Riscoprire modelli di workflow da dati basati su eventi

Abstract

I workflow management system contemporanei sono guidati da modelli di processo espliciti, ossia è necessario un progetto di workflow completamente specificato al fine di mettere in atto un dato processo di workflow. Creare un processo di workflow è un processo complicato e che richiede tempo, e spesso ci sono discrepanze tra i processi di workflow attuali e i processi percepiti dal management. Quindi noi proponiamo una tecnica per riscoprire i modelli di workflow. Questa tecnica usa i log di workflow per scoprire come il processo di workflow viene attualmente eseguito. I log di workflow contengono informazioni riguardanti gli eventi che si svolgono. Assumiamo che questi eventi siano totalmente ordinati ed ogni evento si riferisce ad un task in esecuzione per un singolo caso. Questa informazione può facilmente essere estratta dai sistemi informativi transazionali (ad esempio, ERP come SAP o Baan). La tecnica di riscoperta proposta in questo paper è in grado di affrontare il rumore e può essere usata per validare i processi di workflow scoprendo e misurando le discrepanze tra i modelli prescrittivi e l'esecuzione attuale del processo.

Introduzione

Durante l'ultima decade i concetti e le tecnologie di workflow management sono stati applicati in molti sistemi informativi di impresa. Workflow management system come Staffware, IBM MQSeries, COSA, ecc. offrono modellazione generica e capacità di promulgazione per processi di business strutturati. Realizzando definizioni grafiche del processo, ossia modelli che descrivono il ciclo di vita di un caso tipico (istanza di workflow) isolato, è possibile configurare questi sistemi per supportare i processi di business. Oltre ai workflow management system puri, anche molti altri software hanno adottato tecnologie di workflow. Consideriamo ad esempio gli ERP come SAP, PeopleSoft, Baan e Oracle, i software CRM, ecc. Nonostante le loro promesse, si incontrano molti problemi quando si applicano tecnologie di workflow. Come indicato da diversi autori, i workflow management system sono troppo restrittivi e hanno problemi nell'affrontare i cambiamenti. Molti workshop e numeri speciali di riviste si sono dedicati alle tecniche per rendere il management del workflow più flessibile. Molta della ricerca in questa area si è dedicata alle tecniche per aumentare la flessibilità sia permettendo cambiamenti ad-hoc (come risulta dai workflow management system come InConcert) o fornendo meccanismi per migrare i casi ed evolvere i workflow.
In questo paper prendiamo una prospettiva differente rispetto ai problemi relativi alla flessibilità. Dimostriamo come molti problemi sono dovuti alla discrepanza tra il progetto di workflow (ossia la costruzione di modelli predefiniti di workflow) e la promulgazione del workflow (ossia l'esecuzione attuale dei workflow). I progetti di workflow sono realizzati tipicamente da un piccolo gruppo di consulenti, manager e specialisti. Come risultato, il progetto iniziale di un workflow è spesso incompleto, soggettivo, e ad un livello troppo alto. Durante l'implementazione del workflow, ossia configurazione del workflow management system e addestramento dei lavoratori coinvolti, queste questioni causano molti problemi (“Il diavolo è nei dettagli”). Quindi, noi proponiamo di “invertire il processo”. Invece di partire con il progetto do workflow, partiamo con la raccolta dei dati riguardo ai processi di workflow e come questi si svolgono. Assumiamo che sia possibile registrare eventi in modo tale che (i) ogni evento si riferisca ad un task (ossia un passo ben definito nel workflow), (ii) ogni evento si riferisce ad un caso (ossia un istanza di workflow), e (iii) gli eventi sono ordinati totalmente. Ogni sistema informativo che utilizza un sistema transazionale come ERP, CRM, o workflow management system offriranno queste informazioni in una qualche forma. Si noti che non assumiamo la presenza di un workflow management system. La sola assunzione che facciamo e che sia possibile costruire dei log di workflow con i dati degli eventi. Questi log di workflow sono usati per costruire una specifica di processo, che modelli adeguatamente i comportamenti registrati. Usiamo il termine process mining per il metodo di distillazione di una descrizione di processo strutturata partendo da un insieme di esecuzioni reali. In questo paper,

proponiamo una nuova tecnica di process mining.
Il resto di questo paper è come segue. Inizialmente discutiamo i lavori relativi. La sezione 3 introduce alcuni preliminari includendo un linguaggio di modellazione per i processi di workflow e la definizione dei log di workflow. Poi presentiamo una nuova tecnica di process mining. Infine concludiamo il paper riassumendo i risultati principali ed evidenziando i lavori futuri.

Lavori relativi

L'idea di process mining non è nuova. Cook e Wolf hanno investigato su questioni simili nel contesto dei processi di ingegneria del software. Loro descrivono tre metodi di scoperta dei processi: uno usando le reti neurali, uno usando un approccio algoritmico puro, e uno con approccio markoviano. Gli autori considerano gli ultimi due gli approcci più promettenti. L'approccio algoritmico puro costruisce una macchina a stati finiti dove gli stati sono fusi se le loro caratteristiche (in termini di comportamenti possibili nei prossimi k passi) sono identiche. L'approccio markoviano usa una mistura di metodi algoritmici e statistici ed è in grado di affrontare il rumore. Si noti che i risultati presentati sono limitati a comportamenti sequenziali. In seguito Cook e Wolf hanno espeso il loro lavoro ai processi concorrenti. Propongono anche delle metriche specifiche (entropia, conto dei tipi di evento, periodicità, e causalità) ed usano queste metriche per scoprire i modelli di flussi di eventi. Questo approccio è simile ad uno presentato in questo paper. Comunque, le nostre metriche sono abbastanza differenti ed il nostro obiettivo finale è cercare rappresentazioni esplicite per un ampio range di modelli di processo, ossia noi generiamo una rete di Petri concreta invece di un insieme di relazioni di dipendenza tra eventi. Cook e Wolf forniscono una misura per quantificare le discrepanze tra un modello di processo e il comportamento attuale registrato usando i dati basati su eventi.
Cook e Wolf hanno introdotto per primi l'idea di applicare il process mining al contesto del management del workflow. Il lavoro è basato sui grafici di workflow, che sono ispirati ai prodotti di workflow come IBM MQSeries workflow (conosciuto precedentemente come Flowmark) e InConcert. In questo paper sono definiti due problemi. Il primo è cercare eventi generati da un grafico di workflow che appaiono in un dato log di workflow. Il secondo problema è cercare le definizioni delle condizioni marginali. Per affrontare il primo problema viene fornito un algoritmo concreto. L'approccio è abbastanza differente dall'approccio presentato in questo paper. Data la natura dei grafici di workflow non c'è necessità di identificare la natura (AND o OR) di unioni e divisioni. Inoltre, i grafici di workflow sono aciclici. L'unico modo di affrontare le iterazioni è enumerando tutte le occorrenze di una data attività. Viene anche presentato un tool basato su questo algoritmo.
Anche Herbst e Karagiannis indirizzano la questione del process mining nel contesto del management del workflow. L'approccio usa il linguaggio di modellazione ADONIS ed è basato su un modelli markoviani nascosti dove i modelli sono fusi e divisi al fine di scoprire i processi sottostanti. I lavori presentati si limitano ai modelli sequenziali. Una differenza notevole con gli altri approcci è che la stessa attività può apparire diverse volte nel modello di workflow. I risultati incorporano la concorrenza ma assumono anche che i log di workflow contengano esplicite informazioni causali. L'ultima tecnica è simile a quella degli autori precedenti e soffre lo svantaggio che la natura delle divisioni e delle unioni (ossia AND e OR) non venga scoperta.
Rispetto ai lavori esistenti noi ci focalizziamo sui processi di workflow con comportamenti concorrenti, ossia, individuare la concorrenza è una dei nostri primi interessi. Quindi, vogliamo distinguere esplicitamente divisioni/unioni AND/OR. Per raggiungere questo obiettivo combiniamo tecniche di apprendimento automatico con reti di WorkFlow (WF-net). Le WF-net sono un sottoinsieme delle reti di Petri. Si noti che le reti di Petri forniscono un linguaggio grafico ma formale progettato per modellare la concorrenza. Inoltre, la corrispondenza tra i workflow management system commerciali e le WF-net è ben sottintesa.

Preliminari: le WF-net e i log di evento

I workflow sono per definizione basati su casi, ossia ogni pezzo di lavoro è eseguito per uno specifico caso. Esempi di casi sono una ipoteca, un reclamo assicurativo, una dichiarazione fiscale, un ordine, una richiesta di informazione. L'obiettivo del management di workflow è di trattare i casi quanto più efficacemente ed efficientemente possibile. Un processo di workflow i progettato per trattare casi simili. I casi sono trattati eseguendo dei task in un ordine specifico. I modelli di processo di workflow specificano quali task necessitano di essere eseguiti ed in quale ordine. Termini alternativi di modello solo: 'procedure', 'grafici di workflow', 'diagrammi di flusso' e 'definizione di instradamento'. Nel modello di processo di workflow, gli elementi di instradamento sono usati per descrivere gli instradamenti sequenziali, condizionali, paralleli ed iterativi specificano così il percorso appropriato del caso. Molti casi possono essere trattati seguendo la stessa definizione di processo di workflow. Di conseguenza, lo stesso task deve essere eseguito per molti casi.

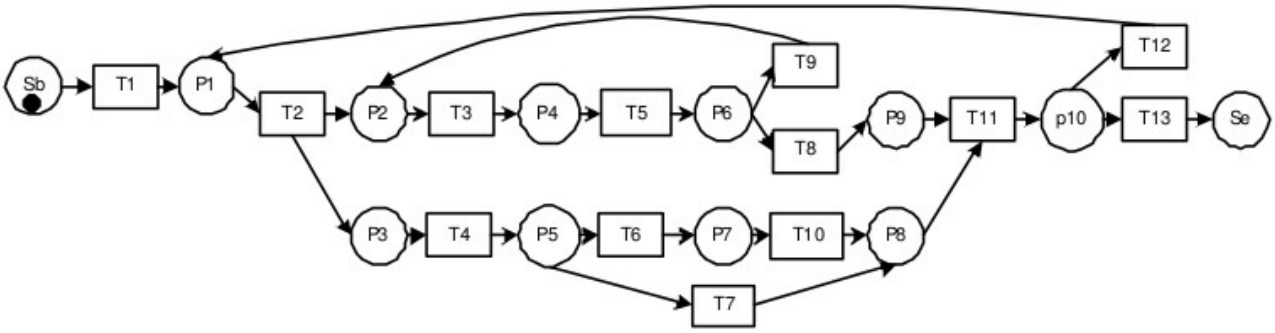

Illustration 1: Esempio di un processo di workflow modellato come una rete di Petri.

Le transazioni T1, T2, …,T13 rappresentano i task, i posti Sb, P1, …, P10, Se rappresentano le dipendenze causali. Infatti, un posto corrisponde ad una condizione che può essere usata come pre-condizione e/o post-condizione per un task. Una divisione-AND corrisponde ad una transazione con due o più posizioni di output (da T2 a P2 e P3), una unione-AND corrisponde ad una transazione con due o più posizioni di input (da P8 e P9 a T11). Una divisione-OR/unione-OR corrisponde a posti con archi multipli uscenti/entranti (da P5 a T6 e T7, e da T7 e T10 a P8). In ogni momento un posto contiene zero o più token, disegnato come un punto nero. Le transazioni sono le componenti attive in una rete di Petri: essi cambiano lo stato della rete seguendo le seguenti regole:

1. una transazione t si dice abilitata se e solo se ogni posto di input di t contiene almeno un token;
2. una transizione abilitata può scattare; se la transizione t scatta, allora t consuma un token da ogni posto di input p di t e produce un token per ogni posto di output p di t.

Una rete di Petri che modella la dimensione del flusso di controllo di un workflow, si chiama rete di WorkFlow (WF-net). Una WF-net ha un posto sorgente (Sb) ed un posto destinazione (Se) in quanto ogni caso (una istanza di workflow) trattata da una procedura rappresentata da una WF-net è creata quando entra nel workflow management system ed è cancellata quando è completamente trattata, ossia la WF-net specifica il ciclo di vita di un caso. Un requisito aggiuntivo è che non dovrebbero esserci “task e/o condizioni pendenti”, ossia task e condizioni che non contribuiscono al trattamento del caso. Quindi, tutti i nodi del workflow dovrebbero essere su qualche percorso tra il sorgente alla destinazione.

La WF-net si focalizza sulla prospettiva del processo e astrae dalle prospettive funzionali, organizzative, informative e operative. Queste prospettive possono essere aggiunte usando ad esempio una rete di Petri ad alto livello, ossia reti estese con colori (dati) e gerarchie. Sebbene le WF-net siano molto semplici, il loro potere espressivo è impressionante. In questo paper ci limitiamo alle cosiddette sound WF-net. Una rete di workflow è sound se sono soddisfatti i seguenti requisiti: (i) la terminazione è garantita, (ii) al termine, nessuna referenza (token) pendente è lasciata dietro, e (iii) non ci sono task morti, ossia dovrebbe essere possibile eseguire un task arbitrario seguendo il percorso appropriato. La soundness è la proprietà minima che ogni rete di

workflow dovrebbe soddisfare. Si noti che la soundness implica l'assenza di blocchi e punti porti. Le sound WF-net possono essere usate per modellare i costrutti base identidicati dal Workflow Management Coalition ed usati contemporaneamente nei workflow management system. L'esperienza con i principali workflow management system come Staffware, COSA, MQSeries Workflow, ecc. mostra che è abbastanza semplice esprimere questi linguaggi specifici dei tool in termini di linguaggio indipendente dai tool delle WF-net. Si rimanda ad una introduzione sulle WF-net.
In questo paper, noi usiamo i log di workflow per scoprire modelli di workflow espressi in termini di WF-net. Un log di workflow è una sequenza di eventi. Ogni evento è descritto da un identificatore del caso e un identificatore del task. Un evento e=(c,t) corrisponde all'esecuzione di un task t per un dato caso c. Per ragioni di semplicità assumiamo che ci sia solo un processo di workflow. Si noti che questa non è una limitazione dato che l'identificatore del caso può essere usato per dividere il log di workflow in log di workflow separati per ogni processo. Un log di workflow può contenere informazioni riguardo migliaia di casi. Dato che non ci sono dipendenze causali tra eventi corrispondenti a casi diversi, possiamo progettare il log di workflow su una sequenza di eventi separati per ogni caso senza perdere alcuna informazione. Quindi, possiamo considerare un log di workflow come un insieme di sequenze di eventi dove ogni sequenza di eventi è semplicemente una sequenza di identificatori di task. Formalmente, $WL \subseteq T^*$ dove WL è un log di workflow e T è un insieme di task. Un esempio di sequenza di eventi della rete di Petri di Illustration 1 è fornita di seguito:

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

Usando la definizione per WF-net e log di eventi possiamo facilmente descrivere il problema cui si dedica questo paper: dato un log di workflow WL vogliamo scoprire una WF-net che (i) generi potenzialmente tutte le sequenze di eventi che appaiono in WL, (ii) generi quanto meno possibile sequenze di T*\WL, (iii) catturi comportamenti concorrenti, e (iv) sia quanto più semplice e compatto possibile. Inoltre, per rendere la nostra tecnica applicabile praticamente vogliamo essere capaci di affrontare il rumore.

Tecnica di process mining

In questa sezione noi presentiamo i dettagli della nostra tecnica di process mining. Possiamo distinguere tre passi di estrazione: (i) costruzione di una tabella di dipendenza/frequenza (D/F-table), induzione di un D/F-graph da una D/F-table, e (iii) ricostruzione della WF-net dalla D/F-table e dal D/F-graph.

Costruzione della D/F-table

Il punto di partenza della nostra tecnica di workflow mining è la costruzione di una D/F-table. Per ogni task A le seguenti informazioni sono astratte dal log di workflow: (i) la frequenza complessiva del task A (denotato da #A), (ii) la frequenza del task A preceduto direttamente da un altro task B (denotato da B<A), (iii) la frequenza di A seguito direttamente dal un altro task B (denotato da A>B), (iv) la frequenza di A direttamente o indirettamente preceduta da un altro task B ma prima della comparsa precedente di A (denotata da B<<<A), (v) la frequenza di A direttamente o indirettamente seguita da un altro task B ma prima della successiva apparizione di A (denotata da A>>>B), e infine (vi) una metrica che indica la forza della relazione causale tra il task A e un altro task B (denotata da $A \rightarrow B$).
Le metriche da (i) a (v) non hanno bisogno di spiegazioni. L'intuizione sottostante la metrica (vi) è la seguente. Se si verifica sempre che, quando occorre il task A, poco dopo occorre anche il task B, allora è plausibile che il task A causa l'occorrenza del task B. D'altro canto, se B occorre (poco) prima del task A è implausibile che il task A sia la causa del task B. Di seguito definiamo la formalizzazione di questa intuizione. Se, in un flusso di eventi, il task A occorre prima del task B e n è il numero di eventi intermediari tra loro, il contatore $A \rightarrow B$-causalità è incrementato di un fattore (δ^n). δ è un fattore di causalità (compreso tra [0,0 … 1,0]). nel nostro esperimento δ è settato a 0,8. L'effetto è che il contributo alla metrica di causalità è al massimo 1 (se il task B compare

direttamente dopo il task A allora n=0) e decresce se la distanza incrementa. Il processo di guardare avanti dal task A fino all'occorrenza del task B si ferma dopo la prima occorrenza del task A o task B. Se il task B occorre prima del task A ed n è nuovamente il numero di eventi intermediari tra loro, il contatore $A \rightarrow B$-causalità è divisa per la frequenza complessiva del task A (#A).

	#A	B<A	A>B	B<<<A	A>>>B	A→B
T10	1035	0	581	348	1035	0,803
T5	3949	80	168	677	897	0,267
T11	1994	0	0	528	1035	0,193
T13	1000	0	0	0	687	0,162
T9	1955	50	46	366	538	0,161
T8	1994	68	31	560	925	0,119
T3	3949	146	209	831	808	0,019
T6	1035	0	0	348	348	0
T7	959	0	0	264	241	-0,011
T12	994	0	0	528	505	-0,093
T1	1000	0	0	687	0	-0,246
T2	1994	0	0	1035	505	-0,487
T4	1994	691	0	1035	505	-0,825

Illustration 2: un esempio di D/F-table per il task T6. Per una spiegazione delle notazioni (i) #A, (ii) B<A, (iii) A>B, (iv) B<<<A, (v) A>>>B, (vi) A→B

Dato il modello di processo di Illustration1 si genera un log di workflow random con 1000 sequenze di eventi (23572 token di eventi). Come esempio Illustration2 mostra le metriche definite precedentemente per il task 6. Si noti che il T6 appartiene ad uno di due flussi di eventi concorrenti (la divisione-AND in T2). Si può notare dalla Illustration2 che (i) solo la frequenza di T6 e T10 sono uguali (#T6=#T10=1035), (ii) T6 non è mai preceduto direttamente da T10(B<A=0), (iii) T6 è spesso seguito direttamente da T10 (A>B=581), (iv) T6 è preceduto alcune volte da T10 (B<<<A=348), e (v) sempre preceduto da T10 (A>>>B=#T6=1035). Infine, (vi) c'è una forte relazione di causalità da T6 a T10 (0,803) e in certa misura a T5 (0,267). Comunque, T6 è di tanto in tanto preceduto direttamente da T5 (B<A=80). Nella prossima sessione useremo la D/F-table in combinazione con euristiche relativamente semplici per costruire un D/F-graph.

Induzione del D/F-graph

Nella sezione precedente osserviamo che le informazioni nella T6-D/F-table suggeriscono che il task T6 è la causa del task T10 in quanto la causalità tra T6 e T10 è alta, e T6 non è mai direttamente preceduto da T10 e frequentemente seguito (direttamente) da T10. La nostra prima regola euristica è in linea con tale osservazione:

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

La prima condizione ($A \rightarrow B \geq N$) usa il fattore di rumore N (valore di default 0,05). Se ci aspettiamo maggior rumore possiamo incrementare questo fattore. La prima condizione richiede una causalità positiva tra il task A e il task B maggiore del valore del fattore di rumore. La seconda condizione ($A>B \geq \sigma$) contiene un valore di soglia σ. Se sappiamo di avere un log di workflow che sia totalmente libero da rumore, allora ogni occorrenza di task protettore è informativo. Comunque, per proteggere il nostro processo di induzione contro le inferenze basate su rumore, solo le occorrenze dei task protettore sopra la soglia di frequenza σ sono abbastanza affidabili per il nostro processo di induzione. Per limitare il numero di parametri il valore di σ è calcolato automaticamente usando la seguente equazione: σ=1+Round(N*#L/#T). N è il fattore di rumore, #L è il numero di linee traccia nel log di workflow, e #T è il numero di elementi (task differenti) nell'insieme di nodi T. Nel nostro esempio di lavoro σ=1+Round(0,05*1000/13)=5. È chiaro ora che la seconda condizione richiede che la frequenza di A>B sia maggiore o uguale del valore soglia σ. Infine, la terza condizione afferma l'esigenza che la frequenza B<A sia maggiore o uguale di σ. Applicando queste semplici regole alla D/F-table basata sull'Illustration1 il log di workflow risulta nel D/F-graph di Illustration3. Se compariamo il D/F-graph di Illustration3 con la rete di Petri di Illustration1 si può notare come tutte le connessioni tra i nodi sono conformi con il modello di workflow sottostante

(tutte le connessioni corrette e non ci sono connessioni mancanti).

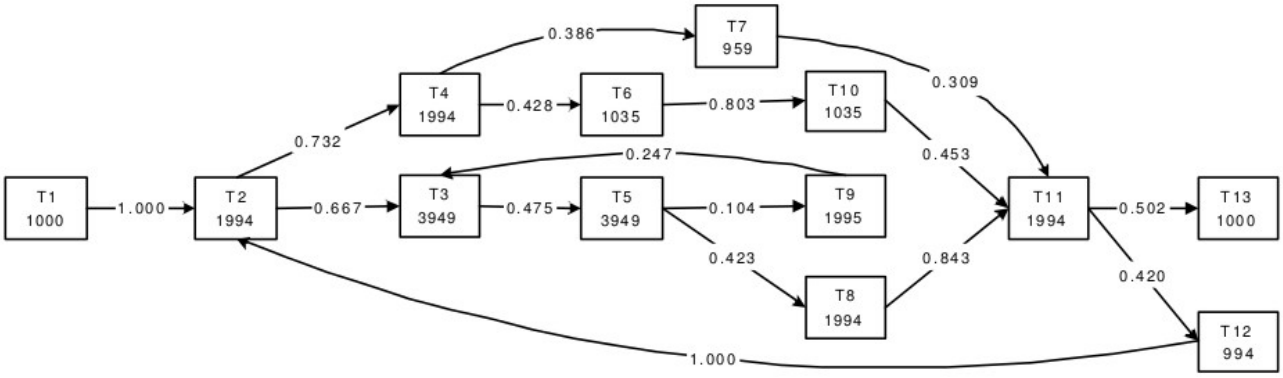

Illustration 3: Grafico D/F estratto automaticamente.

Comunque, le regole euristiche formulate precedentemente non riconoscono la ricorsione (nell'Illustration1, muovendo la freccia T9-P2 a P6) o piccoli cicli (nell'Illustration1, muovendo la freccia T9-P2 a P4). Dopo tutto, la ricorsione in T9 risulterà in pattern come T5, T4, T9, T9, T6, T9, T8. Dato che T9 sembra sia causa che conseguenza di T9 la frequenza della relazione di causalità T9→T9 è prossima allo 0. Comunque possiamo riconoscere la ricorsione in modo relativamente semplice: risulteranno delle frequenze alte e uguali per T9>T9 e T9<T9 e per T9>>>T9 e T9<<<T9. Il piccolo corto da T9 a T5 risulterà in pattern come T5, T4, T9, T5, T9, T6, T5, T8. Ancora T5 sembra sia causa che conseguenza di T9. Comunque, i piccoli cicli possono essere riconosciuti osservando le frequenze alte e quasi uguali per T5>T9 e T9<T5 e per T5>>>T9 e T9<<<T5.

In linea con queste osservazioni la nostra prima regola euristica è estesa con le seguenti:

IF((A→A≈0) AND (A<A+A>A>0,5*#A) AND (A<A-A>A≈0)) THEN <A,A>∈T

IF((A→B≈0) AND (A>B≥σ) AND (B<A≈A>B) AND (A>>>B≥0,4*#A) AND (B<<<A≈A>>>B)) THEN <A,B>∈T

Per prevenire la rottura le nostre euristiche provocata in caso di rumore, usiamo il simbolo di simile (≈) invece del simbolo di uguale (=). Ancora, il fattore di rumore N è usato per specificare cosa intendiamo per “simile”: x≈y se e solo se la differenza relativa tra loro è minore di N.

Generare la WF-net dal D/F-graph

Dato un log di workflow appare relativamente semplice trovare il D/F-graph corrispondente. Ma, i tipi di divisioni e unioni non sono ancora rappresentati nel D/F-graph. Inoltre (i) l'informazione nella D/F-table, (ii) la frequenza dei nodi nel D/F-graph, e (iii) le etichette di causalità contengono utili informazioni per indicare il tipo di un unione o di una divisione.

Per esempio, se dobbiamo individuare il tipo di divisione da A a B AND/OR C, possiamo vedere nella D/F-table i valori di B>C e B<C. Se A è una divisione-AND ci aspettiamo un valore positivo sia per B>C che per B<C (perché possono presentarsi i pattern B, C e B, C). Se è una divisione-OR i pattern B, C e C, B non si presenteranno.

Come esempio di (ii) guardiamo alla frequenza dei nodi nel D/F-graph di Illustration3. T2 è una divisione-AND (perché #T4=#T2 e non ci sono altri archi entranti per T4). T5 è una divisione-OR (perché #T5=#T8+#T9), e analogamente T4 e T11. L'unione nel nodo T11 è un po' più complicato: esso presenta una combinazione di unione-OR tra T7 e T10, combinata con un unione-AND con T8 (#T7+#T10=#T8=#T11).

Come esempio di (iii) comparare i valori dalla divisione-AND in T2 (0,667 e 0,732) e i valori dalla divisione-OR in t4 (0,428 e 0,386). Valori relativamente alti indicano una divisione-AND, mentre valori relativamente bassi indicano una divisione-OR. Osservazioni analoghe si possono fare per l'unione.

Lo pseudo codice per un algoritmo basato sulla prima euristica è data di seguito. Suppone, task A è direttamente preceduto dai task da B_1 a B_n. Da Set_1 a Set_n sono insiemi vuoti. Dopo aver applicato l'algoritmo tutti i task in relazione-OR sono collezionati in un insieme Set_i. Tutti gli insiemi non vuoti sono in relazione-AND.

```
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;
    UNTIL OK;
  END j DO;
END i DO;
```

Possiamo applicare un algoritmo analogo per ricostruire il tipo di un unione. Per esempio, applicando l'algoritmo all'unione-T11 risulterà in due insiemi {T7, T10} e {T8} o in formato preposizionale ((T7 OR T10) AND T8). Usando l'algoritmo precedente siamo stati capaci di ricostruire i tipi di divisione e unione che appaiono nel nostro esempio di lavoro e di ricostruire completamente la WF-net sottostante. Nella sezione successiva riporteremo i nostri risultati sperimentali e applicheremo le euristiche definite precedentemente su altri log di workflow, con e senza rumore.

Esperimenti

Per testare in nostro approccio usiamo una rappresentazione con reti di Petri di sei differenti modelli di workflow scelti a caso. La complessità di questi modelli varia dal comparabile con la complessità del nostro modello di lavoro di Illustration1 (13 task) a modelli con 16 task. Tutti i modelli contengono processi concorrenti e cicli. Per ogni modello abbiamo generato tre log di workflow random con 1000 sequenze di eventi: (i) un log di workflow senza rumore, (ii) uno con il 5% di rumore, (iii) ed uno con il 10% di rumore. Di seguito spieghiamo cosa intendiamo per rumore.
Per incorporare rumore nei nostri log di workflow definiamo quattro tipi differenti di operazioni che generano rumore: (i) cancellare la testa di una sequenza di eventi, (ii) cancellare la coda di una sequenza, (iii) cancellare una parte del corpo, e infine (iv) scambiare due eventi scelti a caso. Durante l'operazione di cancellazione si è cancellato da un minimo di un evento ad un massimo di un terzo della sequenza. Il primo passo nel generare un log di workflow con il 5% di rumore è un log di workflow random generato normalmente. Il passo successivo è la selezione random del 5% delle sequenze di eventi originali e l'applicazione su di esse di una delle quattro operazioni di generazione del rumore descritte precedentemente (ogni operazione di generazione del rumore con una probabilità uguale di ¼).
A causa della mancanza di spazio non è possibile descrivere tutti i modelli di workflow e i D/F-graph risultanti in dettaglio. Comunque, applicando il metodo precedente sui sei log di workflow liberi da rumore si ottengo sei D/F-graph perfetti (ossia tutte le connessioni sono corrette e non ci sono connessioni mancanti), e copie esatte delle WF-net sottostanti. Se aggiungiamo il 5% di rumore ai log di workflow, i D/F-graph e le WF-net risultanti sono ancora perfette. Tuttavia, se aggiungiamo il 10% di rumore ai log di workflow un D/F-graph è ancora perfetto, cinque D/F-graph contengono un errore. Tutti gli errori sono causati dal valore soglia basso σ=5 nella prima regola, che si ripercuote in una applicazione ingiustificata di questa regola. Se incrementiamo il valore del fattore di rumore ad un valore più alto (N=0,10), il valore di soglia calcolato automaticamente sale a 9 e tutti i cinque errori spariscono e non si presenta alcun nuovo errore.

Conclusione

In questo paper noi introduciamo (i) il conteso di processi di workflow e di process mining, (ii) alcuni preliminari includendo un linguaggio di modellazione per i processi di workflow, e (iii) una definizione di log di workflow. In seguito, presentiamo i dettagli dei tre passi della nostra tecnica di process mining: (i) costruzione della D/F-table, (ii) induzione di un D/F-graph dalla D/F-table, (iii) costruzione della WF-net dal D/F-table e D/F-graph.

Nella sessione sperimentale applichiamo la nostra tecnica a sei differenti modelli di workflow sound con circa 15 task. Tutti i modelli contengono processi concorrenti e cicli. Per ogni modello di workflow generiamo tre log di workflow random con 1000 sequenze di eventi: (i) senza rumore, (ii) col 5% di rumore, e (iii) col 10% di rumore. Usando la tecnica proposta siamo stati capaci di ricostruire D/F-graph e le WF-net corrette. I risultati sperimentali dell'esperimento con i log di workflow con rumore indicano che la nostra tecnica è robusta in caso di rumore.
Nonostante i risultati riportati ci sono molti lavori futuri da svolgere. Primo, i risultati riportati si basano su un numero limitato di risultati sperimentali; devono essere svolti più lavori sperimentali. Secondo, cercheremo di migliorare la qualità e le basi teoriche delle nostre euristiche. Possiamo per esempio dimostrare che le nostre euristiche sono applicabili con successo ai log provenienti da WF-net di libera scelta? Infine, estenderemo la nostra tecnica di estrazione al fine di allargare l'insieme di WF-net sottostanti che possono essere estratte con successo.