Paige_Tarjan

L'articolo presenta algoritmi risolutivi per tre problemi differenti.

L'algoritmo presentato migliora quello presentato da Hopcroft, pur utilizzando una strategia simile (ma migliorata).

$split(S, Q)$

Sia $S \subset U$ con U insieme finito, sia R una relazione binaria su U (cioè $R \subset U \times U$), e sia Q una partizione di U.

$$split(S, Q) = \begin{cases} B & \text{se B è un blocco di Q stabile rispetto a S (rispetto a R)} \\ \{B \cap R^{-1}(S), B - R^{-1}(S)\} & \text{altrimenti} \end{cases}$$

Diremo che S è uno splitter di Q se $split(S, Q) \neq Q$.

Proprietà di split

1. S è uno splitter per $Q \iff split(S, Q) \neq Q \iff Q$ è instabile rispetto a S
2. Se Q' è una rifinitura di Q, e Q è stabile rispetto a $S \implies Q'$ è stabile rispetto a S
3. Se Q è stabile rispetto a $S_1, S_2 \implies Q$ è stabile rispetto a $S_1 \cup S_2$
4. Monotonia: se Q' è una rifinitura di $Q \implies split(S, Q')$ è una rifinitura di $split(S, Q)$
5. Commutatività: $split(S, split(Q, P)) = split(Q, split(S, P))$

Algoritmo "naive"

Finchè è possibile trovare un insieme S tale che $S = \alpha \cup \beta$ con α, β blocchi di Q, e tale che S è uno splitter di Q, sosituisci Q con $split(S, Q)$.

Non è necessario usare come splitter unioni di blocchi della partizione attuale, ma questo consente di sviluppare l'algoritmo più veloce.

Algoritmo "fast"

Innanzitutto si preprocessa la partizione iniziale P dell'insieme U, sostituendo ogni blocco B con

$$B' = B \cap E^{-1}(U), \qquad B'' = B - E^{-1}(U)$$

E' evidente che tutti i blocchi B'' contengono tutti gli elementi $x : \nexists y \in U : E(x, y)$. Di conseguenza questi blocchi non verranno mai toccati da $split$, in quanto sono già stabili rispetto a qualsiasi sottoinsieme di U: prendendo un $S \subset U$ si ha che $B'' \cap E^{-1}(S) = \emptyset$ per qualsiasi B''

.
Allora l'insieme dei B' è una partizione di $E^{-1}(U)$, e possiamo usare l'algoritmo soltanto su questi blocchi. Non possiamo unire i blocchi B'' perchè violeremmo la condizione che la partizione risultante deve essere una rifinitura della partizione iniziale.

Il miglioramente consiste nel mantenere due partizioni X, Q di U. Q è sempre una rifinitura di X, e Q è stabile rispetto ad ogni blocco di X. Inizialmente $Q = P, X = U$ (cioè X contiene un unico blocco). Si ripete il seguente algoritmo finchè non si ottiene $Q = X$

1. Trova un blocco $S \in X : S \notin Q$
2. Trova un blocco $B \in Q : B \subset S \wedge |B| \leq |S|/2$
3. Rimpiazza S in X con $B, S - B$
4. Rimpiazza B in Q con $split(S - B, split(B, Q))$

L'obiettivo è scegliere gli splitter in modo più intelligente (scegliendoli casualmente, anche con il pre-processamente l'algoritmo sarebbe $O(mn)$).

Osservazione

Se P è formato da un unico blocco P stesso è il suo coarsest stable refinement.

Osservazione

Ogni blocco B usato come splitter ha cardinalità al più dimezzata rispetto al B del passaggio precedente.

Caso funzionale (da Hopcroft)

Suppongo che

$$\forall x \in U |E(\{x\})| = 1, \text{ cioè } \forall x \ \exists! y \in U : E(x, y) \qquad (1)$$

Sia Q una partizione di U. Sia $S = \cup_{i=1}^{n} b_i$ con $b_i \in Q$. Suppongo Q stabile rispetto a S. Sia $B \subset S$.
Allora

$$split(B, Q) \text{ è stabile rispetto a } S - B$$

Infatti, sia $B_1 \in Q$.

- se B_1 era un blocco di Q già stabile rispetto a B (quindi B_1 non è cambiato), allora
  - $B_1 \subset E^{-1}(B) \implies B_1 \cap E^{-1}(S - B) = \emptyset$, perchè $B \cap S - B = \emptyset$ e per l'ipotesi (1)
  - $B_1 \cap E^{-1}(B) = \emptyset$. Ricordando che B_1 era già un blocco di Q stabile rispetto a S
    - Non può essere $B_1 \subset E^{-1}(S)$, perchè $B_1 \cap E^{-1}(B) = \emptyset$ e $B \subset S$
    - $B_1 \cap E^{-1}(S) = \emptyset \implies B_1 \cap E^{-1}(S - B) = \emptyset$

- se B_1 è stato generato da uno split
  - $B_1 = \widetilde{B} \cap E^{-1}(B)$ per qualche blocco $\widetilde{B} \in Q \implies$ deve essere chiaramente $B_1 \cap E^{-1}(S - B) = \emptyset$ per la (1)
  - $B_1 = \widetilde{B} - E^{-1}(B)$ per qualche blocco $\widetilde{B} \in Q$
    - $\widetilde{B} \subset E^{-1}(S) \implies \widetilde{B} - E^{-1}(B) \subset E^{-1}(S - B)$
    - $\widetilde{B} \cap E^{-1}(S) = \emptyset \implies \widetilde{B} \cap E^{-1}(B) = \emptyset \implies B_1 = \widetilde{B} - E^{-1}(B) = \widetilde{B}$, ma questo caso è già stato trattato.

Lemma 3

Suppongo Q stabile rispetto ad un insieme S (che nel caso dell'algoritmo è uno dei blocchi di X, e Q è stabile rispetto a tutti i blocchi di X). Sia $B \subset S$. Allora per l'operazione $split(S - B, split(B, Q))$ valgono i seguenti risultati

1. Un blocco $D \in Q$ viene rifinito rispetto a $B \iff D \cap E^{-1}(B) \neq \emptyset \wedge D - E^{-1}(B) \neq \emptyset$ (per definizione di $split$).
   In questo caso
   $$D \to \begin{cases} D_1 = D \cap E^{-1}(B) \\ D_2 = D - E^{-1}(B) = D - D_1 \end{cases}$$
2. Un blocco di tipo $D_1 \in split(B, Q)$ viene rifinito rispetto a $S - B \iff D_1 \cap E^{-1}(S - B) \neq \emptyset \wedge D_1 - E^{-1}(S - B) \neq \emptyset$ (per definizione di $split$).
   In questo caso
   $$D_1 \to \begin{cases} D_{11} = D_1 \cap E^{-1}(S - B) \\ D_{12} = D_1 - E^{-1}(S - B) = D_1 - D_{11} \end{cases}$$
3. I blocchi di tipo D_2 non vengono toccati da $split(S - B, D_2)$.
   Poichè D è stabile rispetto a S, $D \subset E^{-1}(S) \vee D \cap E^{-1}(S) = \emptyset$
   Se D viene diviso in $D_1, D_2 \implies D \cap E^{-1}(B) \neq \emptyset$, con $B \subset S$, quindi il secondo caso è impossibile.
   Allora $D_2 \cap E^{-1}(B) = \emptyset$ per costruzione, quindi $D_2 \subset E^{-1}(S - B)$ perchè $D_2 \subset D \subset E^{-1}(S)$.
4. E' sufficiente fare un disegno per verificare che
   $$\begin{cases} D_{11} = D_1 - (E^{-1}(B) - E^{-1}(S - B)) \\ D_{12} = D_1 \cap (E^{-1}(B) - E^{-1}(S - B)) \end{cases}$$

Strutture dati

Record	$y \in U$	xEy	$B \in Q$	$S \in X$

Record	$y \in U$	xEy	$B \in Q$	$S \in X$
Pointer 1	$[(a,y),(b,y),...]$	x	$\|B\|$	$< B_i > : B_i \subset S$
Pointer 2	$B \in Q : y \in B$	$count(x,S)$ (con $S \in X : y \in S$)	$< a,b,.. > \subset B$	
Pointer 3	$count(y,S) = \|S \cap E(\{y\})\|$ (con $S \in X : y \in S$)		$S \in X : B \subset S$	

Dove $< .. >$ denota una doubly linked list.
Queste strutture richiedono uno spazio $O(m)$ (supponendo il pre-processamento, quindi $m \geq n$).