Algoritmi per la visualizzazione

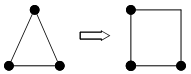
Prof.ssa Tiziana Calamoneri

DISEGNO DI GRAFI:
ALCUNI CASI PARTICOLARI

Disegno 2D ortogonale

Disegno ortogonale 2D (1)

- Punto di vista umano: primo criterio per giudicare la qualità di una visualizzazione: leggibilità
- Punto di vista algoritmico: catturare la qualità tramite funzioni obiettivo:
  - evitare ⇒ massimizzare il min angolo (risoluzione angolare)
  - caso particolare: tutti gli angoli sono multipli di $\pi/2$ (disegno ortogonale) ma diventa inevitabile ricorrere alle svolte ⇒ minimizzare il numero di svolte



Disegno ortogonale 2D (2)

Il problema è complesso… un esempio:

- minimizzare il # di svolte è NP-arduo [Forman et al. '90]
  - punto cruciale della riduzione: trovare l'esatto ordinamento degli archi intorno ai nodi
- se una rappresentazione è fissata: polinomiale

Disegno ortogonale 2D (3)

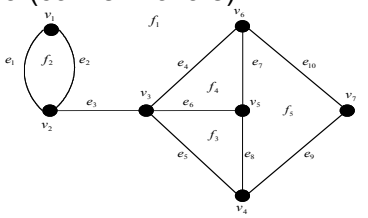
DEF. Un *disegno ortogonale su griglia* di un grafo $G=(V,E)$ è una funzione biunivoca che mappa vertici $v \in V$ in punti a coordinate intere $\Gamma(v)$ ed archi $(v,w) \in E$ in cammini che non si sovrappongono tali che le immagini dei loro estremi $\Gamma(v)$ e $\Gamma(w)$ siano connesse dai cammini corrispondenti. Tali cammini sono costituiti da segmenti orizzontali o verticali con le eventuali svolte a coordinate intere.

Possibile visualizzare solo grafi con grado ≤ 4!

Codifiche di rappresentazioni di grafi (1)

- Codifica di un grafo: elenco dei nodi+elenco degli archi
- Codifica di una rappresentazione piana di un grafo: elenco dei nodi+elenco degli archi+elenco delle facce+verso di percorrenza (convenzionale)

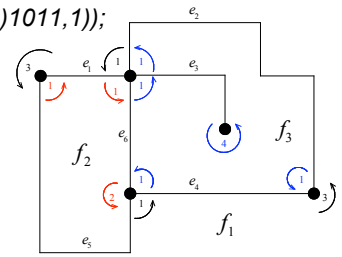
$F = \{f_1, \dots, f_5\}$;
$P(f_1) = (e_1,e_3,e_5,e_9,e_{10},e_4,e_3,e_2)$;
$P(f_2) = (e_2,e_1)$;
$P(f_3) = (e_5,e_6,e_8)$;
$P(f_4) = (e_6,e_4,e_7)$;
$P(f_5) = (e_8,e_7,e_{10},e_9)$;



Codifiche di rappresentazioni di grafi (2)

- Codifica di una rappresentazione ortogonale di un grafo: elenco dei nodi+elenco degli archi+elenco delle facce+info sulla direzione di partenza degli archi+info sulle svolte

$F = \{f_1, \dots, f_3\}$;
$H(f_1) = ((e_1,\varepsilon,3),(e_5,11,1),(e_4,\varepsilon,3),(e_2)1011,1))$;
$H(f_2) = ((e_1,\varepsilon,1),(e_6,\varepsilon,2),(e_5,00,1))$;
$H(f_3) = ((e_2,0010,1),(e_4,\varepsilon,1),(e_6,\varepsilon,1),(e_3,0,4),(e_3,1,1))$;



Codifiche di rappresentazioni di grafi (3)

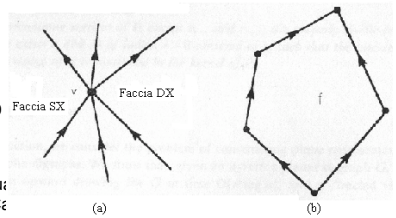
- *regole di consistenza*:
  - deve esistere un grafo 4-planare a cui la rappresentazione si riferisce;
  - ogni arco deve essere memorizzato due volte, una per ciascuna faccia a cui appartiene, e le due memorizzazioni devono essere consistenti;
  - la somma degli angoli lungo il perimetro di una faccia deve essere consistente col fatto che la faccia sia un poligono chiuso;
  - per ogni vertice, la somma degli angoli che insistono su quel vertice deve essere 4.
- N.B. non si parla di lunghezza di archi! si può calcolare la rappr. su griglia di area minima [Tamassia '87]

Un algoritmo basato sulla rappresentazione di visibilità

st-grafi

DEF. Un *st*-grafo *G=(V,E)* è un grafo diretto aciclico che gode delle seguenti proprietà:

- 1.ogni vertice *v∈V* è su un cammino da *s* a *t*;
- 2.per ogni vertice *v∈V,* gli archi entranti appaiono consecutivamente intorno a *v,* e così gli archi uscenti; in questo modo è sempre possibile distinguere in maniera univoca due facce distinte, quella a sinistra di *v* (tra il primo arco entrante e il primo arco uscente) e quella a destra di *v* (tra l'ultimo arco entrante e l'ultimo arco uscente);
- 3.per ogni faccia *f* di *G*, i confini di *f* sono delimitati da due cammini distinti con origine e destinazione comune
- 4.tutti i cammini sono orientati da *s* a *t*, ossia *G* ammette sempre un disegno upward.

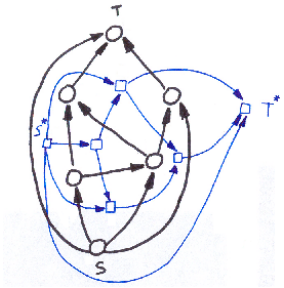


Un algoritmo basato sulla rappresentazione di visibilità (1)

DEF. : Il *duale G** di un grafo orientato *G* è il grafo orientato così definito :

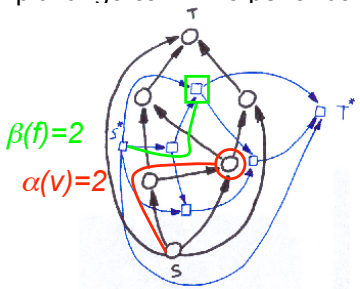
- 1.i vertici di *G** sono le facce di *G*, dove *s** e *t** sono rispettivamente la faccia a destra dell' arco *(s, t)* e la faccia esterna;
- 2.un arco *(f,g)* è in *G** se la faccia *f* condivide con la faccia *g* un arco *(v,w)* ≠ *(s, t)*;
- 3.l'arco *(f,g)* è orientato da *f* verso *g* se *f* si trova a sinistra dell'arco orientato *(v,w)*;
- *4.G** contiene l'arco *(s*,t*)*.

N.B. il duale di un grafo planare è ancora planare, ed è mantenuto l'orientamento (upward→rightward e downward→leftward).



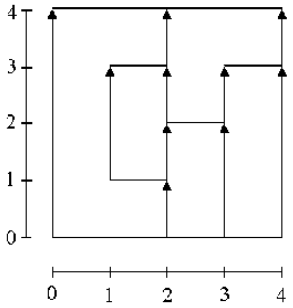
Un algoritmo basato sulla rappresentazione di visibilità (2)

- Dato un qualunque vertice *v* di *G*, il valore $\alpha(v)$ è definito come la lunghezza del più lungo cammino per andare da *s* a *v*.
- Per ogni vertice *f* di *G** il valore $\beta(f)$ è definito come la lunghezza del più lungo cammino per andare da *s** a *f*.



Un algoritmo basato sulla rappresentazione di visibilità (3)

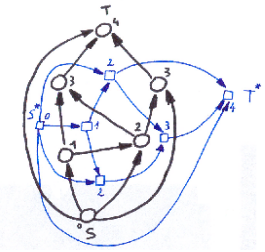
DEF. [Otten & van Wijk '78]: La *rappresentazione di visibilità* $\Gamma(G)$ di un grafo $G=(V,E)$ è una sua rappresentazione che disegna ogni vertice $v \in V$ come un segmento orizzontale $\Gamma(v)$, ed ogni arco $(v,w) \in E$ come un segmento verticale le cui estremità giacciono su $\Gamma(v)$ e $\Gamma(w)$ e non interseca nessun altro segmento vertice $\Gamma(u)$ con $u \neq v,w$.
Inoltre, ne' i segmenti orizzontali ne' i segmenti verticali si sovrappongono



Un algoritmo basato sulla rappresentazione di visibilità (4)

TH. [Rosenthal & Tarjan '86] Sia G un *st*-grafo planare con n vertici. Allora esiste una rappresentazione di visibilità $\Gamma(G)$ per G tale che $\Gamma(G)$ ha esattamente una sorgente $\Gamma(s)$ ed esattamente un pozzo $\Gamma(t)$.

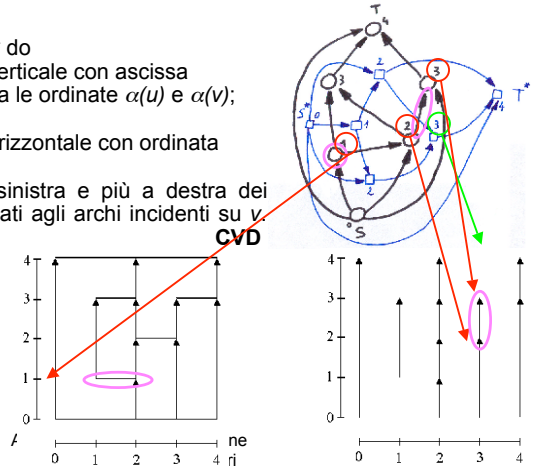
DIM. Costruttiva. Algoritmo che determini un rappr. di visibilità di G:
Input: *st*-grafo planare G
Output: Una rappresentazione di visibilità del grafo G.
1.costruisci il grafo duale G^* di G;
2.calcola, per ogni v di G il valore $\alpha(v)$;
3.calcola, per ogni vertice f di G^* il valore $\beta(f)$;



Un algoritmo basato sulla rappresentazione di visibilità (5)

(segue DIM.)
4.per ogni arco (u,v) di G do
disegna un segmento verticale con ascissa $x(u,v) = \beta(Right(u,v))$ tra le ordinate $\alpha(u)$ e $\alpha(v)$;
5.per ogni vertice v do
disegna un segmento orizzontale con ordinata $y(v) = \alpha(v)$
tra le ascisse più a sinistra e più a destra dei segmenti verticali associati agli archi incidenti su v.

CVD



Un algoritmo basato sulla rappresentazione di visibilità (6)

TH. Sia G un *st*-grafo planare con n vertici ed f facce. L'area della sua rappresentazione di visibilità $\Gamma(G)$ è al più (n-1) x (2n-5).

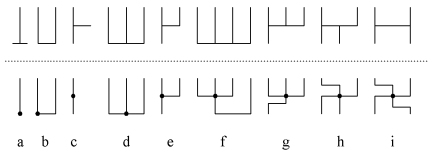
DIM. La larghezza e l'altezza di $\Gamma(G)$ sono limitate dal max valore di $\alpha(v)$ e $\beta(f)$. Il caso peggiore per entrambi è quando il grafo è un cammino $\Rightarrow A \leq (n-1) \times (f-1)$.
Per dim. che $f-1 \leq 2n-5$:
Def. grafo planare massimale: non è possibile aggiungere archi senza perdere la planarità$\Rightarrow$grafo triangolato$\Rightarrow$3f=2m.
Per grafi planari qualunque: $3f \leq 2m$ + formula di Eulero:

$$n-m+f=2$$

$2n-2m+2f=4 \Rightarrow 2n-3f+2f \leq 4 \Rightarrow 2n-4 \leq f \Rightarrow f-1 \leq 2n-5$ **CVD**

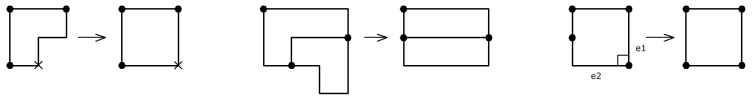
Un algoritmo basato sulla rappresentazione di visibilità (7)

Algoritmo:
INPUT: un st-grafo planare
OUTPUT: un disegno ortogonale su griglia piano
1. creazione rappr. di visibilità (tempo *O(n)*)
2. creazione disegno ortogonale su griglia piano (tempo *O(n)*)



Un algoritmo basato sulla rappresentazione di visibilità (8)

… segue Algoritmo
3. riduzione delle svolte (tempo *O(n)*)



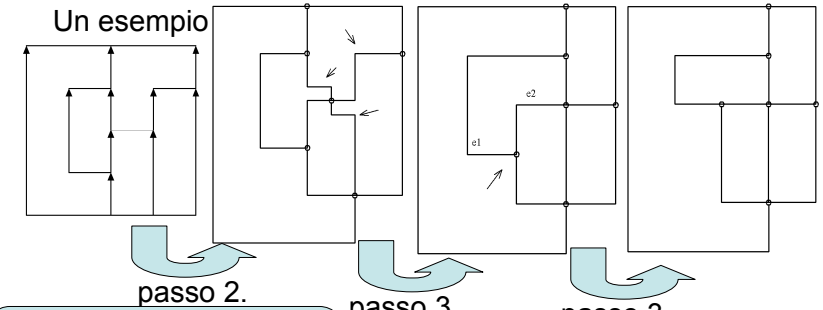
T1
sia concave che convesse

T2
intorno ad un nodo con svolte dello stesso tipo

T3
formano 90° ed uno ha una svolta convessa

Un algoritmo basato sulla rappresentazione di visibilità (9)

Un esempio



passo 2.

passo 3.
trasf. T1

passo 3.
trasf. T3

N.B. il disegno NON è ottimo, ne' in termini di area ne' di num. di svolte!

Un algoritmo basato sulla rappresentazione di visibilità (10)

TH. *Dato un st-grafo G, l'algoritmo precedente ne calcola un disegno ortogonale su griglia piano con $O(n^2)$ area in O(n) tempo.*

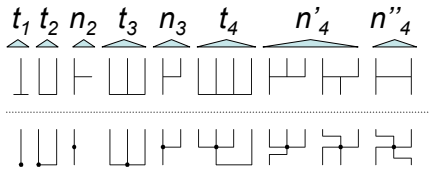
Attenzione: se il grafo non è 2-connesso, lo possiamo partizionare nelle sue componenti 2-connesse, costruire le rappresentazioni, e poi riconnetterle in modo che il grafo finale abbia una sola sorgente e t pozzi.
O(n) tempo

numero di svolte?

Un algoritmo basato sulla rappresentazione di visibilità (11)

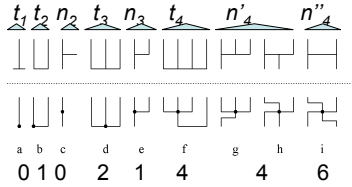
TH. *Se la rappr. di visibilità ha t_i pozzi ed n_i nodi non sorg. di grado i=2,3,4, allora il numero di svolte del disegno è al più $n_3+2n_4+t_2+4t_4+\delta$, dove δ=0,1,2,4, a seconda che la sorgente abbia grado 1, 2, 3, 4.*

DIM.

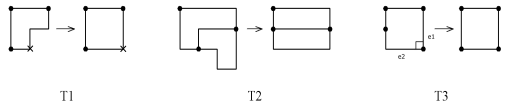


Un algoritmo basato sulla rappresentazione di visibilità (12)

…segue dim.



Dopo il passo 2. il # di svolte è $t_2+2t_3+n_3+4n'_4+6n''_4+4t_4+\delta$



applicata1 volta a g e ad h e 2 volte ad i ➡ elimino almeno $n'_4+2n''_4$ svolte. **CVD**

Un algoritmo basato sulla st-numerazione

st-numerazione (1)

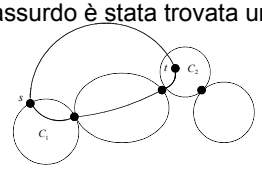
- **Def.** *dato G 2-connesso, una st-numerazione di G è un'etichettatura dei nodi con 1, 2, …, n tale che ogni nodo etichettato j ($2 \le j \le n-1$) abbia almeno un vicino etichettato i<j ed almeno un vicino etichettato con k>j. I nodi etichettati con 1 ed n sono collegati.*
- E' immediato indurre un verso agli archi dalle etichette minori a quelle maggiori.
- Il grafo risultante, con un'unica sorgente (*1*) ed un unico pozzo (*n*) è un st-grafo.

st-numerazione (2)

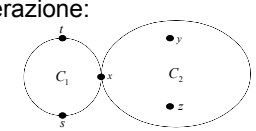
TH. (Even & Tarjan '76) *Sia G un grafo 2-connesso, s e t due suoi nodi adiacenti. Allora, esiste una st-numerazione di G che etichetta s con 1 e t con n, che si può determinare in O(n) tempo.*

Prima di dimostrare il th. osserviamo che se *G* non è 2-connesso, non è possibile trovare una st-numerazione.

Per assurdo è stata trovata una st-numerazione:



1. *s* e *t* sono in comp. distinte



2. *s* e *t* sono nella stessa comp. *y* e *z* hanno etichette max e min in C_2

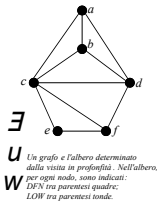
st-numerazione (3)

DIM. Costruttiva: forniamo l'algoritmo.

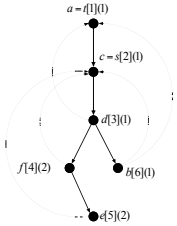
Definiamo le funzioni:

- *DFN(v)*: num. di visita in profondità
- *FATH(v)*: padre di *v*
- *LOW(v): min{DFN(v), DFN(w)* t.c. ∃ *u* un arco all'indietro *(u,w)* con *w* discendente di *v* (o *v* stesso) e *u* antenato di *v*}.
- Definizione ricorsiva:

  LOW(v): min{DFN(v), DFN(x) con *x* figlio di *v, DFN(w)* con *(v, w)* arco all'indietro}



Un grafo e l'albero determinato dalla visita in profondità. Nell'albero, per ogni nodo, sono indicati: DFN tra parentesi quadre; LOW tra parentesi tonde.



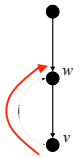
st-numerazione (4)

…segue dim.

PATH(v) inizialmente marca *t,s*, ed *(s,t)*; poi, per ogni *v* restituisce un cammino non marcato da *v* ad un nodo marcato, marcando tutti i nodi e gli archi sul cammino.

4 casi:

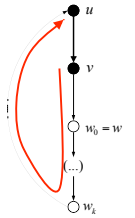
1. ∃ *(v,w)* di riporto non marcato ⇒ marca *(v,w)* e *PATH(v)=vw*



st-numerazione (5)

…segue dim.

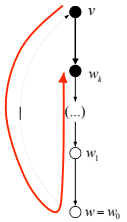
2. ∃ *(v,w)* dell'albero non marcato ⇒ considera tutti i nodi aventi LOW=DFN(u) con u antenato di v. PATH(v)=vw_0…w_ku e marca tutti i nodi e gli archi sul cammino



st-numerazione (6)

…segue dim.

3. $\exists$ (w,v) di riporto non marcato $\Rightarrow$ $PATH(v)=vw_0\ldots w_k$ e marca tutti i nodi e gli archi sul cammino (N.B. qui serve la funzione *FATH*)



4. tutti gli archi incidenti a *v* sono marcati $\Rightarrow$ PATH(v)=$\varnothing$

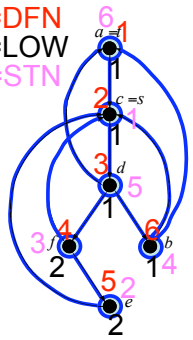
st-numerazione (7)

ALGORITMO ST-NUMBER(*G*)
INPUT: *G* 2-connesso con *s,t* nodi adiacenti di *G*
OUTPUT: una st-numerazione di *G*
esegui una visita in prof. e calcola *DFN, FATH e LOW* di ogni nodo
marca *s, t* ed *(s,t)*
PUSH(PILA, *t);* PUSH(PILA, *s);*
cont=1
v=POP(PILA)
WHILE (*v*$\neq$*t)* DO
  IF PATH(*v*)=$\varnothing$
    THEN STN(*v*)=cont; cont++;
    ELSE (suppondnedo che PATH(*v*)=$vv_1\ldots v_kw$)
        PUSH(PILA, $v_k,\ldots,v_1,v$)
        v=POP(PILA)
END WHILE
STN(*t*)=*n*

st-numerazione (8)

un esempio.

n=DFN
n=LOW
n=STN



PATH(c):
PATH(c)=cda (2)
PATH(c)=cbd (3)
PATH(c)=cefd (3)
PATH(c)=cf (3)
PATH(c)=$\varnothing$
PATH(e)=$\varnothing$
PATH(f)=$\varnothing$
PATH(b)=ba (1)
PATH(b)=$\varnothing$
PATH(d)=$\varnothing$

Ricorda:
1. $\exists$ arco di riporto da v
2. $\exists$ arco dell'albero da v
3. $\exists$ arco di riporto in v

st-numerazione (9)

TH. *L'algoritmo precedente trova una st-numerazione di G 2-connesso in O(n+m) tempo.*

DIM. Ogni nodo riceve un numero:

ciò accade quando esso viene rimosso definitivamente dalla pila, cioè quando tutti i suoi archi incidenti sono marcati. Questo accade sempre perché *G* è 2-connesso $\Rightarrow$ si può sempre arrivare da *s* ad ogi nodo senza passare per *t*.

La numeraz. è proprio st:

STN(s)=1 e STN(t)=n per l'algoritmo. Ogni nodo viene messo in pila per la prima volta come nodo intermedio di un PATH $\Rightarrow$ ha almeno un vicino prima e uno dopo nella pila.

Tempo lineare: ogni nodo ed arco non viene più toccato dopo essere stato marcato. **CVD**

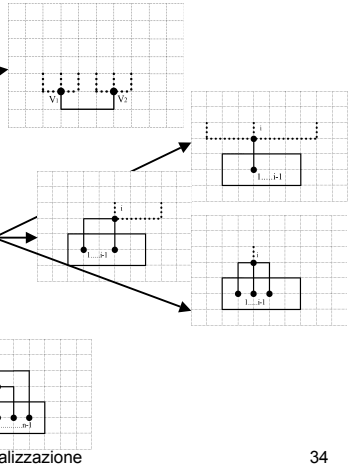
Algoritmo di Biedl-Kant (1)

- L'algoritmo calcola una st-numerazione ed inserisce i nodi secondo questo ordine sulla griglia, aggiungendo gli archi tra il nodo corrente e la parte del disegno già esistente.
- Per fare questo aggiunge una nuova riga per ogni nodo da inserire ed una nuova colonna per ogni arco uscente (gli archi entranti non richiedono spazio aggiuntivo).
- N.B. Il grafo in input deve essere 2-connesso
- Attenzione: funziona anche per grafi non planari. Se il grafo è planare NON garantisce una visualizzazione piana.

Algoritmo di Biedl-Kant (2)

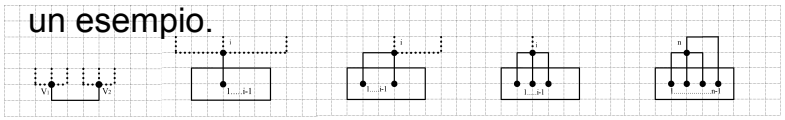
ALGORTIMO
calcola una st-numerazione
metti 1 e 2 così
(N.B. per ogni arco uscente è allocata 1 col.)
FOR 3≤i≤n-1 DO
- metti i su una nuova riga e su una col. già allocata
- disegna gli archi entranti sulle col. già allocate
- alloca 1 col. per ogni arco uscente

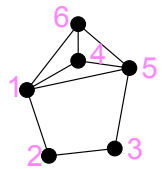
metti n su una nuova riga su una col. già allocata così

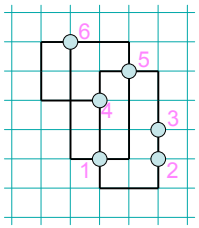


Algoritmo di Biedl-Kant (2)

un esempio.



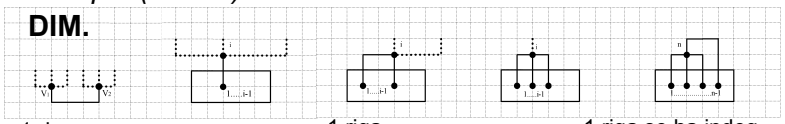




Algoritmo di Biedl-Kant (3)

TH. *L'algoritmo di Biedl-Kant determina un disegno ortogonale su griglia di un grafo 2-connesso con area al più (m-n+1) x n.*

DIM.



- 1 riga
- outdeg(v_1)-1+ outdeg(v_2)-1 col.

- 1 riga
- outdeg(v_i)-1 col.

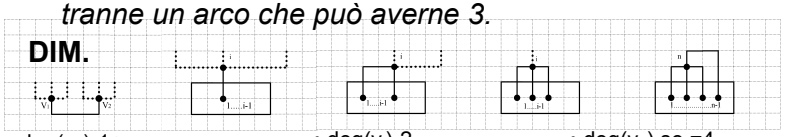
- 1 riga se ha indeg ≤3, 2 righe se ha indeg 4
- outdeg(v_n)-1+1 col.

righe=1+(n-3)+2=n
#col=$\Sigma_{i=1\ldots n}$ (outdeg(v_i)-1)+1=m-n+1

Algoritmo di Biedl-Kant (4)

TH. *L'algoritmo di Biedl-Kant determina un disegno ortogonale su griglia di un grafo 2-connesso con al più 2m-2n+4 svolte, e tutti gli archi hanno al più 2 svolte, tranne un arco che può averne 3.*

DIM.



- deg(v_1)-1+ deg(v_2)-1
- deg(v_i)-2
- deg(v_n) se =4
- deg(v_n)-1 altrimenti

#svolte=deg(v_1)+deg(v_2)-2+$\Sigma_{i=3\ldots n-1}$(deg(v_i)-2)+deg(v_n)= m-2n+4. **CVD**

Algoritmo di Biedl-Kant (5)

TH. *L'arco con 3 svolte si può sempre evitare.*

DIM. Se *G* ha meno di *2n* archi, scegliamo come nodo *t* un nodo con grado minore di 4.

Se il grafo è 4-regolare,lasciamo l'arco senza svolte per ultimo ad ogni passo. **CVD**

Algoritmo di Biedl-Kant (6)

- E' facile convincersi che l'area può essere migliorata, perché ogni nodo viene posizionato su una nuova riga: Algoritmo di Papakostas-Tollis ('97).
- L'algoritmo è simile al precedente, ma cerca di diminuire il # di righe e colonne usate
- IDEA: formare coppie di nodi che possano stare sulla stessa riga o sulla stessa colonna:
  - coppie di riga
  - coppie di colonna

Algoritmo di Biedl-Kant (7)

ALGORITMO PAPAKOSTAS-TOLLIS
calcola una ST-numerazione
calcola un accoppiamento di nodi
FOR *i=1* TO *n* DO
  IF *i* non è stato già inserito
  THEN IF *i* non è accoppiato
    THEN inserisci come Biedl-Kant
    ELSE (*i* e *j, i<j,* sono nella stessa coppia)
      inserisci *i* e *j* nella stessa riga/colonna

AREA: $0.77n^2$ (contro $n^2+o(n^2)$ di Biedl-Kant)
SVOLTE: *2n+4* (=Biedl-Kant)
COMPLESSITA': *O(n)* ma usando una struttura dati sofisticata (Dietz-Sleator '87)