Francesco Andreuzzi - Discussione

Prima slide

- Grazie Prof. Casagrande
- Buongiorno
- Francesco Andreuzzi
- Il mio lavoro di tesi è incentrato sul tema della massima bisimulazione di grafi diretti
- Il problema è stato affrontato sia dal punto di vista teorico, nelle prime due sezioni della tesi, sia dal punto di vista pratico, con la produzione di un software per il calcolo della massima bisimulazione.

Grafi orientati

- Alla base del problema c'è il concetto di grafo
- Un grafo è una struttura matematica che consiste in un insieme di *nodi* e una relazione binaria sull'insieme dei nodi, detta *insieme degli archi*.
- Grazie ai grafi è possibile modellare sistemi di varia natura, come social network, reti fisiche o logistiche.
- Per questo motivo i grafi trovano numerose applicazioni pratiche.
- Se c'è un arco dal nodo a verso il nodo b, diciamo che b è *figlio* di a.

Bisimulazione

- Una relazione binaria sull'insieme dei nodi di un grafo è una bisimulazione se data una coppia di nodi a,b in relazione, ogni nodo figlio di a è in relazione con almeno un nodo figlio di b, e viceversa.
- Vediamo un esempio: la relazione B è una bisimulazione perchè il nodo d (figlio di b) è in relazione con il nodo f (figlio di c) e lo stesso vale per la coppia e,g. Quindi la condizione è soddisfata.
- Due nodi sono bisimili se esiste una bisimulazione per cui sono in relazione.
- Si può osservare che se due nodi sono bisimili si comportano in modo molto simile.

Massima bisimulazione

- Per un grafo esistono in generale più bisimulazioni. La *massima bisimulazione* è la bisimulazione che contiene tutte le bisimulazioni.
- Si può dimostrare che la massima bisimulazione è unica, che è una relazione di equivalenza.
- E' facile osservare che la massima bisimulazione coincide con la relazione binaria che mette in relazione coppie di nodi bisimili.

Applicazioni pratiche

- In questa slide è riportata la massima bisimulazione dell'esempio precedente. Chiaramente è una relazione di equivalenza, e divide l'insieme dei nodi in classi di equivalenza composte da nodi tutti bisimili.
- Esistono numerose applicazioni pratiche della massima bisimulazione, le più immediate sono la ricerca di nodi equivalenti e la minimizzazione del grafo per ridurre lo spazio occupato (visibile nella figura più a destra) ma sono state considerate anche applicazioni più avanzate in altri campi.

Algoritmi per il calcolo della massima bisimulazione

- Visto l'interesse pratico sono stati sviluppati diversi algoritmi per il calcolo della massima bisimulazione.
- Abbiamo l'algoritmo di Paige-Tarjan e l'algoritmo di Dovier-Piazza-Policriti.
- La differenza fondamentale sta nel fatto che l'algoritmo di Dovier-Piazza-Policriti rifinisce l'approcio di Paige-Tarjan utilizzando il *rango*, che può essere definito informalmente come una misura della distanza di un nodo dai pozzi del grafo.
- Si può dimostrare che due nodi bisimili hanno lo stesso rango, per cui possiamo cercare la bisimulazione in modo più oculato quali nodi hanno lo stesso rango.
- Infine abbiamo l'algoritmo incrementale di Saha, che consente di aggiornare la massima bisimulazione in seguito all'aggiunta di un arco, se conosciamo la massima bisimulazione del grafo prima della modifica.

BisPy

- L'analisi della correttezza e della complessità degli algoritmi è stata una parte molto importante del mio lavoro, in quanto mi ha consentito di comprendere a fondo il problema e di realizzare un'implementazione corretta.
- Gli algoritmi sono stati implementati in un pacchetto Python open source, la correttezza dei risultati è stata testata su diversi grafi di dimensioni varie.
- Nel riquadro ho messo un piccolo esempio di utilizzo.

Risultati sperimentali I

- Vediamo alcuni risultati sperimentali.
- In questo primo esperimento ho considerato alberi bilanciati di dimensione varia.
- Come previsto l'algoritmo di Dovier-Piazza-Policriti risulta asintoticamente più efficiente, dopo una prima fase in cui l'algoritmo di Paige-Tarjan risulta più performante visto che non ha bisogno di calcolare il rango.

Risultati sperimentali II

- Nel secondo esperimento ho confrontato il tempo di esecuzione dell'algoritmo di Paige-Tarjan e dell'algoritmo di Saha su grafi randomici, per aggiornare la bisimulazione massima dopo l'aggiunta di un arco randomico.
- Nel grafico sopra abbiamo il tempo medio di esecuzione per i due algoritmi.
- Ricordo che l'algoritmo di Saha prende in input anche la massima bisimulazione *vecchia*.
- Come si può vedere l'algoritmo incrementale risulta più performante in media.
- Guardiamo un ingrandimento di quanto succede nel riquadro verde, e quindi passiamo nel grafico sotto. Guardando i singoli esperimenti che compongono il campione, vediamo che non sempre l'algoritmo incrementale è il più performante.

Conclusione

- Lo sviluppo del pacchetto *BisPy* ha consentito di osservare il comportamento degli algoritmi trattati su numerosi tipi di grafo.
- Ritengo che il progetto meriti di essere sviluppato ulteriormente, per questo motivo ho individuato alcune possibili migliore e funzionalità che sarebbe interessante aggiungere.
- Sarebbe utile aggiungere la possibilità di aggiornare la bisimulazione in modo incrementale in seguito alla rimozione di un arco (è una variante già studiata dell'algoritmo di Saha).

- Inoltre sarebbe utile scrivere in Cython alcuni pezzi particolarmente pesanti del codice per migliorare le performance.

Ho finito, grazie per l'attenzione.