CLUSTERING SU GRAFO

M. MARAVALLE, Facoltà di Economia e Commercio
Università degli Studi dell'Aquila

ABSTRACT

The present paper deals with the following problem: given a connected graph G with n nodes, partition its set of nodes into p classes such that the subgraph induced by each class is connected and a given function, inertia, of the partition is minimized. This paper includes : a heuristic algorithm for general graphs based on the solution of a sequence of restricted problems where G is replaced by one of its spanning tree ; a complete set of real-life applications and a comparison with a hierarchical algorithm.

KEYWORDS : Clustering, constrained clustering.

1. INTRODUZIONE

In molti problemi di classificazione gli oggetti possono essere condizionati a particolari vincoli. I più semplici tipi di vincolo sono legati alla numerosità (ad esempio ciscun gruppo non deve contenere più di un certo numero massimo di soggetti). Un altro tipo di vincolo che spesso si incontra è quello legato alle condizione di contiguità dei membri di un gruppo. In questo caso gli oggetti sono delle località (possono essere città, comuni, quartieri) per le quali l'appartenenza ad uno stesso gruppo richiede che siano geograficamente contigue e che, come si dice in gergo, formino un distretto (particolarmente interessanti per gli studi dei bacini di utenza di servizi quali poste, telefoni - si veda l'esempio di C. Roche 1978 - ecc.). Per poter dare una definizione formale di distretto è necessario introdurre il concetto di grafo, i cui vertici sono le località e laddove esiste uno spigolo fra i due vertici significa che fra queste due località c'è contiguità che in termini pratici indica il collegamento diretto tramite una strada, una ferrovia, un gasdotto ecc. .

La condizione che i siti formino un distretto è espressa dalla condizione che il gruppo dei vertici indotti dalla classificazione sia un sottografo connesso (per la terminologia dei grafi si è utilizzato il testo di Cerasoli, Eugeni e Protasi 1988).

I vincoli di contiguità intervengono anche in altre aree applicative quali la geologia e la botanica, si veda in proposito Gordon 1973 e 1978.

I due principali obiettivi della classificazione sono quelli di massimizzare l'omogeneità all'interno di un gruppo (elementi dello stesso gruppo devono essere simili fra loro) come pure la diversità fra gruppi diversi (elementi di gruppi diversi devono essere differenti fra loro). Questi due obiettivi sono solo apparentemente in contrasto giacché grazie al lemma di Huyghens si può dimostrare che minimizzare la somma delle inerzie interne equivale a massimizzare l'inerzia esterna, si veda Benzécri 1980)

Nel presente articolo è trattato il problema di *classificazione con inerzia minima su un grafo* . A partire da:

- un grafo connesso $G \equiv (V,E)$ con $n=|V|$ vertici ed $m=|E|$ spigoli;
- una funzione vettoriale che associa ad ogni vertice i un vettore di caratteristiche q-dimensionali x(i), cioe' una funzione x: V------> $\Re^q$;
- un intero p , numero dei gruppi, tale che $1 \leq p \leq n$;

si cerca una p-partizione $\pi = \{ C_1,.....,C_p \}$ dei vertici V che sia *ammissibile* nel senso che i sottografi $G(C_k)$ indotti dalla partizione siano tutti connessi; e che sia minima l'inersia interna della partizione

$$h(\pi) = \sum_{k=1}^{p} I(C_k) \qquad (1)$$

dove per inerzia interna del k-mo cluster si intende la quantita' :

$$I(C_k) = \sum_{i \in C_k} \| x(i) - b(C_k) \|^2 ,$$

mentre $\| \cdot \|$ è una norma euclidea in $\Re^p$ ed infine $b(C_k)$ è il baricentro di C_k :

$$b(C_k) = \frac{1}{|C_k|} \sum_{i \in C_k} x(i) \quad .$$

Simili problemi sono stati già trattati nella letteratura scientifica, ed in particolare L. Lebart 1978 ha proposto un algoritmo di tipo gerarchico nel quale ad ogni iterazione non vengono aggregati gli elementi (sottogruppi o semplici individui statistici) più prossimi bensi' quelli che lo sono subordinatamente all'esistenza di uno spigolo che li possa connettere fra loro. Di tale algoritmo esiste anche un programma per computer , AGRAF .

In questo articolo viene presentata nel paragrafo 2 una euristica originale basata su un algoritmo di classificazione non gerarchico concatenato alla soluzione di problemi di ottimo su alberi generati dal grafo stesso. Essendo ben noti i vantaggi/svantaggi di questo tipo di algoritmo rispetto a quelli gerarchici, è stata realizzata, nel paragrafo 3, una comparazione dei risultati di tale euristica comparati con quelli di AGRAF per meglio valutarne le caratteristiche di performance su alcuni problemi reali che coprono un insieme di casi sufficientemente ampio.

2. METODOLOGIA PER LA CLASSIFICAZIONE (EURISTICA A DUE STADI)

Indicato con $\Pi_p(G)$ l'insieme di tutte le p-partizioni di V ammissibili in G; data una partizione π, uno spigolo è chiamato *interno* se i due vertici appartengono entrambi ad uno stesso gruppo indotto dalla partizione π, altrimenti viene indicato come *esterno*. Sia

$$\zeta_p(G) = \min \{ h(\pi) : \pi \epsilon \Pi_p(G) \}, \qquad (2)$$

dove $h(\pi)$ è dato dalla (1).

In Maravalle e Simeone 1992, è stata dimostrata la seguente identità:

$$\zeta_p(G) = \min \zeta_p(T)_{T \epsilon \mathcal{T}} \qquad (3)$$

dove $\mathcal{T}$ è l'insieme di tutti gli alberi di G e per un grafo qualsiasi.

L'identità (3) suggerisce la seguente euristica per il problema (2):

MIDAS (Metodo Iterativo Di Aggregazione Spaziale)

STEP 1 : (*albero iniziale*) Si cerca un albero iniziale "buono" T in G;

STEP 2 : (*partizione iniziale*) Si cerca una partizione iniziale "buona" $\bar{\pi}$ in $\Pi_p(T)$;

STEP 3 : (*ottimizzazione sull'albero*) Partendo dalla partizione $\bar{\pi}$ esegue una ricerca locale per trovare una soluzione quasi-ottimale π^* al problema *ristretto* :

$$\min \{ h(\pi) \mid \pi \epsilon \Pi_p(T) \}$$

STEP 4 : (*modificazione dell'albero*) Si cerca, se possibile, un'altra p-partizione $\bar{\pi}$ ed un altro albero $\bar{T}$ di G tale che :

I) $h(\bar{\pi}) < h(\pi^*)$

II) $\bar{\pi} \notin \Pi_p(T)$

III) $\bar{\pi} \in \Pi_p(\bar{T})$

Se non si trova nessuna coppia $(\bar{\pi}, \bar{T})$ allora il risultato finale è raggiunto con la partizione corrente π^* (poichè π^* è ammissibile in T lo sarà anche in G);

altrimenti sostituisce $\bar{\pi}$ con π^* e si torna allo STEP 3;

END

Discutiamo ora in dettaglio i quattro passi ora descritti

STEP 1 (*albero iniziale*)

Un "buon" albero iniziale può essere trovato nel modo che segue . Definita come $d(i,j)=|| x(i) - x(j)||^2$ la dissimilarità tra i vertici i e j. Si assegna a ciascuno spigolo <i,j> del grafo G una lughezza pari a d(i,j). Successivamente si calcola l'albero di lunghezza minima T di G attraverso l'algoritmo di Kruskal (Lawler, 1976). Questa scelta razionale di T fa si che la lunghezza media di uno spigolo in T sia in generale sufficientemente piccola; quindi è sperabile che, se $\pi=\{C_1,C_2,.....,C_p\}$ è ottimale per T, l'inerzia interna di ogni cluster C_k ed il valore ottimale $\zeta_p(T)$ siano sufficientemente piccoli. I risultati sperimentali del prossimo paragrafo indicano che questa scelta per l'albero iniziale è "buona" nel senso che, partendo da T, MIDAS deve generare pochissimi alberi.

STEP 2 (*partizione iniziale*)

Una buona partizione iniziale $\bar{\pi} \in \Pi_p(T)$ può essere ottenuta attraverso una procedura "ghiotta". In primo luogo si nota che tagliando $p-1$ spigoli, si ottiene una foresta F_p di G consistente in p alberi; si indichino con $C_1,C_2,.....,C_p$ gli insiemi dei vertici di questi alberi. Allora $\pi=\{C_1,C_2,.....,C_p\}$ è una p-partizione ammissibile di T. Eliminando ancora un altro spigolo questo avrà l'effetto di separare un certo cluster C_h in due nuovi clusters $C_{h'}$ e $C_{h''}$. Dopo la separazione il decremento di inerzia interna complessivo è

$$\Delta = \mathbb{I}(C_h) - \mathbb{I}(C_{h'}) - \mathbb{I}(C_{h''}) \geq 0 \ .$$

Si può ora dare una semplice descrizione della procedura "ghiotta".

PROCEDURA GHIOTTA

Passo 1 : Sia $F := T$; $k := 1$;

Passo 2 : Si taglia lo spigolo e di F cui corrisponde la maggiore diminuzione di inerzia interna Δ ; si sostituisce F con $F - e$;

Passo 3 : Se $k = p-1$ allora STOP : si ritorna alla foresta F ed alla partizione associata $\bar{\pi} \in \Pi_p(T)$; altrimenti si incrementa k di un'unità e si torna al Passo 2.

FINE

Il decremento di inerzia interna Δ risultante dal taglio di uno spigolo può essere calcolato con delle semplici formule iterative .

STEP 3 (*ottimizzazione sull'albero*)

L'ottimizzazione (euristica) sull'albero si basa sul fatto esiste una corrispondenza 1-1 fra le partizioni in $\Pi_p(T)$ ed i $(p-1)$-sottoinsiemi di spigoli di T, ed è tale che i $p-1$ spigoli in un sottoinsieme sono precisamente quelli esterni nella corrispondente partizione, cioè sono gli spigoli che devono essere tagliati dalla struttura iniziale per ottenere quella partizione.

Se viene indicata col nome di *base* ogni $(p-1)$-sottoinsieme di spigoli di T, allora uno ***scambio*** (o *pivoting*) consiste nel sostituire uno spigolo nella base con un altro spigolo fuori base.

Un semplice algoritmo di ricerca locale per il problema della classificazione su un albero T è il seguente. Si parte con la base corrispondente alla partizione ammissibile $\overline{\pi}$; ad ogni iterazione verrà indicata con B la base corrente. Se lo spigolo $u \in B$ è sostituito con lo spigolo $v \notin B$, si ottiene una nuova base B'. In corrispondenza ci sarà una variazione $\Delta_v^u = h(\pi) - h(\overline{\pi})$ nella funzione obiettivo, dove π e $\overline{\pi}$ sono le partizioni corrispondenti rispettivamente a B ed ha B'. Se $\Delta_v^u < 0$ si effettua lo scambio e la procedura iterativa continua, se $\Delta_v^u \geq 0$ per tutti gli spigoli $u \in B$ e $v \notin B'$ allora STOP : la partizione corrente π è l'ottimo cercato.

STEP 4 (*modificazione dell'albero*)

Sia T un albero di G e $\pi \in \Pi_p(T)$: si assume che π venga ottenuta attraverso l'algoritmo di scambio descritto precedentemente.

Si selezioni uno spigolo esterno u. I suoi vertici terminali siano i ed j appartenenti a due differenti classi C_h e C_k.

Si considera ora la nuova partizione $\overline{\pi}$ ottenuta partendo dalla π facendo migrare il vertice i dal gruppo C_h a C_k; si dirà in questo caso che $\overline{\pi}$ è ***adiacente*** a π . La migrazione causerà una variazione della funzione obiettivo; se $\overline{\pi}$ è strettamente migliore di π, $\overline{\pi}$ è ammissibile in T : ovvio in quanto π è stata ottenuta attraverso un algoritmo di scambio. La procedura tenta allora di effettuare un intervento di "chirurgia plastica" sull'albero T allo scopo di ottenere un nuovo albero $\overline{T}$ in cui $\overline{\pi}$ sia ammissibile.
È utile, a tal fine, vedere su un piccolo esempio l'essenza di questa procedura.
Prima di descrivere in dettaglio la procedura è necessario introdurre alcune notazioni. Ogni coppia (T,π) induce sul grafo G una partizione degli spigoli in quattro tipologie :

α-*spigolo* : spigolo esterno non appartiene all'albero

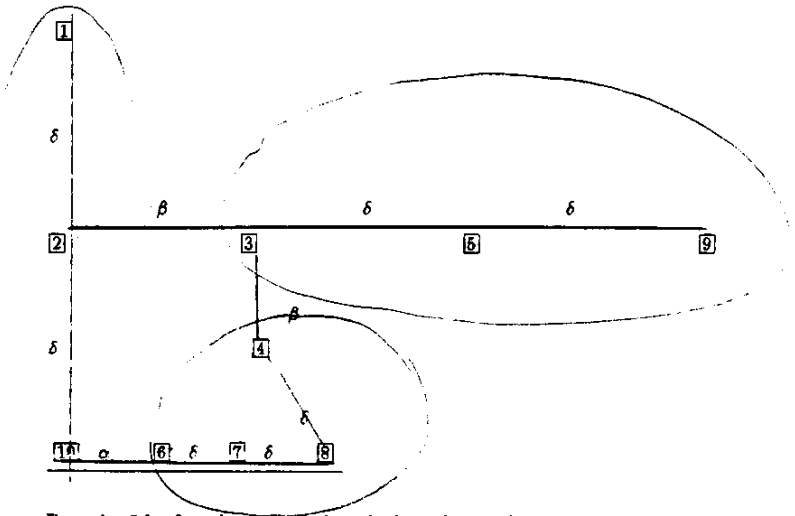
β-*spigolo* : spigolo esterno appartenente all'albero

γ-*spigolo* : spigolo interno non appartenente all'albero

δ-*spigolo* : spigolo interno appartenente all'albero

Per ciascuna tipologia di vertici è definito il suo ...-grado come il numero di ...-spigoli uscenti dal vertice stesso.

Ad esempio nella Fig. 1 è schematizzato un grafo G=(V,E) diviso in tre gruppi e sono indicate le diverse tipologie per gli spigoli :



Il vertice 5 ha δ-grado=2. Il vertice 2 ha b-grado=1 e δ-grado =2.

LEMMA . Se $\bar{\pi}$ è adiacente a π, una condizione necessaria e sufficiente perche' $\bar{\pi}$ sia ammissibile in T è che risulti

a) il δ-grado del vertice x che migra da un cluster all'altro e' 1

b) esiste un β-spigolo che connette x con qualche altro vertice del gruppo C_j in cui x emigra (si noti che il β-spigolo e' necessariamente unico).

Viene ora descritta la procedura di modificazione dell'albero. Come precedentemente indicato , sia x il vertice candidato a migrare da C_i a C_j ed u uno spigolo esterno di G che connette x con qualche altro nodo y di C_j.

MODIFICAZIONE DELL'ALBERO

Step 1 . Se il δ-grado di x è maggiore di 1 passa allo step successivo, altrimenti si va allo Step 3.

Step 2 . Siano $v_1,.....,v_d$ (d>1) i δ-spigoli incidenti x ed S l'albero parziale di T indotto dalla classe C_i. Il grafo $S - x$ è una foresta con d componenti connesse $S_1,S_2,.....,S_d$. Si tratta ora di trovare un γ-spigolo v_0 che connetta S_1 con qualche altro S_k (a tale scopo viene usata una procedura di etichettamento per individuare tale γ-spigolo): Se non si trova alcun γ-spigolo si passa allo Step 9; altrimenti il vertice v_0 viene dichiarato δ-spigolo mentre v_1 γ-spigolo cosi' che il δ-grado di x diminuisce di una unita'. Si ripete lo Step 2 eventualmente fino a che il δ-grado di x non diventi 1. Successivamente si passa allo Step 3.

Step 3 . Sia $v=(x,z)$ l'unico δ-spigolo incidente x. Se il β-grado di x è zero si passa allo Step 6; altrimenti si passa allo step successivo.

Step 4 . Se u è un β-spigolo, v viene dichiarato β-spigolo e si passa allo Step 8. Se u è un α-spigolo e c'è un β-spigolo w che connette x con qualche y' di C_j allora si va allo Step7 , altrimenti passa allo Step 5.

Step 5 . Si indichi con P il cammino (unico) che collega x con y sull'albero T (questo cammino puo' individuarsi facilmente attraverso una procedura di etichettamento). Se P include il δ-spigolo v allora si passa allo Step 6 altrimenti P includerà un β-spigolo w incidente x : in quest'ultimo caso si passa allo Step 7.

Step 6 . Si dichiara u come δ-spigolo, v un α-spigolo e si passa allo Step 8.

Step 7 . Si dichiara u come δ-spigolo, v un β-spigolo , w un γ-spigolo e si passa allo Step 8.

Step 8 . Ciascun γ-spigolo incidente x è dichiarato come α-spigolo; ciascuno α-spigolo differente da u che connette x con qualche $y'' \in C_j$ viene dichiarato come γ-spigolo. La tipologia di ogni altro spigolo incidente x non cambia. Il processo di modificazione dell'albero è completato ed un nuovo albero $\overline{T}$ viene ad essere ammissibile in $\overline{\pi}$.

Step 9 . L'algoritmo non riesce a trovare un albero $\overline{T}$ in cui $\overline{\pi}$ sia ammissibile e la migrazione di x non puo' avvenire.

In quest'ultimo caso, si tenta di verificare la possibilita' di migrare di y da C_j a C_i. Se anche in questo caso l'algoritmo non riesce a modificare l'albero allora si tenta con un altro spigolo esterno u' e si ripete la procedura. Se complessivamente non avviene nessuna migrazione di vertici significa che non c'è nessuna partizione $\overline{\pi}$ che migliora quella di partenza π, che resta percio' la partizione finale.

5 APPLICAZIONI

5.1 Definizione del piano degli esperimenti

Sono stati presi in esame cinque grafi , rispettivamente: a) Grafo TOSCANA b) Grafo ROMA c) Grafo CAMPANIA d) Grafo LAZIO e) Grafo ALTOLAZIO e nella tabella seguente sono fornite sinteticamente le caratteristiche generali dei cinque grafi.

Grafo	numero vertici	numero spigoli	numero caratteristiche	$\rho = \frac{\text{spigoli}}{\text{vertici}}$
TOSCANA	287	773	1	2.69
ROMA	121	302	1	2.50
CAMPANIA	81	165	6	2.04
LAZIO	412	566	1	1.37
ALTOLAZIO	210	331	5	1.58

Allo scopo di avere un elemento di comparazione sono stati messi a confronto i risultati ottenuti con un altro algoritmo di classificazione vincolata AGRAF , di L. Lebart (1978). A differenza di *MIDAS* AGRAF e' un algoritmo ascendente gerarchico.

3.2 Indicatori di performance

Per un confronto fra le diverse partizioni e fra i due algoritmi, MIDAS e AGRAF, sono stati presi in esame i seguenti indici, per ogni predefinito numero di partizioni p (indice che viene per semplicità omesso)

$\%_i = \frac{h(\pi_i)}{h(\pi_0)} \cdot 100$ Percentuale d'inerzia interna corrispondente alla partizione π_i iniziale per l'algoritmo MIDAS [$h(\pi_0)$ è la funzione normalizzante relativa ad un gruppo unico, partizione triviale].

$\%_f = \frac{h(\pi_f)}{h(\pi_0)} \cdot 100$ Percentuale d'inerzia interna corrispondente alla partizione π_f finale . (algoritmo MIDAS o AGRAF).

$\Delta_m = \frac{h(\pi_f)}{h(\pi_i)} \cdot 100$ Guadagno percentuale di inerzia in MIDAS.

$\Delta = \frac{h(\pi_f^a) - h(\pi_f^m)}{h(\pi_f^a)} \cdot 100$ Guadagno percentuale fra le partizioni di MIDAS (π_f^m) e quelle di AGRAF (π_f^a)

Pivot = numero di scambi effettuati nella fase di ottimizzazione sull'albero da MIDAS.

Migr = numero di scambi fra cluster contigui effettuati nella fase di modificazione dell'albero (MIDAS).

Tree = numero di alberi generati (MIDAS).

Tempi di CPU relativi al sistema 1100/20 Univac con sistema opertativo Exec8.

4.3 Analisi e discussione dei risultati

Nell'appendice sono riportate le cinque tabelle relative al confronto puntuale fra le caratteriche di performance dei due algoritmi.

Da questetabelle si possono evincere alcune note generali :

- In tutti i casi contemplati l'algoritmo AGRAF è dominato da MIDAS nel senso che se si prende come criterio di bontà di una partizione π la funzione $h(\pi)$ allora in tutti i casi esaminati, ovviamente a parità di numero di partizioni, risulta

$$h(\pi^a) \geq h(\pi^m)$$

valendo il segno = nel solo caso di p=3 per il grafo CAMPANIA. Un solo caso sui 90 esaminati !

- Si verifica inoltre la circostanza che già l'algoritmo greedy di MIDAS costruisce partizioni migliori, nel senso di cui sopra, rispetto alla partizione finale ottenuta da AGRAF. Nei succitati 90 casi questo si verifica per ben 77 volte. Solo per 14 casi si ha $h(\pi_i)$ di MIDAS inferiore alla $h(\pi_f)$ di AGRAF; rispettivamente per il grafo ALTOLAZIO quattro volte nei casi di p=13, 14, 15 e 16; nel grafo Roma cinque volte nei casi p=3,5,6,7 e 8; nel grafo LAZIO ancora quattro volte per p=4,5,10 e 11 mentre un solo caso, per p=20 , nel grafo della TOSCANA.

La fase di migrazione fra gruppi, relativamente all'algoritmo MIDAS, è sempre significativa per quanto attiene i guadagni d'inerzia, pur se questi sono inferiori in generale a quelli relativi all'algoritmo che la precede. I guadagni medi sono,in ordine crescente, 2.47, 4.99, 5.08, 5.42 e 10.16 rispettivamente per i grafi CAMPANIA,TOSCANA; ALTOLAZIO, LAZIO e ROMA.

-E' noto come negli algoritmi di tipo non-gerarchico si presenti anche il problema della scelta del numero dei gruppi. A tal fine un indicatore interessante è il "guadagno d'inerzia" fra due successive partizioni $v_p = \Delta_m(p\text{-}1) - \Delta_m(p)$. L'andamento di questa funzione del numero dei gruppi (p) mostra abitualmente un forte decremento iniziale per poi presentare uno o piu' massimi relativi per valori di p compresi nell'intervallo $3 \div 20$. Questo fatto "legittima" le motivazioni che hanno spinto a prendere in esame questo intervallo per il numero dei gruppi.

-Δ sembra correlato con ρ. Infatti i guadagni medi, cioè sulle 18 partizioni effettuate per ciascun grafo, sono nell'ordine 6.64, 6.66, 8.04, 22.34 e 66.65 rispettivamente per LAZIO (ρ=1.37), CAMPANIA (ρ=2.04), ALTOLAZIO (ρ=1.58), TOSCANA (ρ=2.69) e ROMA (ρ=2.5). Come si puo' notare, pur

non rispettando fedelmente l'ordine di ρ questo dato medio mette in evidenza una sostanziale correlazione positiva (0.64) tra i due indici.

- il numero di pivot è generalmente basso e sembra essere correlato con il numero di vertici quando il numero di gruppi richiesto è elevato . Per $p \leq 10$ il numero di pivot è molto basso. Se ordiniamo i grafi in ordine crescente del numero dei vertici abbiamo la sequenza CRATL (Campania Roma Altolazio Toscana Lazio) e rispettivamente 0.39 ,0.26 ,0.85 ,0.77 e 0.89 sono le correlazioni tra il numero dei pivot ed il numero delle parti nei quali è ripartito il grafo in questione. I valori medi del numero dei pivot sono, rispettivamente 1.22, 1.44, 2.33, 3.72 e 6.39 in rigoroso ordine con la numerosità dei vertici del grafo.

- il numero delle migrazioni sembra correlato, cioè cresce, al crescere del numero di vertici del grafo. Infatti se ordiniamo i grafi secondo valori del numero di migrazioni crescenti, otteniamo la sequenza dei grafi CRATL (Campania Roma Altolazio Toscana Lazio) che è , a meno di uno scambio fra Toscana e Lazio, lo stesso ordinamento dei grafi per numerosità dei vertici.

3.4 Tempi di calcolo

E' interessante anche il confronto fra i tempi di esecuzione dei due algoritmi, pur tenendo conto della differenza fondamentale che riconosce AGRAF basato su un algoritmo di tipo ascendente gerarchico mentre MIDAS è basato su uno di tipo non-gerarchico. Il confronto è stato fatto utilizzando lo stesso computer, un Univac 110/20. AGRAF fornisce, per il grafo ALTOLAZIO, un tempo complessivo di 17.6 sec di CPU per fornire i risultati relativi a tutte le diciotto partizioni in p classi ($3 \leq p \leq 20$). Per quanto riguarda MIDAS sono stati fatti diciotto "passoggi" diversi essendo l'algoritmo non gerarchico. Nella Tavola 1 sono riportati i tempi di calcolo per i diversi valori di p relativamente al grafo dell'ALTOLAZIO mentre nella Tavola 2 sono iportati quelli relativi al grafo della CAMPANIA .

Tav. 1 Tempi di esecuzione di MIDAS (in secondi). Grafo ALTOLAZIO

numero di classi	tempo di esecuzione	numero di classi	tempo di esecuzione
3	5.89	12	18.03
4	6.61	13	15.08
5	5.20	14	10.39
6	7.23	15	11.76
7	18.54	16	12.15
8	8.45	17	25.63
9	10.64	18	22.80
10	10.90	19	21.85
11	12.11	20	25.13

Tav. 2 Tempi di esecuzione di MIDAS (in secondi). Grafo CAMPANIA

numero di classi	tempo di esecuzione	numero di classi	tempo di esecuzione
3	5.89	12	18.03
4	6.61	13	15.08
5	5.20	14	10.39
6	7.23	15	11.76
7	18.54	16	12.15
8	8.45	17	25.63
9	10.64	18	22.80
10	10.90	19	21.85
11	12.11	20	25.13

BIBLIOGRAFIA

Benzécri J.-P. & F. (1980) *Pratique de L'Analyse Des Données*. Vol.1. Dunod - Paris.

Cerasoli M.,Eugeni, F., Protasi M. (1988) *Elementi di Matematica Discreta*. Zanichelli - Bologna.

Christofides, N. (1975) : *Graphe Theory: an algorithmic approach*. Academic Press, New York.

Gordon A.D. (1973) Classification in the presence of constraints, *Biometrics* - 29 -821 ÷ 827.

Gordon A.D. (1987) Classification and Assignment in Soli Science, *Soil Use and Management*-3-3 ÷ 8.

Lawler E. (1976) . *Combinatorial Optimization: Network and Matroid,* Holt, Rinehart & Winston.

Lebart L. (1978) . Programme d'aggrégation avec contraintes. *Les Cahiers de l'Analyse des Données,* vol. 3 -275 ÷ 287. Dunod, Paris.

Roche C. (1978) . Exemple de classification hiérarchique avec contraintes de contiguité . *Les Cahiers de l'Analyse des Données,* vol. 3 -289 ÷ 305. Dunod, Paris.

Maravalle M. e Simeone B. (1992) - A two-stage heuristic for optimal graph partitioning. Tecnical Report .

Murtagh F. (1985) A Survey of Algorithms for contiguity-constrained clustering and related problems. *The Computer Journal* - 28 - 82 ÷ 88.

APPENDICE

GRAFO CAMPANIA

caratteristiche generali: 81 vertici ; 165 spigoli ; 6 caratteristiche

	M I D A S						AGRAF	
p	l_i	l_f	Δ_m	pivot	migr	tree	l_f	Δ
3	67.418	67.418	0.00	0	0	1	67.418	0.00
4	61.006	61.006	0.00	0	0	1	62.120	1.83
5	55.786	55.786	0.00	0	0	1	56.567	1.40
6	50.825	50.825	0.00	0	0	1	51.104	0.55
7	46.446	44.865	3.52	1	15	2	47.230	5.28
8	43.609	43.442	0.38	0	1	2	44.495	2.42
9	40.407	40.241	0.41	0	1	2	40.672	1.07
10	36.234	36.118	0.32	4	3	2	37.90	4.93
11	33.915	31.708	6.96	3	6	2	35.617	12.33
12	32.037	29.950	6.97	2	8	2	33.48	11.82
13	30.185	28.380	6.36	1	13	2	31.67	11.59
14	28.662	27.963	2.50	2	11	2	30.122	7.73
15	27.444	26.743	2.62	3	11	2	29.770	11.33
16	26.296	25.525	3.02	2	14	2	27.600	8.15
17	25.260	24.488	3.15	1	15	2	27.09	10.62
18	24.362	23.739	2.62	1	15	2	25.96	9.35
19	23.452	22.828	2.83	1	17	3	25.09	9.95
20	22.609	21.987	2.83	1	15	3	24.08	9.55

GRAFO ROMA

caratteristiche generali: 121 vertici ; 302 spigoli ; 1 caratteristica

	MIDAS						AGRAF	
				performance				
p	t_i	t_f	Δ_m	pivot	migr	tree	t_f	Δ
3	24.453	21.950	20.51	0	30	2	22.05	0.456
4	15.702	13.344	17.67	0	18	2	15.882	19.04
5	13.938	12.366	12.71	0	5	2	12.937	4.61
6	11.866	10.307	15.12	2	5	2	11.20	8.63
7	10.388	8.886	16.90	3	15	2	9.27	4.39
8	8.516	7.161	18.92	3	5	2	8.50	18.71
9	7.038	5.740	22.61	3	15	2	7.21	25.61
10	6.080	4.827	25.95	2	26	3	6.79	40.58
11	4.890	4.816	1.5	1	21	2	6.63	37.77
12	4.270	4.258	0.28	0	27	2	6.28	47.42
13	3.918	3.905	0.33	0	27	2	5.47	40.25
14	3.573	3.561	0.33	1	25	2	5.78	62.36
15	3.222	3.210	0.37	0	27	2	5.62	75.08
16	2.957	2.945	0.41	0	19	2	5.51	87.41
17	2.733	2.721	0.44	0	17	2	5.40	98.53
18	2.161	1.904	13.5	5	12	2	5.33	179.94
19	1.937	1.680	15.29	2	11	2	5.31	216.07
20	1.583	1.583	0.00	4	1	2	5.26	232.91

GRAFO ALTOLAZIO

caratteristiche generali: 210 vertici ; 331 spigoli ; 5 caratteristiche

	MIDAS						AGRAF	
				performance				
p	t_i	t_f	Δ_m	pivot	migr	tree	t_f	Δ
3	82.823	80.702	2.63	1	15	2	83.21	3.10
4	77.900	76.071	2.40	1	14	2	81.40	7.00
5	73.307	71.509	2.51	0	11	2	77.34	8.15
6	69.811	66.278	5.33	0	13	2	75.86	14.45
7	66.236	62.192	6.50	3	33	4	71.688	15.27
8	63.648	60.344	5.47	0	24	2	68.558	13.62
9	61.079	57.972	5.36	1	21	2	65.04	12.196
10	58.907	57.580	2.30	1	15	2	61.842	7.40
11	56.974	55.625	2.43	1	15	2	58.912	5.915
12	55.273	52.802	4.68	2	26	3	56.	6.06
13	53.605	50.168	6.85	2	19	3	53.	5.64
14	51.978	47.469	9.66	3	26	2	50.	5.33
15	50.359	47.455	6.24	4	30	2	49.37	4.03
16	48.680	45.720	6.47	5	30	2	48.08	5.16
17	46.997	44.279	6.14	5	29	3	47.505	7.27
18	45.350	42.864	5.8	4	25	3	46.212	7.82
19	43.853	41.370	6.00	4	25	3	44.98	8.73
20	42.532	40.648	4.63	5	18	3	43.762	7.65

GRAFO LAZIO

caratteristiche generali: 412 vertici ; 566 spigoli ; 1 caratteristica

	MIDAS						AGRAF	
				performance				
p	l_i	l_f	Δ_m	pivot	migr	tree	l_f	Δ
3	63.667	61.448	3.61	1	12	2	64.05	4.25
4	58.830	57.022	3.17	1	18	2	57.27	0.43
5	53.225	51.665	3.02	1	27	2	52.08	0.83
6	48.392	47.063	2.82	1	23	2	49.09	4.30
7	45.219	44.037	2.68	1	18	2	46.52	5.6
8	42.301	41.094	2.94	1	24	2	42.7	3.9
9	33.404	30.695	8.82	9	62	2	33.80	10.10
10	31.225	29.144	7.14	9	72	2	30.64	5.3
11	30.541	28.613	6.74	8	62	2	29.73	3.9
12	28.312	26.430	7.12	8	82	3	28.8	8.9
13	27.622	26.243	5.25	7	46	2	28.1	7.00
14	26.502	25.064	5.73	7	49	2	27.28	8.8
15	25.514	24.337	4.83	8	96	3	26.43	8.6
16	24.560	23.175	5.97	8	103	3	25.62	10.5
17	23.677	22.178	6.76	12	102	3	24.18	9.02
18	23.008	21.529	6.87	10	105	3	23.47	9.01
19	22.395	20.943	6.93	11	99	3	22.92	9.4
20	21.789	20.337	7.14	12	105	3	22.29	9.6

GRAFO TOSCANA

caratteristiche generali: 287 vertici ; 773 spigoli ; 1 caratteristica

	MIDAS						AGRAF	
	performance							
p	t_i	t_f	Δ_m	pivot	migr	tree	t_f	Δ
3	60.858	60.839	0.03	0	3	2	75.9143	24.77
4	53.588	50.716	5.66	0	54	2	67.1956	32.50
5	47.433	44.535	6.51	0	54	2	58.7072	31.84
6	40.981	40.885	0.23	2	11	2	51.3704	25.66
7	37.7	36.872	2.24	2	32	2	48.3964	31.245
8	33.897	32.118	5.54	2	44	2	44.9969	40.01
9	31.392	29.617	5.99	2	44	2	41.6173	40.51
10	28.828	27.680	4.15	2	32	2	35.6539	28.79
11	26.866	25.717	4.47	2	32	2	32.5527	26.55
12	25.310	24.116	4.95	2	32	2	29.4923	22.01
13	23.921	22.783	4.99	2	32	2	27.0615	18.79
14	22.686	21.555	5.25	4	52	3	25.0418	16.19
15	21.451	20.442	4.93	3	52	3	23.1666	13.36
16	20.265	19.345	4.75	6	75	4	21.4699	11.01
17	19.097	17.331	10.19	13	139	4	19.8044	14.25
18	17.545	16.347	7.33	7	144	3	18.322	12.05
19	16.353	15.380	6.32	13	137	3	16.9157	9.99
20	16.175	15.225	6.24	5	92	3	15.6133	2.55