UNIVERSITÀ DEGLI STUDI ROMA TRE
FACOLTÀ DI SCIENZE M.F.N.
CORSO DI LAUREA IN MATEMATICA

Tesi di Laurea Magistrale in Matematica

Modelli e algoritmi per il Graph Drawing

Sintesi

Candidato
Stefano Spensieri

Relatore
Prof. Marco Liverani

ANNO ACCADEMICO 2008-2009
Ottobre 2009

Classificazione AMS: 05C10, 05C05, 05C62, 90C27, 90C35, 68W35, 68W27.

Parole chiave: teoria dei grafi, graph drawing, planaritá, disegno ortogonale, algoritmi.

Modelli e algoritmi per il Graph Drawing

Un *grafo* é spesso usato come un modello astratto per descrivere insiemi di oggetti e le relazioni fra di essi. Utilizzando un grafo per costruire il modello, identificheremo gli oggetti con i *vertici* del grafo e le "relazioni" o i "collegamenti" tra gli oggetti con gli *spigoli* del grafo.

L'alto grado di astrazione della struttura la rende adatta a modelizzare la realtá e a codificare informazioni in numerosi ambiti; la stessa natura astratta solleva peró il problema di come rendere fruibili le informazioni cosí codificate.

Appare naturale richiedere che le informazioni codificate nel grafo possano essere visualizzate, ricorrendo a simboli e vari formalismi grafici: in questa tesi affronteremo l'argomento del *Graph Drawing*, ovvero la rappresentazione grafica dell'entità astratta *grafo*.

A questo proposito ci sembra molto efficace la seguente affermazione, tratta da [59]: *"Graph drawing is the art of drawing graphs in such a way that the relationship between the objects (in the graph) are easily understood by looking at the picture."*.

L'argomento ha conosciuto uno sviluppo impetuoso negli ultimi decenni, grazie soprattutto al crescente utilizzo di tecniche di disegno informatiche (*CAD – Computer Aided Design*) e la conseguente richiesta di algoritmi che automatizzassero la produzione dei disegni.

Sono numerosi i punti di contatto con altre discipline e aree di studio, quali ad esempio l'*Information Visualization*, lo studio della percezione delle immagini, la topologia e la geometria dei grafi, la bioinformatica, la progettazione di interfacce per il disegno assistito.

In questa tesi ci siamo occupati dello studio di algoritmi per la produzione au-

tomatica di *layout* specifici, ovvero *layout* che tengano conto di uno o piú vincoli "estetici".

Disegno e costruzione di grafi

Sono stati sviluppati numerosi modelli generali e formalismi grafici specifici per la produzione di disegni; ad esempio nel modello *straight line* si richiede che gli spigoli non effettuino pieghe, mentre nel modello *gerarchico* si richiede di rispettare una scansione in livelli orizzontali che espliciti relazioni non paritarie fra i vertici.

Fra i criteri estetici piú comuni (per la loro indubbia funzionalitá) troviamo la minimizzazione dell'area, del numero di intersezioni fra spigoli, del numero di pieghe sugli spigoli, la risoluzione angolare, l'uniformitá della lunghezza degli spigoli o la loro separazione (massimizzazione della distanza minima fra spigoli che non si intersecano).

Disegni che si attengano ad alcuni di questi vincoli risultano essere non solo piacevoli alla vista, ma anche quelli che meglio si adattano a risolvere problemi applicativi reali, o a illustrare in maniera chiara le informazioni: un disegno ortogonale con poche pieghe e intersezioni fra spigoli é adatto per la fabbricazione di circuiti stampati, mentre un disegno gerarchico con spigoli poligonali e privo di intersezioni modella in maniera soddisfacente un diagramma di flusso o un diagramma Entitá/Relazioni di una base di dati.

Ulteriori applicazioni del *graph drawing* sono la visualizzazione di mappe, di reti di telecomunicazioni, di *Social Network*, di strutture industriali complesse, la bioinformatica, la modellizzazione grafica di progetti software, la visualizzazione di archivi di *file system*.

Alcune tematiche parallele allo sviluppo di questi modelli sono l'etichettatura dei vertici o degli spigoli, l'uso del colore come strumento di esposizione delle informazioni, il disegno interattivo, il disegno su superfici distinte dal piano, anche con differenti proprietá topologiche.

Le differenti finalità applicative accennate evidenziano, fra le procedure per produrre layout di grafi, una possibile distinzione in *disegno* e *costruzione* di un grafo.

Il *disegno* di un grafo è la trasposizione grafica, necessariamente in due dimensioni, dei vertici e degli spigoli, di solito sotto forma rispettivamente di punti e curve in $\mathbb{R}^2$. Operata questa trasposizione, il layout risultante sarà visualizzabile sul terminale

grafico di un computer o su carta; dato il contesto, si porrà dunque attenzione a parametri estetici e vincoli formali che possano migliorare l'efficacia del disegno, dove per "disegno efficace" intendiamo un disegno comprensibile a colpo d'occhio.

La *costruzione* è la mappatura dei vertici all'interno di uno spazio in due o tre dimensioni; i vertici possono essere puntiformi o avere assegnata un'area o un volume, gli spigoli sono sempre rappresentati come curve (nel piano o nello spazio). L'obbiettivo è ottenere una struttura fisicamente costruibile, dunque i parametri estetici perdono di significato, mentre ne acquistano i parametri fisici: si richiedono costruzioni con volume contenuto, o che minimizzino il volume dei vertici o la lunghezza degli spigoli, o che richiedano un basso numero totale di pieghe nel percorso degli spigoli. A seconda del modello utilizzato, la costruzione deve attenersi a vincoli concreti, quali ad esempio l'assenza di intersezioni fra spigoli e vertici, l'ortogonalità fra gli spigoli o l'assenza di pieghe sul percorso degli spigoli. Uno degli ambiti applicativi piú discussi per la costruzione di grafi é la progettazione di circuiti elettronici VLSI.

L'approccio algoritmico

Ciascun problema di ricerca di un layout con determinati vincoli estetici puó essere tradotto come problema di ottimizzazione combinatoria, ove si consideri:

- *un insieme delle soluzioni ammissibili A*, coincidente con l'insieme di tutti i layout che soddisfano le convenzioni grafiche del modello in esame, tenendo conto dei vincoli del particolare layout che si intende ottenere;
- *una funzione obbiettivo f*, funzione che abbia come variabili uno o più parametri "estetici" da ottimizzare.

Nei casi in cui i problemi non siano risolubili con approcci diretti, si ricorre a strategie approssimanti, euristiche, o si riformulano i problemi in termini di problemi di ricerca: è spesso più utile avere algoritmi costruttivi validi e veloci, applicabili in generale, che garantiscano un risultato non ottimo, che strategie di ottimizzazione fortemente dipendenti da vincoli sulla natura del grafo in esame.

L'equivalenza fra produzione di *layout* con elevata "resa estetica" e risoluzione di problemi di ottimizzazione combinatoria permette di sfruttare numerosi risultati teorici sulla complessitá computazionale e sulla trattabilitá di classi di problemi, come punto di partenza per la produzione di algoritmi mirati.

Nel 1994 Di Battista et al. hanno presentato una bibliografia annotata [27] con la classificazione di piú di 300 articoli sul *graph drawing*, in massima parte riguardanti algoritmi per la produzione di disegni vincolati per differenti classi di grafi; va sottolineato che molti degli algoritmi affrontano problemi per classi ristrette di grafi (ad esempio grafi 3-connessi 3-regolari, *st-graph* o grafi con grado massimo fissato) e gran parte dei risultati riguarda la classe dei grafi planari e il problema dell'immersione planare.

Gli stessi autori hanno pubblicato un libro [28] nel quale si integrano gli algoritmi del precedente lavoro e se ne propone una classificazione. Negli ultimi anni risultati con validitá piú generale e algoritmi per trattare classi di grafi via via piú ampie hanno affiancato i precedenti.

Nella tesi abbiamo trattato risultati teorici e algoritmi per la costruzione di *layout* vincolati il piú generali possibile, pur senza trascurare tematiche rilevanti come la planaritá.

In particolare abbiamo affrontato tre classi di grafo: alberi, grafi planari, grafi generici.

Il problema della planaritá é stato trattato insieme a quello del *simoultaneous embedding*, ovvero la visualizzazione contemporanea di coppie di grafi che condividano parte dei vertici, realizzata in modo da essere armoniosa, non intricata e dunque fruibile.

Per gli alberi abbiamo studiato due differenti modelli per il disegno; altrettanti ne abbiamo affrontati per i grafi generici. Per entrambe le classi abbiamo presentato numerosi algoritmi noti in letteratura, assieme ad alcune recenti varianti di algoritmi classici. Abbiamo avuto modo di introdurre anche delle migliorie in alcuni algoritmi, in singoli passi o nella stima di alcune costanti funzionali all'ottimizzazione della resa estetica della resa estetica.

Il disegno di alberi

Gli alberi sono una classe di grafi per cui i più comuni problemi di ottimizzazione legati alla rappresentazione grafica di determinati aspetti della figura (ad esempio la minimizzazione del numero di pieghe su ciascuno spigolo, o la minimizzazione del numero di intersezioni fra gli spigoli) perdono di significato: è immediato disegnare un albero con spigoli privi di pieghe (*straight line*) senza che effettuino intersezioni

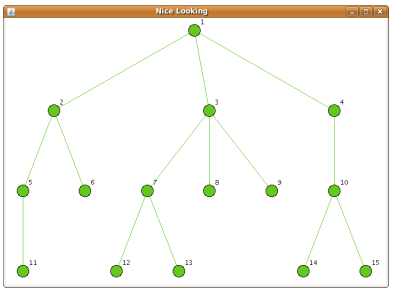

Figura 1: un albero disegnato con l'algoritmo NICELOOKING.

fra di loro. I principali obbiettivi che restano perseguibili sono rendere uniforme la lunghezza degli spigoli, massimizzare l'ampiezza degli angoli formati da due spigoli incidenti nello stesso vertice (risoluzione angolare), conservare una distanza minima fra i vertici.

Il primo dei due modelli per il disegno di alberi (radicati) che abbiamo analizzato é il modello **a gerarchia verticale**: la relazione "padre-figlio" é esplicitata ponendo il padre in posizione superiore rispetto al figlio, generando in questo modo una scansione dei vertici per righe orizzontali (*livelli*).

Vengono individuati due criteri che garantiscono un disegno efficace: i figli devono trovarsi vicino al padre, meglio se in posizione simmetrica, deve essere mantenuta una distanza minima fra vertici dello stesso livello. I due algoritmi *naive* che abbiamo presentato producono disegni in maniera rapida e soddisfano ciascuno un differente criterio. Entrambi gli algoritmi peró falliscono nel perseguire contemporaneamente i due criteri. Per questo motivo abbiamo introdotto un algoritmo presentato in [13], detto NICELOOKINGTREE, che riesce a perseguire entrambi gli obbiettvi, realizzando un disegno vincolato con area ottima (Fig. 1).

Il secondo modello é il modello **Parent-Centered** (PC), nel quale la relazione padre figlio é formalizzata dalla seguente convenzione: i figli di ciascun vertice v sono disegnati lungo una circonferenza c_v centrata in v, con raggio r_v; il raggio di tale circonferenza e la sua divisione in settori da assegnare a ciascun vertice figlio sono variabili; gli algoritmi che presentiamo determinano queste variabili in modo da ottenere disegni in qualche senso ottimali.

In questo ambito abbiamo presentato tre noti algoritmi, distinti ma strettamente correlati fra loro, tanto che si potrebbe affermare che ciascuno costituisce un miglio-

ramento del precedente; il primo utilizza solo una parte della circonferenza c_v per disegnare i figli di v, piú precisamente utilizza la parte esterna alla circonferenza sulla quale, a sua volta, é posizionato v. Tale approccio stimola una migliore comprensione della relazione padre-figlio e, soprattutto, permette di modificare il layout in maniera dinamica (non computazionalmente onerosa) e interattiva (comprensibile per chi assiste al cambio): selezionando arbitrariamente un vertice come nuova radice dell'albero, il layout muterá fino a portare la nuova radice al centro del sistema di riferimento, ricreando un layout con le stesse restrizioni estetiche dell'originale; la mutazione si svolge tramite una successione di "fotogrammi" e durante l'intero processo non si verificano sovrapposizioni fra vertici o intersezioni fra spigoli, permettendo all'utente di orientarsi; tale caratteristica é di grande rilevanza pratica per la "navigazione" di grandi strutture (ad esempio visualizzazzione e navigazione del WWW).

L'algoritmo BALLOONFRACTAL é basato sull'equipartizione della circonferenza c_v di un vertice v fra i suoi figli $s_1, \ldots, s_h$ e la conseguente determinazione di raggi r_{s_i} identici per tutti i vertici fratelli s_i; il raggio r_{s_i} é una funzione del raggio r_v; originariamente si definiva una costante di contrazione globale, ma questo approccio é sconsigliabile, in quanto non garantisce in generale l'assenza di intersezioni; noi proponiamo una semplice funzione $r_{s_i} = r_v \cos(\frac{\pi}{h})/2$, che usa lo spazio in maniera migliore e preserva dlle intersezioni (Fig. 2).

L'algoritmo BALLOONSNS (*SNS – Subtreees of Nonuniform Size*) interviene su una pecca del precedente, ovvero quella di dividere equamente la circonferenza fra i vertici figli s_i, indipendentemente da quanto fossero grandi i sottoalberi con radice nel singolo s_i. definisce i settori circolari in modo che siano proporzionali alla dimensione dei sottoalberi dei vertici che li occupano; alcuni esempi evidenziano il consistente miglioramento in termini di utilizzo degli spazi che questo approccio comporta (Fig. 3). Infine, un riarrangiamento dell'ordine dei figli attorno al vertice padre permette di costruire alberi con risoluzione angolare e *aspect-ratio* angolare ottimali. Anche in questo caso é stato possibile precisare la definizione di alcune quantitá utilizzate nell'implementazione dell'algoritmo.

Disegno di grafi

Per i grafi generici proponiamo due modelli di disegno "antipodali": il disegno ortogonale a griglia e le strategie *force-directed*.

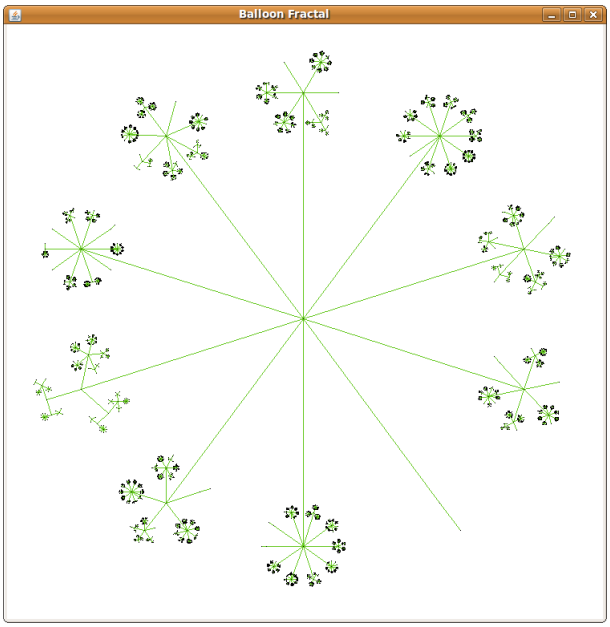

Figura 2: un albero con più di 8.000 vertici disegnato con l'algoritmo BALLOONFRACTAL.

Nel **modello di disegno ortogonale** (OD) le posizioni dei vertici sono vincolate a coordinate intere, come pure le pieghe degli spigoli, i cui percorsi sono soggetti a restrizioni. Molti algoritmi in questo modello prevedono fasi di ordinamento dei vertici e degli spigoli, che sono strategiche nel limitare le intersezioni, le pieghe degli spigoli o la dimensione dei box. Gli obbiettivi comunemente perseguiti dagli algoritmi che adottano il modello di disegno ortogonale sono minimizzazione dell'area, minimizzazione del numero di pieghe, minimizzazione degli incroci fra spigoli. Per tale modello abbiamo presentato un algoritmo *naive*, dalla cui analisi si evidenzia l'importanza di una fase di preordinamento dei vertici per risparmiare spazio e stimare l'area in maniera migliore. Tale fase di ordinamento (*st-ordering*) é implementata invece in uno dei due algoritmi presentati successivamente, che pertanto raggiunge un'area minore di $\frac{m+n}{2} \times \frac{m+n}{2}$ (n e m sono rispettivamente il numero di vertici e spigoli del grafo), con un disegno in cui ciascuno spigolo piega una sola volta. Una modifica di questo algoritmo permette l'implementazione di un algoritmo, ORTOGONALE, per la produzione di *layout incrementali*, ovvero tali da poter essere modificati con l'aggiunta e l'eliminazione di vertici o spigoli; in particolare non é necessario conoscere a priori la struttura del grafo: ciascun vertice puó essere esplicitato nell'algoritmo solo al momento in cui deve essere disegnato. Lo sviluppo di algoritmi che implementino modifiche puntuali

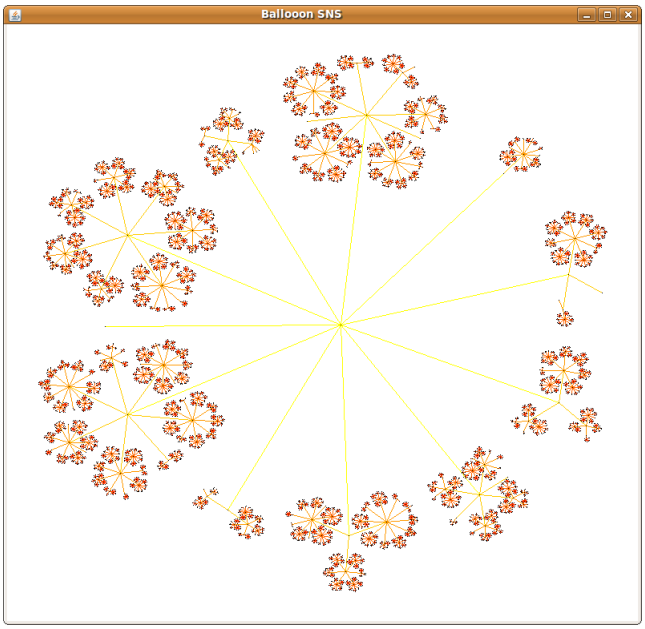

Figura 3: un albero con più di 8.000 vertici disegnato con l'algoritmo BALLOONSNS.

al *layout* del grafo in poche operazioni é studiato tutt'oggi da numerosi ricercatori e costituisce un terreno di indagine scientifica piuttosto fecondo.

Il secondo modello di cui ci siamo occupati nell'ambito dello studio dei metodi per il disegno dei grafi generici é basato sulle cosiddette **strategie force directed** (FD): l'idea del metodo é quella di assimilare la struttura grafo ad un sistema fisico, in cui i vertici sono punti materiali soggetti a forze di vario genere; le coordinate dei vertici (dunque in definitiva il *layout*) derivano dalla ricerca di una configurazione di equilibrio per il sistema fisico modellizzato.

Il metodo è molto popolare per la sua intuitività, per la facilità di implementazione (anche se si richiede una mole di calcoli superiore a molti altri algoritmi per il graph drawing) e per le possibilità di interazione da parte dell'osservatore che, modificando pochi parametri, può ottenere layout molto differenti.

In questo modello non si tiene in nessun conto l'area del disegno, né si considerano gli incroci fra spigoli; inoltre i vertici hanno coordinate reali e subiscono spostamenti non discreti durante l'esecuzione degli algoritmi. I vertici sono sempre puntiformi (salvo nel caso in cui non sia diversamente specificato) e gli spigoli sono sempre *straight-line*, dunque non hanno senso parametri legati a questi aspetti (*aspect-ratio*, area del box, numero pieghe, ecc.). Ciò che viene considerato sono le "relazioni" fra

coppie o gruppi di vertici: nel modello *force-directed* assume rilevanza la distanza fra coppie di vertici nel disegno, oppure la presenza di sottografi isomorfi, che si intende sottolineare disegnandoli con lo stesso layout (eventualmente affiancati). Infine un obbiettivo molto utile nella pratica è quello di produrre un layout "clusterizzato", ovvero che evidenzi sottografi fortemente connessi raggruppando i vertici che li compongono.

Una strategia *force directed* si compone di due fasi:

1. modellizzazione: partendo dalla scelta delle caratteristiche che si vogliono esaltare nel layout, si studia un opportuno modello fisico, assegnando conseguentemente forze attrattivo-repulsive a tutte le coppie di vertici e eventuali campi di forza indipendenti dai vertici;

2. ricerca della configurazione di equilibrio: fissato il sistema di forze, partendo da una configurazione iniziale in cui le posizioni sono assegnate approssimativamente o in maniera casuale, si procede attraverso varie iterazioni alla ricerca di una configurazione che minimizzi l'energia totale del sistema.

La definizione delle forze avviene, come accennato, in funzione delle caratteristiche che si vogliono mettere in risalto; ad esempio nell'algoritmo BARYCENTRIC si vuole ottenere un disegno convesso, che massimizzi le simmetrie: l'obbiettivo viene raggiunto fissando le posizioni di un ristretto numero di vertici, e iterando un procedimento che assegna ad ogni vertice la posizione media fra quella dei suoi vertici adiacenti. Abbiamo quindi presentato un famoso approccio *FD*, noto come metodo *spring embedder*; nella versione originale di tale metodo [38] si utilizza una forza (repulsiva) di natura elettromagnetica fra tutte le coppie di vertici e una forza attrattiva elastica fra vertici adiacenti.

Due ulteriori varianti del metodo affrontano il tema dell'efficienza computazionale, centrale nelle strategie *FD*: la prima variante utilizza un approccio per livelli, che prevede la sistemazione di sottografi (*cluster*) di dimensione via via minore; ogni sottografo viene gestito come un unico vertice puntiforme e fornisce una posizione orientativa per i "*sotto-cluster*" che lo compongono. Il secondo metodo prevede l'utilizzo di una funzione detta *stress-function*, che maggiora l'energia totale del sistema, alla quale é possibile applicare un metodo iterativo di ricerca del minimo ($O(n^2)$ operazioni).

Disegno di grafi in tre dimensioni

Il passaggio alla terza dimensione apre nuove prospettive per il disegno di grafi; sará piú facile disegnare grafi *straight line* privi di intersezioni fra spigoli, in un modello con vertci puntiformi.

Il compito diventa banale qualora si ammettano vertici con assegnato un volume (modello *box-drawing*).

Esistono vari modelli per il disegno in tre dimensioni e noi ci siamo occupati di quello in assoluto piú diffuso per l'abbondanza di applicazioni pratiche e la rilevanza dei problemi di ottimizzazione combinatoria ad esso legati; il modello analizzato é il modello *ortogonale a griglia*, nel quale i *box* rappresentanti i vertici e le poligonali rappresentanti gli spigoli sono costituiti da segmenti paralleli agli assi, i cui estremi abbiano coordinate intere.

Abbiamo analizzato algoritmi il cui obbiettivo è ottimizzare rispettivamente il volume del bounding box, una delle sue dimensioni, il numero di pieghe degli spigoli, tenendo a volte conto di caratteristiche estetiche aggiuntive quali l'*aspect-ratio* (armonia fra le dimensioni dei *box*) e *restrizione al grado* (proporzionalitá fra superficie del *box* e grado del vertice che rappresenta). É chiaro che tali obbiettivi non sono perseguibili contemporaneamente, e questa impossibilità ha portato allo sviluppo di una varietà di algoritmi multiobbiettivo (per riassumere quelli da noi trattati, presentiamo in appendice una tabella).

Riguardo all'obbiettivo di minimizzare il volume del disegno, in [12] si dimostrano limiti inferiori sotto varie restrizioni e per ognuno dei limiti si propone un algoritmo che fornisce un disegno con volume asintoticamente ottimo.

Il primo algoritmo di cui ci siamo occupati è noto come *Optimal Volume Cube Drawing* e realizza un disegno con vertici cubici (*aspect ratio* $= 1$) e superficie limitata (12-ristretti al grado). L'algoritmo effettua $O(m^{3/2})$ operazioni.

L'altro algoritmo *Optimal Volume Box Drawing* elimina le restrizioni estetiche e si concentra nel limitare il volume; l'algoritmo é basato sul posizionamento (nel piano) dei vertici secondo differenti strategie, a seconda che il grado del vertice sia elevato ($\delta(v) \geq 4m/\sqrt{n}$) o meno. Le strategie di posizionamento sono fondamentali nel limitare l'altezza del disegno, dunque il volume, al momento di passare alla terza dimensione. Gli spigoli in entrambi gli algoritmi sono tracciati secondo una strategia nota come *Lifting Half Edge* (LHE), che consiste nello scomporre lo spigolo in due

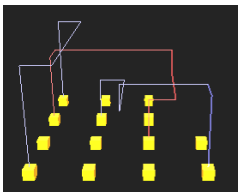

Figura 4: la strategia *lifting-half-edge*.

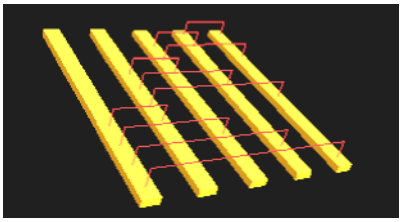

Figura 5: un grafo disegnato con l'algoritmo DOPPIOLIVELLO.

segmenti principali (orizzontale e verticale) e vari raccordi per congiungerne gli estremi; i due segmenti principali occuperanno due Z-piani distinti, evitando le intersezioni (Fig. 4).

Riguardo all'obbiettivo di fissare il numero di livelli abbiamo presentato numerosi risultati distinti, fra i quali spiccano una strategia generale e due algoritmi dinamici, ovvero con *layout* modificabili in tempo costante. La strategia consiste nel prendere un disegno ortogonale bidimensionale con spigoli tracciati con una sola piega (ad esempio uno degli algoritmi della sezione dedicata) e aggiungervi uno Z-piano, aumentando l'altezza dei *box* bidimensionali a 2. Gli spigoli originali vengono ritracciati secondo la strategia *LHE*, producendo un disegno su due livelli privo di incroci.

L'algoritmo, chiamato *DoppioLivello*, approccia il problema sovradimensionando i *box*, in modo da garantire percorsi per gli spigoli privi di intersezioni e con solo due pieghe; tutto questo utilizzando appena due livelli per il disegno: sul piano $Z = 0$ si affiancano n *box* di dimensioni $1 \times m \times 1$; sul piano $Z = 1$ corrono, perpendicolarmente ai vertici, gli spigoli (Fig. 5). Ognuno degli m spigoli disporrá di una Y-retta specifica, garantendo in questo modo l'assenza di intesezioni. Nella tesi osserviamo come, ammettendo un aumento del volume del disegno da $n \times m \times 2$ a $(m+n) \times n \times 2$, sia possibile ottenere un disegno completamente dinamico.

L'algoritmo DYNAMIC utilizza la seguente strategia: i vertici sono punti in posi-

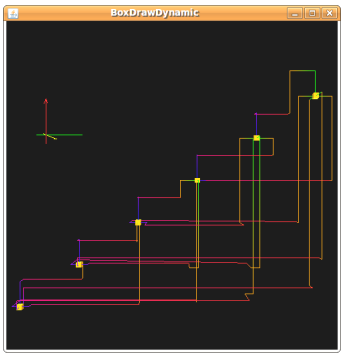

Figura 6: un grafo disegnato con l'algoritmo DYNAMIC.

zione generale nel piano $Y = 0$; ad ognuna delle 6 *porte* (lati del punto o del box cui é possibile "agganciare" spigoli) di un vertice v $(x_-, x_+, y_-, y_+, z_-, z_+,)$ é possibile assegnare uno spigolo (v,w), tracciando il suo percorso fino ad un'altra porta libera di w; ciascun percorso é definito puntualmente a seconda delle porte utilizzate agli estremi (Fig. 6). Ogni vertice é disegnato all'interno di un modulo, una colonna verticale totalmente indipendente dalle altre; la modularitá del disegno é la chiave per modificarlo in tempo costante. Il piú grande limite di questo algoritmo sta nel fatto che puó trattare solo vertici di grado minore o uguale a 6. Nella tesi evidenziamo la possibilitá di generalizzare il procedimento, mantenendo la modularitá e la dinamicitá del disegno, adattandolo a gestire vertici di qualsiasi grado; l'operazione comporta un aumento del volume: il volume iniziale é $6n \times 5 \times 5n$, quello dell'algoritmo generalizzato é stimato da $7\left(\frac{2m+n\Delta}{6}+n\right) \times 5 \times 5\left(\frac{2m+n\Delta}{6}+n\right)$, dove con Δ indichiamo il grado massimo.

Infine abbiamo affrontato il problema di minimizzare il numero di pieghe nel percorso degli spigoli: gli algoritmi presentati in questo ambito sono tre, tutti disegnano il grafo K_n, rispettivamente con 1, 2, 3 pieghe per spigolo; si disegna tale grafo sia perché é un caso limite, sia perché dal suo disegno si ricavano, rimuovendo determinati spigoli, i disegni di grafi generici. Vengono presentati anche risultati teorici quali l'impossibilitá di disegnare grafi generici senza pieghe sugli spigoli e limiti inferiori per il volume dei disegni; in particolare si dimostra che il disegno con 3 pieghe ha volume ottimale.

Progetto Object-Oriented per il disegno di grafi

Lo studio degli algoritmi di *graph drawing* affrontato nel corso di questo lavoro è stato affiancato dallo sviluppo di un pacchetto software originale, in linguaggio *Java*, in grado di offrire un insieme di procedure e di strutture dati per la creazione e manipolazione di grafi, corredate da procedure specifiche che implementano gli algoritmi presentati nei capitoli precedenti.

Abbiamo usato un linguaggio di programmazione *Object Oriented* per l'elevato grado di astrazione di cui questi linguaggi sono dotati, che permette di codificare nozioni matematiche astratte (ad esempio vertici, spigoli, grafi) in maniera indipendente, senza per questo perdere di vista le relazioni che intercorrono fra loro.

Per la visualizzazione grafica in tre dimensioni é stato utilizzato il *package Java 3D* che offre numerose classi per la rappresentazione grafica di una scena tridimensionale.

Il progetto implementa buona parte degli algoritmi introdotti e offre anche metodi per la manipolazione di attributi secondari del disegno. I disegni e le costruzioni di grafi prodotte dai metodi, assieme alla possibilitá di intervenire sulla struttura del grafo, sono un utile supporto alla comprensione del funzionamento degli algoritmi, nei vari modelli trattati.

Bibliografia

[1] M. Bertacca, A. Guidi, *Programmare in Java*, McGraw-Hill, 2006.

[2] P. Bertolazzi, G. Di Battista, C. Mannino, R. Tamassia, *Optimal upward planarity testing of single-source digraphs*, SIAM J. Comput., vol. 22, no. 1, pp. 132–169, (1998).

[3] T. Biedl, *Three approaches to 3D-orthogonal box-drawings*, in S. Whitesides (Ed.), Proc. 6th Internat. Symp. on Graph Drawing (GD98), Lecture Notes in Computer Science, vol. 1547, Springer, Berlin, pp. 30–43, (1998).

[4] T. Biedl, T.M. Chan, *A Note on 3D orthogonal graph drawing*, Discrete Applied Mathematics, vol. 148, 189–193, (2006).

[5] J. Cai, X. Han, R. E. Tarjan, *An $O(m \log n)$-time algorithm for the maximal subgraph problem*, SIAM J. Comput., vol. 22, pp. 1142–1162, (1993).

[6] N. Chiba, K. Onoguchi, T. Nishizeki, *Drawing planar graphs nicely*, Acta Inform., vol. 22, pp. 187–201, (1985).

[7] T. Biedl, J.R. Johansen, T. Shermer, D.R. Wood, *Orthogonal Drawings with Few Layers*, Lecture Notes in Computer Science, (2002) 2265: 89–93.

[8] T. Biedl, G. Kant, *A better heuristic for orthogonal graph drawings*, in Proc. 2nd annual european sympos. algorithms, vol. 855 LNCS, pp. 24–35, Springer Verlag, (1994).

[9] T. Bield, M. Kaufmann, *Area-efficient static and incremental graph drawings*, in Proc. 5th European Symposium on Algorithms (ESA97), Lecture Notes in Computer Science, vol. 1284, pp. 37–52. Springer-Verlag, (1997).

[10] T. Biedl, B. Madden, I. G. Tollis, *The three-phase method: a unified approach to orthogonal graph drawing*, International Journal of Computational Geometry and Applications, vol. 10, no. 6, pp. 553–580, (2000).

[11] T. Biedl, T. Shermer, S. Whitesides, S. Wismath, *Bounds for orthogonal 3-D graph drawing*, Journal of Graph Algorithms and Applications, vol. 3, no. 4, pp. 63–79, (1999).

[12] T. Biedl, T. Thiele, D.R. Wood, *Three-Dimensional Orthogonal Graph Drawing with Optimal Volume*, Algoritmica, vol. 44, pp. 233–255, (2006).

[13] W. Bing, Y. Shiren, G. Xiujun, S. Zhongzhi, *Tree drawing algorithm and visualizing method*, CAD/Graphics'2001, August 22-24, Kunming, International Academic Publishers, (2001).

[14] J. Bishop, *Java gently*, Addison-Wesley, 2005.

[15] A. Brandstadt, V. B. Le, J.P. Spinrad, *Graph classes: a survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[16] P. Burchard, T. Munzner, *Visualizing the Structure of the World Wide Web in 3D Hyperbolic Space*, in Proceedings of the VRML '95 Symposium, pages 33–38. ACM SIG-GRAPH, (1995).

[17] S. K. Card, J. D. Mackinlay, B. Shneiderman, *Readings in information visualization: using vision to think*, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1999), ISBN:1-55860-533-9.

[18] A. Clauset, M. E. J. Newman, C. Moore, *Finding community structure in very large networks*, Physical Review E, vol. 70, 066111, (2004).

[19] M. Closson, S. Gartshore, J. Johansen, S.K. Wismath, *Fully Dynamic 3-Dimensional Orthogonal Graph Drawing*, Lecture Notes in Computer Science, (1999) 1731: 49–58.

[20] R. F. Cohen, P. Eades, T. Lin, F. Ruskey, *Three-Dimensional Graph Drawing*, Algorithmica, vol. 17, pp. 199–208, (1997).

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduzione agli algoritmi e strutture dati*, seconda edizione, McGraw-Hill, 2005.

[22] I. Cruz, R. Tamassia, *Graph Drawing Tutorial*, IEEE 10th International Symposium on Visual Languages, VL '94, St. Louis, (1994).

[23] E. Dahlhaus, J. Gustedt, R. M. McConnell, *Efficient and Practical Algorithms for Sequential Modular Decomposition*, Journal of Algorithms, vol. 41, pp. 360–387, (2001).

[24] A. Dean, W. Evans, E. Gethner, J. Laison, M. A. Safari, W. Trotter. *Bar k-visibility graphs*, Journal of Graph Algorithms and Applications, vol. 11, no. 1, pp. 45–59, (2007).

[25] H. De Frasseix, J. Pach, R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, vol. 10, no. 1, pp. 41–51, (1990).

[26] O. Devillers, H. Everett, S. Lazard, M. Pentcheva, S. Wismath, *Drawing Kn in Three Dimensions with One Bend per Edge*, Journal of Graph Algorithms and Applications vol. 10, no. 2, pp. 287–295, (2006).

[27] G. Di Battista, P. Eades, R. Tamassia, I. G. Tollis, *Algorithms for drawing graphs: an annotated bibliography*, Comput. Geom. Theory Appl., 4, 235–282, (1994).

[28] G. Di Battista, P. Eades, R. Tamassia, I. G. Tollis, *Graph Drawing Algorithms for the Visualization of Graphs*, Prentice Hall, (1999).

[29] G. Di Battista, G. Liotta, F. Vargiu, *Spirality of orthogonal representations and optimal drawings of series parallel graphs and 3-planar graphs*, in proc. workshop algorithms data struct., vol. 709 LNCS, pp. 151–162, Springer Verlag, (1993).

[30] G Di Battista, R. Tamassia, *Algorithms for plane representations of acyclic digraphs*, Theoret. Comput. Sci., vol. 61, pp. 175–198, (1988).

[31] G. Di Battista, R. Tamassia, *Incremental planarity testing*, in proc. 30th annual IEEE symp. Found. Comp. Sci., pp. 436–441, (1989).

[32] G. Di Battista, R. Tamassia, I. G. Tollis, *Constrained visibility representations of graphs*, inform. proc. lett., vol. 41, pp. 1–7, (1992).

[33] G. Di Battista, L. Vismara, *Angles of planar triangular graphs*, in proc. 25th annual ACM symp. Theory Comput., pp. 431–437, (1993).

[34] W. Didimo, M. Patrignani, M.Pizzonia, *Industrial plant drawer*, in Proceedings of the 9th International Symposium on Graph Drawing (GD'01), LNCS, pp. 475-476. Springer, (2001).

[35] R. Diestel, *Graph Theory*, Springer, 2000.

[36] E. Di Giacomo, W. Didimo, G. Liotta, S. K. Wismath, *Curve-Constrained Drawings of Planar Graphs*, Computational Geometry, vol. 30, no. 2, pp. 1–23, (2005).

[37] H. N. Djidjev, *A linear time algorithm for the maximal planar subgraph problem*, in proc. 4th Workshop Algorithms Data Structure, LNCS, Springer-Verlag, (1995).

[38] P. Eades, *A heuristic for graph drawing*, Congressus Numerantium, vol. 42, pp. 149–160, (1984).

[39] P. Eades, X. Lin, and R. Tamassia, *An algorithm for drawing a hierarchical graph*, International Journal of Computational Geometry and Applications, vol 6, no. 1, pp. 145–156, (1996).

[40] P. Eades, A. Symvonis, S. Whitesides, *Three dimensional orthogonal graph drawing algorithms*, Discrete Applied Mathematics (2000) 103: 55–87.

[41] P. Eades, N. Wormald, *Fixed Edge Lenght Graph Drawing is NP Hard*, Discrete Applied Mathematics, vol. 28, pp. 111–134, (1990).

[42] P. Eades, N. C. Wormald. *Edge crossings in drawings of bipartite graphs*, Algorithmica, vol. 11, no. 4, pp. 379-403, (1994).

[43] J. Ebert, *st-Ordering the Vertices of Biconnected Graphs*, Computing, vol. 30, pp. 19–33, (1983).

[44] M. Eiglsperger, M. Kaufmann, M. A. Siebenhaller, *Topology-shape-metrics approach for the automatic layout of UML class diagrams*, in "Proceedings of the ACM SoftViz 03". NY, USA: ACM Press; pp. 189–98, (2003).

[45] M. Eiglsperger, M. Kaufmann, M. A. Siebenhaller, *Efficient Implementation of Sugiyamas Algorithm for Layered Graph Drawing*, Journal of Graph Algorithms and Applications, vol. 9, no. 3, pp. 305–325, (2005).

[46] M. Eiglsperger, M. Kaufmann, M. A. Siebenhaller, *Efficient Implementation of Sugiyamas Algorithm for Layered Graph Drawing*, Journal of Graph Algorithms and Applications, vol. 9, no. 3, pp. 305–325, (2005).

[47] C. Erten, S. G. Kobourn, V. Le, A. Navabi, *Simultaneous Graph Drawing: Layout Algorithms and Visualization Schemes*, Journal of Graph Algorithms and Applications, vol. 9, no. 1, pp. 165–182, (2005).

[48] C. Erten, S. G. Kobourn, *Simultaneous Embedding of Planar Graphs with Few Bends*, Journal of Graph Algorithms and Applications, vol. 9, no. 3, pp. 347–364, (2005).

[49] H. Everett, S. Lazard, G. Liotta, S. K. Wismath *Universal Sets of n Points for 1-Bend Drawings of Planar Graphs with n Vertices*, Graph Drawing 2007, pp. 345–351, (2007).

[50] S. Fekete, M. Houle, S. Whitesides, *New results on a visibility representation of graphs in 3d*, in F. Brandeburg, editor, Symp. on Graph Drawing, '95, Passau, Germany, Springer Verlag LNCS, pp. 234–241, (1996).

[51] B. Finkel, *Curvilinear Graph Drawing Using The Force-Directed Method*, Thesis, supervisor R. Tamassia, (2003).

[52] L. C. Freeman, *Visualizing social networks*, Journal of Social Structure, vol. 1, no. 1, (2000).

[53] T. M. J. Fruchterman, E. M. Reingold, *Graph Drawing by Force-Directed Placement*, Software: Practice and Experience, vol. 21, no. 11, (1991).

[54] E. R. Gansner, Y. Koren, S. North, *Graph Drawing by Stress Majorization*, in J. Pach, editor, Graph Drawing, in 12th International Symposium on Graph Drawing, Lecture Notes in Computer Science, Heidelberg/Berlin, Springer, vol. 3383, pp. 239–250, (2005). .

[55] M. R. Garey, D. S. Johnson, *Computer and intractability: a guide to the theory of NP-Completeness*, W. H. Freeman, New York, NY, (1979).

[56] M. R. Garey, D. S. Johnson, *Crossing number is NP-Complete*, SIAM J. Algebraic Discrete Methods, vol. 4, no. 3, pp. 312–316, (1983).

[57] A. Garg, *On drawing angle graphs*, in R. Tamassia and I. G. Tollis, editors, Graph Drawing (Proc. GD'94), LNCS, Springer-Verlag, vol. 894 , pp. 84–95, (1995).

[58] A. Garg, R. Tamassia, *A new minimum costo flow algorithm with applications to graph drawing*, in S. North, editor, Graph Drawing (proc. GD '96), LNCS, Springer Verlag, (1997).

[59] S. Gasper, M.-E. Messinger, R. J. Nowakowski, P. Pralat, *Clean the graph before you draw it!*, Information Processing Letters, vol. 109, issue 10, 463–467, (2009).

[60] R. Graham, B. L. Rothshild, J. Spencer, *Ramsey Theory*, John Wiley (1980).

[61] http://www.graphdrawing.org, An information source for researchers, practitioners, developers, and users working on all aspects of graph visualization and representation.

[62] F. Harary, *Graph Theory*, Addison-Wesley, (1969).

[63] C. Homan, A. Pavlo, J. Schull, *A parent-centered radial layout algorithm for interactive graph visualization and animation*, Arxiv preprint cs.HC/0606007, - arxiv.org, (2006).

[64] C. Homan, A. Pavlo, J. Schull, *Smoother transitions between breadht-first-spanning-tree-based drawings*, Graph Drawing, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 4372, pp. 442–445, (2007).

[65] S-H. Hong, N. S. Nikolov, A. Tarassov, *A 2.5D hierarchical drawing of directed graphs*, Journal of Graph Algorithms and Applications, vol. 11, no. 2, pp. 371–396, (2007).

[66] J. Hopcroft, R. Tarjan, *Efficient Planarity Testing*, Journal ACM, vol. 21, no. 4, pp. 549–568, ACM, New York, (1974).

[67] W. Huang, S. H. Hong, P. Eades, *Effects of Sociogram Drawing Conventions and Edge Crossings in Social Network Visualization*, Journal of Graph Algorithms and Applications, vol. 11, no. 2, pp. 397-429, (2007).

[68] G. Kant, *A more compact visibility representation*, in proc. 19th Internat. Workshop Graph-Theoret. Concepts Comput. Sci., (1993).

[69] G. Kant, *Drawing planar graphs using the canonical ordering*, Algorithmica, vol. 16, special issue on Graph Drawing, edited by G. Di Battista and R. Tamassia, pp. 4–32, (1996).

[70] M. Kaufmann, D. Wagner, *Drawing Graphs, methods and models*, Springer, (2001).

[71] T. Kamada, S. Kawai, *An algorithm for drawing general undirected graphs*, Information Processing Letters, vol. 31, no. 1, pp. 7–15, (1989).

[72] D. J. Kleitman, M. M. Krieger, *An optimal bound for two dimensional bin packing*, 16th Annual Symposium on Foundations of Computer Science, pp. 163–168, (1975).

[73] S. G. Kobourov, *Force-directed drawing algorithms*, disponibile su Internet all'indirizzo http://www.cs.brown.edu/ rt/gdhandbook/chapters/force-directed.pdf, sezione chapters.

[74] W. Kocay, *The Hopcroft-Tarjan planarity algorithm*, Computer science department, University of Manitouba, (1993).

[75] D. E. Knuth, *Computer drawn flowcharts*, commun. ACM, vol. 6, (196).

[76] J. A. La Poutrà, *Alpha algorithms for incremental planarity testing*, in proc. 26th annual ACM simp. Theory Comput., pp. 706–715, (1994).

[77] J. Larkin, H. Simon, *Why a diagram is (sometimes) worth ten thousand words*, Cognitive Science, vol. 11, pp. 65–99, (1987).

[78] A. Lempel, S. Even, I. Cederbaum, *An Algorithm for Planarity Testing of Graphs*, Theory of Graphs, Int. Symp. Rome 1966, pp. 215–232, New York, Gordon and Breanch, (1967).

[79] C.-C. Lin, H.-C. Yen, *On balloon drawings of rooted trees*, Journal of graph algorithms and applications, vol. 11, no. 2, pp. 431–452, (2007).

[80] K.-L. Ma, S. T. Teoh, *RINGS: a technique for visualizing large hierarchies*, in GD 2002, vol. 2528 of LNCS, pp. 268–275, Springer, (2002).

[81] J. Manning, M. J. Atallah, *Fast Detection and display of symmetry in outerplanar graphs*, Discrete Applied Mathematics, vol. 39, pp. 13–35, (1992).

[82] T. Matsubayashi, T. Yamada, *A Force-directed Graph Drawing based on the Hierarchical Individual Timestep Method*, World Academy of Science, Engineering and Technology, vol. 27, (2007).

[83] R. M. McConnell, J. P. Spinard, *Modular decomposition and transitive orientation*, Discrete Mathematics, vol. 201, pp. 189–241, (1999).

[84] A. Mendelzon, *Visualizing the World Wide Web*, White Paper, University of Toronto, 1996.

[85] K. Morgan, G. Farr, *Approximation Algorithms for the Maximum Induced Planar and Outerplanar Subgraph Problems*, Journal of Graph Algorithms and Applications, vol. 11, no. 1, pp. 165–193, (2007).

[86] O. Moriš, Advanced Graph Theory I, disponibile su Internet all'indirizzo http://www.fi.muni.cz/ hlineny/Teaching/AGTT/planarity-testing.pdf.

[87] C. Muelder, K.-L. Ma, *Rapid Graph Layout Using Space Filling Curves*, IEEE Trans. Vis. Comput. Graph., vol. 14, no. 6, pp. 1301–1308, (2008).

[88] T. Murata, *Petri nets: Properties, analysis and applications*, Proceedings of the IEEE, vol. 77, pp. 541–580, (1989).

[89] P. Mutton, *Inferring and visualizing social networks on internet relay chat*, Proc. Information Visualization, IEEE Computer Society, London, UK, (2004).

[90] A. Noack, *Energy models for drawing clustered small-world graphs*, Technical Report 07/03, Brandenburg University of Technology at Cottbus, Institute of Computer Science, (2003).

[91] C. Papadopoulos, C. Voglis, *Drawing Graphs Using Modular Decomposition*, Journal of Graph Algorithms and Applications, vol. 11, no. 2, pp. 481–511, (2007).

[92] A. Papakostas, I. G. Tollis, *Improved visibility algorithms and bounds for orthogonal drawings*, in R. Tamassia and I. G. Tollis, editors, Graph Drawing (proc. GD '94), vol. 894 LNCS, pp. 40–51, Springer Verlag, (1995).

[93] A. Papakostas, I. G. Tollis, *Issues in interactive orthogonal graph drawing*, F. Brandeburg, editor, Symp. on grapg drawing, GD '95, vol. 1027, LNCS, pp. 419–430, Springer Verlag, (1996).

[94] A. Papakostas, I. G. Tollis, *A pairing tecnique for area efficient orthogonal drawings*, in S. North, editor, Graph Drawing (proc. GD '96), vol. 1190 LNCS, pp. 354–370, Springer Verlag, (1997).

[95] A. Papakostas, I. G. Tollis, *Algortithms for area efficient orthogonal drawings*, Comput. Geom. Theory appl., vol. 9, no. 1-2, pp. 83–110, (1998).

[96] A. Papakostas, I. G. Tollis, *Efficient Orthogonal Drawings of Hight Degree Graphs*, Algorithmica, vol. 26, pp. 100–125, (2000).

[97] M. Patrignani, *Complexity results for three-dimensional orthogonal graph drawing*, Journal of Duscrete Algorithms, vol. 6, pp. 140–161, (2008).

[98] F. P. Preparata, *Optimal three-dimensional VLSI layouts*, Journal Theory of Computing Systems, Springer New York, vol. 16, no. 1, pp. 1-8, (1983).

[99] P. Rosensthiehl, E. Tarjan, *Rectilinear planar layout and bipolar orientations of planar graphs*, Discrete Computational Geometry, vol. 1, pp. 343–353, (1986).

[100] K. Sugiyama, S. Tagawa, M. Toda, *Methods for visual understanding of hierarchical system structures.*, IEEE Transactions on Systems, Man, and Cybernetics, vol 11, no. 2, pp. 109-125, (1981).

[101] Sun Microsystems Inc., Tutorial Java3D API, disponibile su Internet all'indirizzo http://java.sun.com/products/java-media/3D/collateral/, (ultimo aggiornamento 1999).

[102] R. Tamassia, *On embedding a graph in the grid with the minimum number of bends*, SIAM J. Comput., vol 16, no. 3, pp. 421-444, (1987).

[103] R. Tamassia, I. G. Tollis, *A unified approach to visibility representations of planar graphs*, Discrete Comput. Geom., vol. 1, no. 4, pp. 321–341, (1986).

[104] R. Tamassia, I. G. Tollis, *Planar grid embedding in linear time*, IEEE trans. cicrcuits syst., Cas-36, no. 9, 1230–1234, (1989).

[105] R. J. Trudeau, *Introduction to Graph Theory*, Dover Publications, 1993.

[106] W. T. Tutte, *Convex Representation of graphs*, Proceedings London Mathematical Society, vol. 10, no. 3, pp. 304–320, (1960).

[107] W. T. Tutte, *How to draw a graph*, proceedings London Mathematical Society, vol. 13, no. 3, pp. 743–768, (1963).

[108] F. B. Viàgas, J. Donath, *Social Network Visualization: Can We Go Beyond the Graph?*, Workshop on Social Networks for Design and Analysis: Using Network Information in CSCW, (2004).

[109] C. Walshaw, *A Multilevel Algorithm for Force-Directed Graph Drawing*, GD 2000, LNCS 1984, pp. 171–182, (2001).

[110] M. Yannakakis, *Node-and-edge-deletion NP-Complete problems*, in STOC '78: proceedings of the Tenth Annual ACM Symposium on Theory of Computing, pp. 253–264, ACM Press, New York, (1978).

[111] The premier forum for visualization advances in science and engineering for academia, government, and industry, http://vis.computer.org/VisWeek2009/.

[112] List of VLSI Companies, disponibile su Internet all'indirizzo http://www.vlsi-world.com/index.php.