Reti temporali e applicazioni

Algoritmi per il Graph Matching su Reti Temporali

GIORGIO LOCICERO

UNIVERSITÀ DEGLI STUDI DI CATANIA

CORSO DI LAUREA IN INFORMATICA TRIENNALE (L-31)

TESI DI LAUREA

Relatore: ALFREDO FERRO
Correlatore: MICALE GIOVANNI

ANNO ACCADEMICO 2019/2020

Indice

Elenco degli algoritmi

Sommario

Le tecnologie stanno crescendo in modo molto spedito negli ultimi decenni, i legami tra vari oggetti e soggetti sono sempre di più, nuovi legami si formano con il passare del tempo in certi istanti temporali precisi, e quindi nuove tecnologie servono per studiare questi legami, modellarli e riprodurre lo stesso comportamento reale che si vede nelle interazioni quotidiane, studiando le strutture più comuni per carpire informazioni significative ma celate.

In particolare, i grafi-reti temporali riescono a modellare qualsiasi sistema in cui ci siano delle interazioni che avvengono in certi tempi (che siano quanti temporali o intervalli) e riprodurre gli stessi comportamenti in modo formale, quindi sono di grande importanza per capire come funzionano le interazioni tra vari agenti (che siano esseri umani, macchine, animali, comportamenti emergenti), studiarle nel dettaglio estrapolando informazioni rilevanti alle analisi che si vogliono portare, riprodurre le caratteristiche di sistemi complessi come reti neurali (per la modellizzazione di reti neurali artificiali sulla forma di studi basati su grafi temporali) e reti cerebrali molto più interessanti anche per il futuro della digitalizzazione.

Nella costruzione di una infrastruttura di base utilizzabile ed adattabile ci saranno molte formalità, dato che mantenere la generalità è molto difficile per incorporare la componente temporale, quindi si utilizzeranno delle tecniche che si concentrano su una struttura locale delle interazioni.

Introduzione

I grafi statici sono di grandissima importanza in molti ambiti, e gli studi che li riguardano sono innumerevoli. Nonostante la loro indubbia utilità in molte applicazioni, i grafi semplici non riescono a modellare molte situazioni e sistemi che sarebbe utile emulare e modellare per studi ed analisi, si vedano ad esempio delle reti di social network, dove è possibile rappresentare i legami tra i singoli, ma è difficile rappresentare altre caratteristiche come l'invio di messaggi tra singoli o altre caratteristiche che possono essere molteplici per coppia di nodi/elementi del sistema.

Le reti temporali permettono l'aggiunta di dettagli e attributi aggiuntivi che aumentano la possibilità di rappresentare reti reali, in particolare rappresentazione di reti dinamiche (neurali, internet, infezioni, informazioni e passaggio di informazioni tra nodi).

Come per altri tipi di reti che approssimano situazioni reali e che aggiungono dettagli(come multigrafi e grafi multidimensionali), nelle reti temporali si cerca di rappresentare la componente temporale della realtà e l'interazione intervallare tra varie entità.

Proprio per rappresentare questo tipo di interazioni sono stati studiati ed utilizzati dei grafi complessi che, nel caso sopra citato, sono i multigrafi che permettono più archi tra due nodi oppure, nel caso di rappresentazione di più tipi di relazioni nella rete da rappresentare, grafi **multilayer** (nello specifico reti multidimensionali che sono un tipo specifico di multilayer) che permettono di rappresentare i diversi tipi di relazioni del sistema e nel sistema.

Una parte importante di molte situazioni reali che non è stata ancora definita in modo chiaro è la componente temporale di una rete, che si ritrova molto spesso in qualsiasi sistema complesso. Con componente temporale si intende la temporalità delle interazioni tra gli elementi di una rete, che può essere espressa come intervalli temporali oppure come singoli contatti (queste caratteristiche verranno definite nello specifico nella sezione 1: **Reti temporali** dove verranno definite più formalmente, insieme ai problemi logici e di implementazione che ne risulteranno), in particolare l'utilizzo dei contatti è molto utile ed importante per rappresentare situazioni dove avviene un processo di espansione temporale e a stati, infatti i grafi temporali a contatti sono molto spesso accompagnati da modelli di Markov e altri modelli di transizione di stati, che così riescono a rappresentare e modellare delle situazioni che non si potrebbero ottenere con i semplici grafi statici.

La simulazione e modellizzazione di reti reali che hanno una componente temporale(che può essere un **timestamp** o un**intervallo temporale**) è di grande importanza in molti ambiti che spaziano qualsiasi oggetto di studio:

- **Medicina**: nella modellizzazione di reti cerebrali e di interazioni tra parti del corpo o interazioni cellulari, come per esempio il metabolismo, oppure sintesi di proteine, collegamenti sinaptici ed interazioni microscopiche e mesoscopiche cerebrali, molto utili per lo studio neurologico ed intorni.

- **Trasporti**: nella modellizzazione del traffico e nella simulazione di strade, utilizzabile quindi per il cambiamento di percorsi dinamico in base al tempo, o per la costruzione di nuove strutture nel caso certe situazioni lo richiedano.

- **Blockchain**: dove i dati vengono concatenati temporalmente quindi la componente temporale è utilizzata totalmente per l'implementazione anche per le interazioni di condivisione tra gli utenti.

- **Social Network**: come già specificato, interazioni tra gli utenti sono identificate anche da dei timestamp che, come già accennato, possono essere singoli contatti o intervalli temporali, quindi aumentare l'accuratezza di algoritmi che utilizzano le interazioni tra gli utenti ed utilizzare i dati a vantaggio dell'insieme.

- **Sistemi distribuiti**: per studiare la distribuzione ed i collegamenti dei sistemi distribuiti in modo da trovare la configurazione ottimale per la massimizzazione delle prestazioni.

- **Modelli di espansione in generale**: per esempio modelli epidemici e tumorali, oppure lo studio della distribuzione di informazione o altri campi dove l'espansione di qualche tipo di informazione è portata da eventi ben definiti, come il contatto con altre persone, e le interazioni con community e grandi gruppi di persone, con la creazione di sottogruppi temporali.

Nella prima sezione si daranno le definizioni utili per definire i problemi che si affronteranno, in particolare si vedrà una semplice descrizione dei grafi semplici che poi verrà espansa ai grafi temporali.

Nella seconda parte viene visto il problema principale che verrà trattato, cioè il **matching di grafi** espanso con la **componente temporale**, e verranno presentate le componenti importanti per la risoluzione del problema, già accennate nel primo capitolo.

Verranno infine presentati gli algoritmi che sono stati utilizzati per il matching ed il calcolo degli isomorfismi, che verranno descritti e visti nel dettaglio in seguito.

Si vedranno inoltre le conseguenze dell'aggiunta della componente temporale, che può risultare semplice ad una prima occhiata, ma nasconde una **complessità** tipica delle rappresentazioni reali che cercano di incorporare la componente temporale e componenti non finite e non prevedibili nella costruzione.

Infatti costruzione e analisi di una rete temporale sono pratiche che coinvolgono diverse discipline, una analisi topologica semplice potrebbe non essere abbastanza dato che le informazioni temporali verrebbero perse completamente.

Da un altro punto di vista, l'analisi pura della componente temporale porta alla perdita di informazioni molto importanti di struttura statica che potrebbero essere fondamentali (questa contrapposizione di tecniche di analisi verrà vista in seguito quando si parlerà delle varie tecniche utilizzabili per l'analisi di un grafo temporale).

Di fondamentale importanza per le reti temporali è la parte della teoria che studia i processi e le dinamiche di espansione e propagazione(spreading) di fenomeni, modelli compartimentale come suddivisione in stati caratteristici, rappresentati dai nodi della rete temporale, cioè ogni nodo appartiene ad un compartimento.

Per esempio nel modello di espansione **SIR** (**susceptible infected recovered**) o **SIS** (dove si ritorna suscettibili dopo l'infezione)un nodo può appartenere a tre compartimenti differenti, suscettibile, infetto e recovered(guarito), i passaggi da un compartimento ad un altro avvengono tramite funzioni che tengono conto del tempo di contatto tra i vari compartimenti.

Altri modelli più complicati hanno più compartimenti e modellano ancora più dettagliatamente la realtà di situazioni di espansione epidemica.

La probabilità di infezione e di guarigione(**infection rate** e **recovery rate**) sono importanti parametri per il modello. Applicare questo modello significa supporre che tutte le interazioni avvengano uniformemente nel tempo, cosa che non è vera la maggior parte delle volte. Per esempio l'infezione di un virus avviene anche in base al tempo di contatto con l'infetto, stesso discorso può essere fatto sul passaggio da infetto a guarito.

Vaccinazioni diminuiscono la probabilità di essere infettati, possiamo introdurre modelli di strategie di vaccinazione e di prevenzione. Questo modello, come tutti i modelli che si rifanno a situazioni reali, assume durante le varie interazioni e passaggi di compartimento un comportamento **bursty**.

Questo genere di **modelli epidemici** e di **espansione** in generale è diventato molto di moda in questo periodo specialmente per l'emergenza SARS-CoV-2, quindi di grande rilevanza sia per la modellizzazione di soluzioni, sia per l'analisi stessa delle dinamiche di espansione.

Durante questo studio non verranno trattati nello specifico modelli epidemici, anche se verranno aperte delle piccole parentesi riguardanti qualsiasi modello di espansione.

1 Reti temporali

Di seguito verranno presentate le reti temporali iniziando prima con la definizione di grafi semplici per poi aggiungere la componente temporale.

Un concetto essenziale da capire inizialmente è che ogni link (arco) di una rete temporale **trasmette-trasporta informazioni** solo durante il tempo in cui è attivo, questo **tempo di attivazione** può essere descritto da un **quanto stabilito**(quindi la rete in un tempo stabilito può essere vista come uno stato) o da un **intervallo di tempo** in cui può avvenire il contatto adottando un approccio reale.

Si usano quanti di tempo ben definiti quando, più che un approccio temporale puro, si deve studiare-rappresentare un **approccio a stati** con passaggio da stato a stato in tempi ben definiti (unici). Si usano intervalli di tempo in quasi tutte le situazioni che coinvolgono interazioni reali .

Solitamente si studiano reti reali basate su intervalli, ma le reti dinamiche basate su contatti sono più semplici da utilizzare ed analizzare, anche grazie alla immensa letteratura che riguarda le reti statiche, che sono abbastanza adattabili ad alcuni approcci utilizzati nell'analisi di reti dinamiche. Durante questo studio, verranno utilizzate delle reti temporali a contatti singoli(solo un tempo per arco), perché l'implementazione è risultata abbastanza ostica per reti temporali a più contatti per arco e per reti temporali ad intervalli, ma i ragionamenti che vengono fatti in tutti i capitoli sono assolutamente generali per tutti i tipi di reti temporali, solo l'implementazione e gli algoritmi utilizzati cambiano.

Non verranno trattati nel dettaglio algoritmi specifici per funzionalità poco comuni ma verranno visti algoritmi di utilità generale che serviranno poi anche per gli isomorfismi tra grafi temporali.

Altre definizioni formali specifiche per il graph isomorphism verranno viste in seguito e solo accennate, quindi verranno visti i componenti delle reti temporali utili al fine ultimo del calcolo degli isomorfismi.

Molte definizioni utili per altri problemi sono state tralasciate anche perché di poca rilevanza per il problema del graph matching che utilizza misure di ottimalità locale per la ricerca della soluzione globale, e molte delle misure descritte in [2] utilizzano misure globali o che utilizzano una buona parte del grafico e che, alla fine, non hanno trovato utilità per il problema di graph matching, quindi fare riferimento a quel testo per altre possibilità.

Molte delle definizioni inoltre sono state create per aiutare la definizione di isomorfismo per reti temporali, per esempio la definizione di *propagazione*, con classificazione correlata, è stata creata per descrivere la struttura temporale di un nodo del grafo, e questa struttura temporale è stata utilizzata per descrivere il problema globale di graph matching, che altri autori hanno affidato alla definizione di time-respecting-path.

Si considerano singoli contatti o singoli intervalli per arco, ma durante le definizioni ci saranno delle piccole parentesi per espandere i concetti a più contatti o più intervalli per singoli archi.

Sulla definizione di temporal graph matching si vedranno le varie definizioni date dai vari autori[11, 12, 4] con le conseguenti problematiche che si presentano e la poca generalità di alcune.

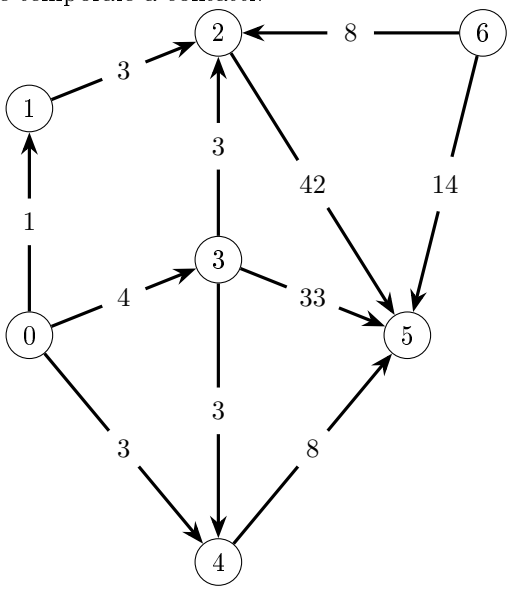
Verranno inoltre visti i problemi che affiorano dall'utilizzo delle reti temporali a contatti e ad intervalli, con delle conseguenti scelte e tradeoff.

Una parentesi importante sarà data da caratteristiche interessanti delle reti temporali, per esempio la distribuzione dei contatti o il comportamento di certe reti reali, che influenzeranno la costruzione dei modelli e del problema di temporal graph matching andando a delineare i contorni delle varie applicazioni delle reti temporali con delle soluzioni ai problemi che riescano a racchiudere la maggior parte dei problemi più rilevanti.

Per vedere altre caratteristiche importanti dei grafi temporali (come relazioni ed eventi uno a uno, molti a molti, uno a molti, oppure se i contatti avvengono in modo sincrono o asincrono, o quali strategie utilizzare nel caso in cui non si voglia lavorare direttamente sulla rete temporale ma su surrogati che perdono alcune caratteristiche o temporali o topologiche) leggere attentamente [2] dato che è il testo più aggiornato e che racchiude molta intuizione utile per la definizione dei singoli problemi.

Vengono considerati grafi temporali direzionati dove gli archi hanno una sola direzione, ma i risultati possono essere espansi a grafi indirezionati molto semplicemente.

Durante tutto il percorso di definizioni e problemi verrà utilizzata la seguente rete temporale a contatti:



1.1 Definizioni formali

Prima di tutto si dà una definizione di grafo statico ed altre definizioni basilari per introdurre i concetti di base che verranno in seguito espansi:

Definizione 1. (Grafo statico). Si dice Grafo Statico G una coppia ordinata (V, E) di insiemi, dove l'insieme Vè l'insieme dei nodi ed $E \subseteq V^2$ è l'insieme degli archi tra due nodi .

Questa definizione è la più basilare che si può avere, non contiene elementi complessi come attributi multipli per arco, pesi degli archi, etichette associate ai nodi o possibilità di archi di diverso tipo tra gli stessi due nodi.

Per espandere questa definizione con componenti aggiuntive come la componente temporale oppure il tipo di ogni arco e la possibilità di vari attributi si possono aggiungere queste caratteristiche direttamente negli archi tra i vari nodi. assegnando ad ogni arco componenti aggiuntive come liste o insiemi, oppure mantenendo delle strutture esterne che associano ad elementi del grafo statico caratteristiche aggiuntive. Nel caso di multigrafi l'insieme degli archi diventa un multiinsieme.

A fine sezione si vedrà come l'aggiunta della componente temporale aggiunga una complessità notevole a tutti i problemi collegati ai grafi.

Definizione 2. (Grafo diretto e indiretto). Se E è una relazione simmetrica allora si dice che il grafo è non orientato (o indiretto), altrimenti si dice che è orientato (o diretto).

Nel caso di grafo statico, il concetto di direzione è opzionale, ma molto spesso nei grafi temporali la direzione è mandatoria, anche perché per i cammini Time-Respecting non valgono le proprietà transitiva e simmetrica, quindi i cammini possono essere visti come unidirezionali.

Di seguito verranno date delle definizioni di base per l'introduzione dei concetti che servono per la descrizione dei sistemi temporali, quindi si parte con la definizione di contatto temporale per poi finire con la definizione di struttura temporale, che verrà usata totalmente durante la definizione del problema di temporal graph matching.

Definizione 3. (Contatto temporale). Un contatto è una tripla $(source, destination, time) \in V^2 \times \mathbb{R}$ che definisce una interazione tra un nodo sorgente ed un nodo destinazione in un un certo attimo temporale.

Questa definizione di contatto temporale esprime un singolo momento in cui l'evento può accadere, nel caso di intervalli si parla di *intervallo di contatto* e la nomenclatura utilizzata resta la stessa (tranne che per *time* che diventa *interval*), anche se solitamente il momento in cui avviene la propagazione è comunque un singolo momento, ma dipende comunque dalle esigenze e dalle descrizioni singole dei problemi che si stanno tentando di simulare o risolvere.

Molto spesso vengono associati dei contatti o intervalli di attivazione anche ai nodi stessi del grafo temporale, anche per simulare in modo più concreto sistemi reali dove vi è questo comportamento, durante questo studio questo concetto non verrà considerato, ma il framework di definizioni resta comunque espandibile, specialmente l'impronta temporale che verrà esposta prossimamente, formata da un insieme di classi espandibile secondo i propri bisogni.

Nonostante si possa definire un grafo temporale senza l'aiuto della definizione di contatti, la medesima è molto utile durante molte definizioni formali e nella descrizione dei vari algoritmi (che utilizzano infatti il concetto di contatto attivamente).

Definizione 4. (Grafo temporale per contatti). Si dice Grafo Temporale per contatti G_T una coppia ordinata (V, E) di insiemi, dove l'insieme Vè l'insieme dei nodi (che è possibile identificare tramite funzioni aggiuntive che associano ad ogni nodo un id univoco) ed $E \subseteq V^2 \times \mathbb{R}$ è l'insieme degli archi tra due nodi dove la componente temporale è un singolo numero che esprime il momento del contatto .

Ad un grafo temporale possono essere associate altre componenti, dipendenti dalle strutture sia topologica che temporale, per esempio si può associare la struttura statica come infrastruttura ed utilizzare delle misure e metriche derivate dalle componenti temporali per descrivere il grafo temporale tramite caratteristiche globali e locali.

Nel caso di più contatti per nodo, questi possono essere visti come singole associazioni per il singolo arco oppure come archi a se stanti.

Tra le caratteristiche locali che serviranno in seguito per la definizione di Matching su reti temporali verrà visto il concetto di *propagazione* e *struttura temporale*.

Definizione 5. (Grafo temporale per intervalli di contatto). Si dice Grafo Temporale per contatti G_T una coppia ordinata (V, E) di insiemi, dove l'insieme Vè l'insieme dei nodi ed $E \subseteq V^2 \times \mathbb{I} | \forall e \in E; a, b \in \mathbb{R}, a \leq b, e.interval = [a, b] \subseteq \mathbb{R}$ è l'insieme degli archi tra due nodi dove la componente temporale è un intervallo identificato da due numeri reali che sono gli estremi che esprime il tempo in cui l'arco era attivo per mandare informazione .

Durante l'attivazione di un arco nell'intervallo di tempo specificato l'informazione si propaga di nodo in nodo, quindi può passare da un nodo all'altro solo durante gli intervalli di tempo stabiliti, non prima e non dopo.

Stesso discorso per più intervalli per nodo, che possono essere visti e definiti come associati al singolo arco oppure indipendenti l'uno dal l'altro.

È importante specificare questo concetto anche perché nel caso di contatti si aveva il δ aggiuntivo rispetto ai singoli contatti che permetteva la trasmissione tra un nodo ed un altro solo se venivano rispettati i vincoli temporali.

Alla coppia ordinata sopra definita è possibile associare altre componenti come per esempio l'insieme di nomi per nodi o per archi, delle funzioni che identifichino matematicamente i singoli nodi o archi e che li associno a identificativi univoci, oltre all'aggiunta di attributi aggiuntivi per ogni componente già definita, cioè per ogni nodo, per ogni arco o per ogni contatto, ma durante questo studio si manterrà una definizione semplice per semplicità e per definire una infrastruttura basilare molto facilmente espandibile

Per il caso di Grafo Temporale ad intervalli vale più o meno lo stesso ragionamento, anche se la complessità aumenta considerevolmente in tecnicismi abbastanza poco rilevanti nel caso di contatti, ma che assumono grande importanza nel caso di definizione di intervalli di propagazione che verranno visti in seguito.

Definizione 6. (Cammino Time-respecting). Si dice cammino time-respecting una sequenza di contatti con tempi non-decrescenti, più formalmente :

Data una sequenza di archi $\{e_1, \ldots, e_n\}$, questa sequenza forma un time-respecting path se e solo se $r \in [1, n[\subseteq \mathbb{N}, n = |V_G|, e_r = (s_r, d_r, t_r) \in E_G, d_r = s_{r+1}, t_r < t_{r+1}$, cioè un cammino time-respecting deve avere delle interazioni tra i vari nodi che rispettino il normale scorrere del tempo.

Nel caso di contatti multipli per singolo arco, questa definizione non cambia.

Un discorso diverso deve essere fatto nel caso di intervalli di contatto, che complicano leggermente la definizione dato che gli intervalli si portano dietro il minimo delle intersezioni tra gli intervalli dell'arco entrante e dell'arco uscente.

La definizione è comunque dipendente dalle condizioni singolari di ogni caso, quindi non si daranno definizioni aggiuntive, ma verranno accennate le intuizioni utili per la definizione di problemi in caso di Grafo Temporale ad intervalli.

Definizione 7. (Connessione tra due nodi i, j). Si dice che due nodi siano connessi se esiste un cammino Time-Respecting tra i due nodi.

Questa definizione definisce la connessione tra i due nodi come l'esistenza di un cammino che colleghi questi due che rispetta l'ordinamento temporale.

Si può notare come se due nodi (i, j) sono connessi tra di loro e (j, t) anche, non è detto che esista un cammino Time-Respecting tra i e t, quindi non vale la proprietà transitiva.

Un altro fattore importante da notare è il fatto che anche le connessioni sono temporali, in quanto se due nodi sono connessi ad un tempo definito, non è detto che lo siano all'istante successivo, quindi bisogna riportare anche degli attributi temporali per la connessione per non perdere tempo.

Un altro grafo importante che è possibile ricavare dalle connessioni è il reachability graph. In un reachability graph un nodo ha un arco orientato verso un altro nodo se e solo se è il primo è connesso con il secondo. Ovviamente l'arco è orientato perché un cammino da un nodo a ad un nodo b che è Time-Respecting non può essere utilizzato all'inverso dato che i tempi di contatto saranno tutti decrescenti.

Durante i prossimi capitoli e la sezione in cui vengono visitate le varie possibilità di graph matching vengono viste altre possibilità di rappresentazione di Grafi Temporali che semplificano o arricchiscono il modello da analizzare.

Definizione 8. (Sotto-grafo temporale di un grafo G_T). Si dice sotto-grafo temporale di un grafo temporale $G_t = (V, E)$ la coppia $G_s = (V_s, E_s)$ formata dal sottoinsieme dei nodi $V_s \subseteq V$ ed il sottoinsieme dei contatti
$E_s \subseteq E | \forall e_s \in E_s, e_s.source \in V_S \wedge e_s.destination \in V_S$.

Questa definizione è indipendente dal tipo di temporalità delle interazioni, quindi è una definizione valida sia per contatti che per intervalli.

Definizione 9. (Propagazione tra due nodi (i, j) attraverso un nodo t). Si dice propagazione (o flusso) $p_{itj} \in E \times E$ la coppia ordinata di archi
$p_{itj} = (e_r, e_s), e_r, e_s \in E | e_r.destination = e_s.source$.

Ad una propagazione vengono associati altre caratteristiche come il nodo centrale di propagazione (nel caso di p_{itj} ,$p_{itj}.node = t$), latenza di propagazione (che si vedrà nella prossima definizione) e, nel caso di intervalli temporali, l'intervallo di propagazione che viene ottenuto nel seguente modo:

$$p_{itj}.intersect = p_{itj}.e_s.interval \cap p_{itj}.e_r.interval \tag{1}$$

$$p_{itj}.propInterval = \begin{cases} [min(p_{itj}.intersect), max(p_{itj}.e_s.interval)] & \text{if } p_{itj}.intersect \neq \emptyset \\ \emptyset \quad \text{if } p_{itj}.intersect = \emptyset \end{cases} \tag{2}$$

Si può notare come nel caso di contatti il primo tempo è sempre quello dell'arco uscente, e l'ultimo tempo dell'intervallo viene dato dal δ definito, anche se il fine ultimo è abbastanza differente ed alcune risultati non coinciderebbero effettivamente.

Nel caso di contatti multipli, si deve fare una distinzione aggiuntiva delle propagazioni con gli stessi archi entranti ed uscenti, se si considerano i contatti (o gli intervalli) singoli come archi a sé stanti, cioè triple, la distinzione è automatica, ma la notazione deve essere cambiata leggermente (mettendo in apice i tempi di contatto entrante ed uscente), oppure definendo degli insiemi

di propagazioni per un arco entrante ed uno uscente per ogni combinazione di contatti-intervalli.

Per esempio, la notazione dovrebbe essere cambiata in $p_{itj}^{t_1 t_2}$ per esprimere una propagazione formata dai due archi che hanno tempi multipli, e quindi devono essere individuati univocamente, mettendo in apice i tempi. Oppure la notazione può rimanere la stessa, ma una propagazione non sarebbe più la coppia ordinata $(e_r, e_s), e_r, e_s \in E | e_r.destination = e_s.source$ ma l'insieme delle coppie ordinate $\{(e_r, e_s), e_r, e_s \in E | e_r.destination = e_s.source\}$ dove possono esistere propagazioni con stessi archi entranti ed uscenti, ma con tempi-intervalli differenti.

Il caso di contatti-intervalli multipli verrà rivisitato più volte in seguito anche perché rappresenta la generalizzazione del problema per eccellenza, oltre a portare con se delle tecnicità notevoli sia in fattore logico-deduttivo, sia nelle performance degli algoritmi che si vedranno, che dovranno confrontare una quantità fattorialmente maggiore (per le combinazioni ottenibili tra i vari contatti di ogni arco).

Definizione 10. (Latenza di propagazione per contatti). Ad ogni propagazione viene associato un numero che rappresenta l'intervallo temporale tra il tempo di contatto dell'arco entrante e quello dell'arco uscente, più formalmente:

Data p_{itj}, si dice latenza $\Delta(p_{itj}) = p_{itj}.e_s.time - p_{itj}.e_r.time$, per limitare la ridondanza la latenza verrà direttamente associata ad ogni propagazione come attributo.

Il concetto di latenza si può ampliare anche per altre vie per esempio definendo una *information latency* come l'intervallo di tempo tra la trasmissione di una informazione da un nodo ed il suo arrivo ad un altro nodo, ma questa definizione non ci serve per il problema che verrà trattato in seguito sul matching dato che questo tipo di informazione è globale ed ha pochi risvolti nella risoluzione del matching. Per approfondimenti riguardanti altre misure utilizzabili per la creazione dei modelli guardare [2].

Definizione 11. (Insieme delle propagazioni $P_t$di un nodo t). Si definisce l'insieme di tutte le propagazioni di un nodo l'insieme formato da tutte le propagazioni che passano dal nodo t, più formalmente $P_t = \{p_{iqj} | p_{iqj}.node = t\}$

Questa definizione stabilisce una buona base da dove derivare altre definizioni significative per molti problemi, ma include effettivamente soltanto grafi per contatti ed intervalli singoli per ogni arco.

Nel caso di contatti o intervalli molteplici per arco, bisogna effettivamente scegliere due strade intuibili già accennate precedentemente, cioè considerare i contatti come archi indipendenti oppure appartenenti ad un insieme associato all'arco.

Nel primo caso, la definizione non cambia, e quindi l'insieme sarebbe sostanzialmente lo stesso con le dovute distinzioni per le propagazioni, che dovranno tenere conto dei tempi diversi, quindi utilizzando la notazione già definita precedentemente $p_{itj}^{t_1 t_2}$ che tiene conto pure dei tempi di contatto oltre agli archi entranti ed uscenti.

Nel secondo caso si può fare riferimento alla definizione di propagazione come insieme di coppie ordinate di archi. Quindi costruire la definizione di insieme delle propagazioni come insieme di insiemi, ognuno formato dalle combinazioni dei singoli insiemi contenuti nella propagazione corrispondente, la seguente definizione formale è abbastanza ostica e non saprei descrivere la sua accuratezza.

Sia $prop_{a_itb_i}[i]$ l'i-esima propagazione (come insieme di coppie ordinate) associata al nodo t e aventi sorgente e destinazione a_i e b_i rispettivamente, con $i \in \mathbb{N}|0 < i < inDeg(t) * outDeg(t)$,

allora $P_t = \{\{(a_1, a_2), (a_3, a_4), ...(a_{Deg(t)*2-1}, a_{Deg(t)*2})|$
$(a_1, a_2) \in prop_{atb}[1] \wedge$
$(a_3, a_4) \in prop_{atb}[2] \wedge$
$... \wedge (a_{inDeg(t)*outDeg(t)-1}, a_{inDeg(t)*outDeg(t)*2}) \in prop_{atb}[Deg(t)]\}|$
$p_{atb}.node = t\}$

Esempio 12. (Insieme delle propagazioni in un grafo temporale con contatti multipli per arco). Dato un grafo temporale G_t con i seguenti due insiemi di propagazioni associate ad un nodo :

$p_{123} = \{((1,2,1),(2,3,4)),((1,2,3),(2,3,4))\}, p_{428} = \{((4,2,6),(2,8,10)),((4,2,11),(2,8,10))\}$

L'insieme di propagazioni associate al nodo 2 è il seguente:

$P_2 = \{\{((1,2,1),(2,3,4)),((4,2,6),(2,8,10))\},$
$\{((1,2,1),(2,3,4)),((4,2,11),(2,8,10))\},$
$\{((1,2,3),(2,3,4)),((4,2,6),(2,8,10))\},$
$\{((1,2,3),(2,3,4)),((4,2,11),(2,8,10))\}\}$

Notare come la cardinalità dell'insieme risultante sia $|p_{123}|*|p_{428}| = 4$, quindi 4 strutture temporali diverse associate ad un singolo nodo, che sono le combinazioni dei contatti dell'arco $1 \to 2$ con i possibili tempi di contatto dell'arco $4 \to 2$, quindi l'insieme risultante avrà cardinalità uguale a quella del prodotto cartesiano degli insiemi delle propagazioni associati al nodo.

Si definisce l'insieme di tutte le propagazioni appartenenti ad un grafo temporale come $Propagations = \bigcup_{i}^{n} P_i$

Definizione 13. (classificazione di una propagazione). Una propagazione p_{itj}può essere di due tipi diversi:

Nel caso di contatti:

1. (propagazione Time-respecting). Una propagazione p_{itj} viene detta Time-Respecting se $p_{itj}.e_s.time > p_{itj}.e_r.time$, a sua volta si divide in altri due sotto tipi:
   (a) (propagazione δ-time-respecting). Una propagazione p_{itj} viene detta δ-Time-Respecting se $p_{itj}.e_s.time - p_{itj}.e_r.time < \delta$
   (b) (propagazione non δ-time-respecting). Una propagazione p_{itj} viene detta non δ-Time-Respecting se $p_{itj}.e_s.time - p_{itj}.e_r.time \geq \delta$
2. (propagazione non Time-respecting). Una propagazione p_{itj}viene detta non Time-Respecting se $p_{itj}.e_s.time \leq p_{itj}.e_r.time$

Nel caso di intervalli:

1. (propagazione Time-respecting). Una propagazione p_{itj} viene detta Time-Respecting se $p_{itj}.e_s.interval \cap p_{itj}.e_r.interval \neq \emptyset$

2. (propagazione non Time-respecting). Una propagazione p_{itj} viene detta non Time-Respecting se $p_{itj}.e_s.interval \cap p_{itj}.e_r.interval = \emptyset$

Come si può facilmente intuire, il caso di propagazione su intervalli generalizza quello su contatti, anche perché la definizione del δ può essere vista come un intervallo che inizia al tempo stabilito e finisce dopo un certo tempo ($[e_r.time, e_r.time + \delta[$), ma in realtà la funzione del δ è molto spesso differente, quindi si manterrà la distinzione.

Si possono aggiungere altre distinzione oppure togliere delle categorie, la definizione di classificazione è molto facilmente espandibile rispetto al problema di base che si sta cercando di risolvere. Per esempio se si volesse considerare il caso di grafo temporale Time-Respecting (cioè quello utilizzato da molti autori [8, 11, 12]) basta considerare soltanto le propagazioni δ-time-respecting e fare i confronti di conseguenza.

L'insieme delle classi possibili (che possono essere stringhe o enumerazioni) di una propagazione è il seguente :

$PropType = \{TimeRespecting, NotTimeRespecting\}$

Eventualmente è possibile associare altre classi come per esempio la classe $deltaTimeRespecting$ ma comunque dipende dalle esigenze del singolo problema.

La funzione che associa ad ogni propagazione la sua classe specifica è:

$classProp : Propagations \rightarrow PropType$

Secondo la seguente legge:

(caso contatti)

$$classProp(prop) = \begin{cases} TimeRespecting & \text{if } 0 < p_{itj}.e_s.time - p_{itj}.e_r.time < \delta \\ NotTimeRespecting & \text{if } 0 < p_{itj}.e_s.time - p_{itj}.e_r.time \geq \delta \end{cases} \quad (3)$$

(caso intervalli)

$$classProp(prop) = \begin{cases} TimeRespecting & \text{if } p_{itj}.intersect \neq \emptyset \\ NotTimeRespecting & \text{if } p_{itj}.intersect = \emptyset \end{cases} \quad (4)$$

Come già detto si potrebbe generalizzare il tutto considerando il delta per la definizione di intervallo, ma la definizione è stata definita separatamente per far capire che il δ è un parametro globale, e anche perché l'intersezione di certi intervalli (contatto entrante minore di un contatto uscente, quindi l'intersezione tra gli intervalli risultanti sarebbe non nulla, quindi ambigua e contraddittoria rispetto alla definizione di contatti e di time-respecting path per contatti) risulterebbe in propagazioni Time-Respecting quando non dovrebbero esserlo.

Nel caso di grafi con più contatti o intervalli per arco, si può utilizzare una delle due definizioni date precedentemente, cambiando l'equazione sopra descritta considerando ogni singolo elemento della propagazione, e restituendo

l'insieme delle classi associate, oppure utilizzando sempre la stessa funzione, ma andando a lavorare sui singoli elementi della propagazione nel caso in cui una propagazione sia l'insieme di coppie di contatti associati al nodo, aventi stessi sorgenti e destinazioni, ma tempi differenti.

Definizione 14. (impronta o carattere temporale). Il multiinsieme ordinato formato da tutti i tipi di propagazioni di un nodo s è chiamato impronta(o carattere) temporale e viene definito nel seguente modo:

$impronta_s = (PropType, classProp)$

$impronta_s = \{\forall prop_s \in P_s | prop_s^{classProp(prop)}\}$

Questo multiinsieme è formato dalle classi delle propagazioni del nodo s insieme alle molteplicità dei singoli elementi.

Nel caso di intervalli la definizione non cambia dato che le classi sono ben definite e si può costruire una impronta univoca del nodo.

Per più contatti o intervalli per arco la situazione cambia maggiormente, ma utilizzando le definizioni di propagazione come insieme di contatti o intervalli, è possibile definire l'impronta come un insieme di sottoimpronte, associate all'insieme delle propagazioni del nodo (cioè P_t), quindi costruendo delle sotto-impronte per ogni insieme contenuto in P_t.

In questo modo si possono svolgere i confronti che si vedranno in futuro e che sono essenziali per il calcolo della similarità senza problemi di ambiguità associati al primo approccio nel caso di più contatti, dove le propagazioni erano tutte indipendenti, dato che l'impronta risultante in quel caso sarebbe stata unica, ma i casi non sarebbero stati scindibili l'uno dall'altro, e la struttura temporale effettiva potrebbe portare a falsi positivi durante il confronto.

Questo argomento verrà espanso prossimamente.

Esempio 15. $impronta_3 = \{(TimeRespecting, 2), (NotTimeRespecting, 1)\}$

Questo esempio considera il nodo 3 del grafo visto nella sezione 1, vengono considerate tutte le propagazioni, e tra di loro 2 risultano Time-Respecting ed 1 non Time-Respecting

Esempio 16. (impronta nel caso di più contatti per arco). Prendendo l'esempio già discusso durante la definizione di insieme di propagazioni per più contatti associati ad un arco, l'impronta risultante sarà la seguente:

$impronta_2 = \{(\{(TimeRespecting, 2), (NotTimeRespecting, 0)\}, 2),$
$(\{(TimeRespecting, 1), (NotTimeRespecting, 1)\}, 2)\}$

Definizione 17. (struttura temporale di un nodo t). Si dice struttura temporale di un nodo t e si segna ST_t la coppia $(P_t, impronta_s)$, cioè la struttura temporale di un nodo formata dal suo insieme di propagazioni e dall'impronta stessa.

Questa definizione resta la stessa nel caso di intervalli e di contatti-intervalli multipli per arco.

Avere dei contatti o degli intervalli multipli per singolo arco potrebbe sembrare una sciocchezza rispetto alla definizione del problema principale, ma in realtà nascondono una complessita aggiuntiva immensa, specialmente durante

la fase di implementazione, in particolare durante il calcolo della similarità tra due nodi.

Per esempio, per controllare la struttura temporale di un nodo, si dovrebbero vedere tutte le possibili combinazioni tra i vari contatti di ogni arco e gli altri archi che condividono una quantità di contatti-intervalli maggiore di 1.

Definizione 18. (equivalenza di struttura temporale). Due nodi hanno la stessa struttura temporale se hanno la stessa impronta. Più formalmente:

$j \equiv s \iff impronta_j = impronta_s$

In realtà sarebbe più logico separare il concetto di impronta da quello di struttura temporale, che è comunque strettamente legata al grafo a cui appartiene il nodo, e non solo ai tipi di propagazioni che passano dal nodo, costruendo per esempio una funzione biettiva che mappa ogni propagazione del nodo j alle propagazioni del nodo s, ma il risultato sarebbe comunque lo stesso per gli scopi che si vedranno in seguito.

In particolare sarebbe meglio definire l'equivalenza temporale in base ad una funzione aggiuntiva che mappa le propagazioni del nodo da confrontare alle propagazioni dell'altro nodo, oltre al confronto delle impronte, questo non è stato fatto anche perché è più una formalità che un requisito, dato che l'equivalenza delle impronte fa presumere una stessa struttura temporale dei nodi confrontati.

Stesso discorso fatto precedentemente per più contatti-intervalli ad arco si applica anche in questo caso, la cardinalità dell'impronta aumenta esponenzialmente alla quantità di contatti-intervalli multipli per arco nel caso in cui tutta l'impronta sia considerata per tutte le possibili propagazioni, altrimenti è possibile costruire più impronte in base a tutte le combinazioni di contatti e definire la stessa struttura temporale se tutte le impronte hanno un corrispondente, non per forza diverso.

Inoltre, sempre nel caso di contatti-intervalli multipli per arco, non è scontato il confronto delle impronte, quindi bisogna stare particolarmente attenti nella definizione del problema.

Corollario 19. *Il confronto tra strutture temporali è una relazione di equivalenza*

Dimostrazione. (equivalenza di strutture temporali) Il confronto tra strutture temporali è una relazione di equivalenza dato che valgono le proprietà simmetrica, transitiva e riflessiva.

- (simmetrica). $\forall j, s \in V | j \equiv s \iff s \equiv j$
- (riflessiva). $\forall j \in V | j \equiv j$
- (transitiva). $\forall a, b, c \in V | a \equiv b \cap b \equiv c \implies a \equiv c$

□

Definizione 20. (similarità di struttura temporale). Due nodi sono simili temporalmente se da un sottoinsieme delle propagazioni di un nodo si ricava la stessa impronta dell'altro nodo, formalmente:

S è simile ad J e si scrive $(S \simeq J) \iff J \in V, P_{subJ} \subseteq P_J \Rightarrow impronta_{subJ} = impronta_S$

Si può notare come questa relazione non sia simmetrica ma antisimmetrica, cioè se J è simile ad S, non è detto il contrario, e questo fattore è molto importante per le definizioni di Temporal Graph Matching nella prossima sezione.

Corollario 21. *La similarità di due nodi è una relazione d'ordine (parziale perché non vale la* ***connex****, quindi non è totale)*

Dimostrazione. (similarità come relazione d'ordine) La similarità tra due nodi è una relazione d'ordine dato che valgono le proprietà antisimmetrica, transitiva e riflessiva.

- (antisimmetrica). $\forall j, s \in V | j \simeq s \cap s \simeq j \iff j \equiv s$
- (riflessiva). $\forall j \in V | j \simeq j$
- (transitiva). $\forall a, b, c \in V | a \simeq b \cap b \simeq c \implies a \simeq c$
- (no connex). $\forall a, b \in V a \not\simeq b \not\Rightarrow b \simeq a$

□

Corollario 22. *Più semplicemente j è simile ad s se e solo se $impronta_j \subseteq impronta_s$*

Dimostrazione. (similarità per sottoinsiemi delle impronte) Se J è simile ad S, tra i possibili sottoinsiemi di propagazioni dei nodi considerati ci sarà uno dei quali avrà una impronta che sarà uguale a quella del nodo S □

La similarità di un nodo ad un altro è stata definita perché sarà centrale durante la definizione di Temporal Graph Matching.

Corollario 23. *Se j ed s hanno la stessa struttura temporale allora sono simili. Formalmente $J \equiv S \Rightarrow J \simeq S$*

Lo stesso discorso non vale al contrario ovviamente, quindi durante i confronti bisogna controllare se un nodo i sia similare ad un altro j, allora anche $j \simeq i$.

Molto importante per la parte di pruning per la ricerca di nodi equivalenti in un sottografo è la contronominale di questo corollario, cioè $J \not\simeq S \Rightarrow J \not\equiv S$, utile durante la fase di controllo per il mapping tra nodi che si vedrà nella prossima sezione.

1.1.1 Grafi di flusso (Stream Graphs)

Una parentesi di rilevanza non trascurabile deve essere portata per i grafi di flusso, dato che sono un tentativo della generalizzazione di definizione formale per le reti temporali.

Nonostante la malleabilità e l'adattabilità delle reti temporali ai singoli casi (che adattano sia le definizioni che l'implementazione per il caso specifico di problema da affrontare), si è tentato di definire un framework generale di lavoro formale che si vuole ricollegare alla teoria dei grafi ed ampliarla tramite la componente temporale.

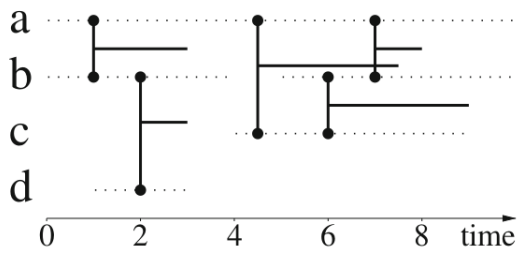


Figura 1 : Un esempio di grafo di flusso:$S = (T, V, W, E)$ con $T = [0, 10] \subseteq R, V = a, b, c, d, W = [0, 10] \times a \bigcup ([0, 4] \bigcup [5, 10]) \times b \bigcup [4, 9] \times c \bigcup [1, 3] \times d$, ed $E = ([1, 3] \bigcup [7, 8]) \times ab \bigcup [4.5, 7.5] \times ac \bigcup [6, 9] \times bc \bigcup [2, 3] \times bd$. In altre parole, $T_a = [0, 10], T_b = [0, 4] \bigcup [5, 10], T_c = [4, 9], Td = [1, 3], T_{ab} = [1, 3] \bigcup [7, 8], T_{ac} = [4.5, 7.5], T_{bc} = [6, 9], T_{bd} = [2, 3]$, and $T_{ad} = T_{cd} = \emptyset$

Definizione 24. (grafo di flusso). Un grafo di flusso S è definito come la tupla (T, V, W, E), dove T è l'insieme temporale , V è l'insieme dei nodi , $W \subseteq T \times V$ è un insieme di nodi temporali ed $E \subseteq T \times V \otimes V$ è un insieme di archi temporali .

Una definizione relativamente più complessa rispetto a quelle viste precedentemente, che aggiunge la componente temporale anche ai nodi, oltre ad aggiungere dei limiti alla componente temporale.

Definizione 25. (tempi di presenza di un nodo). Ogni nodo $v \in V$ ha un insieme di tempi di presenza $T_v = t, (t, v) \in W$.

Definizione 26. (tempi di presenza di un arco). $T_{uv} = t, (t, uv) \in E$ è l'insieme di tempi di presenza di un arco $u \to v$.

Definizione 27. (insieme dei nodi e degli archi derivati). $V_t = v, (t, v) \in W$ e $E_t = uv, (t, uv) \in E$ sono i nodi e gli archi associati ad un tempo o ad un intervallo, grazie al quale è possibile definire $G_t = (V_t, G_t)$.

Questo è il grafo indotto temporalmente molto simile ad uno snapshot, anche se in realtà t sarebbe un intervallo, quindi può essere visto come un insieme di layer del grafo.

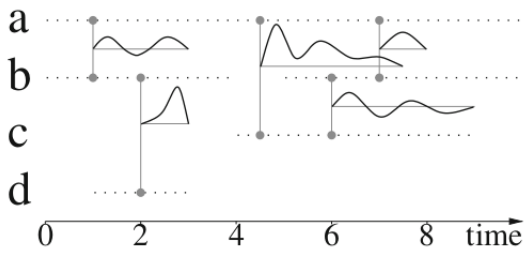


Figura 2: Un esempio di grafo di flusso pesato, come si può vedere, i pesi sono delle funzioni che variano con il tempo dell'intervallo, solo gli archi sono pesati.

È possibile derivare anche altre misure e metriche (come per esempio grado temporale medio di un nodo oppure la densità di un grafo di flusso che è la probabilità per cui, quando si sceglie a caso un istante di tempo e due nodi presenti in quel momento, questi due nodi siano collegati insieme in quel momento).

È possibile definire anche i concetti di grafo bipartito (sfruttando anche le componenti temporali), pesato (con funzioni che cambiano con il tempo in base al momento in cui si trova l'intervallo di contatto, molto utili per molti casi) e diretto, ma non sono utili per il fine ultimo di questo studio, per approfondimenti vedere [5, 2].

Durante questo studio però non verranno utilizzati anche perché non aggiungono nulla di particolarmente utile al problema che si affronta principalmente, che è quello del Temporal Graph Matching, e la formalizzazione di definizioni utili per la descrizione del problema è già stata fatta precedentemente.

1.2 Caratteristiche significative delle reti temporali

Vi sono molte caratteristiche che possono essere rilevanti e significative durante una analisi, per il graph matching sono di fondamentale rilevanza delle misure di identificazione e similarità locale per riuscire ad identificare le stesse caratteristiche di certi nodi e confrontare la similarità degli stessi per vedere se è possibile trovare delle funzioni di map per attuare gli isomorfismi e gli endomorfismi nello stesso grafo (nella sezione di graph matching si vedrà che servirà il calcolo degli endomorfismi per la rottura delle simmetrie nel grafo query).

Altre misure che descrivono le caratteristiche di una rete temporale sono la distribuzione dei gradi e dei tempi di contatto, molto utili per vedere se la rete segue una normale distribuzione di poisson, oppure la più frequente per reti temporali *Burstiness* che descrive molti contatti in brevi tratti temporali.

Una parte importante da definire e da modificare in base alle esigenze del caso è il concetto di sincronia e asincronia delle interazioni tra i nodi del grafo (gli agenti del sistema), che possono richiedere particolare attenzione durante la costruzione della struttura dei modelli e degli algoritmi, in quanto queste

caratteristiche influenzano attivamente l'infrastruttura di base e la definizione del problema.

Stesso discorso può essere fatto sui tipi di relazioni, cioè uno a uno, uno a molti e molti a molti, che potrebbero diminuire significativamente le prestazioni di qualsiasi algoritmo che lavora sulle componenti temporali e topologiche.

Una considerazione aggiuntiva può essere fatta sui contatti che avvengono in un intervallo temporale molto vicino cioè, andando a definire un intorno temporale per la propagazione sincronizzata dallo stesso nodo, si possono considerare delle propagazioni non Time-Respecting delle differenze tra i contatti o tra gli intervalli relativamente piccole, magari causate dalle latenze degli strumenti di misura utilizzati per la cattura del sistema di base.

Altre condizioni aggiuntive che possono essere aggiunte per il controllo della similarità tra nodi riguardano la valenza di un nodo rispetto a tutto il grafo, o misure di similarità che sfruttano una sotto-struttura del grafo temporale.

Caratteristiche aggiuntive possono essere ottenute utilizzando la trasformazione del grafo dinamico in altre forme e strutture, come per esempio grafi statici speciali o derivati dalla rete temporale, oppure utilizzando strutture come liste o min-heap per derivare proprietà aggiuntive.

Anche se rappresentare i dati come una rete temporale significa che alcune informazioni devono essere scartate per motivi di semplificazione e di digitalizzazione della componente temporale, questo spesso non è sufficiente per ottenere una rappresentazione su larga scala del sistema comprensibile ed analizzabile con tecniche comuni.

Forse il modo più ovvio di semplificare una rete temporale è trasformarla in una rete statica utilizzando delle tecniche che mantengano qualche forma di temporalità tra le caratteristiche del grafo. Nella prossima sottosezione verranno visti dei modi per ottenere delle reti statiche dalla rete dinamica di base.

Come si può intuire facilmente, i grafi temporali trovano molti punti in comune con modelli statistici a stati come i modelli di Markov e simili, infatti in alcune implementazioni i grafi temporali vengono solitamente accompagnati da modelli di Markov costruiti sui compartimenti di base e sulle classi dei singoli nodi.

Metriche globali per i grafi temporali sono principalmente derivate dalla struttura topologica del grafo, con integrazioni minime della componente temporale.

Alcune metriche temporali possono essere la distribuzione dei tempi per ogni nodo, la distribuzione dei tempi di ogni cammino Time-Respecting, ed i parametri derivati da queste distribuzioni ed i fit alle distribuzioni più conosciute.

Altre metriche sono derivabili dai grafi di flusso brevemente descritti precedentemente, per approfondimenti riferirsi sempre alla bibliografia [2, 5].

Di grande importanza per molti sono le strutture e caratteristiche **mesoscopiche**.

Il termine mesoscopico si riferisce alla scala tra macroscopico e microscopico. Nella network science, ciò significherebbe strutture più grandi dei nodi ma più piccole dell'intera rete e, in effetti, il termine è spesso usato nei contesti in cui vi è una esigenza di raggruppare i nodi in classi in base a come sono collegati

tra loro e al resto della rete. L'esempio principale di strutture mesoscopiche è la struttura della comunità in cui alcune reti hanno gruppi di nodi che sono fortemente connessi all'interno del gruppo e debolmente collegati tra loro.

Insiemi importanti durante molte analisi sono l'**insieme di influenza** e l'**insieme di origine dell'informazione**.

Si definisce l'insieme di influenza come l'insieme dei nodi raggiungibili dal nodo sorgente.

Alternativamente si può definire l'insieme di origine dell'informazione come l'insieme di tutti i nodi sorgente da cui è possibile raggiungere il nodo considerato.

Questi due insiemi possono essere visti come il passato ed il futuro del nodo e dell'informazione condivisa nella rete.

Di seguito verranno presentate varie definizioni per metriche e misure sui grafi temporali a contatti ma le definizioni possono essere ampliate ad intervalli:

Definizione 28. (distanza). La definizione di distanza è la stessa delle reti statiche adattata utilizzando i cammini Time-Respecting e le definizioni di base che dipendono dal problema da trattare, formalmente:

La distanza è una misura di ottimalità sui cammini che dipende da quello che si vuole ottimizzare, per esempio il numero di archi o i pesi degli archi.

Definizione 29. (durata temporale). Viene introdotta una nuova grandezza che è la durata, definita in un cammino come la differenza del tempo di arrivo dell'informazione e del tempo di inizio trasmissione dell'informazione, formalmente:

avendo i e j il nodo iniziale ed il nodo finale di un cammino Time-Respecting, ed $i^{>}$ ed $j^{<}$ i nodi appartenenti al cammino Time-Respecting che collega i e j tale che $i^{>} \in i.Adj \cap j^{<} \in j.Adj$, viene chiamata distanza il numero $e_{jj^{<}}.time - e_{ii^{>}}.time$.

Nel caso di intervalli bisogna tenere conto del primo contatto con il nodo nell'intervallo tra il primo ed il secondo nodo del path, e del primo contatto tra il penultimo ed ultimo nodo del path.

Definizione 30. (latenza temporale). La latenza è la durata minima tra tutti i tempi impiegati nel percorrere cammini tra due nodi .

Questa definizione è parallela a quella di distanza minima tra due nodi, infatti tutte e due si basano sulla minimizzazione di una caratteristica associata a due nodi.

Considerazioni algoritmiche sulla ricerca della latenza temporale sono molto simili alla ricerca dei cammini minimi, quindi come problema dinamico scomponibile in sottoproblemi con soluzioni ottimali.

Definizione 31. Si definisce la quantità $\varphi_{i,t}(j)$ come l'ultimo tempo prima di t in cui l'informazione partita da j sia arrivata a I

Definizione 32. (latenza di informazione). Viene chiamata information latency la funzione $\lambda_{i,t}(j) = t - \varphi_{i,t}(j)$, che misura di quanto è vecchia l'informazione.

Definizione 33. (vector clock). Definiamo inoltre $[\varphi_{i,t}(1), ..., \varphi_{i,t}(N)]$ come il vector clock , cioè il vettore delle latenze di informazione che fa vedere tutta la freschezza delle informazioni che arrivano al nodo da altri nodi.

Definizione 34. (Closeness Centrality). La closeness centrality è definita secondo la seguente legge:

$C_c(i) = \frac{N-1}{\sum_{i,j\neq i}^{n} \lambda_{i,t}(j)}$

dove N è il numero di nodi nel grafo

Definizione 35. (Betweenness Centrality). La betweenness centrality è definita secondo la seguente legge:

$C_B(i) = \frac{\sum_{i,k\neq j\neq i}^{n} v_i(j,k)}{\sum_{i,k\neq j\neq i}^{n} v(j,k)}$

Dove $\nu_i(j,k)$ è il numero di cammini minimi che passano per i e $\nu(j,k)$ è il numero totale di cammini minimi tra j e k

Esistono altre misure di centralità ed altri modi di interpretare le varie metriche che vengono usate per calcolarle(Path lengths, correlations, and centrality) ma non verranno viste dato che sono molto situazionali.

Altre misure sono importanti per la creazione di modelli random utili per la creazione di reti temporali randomiche.

La creazione della rete temporale randomica parte da una ipotesi iniziale (da questo il modello prende il nome di null model dato che dell'ipotesi viene costruita l'infrastruttura a partire dall'ipotesi arbitraria), che può essere per esempio il grado medio di ogni nodo, la distribuzione di intervalli medi di attivazione degli archi, o la probabilità di effettuare contatti tra nodi in certi intervalli temporali.

Non esiste un modello generalizzato di reti temporali randomizzate, data la natura arbitraria dei null models, piuttosto si possono o seguire delle linee guida sapendo la struttura base della rete che si vuole analizzare e modellare, oppure si può seguire un approccio di forza bruta fino ad esaurimento delle possibilità, cioè provare diversi null models e vedere quale tra questi restituisce i risultati più soddisfacenti. Seguendo una struttura di base, si può prendere ad esempio come infrastruttura un grafo temporale con certe proprietà e seguire strutture topologiche e temporali.

Randomized edges (RE) è un modello randomico per la creazione di grafi temporali, dove cambiamo le destinazioni con probabilità p.

Randomly permutated times è la controparte temporale, la struttura statica viene mantenuta, i tempi di contatto rimangono lo stesso numero ma vengono permutati tra i vari archi, la cardinalità dei tempi di contatto rimane uguale.

RP conserva l'insieme dei tempi di contatto, ma distrugge il comportamento tipico delle reti temporali di burstiness, un altro modello che distrugge ancora di più questi pattern di comportamento è il random times(RT) dove vengono scelti tempi randomici per ogni contatto, ovviamente la generazione dei tempi random non deve seguire per forza una distribuzione uniforme.

RE +RP è un ibrido tra i due modelli.

Random contacts(RC) mantiene la topologia e ridistribuisce i contatti.

Equal weight edge randomization(EWER) dove archi con lo stesso numero di contatti possono essere sostituiti. Questo modello mantiene la burstiness, ma ha bisogno di una rete abbastanza grande da avere abbastanza contatti.

Edge randomization(ER) è la generalizzazione in cui si può scambiare qualsiasi arco. La topologia della rete viene distrutta con questo modello, ma la temporalità relativamente conservata.

Time reversal(TR) con tempi inversi, cioè si percorrono i contatti in senso inverso, si cercano correlazioni tra la rete normale e quella inversa.

Euristico ed empirico è principalmente quasi tutto il procedimento di ricerca e di costruzione dei null models, ogni modello favorisce qualche caratteristica e distrugge le altre, solitamente si fa un analisi sfruttando un gruppo di questi modelli in modo da analizzare una buona parte delle caratteristiche della rete temporale, oppure si sfruttano ibridi(e.g. RE + RP) che permettano di mantenere un equilibrio tra prevedibilità e casualità.

Per grafi con molti contatti il risultato nell'utilizzo di EWER è abbastanza soddisfacente, alternato con altri modelli risulta come uno dei "migliori" metodi adottati nella creazione dei grafi temporali randomici.

Durante questo studio non verranno trattati nel dettaglio modelli generativi, anche se il bisogno si avrà durante la fase di sperimentazione, ma non si è voluto spendere altro spazio per questo tipo di studi dato che lo studio si concentra su una definizione generale di una infrastruttura flessibile ed utilizzabile nella maggior parte delle applicazioni.

1.3 Considerazioni sull'implementazione di reti temporali

Durante le definizioni si sono utilizzati i nodi come identificativi, ma solitamente nella realtà si hanno delle etichette associate ad interi che sono le entry delle varie liste di adiacenza, quindi bisogna non cadere in inganno con la teoria.

La rappresentazione di una rete temporale per contatti è molto spesso vantaggiosa.

Questo tipo di approccio favorisce la visione della rete temporale come una serie di diapositive (snapshot) della rete in certi tempi, in modo da vedere la rete temporale come una sequenza di reti statiche, ma vi sono anche altri tipi di rappresentazioni che vengono favorite dai contatti, per approfondimenti vedere [2].

Nel caso di reti a contatti più che definire una sequenza di reti statiche si può piuttosto definire una matrice di adiacenza tridimensionale dove la terza dimensione è il tempo(se i contatti appartengono all'insieme $\mathbb{N}$ ovviamente, magari denormalizzando i valori oppure, nel caso di intervalli, trasformare i numeri in interi, e segnare tutte i posti di una matrice che appartengono all'insieme ottenuto $interval \subset \mathbb{N} \times \mathbb{N}$), e lavorare su questa matrice, tramite questa definizione è possibile .

Questa matrice di adiacenza viene definita nel seguente modo:

caso contatti

$$TemporalAdj(i,j,t) = \begin{cases} 1 & \text{if there is an edge from i}\rightarrow\text{j at time t} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

caso intervalli

$$TemporalAdj(i,j,t) = \begin{cases} 1 & \text{if there is an edge from i}\rightarrow\text{j}\cap\text{t} \in e_{ij}.interval \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Nel caso di liste di adiacenza, bisogna semplicemente tenere conto dei contatti nel caso in cui si vogliano mantenere singoli contatti per arco come diversi, altrimenti si possono tenere degli insiemi associati ad ogni arco che tengono conto dei tempi di contatto dell'arco a cui sono associati.

Come già accennato precedentemente, le reti temporali possono avere altre rappresentazioni (trasmission graph, line graph, reachability graph).

Le reti statiche più semplici che codificano veramente alcuni effetti temporali sono i grafi di raggiungibilità (*reachability graph*). Questi grafi hanno un arco diretto (i, j) se un evento può raggiungere j da i attraverso un percorso Time-Respecting. Infatti, grazie a questi grafi di raggiungibilità, è possibile stabilire delle metriche abbastanza rilevanti per il calcolo di varie statistiche e per la modellizzazione dei grafi temporali, per esempio si può definire una specie di betweeness sulla base del grado di un nodo.

Si può dimostrare che per ogni rete temporale si possono costruire uno o più reachability graph, e si può dimostrare che per ogni reachability graph si può costruire una rete temporale che lo rispetti.

Un modo più elaborato di ridurre le reti temporali a quelle statiche è l'estrazione di dorsali specificatamente adattate rispetto ai processi di diffusione su reti temporali, per esempio utilizzando le propagazioni Time-respecting da nodo a nodo, e costruendo una rete in cui tutti i nodi siano raggiungibili in qualche modo da un nodo a scelta, e quindi costruire i modelli sulla base di questo nuovo grafo statico facilmente utilizzabile.

Altro modo di rappresentare le reti dinamiche come reti statiche usa il concetto di line graph, dove il ruolo dei nodi e degli archi viene invertito. Viene costruito quindi un trasmission graph, che incorpora nella definizione dei nodi e degli archi del line graph anche la componente temporale(sia del momento del contatto, sia per quanto tempo avviene).

Un approccio comune che può essere interpretato come proiezione a reti statiche consiste nell'utilizzare reti multistrato (multilayer), utilizzando degli snapshot ad intervalli secondo il seguente ragionamento: il tempo viene suddiviso in intervalli conseguenti e gli strati di una rete multistrato corrispondono alle reti aggregate per ciascun intervallo. Una volta accoppiati gli strati si possono quindi applicare al sistema metodi di modellizzazione e mining per reti (statiche) multistrato. È importante sottolineare che gli strati in tale proiezione sono ordinati per tempo.

Invece di ridurre i dati della rete temporale in reti statiche, si può provare a conservare alcune ma non tutte le componenti temporali.

Un esempio sono le statistiche dei tempi tra i contatti. È stato riconosciuto già dai primi studi che spesso i tempi tra gli eventi, sia per i nodi che per i collegamenti, hanno distribuzioni heavy-tailed (bursty come già accennato precedentemente). Quindi basta riconoscere e simulare questa proprietà senza tenere conto di altre caratteristiche temporali per avere una rete temporale che simuli bene la maggior parte delle situazioni che si vorrebbero studiare e modellare.

Un altro modo di semplificare le reti temporali è ignorare le dinamiche dei contatti e pensare semplicemente ai collegamenti presenti tra la prima e l'ultima osservazione di un contatto nei dati e ignorare il preciso tempismo delle interazioni temporali. Rispetto alla semplificazione del sistema sotto forma di dinamiche bursty su reti statiche, questa tecnica enfatizza strutture temporali più durature come la crescita generale e il declino dell'attività nei dati.

Esistono anche altri tipi di rappresentazione(per esempio si può aggiungere il fatto per cui l'attraversamento di un arco comporta un tempo ben definito, e costruire la rete e gli algoritmi di conseguenza) che si basano su altre proprietà, ma non verranno viste nel dettaglio dato che lo scopo di questo studio è la costruzione di un framework utile nella maggior parte dei casi e particolarmente dettagliato per il graph matching.

Per la rappresentazione utilizzata durante il graph matching, viene utilizzato una rappresentazione standard arricchita di contatti tra i vari nodi tramite un hashset per avere delle buone prestazioni nel controllo dell'adiacenza. vengono utilizzati grafi con singoli contatti per arco ma gli algoritmi utilizzati dovrebbero essere generali per più contatti ad arco.

Il caso ad intervalli non è stato implementato ma i ragionamenti di base sono sempre gli stessi (cambia solo il modo di classificare le propagazioni ed il problema dei tempi di contatto iniziali dell'intervallo che si vedrà nella prossima sottosezione)

Come si può intuire, è abbastanza semplice aggiungere la componente temporale a partire da un grafo statico, ma non si deve cadere nella banalità superficiale del problema, dato che nasconde una quantità non indifferente di problemi legati a molte strutture reali, come per esempio la consistenza interna di altre strutture di supporto che dipendono dalla componente temporale, magari ricorsivamente, e che quindi richiedono una complessità rilevante all'analisi di grafi di grandi dimensioni (e.g. la durata di un cammino, le latenze di informazioni, o altre misure che dipendono da più contatti-intervalli).

Alternativa all'implementazione diretta nel grafo statico e nella sua struttura è la creazione di una rappresentazione completamente basata sulle componenti temporali, per esempio che sfrutta contatti e propagazioni per il modello che è stato definito durante questo studio, oppure che implementa tutte le possibilità offerte dai grafi di flusso descritti brevemente nelle sezioni precedenti.

Nonostante l'importanza delle propagazioni nella definizione e comprensione del problema che si affronta e verrà affrontato nel dettaglio successivamente, cioè il graph matching, molto spesso le informazioni che posseggono sono ridondanti e derivabili da alcune componenti del grafo temporale semplice, anche perché l'integrazione delle componenti temporali è stata fatta direttamente su

una rappresentazione statica di un grafo quindi non si può ottenere vantaggio sul tenere delle strutture per le propagazioni.

Questa parentesi sulla utilità bassa delle propagazioni (solo in questo caso dato che si doveva riutilizzare altro codice ed adattarlo per la ricerca di isomorfismi su grafi temporali) non toglie la loro indeterminabile importanza per la definizione di molti problemi che considerano una struttura locale per la ricerca di qualcosa, come ricerca di clique, motifs, communities, ed altre strutture legate ad una visione microscopica e mesoscopica del grafo temporale.

Una possibile ottimizzazione al continuo calcolo delle strutture e delle impronte digitali sta nel tenere una struttura interna per ogni nodo che cambia all'aggiunta di archi tra nodi (ovviamente correlati di tempo di contatto o intervallo di contatto), anche se di fondo questa computazione di struttura è effettuata soltanto per il calcolo dei domini di compatibilità che si vedranno in seguito.

1.4 Problemi tipici delle reti temporali

La maggior parte dei problemi che potrebbero risaltare durante la costruzione di una rete temporale e dell'analisi della stessa sono solitamente collegati con il problema di base che si cerca di analizzare.

Per esempio la trasposizione di una rete di eventi come modelli epidemici o metabolici comporta innumerevoli caratteristiche aggiuntive da tenere conto durante l'analisi sia temporale che topologica.

Altre possibili complicazioni che possono venire a galla vengono dalla banalità di analisi e comprensione con cui si tratta il problema di base, che solitamente non dipende soltanto dalla componente temporale o dalla topologica della rete con cui viene rappresentato, ma da molti altri fattori che influenzano i comportamenti all'interno della rete.

Quindi, più si tenta di rappresentare una realtà effettiva, più problemi si incontreranno durante il percorso di costruzione, implementazione ed analisi.

Per esempio, nel caso di intervalli temporali, che sembra non aggiungano complessità aggiuntiva, si può facilmente notare che nel momento della propagazione bisogna tenere conto dell'intervallo in cui è avvenuta la propagazione precedente, quindi fare iniziare l'intervallo della nuova propagazione in un range di quella precedente o, per semplificare, dal minimo dell'intervallo risultante (questa considerazione del minimo dell'intersezione è stata già fatta nella definizione del propInterval di una propagazione).

Definizione 36. Dati p_{itj} e p_{tje} due propagazioni, l'intervallo della propagazione $p_{tje}.e_s.interval = p_{itj}.propInterval$ e non l'intervallo di propagazione dell'arco tj.

Questa definizione è stata creata per simulare la scomposizione di cammini in singole propagazioni, adattando l'infrastruttura definita precedentemente ad una definizione corretta rispetto al problema comune.

Questa definizione fa risalire molti problemi dato che definire l'intervallo secondo una specifica propagazione da altre propagazioni dei nodi precedenti allo

stesso è molto ambiguo, quindi non isolato come problema, che invece nasconde una ricorsione interna per il calcolo effettivo di questi intervalli, ricorsione dipendente dal cammino che si sta considerando.

Nell'implementazione questo problema è stato evitato grazie a dei controlli aggiuntivi molto simili a quelli dell'impronta, ma che utilizzano il singolo arco da mappare (o l'intervallo nel caso di grafo ad intervalli) per vedere la fattibilità del mapping effettivo, questo per vedere se la propagazione sia effettivamente correttamente mappata rispetto al cammino utilizzato per arrivare dal nodo di partenza al nodo destinazione.

Prima e durante la costruzione di reti temporali, bisogna tenere conto di quante informazioni si perdono e quanto valga la pena costruire una rete temporale, cioè considerare la fattibilità, l'accuratezza e la manutenzione della rete, svolgendo una analisi dei rischi effettiva, dato che la costruzione di una rete, correlata dalla costruzione dei modelli associati, non è qualcosa da prendere sottogamba.

Nella costruzione di reti temporali randomiche considerazioni sono state già fatte sulla base di modelli randomici, che comunque non tenevano conto di alcune caratteristiche o metriche utilizzabili, ma che utilizzavano degli approcci ***trial and error***, abbastanza poco performanti.

Molto spesso inoltre vi sono dei problemi di scalabilità, in quanto se i cambiamenti di una rete statica sono minimi temporalmente (troppo lenti o troppo veloci), ma abbastanza topologicamente (aggiunta di nodi o archi aggiuntivi), la costruzione dell'infrastruttura temporale diventa tediosa ed inutile a lungo termine, quindi bisogna svolgere un confronto della rapidità di cambiamento della rete statica e temporale prima di continuare con la costruzione di una rete.

Sempre riguardante la scalabilità è la dimensione stessa della rete ed il tipo, come si è visto durante la definizione dei vari problemi, nel caso di più contatti o intervalli, le soluzioni sono molto pesanti computazionalmente dato che si visitano tutte le possibilità possibili, quindi si aumenta fattorialmente la complessità asintotica temporale dei possibili algoritmi che si potrebbero utilizzare.

Questi problemi possono essere comunque evitati tenendo delle strutture aggiuntive, sacrificando lo spazio per il tempo, e diminuendo le ridondanze di computazione ripetuta per il calcolo delle strutture temporali delle componenti di un grafo.

Alla banalità della costruzione superficiale di una rete temporale non contribuisce neanche la visualizzazione e rappresentazione di una rete, che potrebbero comprometterne sia la complessità che l'interpretazione, specialmente ad occhi di terzi che non hanno molta esperienza sulla infrastruttura temporale su cui dovrebbero lavorare.

Uno dei problemi più importanti e che preclude l'usufruizione di molte tecniche basata su di essa è l'intransitività, caratteristica basilare di molti algoritmi che lavorano sui grafi, come per esempio la ricerca di cammini ottimali oppure la ricerca di componenti connesse.

È possibile aggiungere componenti e vincoli aggiuntivi ad una rete temporale ma bisogna fare particolarmente attenzione alla possibilità di far diventare

l'infrastruttura troppo specifica per un singolo problema e poco flessibile per espansioni di codice.

Nonostante la letteratura scientifica sulle reti temporali stia crescendo con gli ultimi anni, non è ancora per nulla paragonabile alla letteratura sulle reti statiche, quindi la maggior parte degli argomenti è ancora territorio inesplorato.

La letteratura sulle reti temporali è molto sparsa, vi sono molte ambiguità durante lo studio dell'argomento e la costruzione di una rete specifica per un singolo caso particolare, e solitamente si lascia la libertà di scelta ed interpretazione. Tentativi di standardizzazione e formalizzazione sono stati fatti con i grafi di flusso [5, 2]citati precedentemente, ma ancora non sono stati testati totalmente per qualsiasi caso data la loro giovinezza.

Le reti temporali sono sistemi complessi particolarmente delicati e ingannevoli, anche perché la componente temporale non è un semplice numero o intervallo, lo scorrere del tempo in qualche modo è qualcosa di complesso e non semplice da studiare.

Nel caso di modelli randomici, per essere significativa la costruzione ed analisi della rete bisogna avere delle interazioni binarie, cioè tra due nodi, in modo da mantenere sia un certo livello di prevedibilità, sia una certa randomicità nel comportamento della rete, in modo che simuli bene i sistemi reali. Inoltre le reti temporali costruite non dovrebbero essere né troppo casuali né troppo regolari per non allontanarsi troppo dalla rete statica a cui sono state interfacciate.

Bisogna tenere particolare attenzione alla velocità dei contatti rispetto alla velocità di cambiamento del sistema dinamico della rete. Essendo un sistema dinamico, la rete temporale è particolarmente difficile da analizzare, nodi possono rientrare ed uscire, archi possono non essere percorsi, bisogna tenere conto di molte variabili ed alcune potrebbero non essere intuibili.

Nella ricerca di motifs, cliques, community, e altri tipi di pattern che hanno una certa regolarità esistono diverse interpretazioni, come per il problema trattato in questo studio che è quello del Temporal Graph Matching, dove molti autori danno definizioni completamente diverse o non chiare, che molte volte lasciano casi abbastanza semplici ed utili in molte applicazioni, questo tipo di problemi verrà visto nella prossima sezione.

Importanti nell'analisi di reti temporali non sono soltanto le cause topologiche o pratiche per cui avvengano certe interazioni, non vi sono solo interpretazioni ed analisi differenti della natura temporale ma, molto spesso, vi è una ricerca anche delle origini delle interazioni tra i vari agenti, per esempio lo studio del contatto tra due nodi che, come già detto, assume un comportamento bursty più che seguire una distribuzione conosciuta (e.g. poisson).

Lo studio delle reti temporali potrebbe giovare di molti studi che vengono fatti sulle reti statiche(come per esempio la colorazione dei grafi, o la grande quantità di metodi utili per la ricerca dei cammini minimi, per la ricerca delle chiusure transitive), ma dato che il soggetto è abbastanza giovane, nonostante degli studi sono stati fatti fin dagli anni novanta in modo approssimativo, vi è la necessità di espandere molte delle ricerche che vengono fatte su grafi, multigrafi, etc. anche ai grafi temporali.

Si è accennata alla modellizzazione e generazione di reti temporali con caratteristiche particolari, in questo campo di studio vi sono dei metodi molto simili a quelli delle reti statiche, con opportune modifiche che variano da caso a caso e che vedono di particolare rilevanza la componente temporale, quindi costruiscono delle reti sulla base di quello che si vorrebbe ottenere, magari adottando metodi unici e non standardizzabili, quindi poco generali.

Durante tutto questo studio si è data particolare attenzione a dei descrittori di struttura temporale, come l'impronta temporale definita precedentemente, si potrebbe giovare comunque di altri tipi di strutture che rispecchino la vera natura della componente temporale, che siano create e studiate per essere utili nella analisi delle reti dinamiche, e che riescano a descrivere proprietà globali o mesoscopiche in modo da poter trovare delle similarità tra le varie reti temporali.

Le reti temporali a contatti restano comunque lo state of the art di molti studi (quasi tutti gli studi utilizzano singoli contatti). Questo anche perché riescono a rappresentare delle situazioni abbastanza normali e non troppo banali senza intaccare pesantemente la complessità del sistema di base. Resta il fatto che singoli momenti non rappresentano molte delle situazioni reali più importanti, che invece hanno degli agenti che agiscono per intervalli.

Per altri problemi riguardanti il singolo sistema da voler rappresentare vedere [2] che fa un lavoro immenso nelle possibilità di rappresentazione di sistemi che contengono ed incorporano la componente temporale in qualche modo.

2 Isomorfismi di grafi

La ricerca di isomorfismi e pattern ripetuti all'interno della struttura di un grafo è qualcosa che ha prodotto una quantità immensa di materiale scientifico, anche per il fatto che i risultati sono molto spesso rilevanti in molti campi e fanno luce su certe meccaniche che non si potrebbero apprendere in sistemi complessi, come comportamenti emergenti o agenti che svolgono ruoli diversi ma che, alla fine, comunicano molto spesso tra di loro scambiandosi messaggi e informazioni.

In modo molto approssimativo e generale, la ricerca di isomorfismi in grafi si riduce alla ricerca delle strutture che rispettano le condizioni stabilite in base al caso ed al tipo di struttura che si sta analizzando.

Nel caso dei grafi temporali, la ricerca di isomorfismi è importantissima in molti campi, e spesso molto differente in base al problema di base che si cerca di risolvere.

Infatti , come si vedrà nelle prossime sottosezioni, molti autori non sembrano concordare su un framework unico da utilizzare e sulle definizioni da sfruttare per la descrizione del problema di base, nonostante i problemi che trattano sono molto spesso molto simili ed indipendenti dal problema di base.

Per esempio, da un lato si ha una ricerca di motif temporali che adotta una ricerca edge-driven, che tiene conto solo di una singola definizione abbastanza stretta anche per il fatto che l'algoritmo vero e proprio è incentrato sull'ottimizzazione per la ricerca ed il conteggio di piccole query-pattern(tre nodi per query, si vedrà in seguito)[11] e dall'altro una definizione più generica ma che riguarda soltanto una serie di sotto-problemi e dipende da condizioni semplici che non espongono realmente la complessità del problema[12, 10, 8].

Il problema sembra quindi non essere stato affrontato ancora nel dettaglio ed un framework effettivamente utilizzabile non è stato ancora trovato (o almeno non è stato trovato durante le ricerche, magari un ricercatore sconosciuto ci ha già pensato ma non sta ricevendo l'attenzione che gli spetta ed i motori di ricerca non aiutano).

Durante tutto questo studio, sono state date delle definizioni che saranno chiave per la definizione del problema di base, e quindi necessario non dimenticare quello che è stato detto precedentemente e rivedere le definizioni nel caso in cui ci siano dei dubbi.

Le definizioni date precedentemente e prossimamente non sono comunque totalmente generali, dato che esisterà quasi sicuramente un caso che ha delle condizioni diverse, anche se tutto il framework descritto è stato costruito per essere il più generale possibile, grazie alle definizione di classi delle propagazioni che è facilmente espandibile con altre classi.

Verranno accennate altre definizioni e concetti che vengono esposti in altre ricerche [13, 12, 11, 7, 6, 10, 3], ed i concetti saranno talvolta implementati nel caso di studio che si prenderà ad esempio, ma non si implementeranno dei controlli globali dato che non tengono conto di gruppi di interazioni indipendenti e quasi sempre sono dipendenti dal problema di base. Nel caso in cui si abbia il bisogno di espandere l'infrastruttura con controlli aggiuntivi che non siano locali, basta aggiungere dei controlli a fine ricerca dell'isomorfismo che controlli se i

vincoli stabiliti (che siano mesoscopici o globali) sono rispettati nel sotto-grafo corrispondente.

Di seguito verrà mostrato il graph matching su grafi statici per presentare l'algoritmo che si utilizzerà durante il matching e le sue modifiche, non si scenderà troppo sui dettagli per il Sub-Graph Isomorphism per i grafi statici ma verranno presentate le idee di base, dato che verrà sviscerato il concetto durante la discussione delle possibili modifiche apportabili al sistema di base di strutture e algoritmi per l'adattamento ai grafi temporali.

Verso la fine verranno presentati gli algoritmi utilizzati e le descrizioni di tali algoritmi al fine di una comprensione completa.

Infine, verranno visti e discussi dei problemi riguardanti l'isomorfismo su grafi e possibili soluzioni , questi problemi sono molto rilevanti dato che chiariscono ancora una volta il collegamento delle definizioni di grafi temporali e isomorfismi alla casistica del problema.

2.1 Graph matching per grafi statici

Il problema del graph matching su grafi statici è stato smembrato in molti modi già dall'inizio degli studi sulla teoria dei grafi.

Gli studi sono innumerevoli, le soluzioni spaziano dalle soluzioni semplici di visita di tutte le possibili soluzioni alle più complesse, che utilizzano le proprietà parallele della computazione della ricerca dei sotto-grafi in modo da dividere il problema e ottimizzare l'uso della singola macchina(utilizzando la GPU integrata o la discreta, il multi-threading ed altre tecnologie che permettono la parallelizzazione della computazione), oltre all'utilizzo di euristiche aggiuntive per la diminuzione dello spazio di ricerca, anche perché il problema della ricerca degli isomorfismi in grafi è NP-hard come minimo, mentre la ricerca di isomorfismi su sotto-grafi è NP-completo, quindi delle soluzioni che diminuiscano il tempo di calcolo e la complessità asintotica in generale sono assolutamente ben accette.

Tra le implementazioni che fanno un controllo di isomorfismo su singolo grafo vi sono principalmente algoritmi che attuano una trasformazione del grafo in una forma canonica univoca che è possibile identificare e confrontare per vedere se le condizioni di isomorfismo sono soddisfatte (tra questi algoritmi risaltano **Bliss, Saucy, Conauto e Traces**).

La ricerca della forma canonica non è un problema semplice, ma esistono molti studi a riguardo.

Per il problema riguardante ricerca di isomorfismi su sotto-grafi esistono quindi altrettante soluzioni, dato che il problema è una generalizzazione della ricerca di isomorfismi tra grafi.

Queste soluzioni del Sub-Graph isomorfism non verranno viste tutte, ma verranno solo accennate approssimativamente nelle componenti più importanti che influenzano direttamente le prestazioni ed i ragionamenti logici per la ricerca di soluzioni ottimali.

Verrà visto invece nel dettaglio l'algoritmo per la ricerca di isomorfismi su sotto-grafi **RI**[1, 9], che incorpora molte euristiche di altri algoritmi, e favorisce la velocità dell'algoritmo e la diminuzione dello spazio di ricerca delle soluzioni.

Un semplice algoritmo di enumerazione per trovare tutti gli isomorfismi del sotto-grafo (cioè le occorrenze) di un grafo di pattern in un grafo target funziona come segue: si generano tutte le possibili corrispondenze tra i vertici dei due grafi e si controlla se qualsiasi corrispondenza generata è un isomorfismo di sotto-grafo (che verrà chiamato match). Considerando che questo algoritmo è totalmente inefficiente se implementato in modo ingenuo, oltre ad essere una vera e propria forza bruta, serve come punto di partenza.

Tutte le corrispondenze possono essere rappresentate utilizzando un albero dello spazio di ricerca.

L'albero ha una radice fittizia. Ogni nodo rappresenta una possibile corrispondenza tra qualche vertice u del grafo query G e qualche vertice u' del grafo target G'. Il percorso dalla radice a un dato nodo rappresenta una corrispondenza parziale tra G e G'. Solo alcuni percorsi dalla radice alle foglie corrispondono agli isomorfismi del sotto-grafo tra il grafo query e il grafo target, ed è proprio quelli che si devono cercare primariamente in modo da avere un matching veloce.

Trovare una soluzione per il problema di ricerca di isomorfismi in sotto-grafi è intrinsecamente difficile e quindi l'efficienza di qualsiasi software che utilizza algoritmi di ricerca di isomorfismi dipende in gran parte da

1. trovare euristiche efficienti per rendere gli algoritmi di isomorfismo più veloci, quindi utilizzarle in modo ibrido così da avere le migliori caratteristiche possibili;

2. ridurre il numero di chiamate di isomorfismo del sottografo, quindi diminuire sommariamente il tempo di esecuzione;

3. diminuire e rendere semplici e veloci le condizioni di isomorfismo, in modo da avere una base solida e che non abbia troppo carico sul sistema.

Molti algoritmi funzionano svolgendo dei confronti locali rispetto al nodo, attuando delle regole di lookahead per il controllo delle condizioni per un nodo candidato ed i nodi non ancora mappati collegati al nodo da mappare, questa funzione è utile in molti casi (è stata pure implementata nel codice di implementazione del grafo temporale una funzione che controlla regole di lookahead fino al terzo livello con il nome di **testNodeMapping**, ma non verrà utilizzata nello studio vero e proprio perché RI non utilizza regole di lookahead dispendiose) ma è molto costosa nelle performance, ed il pruning può essere fatto senza troppi problemi per piccoli intorni di un nodo senza spendere troppo tempo a controllare delle condizioni ridondanti . Nel momento in cui un nodo non rispetta le condizioni per essere mappato, si ritorna indietro (rollback o backtracking ad un nodo più in alto nell'albero di ricerca) e si controllano altri possibili mapping.

Durante la visita vengono controllate le condizioni di isomorfismo per verificare gli abbinamenti parziali. Quando le condizioni non sono soddisfatte,

l'algoritmo elimina i rami sottostanti e torna indietro sui nodi principali dell'albero di ricerca. La dimensione dell'albero dello spazio di ricerca aumenta esponenzialmente con la dimensione del grafo.

L'obiettivo principale è eliminare cammini dalla radice alle foglie che non sono degli isomorfismi il più velocemente possibile.

Una caratteristica che influenza molto la strategia di ricerca di soluzioni è l'ordinamento dei nodi del grafo query che vengono analizzati durante la fase di matching.

Grazie a questo ordinamento, è possibile diminuire lo spazio di ricerca molto velocemente, e trovare le soluzioni è conseguentemente più veloce.

L'ordinamento dei nodi può essere di due tipi: statico e dinamico.

Tramite un ordinamento statico, si definisce un ordinamento iniziale che è uguale per qualsiasi percorso da radice a foglia. Nell'ordinamento dinamico, vi è un ordine per ogni singolo cammino e questi ordini sono solitamente stabiliti a tempo di esecuzione.

Tra gli algoritmi più utilizzati per la ricerca di isomorfismi in sottografi al giorno d'oggi vi è VF (con le successive versioni che ottimizzano e aumentano le performance, fino ad oggi la versione è VF3).

VF (acronimo di Vento-Foggia) utilizza una strategia di ricerca dinamica.

Data una soluzione parziale, per prima cosa sceglie i vertici del pattern non corrispondenti aventi archi che hanno sorgente vertici della soluzione parziale; quindi sceglie quei vertici non corrispondenti che hanno archi con destinazione in vertici della soluzione parziale. Per ridurre lo spazio di ricerca, l'approccio utilizza le seguenti due euristiche di lookahead. Una coppia di mappatura (u, u '), dove u ed u' sono rispettivamente dei vertici dei grafi query e target, è considerata una corrispondenza valida se soddisfa le seguenti regole:

1. u ed u' sono entrambi vicini dei vertici che sono stati già mappati;

2. il numero di vertici del pattern non abbinati che sono vicini ai vertici abbinati e sono collegati con u deve essere minore o uguale al numero di vertici del target non abbinati che sono vicini ai vertici abbinati e sono collegati con u' .

La regola (ii) è suddivisa in quattro casi a seconda della direzione dei bordi coinvolti tra i vicini di u e l'insieme in (i).

Vi era inoltre un altra regola di lookahead che considerava tutti i nodi connessi, ma non viene descritta dato che viene utilizzata solo per sottografi indotti.

Si distinguono comunque due grandi famiglie di algoritmi di ricerca di isomorfismi : algoritmi Tree Search (TS) e algoritmi Constraint Propagation (CP).

Gli algoritmi di ricerca ad albero (Tree Search) formulano il problema di corrispondenza in termini di una rappresentazione State Space Representation (SSR), che consiste nell'esplorazione di un albero dello spazio di ricerca. Ogni stato dell'SSR corrisponde a una soluzione parziale. Lo spazio di ricerca viene visitato in un primo ordine profondo e l'euristica è concepita per evitare di

esplorare parti dello spazio di ricerca utilizzando regole aggiuntive. Gli algoritmi di questa classe includono l'algoritmo di Ullmann, VF2, VF3, RI e RI-DS.

Nei metodi di propagazione dei vincoli (Constraint Propagation), il problema di corrispondenza dei sottografi è rappresentato come un problema di soddisfazione dei vincoli (Constraint Satisfaction Problem). I nodi del grafo query sono rappresentati come variabili ed i nodi del grafo destinazione rappresentano i valori che possono essere assegnati a tali variabili. Gli archi vengono tradotti in vincoli che devono essere soddisfatti.

Gli algoritmi CP calcolano prima un dominio di compatibilità per ogni nodo del grafo pattern-query e quindi propagano iterativamente i vincoli per ridurre tali domini e filtrare i nodi candidati per la corrispondenza. I metodi CP più famosi sono nRF +, Focus-Search e LAD.

Diverse tecniche di filtraggio, come il forward-checking (controllo a priori), eliminano i rami dell'albero di ricerca propagando i vincoli per rimuovere i valori dai potenziali domini di compatibilità. Un ramo viene eliminato quando un dominio diventa vuoto.

Nel forward-checking, prima viene assegnata una variabile, quindi tutti i vincoli che coinvolgono tali variabili vengono propagati per rimuovere i valori da altri domini che non sono coerenti con l'assegnazione corrente (inferenza).

I metodi basati sull'inferenza, che propagano i vincoli fino alla convergenza (ad esempio LAD), riducono al massimo il tempo di ricerca. Sfortunatamente, tale inferenza viene fatta al prezzo di un maggiore costo computazionale e di preprocessing. D'altra parte, quando la verifica dei vincoli viene applicata solo localmente (ad esempio, l'inferenza locale utilizzata da FocusSearch e le regole di pruning di VFlib), è fondamentale definire una strategia di ricerca che cerchi di sfoltire lo spazio di ricerca in modo ottimale ed il più presto possibile a basso costo.

Le caratteristiche più importanti per le performance di un algoritmo descritte precedentemente vengono, per la maggior parte, implementati in **RI**.

RI adotta una strategia di ricerca basata solo sulla topologia del grafo query. L'ordine viene scelto per creare i vincoli il prima possibile nella fase di matching. Approssimativamente, i vertici che hanno alta valenza e che sono altamente connessi con i vertici precedentemente presenti nell'ordinamento tendono a venire prima nell'ordinamento finale delle variabili. Durante la fase di abbinamento, RI non applica regole di pruning o inferenza onerose per il calcolo e per la computazione. Un potente ordinamento dei vertici del grafo query insieme alla verifica di vincoli non troppo onerosi, è più efficiente di una procedura di inferenza locale o globale.

L'algoritmo RI si divide in quattro fasi principali:

1. La prima fase di RI calcola i domini di compatibilità $Dom(q)$ per ogni nodo del grafo query q , questo dominio è l'insieme di nodi nel grafo target che possono essere mappati ad uno dei nodi del grafo query in base a condizioni topologiche come il grado del nodo ed altre metriche o strutture confrontabili(poi si vedrà quali condizioni aggiungere per integrare la componente temporale) . Questa fase aumenta le performance proprio

perchè solo i nodi del grafo target nel dominio di q saranno considerati come possibili candidati per il match di un nodo durante la fase di ricerca dell'isomorfismo. Formalmente, sia $Q = (V_Q, E_Q)$ un grafo chiamato query e $T = (V_T, E_T)$ un grafo chiamato target. Un nodo $t \in V_T$ è compatibile con un nodo $q \in V_Q$ se la seguente condizione è soddisfatta:

- $deg(q) \leq deg(t)$

Dopo il calcolo dei domini di compatibilità, una tecnica di Arc Consistency (AC) viene applicata per togliere possibili candidati che non rispettano i requisiti minimi. La procedura AC afferma ce se esiste un arco tra due nodi del target, q e q', allora per ogni nodo target tra quelli compatibili (Dom(q) come sorgente e Dom(q') come destinazione) deve esistere un arco tra i nodi del dominio del nodo sorgente ed almeno un nodo tra quelli del nodo destinazione e viceversa. Ciò implica che se un nodo di destinazione t appartiene al dominio di un nodo di query q ma non soddisfa la condizione AC, può essere rimosso da Dom (q).

2. La seconda fase riguarda l'ordinamento dei nodi del grafo query dove RI calcola l'ordine in cui i nodi del grafo query saranno processati durante la fase di matching. La base dell'algoritmo è quella di definire un possibile ordine all'elaborazione dei nodi della query. In RI, i nodi di query che hanno entrambi un grado elevato e sono altamente connessi ai nodi già presenti nell'ordinamento parziale vengono prima nell'ordinamento finale. Formalmente, sia n il numero di nodi nel grafo query e $\mu^{i-1} = (q_1, q_2, ..., q_{i-1})$ l'ordinamento parziale fino a (i − 1)-esimo nodo, con i < n. Viene anche definito un insieme U^{i-1} di nodi non ancora nell'ordinamento.Per scegliere il nodo successivo dell'ordinamento, vengono definiti, per ogni nodo del grafo query candidato q, tre insiemi:

   (a) $V_{q,vis}$: L'insieme dei nodi in μ^{i-1} e adiacenti a q;

   (b) $V_{q,neig}$: L'insieme dei nodi in μ^{i-1} che sono adiacenti ad almeno un nodo in U^{i-1} e connessi a q;

   (c) $V_{q,unv}$: L'insieme dei nodi adiacenti a q che non sono in μ^{i-1} e non sono adiacenti a nodi in μ^{i-1} .

   Il prossimo nodo nell'ordinamento è quello che rispetta le seguenti condizioni: (i) ha il maggior valore di $|V_{q,vis}|$, (ii) nel caso in cui vi siano più match in (i), quello che ha il valore maggiore in $|V_{q,neig}|$, (iii) nel caso in cui vi siano più match in (ii), il nodo che ha il valore maggiore in $|V_{q,unv}|$. Nel caso di più candidati in ogni condizione, il nodo è scelto arbitrariamente. Questa definizione può essere espansa con altre condizioni dipendenti dal singolo problema, ma in realtà questa fase non verrà modificata dato che lo scopo della riduzione dell'albero di ricerca è già pienamente implementata anche nel caso di grafi temporali con questa definizione.

3. Un problema che sorge nel calcolo degli isomorfismi in sotto-grafi è che il processo di corrispondenza può produrre occorrenze ridondanti. Al fine di

escludere occorrenze ridondanti durante il processo di ricerca delle soluzioni e potare l'albero dello spazio di ricerca dalle soluzioni ridondanti, RI definisce condizioni di rottura della simmetria in base agli identificatori dei nodi del grafo della query. Le condizioni di rottura delle simmetrie sono collegati ai concetti di automorfismi ed orbite di un grafo query (definiti nella sezione 2.3).

Dati due nodi di query q e q_0 appartenenti alla stessa orbita, con $id(q) < id(q_0)$, una condizione di rottura della simmetria è una disuguaglianza della forma $q \prec q_0$, indicando che il nodo q deve precedere il nodo q_0. In altre parole, una condizione di rottura della simmetria è una condizione che impone un ordine relativo tra due nodi della query appartenenti alla stessa orbita.

4. Ultima e più importante fase è la fase di matching. Seguendo l'ordine definito in precedenza dei nodi del grafo query, RI avvia il processo di matching per trovare le occorrenze della query all'interno del grafo target, utilizzando le condizioni di rottura delle simmetrie per l'eliminazione delle ridondondanze man mano che procede. La corrispondenza viene eseguita costruendo una funzione di mappatura $f : V_Q \to V_T$ (inizialmente non definita per tutti i nodi di query) ed il match M corrispondente, che inizialmente è vuoto. Ogni volta che viene trovata una nuova corrispondenza tra un nodo di query q e un nodo di destinazione t, la coppia (q, t) viene aggiunta a M. Quando tutti i nodi del grafo query sono stati abbinati, M costituisce una nuova corrispondenza di Q in T, quindi può essere aggiunto all'elenco delle corrispondenze trovate.

2.2 Graph matching per grafi temporali

Durante questo studio verrà fatto un matching su grafi temporali a singoli contatti per arco, questo è sia per semplificare la comprensione del problema di base che può essere facilmente espanso, sia per stabilire una base solida da poter far evolvere in base alle proprie esigenze.

La parte teorica resterà comunque generale rispetto ad ogni tipo di grafo temporale, e considerazioni aggiuntive per più contatti o intervalli per arco saranno fate durante le definizioni formali del problema.

Si deve comunque tenere a mente che l'algoritmo implementato considera soltanto grafi con singoli contatti per arco (che è comunque facilmente espandibile dal problema di base per più contatti singoli, non così tanto semplice da implementare per troppi contatti e per intervalli in generale).

Si sono utilizzati inoltre singoli contatti perché la maggior parte dei lavori visti in campo Temporal Graph Matching utilizzavano o singoli contatti, oppure stabilivano condizioni e vincoli molto debilitanti che verranno viste di seguito in alcune definizioni date da alcuni autori[11, 12, 8].

In un altro studio per la ricerca di motif su grafi temporali [11], lo studio si concentra principalmente sul definire un processo di conteggio basato su confronti edge-driven, cioè confrontando gruppi di archi che rispettino una condizione

definita nella definizione di temporal motif, ma questa definizione è molto simile a quella di cammino, e non lascia molto spazio per l'espansione con altri concetti, restando quindi una definizione singola e specifica rispetto al problema del conteggio di motif in un grafo (lo studio inoltre si concentra su motif con 3 nodi -3 archi, dedicando una piccolissima parte al problema generale).

La definizione di motif data in [11] non permette contatti che possono avere gli stessi tempi, quindi precludendo la maggior parte delle reti temporali esistenti che si appoggiano su questa caratteristica. L'algoritmo utilizzato ordina gli archi e svolge un confronto edge-driven, ottimizzato per la ricerca di motif con 3 archi e 3 nodi. Il problema generale è veramente poco definito e non sembra essere neanche abbastanza generale per essere utilizzato in pratica.

Per la definizione di isomorfismi su sottografi, [12] non ha dato una definizione concreta, concentrandosi sulla definizione di sottografo temporale che rispetti la proprietà di time-respecting, ed ha tralasciato la generalizzazione del problema andando direttamente alla soluzione adatta al problema che voleva risolvere (quindi considerando i sottografi nel grafo target che rispettassero la definizione di time-respecting subgraph, dove ogni cammino doveva essere Time-Respecting, quindi non si potevano definire altri tipi di grafi query se non quelli che rispettassero questa proprietà).

La definizione di sub-graph isomorphism su grafi temporali data in [12] si poggia sulla definizione di cammini Time-Respecting, non definendo delle strutture temporali effettive da poter utilizzare per i confronti e le condizioni utilizzabili durante il matching, mantenendo così la definizione abbastanza ambigua e poco generale.

Ci possono essere tre possibilità per la ricerca di isomorfismi su sotto-grafi temporali:

- Time before topology: estrarre i sotto-grafi temporali che rispettano le ipotesi di time-respecting, poi procedere con il matching dei sottografi topologico semplice. Questo approccio è particolarmente pesante come prestazioni, anche perché possono presentarsi ridondanze non trascurabili.

- Topology before time: svolgere il matching direttamente sulla topologia statica del grafo, e poi svolgere delle operazioni sulle componenti temporali dei grafi ottenuti, quindi pruning sui sottografi temporali effettivi che rispettino le ipotesi, questa è la strategia più semplice, anche perché si possono utilizzare direttamente algoritmi di subgraph matching noti, per poi gestire come si vuole i risultati ottenuti.

- Time and topology together: Il controllo temporale ed il matching vengono fatti nello stesso momento e quindi, prendendo come esempio un algoritmo di Ullman semplice, viene fatto un controllo sul matching di ogni nodo e di ogni edge, facendo pruning di soluzioni che non rispettino le ipotesi di time-respecting match.

Di particolare importanza per la definizione di isomorfismi su sotto-grafi temporali è la definizione di grafo query e target nel caso temporale, dato che questa

definizione solitamente non viene data formalmente o dettagliatamente dagli autori, e quindi molto spesso lascia molti dubbi.

In questo studio, viene visto il grafo temporale query come un pattern temporale da ricercare. Questo significa che i tempi dei contatti non sono importanti, ma i tipi di propagazioni che producono con le conseguenti impronte temporali sono al centro della ricerca degli isomorfismi.

Per la definizione sono state scelte solo due classi possibili per la classificazione di una propagazione, però durante l'implementazione reale sono state utilizzate 3 classi (o più durante il confronto degli archi), dipendenti dalla latenza della propagazione (minore di 0, maggiore di 0 ma minore del δ, maggiore del delta).

Durante la ricerca di possibilità utilizzabili per effettuare il matching in modo consistente con il problema, si sono trovati due metodi principali da cui è possibile derivare altre metodologie più articolate:

- Utilizzare una trasformazione del grafo temporale query in un grafo statico, sempre secondo i pattern di trasformazione che sono state definite e verranno approfondite praticamente nel prossimo capitolo, e poi utilizzare questo grafo statico per il submatching sul grafo temporale target.

- svolgere dei confronti e un matching basato su confronti complessi, alternando e parallelizzando confronti e computazione sia sulla struttura statica che temporale di ogni nodo e dell'insieme.

Per implementare la strategia di *topology before time* bisogna semplicemente fare il matching sulla struttura statica del grafo target e poi fare il controllo sulla struttura dinamica, per esempio svolgendo un controllo sui nodi mappati e sui tempi di contatto di ogni nodo, in modo che rispettino le regole di time-respecting path definite.

Per implementare la strategia di *Time and topology together*, il ragionamento dovrebbe essere abbastanza semplice e si concentrerebbe principalmente sulla fase di matching vera e propria (prendendo ad esempio Ullman).

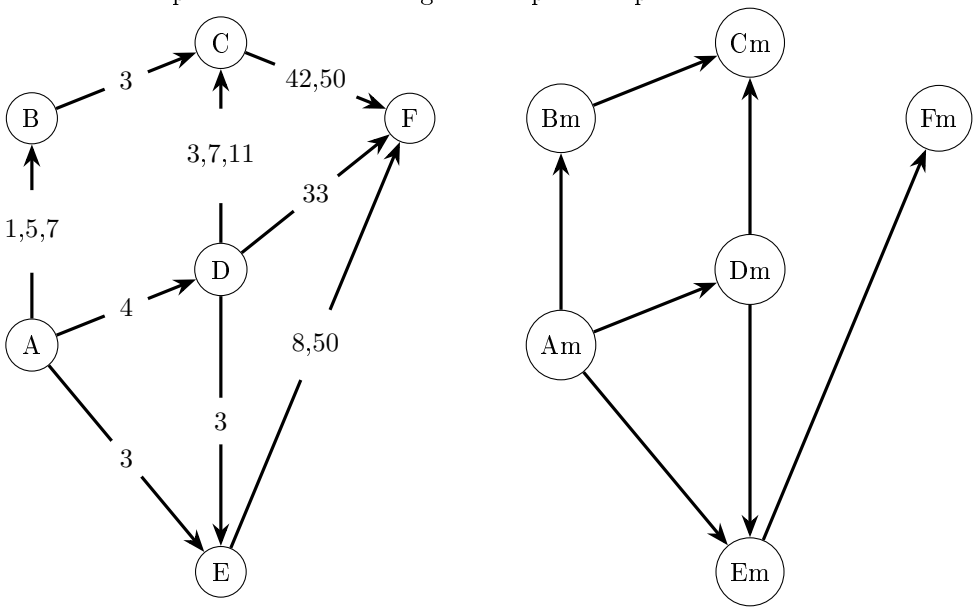
Dopo la definizione del problema e dei punti chiave sul matching di grafi temporali, non dovrebbe essere difficile utilizzare le ipotesi di time-respecting path e contatti per ampliare e implementare regole di look-ahead e rollback.

Una possilbilità di risoluzione per gli algoritmi di subgraph matching di grafi temporali utilizzata da altri autori [11, 8] è la conversione del grafo temporale in una o più forme statiche indotte che permettano il matching statico della rete, in particolare le strategie utilizzate introducono dei metodi molto semplici nella conversione, che contano semplicemente gli archi che rispettino una o più condizioni temporali, di seguito un possibile algoritmo di conversione abbastanza semplice:

Algorithm 1 InducedStaticGraphFromTemporal(G)

```
1: function InducedStaticGraphFromTemporal(G,δ)
2:     edges sorted in sequence, select a node that have the less time of contact
3:     BFS modified with control over temporal attribute and δ, add edge to
   Gtemp if conditions are satisfied
4:     for tedge ∈ E_Gtemp do
5:         SG.addedge(tedge.s,tedge.d)
6:     end for
7:     return SG
8: end function
```

e sotto un esempio di conversione di grafo temporale a più contatti:



Si è scelto un intervallo temporale di attivazione $\delta = 10$. Come si può facilmente notare, si è ottenuto un solo grafo statico, questa non è sempre la situazione, specialmente quando si ha il bisogno di tenere conto di tutti i pattern di attivazione che avvengono nel grafo temporale, quindi la soluzione presentata è abbastanza temporanea e non porterebbe risultati concreti (il problema resta comunque il caso di archi con più contatti).

Alla fine viene comunque utilizzato un approccio ibrido di *Time and Topology Together*, dove il controllo temporale è fatto dinamicamente insieme al controllo di struttura topologica.

Viene utilizzato un algoritmo che non si serve di regole di look-ahead troppo complicate che andrebbero a confrontare una quantità di strutture temporali non indifferente che è RI esposto precedentemente modificato per il controllo della similarità tra nodi e delle strutture temporali per nodo durante varie fasi.

È possibile comunque espandere altri concetti, come per esempio le fasi di preprocessing di RI di calcolo dei compatibility domains e dell'ordine dei nodi del grafo query per la fase di matching [9, 1], sempre utilizzando le definizioni definite precedentemente.

Per esempio, per il calcolo dei compatibility domains, si può fare comunque un controllo sul grado del nodo, e si può alternare ad un controllo sui tempi di contatto per ogni arco uscente dal nodo, o direttamente un controllo dell'impronta del nodo da analizzare.

Infatti è questa la soluzione che è stata adottata, in aggiunta a vari controlli sul grado e sulle simmetrie, vi è stato aggiunto un controllo per la compatibilità di un nodo del grafo temporale target con uno del grafo query, sfruttando le impronte dei due e vedendo se $v_{query} \simeq v_{target}$, e quindi procedendo all'aggiunta del nodo tra i nodi candidati per il mapping.

Per il calcolo dell'ordinamento dei nodi del grafo query, si potrebbe fare un ordinamento locale (rispetto al grado dei nodi) ed in seguito un ordinamento sul numero di contatti per arco (potrebbero sorgere dei problemi che verranno visti nel prossimo capitolo).

Nella implementazione effettiva il calcolo dell'ordinamento non è stato modificato, anche perché potrebbe portare a comportamenti particolari e poco prevedibili (nodo con molti contatti entranti e pochi uscenti), quindi si è voluto lasciare inalterato l'algoritmo di base basato sul grado di un nodo.

Una parte importantissima che è stata cambiata è la fase del controllo degli archi di un possibile nodo da mappare, a cui è stato aggiunto un controllo aggiuntivo su alcune caratteristiche derivate dall'arco (molto simile al calcolo della struttura temporale, ma in realtà più incentrato alla struttura dei singoli archi rispetto a quello preso in input, che dovrebbe essere l'arco da mappare, che deve quindi dare lo stesso tipo di struttura temporale alternativa derivata), la funzionalità da cambiare per il controllo degli archi (*edgeCheck*) svolge la seguente funzione:

Sia $t \in G_T$ e $q \in G_Q$ e si voglia sapere se sia possibile mappare t con q, associandolo al match parziale. Sia $M = (t1, q1), (t2, q2), \ldots, (tk, qk)$ il mapping parziale ottenuto fino al momento del controllo del possibile mapping tra il nodo q ed il candidato t.

La funzione edgeCheck controlla che nel caso in cui q sia collegato al nodo q_i in M, allora anche t deve essere collegato con il nodo t_i già contenuto nel mapping parziale e mappato con il nodo del grafo query q_i. Se questo controllo è vero per tutti i q_i già mappati, allora edgeCheck ritorna true, che significa che il nodo t può essere mappato al nodo q.

Quindi, se si vuole aggiungere la componente temporale durante il controllo, basta prendere l'arco e controllare che il nodo collegato sia simile al nodo nel mapping (quindi confrontando le impronte oppure altre informazioni temporali dipendenti dall'arco).

Le nuove informazioni temporali da confrontare vengono date dall'arco da mappare, che ha un singolo tempo, e definisce la struttura completa di tutte le informazioni possibili per vedere se l'arco si può effettivamente mappare al conseguente candidato (cioè i nodi si possono mappare).

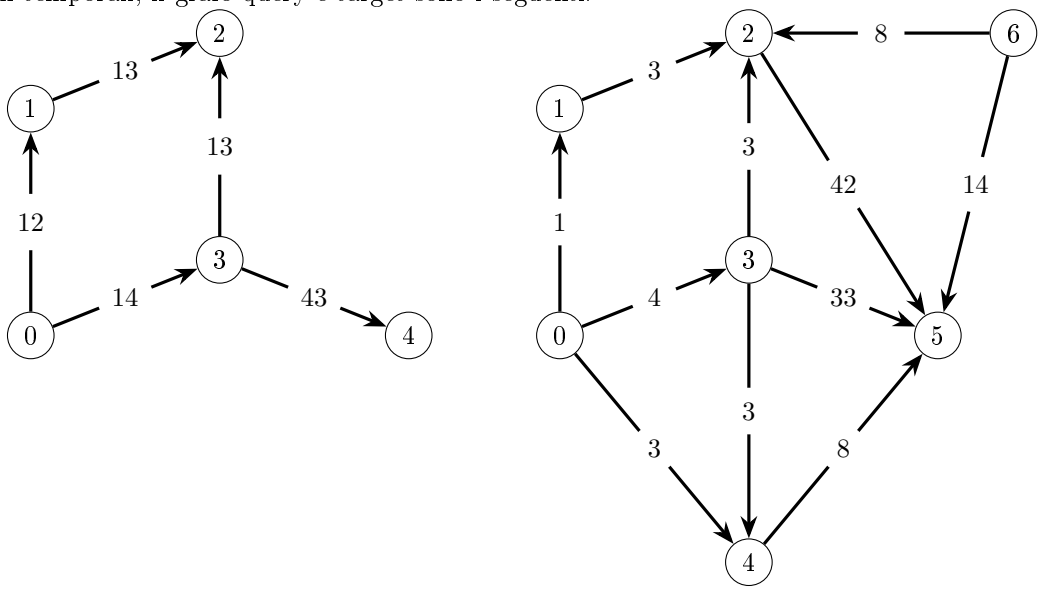
Non è stato utilizzato un controllo di impronta diretto perchè non stabilisce una struttura ben definita temporalmente nel caso in cui vi siano solo nodi entranti e solo nodi uscenti.

Dunque nel controllo sull'arco (q,qi) bisogna verificare che:

- (t,ti) è un arco del target
- I vincoli temporali per (t,ti) sono soddisfatti.

Le implementazioni di tutto ciò che è stato descritto e di questo controllo in particolare sono disponibili online (il codice è disponibile sia in Java che in Scala nel mio github `https://github.com/josura/university-sad/tree/master/tesi/realtesi`).

Esempio 37. In questo esempio viene vista una istanza di isomorfismi su sottografi temporali, il grafo query e target sono i seguenti:



I domini di compatibilità dei nodi della query sono i seguenti:
$Dom(0) = \{0, 6\}$
$Dom(1) = \{1, 4\}$
$Dom(2) = \{2, 4, 5\}$
$Dom(3) = \{3, 5\}$
$Dom(4) = \{1, 2, 3, 4, 5\}$

Si è scelto un $\delta = 10$ e sono state definite le classi di appartenenza di una propagazione $notTimeRespecting, \delta - TimeRespecting, \delta - notTimeRespecting$

Le impronte temporali di ogni nodo del grafo temporale query sono le seguenti:
$impronta_0 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
$impronta_1 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 1), (\delta{-}notTimeRespecting, 0)\}$
$impronta_2 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$

$impronta_3 = \{(notTimeRespecting, 1), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 1)\}$
$impronta_4 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$

Invece le impronte temporali di ogni nodo del grafo temporale target sono le seguenti:

$impronta_0 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
$impronta_1 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 1), (\delta{-}notTimeRespecting, 0)\}$
$impronta_2 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 3)\}$
$impronta_3 = \{(notTimeRespecting, 1), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 2)\}$
$impronta_4 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 2), (\delta{-}notTimeRespecting, 0)\}$
$impronta_5 = \{(notTimeRespecting, 0), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 0)\}$
$impronta_6 = \{(notTimeRespecting, 1), (\delta{-}TimeRespecting, 0), (\delta{-}notTimeRespecting, 1)\}$

Durante la fase di matching si passa dai possibili candidati, si controllano gli archi e le impronte, e si continua la mappatura fino a quando si sono trovate tutte le occorrenze.

Alla fine il numero di occorrenze del grafo query nel grafo target in questo caso è 1.

2.3 Definizioni

Definizione 38. (Sub Graph matching su grafi statici). Un grafo G1 è isomorfo ad un sotto-grafo di un grafo G2 se e solo se esiste una corrispondenza uno a uno tra i nodi del sotto-grafo di G2 e G1 che preserva l'adiacenza, più formalmente $\exists f : V_{G1} \rightarrow V_{G2} | (s, d) \in E_{G1} \implies (f(s), f(d)) \in E_{G2}$

Definizione 39. (Sub Graph matching su grafi temporali a contatti). Un grafo Temporale G1 è isomorfo ad un sotto-grafo di un grafo Temporale G2 se e solo se esiste una corrispondenza uno a uno tra i nodi del sotto-grafo di G2 e G1 che preserva l'adiacenza, più formalmente $\exists f : V_{G1} \rightarrow V_{G2} | (s, d) \in E_{G1} \implies (f(s), f(d)) \in E_{G2}$ e se il nodo target da mappare ha la stessa struttura temporale del nodo query, formalmente $s \equiv f(s)$

Per la definizione è stata sfruttata la relazione di equivalenza di struttura temporale perché il sotto-grafo non ha tutti gli archi compresi tra i nodi del sotto-grafo, in seguito sarà data una definizione con la similarità che toglie questo limite, ma che deve essere comunque utilizzata solo per la definizione di pruning e regole di fattibilità, dato che la similarità con il nodo del grafo target non garantisce l'equivalenza temporale tra il nodo del grafo query ed il nodo del sotto-grafo target.

Definizione 40. (isomorfismo tra grafi temporali tramite equivalenza e similarità). Un grafo temporale GT1 è isomorfo ad un sotto-grafo temporale di un grafo Temporale GT2 se e solo se esiste una corrispondenza uno a uno tra i nodi del sotto-grafo temporale di GT2 e GT1 che preserva l'adiacenza ed il sotto-grafo risultante (come sottoinsieme dei nodi e degli archi del grafo target) abbia tutti i nodi mappati con la stessa struttura temporale di quelli del grafo query a cui vengono mappati, cioè i nodi del grafo query sono simili ai nodi del grafo target mappato.

Definendo l'isomorfismo di sotto-grafi tramite similarità è possibile svolgere un matching senza troppi problemi semplicemente confrontando le impronte dei singoli nodi che sono mappati.

Bisogna comunque stare attenti ad effettuare i controlli sulla sottostruttura e sulla equivalenza temporale tra i nodi del sotto-grafo (formato da un sottoinsieme di nodi ed un sottoinsieme di contatti) ed il grafo query, come già accennato in precedenza.

Definizione 41. (automorfismo di un grafo temporale). Dato un grafo temporale $G_t = (V, E)$, un automorfismo di G_t è una permutazione ϱ dell'insieme dei nodi di V che rispetta le seguenti condizioni:

- $\forall u, v \in V : (u, v, t) \in E \iff (\rho(u), \rho(v), t') \in E$
- $\forall u, v \in V : u \equiv v \iff \rho(u) \equiv \rho(v)$

In altre parole, un automorfismo è un riarrangiamento dell'insieme di nodi che conserva la struttura di un grafo. Il risultato dell'applicazione di un automorfismo è un nuovo grafo $G_0 = (V, E)$, ottenuto da G permutando gli identificatori dei suoi nodi in base a ϱ. Si dice che G_0 sia automorfico per G.

Definizione 42. (Matrice degli automorfismi). Dato un grafo $G = (V, E)$ with k nodes $v_1, v_2, ..., v_k$ e h automorfismi $\rho_1, \rho_2, ..., \rho_h$. La matrice degli automorfismi A, è una matrice contenente h righe e k colonne, dove $A[i, j] = \rho_i(v_j)$.

La definizione è la stessa di grafo statico, dato che la componente temporale non inficia direttamente questa definizione.

Definizione 43. (Orbite di un grafo). Data una matrice degli automorfismi A, per una certa colonna della matrice, $A[, j]$, Il suo set di nodi è un orbita di G. Si indica con $Orb(u)$ l'orbita a cui appartiene il nodo u.

Questa definizione non cambia per grafi temporali.

Le definizioni di automorfismo e orbita verranno utilizzate per la ricerca di simmetrie che potrebbero dare luogo a ridondanze durante la fase di matching.

2.4 Problemi con gli isomorfismi e la definizione di grafo query e target

Come già brevemente descritto precedentemente durante la definizione del problema di isomorfismi su sotto-grafi temporali, sorgono problemi non indifferenti nell'interpretazione del grafo query, che è stato definito come il pattern temporale da trovare nel grafo target.

Cosa trovare nel caso di più contatti per arco? Come fare il matching e come adattarlo in base alle esigenze? Come adattare il modello nel caso di intervalli?

Tutte queste ed altre domande hanno risposte dipendenti dal caso di studio che si vuole portare, quindi bisogna adattare il modello di base sulla forma del sistema che si sta cercando di catturare o simulare.

Per esempio l'implementazione adottata durante questo studio della ricerca degli isomorfismi su grafi temporali trova grafi che abbiano nodi con la stessa struttura temporale o che siano simili, inoltre vengono aggiunte classi aggiuntive per la classificazione delle propagazioni che non rispettano le condizioni e che sono maggiori o minori del tempo di contatto iniziale.

Questa caratteristica può essere fraintendibile e poco consona al significato di classificazione di propagazione dato precedentemente che classifica soltanto in due categorie, ma in realtà il sistema è stato progettato così proprio per fare vedere come l'implementazione non è sempre la stessa in base alle condizioni basilari stabilite, e che la maggior parte delle volte il framework per il matching si deve adattare per ricoprire bene le esigenze del singolo problema, come già detto molte volte.

Nel caso di soluzione di matching basata su *Time and topology together* , la modifica dell'algoritmo di matching potrebbe portare all'inutilità della fase di pre-processing, in particolare l'ordinamento dei nodi della query (perchè in alcuni casi ci sarebbe l'obbligo di iniziare dal nodo che ha l'edge con il tempo minore), anche se questo rischio è discutibile e bisognerebbe provare una implementazione vera e propria.

Dato che non vi è una definizione univoca e standard del problema di isomorfismi su grafi (e di tecniche utilizzabili su grafi temporali in generale), molto spesso gli autori non definiscono un framework di base da utilizzare per la definizione del problema, ma costruiscono una soluzione specifica del singolo problema secondo definizioni abbastanza strette e che lasciano poca possibilità di espansione.

Per la ricerca di pattern temporali il framework definito precedentemente potrebbe non essere adatto, dato che trova sotto-grafi con una specifica struttura temporale, quindi bisogna sempre adattare la sotto-struttura al caso unico.

Nell'implementazione bisogna ricordare che la similarità è utile per fare il pruning dei possibili candidati, ma per il calcolo dell'isomorfismo su sotto-grafo, bisogna considerare la struttura temporale del solo sotto-grafo (senza archi o nodi aggiuntivi), quindi effettuare controlli aggiuntivi durante la fase di matching (nel controllo dell'adiacenza) o alla fine del matching, che possono rallentare di molto la computazione.

Nella prossima sezione verranno visti gli algoritmi e le considerazioni sull'implementazione per la ricerca di isomorfismi su sotto-grafi temporali.

3 Algoritmi e implementazione

Di seguito gli algoritmi utilizzati per il calcolo delle funzioni di base che serviranno per il graph matching:

Algorithm 2 classifyPropagation(prop,δ)

Input: *prop*: propagation to classify,
δ: conditional parameter used to classify propagations
Output: the type of the propagation *prop*

```
1: function classifyPropagation(prop,δ)
2:     if 0 <prop.Δ < δ then
3:         return timeRespecting
4:     else
5:         return notTimeRespecting
6:     end if
7: end function
```

Questa funzione è particolarmente importante ai fini del calcolo delle impronte, in quanto classifica le singole propagazioni per farle rientrare nella classe a cui dovrebbero appartenere.

In questo caso, come definito precedentemente, vengono utilizzate solo due classi per facilitare la comprensione, ma nel codice le classi definite sono tre.

Sostanzialmente per ogni propagazione passata a riga di comando vengono controllate le condizioni stabilite dalle possibili classi che si possono ottenere, quindi viene ritornato il tipo di classe di appartenenza. Per espandere questa funzione con altri casi e classi ben definite, basta aggiungere le condizioni ed i condizionali aggiuntivi, implementandoli tramite uno switch case, oppure trattando la propagazione come un oggetto ed assegnando uno stato (la classe di appartenenza) ogni volta che una propagazione venga creata.

Questa funzione è molto semplice e poteva essere direttamente sostituita alle parti di codice che la richiedono, ma si è lasciata indipendente data la sua natura dipendente dalle classi definibili e per refattorizzazione dato che viene utilizzata anche in altri punti del codice (che non vengono visti dato che non vengono utilizzati per il matching effettivo).

L'ultima possibilità non è stata implementata dato che non si sono utilizzate le propagazioni effettive ma coppie di archi, e tenere delle strutture aggiuntive comportava spazio extra relativamente ridondante , e dato che la classificazione non svolge calcoli complessi ma delle semplici decisioni si è lasciata questa scelta che non aggiunge quasi nulla alla complessità effettiva del problema.

Notare che la funzione lavora per contatti, ma non fa distinzione tra più contatti di un singolo arco o di archi diversi, quindi è indipendente dal grafo temporale a contatti utilizzato.

Il caso di grafo ad intervalli è leggermente differente, ma i principi sono comunque quelli descritti precedentemente durante la definizione del problema.

Nel caso di grafo temporale ad intervalli l'algoritmo è il seguente:

Algorithm 3 classifyPropagationInterval(prop,δ)

Input: *prop*: propagation to classify,
δ: conditional parameter used to classify propagations
Output: the type of the propagation *prop*

```
1: function classifyPropagationInterval(prop,δ)
2:     if prop.interval ≠ ∅ then
3:         return timeRespecting
4:     else
5:         return notTimeRespecting
6:     end if
7: end function
```

Notare come sia possibile fare un confronto semplice con l'insieme vuoto per vedere se la propagazione sia effettivamente Time-Respecting o no, questo solo grazie alla definizione di *interval* nella propagazione, che ingloba all'interno l'intersezione ricorsiva con le propagazioni differenti (che comunque resta ambigua ma questo fattore verrà tralasciato dato che questo studio si concentra su un implementazione a contatti).

Di seguito la funzione che utilizza la classificazione di una propagazione (per grafo a contatti) per il calcolo dell'impronta temporale di un nodo:

Algorithm 4 temporalFingerprint(G_t,node,δ)

Input: G_t: Temporal Graph of the node,
node: is the node to test,
δ: is the conditional parameter used to classify propagations
Output: the Temporal fingerprint of the node in the Temporal Graph

```
1: function temporalFingerprint(G_t,node,δ)
2:     fingerprint ← vector[2]
3:     for prop ∈ G_t.P_node do
4:         if classifyPropagation(prop,δ) = timeRespecing then
5:             fingerprint[1] ← fingerprint[1] +1
6:         else
7:             fingerprint[2] ← fingerprint[2] +1
8:         end if
9:     end for
10:    return fingerprint
11: end function
```

Il codice è sostanzialmente lo stesso sia nel caso di grafo a contatti, sia nel caso di intervalli di propagazione.

Ambiguità potrebbe risalire però nel caso di grafo con più contatti per arco, caso già discusso precedentemente, in cui esistono due possibilità: o considerare le propagazioni come indipendenti, oppure propagazioni con stesso nodo

destinazione e nodo sorgente andranno a finire in impronte differenti, quindi combinazioni tra le propagazioni uguali e quelle differenti.

Nel primo caso, potrebbero venire fuori delle ambiguità dato che una singola impronta temporale che ingloba informazioni da diversi pattern per gli stessi archi potrebbe portare a falsi positivi, cioè nodi cui archi sono maggiori rispetto a quelli del nodo query, ma aventi stessa impronta finale.

Il secondo caso dovrebbe essere quello più sicuro, ma anche il più complesso algoritmicamente, dato che bisognerebbe fare le combinazioni tra i vari archi in base ai contatti di ogni arco (restituendo un insieme di impronte temporali di tutte le combinazioni), quindi confrontare tutte le permutazioni possibili e vedere se sono simili.

Questo caso non verrà trattato dato che lo studio si concentra su singoli contatti, ma in futuro si vedranno le possibili implicazioni ed utilizzi dei grafi temporali ad intervalli multipli, così da avere più informazioni sulle possibili esigenze più comuni che si possono soddisfare con le possibili implementazioni e definizioni.

Per fare il controllo di due strutture temporali e verificare la similarità basta fare un operazione vettoriale sulle componenti dell'impronta del nodo della query q e del nodo target t , controllando quindi che sia

$$result = temporalFingerprint(q) \overset{component-wise}{\leq} temporalFingerprint(t)| \quad \forall i \in [1, result.size], result[i] = true$$

altrimenti la condizione non è verificata ed il nodo non è simile

Di seguito l'algoritmo che viene chiamato per effettuare il matching tra grafi temporali:

Algorithm 5 TemporalRI(G_{query},G_{target},δ)

Input: G_{query}: query Temporal Graph,
G_{target}: target Temporal Graph,
δ: is the conditional parameter used to classify propagations
Output: number of occurrences of Query in Target

```
1: function TemporalRI(G_query,G_target,δ)
2:     Dom ← ComputeDomains(G_query, G_target,δ)
3:     μ ← OrderQueryNodes(G_query, Dom)
4:     ς ← ComputeSimmetryBreakingCond(G_query,δ)
5:     Matches ← SubgraphMatching(G_query, G_target, Dom, μ, ς, δ)
6:     return Matches
7: end function
```

Non vengono visti nello specifico tutti gli algoritmi utilizzati (lo pseudocodice è disponibile nell'Appendice di[9]) dato che le modifiche sono minimali o nulle per molte parti del codice, le uniche parti in cui è effettivamente cambiato qualcosa sono i codici del calcolo dei domini di compatibilità, il calcolo degli automorfismi durante la ricerca delle simmetrie ed una parte del codice durante

la fase di matching in cui si controllano gli archi tra i nodi già mappati ed un candidato (controllando la similarità).

Quest'ultimo caso non verrà riportato come pseudocodice perché la modifica è minima e riguarda un controllo della struttura temporale a partire dall'arco che si sta mappando rispetto al candidato ed al nodo già mappato (vi è comunque il solito controllo della similarità delle impronte temporali, l'algoritmo è già stato descritto).

Sotto viene invece riportato il codice per il calcolo delle rotture di simmetria:

Algorithm 6 ComputeSimmetryBreakingCond(G_{query},δ)

Input: G_{query}: query Temporal Graph,
δ: is the conditional parameter used to classify propagations
Output: set of symmetry breaking condition of query graph

1: **function** ComputeSimmetryBreakingCond(G_{query},δ)
2: $\varsigma \leftarrow \emptyset$
3: $Aut_{|\varsigma} \leftarrow$ ComputeAutomorphismMatrix(G_{query},δ)
4: $Orb_{|\varsigma} \leftarrow$ ComputeOrbits($Aut_{|\varsigma}$)
5: **while** $|Aut_{|\varsigma}| > 1$ **do**
6: $q' \leftarrow argmin_{q \in V_Q : |Orb_{|\varsigma}| > 1} \{id(q)\}$
7: **for** $q \in V_Q | q \neq q' \cap Orb_{|\varsigma}(q) = Orb_{|\varsigma}(q')$ **do**
8: $\varsigma \leftarrow \varsigma \cup \{q' \prec q\}$
9: **end for**
10: $Aut_{|\varsigma} \leftarrow \left\{ Aut_{|\varsigma}[i] | \rho_i(q') = q' \right\}$
11: $Orb_{|\varsigma} \leftarrow$ ComputeOrbits($Aut_{|\varsigma}$)
12: **end while**
13: **return** ς
14: **end function**

Una ottimizzazione possibile sia per singoli contatti che per più contatti o più intervalli per arco è quella di effettuare un binary search sui tempi di contatto per velocizzare il controllo della similarità e del rank.

L'algoritmo per il calcolo delle condizioni di rottura delle simmetrie si basa su un calcolo iterativo degli automorfismi della query e delle orbite a partire dall'attuale insieme di condizioni di rottura della simmetria scoperte dall'algoritmo. Dato un insieme di condizioni di rottura C, indichiamo con $Aut_{|\varsigma}$ e $Orb_{|\varsigma}$ la matrice dell'automorfismo e l'insieme delle orbite che rispettano le condizioni di rottura in C.

Quando C è vuoto, $Aut_{|\varsigma}$ corrisponde alla matrice di automorfismo di Q e $Orb_{|\varsigma}$ è l'insieme di orbite corrispondente. Una condizione di rottura $q_0 \prec q$ in C, impedisce che q_0 venga mappato a q in qualsiasi automorfismo. Quindi, $Aut_{|\varsigma}$ diventa la matrice degli automorfismi ottenuta dall'insieme di tutti gli automorfismi del grafo query dove q_0 non è mappato a q, per tutti i nodi q_0 e q tale che $q \prec q_0 \in C$. $Orb_{|\varsigma}$ è l'insieme delle orbite relativo ad $Aut_{|\varsigma}$. Il primo passo dell'algoritmo è il calcolo della matrice degli automorfismi di Q.

Ciò può essere ottenuto utilizzando qualsiasi algoritmo di matching del grafo. In seguito, vengono calcolate le orbite di Q e viene calcolato il nodo del grafo query q_0 con id minimo su tutte le orbite con almeno due nodi equivalenti. Per ogni nodo $q \neq q_0$ appartenente alla stessa orbita di q_0, viene definita una nuova condizione di rottura della simmetria $q_0 \prec q$ che viene aggiunta all'insieme finale di condizioni. Questo passaggio equivale a impedire che il nodo q_0 venga mappato a qualsiasi altro nodo e a mettere q_0 in un'orbita separata. Per essere coerenti con ciò, dobbiamo mantenere dalla matrice degli automorfismi corrente solo le righe corrispondenti agli automorfismi che mappano q_0 a se stesso e aggiornare il relativo set di orbite $Orb_{|\varsigma}$.

Di seguito viene riportato lo pseudocodice per il calcolo dei domini di compatibilità:

Algorithm 7 ComputeDomains(Q,T,δ)

Input: Q: query Temporal Graph,
T: target temporal graph
δ: is the conditional parameter used to classify propagations and for the similarity
Output: set of compatibility domains of nodes in G_{query}

```
1: function ComputeDomains(Q,T,δ)
2:     for q ∈ V_Q do
3:         Dom(q) ← ∅
4:     end for
5:     for t ∈ V_T do
6:         for q ∈ V_Q do
7:             if deg(q) ≤ deg(t) ∧ q ≃ t then
8:                 Dom(q) ← Dom(q) ∪ {t}
9:             end if
10:        end for
11:    end for
12:    for q' ∈ V_Q do
13:        for t' ∈ Domq' do
14:            for q'' ∈ V_Q|(q', q") ∈ E_Q do
15:                if ∄t'' ∈ Dom(q'')|(t', t") ∈ E_T then
16:                    Dom(q') ← Dom(q')\ {t'}
17:                end if
18:            end for
19:        end for
20:    end for
21:    return Dom
22: end function
```

Come si può facilmente vedere, un nodo del grafo target entra nel dominio dei candidati di un nodo del grafo query solo e soltanto se ha lo stesso grado e se il nodo query è simile al nodo target.

Viene inoltre visto il requisito minimo già descritto della Arc Consistency,

viene quindi visto se tra due nodi del grafo query esiste un arco, allora per ogni nodo del dominio della sorgente si controlla se esiste almeno un arco verso i nodi nel dominio della destinazione.

Il parametro δ sembra non venire utilizzato, ma in realtà viene utilizzato durante il calcolo della similarità tra i due nodi (per la computazione delle impronte).

Di seguito viene descritto brevemente l'algoritmo di Match modificato per integrare la componente temporale.

Il nucleo dell'algoritmo di matching è la procedura MATCH ricorsiva. Sia M la corrispondenza parziale trovata e q un nodo del grafo temporale query che non è stato ancora mappato. Se q_0 è il nodo che precede q nell'ordine μ, l'insieme $Cand(q)$ di nodi target candidati da abbinare a q è dato da $Neigh(f(q_0)) \bigcap Dom(q)$, cioè l'insieme di nodi che sono adiacenti del nodo destinazione che è già stato abbinato a q_0 e sono nel dominio di compatibilità di q.

Se q è il primo nodo in μ, l'insieme di nodi candidati è solo il dominio di compatibilità di q. Se alcune regole di fattibilità sono soddisfatte (tra cui la similarità tra i nodi e il controllo aggiuntivo sugli archi associati), RI aggiunge la coppia (q, t) alla corrispondenza parziale M e aggiorna la funzione di mappatura f. Quando tutti i nodi della query sono stati mappati, viene trovata una nuova occorrenza di Q in T e la corrispondenza corrispondente viene aggiunta all'elenco delle corrispondenze trovate. Ogni volta che tutti i nodi del grafo temporale query sono stati mappati o non è stata trovata alcuna corrispondenza per un nodo del grafo query, l'algoritmo esegue il backtracking e continua la ricerca dall'ultimo nodo abbinato (precedentemente). Il backtracking implica la rimozione dell'ultima coppia di query corrispondenti e dei nodi di destinazione da M e la mappatura tra tali nodi utilizzando f. Quando non è possibile creare altre corrispondenze per alcun nodo della query, TemporalRI si interrompe. Alla fine del processo di corrispondenza, l'algoritmo restituisce l'elenco di tutte le corrispondenze trovate.

Inoltre, dato che l'algoritmo controlla la similarità tra nodi utilizzando il grafo target completo, bisogna controllare alla fine che il grafo query ed il sottografo (senza archi aggiuntivi rispetto a quelli necessari), siano isomorfici, in particolare ogni nodo abbia la stessa struttura temporale (e che non sia più solo similare).

Alla fine si avranno tutte le corrispondenze di sottografi isomorfi al grafo query.

3.1 Analisi complessità

Vengono analizzate le complessità spaziale e temporale, iniziando dall'analisi degli algoritmi ed esplorando alternative sia come cambiamento di codice e funzionalità, sia come utilizzo differente di strutture aggiuntive, per esempio aggiungendo dei parametri alla definizione di grafo temporale.

Sia n_Q il numero di nodi del grafo temporale query, ed n_T il numero di nodi della destinazione. Il primo passo di TemporalRI è il calcolo dei domini di

compatibilità. La costruzione di domini richiede $O(n_Q n_T)$, mentre la procedura AC richiede $O(n_Q^2 n_T^2)$ perché, nel peggiore dei casi, per ogni arco in uscita della query da q_0 bisogna controllare tutti gli archi in uscita da nodi di destinazione in $Dom(q'')$. Quindi, il calcolo dei domini di compatibilità richiede sommariamente $O(n_Q^2 n_T^2)$. Quindi, TemporalRI calcola l'ordinamento dei nodi del grafo query per il processo di corrispondenza. La parte importante di questo passaggio è la costruzione degli insiemi $V_{q,vis}$,$V_{q,neig}$ e $V_{q,unv}$, che richiede $O(n_Q^2)$.

La terza fase di TemporalRI è il calcolo delle condizioni di rottura della simmetria. Il calcolo della matrice di automorfismo per la query è difficile almeno quanto la risoluzione dell'isomorfismo del grafo e la sua complessità è limitata da $O(2^{n_Q})$. Le orbite vengono calcolate scansionando colonna per colonna la matrice dell'automorfismo. Poiché il numero di automorfismi di un grafo con n nodi è al massimo $n!$ (nel caso in cui il grafico sia completo), il calcolo delle orbite richiede $O(n_Q! n_Q)$. Nel peggiore dei casi ad ogni passo del ciclo while il numero di automorfismi diminuisce solo di uno e il ciclo viene eseguito $n_Q!$ volte. Quindi, il calcolo delle condizioni di rottura richiede $O(n_Q! 2n_Q)$.

Il processo di matching non verrà commentato nello specifico, ma basta sapere che il calcolo delle similarità tra nodi non aggiunge nulla nell'analisi asintotica, che resta comunque la stessa rispetto a quella già descritta in [1, 9], inoltre se si tengono le strutture temporali dei singoli nodi direttamente come struttura aggiornata, il controllo diventa un semplice confronto.

Per quanto riguarda la complessità spaziale di TemporalRI, data la possibilità di avere un contatto per ogni arco, il grafo temporale query e target richiedono $O((2n_Q)^2)$ e $O((2n_T)^2)$ spazio nel peggiore dei casi, rispettivamente, quindi restano sempre nell'ordine del quadratico di RI. Le strutture dati aggiuntive utilizzate dall'algoritmo durante la ricerca includono l'insieme di domini di compatibilità per ogni nodo di query (spazio $O(n_Q n_T)$ nel caso peggiore), l'insieme di condizioni di rottura della simmetria per la query $O(n_Q^2)$ spazio), la mappatura e la corrispondenza parziale (entrambe richiedono uno spazio $O(n_Q)$) e l'insieme di nodi di destinazione candidati per la corrispondenza per ciascun nodo di query $O(n_Q n_T)$. Pertanto, la complessità spaziale totale di TemporalRI è $O((2n_Q)^2) + O((2n_T)^2) + 2 * O(n_Q)) + 2 * O(n_Q n_T) + O(n_Q^2)$. Dato che molto spesso il grafo query è molto più piccolo del grafo target ($n_Q << n_T$), la complessità risultante asintoticamente sarà dominata da $O(n_T^2)$, cioè lo spazio necessario per memorizzare il grafo temporale target

Possono essere aggiunte comunque delle strutture o variabili aggiuntive come per esempio l'insieme delle propagazioni che aggiungerebbe una complessità spaziale aggiuntiva di $(O(|inedges(Q)| * |outedges(Q)|) + O(|inedges(T)| * |outedges(T)|))$ che non conviene assolutamente tenere data la sua situazionalità e facile derivabilità, oppure l'impronta (o l'insieme di impronte nel caso di più contatti per arco) del singolo nodo, che viene aggiornata ad ogni aggiunta di arco entrante o uscente da un nodo $(O(n_T) + O(n_Q))$, e grazie al quale è possibile evitare il calcolo continuo delle impronte per effettuare un controllo diretto sulle strutture ausiliarie.

4 Conclusioni

Lo sviluppo delle reti temporali è particolarmente difficile perché o ci mancano le misure di base e le nozioni necessarie a modellare reti reali, o semplifichiamo troppo l'analisi stessa.

Nel momento in cui si saranno stabilite delle convenzioni comuni che siano di notevole importanza per abbastanza reti reali e non perdano rilevanza con il cambiare della struttura di base (in poche parole siano indipendenti dal caso e comuni a tutti i problemi che incorporano la componente temporale) si potrà riuscire a simulare la realtà in modo più accurato. Il problema finale resta comunque quello di minimizzare l'entropia e massimizzare l'informazione.

Come si è visto più volte, molto spesso il problema delle reti temporali è quello di adattare l'infrastruttura al caso singolo, in base alle esigenze.

Molto spesso questo viene sfruttato da altri autori per creare le proprie soluzioni al problema, quindi creando delle soluzioni molto selettive e poco generali, senza definire una vera e propria infrastruttura solida, ma confondendo molto spesso le idee con definizioni troppo semplici per descrivere a pieno la componente temporale.

Anche grazie a questa tendenza, molti autori finiscono per prendere ad esempio le singole soluzioni specifiche per il singolo problema in modo superficiale, adattando la soluzione vista e spiegata in altri studi per il proprio caso, riuscendo così a definire un infrastruttura abbastanza confusa rispetto all'obiettivo finale a cui si voleva arrivare.

Alcuni autori invece si sono concentrati su una definizione di framework generale ed utilizzabile in molti contesti, senza essere pesantemente influenzati da bias verso i singoli casi, ma prendendo le descrizioni e soluzioni adottate da altri studi ed integrando una spiegazione generale capace di permettere la costruzione di un framework generale ed adattabile al singolo caso [2].

In questo studio si è deciso di adottare lo stesso approccio, non concentrandosi su un singolo problema, ma definendo prima una struttura di base flessibile, e poi adattarla per la definizione del problema (che in questo caso era il Temporal Sub-Graph Isomorphism), in modo da portare un esempio reale di modellazione di una soluzione a partire da definizioni chiare e concretamente utili nella maggior parte dei casi.

Dopo la definizione del framework di base, si è definito il problema del Temporal Sub-Graph Isomorphism introducendolo prima con una descrizione del problema di base nei grafi statici con una possibile soluzione che utilizza un algoritmo veloce basato su euristiche utili per la riduzione dello spazio di ricerca (**RI**), ed espandendo il concetto ai grafi temporali in modo da avere una base solida espandibile in modo semplice grazie anche alle definizioni create durante la descrizione della struttura di base e delle operazioni nelle reti temporali.

Si sono visti dei problemi di definizione nella descrizione di grafo temporale query e target, dato che la ricerca della struttura è totalmente dipendente dal problema che si sta cercando di risolvere, cioè il tipo di struttura che si sta cercando.

Si è effettuata una analisi della complessità che ha fatto vedere come effettivamente il problema di base non sia di semplice soluzione ed allo stesso tempo molto dipendente dalle variabili che vengono passate (cioè grafo query e target), e si vedrà in dati sperimentali come questa affermazione sia sostanzialmente vera dato che i tempi di esecuzione dipendono molto spesso dalla struttura di base dei grafi temporali utilizzati come variabili.

Una considerazione aggiuntiva deve essere fatta sull'analisi delle performance che si vedranno di seguito e che sono molto importanti in molti studi. In particolare bisogna fare particolare attenzione all'implementazione stessa ed i metodi utilizzati.

Quasi tutti gli algoritmi utilizzano tecnonogie studiate per essere implementate su un singolo core, solo alcuni utilizzano un multithreading che sfrutta una parallelizzazione delle istruzioni, quindi aumentando le prestazioni globali dell'algoritmo, ma nessuno utilizza metodi appositi effettivamente pensati per essere normalmente traslati in una computazione mirata alla parallelizzazione.

Nessuno degli autori menziona tecnologie di computazione distribuita come SPARK, ma questo è abbastanza comprensibile dato che la tecnologia è abbastanza moderna.

Inoltre non viene assolutamente menzionata la possibilità di sfruttare la parallelizzazione della computazione e dei dati in modo da utilizzare al massimo la macchina, per esempio utilizzando OpenCL o CUDA per la scrittura di programmi utilizzabili in GPU o FPGA, che sfrutterebbero la natura parallela di base di molti degli algoritmi utilizzati (che si basa comunque su un concetto di programmazione dinamica, dove soluzioni parziali possono essere utilizzate per arrivare a soluzioni globali).

Tutte le possibili implementazioni presentate sono quindi bloccate dal fatto di essere single thread (nella maggior parte degli algoritmi), oppure multithread ottimizzato per un singolo task (il caso di SNAP per la ricerca di motif temporali).

In conclusione, questa ricerca e costruzione di una nuova infrastruttura generale facilmente adattabile ai singoli casi è stata fatta su dei principi di similarità locale, in modo da sfruttare i sottoproblemi, e costruire una soluzione globale in modo semplice adattando un algoritmo di non semplice modificabilità. In contrasto, molti autori utilizzano delle definizioni globali che non considerano concetti come indipendenza tra gruppi di interazioni (definendo degli ordinamenti su tutto il grafo), o stabilendo dei vincoli mesoscopici su cammini e sottografi che devono essere rispettati (che dipendono sempre e comunque dal caso che si sta studiando e simulando).

Si vedrà nella prossima sottosezione come l'andamento dell'esecuzione dipenda comunque non dalla quantità di nodi in un grafo, ma dagli archi-contatti, che definiscono la struttura temporale ed influenzano direttamente la quantità di nodi da analizzare (che appartengono ai domini di compatibilità) ed il controllo utilizzato nella fase di matching.

4.1 Dati sperimentali

Di seguito vengono presentati i grafici che rappresentano le performance all'aumentare della dimensione e della struttura di un grafo temporale.

Si fa presente che questo studio non era incentrato su una implementazione che sfrutta l'hardware in modo ottimale, ma sulla definizione del problema generale e delle strategie per risolverlo, quindi le prestazioni di molti algoritmi generali (Mackey e TemporalRI) sono molto insoddisfacenti, e di fatto non sono stati scelti grafi particolarmente complessi.

Altri algoritmi specifici per singoli problemi dove è stata curata l'implementazione in modo da sfruttare un minimo di parallelismo (come il temporal-motif di SNAP) saranno ovviamente ottimizzati per i task per cui sono stati creati, e quindi più veloci degli algoritmi presentati.

Per la costruzione dei grafi target l'intenzione era quella di utilizzare un generatore di grafi temporali che si basasse su due distribuzioni, una per i possibili archi (che dovrebbe essere una Poisson per il grado di un nodo, quindi un modello di Barabasi-Albert basato su preferential attachment può andare bene), ed una dei possibili tempi di contatto (che in realtà stabilisce delle mode intervallari dove poter far accadere i contatti, quindi non è semplice da definire), alla fine si è optato per l'utilizzo di grafi temporali già disponibili leggermente modificati (prendendo i sottografi) per facilitare la ricerca delle occorrenze e non aumentare la complessità significativamente, e per la costruzione dei grafi target sintetici si è utilizzato un semplicissimo modello di Barabasi leggermente modificato per aggiungere tempi di contatto ad ogni arco aggiunto.

Per la costruzione del grafo query sono stati utilizzati grafi creati ad-hoc anche per simulare un procedimento reale, dove si vogliono trovare dei pattern specifici. Sono stati provati grafi query di dimensione crescente (uno da 4 nodi,uno da 16 nodi, uno da 32 e l'ultimo a 50 nodi), gli archi sono stati creati a mano.

Di seguito verranno considerati solo i casi di grafo query a 32 e 50 nodi, dato che è il più pesante computazionalmente.

Verrà considerato un confronto con l'algoritmo utilizzato in [8] modificato dato che non trovava le stesse occorrenze dell'algoritmo utilizzato in questo studio ed anche perché non era molto performante, anche dopo la modifica non trova le stesse occorrenze, ma almeno funziona per grafi abbastanza complessi.

L'implementazione di base di questo algoritmo è scaricabile su github (`https://github.com/pnnl/temporal_subgraph_isomorphism`). Non sono stati trovati altri algoritmi utilizzabili, funzionanti o modificabili per fare un confronto completo, inoltre i risultati non potrebbero essere quelli visti ma leggermente diversi (dato che il periodo degli esperimenti era intorno ad Agosto e la temperatura impediva il normale funzionamento dei dispositivi disponibili), comunque i rapporti tra i tempi e le performance dovrebbero essere gli stessi.

In seguito è stato fatto un confronto di SNAP e la sua implementazione di ricerca di motivi temporali e l'algoritmo TemporalRI, utilizzando le query definite che vengono utilizzate in SNAP (cioè 2-3 nodi e 3 archi), nei limiti dei singoli contatti possibili nella implementazione fatta per TemporalRI.

Vengono presentati singole misure per variabile perché durante alcune prove il tempo di esecuzione non sembrava fluttuare particolarmente (anche perchè il procedimento è poco variabile in tutti gli algoritmi).

Durante i test sperimentali è stato utilizzato un Acer Nitro 5 AN515.

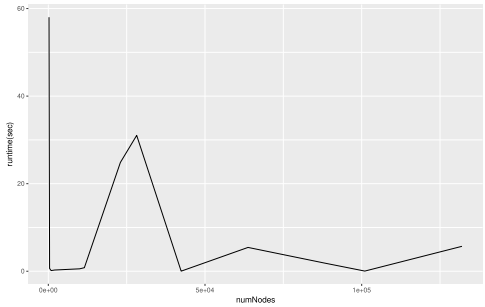
I grafi target presi sono relativamente pochi anche perché il tempo di esecuzione non variava particolarmente con alcune variabili (i nodi del grafo target), mentre al variare degli archi nel grafo target sembra seguire un andamento crescente (con una correlazione abbastanza tendente allo 0 perché comunque dipende dalla struttura di tutto il grafo e da come sono distribuiti gli archi, oltre all'andamento della struttura temporale).

Inoltre i tempi di esecuzione per certi tipi di grafi query e target erano abbastanza lunghi da mandare in crash il mio sistema.

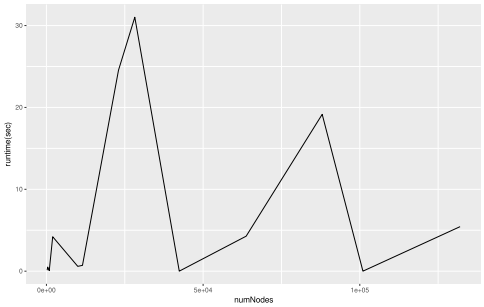
I seguenti grafici presentano i tempi di esecuzione preliminari generali dell'algoritmo sulla ricerca di grafi temporali query definiti su grafi temporali target reali.

I dataset utilizzati per queste prove non sono gli stessi della tabella 1, ma sono stati persi durante le prove dato che questi esperimenti erano tra i primi per il controllo di performance.

Questi grafi sono stati lasciati perché si può vedere come i tempi di esecuzione siano relativamente bassi, questo perché i grafi utilizzati venivano da reti che non erano propriamente temporali ma erano dati per sistemi di raccomandazione con un tipo di componente temporale che non convinceva, e dato che i tempi ricadevano in un intervallo temporale minimo , i domini di compatibilità risultanti erano molto piccoli e la computazione del matching vero e proprio ne risentiva pesantemente.

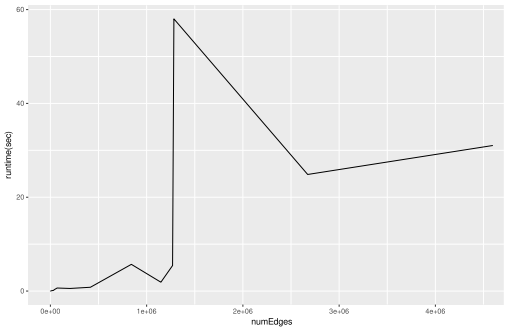


Query di 50 nodi

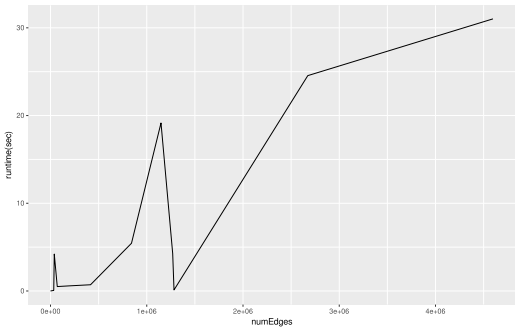


Query di 32 nodi

Il numero di nodi non sembrà influenzare direttamente il tempo di esecuzione, dato che non agisce direttamente sulla ricerca dei possibili match.



Query di 50 nodi



Query di 32 nodi

Nel caso della correlazione con l'aumento di archi in un grafo, sembra invece che l'algoritmo risenta del maggiore carico computazionale nel controllo dei vari archi, quindi il tempo di esecuzione assume un andamento crescente all'aumentare dei contatti in un grafo temporale.

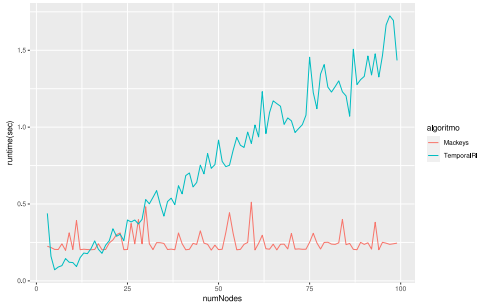
Una considerazione aggiuntiva può essere fatta sul fatto che i grafi query utilizzati siano stati creati ad-hoc, questo perché durante altri esperimenti con grafi query estratti da altri grafi temporali, i tempi di esecuzione aumentavano o diminuivano senza un pattern stabile, anche perché, nella maggior parte dei casi, questi dipendono pesantemente dalla struttura temporale sia del grafo query che del grafo target.

Di seguito si presentano i confronti tra TemporalRI e l'algoritmo utilizzato in [8] in grafi sintetici e reali, verso la fine viene presentato inoltre un confronto con SNAP di TemporalRI con grafi temporali reali:

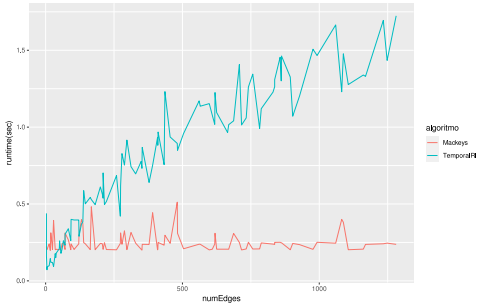
4.1.1 Performance su grafi sintetici

I grafi sintetici utilizzati sono, nel caso dei grafi query, sotto-grafi di altre reti temporali (100 sotto-grafi della rete *fb-forum*, con nodi da 3 a 103), e nel caso di studio della variazione delle performance in base alla dimensione del grafo target, sono grafi temporali generati con un basilare modello di Barabasi (cioè sfruttando il principio del preferential attachment) modificato per l'aggiunta della componente temporale, che viene stabilità in base a quando il nodo entra nella rete (per simulare l'entrata di un agente e l'interazione tra altri agenti in un certo tempo), il grado medio è all'incirca 3.

Di seguito viene presentato il confronto in base al numero di nodi e contatti nella query (e non nel grafo target), sempre di TemporalRI con Mackey:



Query incrememental i secondo il numero di nodi



Query incrementali secondo il numero di archi

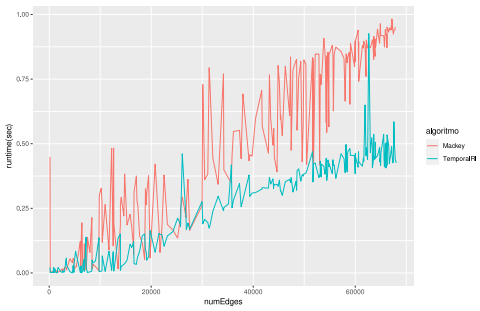
Come si può vedere, l'algoritmo di Mackey risulta migliore in quasi tutti i casi dato che mantiene tempi di esecuzione costanti indipendentemente dal numero di nodi o archi del grafo query, dove invece TemporalRI sembra assumere un comportamento lineare.

Viene presentato inoltre un confronto dei tempi di esecuzione della ricerca di isomorfismi di un grafo definito (con 10 nodi) in grafi target con nodi o archi variabili (secondo quanto esposto precedentemente), i risultati di seguito:

Nonostante TemporalRI sembra prevalere nelle performance, nella realtà i due algoritmi hanno tempi molto simili, e mediamente ci mettono lo stesso tempo per la ricerca di isomorfismi-motif.

Da questo risultato si può vedere che l'approccio edge-driven adottato da molti autori non è sempre la scelta migliore dato che, oltre al fatto che si possano trovare solo certi sotto-grafi senza contare certe occorrenze dipendenti dalla struttura definita dal problema (che non è detto sia sempre la solita con cammini time-respecting), si deve far fronte al numero crescente di archi da confrontare,

Target incrememental secondo il numero di nodi



Target incrementali secondo il numero di archi

cosa che viene tamponata dall'approccio utilizzato in RI con il controllo preventivo delle impronte e durante la fase di matching per togliere i candidati che non rispettino certe regole.

4.1.2 Performance su grafi reali

Durante questa sotto-sezione sono stati utilizzati grafi temporali reali trovabili in `http://networkrepository.com/temporal-networks.php`.

I dataset utilizzati sono i seguenti

Graph	|V|	|E|	d_{max}	d_{avg}
SFHH-conf-sensor	403	70K	2K	348
ca-cit-HepPh	28K	5M	11K	327
copresence-InVS15	219	1M	40K	12K
edit-enwiki-books	8K	162K	11K	38
email-dnc	2K	37K	6K	40
fb-forum	899	34K	2K	74
fb-wosn-friends	64K	1M	2K	39
ia-contacts-dublin	11K	416K	616	75

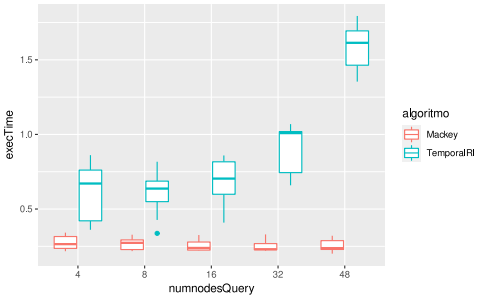
Tabella 1: dataset utilizzati

Come si vedrà dai prossimi dati sperimentali, il numero di nodi non sembrerà influenzare direttamente il tempo di esecuzione, e nonostante l'algoritmo confrontato con RI sia abbastanza poco performante per certi tipi di grafi temporali, sembra essere più veloce rispetto a TemporalRI per grafi temporali piccoli e con pochi contatti, e molto spesso i due algoritmi hanno gli stessi tempi di esecuzione, considerando dei limiti di possibili archi.

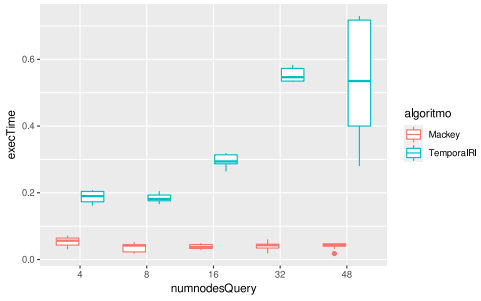
Durante gli esperimenti TemporalRI risulterà migliore nel caso in cui gli archi di notevole quantità, dato che il confronto non è edge-driven come l'algoritmo di Mackey, oltre a dare il giusto risultato in base alla definizione data durante questo studio, questi risultati non saranno comunque molto affidabili,

dato che i grafi utilizzati come target sono estremamente variabili, e molto spesso i contatti che vengono forniti avvengono tutti in un intervallo molto piccolo, quindi TemporalRI effettua un calcolo dei compatibility domains che distrugge completamente la struttura utilizzata da Mackey per la ricerca di motif, ed allo stesso tempo l'algoritmo di Mackey setaccia i contatti in modo da effettuare la ricerca in modo performante.

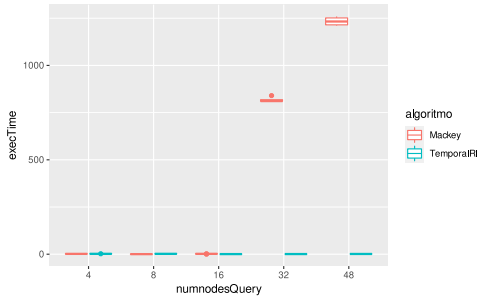
Viene effettuato un boxplot in base al numero di nodi di più query, in modo da analizzare i comportamenti dei due algoritmi su più dataset (alcuni di quelli specificati precedentemente) ed al variare dei grafi query:



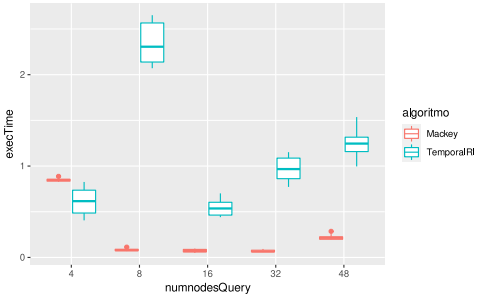
Boxplot 1: dataset edit-enwikibooks



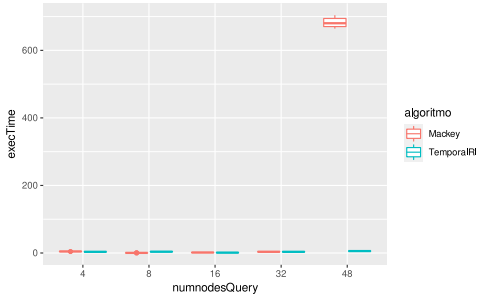
Boxplot 2: dataset fb-forum



Boxplot 3: dataset SFHH-conf-sensor



Boxplot 4: dataset email-dnc



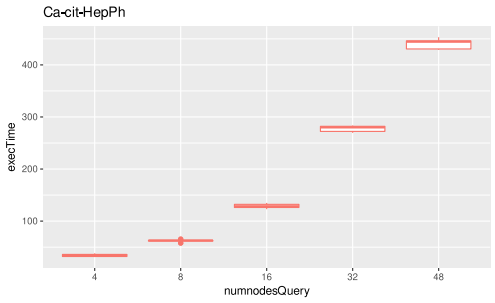
Boxplot 5: dataset ia-contacts-dublin

L'algoritmo di Mackey sembra essere più veloce nella maggior parte degli esperimenti, ma talvolta assume dei comportamenti inaspettati con nessun criterio apparente (per esempio aumentando il numero di nodi, l'algoritmo diventa immensamente più lento).

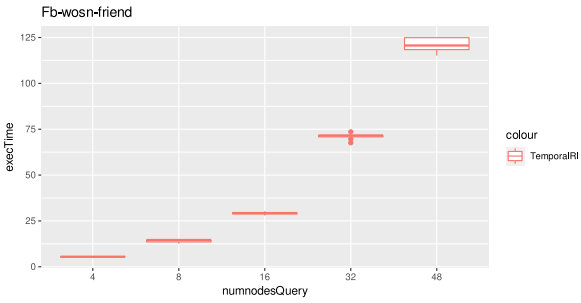
Questi risultati sono inoltre attuati su dei grafi abbastanza piccoli, di fatto tutti e due gli algoritmi finiscono la computazione entro pochi secondi. Non è

stato possibile utilizzare dataset più grandi dato che l'algoritmo di Mackey ha una gestione della memoria abbastanza naive, e per dimensioni di grafi relativamente grandi (sopra i 300000 archi), l'algoritmo occupa tutta la memoria ed infine si suicida.

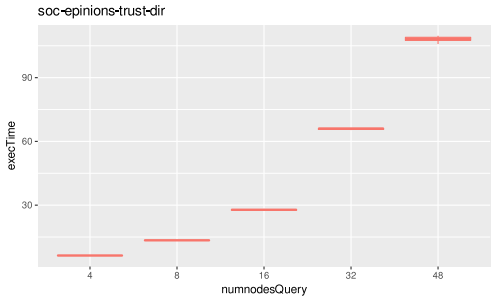
Vengono presentati i boxplot dei grafi reali dove l'algoritmo di Mackey andava out of memory:



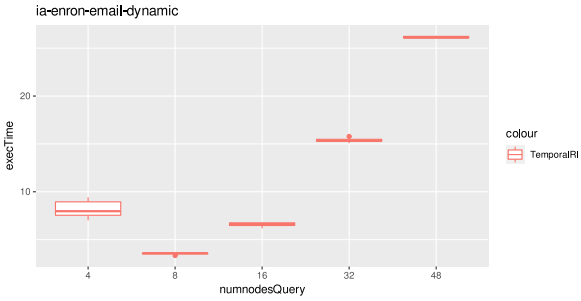
Boxplot 6:dataset Ca-cit-HepPh



Boxplot 7:dataset fb-wosn-friend

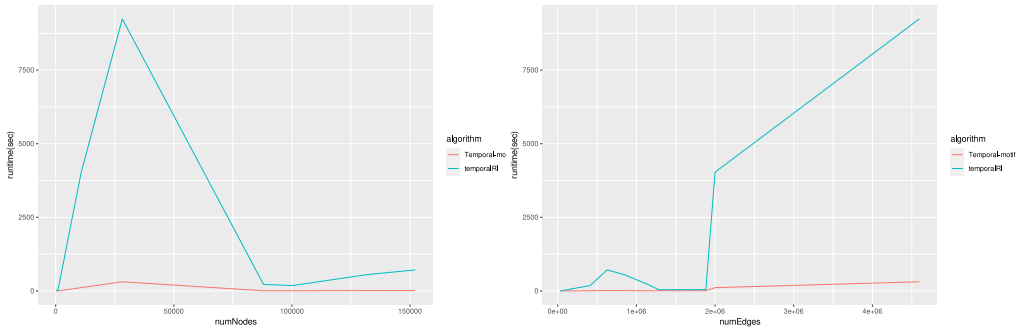


Boxplot 8:dataset soc-epinions-trust-dir



Boxplot 9:dataset ia-enron-email-dynamic

Di seguito il confronto tra i tempi di esecuzione di SNAP per ricerca di motif temporali (chiamato Temporal-motif nel grafico) e TemporalRI, in cui si considerano tutti i tempi di esecuzione di tutte le query per singolo grafo target (rappresentato dal numero di nodi o contatti appartenenti al grafo temporale) dato che SNAP non permette di sapere i singoli tempi di esecuzione per ogni grafo query, non è stato confrontato l'algoritmo di Mackey dato che per grafi di notevole grandezza non sembra funzionare (con il mio hardware), i dataset utilizzati sono sempre quelli definiti nella Tabella 1.



Ovviamente, l'algoritmo per la ricerca di temporal motif di SNAP è più veloce per qualsiasi input, dato che è stato creato per il singolo task di trovare tutte le occorrenze di 2-3 nodi e 3 archi, oltre a sfruttare OpenMP per l'utilizzo parallelo durante la computazione, e multithreading.

Alla fine si può vedere come un confronto edge-driven ottimizzato che utilizza anche strategie di pruning di archi e parallelismo dei controlli sia migliore di un normale algoritmo di ricerca basato su struttura che non utilizza parallelismo dati e task, quindi non ottimizzato per utilizzare componenti molto utili per diminuire il tempo di esecuzione.

Bisogna ricordare comunque che gli algoritmi presentati non danno gli stessi risultati (quantità di match-motif) anche perché utilizzano definizioni differenti di isomorfismo o motif oltre ad avere tempi abbastanza fluttuanti rispetto alla struttura temporale di base sia delle reti temporali query utilizzate, sia dei target dove ricercare gli isomorfismi.

5 Futuro

Per il futuro vi sono la definizione chiara di una base di formalità per gli isomorfismi per intervalli multipli per ogni arco, dato che è molto più complicata e dipendente da quello che è stato esposto precedentemente.

Dato che lo studio non era incentrato sulle performance, non si è scritto l'algoritmo in modo ottimale per aumentarne le prestazioni, quindi nel futuro si può fare il controllo della struttura temporale e l'equivalenza direttamente durante la fase di matching e controllo, dato che nella data in cui questa ricerca è stata scritta, il controllo di struttura temporale viene fatto alla fine del confronto sul sottografo per vedere l'equivalenza.

Inoltre il codice è stato scritto in scala anche per favorire una possibile implementazione con SPARK e la trasposizione nel mondo dello stream processing o della computazione distribuita di grafi con librerie come Neo4j o GraphX.

Possibile implementazione che sfrutta la natura dinamica e parallela del problema utilizzando tecnologie di GPGPU come OpenCL o CUDA, aumentando il throughput generale dell'algoritmo.

Una possibilità non visitata è l'implementazione dei grafi temporali insieme a modelli di Markov o altri modelli statistici in modo da modellare pure il cambiamento di stati ed i collegamenti in modo puramente statistico, creando una struttura di base più completa ed ibrida.

Elenco dei simboli

p_{itj} (e_{it},e_{tj}) propagazione dal nodo i al nodo j attraverso il nodo t

P_t insieme delle propagazioni di un nodo t

E insiemi degli archi di un grafo

G (V,E) grafo statico

GT (V,E) grafo temporale

V insiemi dei nodi di un grafo

Riferimenti bibliografici

[1] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14(S7):S13, 2013.

[2] Petter Holme and Jari Saramäki. *Temporal Network Theory*. Springer, 2019.

[3] Ali Jazayeri and Christopher C Yang. Motif discovery algorithms in static and temporal networks: A survey. *arXiv preprint arXiv:2005.09721*, 2020.

[4] Lauri Kovanen, Márton Karsai, Kimmo Kaski, János Kertész, and Jari SaramÃ€ki. Temporal motifs in time-dependent networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(11):P11005, nov 2011.

[5] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61, 2018.

[6] Paul Liu, Austin R Benson, and Moses Charikar. Sampling methods for counting temporal motifs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 294–302, 2019.

[7] Penghang Liu, Valerio Guarrasi, and A Erdem Sarıyüce. Temporal network motifs: Models, limitations, evaluation. *arXiv preprint arXiv:2005.11817*, 2020.

[8] Patrick Mackey, Katherine Porterfield, Erin Fitzhenry, Sutanay Choudhury, and George Chin. A chronological edge-driven approach to temporal subgraph isomorphism. *mackey2018chronological*, pages 3972–3979, 2018.

[9] Giovanni Micale, Vincenzo Bonnici, Alfredo Ferro, Dennis Shasha, Rosalba Giugno, and Alfredo Pulvirenti. Multiri: Fast subgraph matching in labeled multigraphs. *arXiv preprint arXiv:2003.11546*, 2020.

[10] Majid H Mohajerani, Allen W Chan, Mostafa Mohsenvand, Jeffrey LeDue, Rui Liu, David A McVea, Jamie D Boyd, Yu Tian Wang, Mark Reimers, and Timothy H Murphy. Spontaneous cortical activity alternates between motifs defined by regional axonal projections. *Nature neuroscience*, 16(10):1426, 2013.

[11] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610, 2017.

[12] Ursula Redmond and Pádraig Cunningham. Subgraph isomorphism in temporal networks. *arXiv preprint arXiv:1605.02174*, 2016.

[13] Xiaoli Sun, Yusong Tan, Qingbo Wu, Jing Wang, and Changxiang Shen. New algorithms for counting temporal graph pattern. *Symmetry*, 11(10):1188, 2019.