda, se no rexistro simulado podemos observar absolutamente todas as variantes posibles, teremos un recall de 1.

Para introducir os conceptos matemáticamente previamente precisaremos introducir distintos conceptos matemáticos onde moito de eles foron mencionados no Capítulo 2.

Definición 4.2 *(Traza) Sexa A o conxunto de todas as actividades, unha traza $t \in A^*$ é unha secuencia de actividades. Notaremos por $T = A^*$ e $|t|$ a lonxitude da traza.*

As trazas permitiranos defininir o concepto de rexistro de eventos que pode verse como o conxuto de todas as trazas que recollemos.

Definición 4.3 *(Rexistro de Evento) Sexa $L = B(T)$ o conxunto dos rexistros de eventos. Un rexistro de evento $l \in L$ é un multiconxunto de trazas observadas. Ademais tense que $\hat{l} = \{t \in l\}$ é o conxunto de trazas que aparecen en $l \in L$ e denotaremos por $|l|$ o número de trazas que hai no rexistro de eventos e por $l(t)$ o número de trazas t que hai no rexistro de eventos.*

O rexistro de eventos é o equivalente a mostra no contexto de inferencia, polo que será necesario relacionalo con un modelo cuxo conocemento é idealizado.

Definición 4.4 *(Proceso do Modelo) Un proceso do modelo $m \in M$ permitirá unha serie de trazas que denotaremos por $\hat{m} \subseteq T$. Denotaremos por M o conxunto de todos os procesos do modelo.*

O modelo pode ser representado en distintos lenguaxes como a Petri Nets que vimos no Capítulo 2, pero a nós nos interesará o comportamento que permite ese modelo, é decir, nos centraremos nas trazas que pode construír o modelo. Por exemplo un modelo que pode crear tres trazas distintas denotaríamolo por $\hat{m} = \{< a, b, c >, < a, b, d >, < a, c, d >\}$.

Neste momento xa temos todas a ferramentas para recoñecer a estructuras dos modelos que nos encontramos nos rexistros de eventos. Sen embargo parece lóxico ter en conta a hora de comparar os dous modelos que non todos as trazas son iguais de probableis polo que será necesario introducir unha compoñente de probabilidade de forma que cada traza terá asociada unha probabilidade de ocurrencia.

Definición 4.5 *(Función de probabilidade das trazas) Diremos que $\Pi : T \longrightarrow [0, 1]$ é unha función de probabilidade das trazas se para cada $\pi \in \Pi$ asígnase o valor $\pi(t) \in [0, 1]$ para toda traza $t \in T$ tal que $\sum_{t \in T} \pi(t) = 1$.*

Finalmente podemos combinar as dúas definicións anteriores para crear unha nova estructura coñecida como proceso estocástico do modelo onde nos aparecen todas as posibles trazas que pode crear o modelo e a probabilidade de que salga cada unha.

Definición 4.6 *(Proceso Estocástico do Modelo) Un proceso estocástico do modelo será un par* $s = (m, \pi) \in S$ *que combina o modelo* m *e a función de probabilidade das trazas* π*. Así notaremos por* $S = M \times \Pi$ *o conxunto de todos os posibles procesos estocásticos do modelo.*

Polo que agora xa temos a estructura necesaria para comparar os dous modelos. Neste caso temos que ter en conta que supoñemos que estamos nunha situación idealizada onde coñecemos tanto o modelo real $s_r = (m_r, \pi_r)$ como o modelo descuberto $s_d = (m_d, \pi_d)$.

Definición 4.7 *(Precision e Recall) Sexan* $s_r = (m_r, \pi_r) \in S$ *e* $s_d = (m_d, \pi_d) \in S$ *dous procesos estocásticos de dous modelos.(real e o descuberto). Entón terase:*

$$rec(s_r, s_d) = \frac{\pi_r(\hat{m}_d \cap \hat{m}_r)}{\pi_r(\hat{m}_r)} \qquad prec(s_r, s_d) = \frac{\pi_d(\hat{m}_d \cap \hat{m}_r)}{\pi_d(\hat{m}_r)}$$

Entón para calcular o recall o que se fai é comparar as trazas de ambos modelos $\hat{m}_d \cap \hat{m}_r$ coas trazas do modelo real $\hat{m}_r$ de forma que se utiliza para cuantificar esa cantidade π_r. Por outro lado para a precision o que se fai é comparar as trazas de ambos modelos $\hat{m}_d \cap \hat{m}_r$ coas trazas do modelo descuberto $\hat{m}_d$ de forma que se utiliza π_d para cuantificar esa cantidade. Sen embargo como comentamos antes no podemos coñecer esas cantidade no modelo real polo que *van der Aalst(2018)* propón utilizar unha estimación basándonos no rexistro de eventos como se pode ver na seguinte definición.

Definición 4.8 *Sexa* $l \in L$ *e* $s_l = (m_l, \pi_l) \in S$ *e* $s_d = (m_d, \pi_d) \in S$ *tal que* $\pi_l(t) = \frac{l(t)}{|l|}$ *para* $t \in T$ *e* $\hat{m}_l = \hat{l}$*. Entonces terase:*

$$rec(s_l, s_d) = \frac{\pi_l(\hat{m}_d \cap \hat{m}_l)}{\pi_l(\hat{m}_l)} \qquad prec(s_l, s_d) = \frac{\pi_d(\hat{m}_d \cap \hat{m}_l)}{\pi_d(\hat{m}_l)}$$

En resumo o cálculo de estos dous estatísticos consistirá en contabilizar o número de trazas nun só dos dous rexistros dependendo de si as trazas aparecen en ambas rexistros e dividilo polo número de trazas dun dos rexistros dependendo do estatístico. Estos estatísticos tomarán valores en [0,1] sendo 1 indicativo de un modelo moi bo na dimensión a que afecta o estatístico. Estes dous valores poden combinarse nunha única medida que denotamos por F- score e que consiste na media harmónica de ambos os dous valores obtendo unha medida máis xeral,

Definición 4.9 *Sexa* $l \in L$, $s_l = (m_l, \pi_l) \in S$, $s_d = (m_d, \pi_d) \in S$, $prec = prec(s_l, s_d)$ *e* $rec = rec(s_l, s_d)$*. Entón definimos o F-score como,*

$$\textit{F-score} = 2 \cdot \frac{prec \cdot rec}{prec + rec}$$

Estas medidas como comentamos nacen no contexto de recuperación de información, e aparecen no contexto do análisis estatístico da clasificación binaria que está intimamente relacionada cos tests de diagnóstico. En este contexo o recall é a cantidade de personas que foron etiquedas correctamente coa condición positiva entre todos os que teñen a condición positiva e polo tanto coincide coa sensibilidade. Por outro lado o precision é a cantidade de persoas que forons etiquetadas correctamente coa condición positiva entre todos os que foron etiquetados positivamente e por tanto coincide co valor positivo predictivo. Como vemos estas dúas medidas están relacionadas coas curvas ROC, xa que sin ir máis lexos o recall é a primeira dimensión da mesma. Neste contexto son importantes as medidas de resumo que nos indican como de boas son as clasificacións que se fixeron como poden ser a área baixo a curva ou F-score que presentamos neste apartado. Estas nos indicarán con un simple número como

funcionan o clasificador de forma global respecto as dimensións que o conforman. En xeral a ventaxa que se lle atribúe ao F-score con respecto a área baixo a curva é que esta área é a media de todos os posibles límites que se fixan para calcular a curva ROC o cal non é bo se temos datos desiquilibrados entre mostras positivas e negativas.

No contexto da clasificación binaria elíxese unha ou outra medida a raíz das consecunecias que ten unha boa ou mala clasificación según a condición do clasificado. No noso caso utilizamos o recall e o precision seguindo os mesmos razoamentos, así pois o recall terá máis importancia se a ocurrencia de un falso negativo é inaceptable e se utilizará o precision se tes moita confianza en que haxa persoas etiquetadas correctamente. No contexto dos modelos de minería de procesos nos centraremos na precision se consideramos importante que o noso modelo non cree trazas que non aparecen na realidade e nos centraremos no recall se creemos que o importante é que o modelo cree como mínimo as trazas que aparecen na realidade e non nos importe que cree novas variantes. Como medida de resumo de ámbos os dous teríamos o F-score que non é máis que a media armónica dos dous anteriores. En canto a esta medida, autores como *Hand e Christen(1967)* critican que esta medida dalle igual importancia tanto o precision como o recall e na práctica estos soen ter costes moi distintos.

Capítulo 5

Líneas futuras de traballo

Neste capítulo presentaremos algúns dos aspectos que deben ser obxecto de estudio para realizar distintas melloras no noso simulador. Así neste documento desenvolveuse o traballo realizado nas prácticas en Gradiant durante os meses de Agosto a Febreiro do ano 2018. Nelas comencei a desenvolver distintos algoritmos que simulaban distintas dimensións, que penso que poden ser mellorados xa que creo que existen aspectos a correxir ou perfeccionar. Primeiro de todo gustaríame partir da necesidade imperiosa de datos novos, pois a confidencialidade fai que sea moi difícil o acceso a estos provocando dificultades para desenvolver proxectos no ámbito médico. Dende o punto de vista estatístico gustaríame contar con datos de varios hospitais para comprobar se as hipóteses comprobadas para os datos dun hospital galego son extensibles a outros.

A parte da necesidade de conseguir máis datos para realizar comprobacións máis efectivas que radiquen na mellora do simulador, creo que hai partes do simulador que han de ser melloradas así como os algoritmos que desenvolvín. En canto a parte de minería de datos sería interesante ver como funcionan outros algoritmos e elexir en cada caso particular que algoritmo é mellor en función do recall, precision e F-score presentados aquí. A principal dificultade para esto é a gran cantidade de tempo necesaria para programar cada un destes algoritmos. Coa intención de evitar esta programación considero esencial realizar probas con ProM, ferramenta desenvolta pola unirversidade de Eindhoven, que incorpora a gran maioría dos algoritmos de minería de procesos.

En canto as melloras nos algoritmos desenvoltos neste TFM considero que moitos de estos ainda poden alcanzar un mellor desenvolvemento. O algoritmo no que traballei en primeiro lugar foi o de entrada de doentes, polo que considero máis completo. Durante a súa realización valorouse utilizar series de tempos como as propostas en *De Livera et al.(2011)*, sen embargo a lentitude da construcción das mesmas produxo que se tratará o problema dende o enfoque top-down descrito na Sección 3.2. Sen embargo creo que aínda se debe mellorar a estacionalidade por días pois os datos indica que existe pero o algoritmo non o está a captala dunha forma óptima. Unha mellor forma de tratar esto poder ser cambiando o límite proposto e realizar simulacións ata obter un límite óptimo.

Por outro lado a simulación do tempo foi a parte do traballo a que menos tempo lle puiden adicar polo que posiblemente é onde haxa maior marxe de mellora. Neste caso estamos moi condicionados polo axioma de que as distribución deben ser coñecidas polos médicos e que estas deben ter parámetros facilmente manexables. Se ben esto descarta por completo calquer metodoloxía non paramétrica, non se deben olvidar outros métodos máis complexos como modelos de mixtura que fan uso do algoritmo EM, de maneira que tratemos adaptalos para que o usuario só deba cambiar uns parámetros facilmente manexables.

Finalmente gustaríame destacar a necesidade de mellorar o algoritmo a hora de tomar decisións

sobre que actividade vendría despois de outra. Na sección 3.1 explicamos no punto 4 que se elixe a seguinte actividade seguindo a Petri Net, pero en ocasións temos que elixir entre dúas actividades como por exemplo na Figura 2.1. Neste caso se elexirá mediante a probabilidade que se lle asigna a cada actividade estimada a partir do rexistro de eventos. É decir, a cada actividade se lle asigna unha probabilidade de vir despois de outra que se calcula como o número de veces que aparece esa actividade despois da outra dividido entre o número de veces que veu calquera actividade. Esta forma de decidir que actividade ven despois é bastante sinxela polo que creo que pode ser mellorada utilizando modelos de regresión que teña en conta actividades pasadas ou outras variables como o color polo que foi triado ou as probas polas que pasou. Neste aspecto cando haxa unha mellora na calidade dos datos, de forma que se comence a incluir aspectos como as probas polas que pasa no laboratorio ou algúns dos síntomas que sofre o paciente creo que radicará nunha mellora importante no simulador.

Bibliografía

[1] Agrawal, R., Gunopulos, D., Leymann, F. (1998, March). Mining process models from workflow logs. In International Conference on Extending Database Technology (pp. 467-483). Springer, Berlin, Heidelberg.

[2] Babu, G. J., & Rao, C. R. (2004). Goodness-of-fit tests when parameters are estimated. Sankhya, 66(1), 63-74.

[3] Bermejo, R. S., Fadrique, C. C., Fraile, B. R., Centeno, E. F., Cueva, S. P., De las Heras Castro, E. M. (2013). El triaje en urgencias en los hospitales españoles. Emergencias: Revista de la Sociedad Española de Medicina de Urgencias y Emergencias, 25(1), 66-70.

[4] De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. Journal of the American Statistical Association, 106(496), 1513-1527.

[5] De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B. (2012). A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Information Systems, 37(7), 654-676.

[6] Durbin, J. (1976). Kolmogorov-Smirnov tests when parameters are estimated. In Empirical Distributions and Processes (pp. 33-44). Springer, Berlin, Heidelberg.

[7] Gold, E. M., Corporation, T. R. (1967). Language identification in the limit. Information and control, 10(5), 447-474.

[8] Gihman, I. I. (1961). On the empirical distribution function in the case of grouping data. Selected Translations in Mathematical Statistics and Probability, 1, 77-81.

[9] Hand, D., & Christen, P. (2018). A note on using the F-measure for evaluating record linkage algorithms. Statistics and Computing, 28(3), 539-547.

[10] Hartigan J.A. & Wong, J. H. (1979). a. MA Algorithm AS 136: A K-Means Clustering Algorithm. Appl. Stat, 28(1), 100-108.

[11] Kim, E., Kim, S., Song, M., Kim, S., Yoo, D., Hwang, H., Yoo, S. (2013). Discovery of outpatient care process of a tertiary university hospital using process mining. Healthcare informatics research, 19(1), 42-49.

[12] Khoussainov, B., Nerode, A. (1995). Automatic presentations of structures. In Logic and computational complexity (pp. 367-392). Springer, Berlin, Heidelberg.

[13] Kolmogorov, A. (1933). Sulla determinazione empirica di una lgge di distribuzione. Inst. Ital. Attuari, Giorn., 4, 83-91.

[14] Kim, E. S. (2013). Using computer simulation to study hospital admission and discharge processes.

[15] Lilliefors, H. W. (1967). On the Kolmogorov-Smirnov test for normality with mean and variance unknown. Journal of the American statistical Association, 62(318), 399-402.

[16] Lilliefors, H. W. (1969). On the Kolmogorov-Smirnov test for the exponential distribution with mean unknown. Journal of the American Statistical Association, 64(325), 387-389.

[17] Mans, R. S., Schonenberg, M. H., Song, M., van der Aalst, W. M., Bakker, P. J. (2008, January). Application of process mining in healthcare a case study in a dutch hospital. In International joint conference on biomedical engineering systems and technologies (pp. 425-438). Springer, Berlin, Heidelberg.

[18] Massey Jr, F. J. (1951). The Kolmogorov-Smirnov test for goodness of fit. Journal of the American statistical Association, 46 (253), 68-78.

[19] Natrella, M. (2010). NIST/SEMATECH e-handbook of statistical methods.

[20] Parsons, F. G., & Wirsching, P. H. (1982). A Kolmogorov-Smirnov goodness-of-fit test for the two-parameter weibull distribution when the parameters

[21] Petri, C. A. (1962). Kommunikation mit automaten.

[22] Smirnov, N. (1939). Sur les écarts de la courbe de distribution empirique. Matematicheskii Sbornik, 48(1), 3-26.

[23] Souto-Ramos, A. I. (2008). Aclaraciones sobre los sistemas de triaje en urgencias. El sistema de triaje Manchester. Enfermería Clínica, 18(5), 284-286.

[24] Tiwari, A., Turner, C. J., Majeed, B. (2008). A review of business process mining: state-of-the-art and future trends. Business Process Management Journal, 14(1), 5-22.

[25] van der Aalst, W. M. (2011). Process Mining Discovery, Conformance and Enhancement of Business Processes.

[26] van der Aalst, W. M. (2018, July). Process mining and simulation: a match made in heaven!. In Proceedings of the 50th Computer Simulation Conference (p. 4). Society for Computer Simulation International.

[27] Wang, J., Tan, S., Wen, L., Wong, R. K., Guo, Q. (2012, March). An empirical evaluation of process mining algorithms based on structural and behavioral similarities. In Proceedings of the 27th Annual ACM Symposium on Applied Computing (pp. 211-213). ACM.

[28] Weber, P., Bordbar, B., Tino, P. (2013). A framework for the analysis of process mining algorithms. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(2), 303-317.

[29] Weijters, A. J. M. M., Ribeiro, J. T. S. (2011, April). Flexible heuristics miner (FHM). In Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on (pp. 310-317). IEEE.

[30] Wilcoxon, F., Katti, S. K., & Wilcox, R. A. (1970). Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. Selected tables in mathematical statistics, 1, 171-259.

[31] Yue, D., Wu, X., Wang, H., Bai, J. (2011, May). A review of process mining algorithms. In Business Management and Electronic Information (BMEI), 2011 International Conference on (Vol. 5, pp. 181-185). IEEE.