4.3. Disegno ortogonale interattivo: alcuni scenari e algoritmi

Riferimenti: [KW98] pp. 228-240 escluse pp. 233 (da 'In the following…'), 234, 236; [Dal99] pp. 218-235.

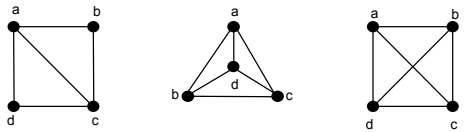
4.3.1. Mantenere la mappa mentale

Dato un grafo G, ogni algoritmo di visualizzazione fin qui presentato prende G in input e ne produce una rappresentazione grafica. Un sistema interattivo potrebbe permettere all'utente di editare il grafo manualmente, inserendo o cancellando nodi e archi, o settando nuovi vincoli (per esempio il fatto che un certo nodo debba stare sopra un altro). Può anche accadere che il grafo rappresentato sia intrinsecamente dinamico, come ad esempio una porzione del grafo del web, i cui nodi possono essere tra loro connessi o no al variare del tempo, con conseguente variazione del disegno.

Se viene inserito o cancellato un nodo di G, allora abbiamo un nuovo grafo, e l'approccio più semplice consiste nell'applicare nuovamente lo stesso algoritmo di visualizzazione, ed avere in output una nuova rappresentazione, che può essere anche molto diversa dalla precedente. Tutto ciò è uno spreco di tempo e di risorse da due punti di vista:

- è inefficiente, poiché viene eseguito ogni volta l'algoritmo sull'intero grafo, con dispendio di tempo;
- l'utente ha probabilmente perso del tempo a lavorare sul primo disegno e non vorrebbe trovarsi di fronte ad un disegno diverso (v. figura 4.37 per un'idea intuitiva della mappa mentale).

Figura 4.37.



Quando un utente guarda un disegno, ne apprende la struttura, familiarizza con esso navigandovi attraverso e comprendendone il significato. Questo sforzo di 'familiarizzare' con un disegno è stato chiamato *costruzione di una mappa mentale* [Eal91]. Nel caso del disegno dinamico, quando il disegno cambia nel tempo, l'utente deve ripetutamente modificare la sua mappa mentale, e sarebbe vantaggioso minimizzare questo sforzo dell'utente. Per mantenere il più possibile la mappa mentale dell'utente, si può procedere in due direzioni: se si sceglie di modificare molto il disegno di passo in passo, questa modifica dovrebbe essere mostrata tramite un'*animazione*, in modo che l'utente abbia l'opportunità di vedere dove i vari oggetti vengano spostati e avere meno problemi nel modificare la propria mappa mentale. L'altro approccio è quello di modificare il disegno il meno possibile; tuttavia ciò è spesso in conflitto con i tradizionali criteri estetici (ad esempio, la minimizzazione del numero di incroci, la equa distribuzione dei nodi sulla pagina, ecc.), ed è quindi necessario giungere ad un compromesso.

Ovviamente, l'animazione e la minimizzazione dei cambiamenti non sono in conflitto tra loro, ma anzi sono complementari per il raggiungimento del risultato migliore, infatti prima si può calcolare un nuovo disegno che minimizzi i cambiamenti, e poi si può visualizzarlo utilizzando un'animazione che parte dal disegno vecchio e lo trasforma nel nuovo.

Poiché l'animazione non presenta problematiche particolari, non sarà discussa qui, mentre sarà dettagliata maggiormente la problematica di costruire un nuovo disegno che minimizzi i

cambiamenti. Prima di procedere, è necessario chiarire il significiato, pure intuitivo, della frase "minimizzare i cambiamenti in modo che la mappa mentale venga preservata", a volte riassunta nella frase "mantenere la stabilità dinamica".

Ci sono molti modi di intendere il mantenimento della mappa mentale. Uno piuttosto stringente consiste nel non permettere alcun cambiamento al disegno corrente, o meglio ai vertici ed archi già posizionati che non sono coinvolti direttamente dalla modifica del grafo. In questo modo, l'algoritmo di disegno deve solo decidere dove posizionare i nuovi oggetti in modo da minimizzare il più possibile i criteri estetici scelti. Ovviamente, questo approccio mantiene perfettamente la mappa mentale ma non necessariamente la consistenza, cioè l'aderenza alle regole di stile prescelte (ad esempio la "forma" ad albero [N96]); inoltre, spesso il risultato sarà decisamente poco gradevole rispetto agli altri criteri estetici (ad esempio, è inevitabile inserire un gran numero di incroci).

Algoritmi che seguono questo approccio sono dovuti a Fößmeier [F97] e Papakostas e Tollis [PT98].

Questa resrizione può essere indebolita leggermente permettendo un certo aggiustamento del disegno nelle "vicinanze" del cambiamento del grafo. Ad esempio, Böringer a Paulisch [BP90] propongono una definizione di "vicinanza" ad un cambiamento come l'insieme di tutti i vertici direttamente affetti dal cambiamento più tutti i vertici a distanza (misurata sugli archi già rappresentati) minore di una certa soglia predefinita, pensando che l'utente possa tollerare cambiamenti all'interno di una piccola porzione di grafo vicina al cambiamento vero e proprio, mentre preferisce che il resto del grafo rimanga invariato. Si osservi che restringere l'insieme di vertici che può essere spostato è particolarmente utile nel caso di grafi molto grandi.

Invece di cercare di fissare le posizioni in modo categorico, alcuni autori cercano di definire alcune misure di simiglianza (o meglio di dissimiglianza) tra rappresentazioni, per catturare la quantità di sforzo, da parte dell'utente, di ricostruire la mappa mentale dopo il cambiamento. Lo scopo è quello di misurare il più accuratamente possibile quanto le sembianze di un disegno varino con gli aggiustamenti dovuti ad un cambiamento del grafo. Un buon algoritmo dovrebbe costruire un nuovo disegno che sia un buon compromesso tra ottimizzazione dei criteri estetici e somiglianza del disegno corrente con il precedente, anche detta stabilità dinamica.

Questo approccio presenta il vantaggio di permettere cambiamenti arbitrari al disegno se necessario; d'altra parte, potrebbe essere difficile trovare un buon *trade off* tra criteri estetici tradizionali e stabilità dinamica.

Sono stati proposti molti modelli in letteratura per catturare il significato di mappa mentale e di stabilità dinamica, che possono essere raggruppati in alcune categorie, secondo l'idea generale che li orienta. Alcuni di essi sono:

Posizione assoluta dei vertici: un modo di misurare la simiglianza più o meno ovvio è quello di valutare la somma delle distanze (euclidee o di altro tipo) di ciascun nodo nel disegno corrente dallo sua posizione nel disegno precedente. Un problema immediato di questa metrica è che operazioni come la traslazione, la rotazione o la scalatura potrebbero portare a enormi valori di dissimiglianza indicando quindi una mutazione grandissima del disegno, mentre l'utente capirebbe immediatamente di quale cambiamento si tratta e lo seguirebbe senza problemi. Per evitare questo problema, alcuni autori, prima di valutare la metrica, cercano di allineare i due disegni applicando un algoritmo di *point set matching*, tenendo quindi conto di scalature, traslazioni e rotazioni.

Infine, è da notare che questa metrica considera solo il caso in cui i vertici siano rappresentati da punti. Per diminuire questo problema Bridgemann e Tamassia [BT98] propongono che – quando la dimensione dei vertici possa essere modificata – si guardi ai 4 angoli del più piccolo rettangolo circoscritto.

Ordinamento ortogonale/Posizione relativa dei vertici: si può pensare che preservare l'ordinamento relativo dei vertici sia più importante che preservare le loro posizioni assolute: in [Eal91] si propone di mantenere l'ordinamento ortogonale, cioè l'ordinamento dei vertici proiettati su ciascun asse cartesiano. Ispirati da quest'idea, gli autori di [BT98] propongono di valutare gli angoli tra i segmenti costruiti tra tutte le coppie di vertici nel vecchio e nel nuovo disegno, infatti la variazione di un angolo è qualcosa di più graduale rispetto alla variazione di un ordinamento. Il lato positivo di questa metrica è che riflette l'intuizione che a vertici molto distanti possano essere permessi movimenti più ampi che a vertici vicini (entrambi corrispondenti ad uno stesso angolo).

Prossimità: l'idea alla base di questa metrica è che gli "agglomerati" dovrebbero essere rispettati, cioè vertici che siano vicini in un vecchio disegno dovrebbero rimanere vicini anche nel successivo. Il vantaggio di misurare il mantenimento degli agglomerati è che tale metrica cattura il fatto che se un sottografo si sposta nel disegno la distanza sarà inferiore a quella che si avrebbe nel caso in cui un vertice di un agglomerato venga allontanato.

Instradamento degli archi: North [N96] osserva che la posizione dei vertici è più importante della posizione degli archi perché i vertici vengono spesso ricordati per dove si trovano, mentre gli archi per cosa collegano. Tuttavia, nel caso del disegno ortogonale, anche l'instradamento degli archi andrebbe considerato come una possibile metrica.

Anzianità: tutto quanto detto fin ora riguarda una singola modifica di un grafo (si misura la distanza tra il disegno corrente ed il precedente). Tuttavia, spesso si ha a che fare con una sequenza di cambiamenti, che producono una storia. In tal caso, un vertice che sia stato spostato di recente è un miglior candidato ad essere nuovamente spostato rispetto ad altri, cioè l'"anzianità" di un vertice in una stessa posizione va tenuta in conto quando si prendano delle decisioni.

Raggruppamento dei cambiamenti: quando una sequenza di cambiamenti sia nota in anticipo, potrebbe essere conveniente raggruppare più cambiamenti del grafo in un unico cambiamento del disegno piuttosto che mostrare all'utente tutti i passaggi: questo facilita ovviamente il mantenimento della mappa mentale.

Capire quale, tra tutti i modelli proposti, sia il migliore, è ancora un problema aperto. Bridgemann e Tamassia [BT98] e Lyons et al. [Lal98] hanno confrontato molte metriche proposte per il disegno ortogonale, ed hanno concluso che la maggior parte di esse funzionano bene, nel senso che misurano in modo crescente l'aumentare dei cambiamenti della mappa mentale. Per confrontare le varie metriche, si è operata una normalizzazione dividendo per il massimo valore possibile per quella metrica ed ottenendo, per tutte le metriche, valori compresi tra 0 e 1.

4.3.2. La problematica del disegno interattivo

L'area del *disegno interattivo* si occupa di studiare tecniche che gestiscono efficientemente – da entrambi i punti di vista appena presentati – le piccole variazioni del grafo in input. Essa si contrappone all'aproccio statico che assume che tanto l'intero grafo quanto i vincoli del disegno siano noti fin dall'inizio e non vengano modificati nel tempo.

Nel seguito faremo l'assunzione che, dopo ogni modifica, il massimo grado del grafo si mantenga al più 4 perché ci focalizziamo sul disegno interattivo ortogonale.

Il software che supporta l'interazione dell'utente dovrebbe essere in grado di fare un disegno di un grafo seguendo dei criteri standard (ortogonale, rettilineo, …) e dare all'utente la possibilità di interagire con il disegno nei seguenti modi:

- inserire un arco tra due specificati nodi;
- inserire un nodo e gli archi ad esso incidenti;
- cancellare archi o nodi;
- spostare un nodo.

La cancellazione è un'operazione relativamente semplice, e quindi non sarà qui presa in considerazione. Per quello che riguarda l'inserimento di archi, Miriyala ed altri [Mal93] hanno proposto un'euristica per disegnare gli archi senza modificare il disegno esistente, separando il disegno in regioni ed usando una variante dell'algoritmo di Dijkstra per determinare la sequenza di regioni attraverso cui l'arco deve passare per ottenere un costo minimo in termini di lunghezza di archi, incroci e svolte. Pertanto, l'operazione di cui ci si occuperà qui è l'inserimento di vertici.

Ci sono vari fattori che influenzano le decisioni che il sistema interattivo deve prendere al momento della richiesta dell'utente, prima che venga visualizzato il disegno successivo. Alcuni di essi sono:

- quanto controllo ha l'utente sulla posizione di un nodo appena inserito;
- quanto controllo ha l'utente sul percorso di un arco appena inserito;
- quanto possono essere diversi due disegni consecutivi.

Basandosi su questi fattori, si può proporre una serie di possibili scenari:

Scenario full-control: L'utente ha pieno controllo sulla posizione dei nuovi nodi, nel senso che può fornire il range delle sue coordinate, o le coordinate esatte; gli archi possono essere disegnati dall'utente o, a scelta, dal sistema; questo scenario dà piena libertà all'utente ma il disegno risultante non è – in generale – ben ottimizzato.

Scenario draw-from-scratch: Ogni volta che l'utente fa una nuova richiesta, il sistema ridisegna completamente il nuovo grafo usando un algoritmo classico di disegno. A parte il fatto che questo scenario è, in generale, computazionalmente dispendioso, due disegni consecutivi possono essere completamente differenti, quindi questo scenario non mantiene la mappa mentale dell'utente, sebbene sia quello che raggiunge i risultati migliori dal punto di vista dell'ottimizzazione del disegno.

Scenario relative-coordinates: La forma generale del disegno rimane invariata, ma alcune coordinate possono cambiare di una costante, a causa dell'inserimento o della cancellazione di alcuni elementi del grafo. Poiché l'ordine relativo delle coordinate non cambia, la mappa mentale dell'utente viene mantenuta.

Scenario no-change: Le coordinate degli oggetti già posizionati non possono essere modificate affatto; ne segue che la mappa mentale dell'utente è fortemente conservata. Tuttavia, si osserva sperimentalmente, che la qualità del disegno - all'aumentare del numero di vertici - tende a peggiorare; sono pertanto necessari dei passi di ricostruzione del disegno (con perdita totale della mappa mentale), se si vuole mantenere una qualità del disegno discreta.

E' facile vedere che gli scenari per cui ha realmente senso parlare di disegno interattivo sono gli ultimi due, ed è di questi che ci occuperemo nel seguito.

Nelle successive sottosezioni, chiameremo *grado locale* del nodo v, inserito al tempo t, il grado del nodo v al momento del suo inserimento. In generale, se l'utente ha necessità di inserire un nodo con grado locale pari a zero, tale nodo non sarà inserito finché, tramite l'inserimento di altri archi a lui incidenti ma prima assenti, il suo grado locale non sarà aumentato; ne segue che trattiamo l'inserimento dei soli nodo con grado locale pari a 1, 2, 3 o 4. Equivalentemente, assumiamo che l'inserimento di un nuovo nodo del grafo non aumenti il numero di componenti connesse; l'unica eccezione riguarda l'inserimento del primo nodo nel grafo vuoto.

Definiamo *direzioni libere* al tempo *t* di un nodo *v* presente nel disegno, l'insieme delle direzioni nell'insieme *{alto, basso, sinsitra, destra}* che non siano già impegnate da un arco incidente a *v*, cioè l'insieme delle direzioni delle linee della griglia uscenti da *v* che non siano già coperte da un arco incidente a *v*.

Denotiamo con *n(t)* ed *m(t)* il numero di nodi e il numero di archi inseriti a tempo *t*, e con $n_i(t)$, *i=1, 2, 3, 4,* il numero di nodi di grado locale *i* inseriti al tempo *t*.

4.3.3. Scenario Relative-Coordinates

In questo scenario, ogni volta che un nodo deve essere inserito nel disegno corrente, il sistema prende una decisione su dove posizionare il nuovo nodo e gli archi ad esso incidenti verificando tutte le possibili posizioni assumibili dal nodo in considerazione (in base ai nodi adiacenti).

Nel disegno possono, ovviamente, essere inserite nuove righe o nuove colonne per consentire l'inserimento del nuovo nodo e dei suoi arhi.

Le coordinate del nuovo nodo *v,* così come il posizionamento delle nuove righe e colonne, dipendono dai seguenti fattori:

- il grado locale del nodo *v*;
- per ogni nodo *u* adiacente a *v*, quali direzioni intorno ad *u* possano essere utilizzate
- se l'instradamento degli archi possa essere fatto utilizzando segmenti di righe e colonne già inserite
- i criteri estetici da ottimizzare.

Lo scopo è quello di minimizzare il numero di nuove righe o colonne inserite nel disegno, tenendo basso il numero di svolte.

Ci sono molti possibili casi che si possono verificare, dipendentemente dalle possibili combinazioni, ed è relativamente semplice enumerarli. Nel seguito, dato un nodo *v* da inserire al tempo *t* con grado locale $ld_t(v)$, distinguiamo i seguenti 4 grandi casi:

1. $ld_t(v)=1$, l'arco utilizza una direzione lasciata libera nell'unico nodo *u* adiacente a *v* nel disegno già esistente, ed è previsto l'inserimento o di una nuova riga o di una nuova colonna, a seconda della direzione prescelta; non vengono inserite svolte (figura 4.38);

Figura 4.38.

2. $ld_t(v)=2$, si possono verificare diversi casi a seconda delle direzioni libere nei nodi adiacenti a *v* nel disegno già esistente. E' previsto l'inserimento di 0 righe/colonne e 0 svolte nel caso migliore (figura 4.39.a), e 3 tra righe e colonne e 3 svolte nel caso peggiore (figura 4.39.b);

3. $ld_t(v)=3$, anche qui il numero di righe e colonne e il numero di svolte può variare da caso a caso; è previsto l'inserimento di 4 tra righe e colonne e 5 svolte nel caso peggiore (v. figura 4.40.a);

4. $ld_t(v)=4$, è previsto l'inserimento di al più 6 tra righe e colonne e 8 svolte nel caso peggiore (v. figura 4.40.b);

Figura 4.39.

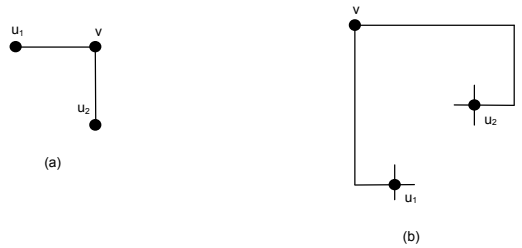
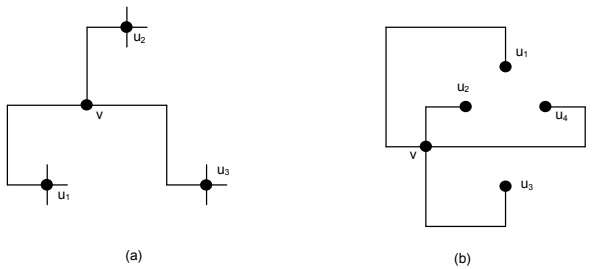


Figura 4.40.



Teorema. *Sia G un grafo di massimo grado 4 disegnato sotto lo scenario a coordinate relative; il disegno di G ha le seguenti caratteristiche:*

- *è ortogonale su griglia;*
- *un arco ha al massimo 3 svolte;*
- *il numero di svolte totali è al più 3n(t)+3;*
- *l'area è al più* $(9/4)n^2(t)$.

Inoltre per ogni operazione, le coordinate di ogni nodo sono spostate di al massimo 6 unità lungo l'asse delle x e delle y.

Dimostrazione. Le prime due affermazioni seguono direttamente dall'algoritmo.

Anche il fatto che le coordinate vengano spostate di al massimo 6 unitàderiva direttamente dal fatto che l'inserimento di ciascun nodo aggiunge al più 6 tra righe e colonne.

Per quanto riguarda la terza affermazione (numero di svolte), si consideri che il primo nodo inserito nel disegno ha grado locale pari a zero, per cui si ha:

$$n_1(t)+n_2(t)+n_3(t)+n_4(t) = n(t)-1 \qquad \text{(a)}$$

Analizzando i casi definiti dall'algoritmo, detto *B(t)* il numero di svolte inserite al tempo *t*, risulta:

$$B(t) \le 0n_1(t)+3n_2(t)+5n_3(t)+8n_4(t) = 3n_2(t)+5n_3(t)+8n_4(t) \qquad \text{(b)}$$

Ora, osservando che l'inserimento di un nodo con grado locale i introduce i archi nel disegno, risulta:

$$m(t) = n_1(t)+2n_2(t)+3n_3(t)+4n_4(t) \leq 2n(t) \quad \text{(c)}$$

Sottraendo l'equazione (a) dalla disequazione (c), e moltiplicando per 3, si ottiene:

$$3n_2(t)+6n_3(t)+9n_4(t) \leq 3n(t)+3 \quad \text{(d)}$$

Da (b) e (d) si deduce:

$$B(t) \leq 3n_2(t)+5n_3(t)+8n_4(t) \leq 3n_2(t)+6n_3(t)+9n_4(t) \leq 3n(t)+3$$

Con un'analisi più complessa si può dimostrare il risultato leggermente migliore $B(t) \leq 3n(t)-1$.

Rimane infine da dimostrare la quarta affermazione. Siano $h(t)$ e $w(t)$ rispettivamente l'altezza e la larghezza massima del disegno al tempo t. Intanto si osservi che, a seconda dei vari casi, l'introduzione di un nuovo nodo, può inserire un certo numero di righe, oppure un certo numero dicolonne, oppure entrambe. Poiché risulta difficile distinguere tra casi in cui si aggiungono nuove righe e quelli in cui si aggiungono nuove colonne, si preferisce qui valutare l'aggiunta di una nuova "striscia" di disegno, sia essa una riga oppure una colonna. Per questo valuteremo la somma $h(t)$ e $w(t)$.

Dai casi peggiori dell'algoritmo risulta:

$$h(t)+w(t)=n_1(t)+3n_2(t)+4n_3(t)+6n_4(t) \quad \text{(e)}$$

Moltiplicando la (c) per 3/2 si ottiene:

$$3/2\ n_1(t)+3n_2(t)+9/2\ n_3(t)+6n_4(t) \leq 3n(t)$$

Da quest'ultima disuguaglianza e dalla (e) si ottiene che $h(t)+w(t) \leq 3n(t)$.

Poiché l'area $A(t)$ è massima per $h(t)=w(t) \leq 3/2\ n(t)$, segue che $A(t) \leq 9/4\ n^2(t)$. **CVD**

In questo scenario non viene garantita la planarità, cioè il disegno può presentare degli incroci anche se il grafo finale è planare. Inoltre, tale scenario mantiene la forma del disegno dopo un aggiornamento.

Prima di concludere, osserviamo che non abbiamo qui fatto alcun cenno alla complessità computazionale e alle strutture dati necessarie per mantenere in memoria il disegno. Si fa presente che, con l'utilizzo di stutture dati piuttosto sofisticate ed un'analisi non banale, si può dimostrare che la complessità si mantiene lineare.

4.3.4. Scenario No-Change

In questo scenario il sistema non modifica *mai* le posizioni dei vertici e delle svolte dal disegno corrente al successivo: esso semplicemente incrementa il disegno aggiungendovi nuovi oggetti (nodi o archi). Come lo scenario precedente, anche questo produce un disegno ortogonale sotto l'assunzione che il massimo grado di ogni nodo in qualsiasi tempo non sia superiore a 4. Assumiamo inoltre di costruire un grafo in modo tale che esso si mantenga sempre connesso.

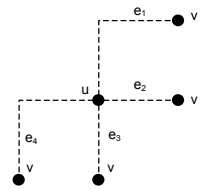
La costruzione del disegno avviene per passi: ad ogni passo si inserisce un nodo la cui posizione non cambia nei passi successivi.

Il nodo v_i che viene inserito al generico passo, e viene connesso a nodi appartenenti al grafo corrente G_i.

Sia $u \in G_i$, i modi per connettere v_i ad u sono quattro (vedi fig. 4.41), dove gli archi e_1, e_2, e_3 ed e_4 sono detti *percorsi* di u: e_2 ed e_3 sono detti percorsi *diretti* perché senza svolte, mentre e_1 ed e_4 sono detti *non diretti*. Inoltre, il nodo u ha al più due percorsi alla sua destra, attraverso e_1 ed e_2 , ed al più due percorsi verso il basso, attraverso e_3 ed e_4 . Più in dettaglio, u ha un percorso a destra se e solo se nessun arco del grafo corrente usa la porzione di riga alla destra di u corrispondente ad e_2 o la porzione di colonna sopra u corrispondente ad e_1 . Nel seguito, per mettere in evidenza la differenza tra archi già tracciati ed archi da tracciare, quando un nodo x viene inserito nel grafo corrente G_i, le direzioni libere restanti sono dette *percorsi*. Ad esempio, se x è connesso con un solo

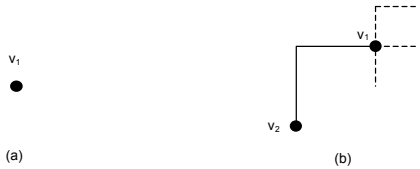
nodo $y \in G_i$, nei passi successivi dell'algoritmo, x ha tre percorsi incidenti (due diretti ed uno non diretto). Poiché l'algoritmo ha l'intento di minimizzare il numero di svolte e l'area, quando si sceglie un percorso da attribuire ad un arco si preferisce sfruttare le direzioni in alto e a sinistra, possibilmente con archi rettilinei.

Figura 4.41.



Si consideri ora il seguente algoritmo. Il primo passo consiste nel posizionare il nodo v_1 (vedi fig 4.42.a) ed il nodo v_2 usando un percorso non diretto (vedi fig. 4.42.b). Come si può vedere, ad entrambi i nodi rimane da utilizzare un unico percorso non diretto, ed essi introducono una svolta. Inoltre, il disegno costituito dai soli nodi v_1 e v_2 e dall'arco che li congiunge occupa due righe e due colonne.

Figura 4.42.



Sia v_i il nodo da inserire nel grafo corrente e u il generico nodo già disegnato a cui collegarlo. E' immediato notare che in ognuno dei casi, il maggior numero di svolte, di righe e di colonne viene introdotto quando il nodo u ha libero soltanto un percorso non diretto e quando non si possano riutilizzare le righe e le colonne dei passi precedenti. Questa situazione è il caso pessimo per l'inserimento.

Come nello scenario precedente, dipendentemente dal grado locale di v, possono verificarsi quattro casi:

1. $ld(v_i)=1$ (vedi fig. 4.43), ci sono quattro modi per inserire v_i (nella figura sono indicati quelli che usano gli archi diretti), che occupano sempre o la direzione sinistra o quella in alto di v_i , perciò si ha sempre che a v_i rimangono 2 percorsi diretti ed uno non diretto. Nel caso pessimo, con l'inserimento di v_i si introduce una svolta e 2 tra righe e colonne.
2. $ld(v_i)=2$ (vedi fig. 4.44), come è facile provare, si possono verificare quattro casi in base alla combinazione dei quattro percorsi dei nodi u_1 e u_2 ai quali v_i dovrà essere connesso. In ogni caso, sfruttiamo entrambi i percorsi non diretti di v_i. Nel caso pessimo introduciamo 2 svolte e 2 tra righe e colonne.

Figura 4.43.

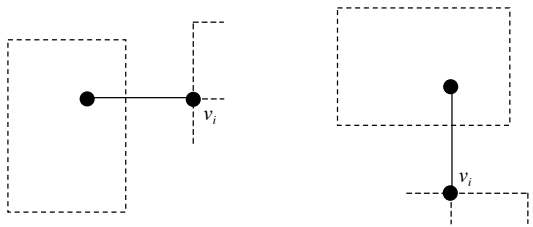
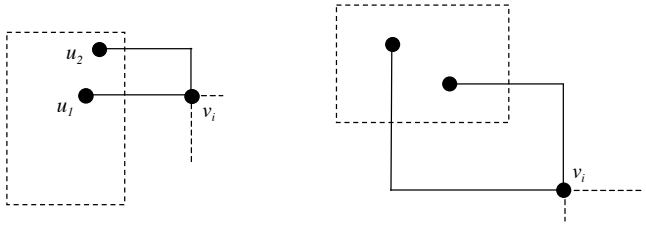


Figura 4.44.



3. *ld(v_i)=3* (vedi fig. 4.45), i casi che si possono verificare sono 8; ovunque si posizioni v_i ad esso rimane libero solo un percorso diretto. Nel caso pessimo introduciamo *3* svolte e *2* tra righe e colonne.

Figura 4.45.

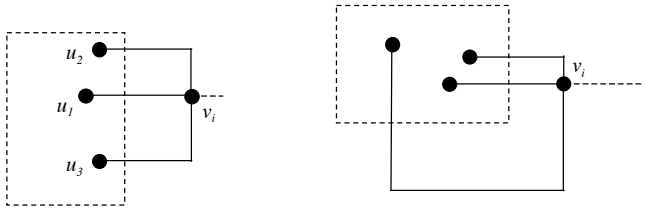
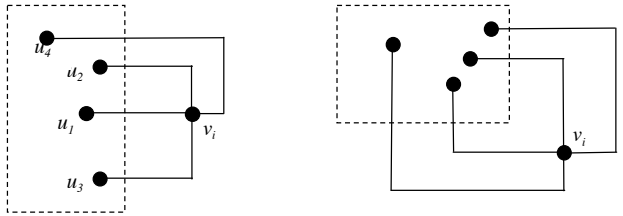


Figura 4.46.



4. *ld(v_i)=4* (vedi fig. 4.46), in base alla combinazione di tutti i possibili percorsi dei nodi u_1, u_2, u_3 ed u_4, si possono avere 16 diversi casi. Con l'inserimento di v_i si introducono al più *6* svolte e *4* tra righe e colonne.

Osserviamo che un nodo v_i inserito con grado locale 1 avrà a disposizione 3 direzioni libere, corrispondenti ad 1 percorso non diretto e 2 percorsi diretti; un nodo v_i inserito con grado locale 2 avrà a disposizione 2 direzioni libere, corrispondenti a 2 percorsi diretti; un nodo v_i inserito con grado locale 3 è posizionato in modo tale da avere una sola direzione libera, corrispondente sempre ad 1 percorso diretto.

Teorema. *Sia G un grafo di massimo grado 4 disegnato sotto lo scenario no-change; il disegno di G ha le seguenti caratteristiche:*
- *è ortogonale su griglia;*
- *un arco ha al massimo 3 svolte;*
- *il numero di svolte totali è al più 8/3n(t)+1;*
- *l'area è al più $(4/3n(t)+1/3)^2$.*

Dimostrazione. Sia $n_1(t)$, $n_2(t)$, $n_3(t)$ ed $n_4(t)$ il numero di nodi di G inseriti nel disegno con grado locale rispettivamente 1, 2, 3 e 4. Allora si ha:

$$B(t) \le n_1(t)+2n_2(t)+3n_3(t)+6n_4(t)$$

Da questa espressione e da quella di $m(t)$ nella dimostrazione precedente, si ha:

$$B(t) \le n_1(t)+2n_2(t)+3n_3(t)+6n_4(t) \le m(t)+2n_4(t) \le 2n(t)+2n_4(t).$$

Per stabilire una limitazione superiore per $B(t)$, calcoliamo il numero massimo di nodi con grado locale 4 che possono essere inseriti al tempo t. Il numero di archi incidenti ai nodi con grado locale 4 al tempo t è $4n_4(t)$. Sia $m'(t)$ il numero di archi rimanenti; allora: $m'(t)+4n_4(t) = m(t) \le 2n(t)$.

Poiché bisogna ottenere una maggiorazione per $n_4(t)$ è necessario determinare una minorazione per $m'(t)$. Questa si ottiene considerando tutti i nodi di grado locale minore di 4: al momento del loro inserimento ciascuno avrà inserito almeno un arco (tranne v_1), perciò $m'(t) \ge n(t) - 1 - n_4(t)$. Sostituendo quest'ultima disuguaglianza in $m'(t)+4n_4(t) \le 2n(t)$ segue che $3n_4(t) \le n(t)+1$.

Sostituendo nell'espressione di $B(t)$ si ottiene:

$$B(t) \le \frac{8}{3} n(t)+1.$$

Per quanto riguarda l'area, procediamo in modo analogo al teorema precedente.

$$h(t) + w(t) \le 2n_1(t)+2n_2(t)+2n_3(t)+4n_4(t)+2=$$
$$= 2[n_1(t)+n_2(t)+n_3(t)+n_4(t)]+2n_4(t)+2 \le 2(n(t)-1)+2n_4(t).$$

Poiché, come si è visto prima, $n_4(t) \le (n(t)+1)/3$ risulta $h(t)+w(t) \le 8/3\ n(t)+1$.

Dal fatto che l'area è massimizzata quando le due lunghezze $h(t)$ e $w(t)$ sono uguali, segue che

$$A(t) \le (4/3n(t)+1/3)^2.$$ **CVD**

4.3.5. Confronto tra i due scenari

I risultati riportati nei due teoremi precedenti, e riassunti nella tabella seguente, suggeriscono che sia migliore lo scenario no-change, sia in termini di area che di numero di svolte. Questo però risulta essere vero solo nel caso peggiore, mentre in pratica lo scenario relative-coordinates è migliore. E' stato condotto uno studio sperimentale con l'intento di paragonare le performances nei due scenari su un insieme di 8000 grafi, ed è stato verificato che lo scenario relative-coordinates dà

sempre risultati migliori sia in termini di area che di numero di svolte. Inoltre, i risultati teorici prodotti che si riferiscono ai casi peggiori solo molto raramente si verificano.

I punti a favore dello scenario no-change sono il vincolo di non poter muovere nodi o svolte già inserite e che ogni nuovo inserimento prende tempo costante. Nell'altro scenario, invece, si tenta di mantenere inalterata la "forma" del disegno, e l'utente ha la possibilità di inserire un nodo all'interno del disegno, e non solo nella faccia esterna.

Entrambi gli algoritmi presentati per i due scenari possono essere facilmente estesi al caso in cui il grafo non debba mantenersi necessariamente connesso durante il processo interattivo (per dettagli si veda [PT96]).

	Relatives coordinates	No change
Numero di svolte per arco	3	3
B(t): numero totale di svolta inserite a tempo t	3n(t)-1	$\frac{8}{3}$n(t)+1
A(t): area totale a tempo t	$\left(\frac{3}{2}n(t)\right)^2$	$\left(\frac{4}{3}n(t)\right)^2 + o(n(t)^2)$

4.3.6. Visualizzazione di grafi in evoluzione (network evolution)

Riferimenti: [C04] pp 278-282.

Abbiamo parlato della problematica di modificare il disegno di un grafo al suo variare nel tempo, ottimizzando i cambiamenti. Ci occupiamo ora del problema di scoprire e visualizzare i cambiamenti di un grafo che si evolve nel tempo. L'obiettivo è completamente diverso: si vogliono mantenere tutte le varie versioni del grafo ed, anzi, evidenziarne i cambiamenti.

Un primo modo di visualizzare grafi temporalmente dinamici è quello di usare *piani multipli del tempo*: ogni piano del tempo rappresenta una visualizzazione del grafo, ed una sequenzializzazione dei piani del tempo rivela lo scorrere del tempo. L'evoluzione si può visualizzare attraverso una sequenza di disegni bidimensionali del grafo, G_1, G_2, ..., G_n, uno per ciascun momento di interesse. Dovrebbe essere prestata un'attenzione particolare al mantenimento della mappa mentale dell'utente nel passaggio tra due disegni consecutivi; per fare questo si può costruire un disegno su piani semitrasparenti (vedi figura 4.47), oppure costruire un grafo $G_{1..n}$, che contiene i nodi di tutti i grafi, ed un arco collega nodi corrispondenti allo stesso nodo del grafo su piani adiacenti (vedi figura 4.48).

Figura 4.47.

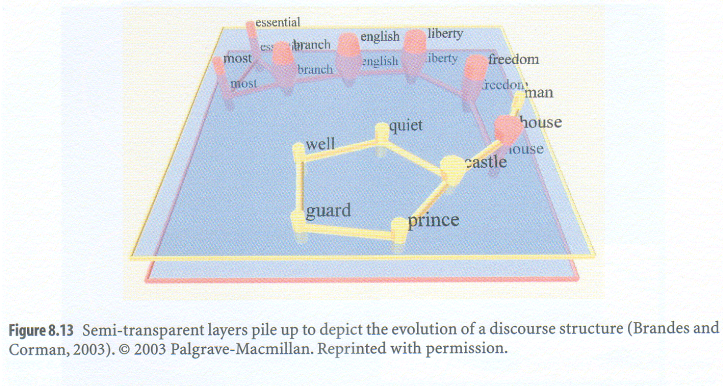
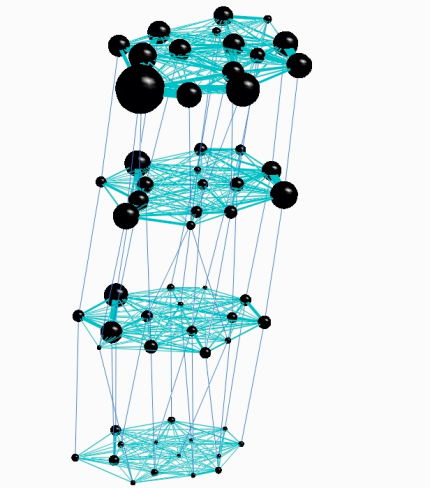


Figure 8.13 Semi-transparent layers pile up to depict the evolution of a discourse structure (Brandes and Corman, 2003). © 2003 Palgrave-Macmillan. Reprinted with permission.

Figura 4.48.



Questo tipo di rappresentazione è detta *disegno a due dimensioni e mezzo*, in quanto la terza dimensione è usata in modo totalmente differente dalle altre due (tradizionalmente gli assi sono intercambiabili nel disegno 3D, ma non qui).

Con questa tecnica è possibile rappresentare sia l'evoluzione di un grafo che anche l'evoluzione dei soli pesi ad esso assegnati, infatti, se rappresentiamo la dimensione di ciascun nodo e lo spessore di ciascun arco in modo proporzionale al loro peso, il passaggio da un piano all'altro evidenzia il cambiamento di peso, particolarmente importante quando i pesi hanno uno specifico significato.

Un metodo tipico per rappresentare soprattutto grafi di citazioni e cocitazioni consiste nell'assegnare un significato ben preciso a colori e dimensioni. Ad esempio, in Figura 4.49, il colore indica l'anno di pubblicazione della citazione, la dimensione del nodo il numero di citazioni e la larghezza del tratto il numero delle cocitazioni. In tal caso, però, non c'è traccia di evoluzione.

Nel caso in cui sia necessario visualizzare l'evoluzione nel tempo, un modo per esprimerla ci viene suggerito dalla botanica: si osservi che gli anelli dei tronchi che ne indicano l'età aiutano gli studiosi a ricostruire i cambiamenti climatici del passato. Sfruttando questa analogia, si pensi ad un articolo come ad un albero: i nodi del grafo delle citazioni crescono aggiungendo uno strato ogni anno, come se fosse uno strato di legno. Guardando gli anelli si avrà un'idea della crescita annuale delle citazioni: guardando lo spessore degli anelli si deduce se una pubblicazione sia un classico, ma anche in quali specifici anni essa sia stata particolarmente citata (vedi figure 4.50 e 4.51).

Concludiamo questa sezione ricordando che l'evoluzione può sempre essere espressa tramite animazione.

Figura 4.49.

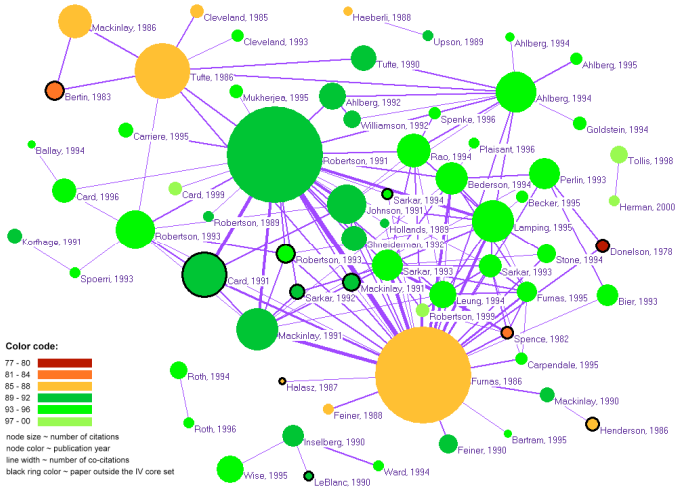


Figura 4.50.

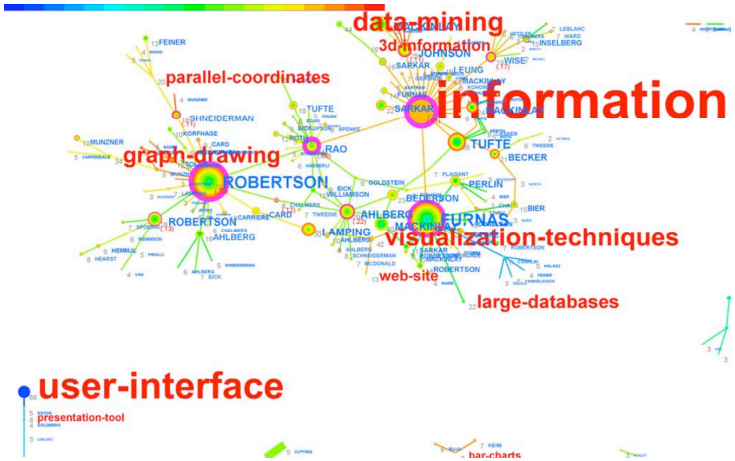


Figura 4.51.

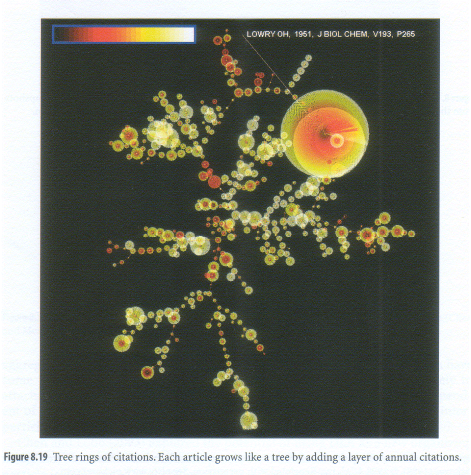


Figure 8.19 Tree rings of citations. Each article grows like a tree by adding a layer of annual citations.

RIFERIMENTI BIBLIOGRAFICI

[BP90] K.-F. Böhringer, F.N. Paulisch. Using constraints to achieve stability in automatic graph layout algorithms, *Proc. Of the ACM Human factors in Computing Systems Conference (CHI'90)*, 43-51, 1990.

[BT98] S. Bridgeman, R. Tamassia. Difference metrics for interactive orthogonal graph drawing algorithms. *Proc. GD '98* Lecture Notes in Computer Science 1457, 57-71, 1998.

[C04] C. Chen: *Information Visualization beyond the horizon* – 2nd Edition, Springer 2004.

[Dal99] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis*: Graph Drawing – Algorithms for the visualization of graphs*, Prentice Hall, 1999.

[Eal91] P. Eades, W. Lai, K. Misue, K. Sugijama. Preserving a mental map of a diagram, *Proc. Compugraphics '91*, 24-33, 1991.

[F97] U. Fößmeier. Interactive orthogonal graph drawing: Algorithms and Bounds, *Proc. 5th Int.l Symp. On Graph Drawing (GD'97)*, LNCS 1353, 242-253, 1997.

[KW98] M. Kaufmann, D. Wagner (Eds.): *Drawing Graphs – Methods and Models*. Lecture Notes in Computer Science 2025, Springer 1998.

[Lal98] K.A. Lyons, H. Meijer, D. Rappaport. Algorithms for cluster busting in anchored graph drawing, *Journal of Graph Algorithms and Applications,* 2(1), 1-24, 1998.

[Mal93] K. Mirijala, S.W. Hornick, R. Tamassia. An Incremental Approach to Aesthetic Graph Layout, *Proc. Intern. Workshop on Computer-Aided Software Engineering*, 1993.

[N96] S.C. North. Incremental layout with DynaDag, *Proc. 3rd Int.l Symp. On Graph Drawing (GD'96)*, LNCS 1027, 409-418, 1996.

[PT96] A. Papakostas, I.G. Tollis. Issues in Interactive Orthogonal Graph Drawing, *Proc. GD'95,* LNCS 1027, 419-430, 1996.

[PT98] A. Papakostas, I.G. Tollis. Interactive Orthogonal graph Drawing, *IEEE trans. On Computers,* 47(11), 1297-1309, 1998.