ANALISI

Nicola Sanitate

Rediscovering Workflow Models from Event-Based Data

Progetto di Intelligenza Artificiale

A.A. 2010 - 2011

UNIVERSITÀ DEGLI STUDI DI BARI

INDICE

ANALISI

OBIETTIVO

Si vuole realizzare un sistema che sia in grado di ricostruire un modello di workflow tramite l'analisi dei log di workflow estrapolati da sistemi informativi transazionali. In questo modo si automatizza e si rende più oggettiva la realizzazione dei progetti di workflow utili a guidare i workflow management system o per reingegnerizzare l'intera organizzazione dei processi. Un ruolo fondamentale verrà coperto dalla gestione del rumore nei dati di input.

Si scegli di implementare la tecnica presentata nell'articolo accademico “Rediscovering Workflow Models from Event-Based Data” di A.J.M.M. Weijters e W.M.P. van der Aalst del Department of Technology Management presso la Eindhoven University of Technology.

Questo documento ha lo scopo di riportare le nozioni fondamentali di questa tecnica utili per la successiva implementazione della stessa.

RAPPRESENTAZIONI

I dati di ingresso della tecnica presentata sono dei log di workflow, ossia una sequenza di eventi descritti da un identificatore del processo ed un identificatore del task. Dato che non è possibile trovare delle dipendenze causali tra task di due processi distinti, è possibile considerare una semplificazione assumendo che esista un unico processo di workflow. Tramite l'identificatore del processo sarà possibile suddividere il log di workflow in più parti, uno per ogni processo, ed applicare la tecnica ad ognuna di esse senza perdere informazione.

Date le considerazioni precedenti è possibile considerare un log di workflow come un insieme di sequenze di eventi dove ogni sequenza di eventi è rappresentabile tramite una sequenza di identificatori dei task.

T1, T2, T4, T3, T5, T9, T6, T3, T5, T10, T8, T11, T12, T2, T4, T7, T3, T5, T8, T11, T13

La tecnica presentata produce un modello di workflow di un singolo processo rappresentato tramite una rete di workflow, ossia una rete di Petri che modella il flusso di controllo di un workflow.

Le transizioni rappresentano i task, mentre i posti rappresentano le dipendenze causali, ossia ad un posto corrisponde una condizione che può essere usata come pre- condizione e/o post-condizione per un task . Le transizioni scattano seguendo le classiche regole delle reti di Petri.

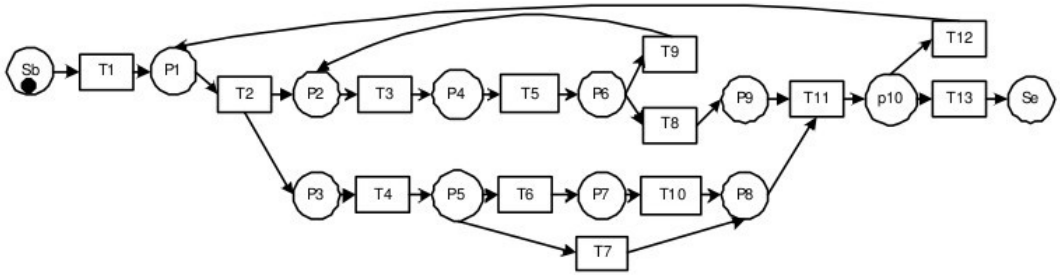

Immagine 1: Rete di workflow

La rete deve rispettare i seguenti requisiti:

1. possedere un solo posto sorgente ed un solo posto destinazione, in quanto essa specifica il ciclo di vita di un unico processo;
2. non presentare task e condizioni che non contribuiscano al trattamento del caso, ossia tutti i nodi del workflow dovrebbero essere su qualche percorso tra il posto sorgente ed il posto destinazione;

3. soundness, ossia
    - terminazione garantita,
    - assenza di token pendenti nella rete al termine del processo,
    - assenza di task morti.

La reale complessità nella ricostruzione del modello di workflow a partire dalle sequenze di workflow sta nell'individuazione delle seguenti quattro tipologie di connessione:

- AND-split, transizioni con due o più posti di output;
- AND-join, transizioni con due o più posti di input;
- OR-split, transizioni con un posto di output che presenta archi multipli uscenti;
- OR-join, transizioni con un posto di input che presenta archi multipli entranti.

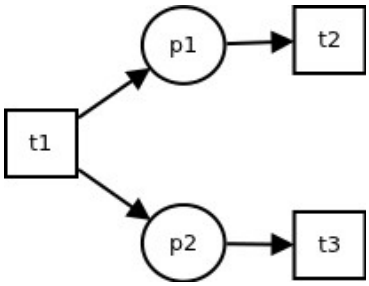

Immagine 2: AND-split

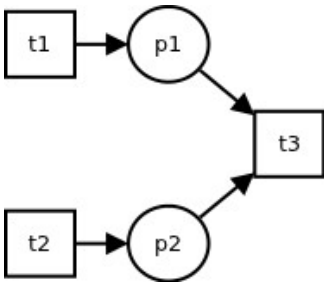

Immagine 3: AND-join

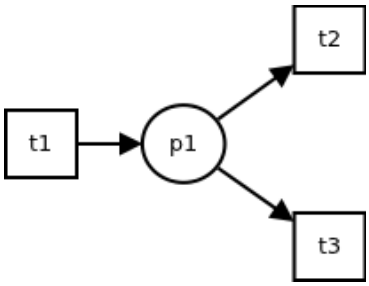

Immagine 4: OR-split

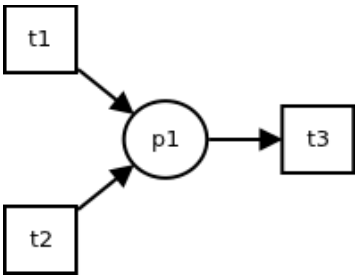

Immagine 5: OR-join

Date le considerazioni precedenti è possibile riformulare con maggior precisione l'obiettivo di progetto: dato un log di workflow si vuole ricostruire un modello di workflow che generi potenzialmente tutte le sequenze di eventi che appaiono nel log di workflow, generi quanto meno possibile sequenze di eventi non presenti nel log di workflow, catturi comportamenti concorrenti e/o ciclici, e sia quanto più semplice e compatta possibile ai fini della comprensione.

TECNICA

La tecnica si sviluppa in tre fasi:

1. costruzione di una tabella delle dipendenze e frequenze (D/F-table) a partire dalle sequenze di task presenti nei log di workflow;
2. induzione di un grafo delle dipendenze e frequenze (D/F-graph) a partire dai dati della D/F-table ottenuta precedentemente;
3. generazione della rete di workflow (WF-net) a partire dai dati della D/F-table e dal D/F-graph ottenuti precedentemente.

COSTRUZIONE DELLA D/F-TABLE

La D/F-table è una tabella che raccoglie per ogni task (A) una serie di informazioni in relazione ai task presenti nel log di workflow. In particolare:

- la frequenza complessiva del task ($\#A$);
- per ogni task (B):
  - la frequenza della precedenza diretta tra il task B ed il task A ($B<A$);
  - la frequenza della successione diretta tra il task B ed il task A ($A>B$);
  - la frequenza della precedenza tra il task B ed il task A ($B<<<A$);
  - la frequenza della successione tra il task B ed il task A ($A>>>B$);
  - una metrica che indica la forza della relazione causale tra A e B ($A \rightarrow B$).

L'ultima metrica presentata quantifica l'intuizione per la quale se il task A occorre spesso prima del task B e il task B occorre poco spesso prima del task A, allora è presumibile che il task A sia causa del task B. Corrisponde ad un numero compreso tra [-1,1] calcolato incrementandolo di un fattore δ^n quando il task A occorre prima del task B, decrementandolo di un fattore δ^n quando il task B occorre prima del task A, e dividendolo infine per la frequenza complessiva del task A. Nel fattore δ^n, n rappresenta il numero di eventi intermediari tra i due task esaminati, e δ è un fattore di causalità compreso tra [0,1] scelto arbitrariamente.

A	#A	B	B<A	A>B	B<<<A	A>>>B	A→B
T6	1035	T10	0	581	348	1035	0,8
		T5	80	168	677	897	0,27
		T11	0	0	528	1035	0,19
		T13	0	0	0	687	0,16
		T9	50	46	366	538	0,16
		T8	68	31	560	925	0,12
		T3	146	209	831	808	0,02
		T6	0	0	348	348	0
		T7	0	0	264	241	-0,01
		T12	0	0	528	505	-0,09
		T1	0	0	687	0	-0,25
		T2	0	0	1035	505	-0,49
		T4	691	0	1035	505	-0,83

Tabella 1: Riga della D/F-table relativa al task T6

INDUZIONE DEL D/F-GRAPH

Il D/F-graph è un grafo orientato i cui nodi rappresentano i diversi task presenti nel log di workflow, e gli archi rappresentano la relazione di causalità dal nodo in uscita al nodo in entrata.

Tale grafo si ottiene partendo dai dati presenti nella D/F-table ottenuta nella fase precedente e applicando la seguente euristica per ogni coppia di task:

$$IF((A \rightarrow B \geq N) \text{ AND } (A > B \geq \sigma) \text{ AND } (B < A \leq \sigma)) \text{ THEN } <A,B> \in G$$

dove N è il fattore di rumore compreso tra [0,1] e scelto in base alla stima di rumore atteso nel dati di input, e σ è un valore soglia calcolato automaticamente:

$$\sigma = 1 + Round(N * \#S / \#T)$$

dove N è il fattore di rumore, $\#S$ è il numero di sequenze presenti nel log di workflow, e $\#T$ è il numero di task distinti presenti nel log di workflow.

Tale euristica formalizza l'intuizione per la quale se la causalità tra il task A ed il task B è alta, e il task A è frequentemente seguito direttamente dal task B, e ancora il task A è poco frequentemente preceduto direttamente dal task B, allora è presumibile che il task A sia causa del task B, e quindi l'arco orientato $<A,B>$ può essere incluso nel grafo. I termini "frequentemente" e "poco frequentemente" si quantificano rispettivamente come al di sopra ed al di sotto del valore soglia σ.

Purtroppo la prima euristica fallisce in presenza di ricorsioni o piccoli cicli: in questi casi i pattern nelle sequenze mostrano come i task in esame sembrino sia causa che conseguenza di essi stessi in caso di ricorsione, o di altri task in caso di piccoli cicli. In entrambi i casi quindi la causalità sarà prossima allo 0, impedendo di applicare la prima euristica.

I problemi illustrati in precedenza possono essere risolti applicando le seguenti euristiche per ogni coppia di task scartata dalla prima euristica:

$$IF((A \rightarrow A \approx 0) \text{ AND } (A < A + A > A > 0{,}5 * \#A) \text{ AND } (A < A - A > A \approx 0)) \text{ THEN } <A,A> \in G$$

$$IF((A \rightarrow B \approx 0) \text{ AND } (A > B \geq \sigma) \text{ AND } (B < A \approx A > B) \text{ AND } (A >>> B \geq 0{,}4 * \#A) \text{ AND } (B <<< A \approx A >>> B)) \text{ THEN } <A,B> \in G$$

Tali euristiche risolvono rispettivamente il problema della ricorsione ed il problema dei piccoli cicli. La prima infatti formalizza l'intuizione per la quale se analizzando un task A nei confronti di se stesso si rileva una causalità prossima allo 0, la somma tra precedenza diretta e successione diretta maggiore del 50% della frequenza complessiva, e ancora la differenza tra precedenza diretta e successione diretta prossima allo 0, allora è presumibile che si sia in presenza di un caso di ricorsione, e quindi l'arco orientato $<A,A>$ può essere incluso nel grafo. La seconda invece formalizza l'intuizione per la quale se analizzando un

task *A* nei confronti di un task *B* si rileva una causalità prossima allo 0, una successione diretta frequente, una precedenza diretta prossima alla successione diretta, una successione maggiore del 40% della frequenza complessiva, e ancora una precedenza prossima alla successione, allora è presumibile che si sia in presenza di un caso di piccolo ciclo, e quindi l'arco orientato *<A,B>* può essere incluso nel grafo. Con il termine "prossimo" si intende che le differenza relativa tra i due membri è inferiore al fattore di rumore.

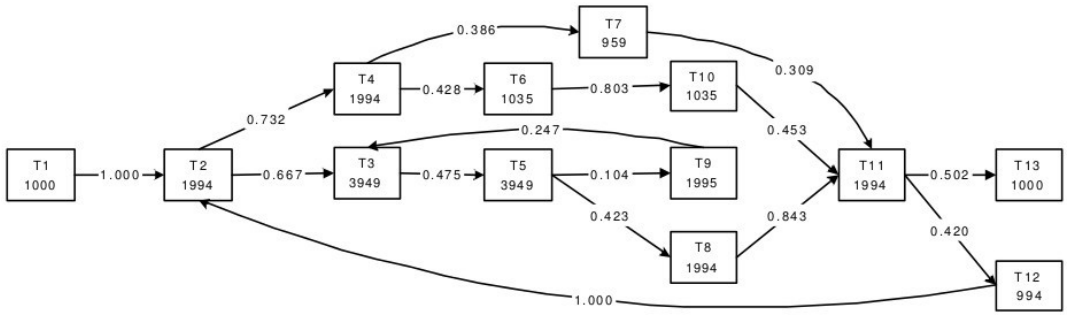

Immagine 6: D/F-graph ottenuto dalla rete di workflow presentata precedentemente

GENERAZIONE DELLA WF-NET

La WF-net è il prodotto finale della tecnica di Process Mining presentata. Essa aggiunge al D/F-graph informazioni relative alla tipologia di split e di join che consentono raggiungere una rappresentazione completa del modello di workflow.

Tali informazioni si possono ottenere in più modi equivalenti:

- dai valori delle precedenze dirette e successioni dirette nella D/F-table;
- dai valori delle frequenze dei nodi nel D/F-graph;
- dai valori di causalità tra i nodi nel D/F-graph.

Delle precedenze dirette e successioni dirette frequenti di due o più task di destinazione (partenza) di uno split (join) sono indice del fatto che questi task vengano eseguiti in modo parallelo, e quindi è presumibile che si sia in presenza di un AND-split (AND-join). Al contrario, delle precedenze dirette e successioni dirette poco frequenti sono indice del fatto che questi task vengano eseguiti in modo opzionale esclusivo, e quindi è presumibile che si sia in presenza di un OR-split (OR-join).

Inoltre una frequenza del task di partenza (destinazione) di uno split (join) uguale alle frequenze dei task di destinazione (partenza) è indice del fatto che per ogni attivazione del task di partenza (destinazione) c'è una attivazione di tutti i task di destinazione (partenza), e quindi è presumibile che si sia in presenza di un AND-split (AND-join). Al contrario, una frequenza del task di partenza (destinazione) di uno split (join) uguale alla somma di tutte le frequenze dei task di destinazione (partenza) è indice del fatto che ad ogni attivazione del task di partenza (destinazione) corrisponde l'attivazione di un solo task di destinazione (partenza), e quindi è presumibile che si sia in presenza di un OR-split (OR-join).

Infine una forte causalità tra il task (i task) di partenza di uno split (join) e i task (il task) di destinazione è indice del fatto che il task (i task) di partenza causa (causano) l'attivazione di tutti i task (del task) di destinazione contemporaneamente, e quindi è presumibile che si sia in presenza di un AND-split (AND-join). Al contrario una debole causalità tra il task (i task) di partenza di uno split (join) e i task (il task) di destinazione è indice del fatto che il task (i task) di partenza causa (causano) l'attivazione di tutti i task (del task) di destinazione alternatamente, e quindi è presumibile che si sia in presenza di un OR-split (OR-join).

Di seguito si riporta in pseudo-codice un algoritmo basato sulla prima osservazione il quale basta a classificare ogni connessione in quanto, come specificato precedentemente, le osservazioni sono da ritenersi equivalenti. Tale algoritmo prende in input uno split (join) e suddivide i task di destinazione (partenza) in insiemi. I task all'interno di un insieme sono da ritenersi in OR-split (OR-join) e gli insiemi sono da ritenersi in AND-split (AND-join).

```
FOR i:=1 TO n DO
BEGIN
   FOR j:=1 TO n DO
   BEGIN
      OK:=False;
      REPEAT
         IF ∀X∈Set_j [(B_i>X< σ) AND (X>B_i< σ)] THEN
         BEGIN
            Set_j:=Set_j∪{B_i};
            OK:=True;
         END;
            IF Set_j = ∅  THEN
         BEGIN
            Set_j:={B_i};
            OK:=True;
         END;
      UNTIL OK;
   END j DO;
END i DO;
```

ERRORI

Di seguito vengono riportati le carenze riscontrate durante lo studio dell'articolo scientifico alla base di questo progetto.

1. Pagina 5, colonna destra. Nella prima euristica, e successivamente anche nelle altre due, viene mostrato che un arco che risponde ai requisiti richiesti dall'euristica può essere conservato in un insieme *T*. In seguito, durante la spiegazione del calcolo del valore di soglia σ, viene definito *T* come un insieme di nodi. Questa è senza dubbio una ambiguità dovuta ad uno scorretto utilizzo della stessa variabile: nel primo caso si parla di un insieme di archi che rientrano nella definizione del D/F-graph, nel secondo caso si parla dell'insieme di nodi presenti nel D/F-graph, e quindi di task distinti presenti nel log di workflow. Tale ambiguità è stata risolta dando nome *G* al primo insieme.
2. Pagina 5, colonna destra. Nell'enunciato della prima euristica un requisito richiede che la precedenza diretta tra due task $B<A$ debba essere minore o uguale del valore di soglia σ. In seguito, nella spiegazione di tale euristica viene affermato esattamente il contrario, ossia che la precedenza diretta tra i due task debba essere maggiore o uguale del valore di soglia. Data l'intuizione alla base dell'euristica sembra ovvio che la prima è l'interpretazione giusta: un arco tra due task $<A,B>$ può essere incorporato nel D/F-graph se la precedenza diretta tra il nodo di partenza *A* ed il nodo destinazione *B* è poco frequente, e quindi al di sotto del valore di soglia σ.
3. Pagina 6, colonna destra. Nello pseudo-codice dell'algoritmo presentato non si tiene in considerazione il fatto che gli insiemi iniziali siano vuoti, quindi non è possibile effettuare un confronto con i loro elementi. In tal caso non sarà mai possibile riempire gli insiemi in quanto la struttura di controllo *IF-THEN* fallirà sempre. Tale problema è stato risolto aggiungendo un ulteriore struttura di controllo *IF-THEN* al di sotto della precedente che prevede l'aggiunta di un task ad un insieme qualora quest'ultimo risulti vuoto, nonché l'uscita dalla struttura REPEAT-UNTIL.