Algotimi per la Visualizzazione

Prof.ssa Tiziana Calamoneri

VISUALIZZAZIONE DI CARTE GEOGRAFICHE

PREMESSA:
Problemi NP-completi

PROBLEMI NP-COMPLETI (1)

- Def. Problema di decisione
- Def. Problema Concreto
- Def. Classe di complessità P
- Def. Verifica in tempo polinomiale
- Def. Classe si complessità NP
- Riducibilità Polinomiale e NP-Completezza

PROBLEMI NP-COMPLETI (2)

- Def. 3-SAT
- Teorema di Cook
- Th. 2-SAT è in P

Il problema dell'etichettatura degli oggetti

Etichettatura di oggetti (1)

- Problema: visualizzare dati (sotto forma di etichette) insieme agli oggetti.
- *automatic map labeling*. Più recentemente: *automatic graph labeling*
- Input: disegno dato in input
- Output: disegno etichettato
- 3 tipi di oggetti da etichettare:
  - punti: città, luoghi particolari, nodi di grafi o diagrammi
  - linee: fiumi, confini, strade, archi di grafi
  - aree: montagne, isole, paesi, laghi, facce di grafi (FACILE ⇒ non lo trattiamo)

Etichettatura di oggetti (2)

Alcuni cartografi, come Imholf e Yoeli (anni '70) hanno tentato di dare regole per misurare la chiarezza semantica di una etichettatura di una carta geografica.

3 concetti comunemente accettati come regole di base per etichettare bene non solo carte geografiche ma anche grafi:

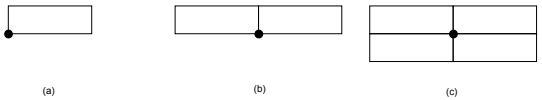
- leggibilità: la misura del carattere deve essere sufficientemente grande;
- non ambiguità: ogni etichetta deve essere univocamente associata all'oggetto cui si riferisce;
- non sovrapposizione: ne' tra etichette ne' con oggetti della mappa o del grafo.

Le possibili posizioni sono dette *etichette candidate*, e ad esse si può associare un *costo*, che ne riflette la qualità rispetto alle regole appena elencate.

Panoramica dei problemi di etichettatura

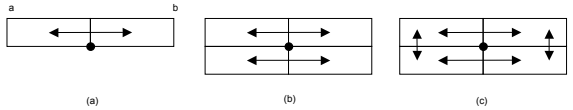
Etichettatura di punti (1)

- Differenti modi per etichettare un punto:
  - modello a posizione fissa: ogni punto ha un insieme fisso di etichette candidate



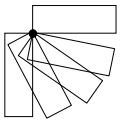
Etichettatura di punti (2)

  - modello a posizione fissa con etichette scalabili: la dimensione può essere scalata
  - modello a etichetta scorrevole: l'etichetta può essere messa in ogni posizione che tocchi l'oggetto (le etichette possono essere *shiftate* in modo continuo



Etichettatura di punti (3)

  - modello a etichetta ruotata: l'etichetta può essere messa in ogni posizione che tocchi l'oggetto con qualunque angolazione



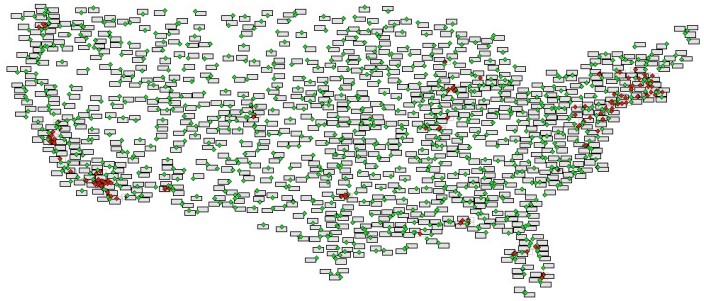
Problemi di etichettatura di punti (1)

- PROBLEMA DI DECISIONE (DP)
  Esiste un assegnamento di etichette di dimensione fissata *s* tale che ogni oggetto sia etichettato con un'etichetta scelta nel suo insieme delle etichette candidate senza alcuna sovrapposizione?
- PROBLEMA DI ETICHETTATURA (LP)
  Se la risposta al problema di decisione è si, trova un assegnamento di etichette di dimensione fissata *s* tale che ogni etichetta sia scelta nel suo insieme delle etichette candidate senza alcuna sovrapposizione

Problemi di etichettatura di punti (2)

- MASSIMIZZAZIONE DEL NUMERO DI ETICHETTE (MAX#P)
  Assegna quante più etichette possibile di dimensione fissata *s* tale che ogni etichetta sia scelta nel suo insieme delle etichette candidate senza alcuna sovrapposizione

Esempio: mod. ad estichette scorrevoli. Pti rossi: pti in cui la sovrapposizione non può essere evitata.



Problemi di etichettatura di punti (3)

- MASSIMIZZAZIONE DELLA DIMENSIONE DELLE ETICHETTE (MAXsizeP)
  Trova un fattore di scalatura *s* massimo e un corrispondente assegnamento di etichette tale che ogni etichetta sia scelta nel suo insieme delle etichette candidate senza alcuna sovrapposizione.

- N.B. differenza tra *map labeling* e *graph labeling*: in quest'ultimo posso decidere di muovere leggermente gli oggetti per fare entrare meglio le etichette, nelle carte geografiche questo non si può fare!

Problemi di etichettatura di punti (4)

- Relazioni tra i vari problemi:
  - LP ≥ DP: ovvio
  - LP ≥ MAXsizeP
    supponiamo di avere un algoritmo *A* che risolva il problema LP, allora con una ricerca binaria posso trovare la max dimensione.
  - MAXsizeP ≥ LP e MAXsizeP ≥ DP
    analogamente al caso precedente, un algoritmo che risolve MAXsizeP risolve anche LP semplice e ovviamente DP.
  - MAX#P ≥ LP: ovvio
- Da queste relazioni si deduce, in particolare, che MAXsizeP ed LP sono nella stessa classe di complessità (informalmente, sono ugualmente difficili), visto che LP ≥ MAXsizeP ≥ LP.

Etichettatura di linee (1)

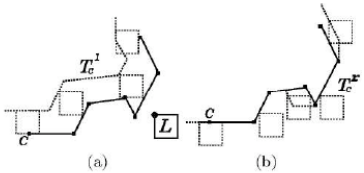
- carte geografiche: difficile etichettare le linee perché non sono linee rette.
- problema più semplice: linee rette orizzontali o verticali, che funziona bene per disegni di grafi ortogonali.

- MASSIMIZZAZIONE DEL NUMERO DI ETICHETTE CON LINEE ORIZZONTALI (LHLP)
  considerare un insieme di segmenti orizzontali (o verticali), ed etichettarli con un rettangolo di altezza fissata e lungo quanto tutto il segmento, se questo è possibile. Per ciascuna etichetta sono possibili 3 posizioni:

Etichettatura di linee (2)

- caso generale: una curva è rappresentata da una serie di segmenti connessi fra loro, chiamiamo questo insieme di segmenti *catena*.
- nel caso di etichettatura di curve possiamo alternativamente considerare di nuovo i rettangoli o, per aumentare la precisione, i quadrati che racchiudono ogni singola lettera dell'etichetta e l'etichetta è un insieme di quadrati.
- *spazio di etichettatura:* lo spazio in cui il quadrato si può muovere lungo la catena senza sovrapporsi ad altri punti.
- *punto di riferimento* del quadrato il suo vertice in alto a sinistra

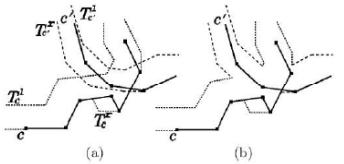
spazio di etichettatura sopra la linea



spazio di etichettatura sotto la linea

Etichettatura di linee (3)

- assegniamo una direzione alla catena, per esempio da sinistra in basso a destra in alto, e chiamiamo T_c^l (T_c^r) la spezzata di tutte le possibili posizioni del punto di riferimento a sinistra (destra) della catena.
- Se due catene si sovrappongono e si intersecano anche i loro spazi di etichettatura, e le intersezioni fra i due spazi di etichettatura vengono cancellati. Può accadere che, pur se due catene non si incrociano, i loro spazi di etichettatura si intersechino; anche in questo caso rimuoviamo le intersezioni.



Problemi di etichettatura di linee

- Il problema di etichettatura delle linee consiste nel posizionare tutte le lettere (quadrati) all'interno dello spazio di etichettatura (non ce ne occuperemo)
- Un altro problema di etichettatura consiste nell'etichettare linee curve nel caso in cui l'etichetta sia un singolo rettangolo.

Algoritmi di etichettatura

Etichettatura di punti (1)

DP

- La versione decisionale del problema dell'etichettatura di punti nel modello a 4 posizioni è NP-completa, anche se le etichette sono quadrate e tutte della stessa misura. Questo risultato è stato ottenuto indipendentemente da Kato e Imai ('88) e da Marks e Shieber ('91).

  Riduzione da 3-SAT: la risposta a DP è 'SI' sse la formula associata è soddisfacibile.

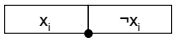
LP

- Formann e Wagner ('91) hanno studiato il problema dell'etichettatura di punti nel modello a 4 posizioni, fornendo degli algoritmi approssimanti.

Etichettatura di punti (2)

LP (segue)

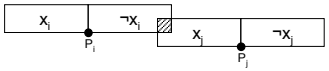
- Formann e Wagner hanno pure mostrato che il modello a 2 posizioni si risolve in tempo polinomiale ($O(n)$), riducendolo a 2-SAT nel modo seguente:
  - i punti diventano le variabili della formula; se l'etichetta è a sx del punto i allora ho la variabile x_i, altrimenti ho la sua negata.



Etichettatura di punti (3)

LP (segue)

- Siano P_i e P_j due punti, se è possibile una sovrapposizione tra l'etichetta destra di P_i (x_i) e l'etichetta sinistra di P_j (x_j) allora questa può essere evitata se $\neg x_i$ e x_j non sono entrambe vere. Quindi si costruisce la clausola $\neg((\neg x_i) \wedge x_j)=(x_i \vee (\neg x_j))$.
- Si può costruire una tale clausola per ogni coppia di etichette candidate incompatibili.



$$\neg((\neg x_i) \wedge x_j)=(x_i \vee (\neg x_j)))$$

E' ovvio che ad un assegnamento di verità per la formula corrisponde un assegnamento fattibile di etichette e, viceversa, se un assegnamento di verità non esiste, allora i punti non possono essere etichettati completamente.

Etichettatura di punti (4)

MAXsizeP

- MAXsizeP nel modello a 4 posizioni è NP-arduo, e si può dimostrare che non esiste un'approssimazione migliore di 2, a meno che P=NP. Wagner ('94) ha dimostrato che esistono algoritmi di approssimazione con rapporto di approssimazione 2 che può essere eseguito in tempo $\Omega(n \log n)$, nel caso particolare di etichette quadrate.
- Idea dell'algoritmo:
  - costruisci le 4 etichette candidate con dim. minima per ogni punto;
  - ingrandisci via via la dimensione e cancella una delle due etichette che eventualmente si possono intersecare (scegli a caso quale delle due)
  - prosegui in questo modo fino a quando ogni punto ha solo 2 etichette candidate
  - applica l'algoritmo precedente (riduzione a 2-SAT).

Etichettatura di punti (5)

MAXsizeP (segue)

- Questo risultato è interessante dal punto di vista teorico, perché dimostra l'approssimabilità del problema, ma è inapplicabile in pratica, perché il rapporto di approssimazione 2 per la dimensione delle etichette è moltissimo!!! Inoltre, questo algoritmo funziona solo per etichette quadrate.

- L'algoritmo ora descritto non tiene conto dell'esistenza di archi. Questi possono solo peggiorare la situazione: in presenza di archi, si dimostra che il rapporto di approssimazione rimane 2, ma il tempo computazionale peggiora.

Etichettatura di punti (6)

MAX#P

- Agarwal, vanKreveld e Suri ('98) hanno formulato il problema in termini di *massimo insieme indipendente*
  - dato *G=(V,E)*, un *insieme indipendente I* è un sottoinsieme di *V* tale che comunque si scelgano 2 nodi in *I*, questi non devono essere adiacenti. Un insieme indipendente banale è quello costituito da un nodo singolo; lo scopo del problema è quello di massimizzare la cardinalità di *I*
- Idea dell'algoritmo (per ogni punto vorrei mettere un'etichetta)
  - costruisci un grafo con tanti nodi quante sono le possibili etichette candidate;
  - metti un arco tra 2 nodi se le etichette corrispondenti hanno intersezione non vuota (N.B. le etichette candidate di uno stesso punto formano una clicca).
  - stiamo così riducendo il problema a quello del massimo insieme indipendente, che è NP-arduo.
- l'algoritmo ora descritto è *log n*-approssimante. Nel caso in cui le etichette abbiano altezza unitaria, esiste una PTAS, che è (1+1/k)-approssimante e può essere eseguita in tempo $O(n \log n + n^{2k-1})$.

Etichettatura di punti (7)

MAX#P con etichette scorrevoli

- Iturriaga e Lubiw ('97) hanno dimostrato che il problema è NP-arduo già con un solo grado di libertà (cioè quando l'etichetta può scorrere, ad esempio, solo orizzontalmente).
- Van Kreveld, Strijk e Wolff ('98) hanno confrontato i 3 modelli scorrevoli (ad 1, 2 e 4 gradi di libertà) con quelli a posizione fissa, tentando di rispondere alla domanda: "quanti più punti si possono etichettare usando un modello anziché un altro?" Per questa quantificazione, hanno definito il *rapporto tra due modelli*:

  Sia *P* un insieme di punti del piano. Siano *M1* ed *M2* due modelli per etichettare *P*, e siano *opt(M1,P)* e *opt(M2,P)* il massimo numero di punti di *P* che ricevono un'etichetta nei due modelli. Il *rapporto tra i modelli M1 ed M2* è definito come il max su tutti i P di fissata cardinalità n del sup per n che tende all'infinito del rapporto tra opt(M1,P) e opt(M2,P).

Etichettatura di punti (8)

MAX#P con etichette scorrevoli (segue)

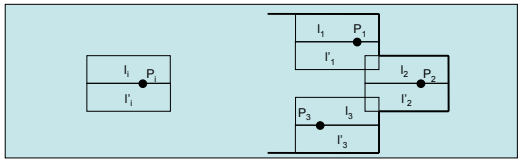
- Gli autori hanno mostrato limitazioni inferiori e superiori per i rapporti di molte coppie di modelli. Ad esempio, il max # di pti etichettati con il mod. a 2 pos. è al più il doppio di quello ad una pos.
- Questi valori permettono di progettare algoritmi approssimanti per quasi tutti i modelli.

Etichettatura di punti (8)

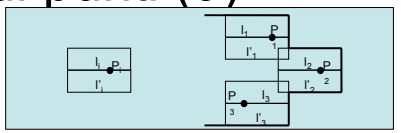
MAX#P con etichette scorrevoli (segue)

- IPOTESI AGGIUNTIVA: tutte le etichette hanno uguale altezza

  Algoritmo 2-approssimante per il mod. a 4 pos.(compl. *O(n log n)*).
  - Dato un insieme *P* di *n* punti, all'*i*-esimo passo si supponga di avere già etichettato *i-1* punti (quelli più a sinistra) e si etichetti l'*i*-esimo punto.
  - L'algoritmo segue la filosofia greedy, e sceglie l'*etichetta più a sinistra*, cioè l'etichetta candidata più a sx tra tutte quelle possibili dei punti non etichettati, sfruttando la def. di *involucro destro delle etichette di un insieme di punti P*, cioè una funzione f definita come f(y)=max{$-\infty$, max{*x:(x,y)* in l_i o l'_i}} dove:
    - l_i è l'etichetta già assegnata al punto P_i;
    - l'_i è la copia di l_i posta più in basso



Etichettatura di punti (9)



MAX#P con etichette scorrevoli (segue)

- si utilizza la def. di involucro destro perché, in questo modo, dell'intera etichetta, si può considerare solo il suo spigolo in basso a sx, osservando che essa può essere aggiunta all'insieme di quelle già scelte se tale spigolo cade alla sx dell'involucro destro.
- E' possibile mantenere bassa la complessità dell'algoritmo mantenendo delle opportune strutture dati (3 heap).
- N.B. Questo algoritmo funziona con rapporto di approssimazione 2 solo per rettangoli di altezza costante e lunghezza variabile; esso si può generalizzare ad etichette più generiche, ma non si può più garantire il rapporto di approssimazione.

Etichettatura di punti (10)

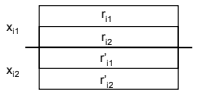
MAXsizeP con etichette ruotate

- Doddi, Marathe, Mirzaian, Moret e Zhu ('99) hanno studiato il problema con etichette quadrate e tutte uguali, collegate al punto in una posizione qualsiasi del contorno e con qualsiasi direzione, tentando di massimizzare la dimensione.
- danno un algoritmo con approssimazione $8\sqrt{2}/sen(\pi/10)$ in tempo *O(n log n)*.
- N.B. Questo algoritmo ha puro interesse teorico, dimostrando che il problema è approssimabile entro una costante, ma non può essere in alcun modo approssimato, essendo tale costante circa 2000!!!

Etichettatura di linee (1)

LHLP

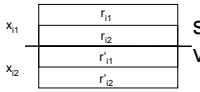
- il problema si risolve polinomialmente, riducendolo a 2-SAT:



- per ogni linea *i* si introducono 2 variabili x_{i1} e x_{i2} dove:
  - $x_{i1}=1 \Leftrightarrow r_{i1}$ è usato $\Leftrightarrow r'_{i1}$ non è usato;
  - $x_{i1}=0$ altrimenti;
  - $x_{i2}=1 \Leftrightarrow r_{i2}$ è usato $\Leftrightarrow r'_{i2}$ non è usato;
  - $x_{i2}=0$ altrimenti.

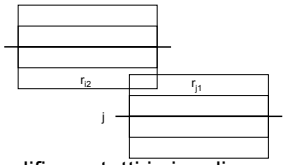
Etichettatura di linee (2)

LHLP (segue)



se si utilizza r_{i1} si sta utilizzando anche r_{i2}, quindi vale $x_{i1} \Rightarrow x_{i2}$ che si può scrivere come $(\neg(x_{i1}) \vee x_i)$.

siano dati ad esempio x_{j1} e x_{i2}, allora le variabili vengono composte nel seguente modo: $\neg(x_{j1} \wedge \neg(x_{i2}))=(\neg(x_{j1}) \vee x_{i2})$.



Usando clausole come queste possiamo codificare tutti i vincoli.
N.B. Una sol. di 2-SAT implica una sol. al problema e viceversa.
OSS. se togliamo il vincolo che i segmenti siano paralleli agli assi possiamo adattare l'algoritmo ma la complessità peggiora.

Etichettatura di linee (3)

etichettatura di linee curve

L'algoritmo dovuto a Edmonson e Christensen ('96):
Divide il problema in tre sottoproblemi:

1. *Scegliere le etichette candidate*. Data una linea, identificare le possibili posizioni per un'etichetta.
2. *Valutare le etichette*. Data una posizione, calcolare (velocemente) il punteggio associato ad una posizione.
3. *Scegliere le etichette*. Scegliere le posizioni delle etichette tale che massimizzino la qualità dell'etichettatura.

Etichettatura di linee (4)

etichettatura di linee curve (segue)

1. scegliere le posizioni candidate (ora infinite!);
   3 proprietà che deve avere un algoritmo che genera pos. candidate:
   a) scegliere un numero limitato di candidati;
   b) trovare un buon compromesso tra qualità ed efficienza (scegliere solo posizioni di alta qualità risulterebbe di difficile e lenta risoluzione);
   c) distribuire uniformemente le posizioni candidate per un'etichetta intorno all'oggetto da etichettare.

Etichettatura di linee (5)

etichettatura di linee curve (segue)

1. scegliere le posizioni candidate (segue);

- Tra le infinite etichette ammissibili per una linea l'algoritmo ne sceglie una certa quantità, in modo che le etichette candidate siano più o meno equidistanti tra di loro, e tale distanza è prefissata;
- N.B. la quantità di etichette scelte in questo modo dipende dalla lunghezza della linea e dell'etichetta.
- Se tale quantità è comunque troppo alta, si calcola una funzione di costo semplificata (che tiene conto solo delle sovrapposizioni tra etichette candidate), e si eliminano le etichette candidate in eccesso tra quelle con costo più alto.

Etichettatura di linee (6)

etichettatura di linee curve (segue)

2. valutare le etichette
   L'algoritmo assegna un costo a ciascuna etichetta candidata; a questo costo contribuiscono vari punti:
   a. sovrapposizioni:
      un'etichetta può sovrapporsi con pti, linee o altre etichette; il costo dovrà essere tanto maggiore quanti più pti l'etichetta interseca, quanto più è lungo il segmento della linea sulla quale si sovrappone, e quante più sono le etichette con le quali si sovrappone;
   b. canoni di valutazioni per il posizionamento di etichette di punti:
      nel caso di etichette di pti si è osservato che è preferibile che l'etichetta sia: i. a dx piuttosto che a sx; ii. sopra piuttosto che sotto; iii. sulla stessa linea del pto che si etichetta.
      Valutate 19 posizioni possibili per l'etichettatura di un punto, e a ciascuna è stato assegnato una valore che contribuisce al suo costo;

Etichettatura di linee (7)

etichettatura di linee curve (segue)

2. valutare le etichette (segue)
   c. canoni di valutazioni per il posizionamento di etichette di linee:
      cruciale la distanza media dell'etichetta dalla linea; inoltre, poiché la linea non è rettilinea, potrebbe avere delle parti tangenti all'etichetta, o che addirittura le si sovrappongono, per questo viene introdotto un parametro che tiene conto della dist. minima; inoltre, una particolare preferenza viene assegnata alle posizioni in cui la linea da etichettare è piuttosto piana; infine, è da preferirsi un'etichetta sopra piuttosto che sotto la linea, ed il più centrata possibile.
3. scegliere le etichette
   tramite la tecnica del simulated annealing

Etichettatura di carte geografiche (1)

- Prevede l'etichettatura di tutti gli oggetti presenti
- Semplificazione: etichette di dim. variabile, ma rettangolari con i lati paralleli agli assi.
- Definiamo il *costo* di un'etichetta candidata come un valore non negativo che riflette il suo "punteggio" in termini di non ambiguità e di numero di sovrapposizioni con altri oggetti ed etichette. La funzione obiettivo da ottimizzare è la somma dei costi delle etichette scelte.
- Algoritmo approssimante di Kakoulis e Tollis ('98): funziona particolarmente bene per grafi ortogonali.

Etichettatura di carte geografiche (2)

L'algoritmo si compone di 3 passi:

1. per ogni oggetto, si generano tutte le etichette candidate e si assegna loro un costo;
2. si costruisce un grafo associato $G_R=(V_l,E_l)$ dove ogni nodo $l \in V_l$ rappresenta l'etichetta candidata; dati $l_1,l_2 \in V_l$, se le etichette corrispondenti si sovrappongono allora si costruisce l'arco $(l_1, l_2) \in E_l$.
   Obiettivo: trovare un sottografo costituto da sole cricche. In ciascuna cricca, solo un'etichetta può essere scelta. Questo ha senso perché tutte le etichette candidate relative allo stesso punto formano una cricca. L'ideale sarebbe trovare le cricche di dim. massima (NP-arduo).
   Euristicamente, si cancellano nodi in modo tale che rimangano solo cricche. Sia G'_R il grafo ottenuto da tali cancellazioni.

Etichettatura di carte geografiche (3)

3. si costruisce un grafo bipartito, $G_m=(V_f \cup V_c, E_m)$ dove ogni oggetto (nodo o linea) è un nodo di V_f, e ogni cricca è un nodo di V_c;
   - si inserisce un arco *(f,c)* solo se esiste un'etichetta candidata di *f* che appartiene a una cricca in V_c;
   - si cerca un accoppiamento massimale di G_m perché ogni nodo in V_c sia adiacente ad una sola cricca, altrimenti 2 oggetti diversi avrebbero etichette che si intersecano, quindi, tra tutti gli accoppiamenti massimi scegliamo quello di costo minimo;
   - questo può essere fatto in tempo $O(n^{2,5})$.

   N.B. se il grafo G'_R è quello di partenza, questo accoppiamento trova la soluzione ottima, ma se G'_R si ottiene da G_R rimuovendo casualmente dei nodi, l'algoritmo è approssimante.

Il problema dell'intersezione dei segmenti

L'intersezione di segmenti (1)

- Osservare una carta geografica può essere difficile, se vi sono rappresentate troppe informazioni. Per questo i sistemi di informazione geografica (GIS) la separano in ciò che vengono chiamati *layers*.
- Ogni layer è una carta tematica e memorizza solo un tipo di informazione (un layer memorizza le strade, uno memorizza le città, uno laghi e fiumi…).
- L'utente di un GIS può selezionare un layer per studiarlo, ma può anche fondere le informazioni contenute in due o più layers per avere un quadro d'insieme. Ad esempio, mettendo insieme la carta stradale e quella in cui sono rappresentate le città, si può avere un'idea di quale percorso fare per raggiungere la propria meta.
- Di particolare rilievo il problema di intercettare le intersezioni tra gli oggetti visualizzati nei due layers.

L'intersezione di segmenti (2)

- Forma più semplice del problema:
  - le due carte da sovrapporre sono di fatto delle reti (si pensi ad esempio alla carta delle strade, dei fiumi, delle ferrovie, e così via) rappresentate come collezioni di segmenti (dato che le curve possono essere approssimate con delle spezzate).
  - lo scopo è quello di trovare le intersezioni tra le due reti.
  - N.B. i segmenti vanno considerati chiusi

L'intersezione di segmenti (3)

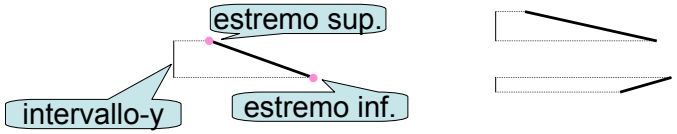
- Formalizzazione del problema in un contesto geometrico:
  Dati due insiemi di segmenti chiusi, calcolare tutte le intersezioni tra un segmento di un insieme ed un segmento dell'altro insieme.
- Per semplificare il problema, è possibile mettere insieme tutti i segmenti in un unico gruppo e cercare le intersezioni tra una qualsiasi coppia di segmenti. A posteriori, se si determina una intersezione tra due segmenti dello stesso gruppo (cosa che accadrà senz'altro perché segmenti consecutivi intersecano ai loro estremi), tale punto dovrà essere eliminato dalla soluzione.

L'intersezione di segmenti (4)

- Il problema non sembra complicato: è sufficiente scorrere tutte le possibili coppie di segmenti e calcolare se ciascuna coppia porta ad un'intersezione oppure no.
- Se i segmenti sono n, la complessità di questo approccio è $\Theta(n^2)$.
- Nel caso peggiore, in cui cioè ogni coppia di segmenti ha un'intersezione, questo algoritmo è ottimo.
- In pratica, le intersezioni sono un certo numero k, di solito molto più piccolo di n^2; in tal caso la complessità è troppo elevata.
- Vogliamo ottenere un algoritmo la cui complessità dipenda non solo dal numero n di segmenti, ma anche dal numero k di intersezioni trovate. Un tale algoritmo prende il nome di *output-sensitive*, visto che la sua complessità dipende da quello che sarà l'output.

L'intersezione di segmenti (5)

- Sfruttiamo la situazione geometrica:
- se due segmenti sono l'uno in prossimità dell'altro, essi sono dei candidati per una possibile intersezione, ma se sono lontani non si potranno intersecare.
- **Definizione**: Dato un segmento s non parallelo all'asse delle ascisse, il suo *estremo superiore (inferiore)* è l'estremo con ordinata maggiore (minore). L'*intervallo-y* di s è la sua proiezione sull'asse y. Se due segmenti hanno i loro intervalli-y disgiunti, allora essi sono detti *lontani*, e non potranno intersecarsi.



L'intersezione di segmenti (6)

- Per testare l'intersezione tra due segmenti, basta considerare quelli che hanno i rispettivi intervalli-y che si intersecano, cioè per cui esiste una linea orizzontale che li intercetti entrambi.
- Per individuare tali coppie introduciamo nel piano una linea orizzontale l che spazza il piano dall'alto verso il basso.
- Lo *stato di l all'istante t* è l'insieme dei segmenti che essa interseca all'istante t, perciò lo stato varia man mano che, col passare del tempo, l si sposta verso il basso, ma lo fa in modo discreto, variando solo in certi punti, detti *punti evento*.
- Tra i punti evento vi sono gli estremi dei segmenti di S.

L'intersezione di segmenti (7)

- punto evento = un estremo inferiore

  l sta abbandonando il segmento corrispondente e questo deve essere eliminato dallo stato di *l*
- punto evento= un estremo superiore

  l sta intersecando un nuovo segmento, la cui intersezione deve essere testata con tutti i segmenti nello stato di *l.*

L'intersezione di segmenti (8)

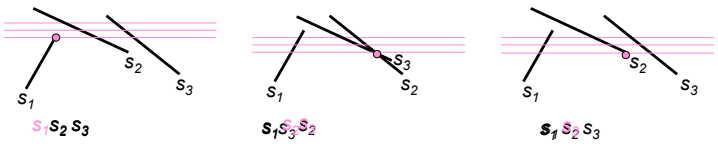
- Sfortunatamente questo approccio non è sufficiente a garantire che l'algoritmo sia output sensitive
- Ci sono dei casi in cui si testa comunque l'intersezione di $O(n^2)$ coppie, anche se le intersezioni trovate saranno molte meno
- Esempio: se *S* è costituto da *n* segmenti verticali tutti posizionati alla stessa altezza: *l* li intercetta contemporaneamente tutti e quindi verifica le intersezioni tra tutte le coppie, ma non ne trova.
- La ragione è che viene verificata l'intersezione tra segmenti che, di fatto, sono situati lontani l'uno dall'altro.

L'intersezione di segmenti (9)

- Ordiniamo i segmenti da sinistra a destra appena essi entrano nello stato di *l*.
- In questo modo vogliamo inglobare l'idea di vicinanza anche rispetto all'asse *x,* oltre che *y*.
- Ora i punti evento non sono più solo gli estremi dei segmenti, ma anche i punti di intersezione che vengono scoperti durante l'esecuzione dell'algoritmo, e che provocano una modifica nell'ordinamento dei segmenti.
- IDEA: Verrà testata l'intersezione di due segmenti nello stesso stato di *l* solo se essi sono adiacenti nell'ordinamento.
- Appena un segmento cambia la sua situazione nello stato di *l* lo confrontiamo solo con al più altri due segmenti, i suoi adiacenti nell'ordinamento.

L'intersezione di segmenti (10)

- come cambia la situazione nello stato di *l*?
- un punto evento vi entra (punto evento=estremo superiore)
- un punto evento viene spostato nell'ordinamento (punto evento=intersezione)
- un vicino di un punto evento viene eliminato dallo stato (punto evento=estremo inferiore)



L'intersezione di segmenti (11)

Teorema. *Se due segmenti s_i ed s_j intersecano, allora esiste uno stato di l in cui s_i ed s_j sono adiacenti nell'ordinamento.*

Dim. 1. intersezione tra s_i ed s_j sull'estremo di uno dei due segmenti: esso verrà facilmente trovato quando l incontra tale estremo e cambia il suo stato.

2. intersezione tra due punti interni: supponiamo che l si trovi leggermente sopra il punto di intersezione; se l è abbastanza vicino a tale punto, cioè se non ci sono punti evento tra la posizione di l e il punto di intersezione, s_i ed s_j saranno adiacenti nell'ordinamento.

N.B. poiché la linea l parte sopra tutti i segmenti, all'inizio il suo stato sarà vuoto; dovrà esistere un momento in cui s_i ed s_j vengono inseriti nello stato ed uno in cui diverranno vicini nell'ordinamento. Allora la loro intersezione verrà testata ed individuata.

- Abbiamo dimostrato la correttezza del metodo.

L'intersezione di segmenti (12)

Strutture dati usate:

- punti evento memorizzati in una struttura dati Q detta *coda degli eventi.*

  Dati due eventi p e q, definiamo la relazione d'ordine $\lhd$ come segue: $p \lhd q$ se e solo se o $y(p)<y(q)$ o $y(p)=y(q)$ e $x(p)<x(q)$.

  Q è un albero di ricerca bilanciato, ordinato secondo la relazione d'ordine $\lhd$. Insieme con ogni punto evento p, in Q memorizziamo anche i segmenti che ammettono p come estremo superiore; in questo modo si hanno le informazioni necessarie per gestire l'evento.

L'intersezione di segmenti (13)

Strutture dati usate (segue):

- Operazioni da poter fare sulla *coda degli eventi Q*:
  - *rimuovere* il punto evento successivo da Q e restituirlo in modo che possa essere trattato. Tale punto evento è il più alto al di sotto della linea l.
  - *inserire* un nuovo evento (intersezione).
  - N.B. due distinti punti evento possono coincidere ma è conveniente trattare tali punti come un unico punto evento, per cui l'operazione di inserimento dovrà essere in grado di verificare se un evento sia già presente in Q. Si è scelto l'albero binario di ricerca e non l'heap per gestire efficientemente questo caso.
  - Entrambe le operazioni prendono tempo $O(\log m)$, dove m è il numero di eventi al momento inseriti in Q, ed m non può mai superare $2n+k$, dove k è il numero di intersezioni.

L'intersezione di segmenti (14)

Strutture dati usate (segue):

- stato di l memorizzato in un albero binario di ricerca bilanciato T in cui i segmenti nello stato risulteranno ordinati visitando l'albero in in-ordine (da sinistra verso destra).

  Operazioni da poter fare su *T: inserimento*, *cancellazione,* ricerca del precedente e del successivo. Ogni modifica dell'albero prende $O(\log m')$ tempo, dove m' è il numero di segmenti presenti nello stato di l in un certo istante, ed m' non può mai superare n.

L'intersezione di segmenti (15)

Overview dell'algoritmo

1. riempi Q con gli estremi dei segmenti — *O(n log n)*
2. inizializza T vuoto — *O(1)*
3. (l inizia a spazzare il piano)
4. finché ci sono pti evento in Q: — *2n+k volte:*
   - estrai l'evento da Q — *O(log (n+k))*
   - modifica T — *O(log n)*
   - se l'evento cambia l'ordinamento in T, verifica intersezioni; se c'è una nuova intersezione inseriscila in Q — *O(1)+ O(log (n+k))*

Totale: *O(n log n)+(2n+k)O(log (n+k))*

Se *k≤n, O(n log n+k log n)*

Se *k>n, O(n log n + k log k)* cioè l'alg. è output sensitive

L'intersezione di segmenti (16)

Lemma. *L'algoritmo ora descritto determina correttamente tutte le intersezioni e tutti i segmenti che le contengono.*

Dim. N.B. la priorità di un evento è data dalla sua ordinata. Dimostriamo il lemma per induzione sulla priorità dei punti evento.

HP induttiva: Sia p un pto d'intersezione, tutti i pti d'intersezione q con priorità maggiore sono già stati determinati correttamente.

TH: anche p ed i segmenti che lo contengono verranno determinati correttamente.

Insieme dei segmenti diviso in:

U(p)=ammettono p come estremo sup.

L(p)=ammettono p come estremo inf.

C(p)=ammettono p come punto interno.

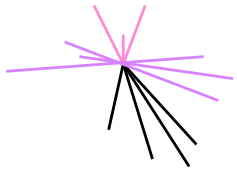
L'intersezione di segmenti (17)

Dim. (segue)

- p sia l'estremo di qualche segmento.

  Allora p è memorizzato in Q all'inizio dell'algoritmo, insieme ai segmenti di *U(p) e di L(p).*

  I segmenti di *C(p)* vengono memorizzati in T prima del momento in cui p viene gestito (quando vengono incontrati da l i loro estremi sup.), così anch'essi saranno individuati. In questo caso, quindi, il lemma è dimostrato.



L'intersezione di segmenti (18)

Dim. (segue)

- p non è l'estremo di alcun segmento, quindi esiste solo *C(p)*. Dobbiamo solo dim. che p sarà inserito in Q in qualche momento come pto di intersezione.

  Si considerino tutti i segmenti coinvolti e si immaginino ordinati per angolo che essi formano intorno a p.

  Siano s_i ed s_j due di tali segmenti e siano essi adiacenti nell'ordinamento appena dato. Allora, c'è un punto evento q con una priorità maggiore di p tale che s_i ed s_j diventano adiacenti quando l oltrepassa q. Per induzione, q è stato gestito correttamente, quindi anche p sarà determinato e memorizzato in Q. **CVD**