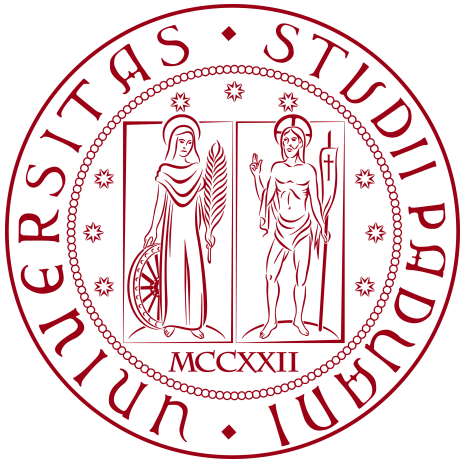
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Magistrale in Informatica



Stima dei timestamp di inizio delle attività di processo tramite simulazione

Tesi di Laurea Magistrale

Relatore

Prof. De Leoni Massimiliano

Studente

Davide Farinelli

Matricola 2005758

Anno Accademico 2023-2024

Ringraziamenti

Desidero esprimere la mia gratitudine al professor De Leoni Massimiliano, mio relatore, per l'aiuto e il sostegno che mi ha dato durante la stesura dell'elaborato.

Vorrei anche ringraziare, con affetto, i miei genitori per il loro sostegno, il grande aiuto e la loro presenza in ogni momento durante gli anni di studio.

Desidero poi ringraziare i miei amici per i bellissimi anni trascorsi insieme e le mille avventure vissute.

Padova, Luglio 2024 *Davide Farinelli*

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei codici sorgenti

Capitolo 1

Introduzione

Il presente elaborato mira a fornire una panoramica nel vasto universo del Process Mining. Infatti, tale materia - collocata equidistantemente fra i settori del machine learning, e della Business Process Intelligence - ha come obiettivo quella di scoprire, monitorare e migliorare i processi aziendali al fine di garantirne migliore qualità dei servizi o prodotti [5].

Il principale requisito per l'applicazione di tecniche di *Process Mining* è la disponibilità di un *event log*. Un *event log* può essere definito come un contenitore di sequenze di eventi, ogniuna di queste sequenze identifica una completa esecuzione del processo. Spesso gli *event log* contengono solamente gli eventi relativi al completamento delle attività, con i relativi *timestamp*; in questi casi non è possibile calcolare le durate delle attività e quindi, non si è in grado di comprendere dove siano gli eventuali colli di bottiglia. Questa tesi si è focalizzata sullo sviluppo di tecniche per la stima dei *timestamp* di inizio delle attività.

Per fare, ciò si è partiti da un *event log* e da un modello di simulazione, poi utilizzando una lista di profili di durata delle attività (alpha) sono stati inseriti i *timestamp* di inizio delle attività all'interno del log. Con questo log è possibile calcolare la probabilità di durata delle attività che vengono inserite nel modello di simulazione. Il modello di simulazione viene poi usato per generare un *event log* simulato. Infine, la tecnica restituisce un log a cui sono stati inseriti i *timestamp* di inizio delle attività tale che il modello di simulazione

corrispondente generi un log simulato il più possibile simile al log originiario.

La tecnica è stata valutata utilizzando un *event log*, con al suo interno le informazioni relative ai *timestamp* di inizio e di fine attività. Per una corretta valutazione, ed al contempo verificare l'efficienza della tecnica, sono poi stati rimossi i *timestamp* di inizio e, successivamente, si è cercato di scoprirli nuovamente. L'esito della valutazione mostra che la tecnica utilizzata è stata in grado di riscoprire i *timestamp* di inizio più precisi, piuttosto che utilizzando tecniche preesistenti, le quali presuppongono l'assenza di tempi di attesa.

Organizzazione del testo

Il presente elaborato è stato organizzato come di seguito:

Capitolo due fornisce una panoramica delle fondamentali nozioni della disciplina del Process Mining, al fine di permettere una maggiore comprensione del documento, quali ad esempio la definizione di *event log* e di simulazione BP, e altresì fornisce una definizione del file Json contenente i parametri per le simulazioni.

Capitolo tre fornisce una panoramica dello stato dell'arte relativo alla stima dei *time stamp*, fornendo altresì esempi delle tecniche esistenti per la loro stima.

I capitoli quattro e cinque presentano i risultati circa l'efficenza degli algoritmi posti alla base del caso di studio, cioè *random search* e *bayesian optimization*, per la ricerca dei valori di alpha, traendo conclusioni circa il grado di efficenza e qualità delle due tecniche.

Capitolo 2

Nozioni Fondamentali

2.1 Process Mining

Il settore di ricerca noto come "*Process mining*" mira allo studio delle analisi dei processi di business, ponendosi come obbiettivo quello di "estrarre" informazioni da dati provenienti e collezionati da sistemi di informazione e, da tali elementi, individuarne le informazioni al fine di migliorare e modellare gli stessi processi produttivi e permettere il raggiungimento di massima efficienza di un dato processo (ad esempio, suggerendo contro-misure o identificando possibili miglioramenti). Appare, dunque, evidente come tale materia ben si possa collocare a cavallo tra i settori del data science, business process modelling ed analisi. Per fare ciò, tale tecnica di process management si fonda su un pilastro, attorno al quale si sviluppa appunto l'intera branchia del Process Mining: il log degli eventi (*event log*), cioè file strutturati contenenti le registrazioni e tempistiche di eventi ed attività all'interno di un database.
Al fine di comprendere appieno tale ultima definizione, tuttavia, è opportuno effettuare una necessaria premessa logica. In via generale, si ritiene che sia possibile individuare, in ogni tipo di processo produttivo, le varie attività sequenzialmente condotte ed in esso contenute.
Queste sequenze sono custodite all'interno dei c.d. *event log*, i quali fungono da registri delle singole, passate attività svolte all'interno di un dominio applicativo. In essi, ogni operazione viene registrata e catalogata quale "evento",

rappresentandone questa un'esecuzione (e solitamente correlata da informazioni addizionali quali la risorsa o i c.d. *timestamps*).
I *timestamp*, a loro volta, svolgono un compito determinante, in quanto custodiscono la cronologia degli eventi e conseguentemente il modo in cui essi si ordinano in un processo. Nel settore del process mining, i timetamps forniscono una cartina tornasole sulla performance del processo, la sua conformità e modellazione. Gli *event log* servono da base nelle tre tipologie di utilizzo del *Process Mining* come mostrato in Figura 2.1:

- *Discovery*: le tecniche discovery utilizzano differenti algoritmi, i quali considerano gli event log quali "input" di partenza, e da ciò restituendo un flusso di lavoro senza necessitare di ulteriori informazioni aprioristiche.
  In altre parole, "consente l'estrazione (automatica) di un modello di processo a partire da un log" [6].
  Tale tecnica, pertanto, si rivela particolarmente utile al fine di poter comprendere un processo nella sua totalità, e dunque i suoi comportamenti;

- *Conformance checking*: lo scopo di tale tecnica è quella di comparazione, ovvero porre a confronto un esistente modello di processo con un *event log* proveniente dal medesimo processo.
  Una verifica della conformità può condurre in maniera effettiva a quantificare e diagnosticare delle discrepanze; in tal modo, permettendo all'osservatore di condurre un'analisi su un determinato modello produttivo e i suoi comportamenti.

- *Miglioramento (enhancement)*: mentre le precedenti tipologie vengono perlopiù utilizzare per misurare l'allineamento tra un modello astratto e la realtà, lo scopo principale dell'enhancement è quello di migliorare un modello (ad esempio, risolvendone gli errori).
  Inoltre, tale tipologia consente altresì di arricchire il processo con informazioni aggiuntive su risorse, regole decisionali, metrica, ecc...

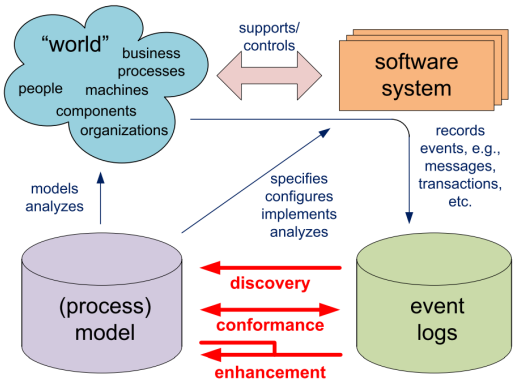


Figura 2.1: Posizionamento delle tre principali tipologie di Process Mining: discovery, conformance, and enhancement

Case	Activity	Time:timestamp	Lifecycle
0	Create Purchase Requisition	2011-01-01 06:37:00	complete
0	Create Request for Quotation	2011-01-01 11:45:00	complete
0	Analyze Request for Quotation	2011-01-01 12:55:00	complete
0	Send Request for Quotation	2011-01-01 18:09:00	complete
0	Create Quotation comparison Map	2011-01-01 22:03:00	complete
0	Choose best option	2011-01-02 05:13:00	complete
0	Settle Conditions w. Supplier	2011-01-02 15:20:00	complete
0	Create Purchase Order	2011-01-02 16:10:00	complete
0	Confirm Purchase Order	2011-01-02 20:43:00	complete
0	Deliver Goods Services	2011-01-03 09:37:00	complete
0	Release Purchase Order	2011-01-03 17:21:00	complete
0	Approve Order for payment	2011-01-04 01:10:00	complete
0	Send Invoice	2011-01-04 06:54:00	complete
0	Release Supplier's Invoice	2011-01-04 21:13:00	complete
0	Authorize Invoice payment	2011-01-04 21:13:00	complete
0	Pay Invoice	2011-01-04 21:31:00	complete

Tabella 2.1: Tabella riportante un campione dell'event log

2.2 Event Logs

Come già precisato nel precedente paragrafo, le tecniche di *process mining* estraggono conoscenza dagli *event log*, allo scopo di modellare, monitorare ed auspicabilmente migliorare i processi di *business*.

Pertanto, appare evidente come gli *event log* consistano nel vero e proprio "punto di partenza" del *Process Mining*. Questi possono essere definiti quali

contenitori di sequenze di eventi, dette tracce. Ogni traccia identifica una completa esecuzione del processo. Gli eventi possono essere rappresentati come una tupla di:

- *Case-ID* identificativi del *trace* di appartenenza dell'evento;
- Attività che identificano gli step/azioni del processo
- *Timestamp* indica quando ogniuna delle attività ha avuto luogo, definendone l'ordine nell'event log
- Risorsa (che svolge l'attività)
- *Life-cycle information* la quale può essere rappresentata dalle due componenti *start* (*token* entra in un elemento nel modello BPMN) e *complete* (ove il *token* indica che lo stesso sta uscendo dall'elemento).
- Altre informazioni

Nella Tabella 2.1 è riporato un esempio di event log con una sola traccia dove ogni riga rappresenta un evento con dei suoi rispettivi attributi.

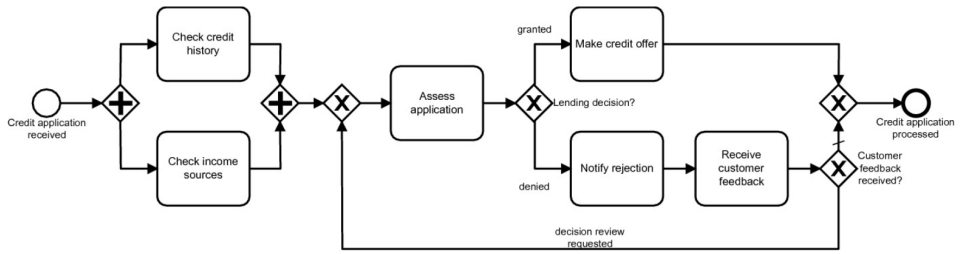


Figura 2.2: Esempio di un diagramma BPMN con dei gateway esclusivi ed inclusivi e sei tasks

2.3 Modello BPMN

Il *Business Process Model and Notation* (BPMN) è una rappresentazione grafica utilizzata al fine di meglio gestire i processi aziendali all'interno di un business process modeling (BPM).
Il focus ed obbiettivo primario del BPMN è quello di fornire una nozione facilmente comprensibile a tutti gli utenti del business, rivolgendosi a questi per acquisire comprensione attraverso una rappresentazione visiva dei vari passaggi produttivi, rendendone così più accessibile la fruizione. Su un piano di coinvolgimento più ampio, si rivolge agli utenti che implementeranno il processo, fornendo dettagli sufficienti per consentire un'implementazione precisa (appunto, dall'analista che ne crea la bozza iniziale fino ai tecnici responsabili del perfezionamento dei processi).
Gli elementi fondamentali di un BPMN sono le attività (o operazioni, siano esse eseguite da persone fisiche o sistemi), i flussi sequenziali (indicatori dell'ordine delle attività da svolgere, solitamente rappresentato con una linea retta o freccia) ed i gataway (c.d. punti di decisione idonei a modellare il percorso del processo). La Figura 2.2 riporta un esempio di un diagramma BPMN.

2.4 Simulazione

Le tecniche per la stima dei *timestamp* proposte in questa tesi si basano sulla costruzione dei modelli di Business Process Simulation (BPS), tecnica di analisi quantitativa dei processi di business.

Essa è ampiamente utilizzata nell'ambito dello studio dei processi economici e commerciali, in quanto permette di generare set di simulazione di possibili tracce esecutive di processi.
In particolare, la simulazione consiste in una metodologia sistematica, il cui scopo è quello di rappresentare il comportamento di un modello simulando - appunto - un *business process* utilizzando parametri reali. Di conseguenza, si avrà altresì l'analisi delle perfomance in via predittiva; mediante le simulazioni, infatti, si potrà predire, con un certo grado di certezza, come un dato processo si comporterà una volta implementato nell'organizzazione.
Attraverso l'utilizzo dei BPS è dunque possibile condurre l'analisi di vari processi produttivi, al contempo permettendone il perfezionamento.

Nello studio condotto in questo elaborato, si è utilizzato una tecnica di *process simulation* allo scopo di simulare un modello di processo, generando un *event log* al fine di confrontarlo con il log originario.
Il primo passaggio necessario al fine di condurre tale simulazione è consistito, in particolare, nella somministrazione di informazioni necessarie e contenute nel c.d. *event log* "originale", dal quale la simulazione origina.
A partire dunque dall'*event log*, sarà possibile definire un modello ed i suoi parametri.

2.5 Simulazione con Prosimos

In questa tesi, è stato utilizzato Prosimos [4] come motore di BPS. Prosimos è uno strumento di simulazione *open source* che implementa un approccio alla simulazione BP mediante risorse differenziate. In Prosimos, le risorse infatti non vengono raggruppate in pool, ma trattate individualmente, ognuna con il proprio profilo-risorsa.
Ciò stante, la performance di ogni singola risorsa risulterà indipendente rispetto ad ogni altra (*differentiated performance*), e – per lo stesso motivo – ogni risorsa potrà (almeno in astratto) possedere un proprio, indipendente calendario di disponibilità (*differentiated availability*).

Contrariamente ai classici modelli di simulazione BP, le attività del processo potranno essere assegnate a molteplici e differenti profili di risorse e, viceversa, al tempo stesso differenti risorse potranno condividere lo stesso profilo di risorsa.

Una simulazione condotta mediante Prosimos produce, quale risultato finale un *event log* della simulazione; di conseguenza, attenzione particolare dovrà porsi sui c.d. *timestamp*, in quanto le criticità in essi originati potranno condurre a rappresentazioni distorte ed erronee del processo minato.

Prosimos richiede due input: un modello BPMN ed un ulteriore file, in formato Json, con le informazioni associate ai parametri di simulazione (quali ad esempio *arrival time distribution*, *resource task associations*, calendario, ecc..) sono separatamente specificati nel c.d. file Json. Il file in questione, in particolare, è suddiviso a sua volta in sei sezioni:

- *Resource profiles*: esso contiene le informazioni delle risorse, raggruppati in c.d. pool. Più nel dettaglio, esso raggruppa un set di Pool di risorse, nel quale ogni Pool è rappresentato da un suo ID (consistente nel nome e nella lista risorsa).

- *Arrival Time Calendar*: esso è relativo agli interva lli di tempo nei quali nuovi processi possono essere iniziati. In particolare, tali intervalli di tempo sono scanditi su base settimanale, aventi quale beginTime un singolo giorno della settimana (ad esempio martedì), ad un determinato orario di partenza, ed un rispettivo endTime, esistente in un giorno della settimana e di un orario predeterminato.

- *Arrival Time Distribution*: esso consiste in una funzione di distribuzione di probabilità che descrive come un nuovo processo viene inizializzato all'interno dell'arrival calendar.

- *Gateway branching probabilities*: se rappresenta la probabilità che un flusso di esecuzione ha di proseguire verso un qualsiasi flusso esecutivo ad ogni split gateway del processo.

- *Task resource distribution*: esso permette la mappatura di ogni task del modello di processo e la lista di risorse che può eseguirla. Per ogni task,

ognuna definita dalla propria ID, è presente una lista di risorse in grado di eseguire tale compito, e la funzione di distribuzione delle probabilità (per ogni singola risorsa) che descrive la sua durata. Tale motivo, non stupisce come la la funzione di distribuzione varia per ogni risorsa. Tale sezione, nel caso di specie, è stata modificata prima di ogni simulazione condotta, al fine di individuare i reali start timestamp (come meglio evidenziato al successivo capitolo 4).

- *Resource calendars*: sarà presenta la lista di intervalli di tempo i quali definiscono quando una risorsa è disponibile ad eseguire un dato compito su base settimanale. Ogni intervallo di calendario è costituito da un dato beginTime (come nel caso dell'arrival time calendar, tradotto in un giorno della settimana ed un determinato orario) e di un endTime (sempre costituito da un giorno della settimana ed un dato orario).

- *Case attributes*: con ciò si intende la descrizione di quali attributi dovrebbero essere generati durante la simulazione e che valori essi dovrebbero possedere.

- *Batch processing*: con ciò si intende una lista di task che possono essere eseguiti insieme sulla base di predeterminate regole.

Capitolo 3

Stato dell'arte

Questo capitolo introduce la tecnica proposta in per la stima dei *timestamp*, utilizzata come punto di partenza in questa tesi [3].
Gli *event log* ed i modelli di simulazione costituiscono, dunque, le fondamenta ed il punto di partenza per la stima degli *start timestamps* della durata delle attività computata. I timestamps infatti si rammenta, sono un requisito di base per un record di dati utilizzato per un'analisi di process mining, infatti, determinano la cronologia di quali eventi si sono verificati e successivamente come sono stati ordinati nella modellazione dei processi
Un modello simulato consiste in un tuple M = (N , S), ovvero risulta composto da un modello di processo di business N (e.g., a BPMN model), ed un set di parametri (S) atto a definire le specificazioni del processo sotto diversi profili (tempo, risorse, decisioni, ecc. . .).

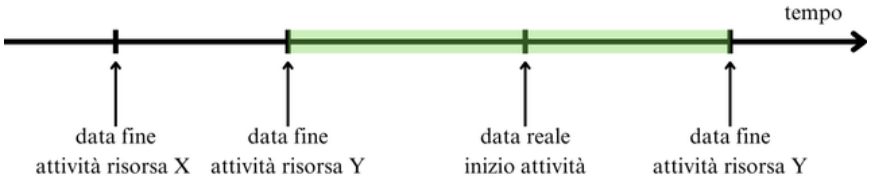


Figura 3.1: Schema che rappresenta i concetti di evento precedente in una traccia tramite control-flow e precedente evento eseguito da una risorsa. Per l'evento c, data(c) è il timestamp relativo all'evento c, e res(c) = Y la risorsa che esegue l'attività act(c). Al contempo a è l'evento precedente in un trace seguendo il control-flow ed b il precedente evento eseguito da una risorsa con res(b) = res(c) = Y . Il segmento verde rappresenta l'intervallo temporale in cui l'evento e potrebbe iniziare datainizio(c) [data(b), data(c)]

3.1 Intervallo in cui un evento potrebbe iniziare

Dato un *event log* privo di *timestamp* di inizio delle attività si vuole costruire un nuovo *event log* con i *timestamp* di inizio.
Per ogni *trace* σ del *log* di partenza, il nuovo *log* avrà un *trace* σ' che conterrà ogni evento di σ.
Quindi in ogni evento di σ' se $lifecycle = start$ dovrà essere presente l'informazione *timestamp*.
La stima del timestamp relative al momento di inizio attività richiede il delinearsi del "*minimum timestamp*", ovvero il momento in cui l'attività avrebbe potuto potenzialmente iniziare, considerando i vari vincoli dei processi stessi.
Questo tipo di computazione viene poi astratta nel *minimum timestamp oracle*.
In particolare, tale intervallo troverà la sua dimensione a partire da un minimo (in altre parole, la data più recente tra quella in cui la risorsa che ha svolto l'attività si è liberata e la data in cui il precedente evento è terminato) ed un massimo (corrispondente alla data in cui l'evento corrente si interrompe, terminando). In Figura 3.1 è stato evidenziato in verde il range temporale in cui un evento potrebbe iniziare.

Ora che è stata stabilita la tecnica con cui gli intervalli di tempo in cui gli eventi potrebbero essere inizati e possible stimare gli *start timestamp*.
La tecnica per la stima dei *timestamp* di inizio delle attività si basa sullo scoprire i valori *alpha* dove *alpha* è una lista di valori compresi tra zero e uno e per ogni evento stimo il *timestamp* di inizio seguendo la formula:

$$timestamp(e') = \alpha(e) \cdot mintime(e) + (1 + \alpha(e)) \cdot timestampcomplete(e)$$

Mintime è una funzione mediante la quale, dati gli eventi di un *log*, si ricava il range da cui, tramite i valori di *alpha*, è possibile stimare il c.d. *start timestamp*.
Più precisamente, l'estremo sinistro del range viene scelto selezionando il più

recente *timestamp* tra due distinti valori: l'ultimo momento in cui la risorsa che sta svolgendo l'attuale attività si è liberata, e la conclusione dell'attività precedente.

Ciò posto, se un valore di *alpha* è uguale a zero, lo *start timestamp* sarà uguale all'estremo destro (quando l'attività sarà dunque conclusa).
Se, invece, un valore di *alpha* è uguale a 1, lo *start timestamp* sarà uguale al suo estremo sinistro.

Amend Purchase Requisition	0.75
Amend Request for Quotation	0.18
Analyze Purchase Requisition	0.59
Analyze Quotation Comparison Map	0.82
Analyze Request for Quotation	0.89
Approve Purchase Order for payment	0.77
Authorize Supplier's Invoice payment	0.38
Choose best option	0.33
Confirm Purchase Order	0.73
Create Purchase Order	0.57
Create Purchase Requisition	0.85
Create Quotation comparison Map	0.02
Create Request for Quotation	0.47
Deliver Goods Services	0.25
Pay Invoice	0.61
Release Purchase Order	0.69
Release Supplier's Invoice	0.48
Send Invoice	0.26
Send Request for Quotation to Supplier	0.56
Settle Conditions With Supplier	0.82
Settle Dispute With Supplier	0.45

Tabella 3.1: Tabella riportante i valori iniziali di alpha

Tramite la funzione sopracitata, è possibile stimare gli *start timestamp*.
Inizialmente, è bene sottolineare come, all'inizio del processo, essendo i valori di *alpha* ignoti, si utilizzano valori randomici compresi tra 0 ed 1; tramite questi, si stimeranno così gli *start timestamp*. Nella tabella 3.1 è possibile osservare un esempio di configurazine di valori *alpha*.

A seguire, con il nuovo *log* si dovranno calcolare le distribuzioni di probabilità per ogni attività.

Tramite *Prosimos*, a quel punto si avvierà una simulazione in cui vengono inserite le informazioni sulle distribuzioni.
Di conseguenza, come risultato si otterrà un *log* simulato.

In questo stadio, dunque, si avranno due distinti *log*: il primo $\mathcal{L}_{orig}$, l'originario privo di *start timestamp*, ed il secondo $\mathcal{L}_{sim}$, quello simulato con *Prosimos*, come poc'anzi esposto.
A questo punto, si dovrà vagliare se e quante differenze intercorrono tra i due distinti *log*; per fare ciò, verrà utilizzata una funzione *activity distribution distance*, la quale per ogni attività calcolerà la distanza intercorrente tra il momento il cui l'attività precedente è stata terminata e il momento in cui l'attività presa in considerazione è stata ultimata.

Successivamente, tramite la funzione *Wasseinstein distance*, si porranno a confronto le liste di distanze ottenuta dai due distinti *log*, evidenziandone e quantificandone la differenza.
Ciò in quanto la metrica di Wasserstein consiste in una distanza nell'insieme delle misure di probabilità, sviluppata all'interno della teoria del trasporto ottimale [2].

La Wasserstein distance, nota altresì come "*the Earth mover's distance o the optimal transport distance*" consiste in una similarità metrica tra due distribuzioni di probabilità. Nei casi discreti, essa può essere intesa come il costo di un piano di trasporto ottimale al fine di converire una distribuzione in un'altra.
Il costo, in questi casi, è calcolato man mano che il prodotto della quantità di massa probabilistica mossa e la distanza al quale questo viene trasportato.
Con tale dato, sarà possibile modulare i valori di *alpha*, reiterando progressivamente tale processo, fino ad ottenere una configurazione che stimerà degli *start timestamp* tali da riflettere quelli che sarebbero stati gli *start timestamp* reali.
A tal riguardo, nei prossimi capitoli verrano approfondite le tecniche utilizzare per portare avanti la ricerca dei valori di *alpha*.

Capitolo 4

Tecniche per la stima dei timestamps

4.1 Modello utilizzato e caso di studio

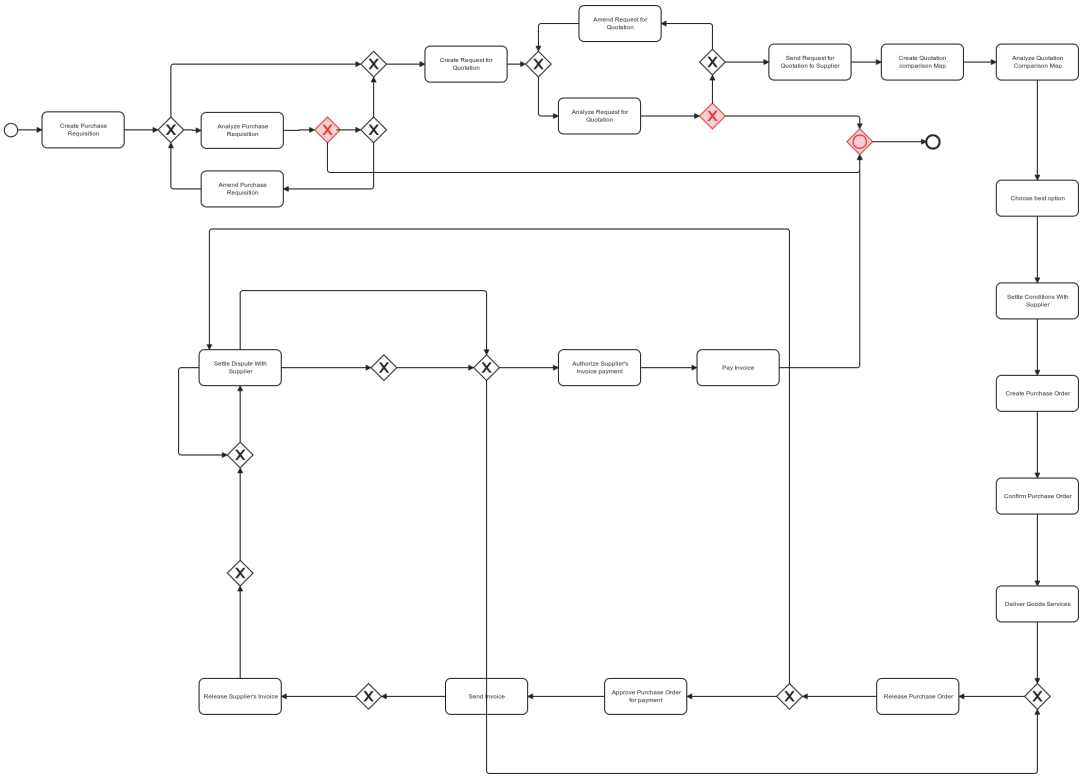


Figura 4.1: BPMN utilizzato

Il primo elemento per lo sviluppo del progetto è il modello BPMN riportato in Figura 4.1. La simulazione è stata portata avanti partendo dal c.d. process model; a partire da ciò, la simulazione dei parametri è stata recuperata partendo dalla funzione Prosimos "*discover a simulation scenario from the log*" (scoprire uno scenario di simulazione a partire da un log).
Mediante la stessa, quindi, è stato di fatto possibile individuare le informazioni che, a seguire, verranno utilizzate per la genesi dei c.d. log simulati, quali:

- *Branching probabilities* che rappresentano la probabilità per il processo di esecuzione di andare verso un qualsiasi flusso in uscita da ogni split gateway (inclusivo o esclusivo) nel process model;
- *Arrival Time Calendar*;
- *Resource Calendar*, ovvero gli intervalli di tempo in cui le risorse sono disponibili per compiere un task su base settimanale;
- *Resources*;
- *Resource allocation* il quale mappa ogni task del process model e la lista di risorse che lo possono svolgere

Tutte queste infomazioni sono poi state usate per svolgere delle simulazioni; di essi, poi, sono stati confrontati risultati con il log originario ed i parametri sono stati aggiustati al fine di ottenere una simulazione riflettente al meglio il log di partenza.
La migliore configurazione dei parametri è stata poi usata per le future simulazione.

4.2 Modulazione valori di alpha via *Random Search*

In questa sezione, si analizzerà la tecnica con cui si modulano i valori di alpha (*alpha values*) finalizzati ad ottenere la stima degli *start timestamp*. Nelle

fasi preliminari, si era valutato di implementare una ricerca dei valori di alpha tramite un algoritmo greedy.

Più precisamente, mediante la definizione "algoritmo greedy" si fa riferimento ad un paradigma algorimico il quale adotta, al fine di ricercare la soluzione ottimale ad un problema, una strategia di tipo euristica di problem solving.
Mediante tale tecnica, la quale si fonda sull'assunto che per individuare una soluzione globalmente ottimale si dovranno trovare soluzioni ottime localmente, l'algoritmo opterà, in ogni passaggio del procedimento, all'ottenimento della soluzione, a livello locale, più adeguata (appunto, ottimale).
In altre parole, gli algoritmi "golosi" effettuano, ad ogni passaggio del procedimento in esame, la scelta che in quel preciso momento si rivela migliore (localmente ottimale), al fine di raggiungere una soluzione ottima a livello globale.

Pertanto, non stupisce come tale algoritmo venga implementato al fine di trovare una soluzione ai tipici problemi di ottimizzazione quali l'allocazione di risorse, la pianificazione, ed il problema del c.d. *scheduling* (ovvero la necessità di ordinare temporalmente le attività).
Tale algoritmo, quindi differisce radicalmente dalla programmazione dinamica e si dimostra più efficace, in quanto generatore di una sola sequenza decisionale (al contrario, il metodo dinamico può, in astratto, portare alla produzione di svariate sequenze decisionali).

Si è ritenuto dunque di procedere utilizzando tale tipologia di algoritmo, con la quale la ricerca dei valori si effettuava modulando *alpha* tramite step di variazioni progressive.

```
input : Event log : L .
input : Simulation model : M = ( Business process model N,
                                 Simulation specification S ).

start_range = max( Previous event in trace,
                   Previous event performed by a resource )
alpha = set_initial_alpha()
L1 = add_start_event (L , alpha , start_range )
for act in activities(N) do
    new_task_resource = modify_task_resource(L1, S(task_resource),
                                             act)
    S( task_resource ) = new_task_resource
endfor
Lsimu = run_simulation(M)
distance = compute distance( L, Lsimu )
best_alpha = find_best_alpha(L, distance, alpha, S(task_resource),
                             0, 1, 0)
return best_alpha
```

Codice 4.1: Algoritmo che preso in input il log originario e il modello di simulazione restituisce i migliori valori di alpha trovati. I valori di alpha verranno poi usati per il calcolare gli start timestamp

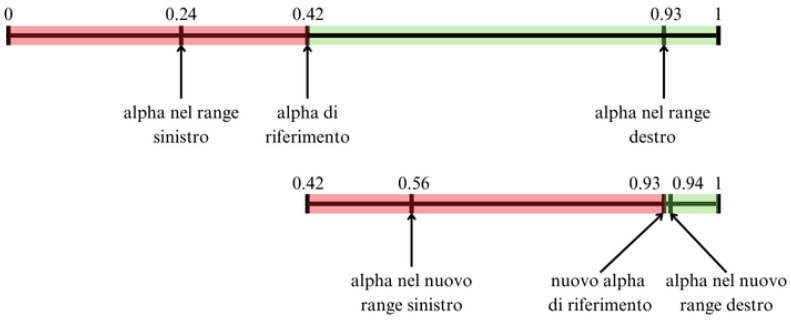


Figura 4.2: Tecnica usata per la ricerca dei valori di alpha

Al fine di diminuire il numero complessivo degli step compiuti, e dunque effettuando meno passaggi intermedi, si è scelto di utilizzare un algoritmo il quale selezionava step in via randomica.

A partire dai valori di alpha selezionati casualmente, due nuovi valori venivano scelti - sempre stocasticamente - rispettivamente, a destra e sinistra dello stesso. In Figura 4.2 viene mostrato come da un valore di alpha (alpha di riferimento) vengano poi trovati due nuovi valori di alpha (destro e sinistro) con cui vengono calcolati gli *start timestamp* di due log distinti.
Da questi ultimi, si ricaveranno le nuove distribuzioni di probabilità.

```
Function modify_task_resource( L, S(task_resource), act )
    for res in resources(N) do
        duration = duration_by_resource( L, act, res )
        new_task = get_best_fitting_distribution( duratio n )
        task_resource[act][res] = new_task
    endfor
return task_resource
end
```

Codice 4.2: Funzione che per ogni risorsa (res) e per ogni attività (act) che quella risorsa svolge, trova la distribuzione che meglio rispecchia la distribuzione delle durate delle attivita act quando viene svolta da res

Le distribuzioni di probabilità vengono calcolate tramite una funzione chiamata *modify task resource* (Codice 4.2), che ricerca all'interno del *log* tutte le durate complessive tra il momento di inizio (*start timestamp*) di una determinata attività svolta da una determinata risorsa e la sua ultimazione (*complete timestamp*) e infine per ogni lista di durate viene trovata la distribuzione di probabilità tramite la funzione *get best fitting distribution*.

Le distribuzioni vengono poi inserite nel modello di simulazione; così facendo, si ricavano due nuove simulazioni, dalle quali si ottengono, rispettivamente, la distanza tra il log originario ed il log simulato ottenuto inserendo la distribuzione di probabilità ricavata usando l'*alpha* a destra e la distanza tra il log originario ed il log simulato ottenuto inserendo la distribuzione di probabilità ricavata usando *alpha* a sinistra.

A seconda dei valori ottenuti, se nessun miglioramento è riscontrabile, il valore *alpha* di partenza viene mantenuto, spostando l'analisi al successivo valore di *alpha*.

Se invece almeno una delle distanze tra i log derivanti da tale operazione si rivela essere minore in via assoluta, ricorsivamente l'*alpha* generante il log con distanza minore diventerà a sua volta l'alpha di partenza; al contempo, l'*alpha* originario (di partenza) diventerà il nuovo estremo (sinistro o destro che sia) dell'intervallo ove si contierà il processo di ricerca di un migliore *alpha*.
Come meglio illustrato nella parte inferiore della Figura 4.2.

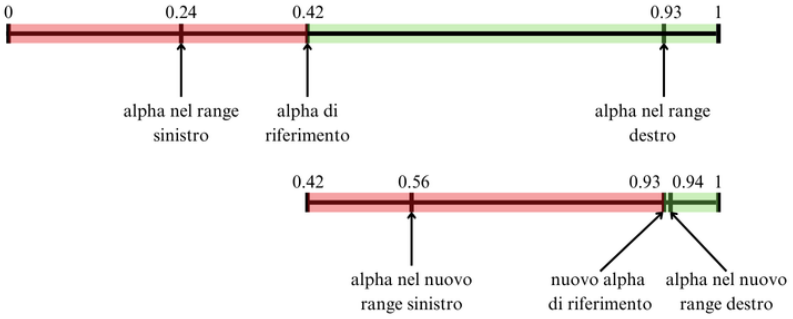


Figura 4.3: Tecnica usata per la ricerca dei valori di alpha

```
Function compute distance( L, Lsimu )
  for act in activities(N) do
    for events in L do
      if (events(lifecycle) = complete) & (events(activity) = act)
        end_time = event(timestamp)
        start_time = max(Previous complete event in trace,
                         Previous event performed by res)
        times += end_time - start_time
      endif
    endfor
    for events in Lsimu do
      if events(lifecycle) = complete & events(activity) = act
        end_time_simu = event(timestamp)
        start_time_simu = max(Previous complete event in trace,
                              Previous event performed by res)
        times_simu += end_time_simu - start_time_simu
      endif
    endfor
    distance += Wasserstein distance(times, times_simu)
  endfor
  return mean(distance)
end
```

Codice 4.3: Funzione usata per il calcolo della distanza tra il log originale e il log simulato

Per calcolare la distanza tra il log originale e il log simulato è stata usata la funzione *compute distance* (Codice 4.3) dove per ogni attività tale che *attività(e) = act*, è necessario visitare il log originario e il log simulato e calcolarne la distanza tra i due.

Per far ciò, si dovrà prendere ogni evento del log che si sta visitando; se Life-

cycle information è uguale a complete, e *attività(event e) = act* allora si dovrà calcolare ed aggiungere ad una lista il tempo che intercorre tra time(event e) ed il più recente tra il momento in cui l'attività precedente nel control-flow all'interno dello stesso trace si è conclusa ed il momento in cui la risorsa che sta svolgendo l'evento *(resource(event e) = risorsa(precedente(event e)))* si è liberata l'ultima volta. Ottenute le liste sia per il log originario che per il log simulato, si andrà dunque a calcolare la distanza tramite la *Wasserstein distance.*

Infine, di tutti questi valori - corrispondenti alla quantità di attività nel *process model* - se ne ricaverà la media aritmetica, la quale verrà in seguito utilizzata per valutare una differente configurazione di *alpha.*

Il procedimento descritto fino a questo momento viene ripetuto per ogni attributo di *alpha* e, solo dopo aver definito per ogniuno di essi il valore migliore, la funzione *find best alpha* (Codice 4.4) restituisce la lista completa di valori di *alpha* ottimali.

```
Function find_best_alpha(L, distance, alpha, task_resource,
                         range_sinistra, range_destra, index)
  alpha_centro = alpha[index]
  alpha_sinistra = alpha
  alpha_sinistra[index] = random_value(range_sinistra,
                                       alpha_centro)
  L_sinistra = add_start_event(L, alpha_sinistra, start_range)
  new_task_resource = modify_task_resource(L_sinistra,
                                           S(task_resource),
                                           activities[index])
  S(task_resource) = new_task_resource
  Lsimu = run_simulation(M)
  distance_sinistra = compute distance(L, Lsimu)
  alpha_destra = random_value(alpha_centro, range_destra)
  L_destra = add_start_event(L, alpha_destra, start_range)
  new_task_resource = modify_task_resource(L_destra,
                                           S(task_resource),
                                           activities[index])
  S(task_resource) = new_task_resource
  Lsimu = run_simulation(M)
  distance_destra = compute distance(L, Lsimu)
  if min(distance_sinistra, distance_destra, distance) == distance
    index += 1
    alpha = find_best_alpha(L, distance, alpha, task_resource,
                            0, 1, index)
  endif
  if min(distance_sinistra, distance_destra, distance) ==
                                            distance_sinistra
    alpha = find_best_alpha(L_sinistra, distance_sinistra,
                            alpha_sinistra, S(task_resource),
                            alpha_sinistra, alpha_centro, index)
  endif
```

```
  if min(distance_sinistra, distance_destra, distance) ==
                                                distance_destra
    alpha = find_best_alpha(L_destra, distance_destra,
                            alpha_destra, S(task_resource),
                            alpha_centro, alpha_destra, index)
  endif
  return alpha
end
```

Codice 4.4: Funzione ricorsiva che data una configurazione iniziale di alpha cerca di trovare una nuova configuzione di alpha che possa generare un log con start timestamp il piu possibile vicini a quelli reali

4.3 Modulazione valori di alpha via *Bayesian Optimization*

L'ottimizzazione Bayesiana (*Bayesian Optimization*) è una delle diverse tecniche di *Hyperparameter Optimization* tesa all'individuazione degli iperparametri degli algoritmi di Machine Learning.[1] L'ottimizzazione Bayesiana (così definita in onore del matematico a Thomas Bayes, il quale ne ha formulato il teorema) consiste in una tecnica di tuning degli iperparametri il cui scopo è quello di ottimizzare i parametri di modelli estremamente complessi (più precisamente, nei casi di reti neurali note come profonde). Ciò viene fatto individuando, nel modello, i iperparametri ottimali; in altre parole, quelli in grado di assicurare un migliore funzionamento del modello stesso.

Essa, unitamente alla tecnica del *Evolutionary Optimization*, rappresenta una tecnica molto sofisticata e, al contrario della tecnica nota come *Grid Search* e

[1] Essa dunque, insieme alle tecniche Grid Search, Random Search ed Evolutionary Optimization, condividono lo scopo di individuare i migliori iperparametri per ogni dato modello o processo.
Nel caso della Ottimizzazione Bayesiana, dunque, per fare ciò viene appunto sfruttato il teorema di Bayes, utilizzando un approccio della c.d. "funzione surrogata" per trovare il giusto set di iperparametri che possano definire nel modo più ottimale la funzione del processo.

Random Search (ritenute più elementari e meno complesse) permette di essere utilizzata con maggior successo nell'ambito dello studio ed ottimizzazione delle reti neurali.

```
Function funzione da massimizzare(parametri alpha)
    L1 = add_start_event(L, alpha, start_range)
    for act in activities(N) do
        new_task_resource = modify_task_resource(L1,
                                                 S(task_resource),
                                                 act)
        S(task_resource) = new_task_resource
    endfor
    Lsimu = run_simulation(M)
    distance = compute distance(L, Lsimu)
    return -(distance)
```

Codice 4.5: Funzione incognita

Per trovare i valori di alpha migliori al fine di ottere un log i cui *start timestamp* sono il più possibile simili a quelli reali si è usata la funzione *Bayesian Optimization* della libreria *python bayes opt* a cui è stata passata la funzione da ottimizzare, *funzione da massimizzare* (Codice 4.5), che in questo caso ricalca quella del *random search*, ove - presa una completa nuova configurazione di *alpha* - vengono inseriti, in un nuovo log, gli *start timestamp* calcolati.
Utilizzando questo log stimato, poi, si calcoleranno le distribuzioni di probabilità per ogni distinta attività.

Una volta eseguito ciò, sarà possibile utilizzare il comando "*run simulation*" di Prosimos, generando così un log simulato.
Di seguito, a tale risultato si applicherà la medesima funzione per il calcolo delle distanze tra i due log utilizzata con la tecnica del *random search* previamente menzionata al paragrafo 4.2.

I valori di *alpha*, a questo punto, non sono più una lista di numeri generati randomicamente, bensì una lista della stessa lunghezza (cioè tanto lunga quante sono le attività nel process model), composta da coppie (0,1) poichè, essendo questa una tecnica di ottimizzazione vincolata ed essendo i valori di *alpha* i parametri, è necessario specificare i valori minimi e massimi per ciascun parametro.

A questo punto viene utilizzato il metodo *maximize*; con ciò, specificando il numero di step che l'ottimizzazione bayesiana deve compiere e il numero di *step* di *random exploration* per aiutare a diversificare lo spazio di esplorazione.

Il metodo *maximize* trova la configurazione di parametri che massimizza la funzione incognita; è per questo che il risultato di suddetta funzione è la distanza tra i due log cambiata si segno, in quanto ciò che si vuole ottenere è la riduzione della distanza tra i due log.

Capitolo 5

Esperimenti e Risultati

5.1 Risultato degli esperimenti

In questo capitolo si procederà alla verifica dell'efficenza degli algoritmi analizzati precedentemente, ovvero *random serch* e *bayesian optimization*, per la ricerca dei valori di *alpha*.
Più precisamente, l'analisi verrà condotta mediante esperimenti, i quali verranno condotti mediante la somministrazione di vari input provenienti da vasti process models agli algoritmi.
A seguire, i risultati di tali operazioni verranno utilizzati per calcolare gli *start timestamp* da inserire nel *log* originale, e confrontati, sia tra loro che con il *log* originario. Infine, verrà offerta una discussione sui risultati di tali esperimenti.

Inizialmente, viene caricato il modello di simulazione di cui al precedente paragrafo 4.1, consistente in un file BPMN.
A seguire, viene generata una lista *alpha*, inizializzata con valori randomici la cui lunghezza corrisponderà al numero delle varie attività del log.
Allo stesso tempo, viene caricato il log privo di *start timestamp* ed il *file json*, contenente diverse informazioni utilizzate quali parametri per la simulazione di Prosimos (es: *task resource distribution, bach processing, ecc...*).

Preliminarmente, prima di avviare la prima simulazione, è necessario modificare - all'interno del *file json* - l'informazione relativa ai *task rescource*.
Sarà proprio sulla scorta di tale informazione, infatti, che ne deriverà la durata

delle attività stesse.
Per fare ciò, verranno utilizzati i primi valori di *alpha* (generati randomicamente) ed effettuata una prima stima degli *start timestamp* all'interno del *log*.

Successivamente, per ogni attività svolta e per ogni risorsa verranno calcolati gli intervalli di durata tra gli *start* and *complete*.
L'informazione ottenuta servirà per individuare la *duration distribution* associate alle attività ed alla sua risorsa (*task rescource[attività][resource] = best duration distribution*).

Da tali risultanze, il *file json* verrà aggiornato; di conseguenza, si potrà condurre una prima simulazione utilizzando la funzione *Prosimos run simulation*, la quale restituirà un *log* simulato al cui interno vi sono gli *start timestamp* e i *complete timestamp*.

Da ciò, per ogni attività verranno calcolate le differenze (gli errori), ovvero la distanza fra il *log* simulato e quello originario. Ovviamente, posto che il *log* originario rimane privo dei c.d. *start timestamp*, l'opera di comparazione si concentrerà sull'analisi dei soli *complete time*.
La distanza tra i due *log* ottenuta per ogni attività, è bene ribadirlo, viene calcolata mediante la funzione *Wasserstein Distance*, la quale generarà una lista di distanze (ognuna corrispondente, appunto ad una diversa attività).
Da essa, se ne ricaverà una media aritmetica e, nel caso della funzione *random search*, questa consisterà nel valore di partenza per il successivo passaggio di modulazione dei valori di *alpha*.
Invece, con l'utilizzo della *bayesian optimization*, la media aritmetica verrà utilizzata ai fini del la valutazione sul miglioramento o peggioramento dei valori di *alpha*.

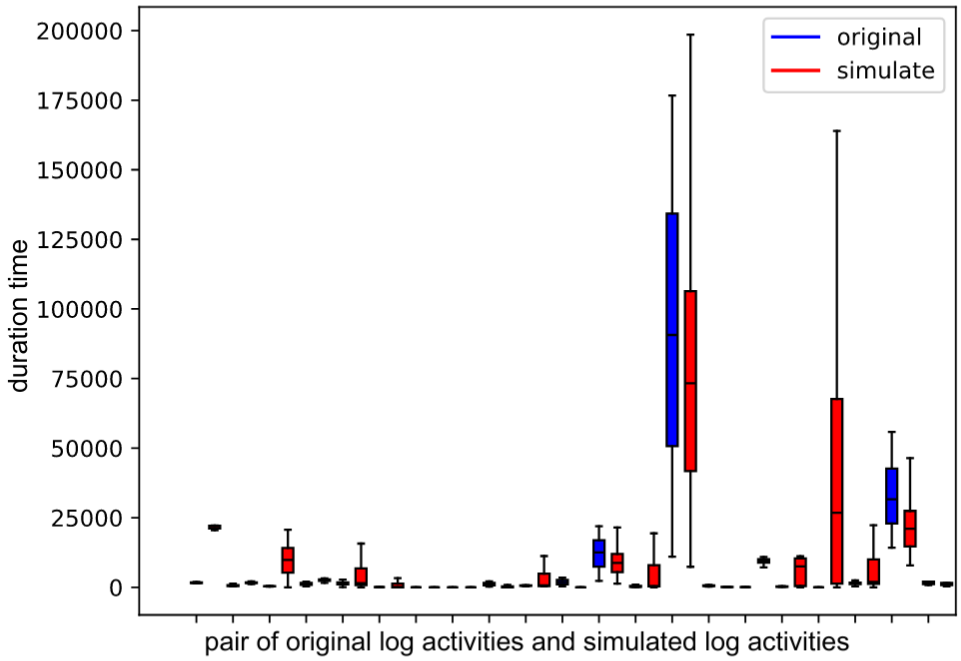


Figura 5.1: Random search result. Ogni coppia di box nel grafico rappresenta una distribuzione associata ad un'attività presente nel process model, ciò al fine di evidenziare la presenza di eventuali similitudini e differenze tra il log originario e il log simulato. Le durate riportate sono in secondi

Alla fine di tale processo di ricerca dei valori di *alpha*, una volta indivuata la miglior configurazione possibile della stessa, quest'ultima verrà utilizzata al fine di inserire dei valori di *start timestamp* all'interno del *log* originario.

Allo scopo di valutare l'accuratezza di questa tecnica, si è preso in esame un *log* i cui *start timestamp* erano tutti presenti, e poi successivamente rimossi. Grazie a tale operazione, è stato dunque facilmente possibile evincere il grado di efficenza e qualità delle due tecniche.

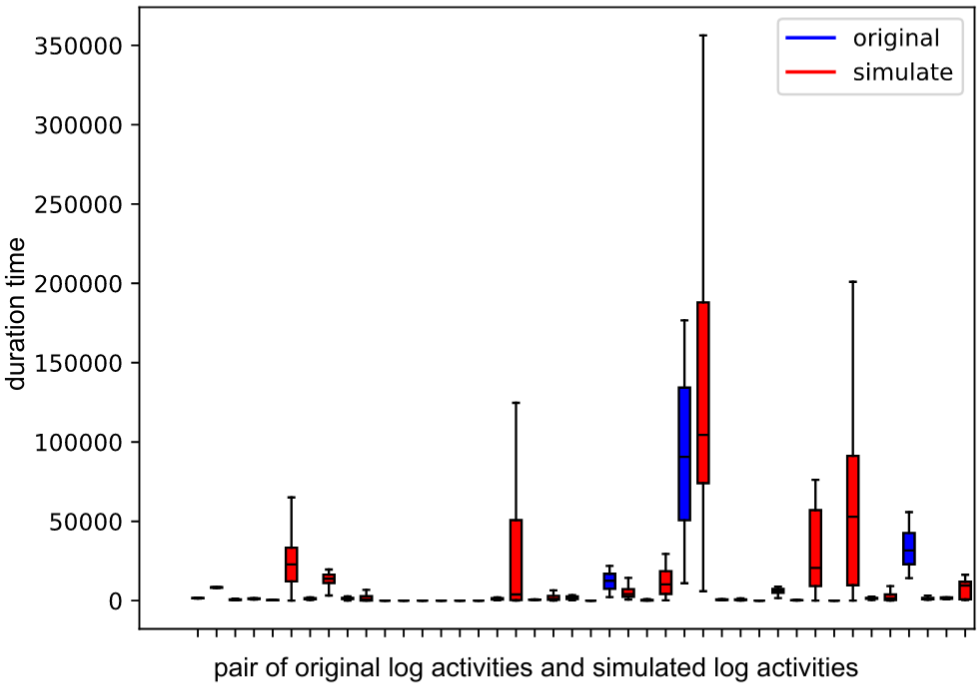


Figura 5.2: Bayesian Optimization result. Ogni coppia di box raffigurata nel grafico di cui sopra rappresenta ciascuna una distribuzione associata ad un'attività presente nel process model. Questo al fine di evidenziare la presenza o meno di similitudini e/o differenze tra log originale e quello simulato. Le durate riportate sono in secondi

5.2 Considerazioni

In questo paragrafo si analizzeranno i risultati ottenuti nei processi analizzati nei precedenti capitoli.

In via preliminare, dalle analisi svolte è emerso che il secondo metodo, ovvero quello relativo all'utilizzo della tecnica *Bayesian optimation*, si rivela complessivamente più adeguato rispetto al primo (*random search*); e ciò emerge inequivocabilmente sotto diversi profili, quali, ad esempio, la distanza media tra i due

log ottenuta, ed il *computational time*.

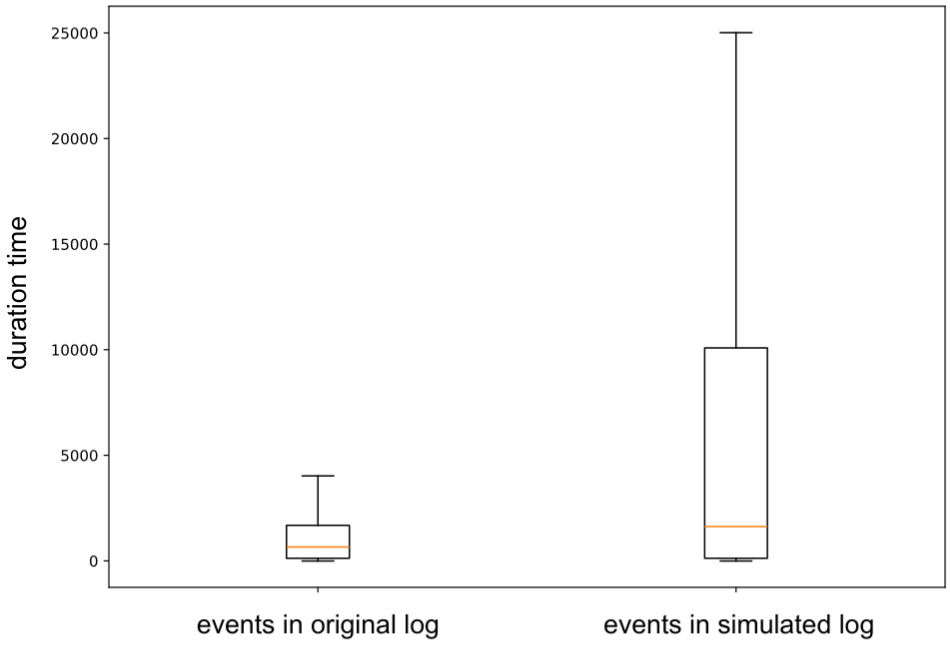


Figura 5.3: Random search result. Il box nel grafico sito a sinistra (events in original log) rappresenta la distribuzione delle durate delle attivita del log originario; a destra, invece, è raffigurata la distribuzione delle durate delle attività nel log simulato (events in simulated log). Le durate riportate sono in secondi

In particolare, dirimenti sono i risultati ottenuti nell'analisi di quest'ultimo, ovvero il tempo computazionale delle differenti soluzioni prospettate.
Infatti, dalle simulazioni svolte è evidente che il secondo metodo (quello portato avanti mediante la *Bayesian optimization*) si è dimostrato il più rapido.

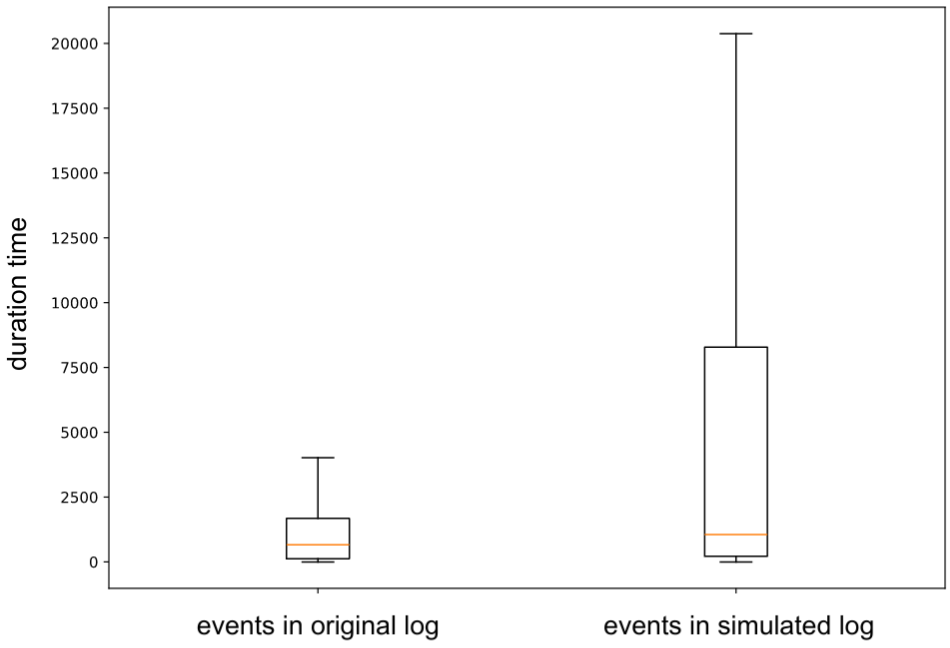


Figura 5.4: Bayesian Optimization result. Il box nel grafico sito a sinistra (events in original log) rappresenta la distribuzione delle durate delle attivita del log originario; a destra, invece, è raffigurata la distribuzione delle durate delle attività nel log simulato (events in simulated log). Da ciò si può evincere che il box del log simulato risulta migliore rispetto quello di cui alla Figura 4.3. Le durate riportate sono in secondi

Al contempo, tuttavia, è bene rilevare come, dal punto di vista della capacità di ricostruzione del *log*, tramite inserimento dei vari *start timestamp*, nessuno dei due modelli si dimostra ottimale.

Ciò in quanto il *log* generato tramite gli *alpha* finali non si rivela essere sufficientemente simile a quello originario.

Infatti, il *log* originario non risulta particolarmente popolato, in quanto contenente al suo interno meramente 608 *trace*; pertanto, non permettendo di aver abbastanza dati al fine di stimare la distribuzione delle durate corretta.

In più, si è partiti da un *log* del tutto privo di *start timesamp*; tale scelta, in particolare, è stata dettata dalla volontà di esaminare un c.d. caso limite (in altre parole, una delle casistiche più complesse prospettabili).
Tuttavia, è verosimile che, invece, un *log* reale possa avere degli *start timestamp*; e, per quanto pochi essi possano essere, questi si rivelerebbero determinanti per stimare gli *alpha* necessari a ricostruire il *log*.

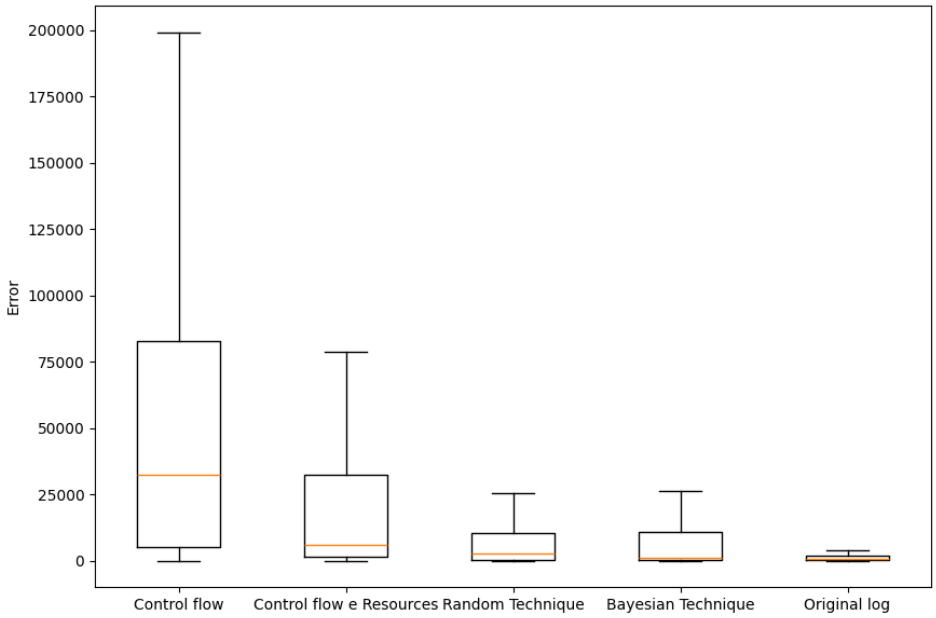


Figura 5.5: Confronto delle durate delle attività del caso di studio, usando le tecniche random search e bayesian optinization. Di seguito due tecniche naïve che assume waiting time = 0. Infine il log originario dove gli start timestamp sono noti

Di conseguenza, la ricerca svolta porta a concludere che le funzioni utilizzate nel presente elaborato si dimostrano adeguate nell'inserire le informazioni relative agli *start timestamp* di inizio delle attività, meglio rappresentando il *log* originario e, di conseguenza, le durate effettive delle attività all'interno del *log* (Tabella 5.1).
Tuttavia, tali operazioni non raggiungono ancora un grado di accuratezza ottimale, residuando delle discrasie tra il *log* originario e quello simulato.

	Media	Standard Deviation
Control flow	132207 secondi	404519 secondi
Control flow e Resources	32535 secondi	75582 secondi
Random Search Technique	11905 secondi	27903 secondi
Bayesian Opt Technique	9712 secondi	24055 secondi
Original log	6860 secondi	22406 secondi

Tabella 5.1: Tabella riportante le durate medie delle attività

Capitolo 6

Conclusioni

Questo progetto di tesi si è concentrato nel campo di ricerca del Process Mining, ed in particolare all'analisi ed estrazione di informazioni da event log dei processi di business.
Ciò al fine di eliminare le criticità esistenti in termini di stima delle tempistiche di tali processi, ponendosi come obbiettivo raggiungimento di un grado ottimale di efficienza.

Lo studio dunque è stato diretto all'implementazione di una funzione in grado di individuare i *timestamp* degli eventi in maniera più precisa, tale da permettere la stima del momento in cui una attività ha inizio, ottenendo così dei *start timestamp* il piu possibile aderenti al momento di inzio dell'attività analizzata.
Ciò allo scopo di ottenere un più puntuale spettro di tempistiche richieste per lo svolgimento di svariati processi produttivi.

Pertanto, la ricerca qui esaminata ha analizzato l'implementazione di due tipi di funzioni, *Bayesian Optimization* e *Random Search*, ponendo a confronto come esse interagivano nel ricercare i valori di alpha tali da condurre a stime di *start timestamp* più aderenti possibili alla realtà.
Di conseguenza, è emerso come - sebbene la funzione *random search* si sia dimostrata sufficientemente adeguata - i risultati ottenuti mediante l'utilizzo della funzione textitBayesian Optimization si sono rivelati qualitativamente superiori; ciò sotto sia il profilo della distanza media tra i log, sia i relativi computational

time.

Quanto a possibili futuri sviluppi, vi è indubbiamente l'implementazione di algoritmi più efficenti sotto il profilo delle prestazioni e tempistiche, così come di stima di *start timestamp* più vicini al dato reale.

Bibliografia

Testi e Articoli

[1] David Chapela-Campa et al. *Can I Trust My Simulation Model? Measuring the Quality of Business Process Simulation Models*. Berlin, Heidelberg: Springer-Verlag, 2023.

[2] Emanuele Fioriti. *La Metrica di Wasserstein*. univesità degli Studi di Bologna (cit. a p. 16).

[3] De Leoni M. Fracca C., Asnicar F. e Turco A. *Estimating Activity Start Timestamps in the presence of Waiting Times via Process Simulation* (cit. a p. 13).

[5] van der Aalst Wil. *Process Mining Data Science in Action*. Berlin, Heidelberg: Springer Berlin Heidelberg Imprint: Springer, 2016 (cit. a p. 1).

[6] van der Aalst et. Al. Wil M.P. *Process Mining. Come estrarre conoscenza dai log dei processi di business* (cit. a p. 5).

Siti

[4] *Prosimos*. URL: https://github.com/AutomatedProcessImprovement/Prosimos (cit. a p. 9).