ALGORITMI PER LA VISUALIZZAZIONE

Prof.ssa Tiziana Calamoneri

VISUALIZZAZIONE DI OGGETTI MOLTO GRANDI (INFINITI)

Tecnica della clusterizzazione

Grafi infiniti

- Problema:
  Molti grafi che scaturiscono da problemi reali hanno dimensioni tali da non poter in alcun modo essere rappresentati su un unico supporto. Nasce, dunque, la necessità di progettare nuove tecniche per poter avere una visione globale dell'intera struttura.

Clusterizzazione (1)

- Consiste nel raggruppare oggetti che hanno tra loro una qualche affinità, tenendoli separati da tutti gli altri.
- N.B. Spesso, i grafi "reali" hanno associata una semantica che induce in qualche modo delle gerarchie nel grafo.

Esempi:

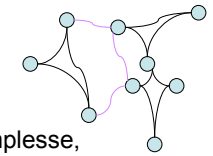
- grafo che rappresenta il traffico di una rete telefonica (nodi= chiamanti, archi= connessioni effettuate). Questo grafo ha associata una ovvia gerarchia indotta dai prefissi e dalla prima parte del numero.
- grafo che rappresenta il traffico nel WWW (struttura simile, basata sui domini e sui sottodomini dell'indirizzo IP).
- classificazione di documenti secondo l'argomento per una ricerca efficiente
- visualizzazione di strutture dati del software object-oriented e parallelo
- mantenimento della traccia della navigazione su Internet o attraverso le informazioni in un database
- …

Clusterizzazione (2)

In tutti questi casi, l'ammontare delle informazioni che si vogliono visualizzare diventano sempre più grandi e le relazioni tra esse sempre più complesse, quindi il classico modello di grafo tende ad essere inadeguato. In molte occasioni, infatti, è necessario visualizzare alcune informazioni strutturali sulle entità, ed è necessario un formalismo più complesso.

Vediamo alcune possibilità…

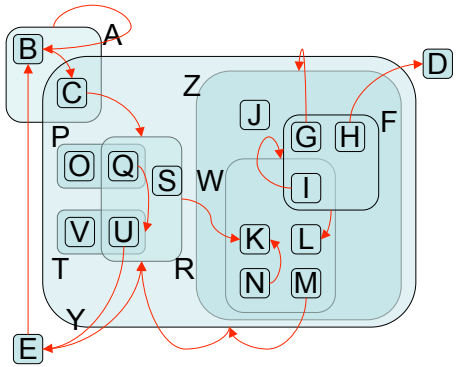
Higraphs (1)



- Introdotto da Harel ('88) partendo dal primo modello di grafo non classico, cioè l'*ipergrafo* (Berge '73).
- Gli higraphs possono rappresentare relazioni complesse, usando dei raggruppamenti, detti *blobs*, multilivello, che possono includersi o intersecarsi arbitrariamente.
- utili per una vasta gamma di applicazioni (ad es. rappresentazione di databases, della conoscenza e di diagrammi degli stati nei sistemi concorrenti).
- PROBLEMA: molto difficile sviluppare metodi di disegno automatico per queste strutture, a causa delle complicazioni che sorgono quando i blobs e i sotto-blobs si intersecano arbitrariamente.

Higraphs (2)

Esempio di Higraph

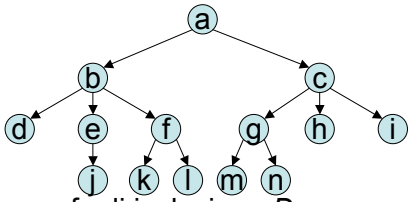


Compound Graphs (1)

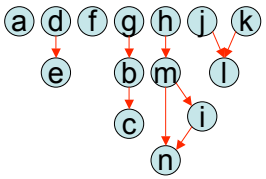
- Sugijama e Misue ('91) presentano allora un nuovo modello, detto *compound graph*, che permette sia l'inclusione che l'adiacenza tra blobs, ma è meno generale del modello higraph.
- **Def.** Un *compound graph (orientato)* è una tripla *D=(V, E, I)* tale che D_a*=(V,E)* è un grafo (orientato) e D_c*=(V, I)* è un grafo orientato. Gli elementi di *E* sono detti *archi di adiacenza*, gli elementi di *I archi di inclusione*. Quindi dire che un arco *(v,w)* appartiene ad *I* equivale a dire che *v* include *w*; questa interpretazione ha senso solo se il grafo diretto D_c è aciclico.

Compound Graphs (2)

Esempio



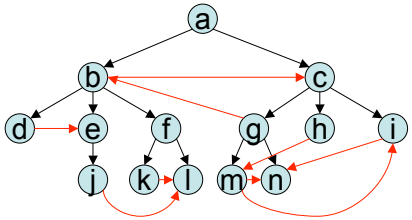
grafo di inclusione D_c

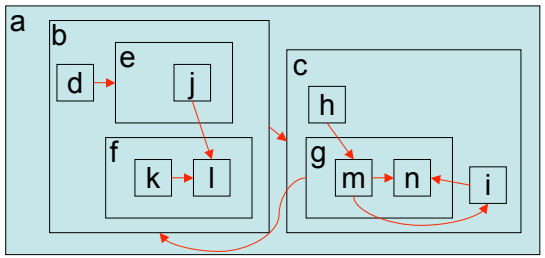


grafo di adiacenza D_a

Compound Graphs (3)

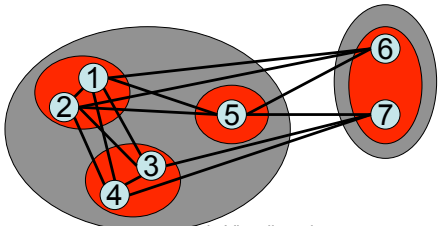
Esempio (segue)





Clustered Graphs (1)

- Feng, Cohen e Eades (‘95), introducono il *clustered graph*. Esso è relativamente generale, poiché riesce ad essere utilizzabile per molte applicazioni; inoltre, sembra essere un modello ben rappresentabile con gli algoritmi di disegno di grafi.
- Un clustered graph consiste in un grafo *G* ed un partizionamento ricorsivo dei nodi di *G;* ogni parte è un *cluster*.

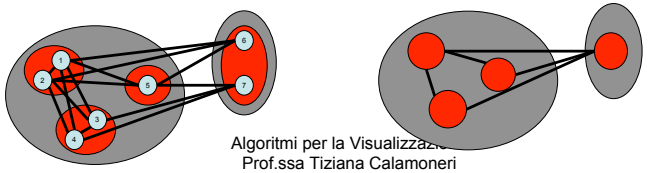


Clustered Graphs (2)

Più formalmente:

Def. Un *clustered graph* è un grafo su cui sia definita una partizione sull’insieme dei nodi; gli insiemi della partizione sono detti *clusters*. Sull’insieme dei clusters può, a sua volta, essere definita una partizione e così via ricorsivamente per un numero arbitrario, ma finito, di volte.
Per una fissata partizione $C_1, C_2, \ldots C_k$ dell’insieme dei nodi di un grafo $G=(V,E)$, il *grafo quoziente* $\mathcal{G}=(\mathcal{V},\mathcal{E})=G/C$ è definito contraendo ogni insieme della partizione in un singolo nodo, cioè:

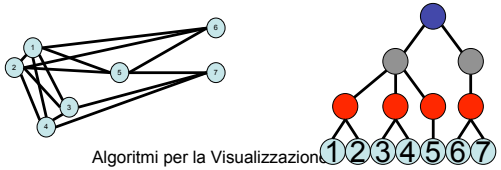
- $\mathcal{V}=\{C_1, C_2, \ldots C_k\}$
- $(C_i, C_j) \in \mathcal{E} \Leftrightarrow i \neq j$ ed $\exists v \in C_i$, $w \in C_j$ t.c. $(v,w) \in E$.



Clustered Graphs (3)

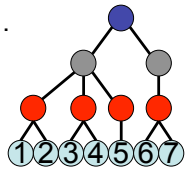
- N.B. il partizionamento ricorsivo dei nodi in clusters e dei clusters in sovra-clusters suggerisce una struttura di inclusione ad albero ⇒ definizione alternativa di clustered graph:

Def. Un *clustered graph C=(G,T)* consiste di un grafo *G* ed un albero radicato *T* tale che le foglie di *T* sono esattamente i nodi di *G*; ogni nodo *v* di *T* rappresenta un cluster dei nodi di *G* che sono le foglie del sottoalbero radicato a *v*. *T* descrive una relazione di inclusione tra i clusters.



Clustered Graphs (4)

- L'*altezza* di un cluster *v*, denotata con *h(v)*, è definita come l'altezza del sottoalbero di *T* radicato in *v*.
- Lo *span* di un arco *(u,v)* di *T* è |*h(u)-h(v)*|. Se lo span di un arco di *T* è maggiore di 1, diremo che quell'arco è *lungo*.
- Possiamo assumere che ogni arco abbia span esattamente 1; questo può essere imposto considerando gli archi lunghi di *T* ed aggiungendo dei nodi fittizi in modo tale che tutti gli archi abbiano span 1.



Clustered Graphs (5)

- Un buon disegno di un clustered graph deve mostrare il disegno del grafo soggiacente come grafo classico, e rappresentare la struttura di clustering sovrastante.
- Tentativo di generalizzare le tecniche di disegno di grafi a clustered graphs.
- N.B. non è sufficiente applicare banalmente algoritmi noti per il disegno di grafi perché nodi che appartengono allo stesso cluster devono essere disegnati vicini, e questo non è sempre facile, soprattutto se si vogliono mantenere visibili alcune proprietà del grafo, come la planarità o le simmetrie.
- SCOPO: preservare la struttura globale del grafo sottostante, clusterizzando ricorsivamente sottografi sempre più grandi e disegnandoli come singoli nodi o regioni.

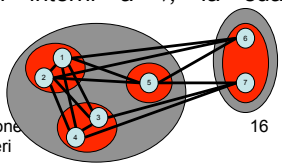
Clustered Graphs (6)

Visualizzazione 2D:

la struttura di clustering è rappresentata dall'inclusione di regioni del piano, cioè, un cluster è rappresentato da una regione che contiene il disegno dei nodi che appartengono al cluster.

Formalmente, un cluster ν è disegnato come una regione chiusa $R(\nu)$ tale che:

- le regioni di tutti i sotto-clusters di ν sono completamente contenuti dentro *R(n)*;
- le regioni degli altri clusters sono completamente fuori da $R(\nu)$;
- se esiste un arco tra due nodi interni a ν, la sua rappresentazione è completamente contenuta in *R(n)*.



Clustered Graphs (7)

Visualizzazione 2D (segue):

Se la struttura clusterizzata cresce, la rappresentazione 2D è insufficiente perché:

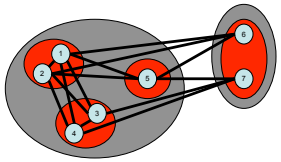
- tempi computazionali alti e complessità intrinseca della struttura dati
- i sistemi di visualizzazione si limitano, al più, ad alcune centinaia di nodi, mentre i grafi che provengono da applicazioni di visualizzazione dell'informazione contengono tipicamente migliaia o anche milioni di nodi

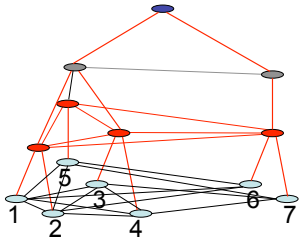
Necessità di usare la terza dimensione…

Clustered Graphs (8)

Visualizzazione 3D:

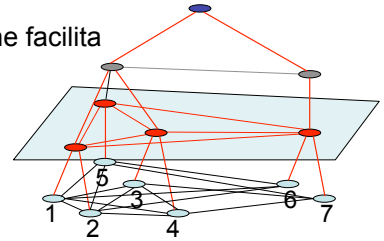
ogni livello di clustering utilizza un piano ad una diversa coordinata *z,* e la struttura di clustering è disegnata come un albero in tre dimensioni





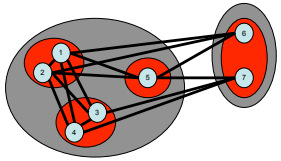
Clustered Graphs (9)

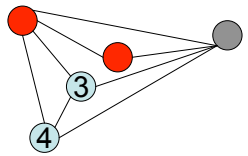
- Per grafi "infiniti" la clusterizzazione è indicata perché permette di vedere un 'riassunto' del grafo, costituito da super-nodi e da super-archi tra super-nodi. Solo a richiesta, alcuni clusters potranno essere mostrati in maggior dettaglio rispetto ad altri.
- Questo tipo di rappresentazione facilita la visualizzazione a differenti livelli di dettaglio, e mantiene anche la mappa mentale dell'utente.



Clustered Graphs (10)

- **Def.** Dato un grafo *G* ed un albero radicato e arbitrario *T*, tale che le foglie di *T* corrispondano ai nodi di *G*, una *vista* del grafo *G* è un sottinsieme *U* dei nodi di *T*, tale che ogni foglia di *T* (e quindi nodo di G) abbia un unico antenato in *U*. La vista induce un grafo quoziente *G/U*, formato contraendo ogni nodo di *G* nel suo nodo rappresentante in *U* e cancellando archi doppi e cappi.





Clustered Graphs (11)

- PROBLEMA quando si usano le VISTE: un super-nodo è collegato ad altri super-nodi allo stesso livello gerarchico; nel momento in cui l'utente espande un super-nodo nella sua rappresentazione più dettagliata, come si fa a determinare come connettere i nodi appena espansi al resto del grafo senza esaminare l'intero grafo sottostante? Infatti, non è chiaro, tramite la semplice struttura di vista, come si possa risalire efficientemente agli archi che collegano diversi livelli della struttura.

Un algoritmo di disegno (1)

- Algoritmo dovuto a Eades e Feng ('96) che visualizza un clustered graph tramite disegno rettilineo 3D.
- L'algoritmo è organizzato in tre passi:
  1. eliminazione degli archi lunghi
  2. costruzione del disegno della vista corrispondente ad ogni livello (bottom-up)
  3. costruzione dell'albero di inclusione (bottom-up).

PASSO 1.

Eliminazione degli archi lunghi

si aggiungono dei vertici fittizi, in modo che gli archi della struttura gerarchica colleghino sempre nodi di livelli adiacenti.

Un algoritmo di disegno (2)

PASSO 2.

Costruzione del disegno livello per livello bottom up

- si usa l'algoritmo di Tutte per ottenere un disegno rettilineo, o un algoritmo di disegno ortogonale
- si considera la struttura di clusters, si prende ciascun cluster e si disegna il grafo che ne è all'interno, posizionando ciascun cluster su un diverso poligono (un rettangolo nel caso ortogonale)
- l'algoritmo viene applicato ad ogni livello della clusterizzazione, tenendo conto che ora i nodi non sono più puntiformi ma dei poligoni.

Un algoritmo di disegno (3)

PASSO 2. (segue)

Costruzione del disegno livello per livello bottom up

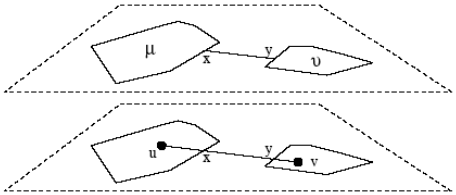
- disegniamo gli archi che collegano clusters allo stesso livello:
  Dati due clusters *c* e *d* sullo stesso livello, sappiamo di doverli congiungere con un arco se esiste almeno un nodo in *c* connesso ad almeno un nodo in *d*. Se queste connessioni sono più di una, possiamo arbitrariamente sceglierne una (ad esempio tra u e v) e procedere al modo seguente:

Un algoritmo di disegno (4)

PASSO 2. (segue)

Costruzione del disegno livello per livello bottom up

- considera l'arco *(u,v)* disegnato sul livello sottostante,
- considera le sue intersezioni *x* e *y* con i poligoni rappresentanti *c* e *d*,
- disegna l'arco *(c,d)* come la parte di arco *(u,v)* compreso tra i punti *x* e *y*.



Un algoritmo di disegno (5)

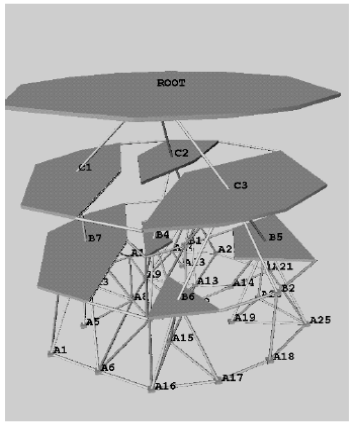
PASSO 3.

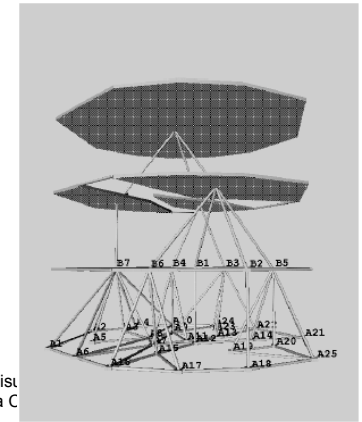
Costruzione dell'albero di inclusione

- dobbiamo disegnare gli archi che congiungono nodi su livelli consecutivi:
  FOR i=1 TO *h* (dove *h* è l'altezza dell'albero delle inclusioni) DO
  IF i=1 per ogni nodo (foglia), posizionalo dove è stato messo dal disegno 2D del grafo
  ELSE per ogni nodo-poligono *c* calcola la media delle coordinate dei suoi figli (a livello *i-1*) ed usa questa media come punto di riferimento per *c*.
- Con questo metodo ogni nodo-poligono viene posizionato sopra la sua rappresentazione più dettagliata
- gli archi dell'albero possono essere rappresentati come segmenti rettilinei.

Un algoritmo di disegno (6)

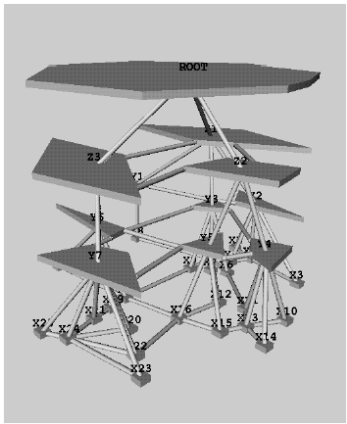
Esempi:

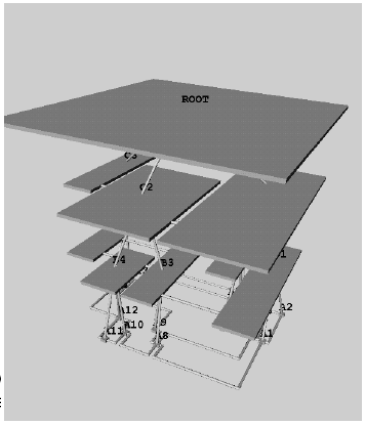




Un algoritmo di disegno (7)

Esempi:





Un algoritmo di disegno (8)

N.B. durante il primo passo, abbiamo rimpiazzato gli archi lunghi con dei segmenti tra livelli adiacenti, quindi non si possono verificare incroci.

Inoltre, quando andiamo ad eliminare i nodi fittizi per rendere nuovamente lunghi gli archi che lo necessitano, l'aver usato la media delle coordinate garantisce che questi archi saranno rettilinei, salvo al più una svolta, in corrispondenza del livello successivo a quello del nodo padre.

Un algoritmo di disegno (9)

La rappresentazione di clustered graphs sembra a tutt'oggi un metodo promettente per visualizzare grandi grafi, sebbene ci siano ancora numerosi problemi da risolvere:

- ci sono pochissimi algoritmi incrementali in letteratura: è tipico che l'insieme degli oggetti costituenti i nodi sia spesso soggetto a modifiche o ad incrementi (si pensi ad una collezione di documenti), ma la strategia comune – sebbene altamente inefficiente – consiste nel ripetere interamente la procedura di clusterizzazione di tanto in tanto; tra un aggiornamento e l'altro vengono fatti solo piccoli aggiustamenti
- non è chiaro come creare i clusters se questi non sono già presenti naturalmente
- non si conoscono strutture dati efficienti per la gestione clusterizzata del disegno

Tecnica della navigazione

Tecnica della navigazione (1)

- Un'alternativa alla clusterizzazione per visualizzare grandi grafi consiste nella *navigazione:* l'utente vede solo un piccolo sottinsieme di nodi e di archi alla volta, avendo a disposizione delle facilities per navigare attraverso il grafo.
- Più in dettaglio, si suppone di avere il disegno del grafo sua una pagina virtuale sufficientemente grande, e si fornisce l'utente di una finestra dotata di scroll bars per permettergli di vedere la parte di interesse e di navigare attraverso il grafo

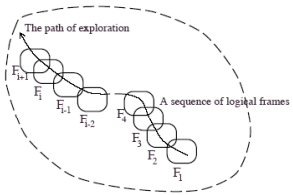


Fig 7. Esplorazione di un grafo.

Tecnica della navigazione (2)

problemi di questo metodo:

- assunzione, spesso poco sensata, che il grafo da visualizzare sia completamente noto e, quindi, memorizzato: in molte applicazioni non è così
- spostamenti attraverso il grafo solo lineari, e non seguendo un percorso logico: l'utente non può saltare ad una parte completamente diversa del grafo
- gli archi lunghi, che non entrano interamente nella finestra, sono difficili da seguire
- perdita della mappa mentale, perché non viene mantenuta alcuna mappa della navigazione dell'utente
- memorizzzione della sua struttura astratta e del disegno dell'intero grafo sulla pagina virtuale molto dispendiosa

Tecnica della navigazione (3)

Relativamente al problema della memorizzazione:

- a disposizione l'intero grafo ma non il suo disegno
- visualizzazione di una finestra definita ad es. in modo da mostrare il sottografo indotto dai nodi che hanno una certa distanza topologica da un dato nodo (considerato come il centro della finestra)
- come costruire i disegni che seguono la navigazione dell'utente in una qualche direzione in modo da mantenere la sua mappa mentale? disegni consecutivi hanno grandi intersezioni che dovrebbero essere rappresentate nello stesso modo
- possibile approccio: aggiungere la parte del disegno verso cui l'utente si sposta alla porzione già costruita ed ancora visibile
- risultati sperimentali pessimi: man mano che ci si allontana dalla porzione iniziale di disegno, la visualizzazione peggiora progressivamente, in termini di numero di svolte, di lunghezza degli archi, di numero di incroci, ecc.

Tecnica della navigazione (4)

Uso integrato delle due tecniche, di clustering e di navigazione, per garantire una soluzione migliore:

- la navigazione dell'utente attraverso il grafo, passando da un cluster all'altro utilizzando la struttura sovrastante di clustering, potrebbe essere un possibile compromesso per evitare di mantenere in memoria l'intero grafo e, allo stesso tempo, per non alterare eccessivamente la mappa mentale dell'utente
- non sono ancora stati progettati algoritmi che funzionino efficientemente sfruttando le due tecniche in modo integrato, sebbene molti tentativi siano stati fatti…

Occhio di pesce

Occhio di pesce (1)

Se il grafo è di dimensioni moderatamente grandi: occhio di pesce (Sarkar e Brown '94):

- il grafo è disegnato per intero ma molto piccolo sullo schermo, e l'utente può vedere in dettaglio un piccolo sottografo avendo a disposizione anche il "contesto" di quel sottografo
- l'area di interesse viene evidenziata, mostrando le altre porzioni dell'immagine con dettaglio sempre minore man mano che ci allontaniamo dal centro

Si risolvono così anche i problemi creati dallo *zoom* che, focalizzandosi su un punto, fa perdere il contesto.

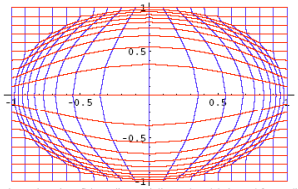


Fig 2. Distorsione fisheye di una griglia regolare del piano. Il fattore di distorsione è 4.

Occhio di pesce

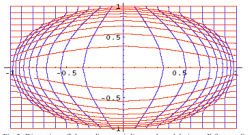


Fig 2. Distorsione fisheye di una griglia regolare del piano. Il fattore di distorsione è 4.

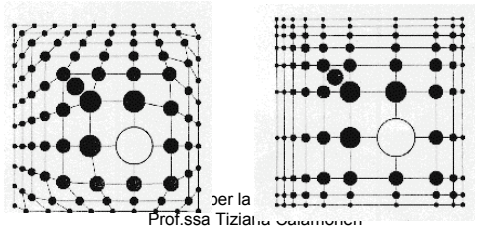
Implementazione:

- il *punto del fuoco* è definito di solito dall'utente;
- la *posizione di un nodo* dipende dalla sua posizione nella vista normale e dalla sua distanza dal fuoco
- la *dimensione di un nodo* dipende dalla distanza dal fuoco e dalla sua dimensione nella vista normale
- la *distanza dal fuoco* di ogni nodo del grafo è distorta da una funzione *h(x)* il cui dominio e codominio sono l'intervallo unitario
- la *distorsione* creata dall'occhio di pesce è la conseguenza della forma della funzione che ha un incremento più veloce intorno a 0 (intorno al fuoco), e un incremento tanto più lento quanto più ci avviciniamo ad 1 (lontano dal fuoco). La definizione esatta della funzione può produrre un minore o maggiore effetto di distorsione

Occhio di pesce (3)

Variazioni allo schema di base:

- La tecnica appena descritta è di solito nota come *distorsione polare*, in quanto si applica radialmente ai nodi in tutte le direzioni che partono dal punto del fuoco.
- Un'alternativa è usare una *distorsione cartesiana*: la distorsione della distanza è applicata indipendentemente su *x* ed *y* prima di stabilire la posizione finale del nodo



Occhio di pesce (4)

ATTENZIONE: L'operazione fondamentale della tecnica ad occhio di pesce è quella di distorcere la posizione di ogni punto.

- Se la distorsione è applicata fedelmente, anche gli archi che connettono i nodi saranno distorti. Matematicamente, il risultato è una curva generica.
- Sistemi grafici standard (per es. X11, Java2D OpenGL) non offrono i necessari strumenti per trasformare le linee in queste curve facilmente.
- SOLUZIONE 1: Di solito si approssimano gli intervalli di linea originali con un numero alto di punti, si trasformano questi punti, e si mostra una spezzata che approssimi l'ideale curva trasformata.
- MA…

Occhio di p

… il numero di punti dell'approssimazione deve essere relativamente alto per ottenere un buona qualità dell'immagine: proibitivo per quantità di calcoli e riduzione delle prestazioni del sistema

- SOLUZIONE 2: applicare la distorsione dell'occhio di pesce sul nodo singolo, e connettere poi i nodi trasformati da un arco diretto
- CONSEGUENZA: introduzione di incroci

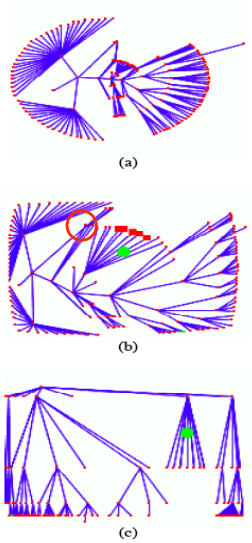


Fig 4. Distorsione Fisheye. Figura (a) rappresenta il grafo senza il fisheye. Figura (b) usa fisheye polare, mentre Figura (c) usa un fisheye cartesiano con un layout differente dello stesso grafo. Il punto verde in figura (b) e (c) denotano i punti focali della distorsione fisheye. Notare l'incrocio extra in figura (b) causato dalla distorsione.

Altre tecniche

Tecniche alternative nel caso in cui il grafo non abbia un numero eccessivo di nodi, ma sia troppo denso:

- è strategia comune ridurre il numero di archi da visualizzare:
  - assegnando dei pesi agli archi secondo certi criteri, e poi imporre una soglia al peso, così che solo gli archi che superino la soglia verranno inclusi nella visualizzazione (Zizi, Beaudouin-Lafon 94). Banale da implementare, estraendo un albero ricoprente da un grafo, riducendo il numero di archi da $O(n^2)$ ad $n-1$.
- ATTENZIONE: si perde una porzione di informazioni.

Altre tecniche