4.4. Generalizzazione della visualizzazione di grafi: il caso particolare del disegno ortogonale

In questa sezione si affronteranno i seguenti problemi di generalizzazione:

1. togliere il vincolo del grado ≤ 4 nel disegno ortogonale 2D negli algoritmi basati sulla rappresentazione di visibilità;
2. togliere il vincolo del grado ≤ 4 nel disegno ortogonale 2D negli algoritmi basati sulla st-numerazione;
3. togliere il vincolo del grado ≤ 6 nel disegno ortogonale 3D;
4. togliere il vincolo che il grafo sia 2-connesso;
5. togliere il vincolo che il grafo sia planare negli algoritmi basati sulla rappresentazione di visibilità;

4.4.1. Esempi di estensione al caso di grado alto

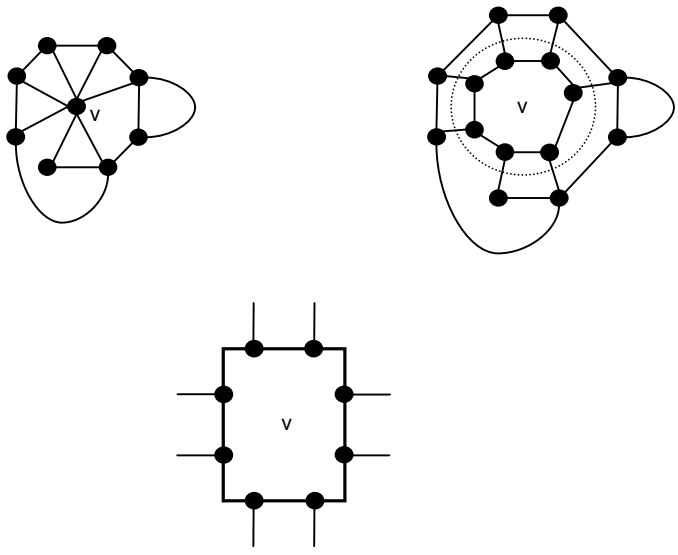
Riferimenti: [Dal99] pp. 168-169, 258-262; [Bal97] tutto.
Questa parte è un po' sintetica: va estesa

4.4.1.1. Disegno 2D ortogonale planare (rappresentazione di visibilità)

Se il grado è >4, Fößmeier e Kaufmann [FK96] propongono un algoritmo che rappresenta i nodi come rettangoli con lati orizzontali e verticali.

Sia G un grafo di cui abbiamo una rappresentazione piana. Per ogni nodo v di grado $d>4$ si espanda v in un ciclo di d nodi $v_1, v_2, \ldots v_d$ dove ogni nodo v_i diventa adiacente ad uno dei nodi precedentemente adiacenti a v (v. Figura 4.49 parte alta). Il ciclo di lunghezza d così definito è detto *ciclo espansione* del nodo v. Sia G' il grafo ottenuto da G dalla precedente procedura di espansione. G' è un grafo di cui si ha ancora la rappresentazione piana con nodi di grado al più 4. In particolare, tutti i nodi dei cicli espansione hanno grado esattamente 3. Bisogna costruire una rappresentazione piana di G' aggiungendo il vincolo che tutti i cicli espansione non abbiano svolte. Si può dimostrare che, tramite la rappresentazione di visibilità, questi cicli vengono disegnati come figure convesse (cioè ogni angolo interno è al più 180°) e quindi come rettangoli (v. Figura 4.49 parte bassa). Sostituendo nuovamente ad ogni ciclo espansione un nodo rettangolare, si ottiene un disegno ortogonale su griglia del grafo G di partenza in cui ogni nodo ha grado arbitrario.

Figura 4.49.



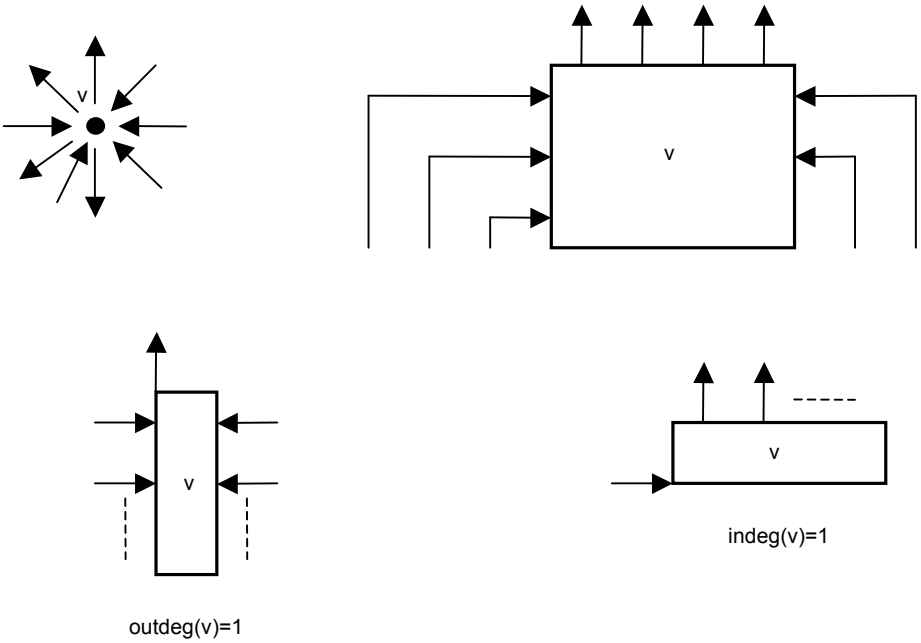
4.4.1.2. Disegno 2D ortogonale non planare (st-numerazione)

Anche in questo caso è possibile sostituire ogni nodo con una scatola rettangolare. Sul bordo si hanno tutti i punti a coordinate intere che sono detti *connettori* (v. Figura 4.50 parte alta). Ogni connettore può avere un solo arco uscente, salvo i connettori ad angolo che ne hanno 2. Il perimetro della scatola deve essere sufficientemente grande da consentire l'attacco di tutti gli archi in connettori tutti distinti.

L'algoritmo si può schematizzare come segue: per semplicità, si assuma G 2-connesso e si calcoli una st-numerazione. La dimensione delle scatole viene decisa nel momento in cui esse vengono rappresentate e dipende dal numero di archi entranti ed uscenti (ricordiamo che una st-numerazione induce un verso agli archi).

Tutti gli archi uscenti sono attaccati sulla parte alta della scatola (vedi Figura 4.50, in alto a destra), questo implica che la larghezza della scatola sia almeno quanto il suo grado uscente. Se la scatola ha grado uscente 1 si deve comunque garantire l'uso di 2 colonne (vedi Figura 4.50 in basso a sinistra). Si utilizza una scatola analoga anche per il pozzo.

Figura 4.50.



Gli archi entranti, invece, si dividono tra la parte sinistra e la parte destra della scatola a metà (vedi Figura 4.50, in alto a destra). Se la scatola ha grado entrante 1, si usano comunque 2 righe per la scatola (vedi Figura 4.50 in basso a destra) e si utilizza una scatola analoga per la sorgente. Gli archi hanno esattamente una svolta ciascuno, perché escono rettilinei da una scatola ed entrano con una svolta in un'altra scatola.

Rimane da definire dove posizionare le scatole. Se si deve inserire il nodo v, si localizzino tutti gli archi che entreranno in v (e che sono già previsti nel disegno); i nodi da cui questi archi provengono sono detti *predecessori*. Si assuma prima che il grado entrante di v sia pari. Poiché le colonne del disegno sono numerate, ne esisteranno 2, c_1 e c_2 tale che:

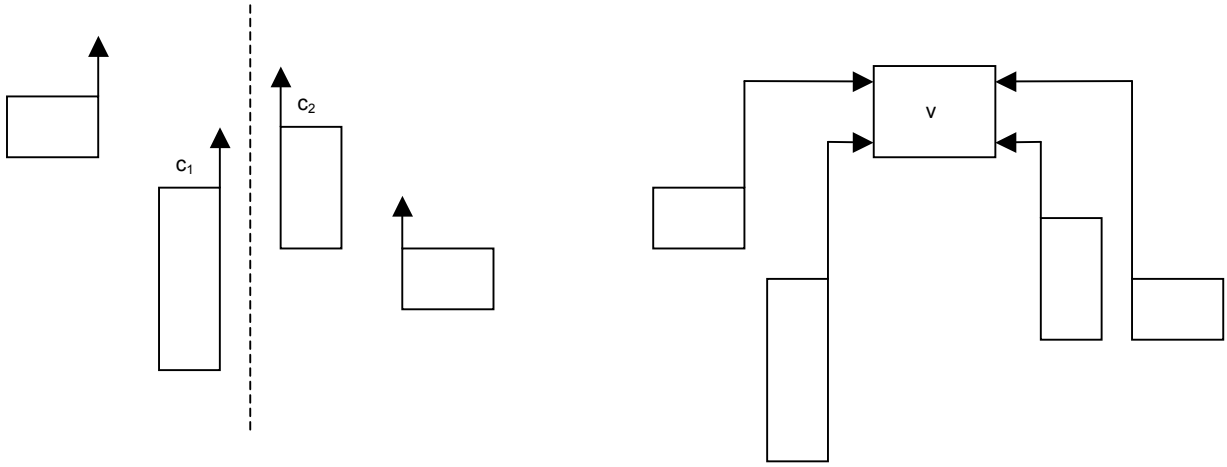
- c_1 è a sinistra di c_2
- c_1 e c_2 hanno un arco entrante in v ciascuna;
- ci sono $indeg(v)/2-1$ archi alla sinistra di c_1
- ci sono $indeg(v)/2-1$ archi alla destra di c_2

Gli archi e_1 ed e_2, corrispondenti a c_1 e c_2 sono detti *mediani entranti di v*, in particolare e_1 è il mediano sinistro ed e_2 è il mediano destro.

Se *indeg(v)* è dispari si avrà un solo mediano su una sola colonna *c*.

Il mediano stabilisce come dividere gli archi entranti tra lato sinistro e destro di *v*. Il disegno viene aperto e vengono inserite nuove righe e nuove colonne per la scatola che rappresenta *v*. Il metodo è schematizzato in Figura 4.51.

Figura 4.51.



Per l'algoritmo ora descritto vale il seguente risultato:

Teorema: *Dato G 2-connesso, l'algoritmo precedente produce un disegno ortogonale su griglia di G in O(m) tempo con le seguenti proprietà:*

- *il perimetro di ogni scatola è proporzionale al grado del nodo;*
- *la larghezza è al più m+n(out1)+2 e l'altezza è al più* $\frac{m}{2}+\frac{n(odd)}{2}+n(in1)+n(in2)+2$ *dove n(out1) è il numero di nodi con un arco uscente, n(odd) il numero di nodi con un numero dispari di archi entranti, n(in1) il numero di nodi con un arco entrante ed n(in2) il numero di nodi con 2 archi entranti;*
- *ogni arco ha al più una svolta.*

Dimostrazione. Per costruzione, il perimetro di ogni nodo-scatola è proporzionale al grado che essa rappresenta, ed ogni arco ha al più una svolta.

Calcoliamo ora la dimensione del disegno. Si osservi che la somma di tutte le larghezze di tutti i rettangoli definisce la larghezza del disegno. Poiché la larghezza di ogni scatola è uguale al grado uscente del nodo che essa rappresenta, allora:

$$\Sigma_{v\in V}\text{larghezza}(v) \geq \Sigma_{v\in V}\text{ outdeg}(v)=m.$$

Al fine di ottenere l'uguaglianza nella disequazione precedente, al termine di destra bisogna aggiungere il contributo ulteriore dei nodi il cui outdeg=1 (essi usano 2 colonne), e tale contributo è proprio *n(out1)*; dobbiamo infine aggiungere le due colonne relative al nodo pozzo, che ha outdeg=0.

L'altezza del disegno è data dal numero di righe. Per ottenerlo, si consideri:

$$\Sigma_{v\in V}\text{altezza}(v) \geq \Sigma_{v\in V}\lceil \text{indeg}(v)/2\rceil.$$

Di nuovo, per ottenere l'uguaglianza, bisogna aggiungere un certo numero di righe, basandosi sui seguenti fatti:

- nodi con uno o due archi entranti richiedono 2 righe, quindi dobbiamo aggiungere un contributo di *n(in1)+n(in2)*;
- $\left\lceil \frac{in\deg(v)}{2} \right\rceil = \frac{in\deg(v)}{2}+\frac{1}{2}$. La sommatoria $\Sigma_{v\in V}\lceil \text{indeg}(v)/2\rceil$ si può scrivere come:

$\sum_{\substack{v\in V \\ in\deg pari}} \left\lceil \frac{in\deg(v)}{2} \right\rceil + \sum_{\substack{v\in V \\ in\deg dispari}} \left\lceil \frac{in\deg(v)}{2} \right\rceil = \frac{1}{2}\sum_{v\in V} in\deg(v) + \sum_{\substack{v\in V \\ in\deg dispari}} \frac{1}{2}$. Dobbiamo quindi aggiungerne un contributo pari alla metà degli *n(odd)* nodi con grado entrante dispari, cioè un totale di $\frac{1}{2}n(odd)$;

- infine, la sorgente contribuisce con 2 righe, anche se ha indeg=0.

Per dimostrare la linearità dell'algoritmo, è necessario utilizzare una struttura dati dovuta a Dietz e Sleator [DS87] per il mantenimento lineare dell'ordinamento. **CVD**

Il motivo per cui non vengono attaccati archi entranti nella parte bassa di un nodo è duplice: innanzi tutto, gli archi entranti non giacciono necessariamente su colonne contigue, ed inoltre, se anche se ne potesse inserire qualcuno dal basso, potrebbe succedere che alcuni nodi debbano essere allungati per fare posto ai nuovi nodi da inserire, e questo porterebbe ad un aumento del perimetro, non più necessariamente proporzionale al grado.

Questo algoritmo può essere migliorato consentendo il riuso di righe e colonne, in un modo simile alla pairing technique usata da Papakostats e Tollis.

4.4.1.3. Disegno 3D ortogonale

Si pensi di dover rappresentare in modo ortogonale un grafo completo. Questa richiesta non è così assurda se si pensa che ogni grafo semplice con n nodi è un sottografo di K_n e che i grafi completi sono spesso critici per alcune argomentazioni sulle limitazioni inferiori.

Nel caso tridimensionale, i nodi vengono rappresentati da parallelepipedi o da segmenti, a seconda degli algoritmi.

Verranno qui presentati dei metodi per visualizzare questi grafi con i seguenti risultati:

- non esiste un disegno senza svolte, se n è sufficientemente grande;
- se il disegno ha al più una svolta per arco il volume è $O(n^3)$
- se il disegno ha al più 2 svolte per arco il volume è $O(n^3)$
- se il disegno ha al più 3 svolte per arco il volume è $O(n^{2.5})$

E' possibile provare che $\Omega(n^{2.5})$ volume è una limitazione inferiore per il volume, quindi il terzo algoritmo è ottimo. Nel seguito dimostreremo uno ad uno tutti questi risultati, tranne la limitazione inferiore, di cui omettiamo la prova.

Non esiste un disegno senza svolte, per n sufficientemente grande

Teorema. *Per n abbastanza grande, K_n non ha un disegno 3D ortogonale su griglia senza svolte.*

Dimostrazione. Indipendentemente da come siano rappresentati i nodi del grafo (come parallelepipedi o come segmenti), un suo disegno ortogonale 3D senza svolte partiziona gli archi in tre classi, una per ogni asse cartesiano, infatti gli archi senza svolte di un disegno ortogonale su griglia non possono che essere segmenti paralleli ad uno dei tre assi. Ognuna di queste classi di archi può essere interpretata come la rappresentazione di visibilità di un sottografo del grafo di partenza.

Per completare la dimostrazione, abbiamo bisogno del risultato e della definizione qui di seguito:

Lemma. [FHW95] *K_{56} non ammette una rappresentazione di visibilità.*

Definizione. [GRS80] Si definisce *3-Ramsey number R(r, b, g)* il più piccolo numero tale che ogni arbitraria partizione degli archi del grafo completo $K_{R(r,\ b,\ g)}$ che li colori rispettivamente di

rosso, blu e verde induce o un sottografo completo K_r rosso, o un sottografo completo K_b blu, o un sottografo completo K_g verde. Questo numero esiste ed è finito.

Se prendiamo $n = R(56, 56, 56)$, in qualunque modo $K_{R(r, b, g)}$ venga rappresentato come disegno rettilineo ed in qualunque modo i suoi archi vengano partizionati in tre classi che possiamo pensare colorate con i colori rosso, blu e verde, si otterrà almeno un K_{56} per la definizione di Ramsey number. Ma un K_{56} non ammette rappresentazione di visibilità per il teorema precedente. Quindi $K_{R(56, 56, 56)}$ non può essere rappresentato con un disegno ortogonale 3-D senza svolte. **CVD**

Diamo ora degli algoritmi per visualizzare grafi completi con i nodi rappresentati da segmenti.

Una *z-linea* è una linea parallela all'asse delle z; le *x-linee* e le *y-linee* si definiscono analogamente. Uno *z-piano* è un piano ortogonale all'asse delle *z*; gli *x-piani* e gli *y-piani* si definiscono analogamente.

Disegno in $O(n^3)$ con una svolta per arco

Teorema. *Esiste un disegno ortogonale 3D su griglia di K_n in volume $O(n^3)$ e ogni arco ha al più una svolta.*

Dimostrazione. La dimostrazione è costruttiva, quindi forniamo un algoritmo.

Si numerino i nodi in modo arbitrario, $v_1, v_2, \ldots, v_n$. Si disegni il nodo v_i come un segmento (la cui lunghezza sarà definita in seguito) e lo si posizioni parallelamente all'asse z a partire dal punto di coordinate (i, i). Si disegni l'arco $e = (v_i, v_j)$, con $i < j$, con una svolta lungo il percorso $(i, i), (i, j), (j, j)$ su una coordinata z che sarà definita più avanti. Si noti che nessun nodo o parte di arco è posizionato su alcun punto a coordinate (x, y) con $y < x$.

Per definire la coordinata z di ciascun arco, si partizionino gli archi di K_n in due insiemi E_i^a e E_i^b, con $i = 1, 2, \ldots, n/2$ definiti come segue:

$$E_i^a = \{(v_{i-l+1}, v_{i+l}), l = 1, 2, \ldots, n/2\} \text{ ed } E_i^b = \{(v_{i-l}, v_{i+l}), l = 1, 2, \ldots, n/2 - 1\}$$

dove tutte le addizioni sono modulo n. Mostriamo ora che questi insiemi partizionano gli archi di K_n, e che non ci sono sovrapposizioni o incroci né tra gli archi di E_i^a né tra quelli di E_i^b.

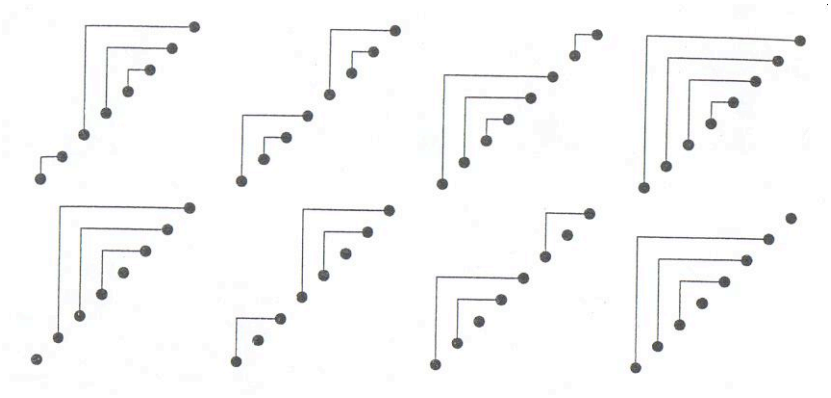
Dimostriamo prima che per ogni arco (u,v) esistono due indici i ed l tale che (u,v) è in E_i^a oppure in E_i^b. Si osservi che gli indici degli estremi degli archi in E_i^a hanno diversa parità, mentre quelli degli archi in E_i^b hanno la stessa parità; quindi, dato l'arco (u,v), si può controllare se le numerazioni di u e di v abbiano diversa o uguale parità per decidere se l'arco debba essere ricercato in un insieme di tipo a o di tipo b. Per determinare i ed l, è sufficiente scrivere due semplici equazioni lineari in due incognite; nel caso di insieme di tipo a esse saranno: $i-l+1=u$ ed $i+l=v$; nel caso di insieme di tipo b esse saranno: $i-l=u$ ed $i+l=v$. Ciascuno di questi due sistemi ammette un'unica soluzione, e dunque gli insiemi costituiscono una partizione degli archi.

Rimane ora da dimostrare che, comunque si prendano due archi nello stesso insieme, questi non verranno disegnati con delle intersezioni. Per fare ciò, si prendano due archi (u,v) e (x,y) nello stesso insieme (ad esempio E_i^a); allora esisteranno due valori l ed l' tali che: $u=i-l+1$; $v=i+l$; $x=i-l'+1$; $y=i+l'$. Allora, è facile dimostrare che o i due archi sono "indipendenti" (cioè, ad esempio, $u<v<x<y$), oppure sono uno dentro l'altro (cioè, ad esempio, $u<x<y<v$), mentre non può mai accadere che gli archi si intreccino (cioè, ad esempio, $u<x<v<y$).

Fatte queste premesse, è dunque possibile posizionare ciascun insieme su uno z-piano differente, occupando n z-piani in tutto (vedi Figura 4.52, dove ogni piano è rappresentato a se' stante).

Il *volume* di questa rappresentazione è $n \times n \times n$ e i punti $\{(x, y, z): y < x\}$ non sono utilizzati.

Figura 4.52.



Si osservi che i due insiemi E_i^a e E_i^b possono essere disegnati sullo stesso piano riflettendo gli archi di E_i^a rispetto alla linea diagonale dei nodi. Questo permette di ottenere un disegno di K_n in una griglia $n \times n \times n/2$.

Osservazione. Questa strategia è strettamente connessa al *page number* di un grafo (definito come il minimo numero di piani, o pagine diverse, necessarie per rappresentare un grafo senza incroci se i nodi sono posizionati tutti lungo una retta), che può rivelarsi utile per i disegni di grafi sparsi. Questa idea permette, per esempio, di disegnare grafi planari in volume $O(n^2)$ in una griglia $n \times n \times 4$, visto che è noto che i grafi planari hanno pagenumber pari a 4 [Y89].

Per migliorare il risultato ora descritto si consideri la seguente strategia.

Siano K^1 e K^2 due disegni di $K_{n/2}$ uguali a quelli descritti precedentemente, che possono quindi essere inseriti in un cubo di dimensioni $n/2 \times n/2 \times n/2$. Si riflettano i punti del cubo che racchiude K^2 rispetto al piano $y = 0$, in modo che tutti i punti del K^2 riflesso abbiano la coordinata y negativa. Successivamente si ruoti il K^2 riflesso in modo che i nodi v_j del K^2 ruotato e riflesso si sovrappongano ai punti di coordinate $(x, -j, j)$, dove $1 \le x \le n/2$, come mostrato in Figura 4.53.

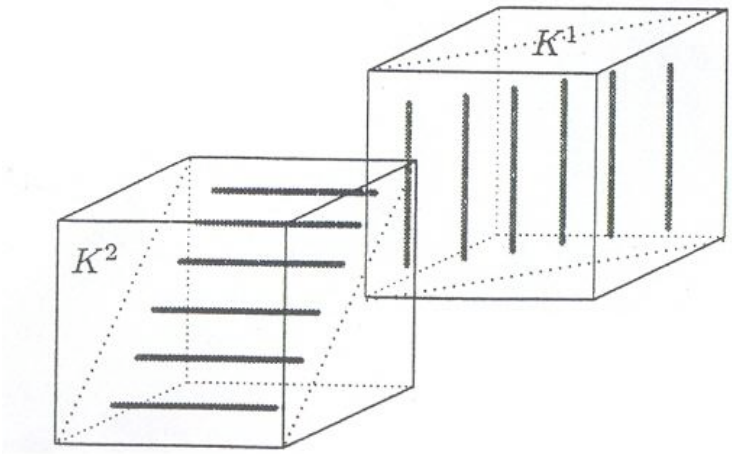
Ogni nodo v_i in K^1 "vede" i nodi v_j nel K^2 ruotato e riflesso lungo la y-linea $[(i, i, j), (i, -j, j)]$. Quindi questi archi possono essere disegnati come linee rette, in modo da produrre un disegno di K_n. Il loro numero è pari ad $n^2/4$, infatti il numero di tutti i modi possibili di collegare due insiemi di $n/2$ nodi è $n/2 \cdot n/2$.

Si ottiene così un disegno di dimensione $X = Z = n/2$ ed $Y = n$. Come detto, ci sono $n^2/4$ archi disegnati senza svolte, i rimanenti invece hanno una sola svolta, così il volume di tale disegno è $n/2 \times n/2 \times n$ ed il numero totale di svolte è $n^2/4 - n/2$. Infatti i due insiemi E_i^a e E_i^b hanno entrambi cardinalità pari ad $n/2$. Inoltre, tutte le configurazioni di E_i^a contengono $n/2$ archi con una svolta ciascuno, e analogamente tutte le configurazioni di E_i^b contengono $n/2$ archi. Allora, ognuno dei due $K_{n/2}$ ha $n^2/8 - n/4$ svolte. Quindi il numero totale di svolte è $2\,(n^2/8 - n/4) = n^2/4 - n/2$. **CVD**

Disegno in $O(n^3)$ con due svolte per arco

Una strategia simile alla precedente può essere applicata quando sono permesse al più 2 svolte per arco. Nella dimostrazione del teorema seguente, K_n viene disegnato con al più 2 svolte per arco e, per diminuire il suo volume, si fanno due copie di un disegno di $K_{n/2}$ e poi si posizionano queste due copie in una griglia con lati di lunghezza $n/2$, fornendo gli archi di connessione tra le due parti.

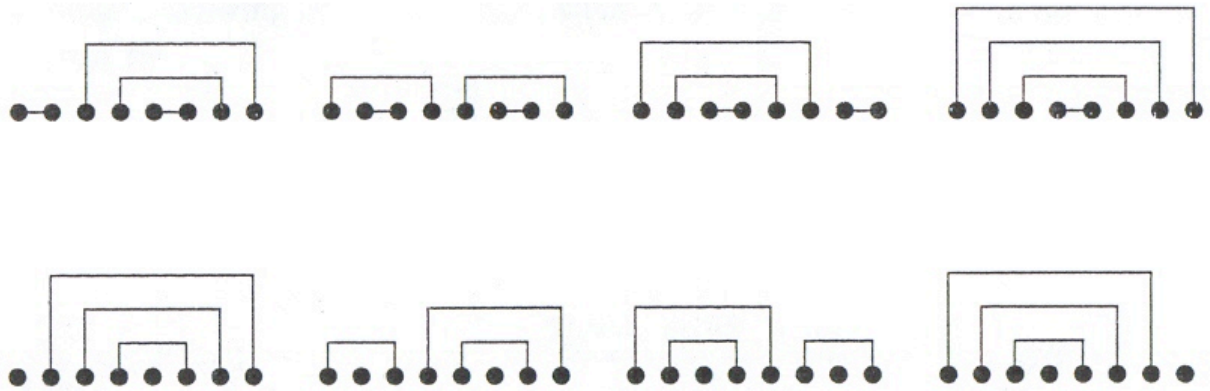
Figura 4.53.



Teorema. *Esiste un disegno ortogonale 3D su griglia di K_n in volume $O(n^3)$ e ogni arco ha al più due svolte.*

Dimostrazione. Si numerino i nodi come $\{v_1, v_2, \ldots, v_n\}$ e si posizioni v_i sulla retta parallela all'asse z di coordinate $(x, y) = (i, 1)$. Si disegni l'arco $e = (v_i, v_j)$, dove $i < j$, seguendo il percorso $(i, 1), (i, y), (j, y), (j, 1)$, creando così due svolte se $y > 1$ e nessuna svolta se $y = 1$. Il valore da dare

Figura 4.54.



ad y è pari alla parte intera superiore del rapporto $\frac{j-i}{2}$ (intuitivamente, più l'arco collega nodi lontani e più la coordinata y corrispondente è grande). Il valore da dare a z è analogo a quello assegnato nella prova precedente, ottenuto definendo i due insiemi E_i^a e E_i^b come sopra: anche qui non ci sono sovrapposizioni o incroci tra archi di uno stesso insieme e quindi sono sufficienti n z-piani. Visto che la coordinata y più grande è $(n - 1)/2$, il parallelepipedo che contiene il disegno ha dimensioni $n \times n/2 \times n$. Gli archi (v_i, v_{i+1}), per $i = 1, 2, \ldots, n-1$, sono rettilinei, quindi ci sono $n-1$ archi rettilinei; tutti gli altri archi hanno 2 svolte, quindi il numero totale di svolte è $n^2 - 3n + 2$.

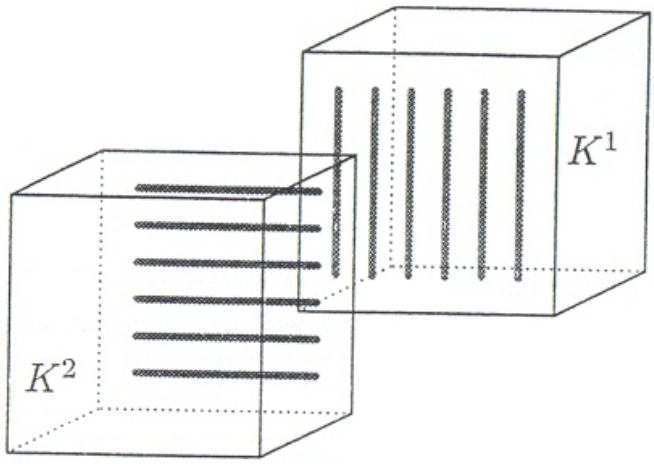
In figura 4.54 sono rappresentati E_i^a e E_i^b per K_8.

Anche questo disegno può essere migliorato considerando K^1 e K^2, due disegni di $K_{n/2}$ uguali a quelli descritti precedentemente, tali che entrambi possano essere inseriti in un volume di dimensioni $n/2 \times n/4 \times n/2$. Si rifletta K^2 rispetto al piano $y = 0$, in modo che tutti i punti del K^2 riflesso abbiano la coordinata y negativa. Successivamente, si ruoti il K^2 riflesso in modo che il

nodo v_j del K^2 ruotato e riflesso si sovrapponga ai punti $(x, -1, j)$, dove $1 \le x \le n/2$, come mostrato in figura 4.55. Ogni nodo v_i in K^1 "vede" i nodi v_j nel K^2 ruotato e riflesso lungo la y-linea $[(i, 1, j), (i, -j, j)]$, quindi questi archi possono essere disegnati come linee rette, in modo da produrre un disegno di K_n. Si ottiene così un disegno di dimensione $X = Y = Z = n/2$ ed il numero totale di svolte è

$$2\left(\frac{n^2}{4} - \frac{3}{2}n + 2\right) = \frac{n^2}{2} - 3n + 4. \qquad \textbf{CVD}$$

Figura 4.55.



Disegno in $O(n^{2.5})$ con tre svolte per arco

Viene ora fornito un disegno di K_n con al più 3 svolte per arco ed un volume di $O(n^{2.5})$. Per semplicità, si assumerà che $n = N^2$ per un certo intero N ma è chiaro che la dimostrazione può essere facilmente generalizzata ad ogni n.

Teorema. *Esiste un disegno ortogonale 3D su griglia di K_n in volume $O(n^{2.5})$ e ogni arco ha al più tre svolte.*

Dimostrazione. Si numerino i nodi in coppie ordinate (i, j), dove $1 \le i \le N$, $1 \le j \le N$, (dove N è $n^{1/2}$) e si posizioni il generico nodo (i, j) a coordinate $(2i, 2j)$ del piano XY. Il generico arco e unisca i nodi (i_1, j_1) e (i_2, j_2). Non è restrittivo assumere che $i_1 \le i_2$ e, nel caso in cui $i_1 = i_2$, allora $j_1 > j_2$. L'arco e verrà detto *L-arco* se $j_1 > j_2$ o *Γ-arco* altrimenti.

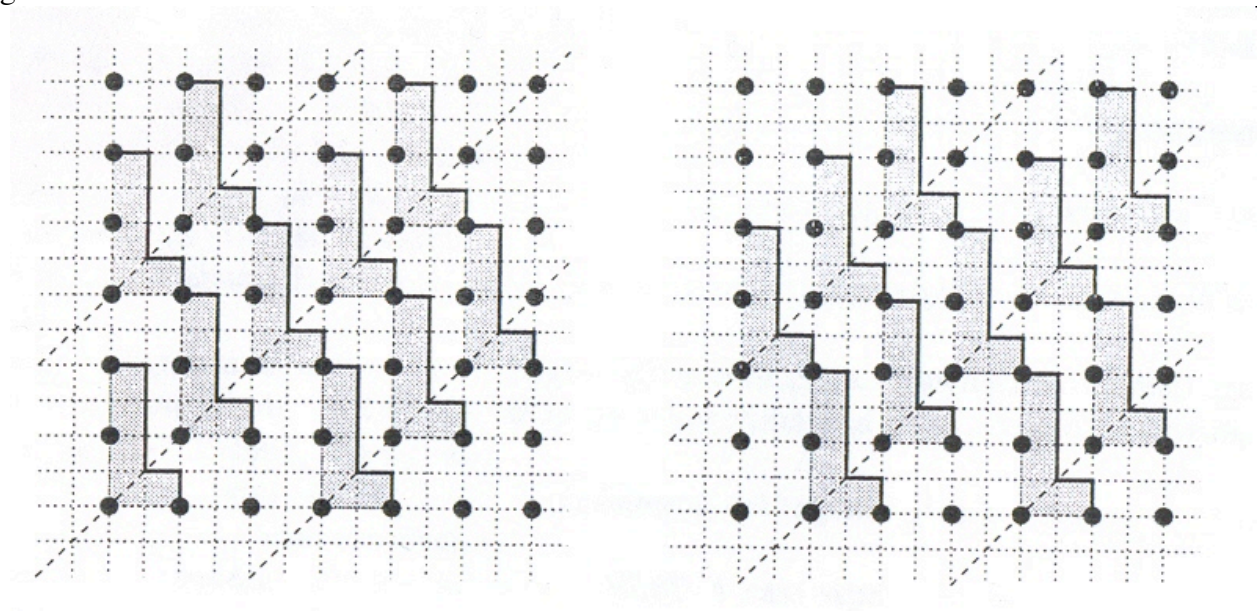
Si disegnino gli L-archi lungo il percorso $(2i_1, 2j_1)$, $(2i_1 + 1, 2j_1)$, $(2i_1 + 1, 2j_2 + 1)$, $(2i_2, 2j_2 + 1)$, $(2i_2, 2j_2)$, in modo da avere tre svolte. Analogamente, si disegnino i Γ-archi lungo il percorso $(2i_1, 2j_1)$, $(2i_1 + 1, 2j_1)$, $(2i_1 + 1, 2j_2 - 1)$, $(2i_2, 2j_2 - 1)$, $(2i_2, 2j_2)$.

Per evitare incroci, è necessario dividere gli L-archi in $N(N - 1)$ gruppi E_{d_x,d_y}, con $0 \le d_x \le N - 1$ e $1 \le d_y \le N - 1$. Analogamente si deve fare per i Γ-archi. Ogni gruppo E_{d_x,d_y} consiste degli archi $((i_1, j_1), (i_2, j_2))$ per i quali si ha $i_2 = i_1 + d_x$ e $j_2 = j_1 - d_y$. Visto che $i_1 \le i_2$ e $j_1 > j_2$ per ogni L-arco, questi gruppi ricoprono tutti gli L-archi. Informalmente, ogni E_{d_x,d_y} raggruppa tutti gli L-archi aventi la stessa dimensione nel disegno.

Si divida ancora ogni gruppo E_{d_x,d_y} in $d_x + d_y$ insiemi di archi nel modo seguente: per $p = 0, 1, \ldots, d_x + d_y - 1$, siano $E^p_{d_x,d_y}$ gli archi in E_{d_x,d_y} per i quali $j_2 - i_1 = p \bmod (d_x + d_y)$. In altre parole, gli angoli in basso a sinistra degli *L-archi* in $E^p_{d_x,d_y}$ si trovano sulle diagonali che intersecano gli assi delle y nel punto $2p \bmod (2d_x + 2d_y)$ (vedi Figura 4.56). È facile controllare che nessun arco in

$E^p_{d_x,d_y}$ sia oggetto di sovrapposizioni o incroci; infatti gli angoli delle L sono posizionati su una sequenza di diagonali distanti tra loro $2(d_x + d_y)$. Inoltre, si noti che $E^p_{d_x,d_y}$ è non vuoto solo se $p \leq 2N - d_x - d_y$.

Figura 4.56.



Se ora assegnamo un piano z ad ogni insieme $E^p_{d_x,d_y}$, otteniamo un disegno legittimo di tutti gli L-archi. Il disegno dei Γ-archi in modo analogo raddoppia il numero di piani utilizzati, producendo un disegno di K_n in una griglia di dimensioni $X \times Y \times Z$ tale che $X = Y = 2N = 2\sqrt{n}$ per costruzione; la dimensione Z è data da $N(N-1)\ 2N = 2N^3 = 2n^{3/2}$. Ne segue che il volume è $O(n^{5/2})$. Ogni arco ha al più tre svolte. Poiché i $2N(N-1) = 2n - 2\sqrt{n}$ archi per cui $d_x = 0$ e $d_y = 1$, oppure dove $d_x = 1$ e $d_y = 0$ possono essere disegnati senza svolte, il numero totale di svolte è

$$3(n^2/2 - n/2) - 3(2n - 2\sqrt{n}) = \frac{3}{2}n^2 - \frac{15}{2}n + 6\sqrt{n}.$$ **CVD**

4.4.2. Esempi di estensione da grafi biconnessi a semplicemente connessi

Riferimenti: [BK94] pp 7-9; [Dal99] pp 253-258

E' noto che è possibile rappresentare ortogonalmente in due dimensioni su griglia grafi 2-connessi di massimo grado 4, su cui si può definire una st-numerazione o una rappresentazione di visibilità.

Nel seguito si mostrerà come estendere questi risultati ai grafi semplicemente connessi.

4.4.2.1. Generalizzazione dell'algoritmo di Biedl e Kant

questa parte è molto semplificata rispetto all'articolo; per i dettagli, si veda [BK94]

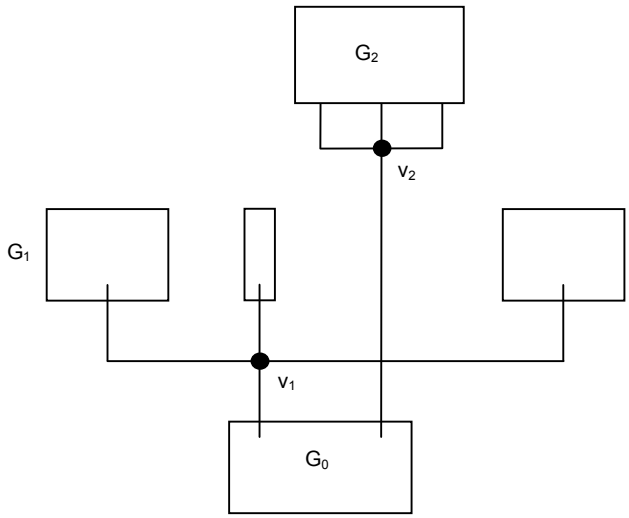
Dato un grafo qualunque G (semplicemente connesso) di grado massimo 4, in tempo lineare è possibile determinare le sue componenti 2-connesse e i suoi punti di articolazione.

Ogni blocco con n nodi si può disegnare in area $(n+1) \times n$ con al più $2n+4$ svolte, sebbene si possa dimostrare che le svolte non possono essere più di $2n-1$ (algoritmo di Biedl e Kant). Il problema consiste nel ricongiungere tutti i blocchi e aggiungere i punti di articolazione.

Per semplicità, si consideri il grafo G semplicemente connesso e diviso in una prima componente G_0 e in altri s sottografi connessi G_1, … G_s collegati a G_0 tramite dei punti di articolazione v_1, …, v_s.

Il blocco G_0 è biconnesso e viene disegnato come tale. Poi viene attaccato v_1 aggiungendo una nuova riga e si aggiunge sopra ancora il disegno di G_1, tagliandolo in al più 3 'fette' in modo che quella centrale sia di larghezza 1 e contenga il nodo a cui è collegato v_1. Se questo è l'unico punto di attacco, le altre due 'fette' vanno messe su porzioni che sono rispettivamente alla sinistra e alla destra del disegno di G_0, in modo che i successivi v_i trovino spazio per far passare i loro collegamenti (Figura 4.57).

Figura 4.57.



Detto n_i il numero di nodi del sottografo G_i esclusi i suoi punti di articolazione, se s è il numero dei i punti di articolazione, si ha che:

- $\sum_{i=0}^{s} n_i = n-s$; allora la larghezza del disegno è pari a $\sum_{i=0}^{s} (n_i+1)=(n-s)+s+1=n+1$;
- l'altezza è pari a $\sum_{i=0}^{s} (n_i)+s=(n-s)+s= n$. Quindi il valore dell'area si mantiene per i grafi connessi così come per i 2-connessi;
- analoghi ragionamenti si possono fare per il numero di svolte, pari a $B(G)=\sum_{i=0}^{s} (2n_i-1)+2s=2n-1-s \leq 2n-1$.

4.4.2.2. Metodo dell'aumentazione

Un metodo più semplice è il seguente, detto dell'*aumentazione*. Si consideri l'albero dei blocchi e dei punti di articolazione (vedi Figura 4.57). Tale albero T è definito al modo seguente:

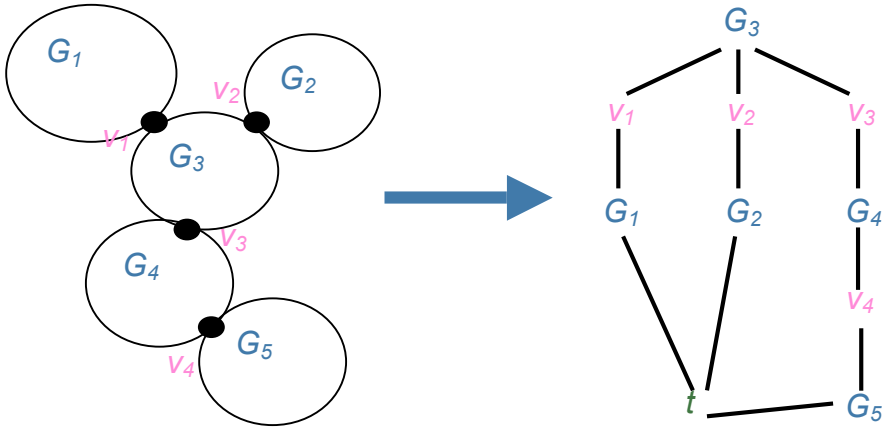
- T ha un B-nodo per ogni blocco di G
- T ha un C-nodo per ogni punto di articolazione di G
- un arco collega un B-nodo ad un C-nodo se il punto di articolazione associato al C-nodo appartiene al blocco corrispondente al B-nodo.

Si osservi che il grafo così costruito non può avere cicli per la definizione di grafo semplicemente connesso, e quindi è necessariamente un albero. Esso si può costruire in tempo lineare rispetto alla dimensione del grafo.

Si inserisca un nodo fittizio t nel grafo collegandolo ad un nodo qualunque (possibilmente di grado <4) di ciascun B-nodo foglia nell'albero dei blocchi e dei punti di articolazione. Il risultante

grafo è 2-connesso. Si calcoli una st-numerazione del grafo aumentato, con *s* vertice qualunque adiacente a *t*, possibilmente di grado <4 e *t* il nodo fittizio. Si rimuova *t*; questo produce un grafo nuovamente semplicemente connesso, con molti pozzi ed una sola sorgente. Si esegua un algoritmo di disegno ortogonale.

Figura 4.57.



Questo metodo non garantisce che l'area e il numero di svolte si mantengano pari al caso 2-connesso, infatti entrambi crescono linearmente con il numero di pozzi (e questo si capisce intuitivamente da come viene inserito nel disegno il nodo *n* nel caso 2-connesso, che occupa più spazio e introduce più svolte). Tuttavia si ricade nel caso peggiore solo se i pozzi sono di grado 4 (se hanno grado <4 non introducono nulla in più), ed è molto più facile da implementare del metodo precedente.

4.4.3. Planarizzazione

Riferimenti: [Dal99] pp 215-216; [KW98] pp 29-33.

Tecniche di planarizzazione sono motivate dalla disponibilità di molti algoritmi efficienti e ben studiati per grafi planari. Se il grafo è non planare, possiamo trasformarlo in un grafo planare per mezzo di un passo preliminare di planarizzazione, a cui segua un qualunque algoritmo per disegnare grafi planari.

E' chiaro che queste tecniche modificano pesantemente il grafo in input, quindi si vorrebbe cercare di eseguire il minimo numero di modifiche possibili. Questo si traduce nel partire dal disegno con il minimo numero di incroci o nel trovare il massimo sottografo planare, problemi entrambi NP-ardui. Perciò tutti gli algoritmi di planarizzazione usano delle euristiche. Se ne presenteranno alcune a titolo di esempio.

Metodo 1. Un modo abbastanza drastico per far diventare il grafo in input planare è quello di cancellare nodi fino a quando il grafo risultante sia planare. Questa non è una tecnica molto usata per due ragioni:

1. è NP-completo decidere se la cancellazione di *k* nodi sia sufficiente per rendere un grafo planare;
2. per scopi di visualizzazione, non ha senso avere il disegno di un altro grafo: è noto che la rimozione di anche pochi nodi può cambiare drasticamente l'essenza del grafo.

Metodo 2. Un altro metodo è quello del *vertex splitting*, che consiste nello spezzare un nodo *v* in due nodi v_1 e v_2 e partizionare gli archi incidenti a *v* tra v_1 e v_2. Sostanzialmente, viene aperto un varco là dove c'è un incrocio. Anche questa è un'operazione drastica, ed inoltre, anche qui, testare se *k* vertex splittings sono sufficienti per planarizzare un grafo è NP-completo.

Figura 4.58.

Metodo 3. Una *operazione di planarizzazione* in un grafo consiste nell'aggiungere un nuovo nodo *x* rimpiazzando una coppia di archi *(a,b)* e *(c,d)* non adiacenti ma con un incrocio, con i quattro archi *(a,x), (b,x), (c,x)* e *(d,x)*. Intuitivamente, un'operazione di planarizzazione aggiunge un nodo fittizio *x* laddove i due archi *(a,b)* e *(c,d)* incrociano.

Figura 4.59.

Una *planarizzazione G'* del grafo *G* è un grafo ottenuto da *G* tramite una sequenza di operazioni di planarizzazione. E' facile mostrare che ogni grafo ammette una planarizzazione. La planarizzazione si può disegnare utilizzando un qualunque algoritmo per grafi planari. Per poi tornare al disegno del grafo originario, bisogna risostituire i nodi fittizi con gli incroci.

Metodo 4. L'ultimo metodo analizzato consiste nella rimozione di archi.

Dato *G* non planare, si vuole trovare un suo sottografo planare. In particolare, un suo albero ricoprente è planare. Se però ci si riduce all'albero ricoprente, si cancellano molti archi. In realtà, decidere se è sufficiente cancellare *k* archi da *G* per renderlo planare è un problema NP-completo, ed il relativo problema di ottimizzazione si chiama MAXIMUM PLANAR SUBGRAPH. Il problema simile, MAXIMAL PLANAR SUBGRAPH, è invece polinomiale.

Tale problema si può risolvere al modo seguente: dato un grafo *G*, si consideri un grafo di partenza *G'=(V, Ø)* e si cerchi un arco di *G* che può essere aggiunto a *G'* mantenendo la planarità. Se si trova un tale arco, esso si aggiunge a *G'*, altrimenti il *G'* corrente è il sottografo planare massimale cercato. Poiché ad ogni passo si deve eseguire un test di planarità, che costa *O(n)* tempo, l'intero algoritmo richiede *O(n m)* tempo. Di Battista e Tamassia [DT90] hanno sviluppato una struttura dati, detta SPQR-tree, che permette di risolvere il problema in *O(m log n)* tempo.

4.5. Ottimizzazione della visualizzazione di grafi: il caso particolare del disegno ortogonale

In questa sezione verranno trattati i seguenti problemi di ottimizzazione:

- minimizzazione del numero di svolte in un disegno ortoginale 2D; poiché la versione decisionale del problema è NP-completa [GT94], verranno proposte delle euristiche;
- compattamento dell'area.

4.5.1. Minimizzazione del numero di svolte

Riferimenti: [KW98] pp 167-169 escluso algoritmo delle 4M
Per approfondire: [Dal99] pp 164-168, 143-150.

E' già stato detto che per minimizzare il numero di svolte si può usare il metodo del flusso [T87], in modo che il problema si riduca a ricercare il flusso di costo minimo su una rete. Questo algoritmo ha complessità $O(n^{7/4} \log n)$. (Per i dedttagli vedi Dal pp. 143-150 – TESINA dagli articoli originali).

L'alternativa è quella di determinare alcuni metodi volti solo a diminuire il numero di svolte, senza la pretesa di minimizzarle, in modo però efficiente, cioè lineare nella dimensione del grafo.

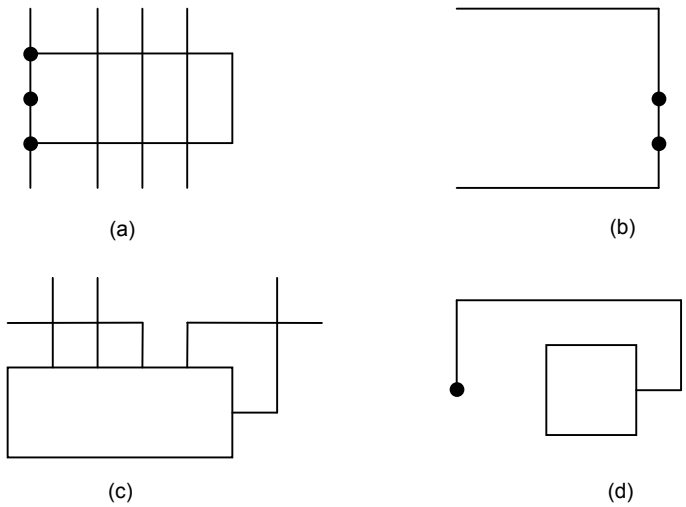
Nell'algoritmo di Tamassia e Tollis per il disegno ortogonale che usa la rappresentazione di visibilità, sono già state introdotte le tre trasformazioni T1, T2 e T3, volte a ridurre il numero di svolte. Six, Kakoulis e Tollis [SKT98] hanno aggiunto a quelle regole altre, in modo da migliorare ulteriormente l'apparenza generale del disegno, cioè non solo il numero di svolte, ma anche il numero di incroci, l'area e la lunghezza totale degli archi. Tali configurazioni sono pensate per il caso più generale di grado alto, in cui i nodi vengono rappresentati da scatole.

Tali configurazioni sono:

- svolte a U (vedi Figura 4.60a): si cerca di vedere se il segmento di mezzo si possa avvicinare agli estremi; questo diminuisce il numero di incroci e la lunghezza totale degli archi;
- nodi di grado 2 'messi male' (vedi Figura 4.60b); si cerca di ridistribuire le svolte, posizionandovi i nodi se possibile, altrimenti i nodi vengono messi in modo che le lunghezze degli archi siano il più bilanciate possibile;
- auto-incroci, tra archi uscenti dallo stesso nodo (vedi Figura 4.60c): si risolvono invertendo l'ordine degli archi coinvolti;
- vertici 'strozzati' (vedi Figura 4.60d): sono quelli che hanno un solo nodo adiacente, posizionato lontano da loro; vengono mossi il più vicino possibile al loro nodo adiacente.

La complessità di un algoritmo di aggiustamento che elimina tutte le configurazioni ora descritte è $O(n+m)$.

Figura 4.60.



Un approccio più complicato e meno efficiente dal punto di vista computazionale è fornito da Foßmeier, Heß e Kaufmann [FHK98], che considerano anche la possibilità di modificare le dimensioni delle scatole per garantire un disegno più piccolo e con meno svolte. Questo algoritmo, a seconda della variante implementata, ha complessità $O(n^2)$ oppure $O(n \log n)$.

4.5.2. Compattamento dell'area

Riferimenti: [Dal99] pp. 144-145, 151-154 e 157-161.

Prima di dettagliare l'algoritmo che minimizza l'area di un disegno ortogonale su griglia, diamo delle definizioni che aiuteranno a capire il seguito.

Definizione. Una *rete* G è un grafo orientato tale che:
- ogni *sorgente (pozzo)* v ha una *produzione (consumazione)* $\sigma(v)$; la somma delle produzioni delle sorgenti è uguale alla somma delle consumazioni dei pozzi;
- ogni arco (u,v) ha le seguenti etichette: lim.inf. $\lambda(u,v)$, capacità $\mu(u,v)$, costo $\chi(u,v)$.

Un *flusso* ϕ nella rete G associa ad ogni arco (u,v) di G *un intero* $\phi(u,v)$ tale che:
- $\phi(u,v)$ non può eccedere la capacità $\mu(u,v)$,
- $\phi(u,v)$ non può essere meno del lim.inf. $\lambda(u,v)$,
- per ogni nodo non pozzo ne' sorgente la somma dei flussi entranti deve essere uguale alla somma dei flussi uscenti.

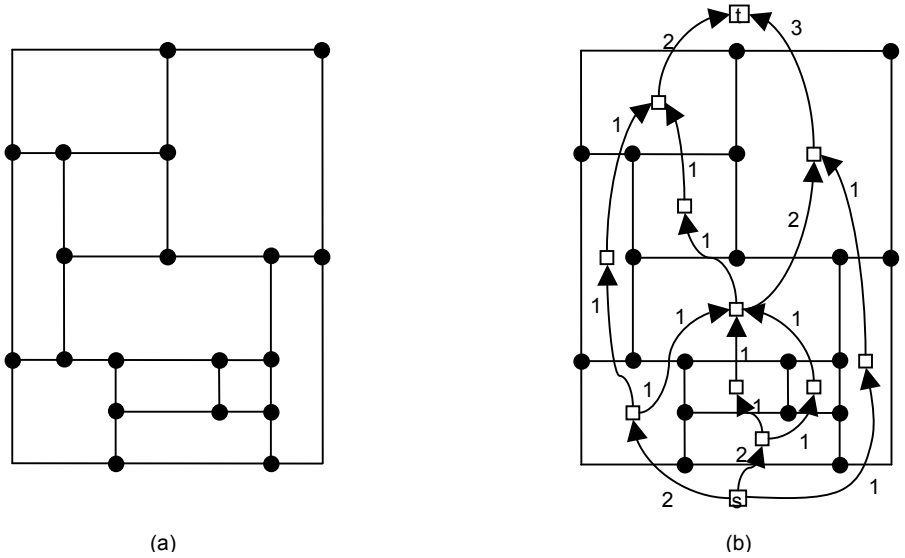
Il *costo* del flusso ϕ è la somma di $\chi(u,v)\ \phi(u,v)$ su tutti gli archi. Il *valore* del flusso ϕ è la somma dei flussi che raggiungono tutti i pozzi.

Il *problema del flusso di costo minimo* è il seguente: data una rete G, trovare un flusso ϕ tale che il suo costo sia minimo.

Descriviamo ora l'algoritmo di compattamento dell'area il quale tratta separatamente segmenti orizzontali e verticali. L'input di tale algoritmo è il disegno 2D ortogonale su griglia piano di un grafo planare di massimo grado 4 (ad esempio l'output dell'algoritmo basato sulla rappresentazione di visibilità), e il suo output è un disegno con le stesse convenzioni di area minima.

Cominciamo la descrizione nel caso speciale in cui ogni faccia abbia forma rettangolare, e poi consideriamo il caso generale.

Figura 4.61.



L'algoritmo che segue usa il problema del flusso di costo minimo per compattare l'area, costruendo due reti di flusso, una per i segmenti orizzontali, NH, ed una per quelli verticali, NV. Definiamo la rete NH, e poi la rete NV sarà definita in modo analogo. La rete NH ha un nodo associato ad ogni faccia interna più due nodi speciali, denotati con s e t, che rappresentano la parte alta e la parte bassa della faccia esterna. Inoltre, NH ha un arco (f,g) per ogni coppia di facce f e g che condividono un segmento orizzontale e, con f sotto g. Il flusso nell'arco (f,g) rappresenterà la lunghezza del segmento e. Perciò, l'arco (f,g) ha un lim.inf. $\lambda(f,g)=1$, capacità $\mu(f,g)=+\infty$, e costo $\chi(f,g)=1$. La Figura 4.61 mostra una rappresentazione piana di un grafo (Figura 4.61a) e la corrispondente rete NH (Figura 4.61b) in cui sia stato calcolato un flusso minimo.

E' immediato vedere che la rete NH è piana e aciclica, con un'unica sorgente ed un unico pozzo, entrambi giacenti sulla faccia esterna. Questo implica che NH sia un st-grafo con $O(n)$ nodi ed archi.

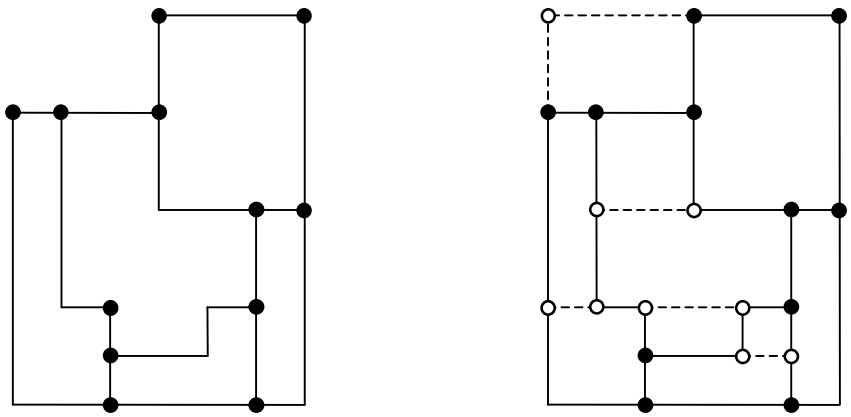
L'algoritmo di compattamento è il seguente:
ALGORTIMO CompattaFacceRettangolari
Input: una rappresentazione piana H ortogonale su griglia con tutte le facce rettangolari del grafo G
Output: una rappresentazione piana Γ ortogonale su griglia con altezza e larghezza minima e con minima lunghezza degli archi di G

1. Costruisci le reti di flusso NH ed NV associate alla rappresentazione H di G
2. Calcola i flussi minimi in NH ed NV
3. Setta le lunghezze di ogni segmento di H uguale al flusso nel corrispondente arco di NH o NV

Si osservi che, dato un disegno ortogonale di G, è possibile calcolare un flusso per NH (ed NV) settando il flusso di ogni arco pari alla lunghezza del suo segmento associato. Segue che il valore del flusso di NH (NV) è esattamente uguale alla larghezza (altezza) del disegno e che la somma dei costi dei flussi di NH e di NV è pari alla lunghezza totale degli archi del disegno. Formalizzando, si ha il seguente:

Lemma. *Dati dei flussi per le reti NH ed NV, settata la lunghezza di ogni segmento della rappresentazione piana H uguale al flusso dell'arco corrispondente in NH o NV, si ottiene un disegno ortogonale piano Γ avente larghezza (altezza) pari al valore del flusso di NH (NV) e lunghezza totale degli archi pari alla somma dei costi dei flussi in NH ed NV.*

Figura 4.62.



Per il lemma precedente, il problema del compattamento di una rappresentazione ortogonale con facce rettangolari può essere ricondotto al problema del minimo flusso in una rete, e formalizzato nell'algoritmo CompattaFacceRettangolari. La complessità di tale algoritmo è $O(n^{7/4} \log n)$, e si calcola tenendo conto che i passi 1. e 3. costano $O(n)$ mentre il passo 2 costa $O(n^{7/4} \log n)$, come mostrato in [GT96].

A questo punto, mostriamo come sia possibile compattare rappresentazioni ortogonali generali, cioè le cui facce non siano necessariamente ortogonali. Modifichiamo la rappresentazione ortogonale aggiungendo degli archi e nodi fittizi, in modo tale che il risultato sia una rappresentazione con facce rettangolari, e poi applichiamo l'algoritmo precedente.

Sia G un grafo planare e sia H la sua rappresentazione ortogonale. Definiamo una *rifinitura rettangolare di H* una rappresentazione ortogonale H' di un grafo G' tale che (vedi Figura 4.62):

- G' si ottiene da G tramite una sequenza di operazioni tra le seguenti:
  - Aggiungi un nodo isolato
  - Aggiungi un nodo lungo un arco
  - Aggiungi un arco
- Il grafo G rappresentato all'interno di H' è esattamente H
- Le regioni di H' sono rettangoli.

Dimostreremo ora che una rifinitura rettangolare esiste sempre e che può essere calcolata in $O(n)$ tempo.

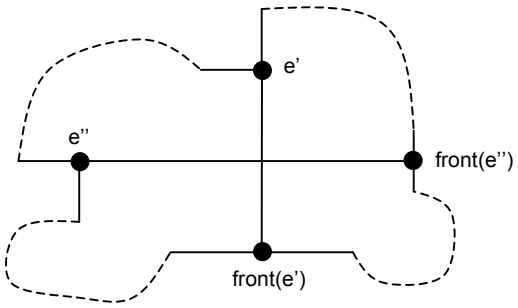
La costruzione di H' comincia inserendo un nodo in corrispondenza di ciascuna svolta di H non sulla faccia esterna; poi, considera le facce una per una. Sia f una qualunque faccia interna; se f ha forma rettangolare non dobbiamo fare nulla, altrimenti procediamo al modo seguente:

1. per ogni arco e di f, si definisca *next(e)* l'arco che segue e percorrendo il contorno di f in senso antiorario, e *corner(e)* il vertice in comune tra e e *next(e)*;
2. per ogni arco e di f, si definisca *turn(e)=1* se e e *next(e)* formano una svolta a sinistra, *turn(e)=0* se sono allineati, e *turn(e)=-1* se formano una svolta a destra;
3. per ogni arco e, si trovi il primo arco e' che segue e in senso antiorario e tale che la somma dei valori *turn* di tutti gli archi tra e (compreso) ed e' (escluso) sia uguale a 1, e si setti *front(e)=e'*;
4. per ogni arco e tale che *turn(e)=-1*, si inserisca un nodo *project(e)* lungo l'arco *front(e)*, e si aggiunga l'arco *extend(e)* tra i nodi *corner(e)* e *project(e)*. Si aggiorni H' imponendo che *extend(e)* non abbia svolte e che e ed *extend(e)* siano allineati. Se *front(e')=front(e'')=e'''* per due archi distinti e' ed e'', allora stabiliamo che *project(e')* segue *project(e'')* in senso antiorario se e solo se e', e'' e *front(e')* formano una sequenza in senso antiorario.

L'algoritmo appena descritto "rifinisce" le facce interne in modo che divengano facce rettangolari. Un esempio del suo funzionamento è mostrato in Figura 4.63.

In ultimo, viene completata anche la faccia esterna, come verrà definito in seguito.

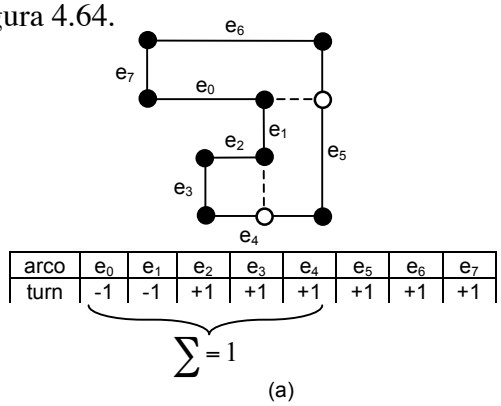
Figura 4.63.



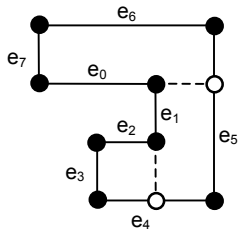
Teorema. *L'algoritmo precedente funziona correttamente, cioè la rappresentazione H' può sempre essere prodotta ed è sempre una rappresentazione piana.*

Dimostrazione. Innanzi tutto, si osservi che *front(e)* è definito per ogni *e* con *turn(e)=-1* grazie alle proprietà degli angoli interni di un poligono (se *e* ha una svolta a destra, vuol dire che sta definendo una concavità della faccia, e quindi devono seguire almeno 4 svolte a sinistra). Si assuma ora, per assurdo, che due nuovi archi *extend(e')* ed *extend(e'')* si incrocino, cioè procedendo in senso antiorario incontriamo, nell'ordine, *e'*, *e''*, *front(e')* e *front(e'')* – vedi figura 4.64. Questo non può accadere poiché contraddice la definizione di *front*. **CVD**

Figura 4.64.



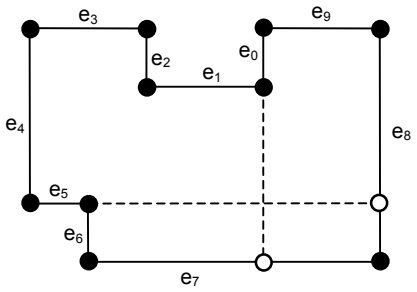
(a)

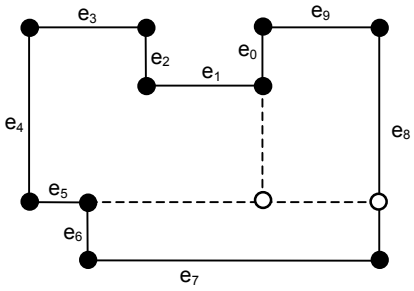


(b)

Esempio. Per capire meglio, osserviamo la Figura 4.65: e_7 sembra $front(e_0)$ ma non è così: in realtà $front(e_0)=e_5$.

Figura 4.65.





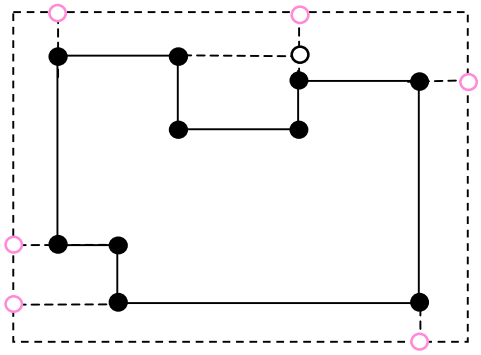
arco	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_8
turn	-1	-1	+1	+1	+1	-1	+1	+1	+1	+1

(a)

(b)

La rifinitura della faccia esterna si può fare usando l'algoritmo precedente con una leggera variazione: ora che *front(e)* potrebbe non essere definito per ogni *e*, per gestire i soli archi che hanno il *front* indefinito, aggiungiamo un rettangolo intorno alla faccia esterna ed allunghiamo questi archi proiettandoli sui lati del rettangolo (vedi Figura 4.66). Osserviamo che l'unica differenza tra la faccia esterna e le altre facce è il verso di percorrenza: girando in senso antiorario la faccia rimane a sinistra; per mantenere anche la faccia esterna alla sinistra è necessario percorrere il suo contorno in senso orario.

Figura 4.66.



E' facile vedere che la rappresentazione *H'* ha $O(n+b)$ vertici, dove *b* è il numero di svolte di *H*. Inoltre, l'algoritmo di rifinitura può essere eseguito in $O(n+b)$ tempo mantenendo i nodi incontrati nell'attraversamento di ciascun contorno di faccia in una coda. Combinando i risultati appena ottenuti con quelli dell'algoritmo precedente si ha:

Teorema. *Dato un grafo G tramite una sua rappresentazione ortogonale H con b svolte, si può costruire un'altra sua rappresentazione piana con area* $O((n+b)^2)$ *in tempo* $O(n^{7/4} \log n)$.

RIFERIMENTI BIBLIOGRAFICI

[BK94] T.C. Biedl, G. Kant: A Better Heuristic for Orthogonal Graph Drawing. *Proc. 2nd European Symp. On Algorithms (ESA '94)*, Lecture Notes in Computer Science 855 24-35, 1994. (anche UU-CS-1995-04).

[Bal97] T. Biedl, T. Shermer, S. Whitesides, S. Wismath: Orthogonal 3-D Graph Drawing, Proc. *Graph Drawing (GD'97)*, LNCS 1353, pp. 76-86, 1997.

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

[DT90] G. Di Battista, R. Tamassia: On-Line Graph Algorithms with SPQR-Trees, *Proc. 17th ICALP* Lecture Notes in Computer Science 442, 598-611, 1990.

[DS87] P.F. Dietz, D.D. Sleator: Two Algorithms for maintaining order in a list, *Proc. 19th Annual ACM Symp. Of Theory of Comput. (STOC '87)*, 365-372, 1987.

[FHK98] U. Fößmeier, U. Heß, M. Kaufmann. On improving orthogonal drawings: the 4M algorithm. *Proc. GD'98* Lecture Notes in Computer Science 1547, 125-137, 1998.

[FHW95] S. Fekete, M. Houle, S. Whitesides. New results on a visibility representation of graphs in 3d. *Proc. GD'95* Lecture Notes in Computer Science 1027, 234-241, 1995.

[GT94] A. Garg, R. Tamassia: On the Computational Complexity of Upward and Rectilinear Planarity Testing. *Proc. GD'94* Lecture Notes in Computer Science, 894, pp. 286-297, 1995.

[GT96] A. Garg, R. Tamassia: A new minimum cost flow algorithm with applications to graph grawing. *Proc. GD'96* Lecture Notes in Computer Science, 1997.

[GRS80] R. Graham, B.L. Rothschild, J. Spencer: *Ramsey Theory*. John Wiley, 1980.

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

[SKT98] J.M. Six, K.G. Kakoulis, I.G. Tollis. Refinement of Orthogonal Graph Drawing. Proc. GD '98 Lecture Notes in Computer Science 1547, 302-315, 1998.

[T87] R. Tamassia: On Embedding a Graph in the Grid with the Minimum Number of Bends. *SIAM J. Comput.* **16(3)**, 421-444, 1987.

[Y89] M. Yannakakis: Embedding planar graphs in four pages. *18th Annual ACM Symposium on Theory of Computing, J. Comput. System Sci.* **38(1)**, 36-67, 1989.