CONSIGLIO
NAZIONALE
DELLE RICERCHE

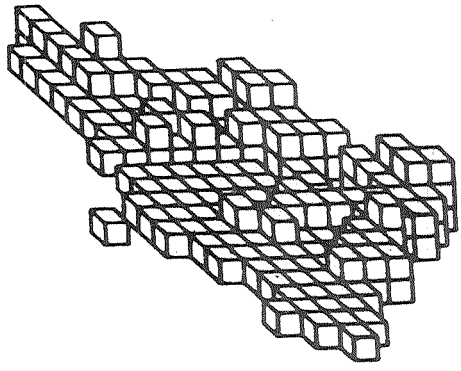
PROGETTO
FINALIZZATO
INFORMATICA

INFORMATIZZAZIONE
DELLA PUBBLICA
AMMINISTRAZIONE

TERRITORIO

Una metodologia automatica per la generazione di carte tematiche di sintesi

C. Montani
M. Morandi Cecchi
F. Pazzini
P. Castini



82 - P2 - TERRI - 4 - ISIPI - 001

PROGETTO FINALIZZATO INFORMATICA

Sottoprogetto P2: Informatizzazione della Pubblica Amministrazione

OBIETTIVO TERRITORIO

UNA METODOLOGIA AUTOMATICA PER LA GENERAZIONE DI CARTE TEMATICHE DI SINTESI

+ C. Montani, ++ M. Morandi Cecchi, F. Pazzini, P. Castini

INTRODUZIONE

Scopo di questo lavoro è lo studio e la sperimentazione di una metodologia automatica per la generazione di carte tematiche di sintesi a partire da carte di base e/o derivate.
Questo lavoro si inquadra nell'implementazione di sistemi di gestione ed elaborazione di informazioni geografiche, che raggiungono un buon compromesso tra occupazione di memoria, dettaglio dell'informazione cioé grado di risoluzione dell'informazione ottenuta, e tempi di elaborazione.
Questa implementazione si pone come linea logica a metà strada fra le organizzazioni cellulari o matriciali e le organizzazioni "linked" o poligonali, essa si rifà a quella detta tightly closed boundary (t.c.b.) descritta da Merrill.
Per arrivare alla configurazione (t.c.b.) si acquisisce il dato in forma poligonale, poi sia la sorgente che i risultati parziali sono mantenuti in codifica di Freeman a sedici direzioni, il data-base è in codifica di Freeman, dove ogni proprietà costituisce un diverso livello dell'archivio e la codifica delle informazioni non è più per coppie di coordinate (x, y) ma è più consono al successivo utilizzo delle carte stesse.
Si usano infatti due diverse codifiche di Freeman: una di soli links ed una invece nella quale ogni elemento ha due elementi, un link ed un contatore.

+ Istituto di Elaborazione dell'Informazione del C.N.R.- PISA
++ Istituto di Matematica, Università degli Studi di Catania.

L'elemento di input é la poligonale, l'elemento in archiviazione è la codifica di Freeman e l'organizzazione (t.c.b.) è solo utile in fase di elaborazione.
Allo stato attuale dell'implementazione il sistema prevede l'archiviazione di tre tipi di carte di base, ossia : quelle direttamente digitalizzate (come: pedologia, geologia, ecc.), carte tematiche altrimenti elaborate (come: carte delle pendenze e delle esposizioni), carte tematiche di sintesi come unità dello stesso sistema (come:carte dell'uso del suolo, ecc.).
Sopra questo archivio agisce una grammatica per l'elaborazione logica delle carte archiviate e tale elaborazione è condotta organizzando le carte in t.c.b..
Attualmente il sistema prevede come possibili uscite, diversi tipi di periferica: su plotter, su video, e su stampante elettrostatica.
Il lavoro può essere pensato logicamente suddiviso in tre fasi in cascata: fase (1), consiste essenzialmente nell'acquisizione dei dati di ingresso al sistema.
L'acquisizione consiste nella digitalizzazione, per ogni tipo di carta, degli archi di curva semplici (da nodo a nodo) rappresentanti i contorni delle singole proprietà descritte in quella carta.
Alla fine di questa fase possiamo ancora immaginare ogni carta come un insieme di proprietà aggregate.
Fase (2), in questa fase viene effettuata la generazione dell'archivio di carte suddivise per proprietà.
Fase (3), ha il compito di elaborare le richieste dell'utente e di produrre le carte in sintesi. Sono possibili le tre operazioni logiche e viene impiegata, per le elaborazioni, la codifica delle informazioni secondo lo schema proposto da Merrill.

Questo lavoro è stato eseguito con fondi del Progetto Finalizzato Informatica Sottoprogetto P2 Obiettivo Territorio.

FASE 2: LA CODIFICA DI FREEMAN

Il metodo proposto da Freeman consiste nel sovrapporre alla curva in esame un immaginario grigliato regolare di passo opportuno, e nel convertire la curva stessa in una sequenza di links scelti da un prefissato insieme.
Essendo questi links di lunghezze e direzioni note é sufficiente, per la codifica di una curva, fornire la posizione nel piano del suo punto iniziale e la stringa dei links successivi.
Fra i possibili insiemi di direzioni abbiamo scelto quello che ne ha sedici (fig. 2) poiché consente una buona gamma di spostamenti, evitando da un lato una segmentazione troppo vistosa del segno (come si ha, a parità di passo, con le catene ad otto direzioni), e consentendo dall'altro l'intera occupazione del semi-byte (16 direzioni sono codificabili con 4 bits) in cui viene memorizzata la codifica dello spostamento senza alcun spreco di memoria e con la possibilità di un facile accesso all'informazione.
In fig. 2 si può confrontare la resa della medesima curva guantizzata nei sistemi ad 8-direzioni, 16-direzioni e a 32-direzioni.
Il numero di bits richiesti per rappresentare una curva nei vari sistemi di codifica varia sensibilmente.
Se si usa una parola codice distinta per ogni link, avremo bisogno rispettivamente di 3, 4, 5 bits per link.
I risultati sono mostrati nella tab. 1. (Freeman (4)).

Tab. 1

Sistema	num. di links	Bits per links	Tot.num.bits
8	87	3	261
16	44	4	176
32	38	5	190

In fig. 4 sono evidenziati i nodi del grigliato selezio-

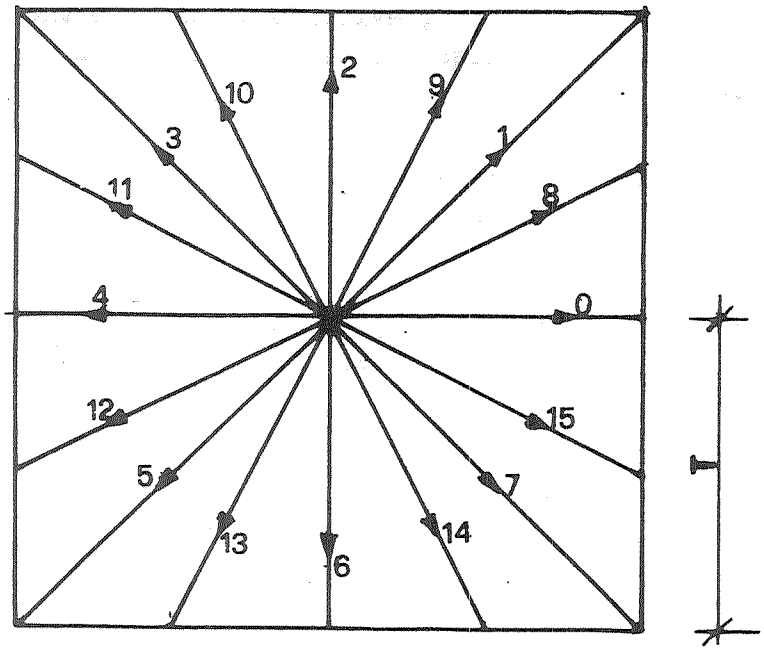


FIG. 1

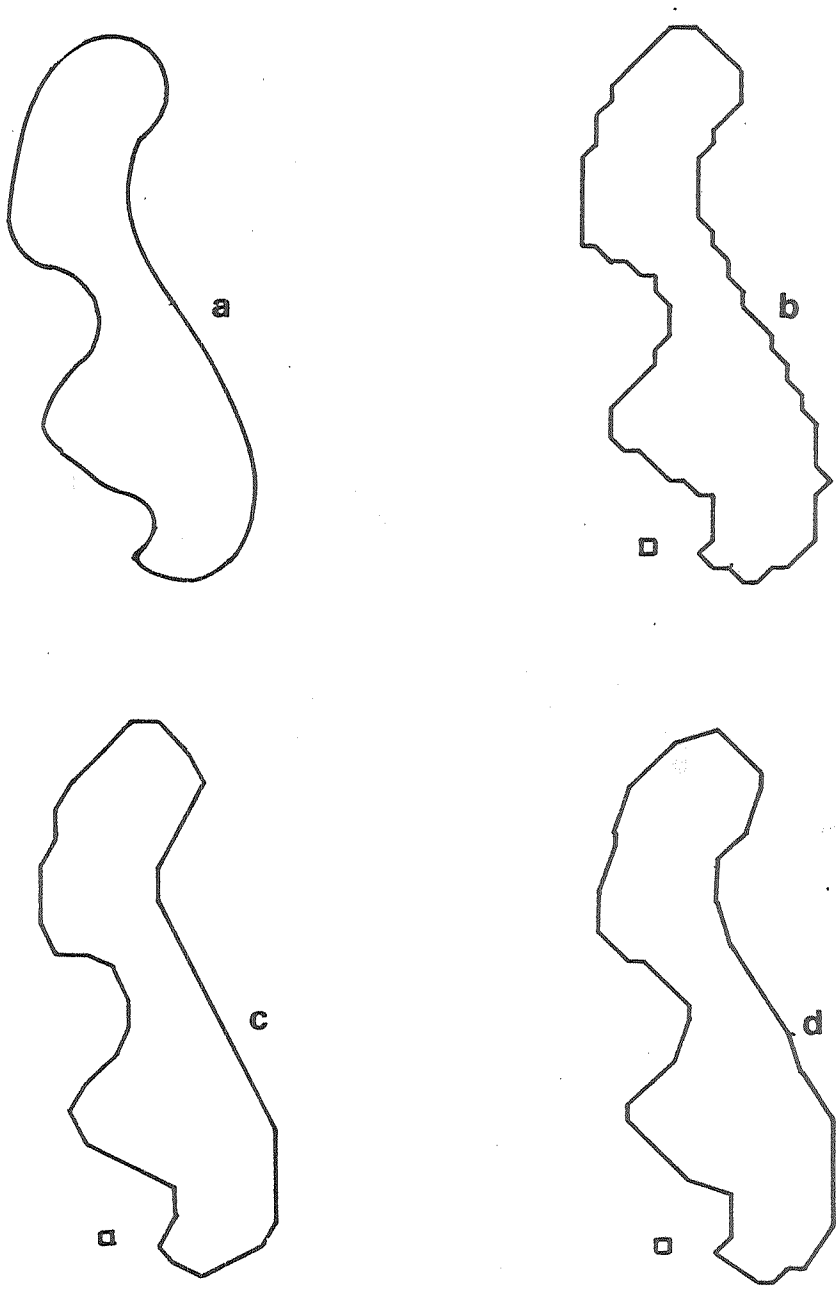


FIG. 2

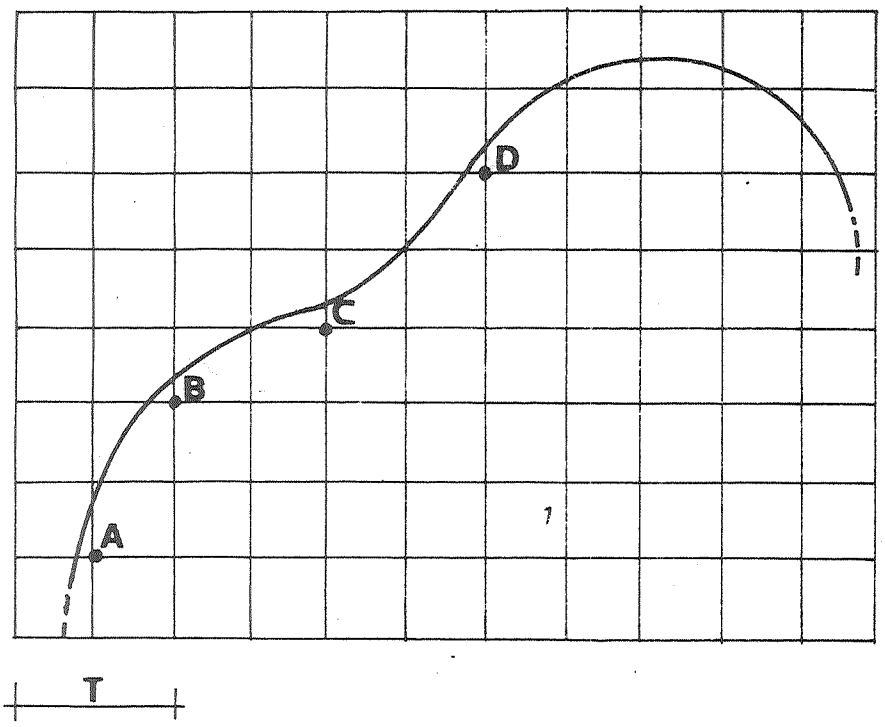


FIG. 3

nati usando lo schema di codifica da noi scelto. Il tracciato è descritto dalla sequenza di links

9, 8, 1, ...

Individuata così la catena che approssima la curva in esame, siamo in grado di conoscerne la lunghezza poiché i links hanno tre misure standard:

T per la direzioni 0,2,4,6
T 5/2 per le direzioni 8,9,10,11,12,13,14,15
T 2 per la direzioni 1,3,5,7

come é facile vedere dalla fig. 4.

La codifica di ogni curva è completata con la memorizzazione del punto di inizio della catena (in coordinate assolute), e con il numero di links presenti.

Col passaggio da archivi mantenuti a coordinate ad archivi organizzati a catene, si ottiene un buon risparmio di mezzi di memorizzazione, che è dipendente dal passo scelto per il grigliato di trasformazione e dal numero di direzioni scelte per la trasformazione a catena.

Per ogni tipo di carta vengono generati tanti files quante sono le proprietà della carta stessa. Ogni file porta il nome della proprietà opportunamente codificato, e l'accesso è assicurato tramite macro assembler.

Concludiamo questo paragrafo con alcune considerazioni sull'uso delle catene a 16 direzioni.

Il fatto di aver adottato questo schema di codifica ha come conseguenza negativa che non è assicurata la "chiusura" di un contorno chiuso.

Può infatti accadere che la catena termini ad una distanza dal punto iniziale, su cui dovrebbe chiudere, tale che la differenza fra le coordinate di inizio e di fine catena soddisfa una delle tre relazioni:

1) $|X_i - X_f| = T/2 \qquad |Y_i - Y_f| = T/2$

2) $|X_i - X_f| = 0 \qquad |Y_i - Y_f| = T/2$

3) $|X_i - X_f| = T/2 \qquad |Y_i - Y_f| = 0$

Questo fatto causa la mancata chiusura fisica del contorno, ma non influisce minimamente sul metodo in quanto abbiamo la certezza che le aree da noi trattate sono aree chiuse e quindi possiamo elaborarle logicamente come tali.

FASE 3.1: CODIFICA DI MERRILL

In questo schema una regione a contorno chiuso viene scandita lungo un insieme di linee parallele a distanza costante e le intersezioni di queste linee col contorno sono usate per rappresentare la regione, Merrill (5).
Come avviene in molte rappresentazioni di dati geografici per computer, il contorno della regione si assume riferito ad un grigliato quadrato e le linee di scansione parallele sono prese come le linee orizzantali del grigliato.
Queste linee del grigliato vengono scandite dal basso verso l'alto, considerando solo quelle che attraversano la regione.
Per ogni scansione vengono memorizzate le coordinate dei nodi del grigliato che appartengono al contorno dopodiché l'insieme di tutte le scansioni forma la rappresentazione codificata del contorno stesso.
Lo schema utilizza la proprietà per cui una linea che attraversa una regione delimitata da un contorno chiuso, lo interseca un numero pari di volte.
Per rendere la rappresentazione (fig. 4) non ambigua, devono essere soddisfatte 3 condizioni:

1) il contorno deve essere continuo e chiuso, nel senso che la distanza assoluta tra successivi punti del cammino non deve essere più grande della diagonale della maglia del grigliato. Se nella scansione si trova che questa restrizione non vale, la coppia di coordinate "mancanti" sono linearmente interpolate ed inserite nel cammino. Questo viene fatto tra i punti A e B dell'esempio.

2) bisogna prevedere i casi in cui la linea di scansione interseca tangenzialmente il contorno.

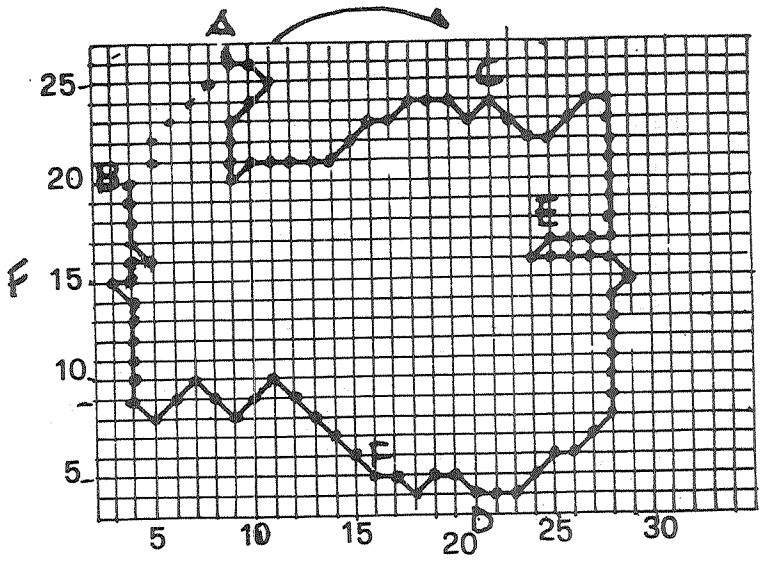


FIG. 4

Questo si ottiene memorizzando due volte ogni minimo o massimo locale della curva (C e D in fig. 4), così che una linea orizzontale e tangente attraversa sempre un numero pari di punti del contorno.
Questa operazione viene fatta durante il processo di scansione del contorno, ripetendo uno dei punti estremi di esso, se il numero dei punti non risulta essere pari.

3) similmente, quando la linea passa attraverso un flesso del contorno (E ed F in fig. 4) e attraversa un numero dispari di punti, bisogna ripetere uno dei punti di flessione.
Questo modo di operare é mostrato in tab. II per alcuni punti della fig. 4, i punti aggiunti sono racchiusi in parentesi graffe.

Tab. II Coordinate dei punti

CONTORNO ORIGINALE:
(9,26), (10,26), (11,25), (10,24), (9,23), (9,22), (9,21)
(9,20), (10,21), (11,21), (12,21), (13,21), (14,21),
(15,22), (16,23), (17,23), (18,24), (19,24), (20,24),
(21,23), (22,24), (23,23), ...
CONTORNO AUMENTATO:
(9,26), (10,26), (11,25), (10,24), (9,23), (9,22), (9,21)
(9,20), -9,20-, (10,21), (11,21), (12,21), (13,21),
(14,21), (15,22), (16,23), (17,23), -17,23-, (18,24),
(19,24), (20,24), -20,24-, (21,23), (22,24), -22,24-,
(23,23), ...

Per rendere più agevole la ricerca, le coordinate dei punti selezionati vengono partizionate ed ordinate in insiemi, in modo che ognuno di questi contenga solo ascisse ordinate in senso crescente che si riferiscono alla stessa y-coordinata.
Ogni insieme ordinato è detto una y-partizione della struttura rappresentata.
La regione R_k ha una struttura P_k così definita:

$$P_k = (y^k_{min}, y^k_{max}, Y^k_1, Y^k_2, \ldots, Y^k_n)$$

dove y^k_{min} è la più piccola ordinata del contorno,

y^k_{max} è la più grande,

ed $n = y^k_{max} - y^k_{min} + 1$ il numero delle partizioni.

La Y-partizione di P_k per Y_i è:

$$Y^k_i = (X^i_1, X^i_2, \ldots, X^i_r)$$

dove r è il numero di ascisse di questa partizione ed è sempre pari.
In ciascuna partizione le ascisse X sono ordinate in senso crescente ed occupano posizioni dispari o pari della stringa. Ogni coppia (X-dispari, X-pari) delimita un intervallo i cui punti sono interni alla regione esaminata, al contrario le coppie (X-pari, X-dispari) delimitano ascisse di punti esterni.
Le Y-partizioni per $y_1=4$ ed $y_{13}=16$ nel contorno di fig.4 sono:

$$y_1 = (18,18,21,21,22,23)$$

$$y_{13} = (4,5,5,24,25,26,27,28)$$

Da un punto di vista implementativo ogni Y-partizione è rappresentata da una lista mantenuta costantemente ordinata sin dalla fase di caricamento.
L'intera struttura a liste è sostenuta da un vettore contenente i puntatori alle singole Y-partizioni.
Per permettere le elaborazioni successive (OPERAZIONI LOGICHE), le configurazioni a liste che vengono create sono due ed il risultato di ogni operazione viene memorizzato unicamente nella prima di esse.

FASE 3.2: LE OPERAZIONI LOGICHE SULLE CARTE

In questo paragrafo vengono presentati i tre algoritmi fondamentali di operazioni logiche sulle carte: UNIONE, INTERSEZIONE, NEGAZIONE (intesa come tutte le proprietà del bacino esaminato diverse da quella negata).

UNIONE

Date due regioni chiuse A^k ed A^m codificate in forma di Merrill, per costruire l'unione di due Y-partizioni bisogna confrontare ciascuna coppia dispari-pari di ascisse relative ad una partizione con le coppie dispari-pari dell'altra partizione.
Nel caso di fig. 5 si verificano le condizioni

$$X^k_d < X^m_d \qquad X^k_p \geq X^m_p$$

quindi la coppia selezionata, come risulta dalla tab.III è:

dispari	pari
X^k_d	X^k_p

Siano date la due strutture:

$$A^k = (y^k_{min}, y^k_{max}, Y^k_i, Y^k_{i+1}, \ldots, Y^k_r)$$

$$A^m = (y^m_{min}, y^m_{max}, Y^m_j, Y^m_{j+1}, \ldots, Y^m_i, \ldots, Y^m_s)$$

la prima operazione che si compie tra le Y-partizioni è il confronto tra i minimi ed i massimi, in modo da evitare una ricerca inutile ed interessarsi solo delle porzioni di A^k ed A^m per cui a priori sappiamo esistere entrambe le Y-partizioni.

Se $Y^m_{min} < Y^k_{min}$ tutte le liste puntate da Y^m_i tale che

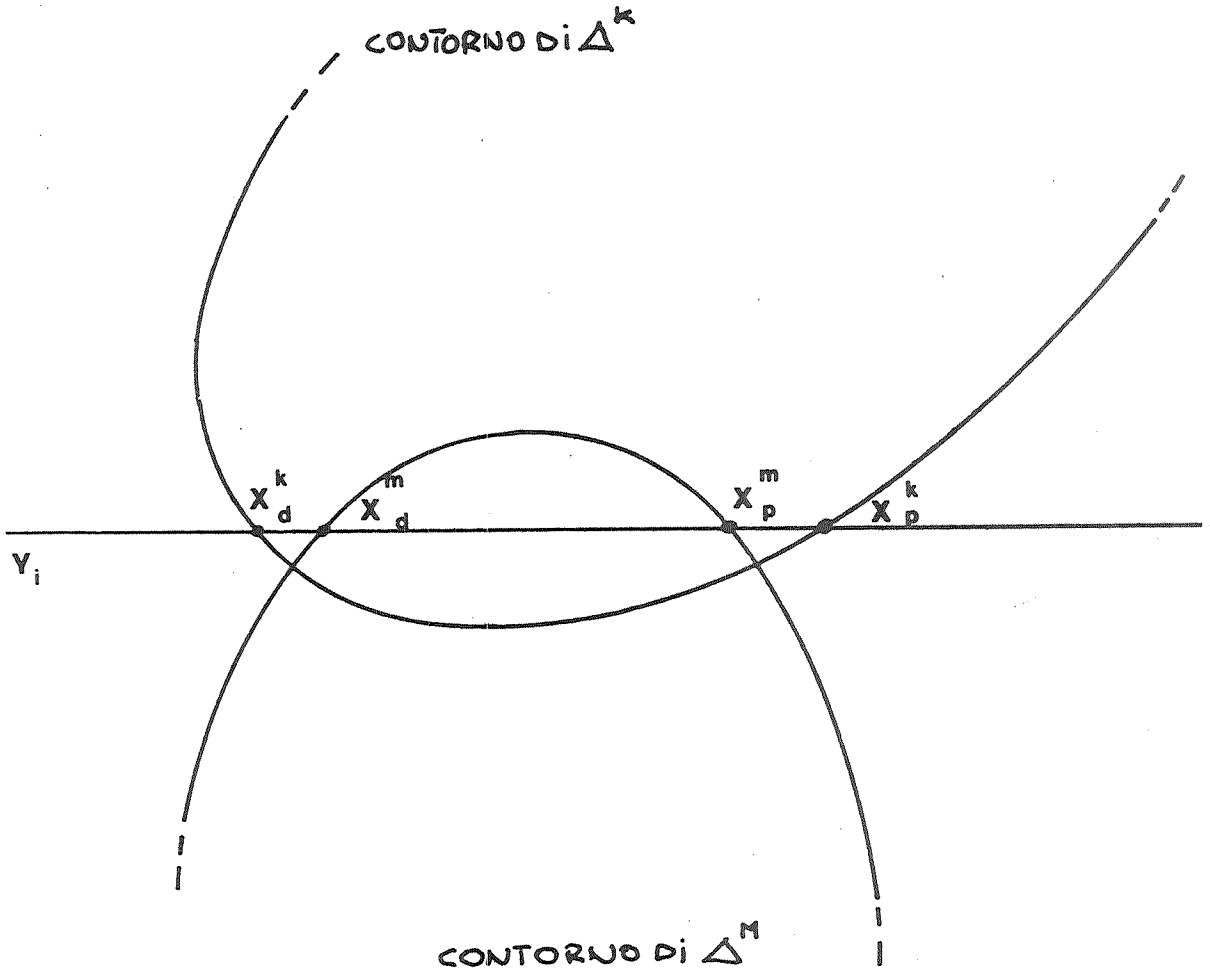


FIG. 5

$$Y^{m}_{min} \leq Y^{m}_{i} < Y^{k}_{min}$$

vengono inserite nella struttura risultante, cioè se sono dipendenti dalla seconda struttura vengono collegate alla prima.
Analoga operazione vien fatta per le Y_{max}.

	DISPARI	PARI	ASSEGNAMENTO D'UNIONE	
	X^{k}_{d}	X^{k}_{p}	X^{r}_{d}	X^{r}_{p}
1)	$\geq X^{m}_{d}$	$< X^{m}_{p}$	X^{m}_{d}	X^{m}_{p}
2)	$< X^{m}_{d}$	$\geq X^{m}_{p}$	X^{k}_{d}	X^{k}_{p}
3)	$< X^{m}_{d}$	$< X^{m}_{p}$	X^{k}_{d}	X^{m}_{p}
4)	$\geq X^{m}_{d}$	$\geq X^{m}_{p}$	X^{m}_{d}	X^{k}_{p}
5)	$= X^{m}_{p}$	non interessa	X^{m}_{d}	X^{k}_{p}
6)	non interessa	$= X^{m}_{d}$	X^{k}_{d}	X^{m}_{p}
7)	X^{m}_{p}	non interessa	X^{m}_{d}	X^{m}_{p}
8)	non interessa	$< X^{m}_{d}$	X^{k}_{d}	X^{k}_{p}

L'algoritmo opera su due strutture a liste ognuna delle quali rappresenta la codifica delle aree interessate dal

l'operazione (fig. 6).

INTERSEZIONE

Questa procedura è del tutto simile a quella di UNIONE variando naturalmente gli assegnamenti delle coordinate per l'area risultante. Una enumerazione delle relazioni e degli assegnamenti per ciascun caso è stata tabulata nella tab. IV.

Tab.IV Assegnamenti dispari-pari dell'intersezione

	DISPARI	PARI	ASSEGN. D'INTERSEZIONE	
	X_d^k	X_p^k	X_d^r	X_p^r
1)	$\geq X_d^l$	$< X_p^l$	X_d^k	X_p^k
2)	$< X_d^l$	$\geq X_p^l$	X_d^l	X_p^l
3)	$< X_d^l$	$< X_p^l$	X_d^l	X_p^k
4)	$\geq X_d^l$	$\geq X_p^l$	X_d^k	X_p^l
5)	$= X_p^l$	non interessa	X_d^k	X_d^k
6)	non interessa	$= X_d^l$	X_p^k	X_p^k
7)	$> X_p^l$	non interessa	niente	
8)	non interessa	$< X_d^l$	niente	

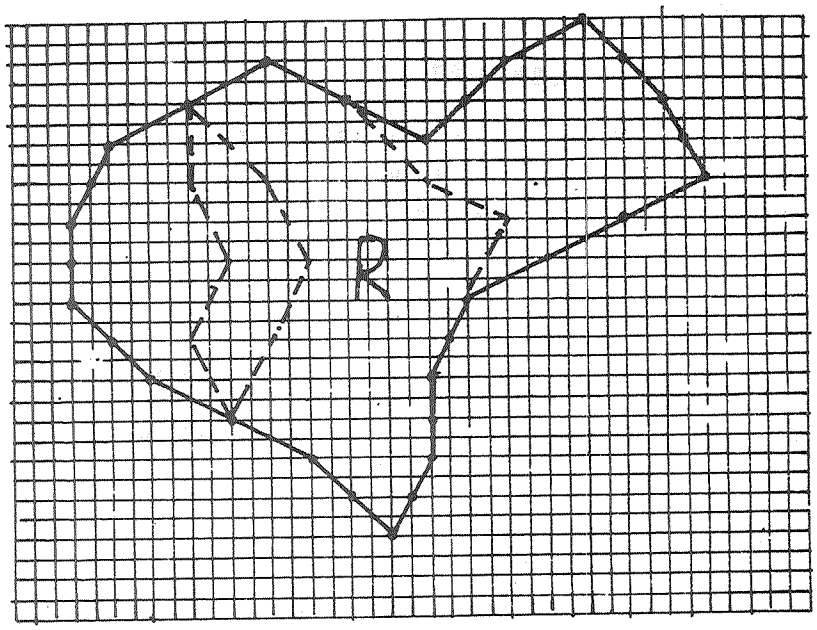
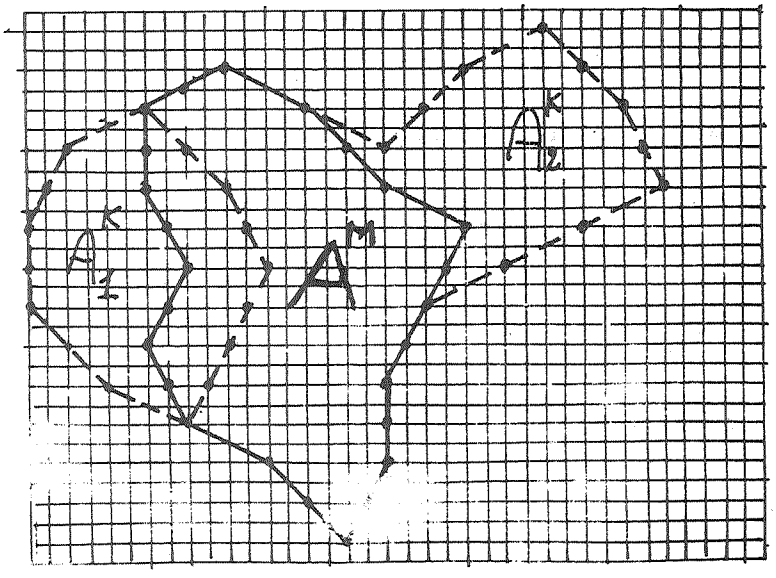


FIG. 6



La ricerca sulle partizioni avrà luogo in questo algoritmo tra:

$$Y_{min} = \max (Y^{k}_{min}, Y^{l}_{min})$$

e

$$Y_{max} = \min (Y^{k}_{max}, Y^{l}_{max})$$

che è il solo intervallo utile perchè possa esserci intersezione non vuota tra le aree K ed L di fig. 7.

NEGAZIONE

L'implementazione di questa operazione è attuata in modo molto rapido attraverso il caricamento dell'UNIVERSO (rappresentato dal contorno esterno dell'area in esame) e della regione di cui si vuole la negazione, nella medesima struttura a liste.
Questo comporta, su ogni lista in cui sono presenti entrambi i contorni, uno scambio di ruoli tra le x-dispari e le x-pari di ogni coppia, così che le'aree' che prima del caricamento dell'insieme UNIVERSO erano interne ad una coppia ora divengono esterne.

FASE 3.3: IL LINGUAGGIO DI INTERROGAZIONE

Per consentire la massima flessibilità delle richieste è stato definito un linguaggio di interrogazione che riconosce espressioni logiche con operatori and, or, not.
A tale scopo abbiamo definito una grammatica le cui produzioni sono date da:

Z:=nome= E
E:=T/E or T
T:=F/T and F
F:=nome/(E)/not F

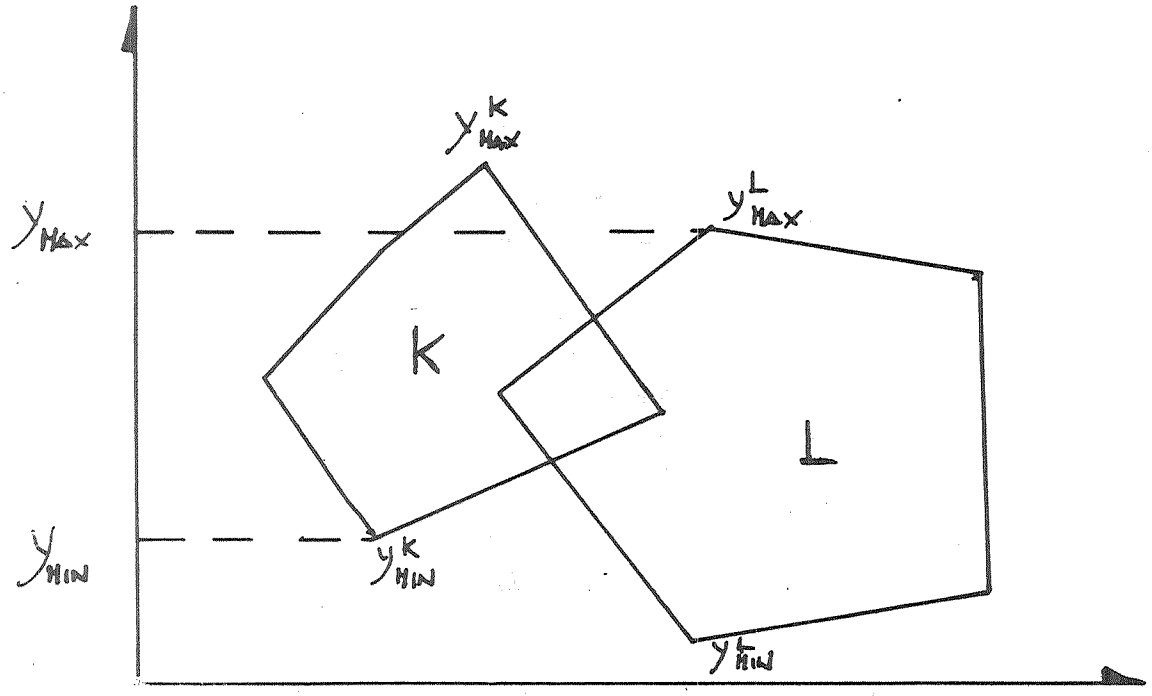


FIG. 7

dove "nome" è il simbolo terminale (assumerà come valori i nomi delle carte da generare se a sinistra del segno di uguale ed i nomi delle carte presenti in archivio se a destra) mentre or, and, not, (,) rappresentano gli operatori ammessi.
L'analizzatore, implementato in linguaggio PL1, consta di due parti distinte: l'analizzatore lessicale e semantico e quello sintattico.
Nella fase lessicale-semantica viene controllata l'eventuale presenza di caratteri o simboli non ammessi, il testo della richiesta è ripulito di eventuali bianchi presenti e limitato a destra da un simbolo speciale di fine stringa ("$").
Per rendere più agevoli le successive elaborazioni si riduce la stringa di input ad una stringa di simboli sostituendo ad ogni terminale "nome" la sua relativa entry in symbol table.
Ad esempio la richiesta:

```
CARTA_A = CARTA_B.and. (CARTA_C.or.CARTA_D)
```

viene trasformata in

```
1 = 2.and.(3.or.4)
```

con relativa symbol table:

1	CARTA_A
2	CARTA_B
3	CARTA_C
4	CARTA_D

L'analizzatore sintattico è un riconoscitore Top-Down a discesa ricorsiva.
Per illustrare il suo funzionamento forniamo di seguito

uno schema delle varie routines in un linguaggio esemplificativo.
Introduciamo alcune convenzioni:
a) SYMBOL è una variabile globale contenente, all'ingresso di ogni routine chiamata, il simbolo della stringa di input da esaminare.
b) La routine SCAN ha la funzione di scorrere sulla stringa di input e mettere il "prossimo" carattere da esaminare nella variabile globale.
c) STACK è un vettore globale rappresentante una pila: l'ultimo valore inserito è indirizzato da LST.
d)Parallelamente alla analisi sintattica della richiesta viene effettuata la generazione del codice oggetto: esso consiste di triple (generate dalla routine TRIPLE) della forma:

operatore, operando 1, operando 2

oppure

operatore, operando 1, ----------

nel caso dell'operazione di negazione.
e) ERROR è, ovviamente, una routine di errore che ritorna il controllo all'utente perchè fornisca una versione corretta della richiesta. Viene anche eliminato anche l'eventuale codice oggetto prodotto sino al momento della chiamata ad ERROR.
f) "vartemp" ha il significato di variabile temporanea in cui è memorizzato il risultato dell'operazione precedente. Volutamente non si è introdotta nessuna differenziazione sui valori che questa variabile può assumere in quanto, il sistema non ha al momento la possibilità di eseguire operazioni di memorizzazione di risultati intermedi o finali. Ne segue che operazioni come la seguente:

CARTA_R = (CARTA_A.or.CARTA_B).and.(CARTA_C.or.CARTA_D)

anche se di fatto corretta sintatticamente e tradotta in una sequenza di triple del tipo:

.or.	CARTA_A	CARTA_B
.or.	CARTA_C	CARTA_D
.and.	vartemp 1	vartemp 2
=	CARTA_R	vartemp 3

non vengono elaborate dal sistema che attualmente lavora solo su due possibili configurazioni e memorizza sempre il risultato di una operazione nella prima di esse.
Il sistema si accorge della impossibilità di proseguire l'elaborazione quando incontra una tripla in cui compaiono due variabili temporanee come operandi.

```
PROCEDURA Z
   LST=0
   IF NXTSYMB ≠ "nome" THEN CALL ERROR;
   LST=LST+1, STACK(LST)="nome";
   CALL SCAN;
   IF NXTSYMB ≠ "=" THEN CALL ERROR;
   CALL SCAN;
         CALL E;
   CALL TRIPLE ("=", STACK(LST), STACK(LST-1));
   LST=LST-2;
   IF LST ≠ 0 THEN CALL ERROR;
END Z;

PROCEDURA E;
   CALL T;
   WHILE NXTSYMB=".or." THEN DO;
   CALL SCAN;
   CALL T;
   CALL TRIPLE (".or.", STACK(LST), STACK(LST-1));
   LST=LST-1; STACK(LST)="vartemp"
   END;
END E;

PROCEDURA T.
   CALL F;
   WHILE NXTSYMB= ".and." THEN DO;
   CALL SCAN;
   CALL F;
   CALL TRIPLE(".and.", STACK(LST), STACK(LST-1));
   LST=LST-1, STACK(LST) ="vartemp";
   END;
```

```
END T;

PROCEDURA F;
 IF NXTSYMB="nome" THEN DO;
    LST=LST+1;
    STACK(LST)="nome";
    CALL SCAN;
    END;
 IF NXTSYMB=".not." THEN DO;
    CALL SCAN;
    CALL F;
    CALL TRIPLE(".not.", STACK(LST),---);
    STACK(LST)="vartemp";
    END;
 IF NXTSYMB ≠ "(" THEN CALL ERROR;
    CALL SCAN;
    CALL E;
    IF NXTSYMB ≠ ")" THEN CALL ERROR;
    CALL SCAN;
END F;
```

L'esempio all'inizio del paragrafo verrà quindi tradotto nelle triple:

```
        .or. CARTA_C(3) CARTA_D(4)
```

e il risultato intermedio è mantenuto nella configurazione 1

```
      .and. vartemp         CARTA_B(2)
        =     vartemp          CARTA_A(1)
```

che dà luogo alla ricostruzione dei contorni della carta risultato ed alla sua restituzione.

Quando l'operazione logica è stata eseguita, si pone il problema della ricostruzione del contorno dell'area risultante.

Sia P la regione risultante, memorizzata in una struttura a liste sotto forma di Merrill; partendo dal basso della struttura si individua il punto che ha minore ordinata e minore ascissa che chiamiamo punto-corrente di ricerca.

Dal momento che lo scopo è quello di ottenere la codifica del contorno, abbiamo pensato di ricostruire una catena sul tipo di quella originaria, quindi dato un punto-corrente si cerca un nuovo punto per mezzo di confronti fra il punto corrente ed i punti delle liste adiacenti.
La ricerca è infatti sempre limitata ad un massimo di tre liste: quella a cui appartiene l'ultimo punto trovato e quelle immediatamente superiore ed inferiore (essendo ciò una caratteristica della codifica di Merrill).
Dato un punto-corrente si individua il nuovo punto corrente attraverso la ricerca di un punto che sia raggiungibile dal vecchio con uno degli spostamenti ammessi.
La strategia di ricerca adottata è sinistra oraria, intendendo con ciò che il contorno viene ricostruito in senso orario e che ad ogni passo si inizia sempre ricercando eventuali punti che rimangono alla sinistra di una persona che percorra il tratto di contorno sinora ricostruito.
Il punto così trovato diviene punto-corrente, viene eliminato dalla lista-risultato e la ricerca continua per raggiungere un nuovo punto.
I links che via via vengono selezionati sono memorizzati; più links successivi uguali, vengono contati in modo da formare una catena i cui elementi sono composti da due parti distinte: nome del link e numero dei links accumulati.
La memorizzazione viene completata col punto di inizio catena in coordinate assolute.
La procedura termina al momento in cui il contorno viene richiuso sul punto di partenza; viene calcolata la superficie della regione-risultato e questa non viene memorizzata se minore di un valore prefissato onde non considerare aree poco significative.

BIBLIOGRAFIA

1) Baxter R.S.: "GINO: a general purpose graphics system for cartography", The Cartographic Journal, 10 (1973).
2) Bonfatti F., Tiberio P.: "Generazione automatica di carte tematiche", Rivista di Informatica, vol.8, n.3, lug_ set. 1978.
3) Fisher H.T.: "SYMAP", Tec. Rep. Lab. for Computer Graphics and Spatial Analysis, Harvard University, Sept. 1975.
4) Freeman H.: "Map data enconding techniques", Atti della scuola "NATO Advanced Study Institute on Map Data Processing', Maratea, giugno 1979.
5) Merrill R.D.: "Representetion of countors and regions for efficient computer search", Comm. ACM, vol. 16, n.2 Feb. 1973
6) Nagy G., Wagle S.G.: "Approximation of polygonal thematic maps by cellular maps", Tec. Rep. Department of computer science University of Nebraska, Feb. 1978.
7) Nagy G. et al.:"SMAPS: soil map processing", Tec. Rep. Depatment of computer science University of Nebraska, Aug. 1978.
8) Salomon K.B.:"A fortran IV program which determines that region of a polygon within a polygonal boundary", Computer Geosciences, vol. 4, 1978.
9) Sutherland, Hodgaman: "Reentrant polygon clipping", Comm. ACM, vol. 17, N. 2, Jan. 1974.
10) Weiler K.:"Polygon comparison using a graph representation", SIGGRAPH 80 Proceedings, SIGGRAPH, 1980.