Appunti sui problemi di partizionamento ottimo di grafi in componenti connesse

Marco Liverani*

Ottobre 2005

1 Introduzione

In queste brevi note si propone una sintesi dei temi che possono essere sviluppati nel contesto di un progetto di ricerca nei settori scientifici dell'informatica e della teoria dei grafi, ossia in particolare nell'ambito dello studio di algoritmi di bassa complessità per la risoluzione esatta o approssimata di problemi di ottimizzazione combinatoria su specifiche classi di grafi.

Il contesto generale nel quale si inquadrano i temi oggetto di studio nell'ambito del progetto, è quello dei problemi di partizionamento o *clustering* di insiemi in cui possano essere presenti relazioni più o meno forti e vincolanti tra gli elementi che si intende raggruppare in componenti, con l'obiettivo di ottimizzare una determinata funzione della partizione stessa.

È noto che il problema SUBSET-SUM, in cui, dato un insieme S di naturali e una costante $k > 0$, si richiede di individuare un sottoinsieme $S' \subseteq S$ tale che $\sum_{s \in S'} s = k$, è un problema NP-completo (lo si può dimostrare costruendo un algoritmo di riduzione di complessità polinomiale in grado di trasformare ogni istanza di 3-CNF-SAT ad una istanza di SUBSET-SUM). Anche il problema SET-PARTITION, in cui, dato un insieme S di naturali, si richiede di costruire una bipartizione $\{S_1, S_2\}$ di S tale che $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$ e $\sum_{s \in S_1} s = \sum_{s \in S_2} s = k$, o la sua generalizzazione da 2 a p componenti sono problemi NP-completi.

Dunque, di per sé, il problema di suddividere in sottoinsiemi una determinata famiglia di oggetti rispettando una certa regola, costituisce un problema interessante dal punto di vista dello studio di algoritmi esatti o approssimanti di bassa complessità.

Il problema diventa ancora più articolato nel caso in cui vengono introdotte delle relazioni tra gli elementi della famiglia da partizionare; tali relazioni, che possono essere omogenee o caratterizzate da un diverso livello di legame tra gli oggetti della famiglia, ci suggeriscono di rappresentare il modello attraverso un grafo. In questo modo si possono formulare altri problemi in cui si richiede di tenere conto dei vincoli (gli spigoli del grafo) esistenti tra gli elementi dell'insieme (i vertici del grafo) da partizionare.

* liverani@mat.uniroma3.it – http://www.mat.uniroma3.it/users/liverani/

In generale si potrà richiedere di produrre delle partizioni in componenti connesse, tali da massimizzare o minimizzare il valore di una specifica funzione obiettivo. Ai vertici e agli spigoli del grafo potranno essere associati dei pesi che possono essere coinvolti nel calcolo della funzione obiettivo, contribuendo così a definire quei vincoli che devono essere rispettati nella costruzione delle componenti della partizione.

Le applicazioni sono numerose e vanno dallo studio dei sistemi elettorali, all'analisi dei gruppi. Sono numerosi i problemi in ambito ingegneristico che possono essere ricondotti ad un problema di partizionamento vincolato: si va dalla progettazione di circuiti elettronici VLSI, alla definizione ottimale di una rete di telecomunicazione o alla ottimizzazione del contrasto in una immagine grafica digitalizzata.

Nelle pagine seguenti sono descritti con meggiore dettaglio alcuni dei principali problemi di partizionamento, distinti sulla base della funzione obiettivo che si intende ottimizzare; vengono introdotte anche le principali tecniche algoritmiche utilizzate in questo contesto, ed infine sono descritte alcune delle applicazioni che possono trarre beneficio dalla soluzione di questo genere di problemi.

2 Alcune classi di problemi di partizione

È possibile raggruppare i problemi di partizione in base alla particolare famiglia di grafi su cui si applicano (grafi planari, alberi, cammini, ecc.), oppure in base a dei vincoli sulla topologia delle componenti della partizione (essere connesse, essere planari, più in generale essere isomorfe ad un grafo appartenente ad una classe particolare). Concentreremo la nostra attenzione sulla classificazione dei problemi di partizione in base alla funzione obiettivo da massimizzare (o minimizzare).

2.1 Massima omogeneità all'interno delle componenti

Dato un insieme $V = \{1, 2, ..., n\}$, si assegna un numero reale non negativo $d_{i,j} \geq 0$ ad ogni coppia $(i, j) \in V \times V$; questo numero rappresenta un indice di quanto gli elementi i e j sono simili tra loro, una distanza tra gli oggetti in esame. Si pone inoltre $d_{i,i} = 0 \ \forall \ i$ e $d_{i,j} = d_{j,i} \ \forall \ i, j$.

Fissata una costante $p \geq 2$, vogliamo individuare la partizione $\pi = \{C_1, ..., C_p\}$ di V che minimizzi la funzione obiettivo $f(\pi)$, definita sull'insieme $\Pi_p(G)$, costituito da tutte le partizioni di V in p componenti.

Per ottenere la massima omogeneità all'interno delle componenti, è possibile considerare le seguenti funzioni obiettivo da minimizzare:

Massimo diametro: definiamo come $d_{C_k} = \max_{i,j \in C_k} d_{i,j}$ il diametro della componente C_k, allora la funzione obiettivo può essere definita ponendo

$$f(\pi) = \max_{k=1,...,p} \left\{ \max_{i,j \in C_k} d_{i,j} \right\}$$

Somma delle distanze:

$$f(\pi) = \sum_{k=1}^{p} \left(\sum_{i,j \in C_k} d_{i,j} \right)$$

In entrambi i casi si può osservare facilmente che una partizione π^* che minimizzi la funzione obiettivo, è tale da rendere il più simili tra loro (meno distanti) gli elementi di ogni componente. Naturalmente la partizione dell'insieme V può avvenire in due modi completamente diversi a seconda che si utilizzi la prima o la seconda funzione obiettivo.

Ad esempio consideriamo il grafo completo K_4 i cui vertici sono gli elementi dell'insieme V; associamo agli spigoli del grafo le "distanze" $d_{i,j}$ nel seguente modo: $d_{1,2} = 1$, $d_{1,3} = 2$, $d_{1,4} = 4$, $d_{2,3} = 2$, $d_{2,4} = 3$, $d_{3,4} = 3$ (vedi Figura 1). Supponiamo di voler suddividere il grafo in due componenti ($p = 2$). La partizione ottimale per minimizzare la funzione obiettivo "massimo diametro" è $\pi^* = \{\{1,2,3\},\{4\}\}$, e risulta $f(\pi^*) = 2$. Per minimizzare la funzione obiettivo "somma delle distanze" la partizione ottimale è invece $\bar{\pi}^* = \{\{1,2\},\{3,4\}\}$, per cui risulta $f(\bar{\pi}^*) = 4$.

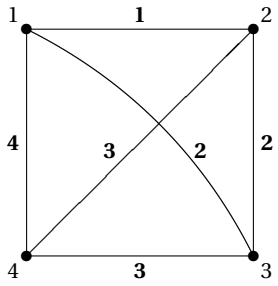

Figura 1: Grafo completo K_4 con pesi assegnati agli spigoli

Se G è un grafo completo e $p = 2$, con queste due funzioni obiettivo si ottengono problemi di complessità polinomiale; se $p > 2$ allora diventano NP-hard (Hansen e Jaumard, 1987). Se il grafo è un albero il problema con funzione obiettivo "massimo diametro" è NP-hard ([16]).

2.2 Massima distanza tra le componenti

Problema correlato al precedente, ma che in certi casi conduce a risultati sorprendentemente diversi, è quello basato sulle due seguenti funzioni obiettivo, mirate entrambe a separare gli elementi, in modo da massimizzare la distanza tra i cluster:

Minimo split:

$$f(\pi) = \min_{C_k \in \pi} \left\{ \min_{i \in C_k,\ j \in V - C_k} d_{i,j} \right\}$$

Somma delle distanze tra cluster:

$$f(\pi) = \sum_{k=1}^{p} \left(\sum_{i \in C_k,\ j \in V - C_k} d_{i,j} \right)$$

Anche in questo caso, come nel precedente, possiamo rappresentare come vertici di un grafo completo $G = (V, E)$ gli elementi da suddividere in p componenti. L'obiettivo è quello di trovare una partizione π^* che massimizzi le due funzioni; chiaramente, come già abbiamo osservato nel problema precedente, pur avendo entrambe come finalità quella di massimizzare la distanza tra le componenti della partizione, possono in taluni casi condurre a partizioni differenti.

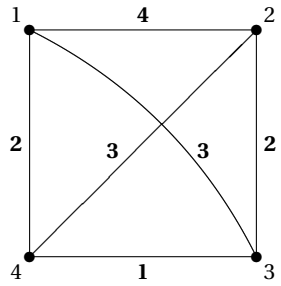

Figura 2: Grafo completo K_4 con altri pesi assegnati agli spigoli

Ad esempio consideriamo il grafo completo K_4 rappresentato in Figura 2; in questo caso abbiamo ridefinito le distanze tra i vertici del grafo rispetto a quelle riportate in Figura 1.

La funzione obiettivo "minimo split" ci porta ad individuare la partizione $\pi^* = \{\{1,2\},\{3,4\}\}$ che fornisce un valore $f(\pi^*) = 3$, mentre con la funzione obiettivo "somma delle distanze tra cluster" si ottiene la partizione ottimale $\bar{\pi}^* = \{\{1,4\},\{2,3\}\}$, con valore $f(\bar{\pi}^*) = 22$.

Il problema di partizione connessa con funzione obiettivo "minimo split" è NP-hard per grafi a griglia (e quindi per grafi planari e bipartiti) (Hansen et al., 1993, Garey e Johnson, 1977). Il problema è invece polinomiale se G è un grafo completo (Delattre e Hansen, 1980), se G è un albero (Maravalle e Simeone, 1992, Hansen et al., 1993) e se G è una *scala*, cioè un grafo a griglia con due righe ed un numero arbitrario di colonne (Lari, 1994).

Il problema di partizione con funzione obiettivo "somma delle distanze tra cluster" è NP-hard per G qualsiasi e $p \geq 2$ (Welch, 1982).

2.3 Equipartizione

In questo tipo di problemi è assegnato un peso non negativo $w_i \geq 0$ ad ogni elemento dell'insieme. L'obiettivo è quello di costruire una partizione dell'insieme in p componenti $C_1, \ldots, C_p$ in modo da rendere il più possibile uniforme il peso $W(C_k) = \sum_{i \in C_k} w_i$ delle componenti. In altri termini, se gli elementi dell'insieme sono i vertici di un grafo $G = (V, E)$, ci si propone di trovare una partizione π di V in modo che i pesi delle componenti siano simili tra loro (l'obiettivo ideale, ma il più delle volte non raggiungibile, sarebbe quello di avere $W(C_k) = cost.\ \forall\ k = 1, \ldots, p$). Si richiede inoltre che il sottografo indotto da ognuna delle p componenti della partizione sia connesso. Osserviamo subito che il problema di partizionare un grafo completo in due componenti di uguale peso, noto anche come SUBSET-SUM, è NP-completo (Karp, 1972).

Indichiamo con $\mu = \frac{1}{p} \sum_{i=1}^{n} w_i$ il peso medio delle componenti della partizione. Alcune delle funzioni obiettivo da minimizzare che ci permettono di effettuare una *equipartizione* del grafo G sono le seguenti:

Norma L_1:

$$f(\pi) = \sum_{k=1}^{p} \left| W(C_k) - \mu \right|$$

Norma L_2:

$$f(\pi) = \sum_{k=1}^{p} \left(W(C_k) - \mu \right)^2$$

Norma L_∞:

$$f(\pi) = \max_{k=1,\ldots,p} \left| W(C_k) - \mu \right|$$

Chiaramente le funzioni obiettivo possono essere le più varie. Ad esempio possiamo considerare la differenza tra il massimo (o il minimo) peso di una componente e μ:

$$f(\pi) = \max_{k=1,\ldots,p} W(C_k) - \mu \geq 0 \quad (\text{o anche } f(\pi) = \max_{k=1,\ldots,p} W(C_k))$$

$$f(\pi) = \mu - \min_{k=1,\ldots,p} W(C_k) \geq 0 \quad (\text{o anche } f(\pi) = \min_{k=1,\ldots,p} W(C_k))$$

In generale questo tipo di problemi sono chiamati *Most Uniform Partitioning* (MUP).

Se f è la norma L_1 il problema è NP-hard per gli alberi e polinomiale per classi speciali di alberi come stelle, cammini e bruchi (Aparo e Simeone, 1975, De Simone et al., 1990). In Becker et al., 1982, Becker et al., 1983, Perl e Schach, 1981, vengono forniti alcuni algoritmi di shifting che risolvono in tempo polinomiale i problemi di equipartizione di alberi, con funzioni obiettivo

Min-max (da minimizzare):

$$f(\pi) = \max_{k=1,\ldots,p} W(C_k)$$

Max-min (da massimizzare):

$$f(\pi) = \min_{k=1,\ldots,p} W(C_k).$$

I problemi MUP sono NP-hard per grafi qualsiasi (Garey e Johnson, 1979), ma esiste un algoritmo polinomiale per risolvere tale problema su cammini (Lucertini et al., 1993). Lari, nella sua tesi di dottorato ([13]), dimostra che tutti i problemi di equipartizione sono NP-hard per grafi a griglia (e quindi anche per grafi planari e bipartiti) e fornisce un algoritmo polinomiale per il problema di equipartizione con funzione obiettivo da massimizzare $f(\pi) = \min_{C \in \pi} W(C)$ su una scala.

In [14] abbiamo formulato un nuovo algoritmo per l'equipartizione di un cammino con funzione obiettivo la norma L_∞ che migliora in modo significativo la complessità del problema, essendo caratterizzato da un'efficienza superiore a quella di altri algoritmi analoghi presenti in letteratura.

Anche in questo caso un esempio elementare potrà aiutarci a chiarire le differenze tra le funzioni obiettivo. Pur essendo tutte finalizzate ad un'equipartizione del grafo, ammettono partizioni ottimali differenti. Da notare che si parla sempre di soluzioni ottimali e non di un'unica soluzione ottima, perché, come è evidente, possono essere più d'una le partizioni di un grafo che portano a minimizzare (o massimizzare) la funzione obiettivo assegnata. Consideriamo un cammino costituito da sette vertici ($n = 7$) a cui assegnamo dei pesi $w_i \geq 0$ come in Figura 3 (come al solito i pesi sono indicati in grassetto). Vogliamo partizionare in tre componenti connesse il cammino ($p = 3$), per minimizzare la funzione obiettivo min-max e la norma L_∞ e per massimizzare la funzione max-min.

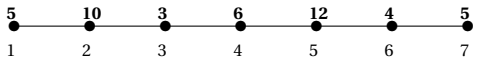

Figura 3: Cammino pesato sui vertici

Si riconosce facilmente che la partizione $\pi^* = \{\{1,2,3\},\{4,5\},\{6,7\}\}$ è ottimale per tutte e tre le funzioni obiettivo, mentre la partizione $\bar{\pi}^* = \{\{1,2\},\{3,4,5\},\{6,7\}\}$ è ottimale per i problemi con funzione obiettivo max-min e la norma L_∞, ma non per quello basato sulla funzione min-max.

2.4 Taglio della partizione

Dato un grafo $G = (V, E)$, si associa ad ogni spigolo un peso non negativo; data una partizione π di G, per taglio della partizione si intende la somma dei pesi degli spigoli i cui estremi appartengono a componenti diverse. In certe applicazioni si richiede che il taglio sia massimo, in altre invece che il taglio sia minimo. È interessante osservare che se i pesi associati agli spigoli sono tutti unitari, allora il taglio fornisce una misura della maggiore o minore connessione della partizione: se il taglio della partizione π_1 è minore di quello della partizione π_2, allora le componenti della prima risulteranno maggiormente connesse di quelle della seconda.

Distinguiamo tre sotto classi di problemi relativi al taglio della partizione:

Partizione non pesata: dato un grafo con vertici non pesati (o di peso costante), si vuole determinare una partizione in due componenti di uguale cardinalità in modo tale da minimizzare il numero di spigoli del taglio (o il peso del taglio). Questo problema è NP-hard (Garey et al., 1976).

Taglio minimo: dato un grafo pesato sugli spigoli, l'obiettivo è quello di determinare una bipartizione dei vertici che minimizzi il taglio (non è richiesto il vincolo della pari cardinalità delle due componenti). Questo problema è risolvibile in tempo polinomiale tramite l'applicazione di algoritmi per il massimo flusso e rimane polinomiale anche per ipergrafi (Lawker, 1983). Diventa in genere di difficile soluzione se si aggiungono vincoli sulla topologia delle componenti.

Taglio massimo: dato un grafo pesato sugli spigoli, lo si vuole ripartire in due componenti in modo che il taglio risulti essere massimo. Questo problema è NP-hard (Karp, 1972). Rimane NP-hard anche nel caso di grafi non pesati (Garey et al., 1976) e se il grado massimo dei vertici è uguale a 3 (Yannakakis, 1978). Può essere risolto in tempo polinomiale su grafi pesati se il grafo è contraibile a K_5 (Orlova e Dorfman, 1972, Hadlock, 1975, Barahona, 1982) e se non contiene cicli dispari "lunghi" (questa classe contiene i grafi bipartiti) (Grötshel e Nemhauser, 1984) e su edge-graphs non pesati (Arbib, 1987).

3 Algoritmi

Per quanto riguarda gli aspetti algoritmici nella soluzione di problemi di partizione, gli approcci utilizzati più frequentemente sono sostanzialmente tre:

- formulazione dei problemi in termini di Programmazione Lineare a Numeri Interi (PLI) ed uso di programmi applicativi standard per la risoluzione in modo esatto o approssimato;
- sviluppo di algoritmi, esatti o approssimati, per specifiche classi di problemi;
- uso di euristiche di ampia applicabilità, di tipo deterministico o probabilistico.

Naturalmente nessuno di questi approcci è ottimale sotto ogni aspetto. Infatti i pacchetti applicativi standard sono tanto meno efficienti ed in grado di gestire problemi di dimensioni elevate, quanto più generale è la classe di problemi su cui sono utilizzabili. D'altro canto lo sviluppo di algoritmi specifici per un singolo problema o per una ristretta classe di problemi, porta con sé il difetto opposto: ad una grande efficienza operativa si contrappone una scarsa riutilizzabilità della procedura su problemi differenti. Infine, la assoluta generalità degli algoritmi euristici di tipo deterministico o probabilistico, può rivelarsi fonte di inefficienza quando non vengano sfruttate a fondo le caratteristiche specifiche del problema in esame.

Di questi tre approcci, comunque, l'ultimo rappresenta spesso l'unica alternativa praticabile. Può essere utile quindi, riportare alcune delle tecniche più diffuse nell'ambito degli algoritmi per il partizionamento (vedi [3]).

Ottimizzazione su reti: la ricerca di una partizione utilizzando questo metodo fa ricorso, ad esempio, ad algoritmi di massimo flusso–minimo taglio o ad algoritmi di massimo taglio (*max-cut*).

Migrazione di gruppi: questa tecnica consiste nel generare una partizione iniziale e nel tentare un suo miglioramento attraverso lo scambio di un piccolo gruppo di vertici (al limite un solo vertice) tra due o più componenti della partizione. Questo passo viene ripetuto fino a quando non è più possibile un miglioramento. Questa tecnica di "ricerca locale" può prevedere l'introduzione di passi di tipo probabilistico tanto nella determinazione della soluzione iniziale quanto nella fase di miglioramento e nel criterio di arresto.

Simulated annealing: differisce dalla tecnica di *migrazione di gruppi* solo per quanto riguarda il criterio di accettazione della nuova partizione generata: vengono infatti accettate anche partizioni che peggiorano il valore della funzione obiettivo, nella speranza che questo peggioramento permetta di superare eventuali minimi o massimi locali. Il criterio di interruzione della ricerca è basato pertanto su condizioni più forti e restrittive che non la sola valutazione del valore della funzione obiettivo.

Semi: questa tecnica viene usata nel caso di partizioni in un numero p di componenti prefissato. Definita sull'insieme dei vertici del grafo una metrica, si individuano i p vertici che distano di più gli uni dagli altri. Questi vertici sono detti *semi*. Compatibilmente con i vincoli sulle dimensioni di ogni componente della partizione, ogni altro vertice del grafo viene assegnato al seme più vicino, o al secondo più vicino, ecc.

Greedy: si tratta di organizzare il problema in una sequenza di sottoproblemi più semplici, in ognuno dei quali viene assegnato in modo ottimo a una delle componenti un insieme di vertici (al limite uno solo). Le soluzioni ottime calcolate non vengono modificate nei problemi successivi, il numero di sottoproblemi risolti è lineare nel numero dei vertici del grafo e la complessità del procedimento dipende dunque dalla complessità di risoluzione dei singoli sottoproblemi.

Programmazione dinamica: come il metodo *divide et impera* ([7]) risolve il problema combinando la soluzione di sottoproblemi. Gli algoritmi di tipo *divide et impera* suddividono il problema originario in sottoproblemi indipendenti, li risolvono ricorsivamente, ed infine combinano insieme le loro soluzioni per ottenere la soluzione del problema originario. Al contrario, la *programmazione dinamica* si applica quando i sottoproblemi non sono indipendenti, cioè quando i sottoproblemi condividono degli ulteriori sottoproblemi. In questo caso un algoritmo di tipo *divide et impera* lavorerebbe più del necessario, risolvendo ripetutamente gli stessi sottoproblemi. Un algoritmo che sfrutta la *programmazione dinamica* invece, risolve ogni sottoproblema una sola volta memorizzando la soluzione per poterla riutilizzare (senza doverla ricalcolare) ogni volta che il sottoproblema viene incontrato.

Numerosi algoritmi riportati in letteratura, riguardanti la soluzione di problemi di partizionamento ottimo in componenti connesse di alberi e cammini, utilizzano la tecnica dello *shifting*, applicabile sia alla strategia della Migrazione di gruppi che a quella Greedy o del Simulated annealing. La partizione viene costruita "tagliando" alcuni spigoli in modo da disconnettere le p componenti. Sugli alberi e sui cammini (un caso particolare di albero) una p-partizione in componenti connesse si ottiene tagliando esattamente $p-1$ spigoli; lo stesso non si può dire su un grafo generico. Dunque, partendo da una assegnazione iniziale arbitraria dei tagli a $p-1$ spigoli distinti, si spostano i tagli da uno spigolo all'altro, avendo l'accortezza di non creare mai componenti vuote (è il caso in cui due o più tagli sono assegnati allo stesso spigolo). Se l'operazione di *shift* degli spigoli avviene sempre nella stessa direzione (ad esempio dalla radice alle foglie), allora si ha anche la certezza che l'algoritmo avrà termine dopo un numero finito di passi, pari al più a $|E(G)| \cdot (p-1)$. Tuttavia, per ovviare a scelte che pur essendo localmente ottime si rivelano errate nelle iterazioni successive dei passi dell'algoritmo, alcune strategie risolutive per il partizionamento di alberi (si vedano ad esempio [4] e [18]) ammettono anche la possibilità di compiere "*shift* laterali", ossia spostamenti di tagli da uno spigolo ad un altro non incidente.

4 Applicazioni

Come abbiamo accennato nell'introduzione, le applicazioni dei problemi di partizionamento ottimo possono essere le più varie: da problemi di classificazione a problemi di ottimizzazione delle comunicazioni, problemi di modellizzazione di un campione di indagine statistica, e molti altri ancora. In estrema sintesi vale la pena citare alcuni esempi più significativi, anche per dare una misura dell'ampiezza della gamma di campi applicativi di questo tipo di problematiche.

In biologia gli elementi dell'insieme da studiare possono essere forme di vita come piante, animali, microorganismi e l'obiettivo dello studio può consistere nella definizione di un'intera tassonomia o nella delimitazione delle sottospecie di una specie. Ad esempio, Rogers e Tanimoto, 1960, hanno proposto un programma per la classificazione delle piante; Sneath, 1957, ha analizzato il problema di individuare una tassonomia per gruppi di batteri; Sokal e Sneath, 1963, hanno utilizzato diverse tecniche di partizione per ottenere "dendrogrammi" per dati biologici.

In geologia sono frequenti problemi di classificazione delle rocce e del suolo, delle città e delle regioni, dei bacini idrici ed in generale legati al deflusso delle acque. Wolf et al., 1972, hanno usato una procedura di clustering per lo studio del movimento delle nuvole.

Alcuni problemi di tipo "ingegneristico" fanno spesso uso di tecniche di clustering, come ad esempio nel riconoscimento delle forme, nella strutturazione della base di conoscenza nei sistemi esperti ed in altre applicazioni dell'intelligenza artificiale, nel progetto di circuiti VLSI, nella progettazione di reti di calcolatori o, più in generale, di telecominicazione; nel trattamento delle immagini digitali si ha un interessante esempio di utilizzo di tecniche di equipartizione. Akers, 1982, Korte et al., 1988, Arbib et al., 1989, Antenucci et al., 1990, hanno individuato problemi di parti-

zione nel progetto di circuiti; per il trattamento delle immagini si veda ad esempio Haralick e Dinstein, 1975, Narendra e Goldberg, 1980, Warthon, 1983, Lucertini et al., 1983.

In medicina gli oggetti dell'analisi possono essere pazienti, sintomi o test di laboratorio. L'obietivo principale è la scoperta di mezzi più efficaci ed economici per la diagnosi. Gose et al., 1972, per esempio, hanno utilizzato un algoritmo di partizione in una procedura per identificare il tumore al seno mediante radiografia.

Nelle scienze comportamentali e sociali trovano frequente applicazione una grande varietà di problemi di partizione. Ad esempio nello studio di metodi di insegnamento, schemi di comportamento, fattori di rendimento umano, famiglie, organizzazioni, ecc. In particolare: Kaskey et al., 1962, hanno utilizzato tecniche di clustering per la diagnosi dei pazienti di un istituto psichiatrico; Haralick e Haralick, 1971, hanno analizzato il comportamento di bambini sordi mediante una partizione di variabili.

Le scienze politiche, economiche e delle decisioni in cui i problemi di partizione riguardano, per esempio, la segmentazione della clientela nel marketing, la sequenzializzazione e l'assegnazione di compiti a macchine in una catena di produzione, scelta di investimenti, dislocazione di impianti. In particolare citiamo: Garfinkel e Nemhauser, 1969, per un'applicazione riguardante la partizione di centri di popolazione in distretti politici; Rizzi, 1981, che riporta un'applicazione alle proiezioni dei risultati elettorali; Freeman e Jucker, 1967, per i problemi di bilanciamento di catene di produzione.

Descriviamo, citando qualche aspetto particolare, alcuni di questi problemi.

4.1 Elaborazione delle immagini digitali

Supponiamo di aver acquisito mediante una strumentazione elettronica un'immagine di qualità fotografica. In genere questo tipo di immagini sono rappresentate nella memoria della macchina tramite una matrice di punti, tanto più grande quanto maggiore è la risoluzione grafica dell'apparato digitale di acquisizione a nostra disposizione. Ogni punto della matrice può essere rappresentato mediante un numero intero c, $0 \leq c \leq max$, in cui il valore 0 rappresenta un punto completamente nero, il valore max un punto bianco[1], mentre i valori intermedi rappresentano quantità crescenti di luminosità in una "scala di grigio" che va, appunto, da nero al bianco. Una rappresentazione analoga può essere effettuata anche nel caso di immagini a colori, in cui il livello cromatico di ogni punto dell'immagine può essere scomposto in tre quantità distinte, associate ai tre colori di base rosso, verde e blu, dalla cui combinazione si può ricavare ogni altro colore in diverse tonalità.

Per problemi legati all'occupazione di memoria di questo tipo di rappresentazione, o alle caratteristiche di alcuni sistemi di riproduzione o stampa delle immagini stesse, è spesso utile ridurre la scala di livelli di grigio, dalle originarie max gradazioni di grigio (o di tonalità distinte di colore) ad un numero molto più piccolo, ad esempio $p = max \cdot 10^{-3}$. È stato verificato per via sperimentale che il migliore

[1] Il valore numerico di max dipende dalla sensibilità cromatica dello strumento di acquisizione utilizzato; con i dispositivi attuali $max \simeq 10^6$.

contrasto tra i p livelli di grigio si ha rendendo omogeneo il numero di *pixel* (punti dell'immagine digitale) per ogni livello di luminosità.

Il problema si presta bene ad essere modellizzato mediante un cammino di max vertici da partizionare in p componenti connesse (sottocammini) secondo un algoritmo di equipartizione. Rappresentiamo il livello di grigio i nella scala cromatica originale mediante l'i-esimo vertice del cammino ed associamo a tale vertice un peso $w_i \geq 0$ pari al numero di pixel aventi tale livello di luminosità nell'immagine grafica originale. Effettuando una partizione del cammino assumendo come funzione obiettivo una delle funzioni viste nel caso del problema della equipartizione (norma L_∞, norma L_1, norma L_2, funzione max-min, ecc.) si ottiene una suddivisione in p componenti connesse; ad ognuna di tali componenti assegneremo progressivamente il nuovo livello di grigio, che dovrà essere applicato a tutti i pixel di ogni componente.

Per maggiore chiarezza facciamo un esempio con un numero di dati molto piccolo. Supponiamo di avere un'immagine di 100×100 pixel in 8 livelli di grigio e supponiamo di voler ridurre il numero di sfumature da 8 a 3. Rappresentiamo la situazione mediante un cammino ed associamo ad ogni vertice il numero di pixel che nell'immagine originale assumono tale intensità (in figura 4 sono rappresentati in grassetto il numero di pixel per ogni livello di grigio: 1104 pixel assumono la tonalità numero 1, 1832 la numero 2 e così via...).

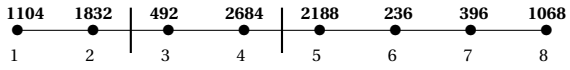

Figura 4: Equipartizione del cammino in 3 componenti con funzione obiettivo norma L_∞

La partizione ottimale, considerando come funzione obiettivo la norma L_∞, è data da $\pi^* = \{\{1,2\},\{3,4\},\{5,6,7,8\}\}$, quindi ai pixel che nell'immagine originaria avevano le tonalità 1 e 2 verrà assegnata la nuova tonalità c_1 (i vertici 1 e 2 del cammino ricadono nella prima componente della partizione), ai pixel che prima avevano tonalità 3 e 4 sarà assegnata la nuova tonalità c_2, mentre i rimanenti pixel assumeranno la tonalità c_3.

Un altro tipo di problema, collegato sempre all'elaborazione delle immagini digitali, potrebbe essere quello di riconoscere delle aree omogenee all'interno dell'immagine, utile, ad esempio, nell'elaborazione di immagini provenienti da un satellite, in cui ad aree caratterizzate da un livello cromatico uniforme corrispondono zone fisicamente omogenee (stesso tipo di territorio oppure stessa temperatura, ecc.). Questo problema può essere formalizzato come un problema di partizionamento di un grafo a griglia.

4.2 Progettazione di circuiti VLSI

Una fase importante nel progetto di circuiti VLSI (Very Large Scale Integration) è la suddivisione dei componenti in *pagine* di capacità fissata. In generale viene richiesto che il numero di collegamenti da una pagina ad un'altra sia minimo. Il circuito

può essere allora rappresentato mediante un ipergrafo $H = (V, E)$ in cui i vertici sono in corrispondenza biunivoca con i componenti; ad ogni vertice si assegna poi un peso $w_i \geq 0$ pari alla dimensione del componente rappresentato dal vertice stesso; gli spigoli del grafo rappresentano i collegamenti tra i componenti.

Con questa modellizzazione il problema di suddividere l'intero circuito in sottocircuiti (paginazione) può essere espresso come un problema di partizione di un ipergrafo pesato in cui il peso di ciascuna componente non può superare una soglia c data (la capacità delle pagine), ed in cui si vuole minimizzare "la connessione" tra le componenti.

Questo problema di partizione è NP-hard, tuttavia spesso il circuito, e quindi anche il grafo che lo rappresenta, ha delle caratteristiche che rendono la soluzione più semplice. Per esempio se ogni componente ha non più di quattro connessioni con altri componenti, allora si può procedere all'implementazione direttamente su una struttura a griglia.

4.3 Progettazione di una rete di calcolatori

Una topologia tipica per le reti di calcolatori vede raggruppate in cluster un certo numero di macchine dislocate in uno stesso ambiente di lavoro o in stanze fisicamente vicine tra loro. Ogni cluster è a sua volta collegato, mediante una macchina appositamente dedicata a questo scopo, chiamata *router*, ad un tratto di collegamento più lungo che permette di comunicare con un cluster adiacente. La rete è quindi strutturata in gruppi di pochi calcolatori collegati ad una linea di comunicazione principale che connette tra loro i gruppi. Questa linea può anche avere delle biforcazioni. I vincoli nella progettazione di una rete di questo tipo sono dati dal massimo numero di calcolatori per ogni cluster e dalla vicinanza dei calcolatori di uno stesso cluster. In generale è poi desiderabile che due calcolatori che debbano comunicare intensamente fra loro appartengano ad uno stesso cluster, in modo da minimizzare l'uso della banda passante sulla linea di comunicazione principale.

La rete descritta si presta ad essere rappresentata facilmente mediante un albero $T = (V, E)$, i cui vertici sono i singoli calcolatori e gli spigoli del grafo sono i collegamenti tra le macchine. Ad ogni spigolo viene associata una distanza $d_{i,j} \geq 0$ pari alla distanza tra le due macchine rappresentate dai vertici i e j.

Il problema è allora quello di partizionare l'albero T in p sottoalberi connessi (i cluster di calcolatori) in modo da minimizzare il peso delle componenti (ad esempio con funzioni obiettivo "massimo diametro" o "somma delle distanze" da minimizzare).

4.4 Proiezione dei risultati elettorali

Si fa ricorso all'analisi dei gruppi per la previsione dei risultati delle elezioni. In queste occasioni infatti si devono anticipare i risultati finali in base ad una rilevazione per campione dei primi spogli, da effettuare in tempi brevissimi. In questi casi non è possibile utilizzare un campione rappresentativo dell'universo delle sezioni elettorali perché non è possibile ottenere tutti i dati del campione in tempo utile. Per la determinazione del campione si ricorre quindi alla stratificazione della popolazione

ed alla ponderazione dei vari strati. Ma una semplice stratificazione in base a caratteri geografici e demografici può portare a distorsioni non trascurabili, per cui si fa ricorso all'analisi dei gruppi come metodo di stratificazione. Si suddivide il Paese in un certo numero di areole e, basandosi sui risultati delle elezioni precedenti, si applica un algoritmo di analisi dei gruppi. Si individuano in questo modo gruppi politicamente omogenei.

Riferimenti bibliografici

[1] Enzo L. Aparo, Bruno Simeone, *Un algoritmo di equipartizione e il suo impiego in un problema di contrasto ottico*, Estratto da Ricerca Operativa, N. 6, 1973, pp. 1-12.

[2] Claudio Arbib, *A polynomial characterization of some graph partitioning problems*, Information Processing Letters, 26, 1987/88, pp. 223-230.

[3] Claudio Arbib, Mario Lucertini, Sara Nicoloso, *Problemi di partizione nel Layout ottimo di circuiti integrati*, Ricerca Operativa, 49, 1989, pp. 3-54.

[4] Ronald I. Becker, Stephen R. Schach, Yehoshua Perl, *A shifting algorithm for min-max tree partitioning*, Journal of the Association for Computing Machinery, Vol. 29, No. 1, January 1982, pp. 58-67.

[5] Michal Benelli, Refael Hassin, *Optimal separable partitioning in the plane*, Discrete Applied Mathematics, 59, 1995, pp. 215-224.

[6] Francesco Conti, Federico Malucelli, Sara Nicoloso, Bruno Simeone, *On a 2-dimensional equipartition problem*, preprint.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduzione agli algoritmi e strutture dati*, seconda edizione, McGraw-Hill, 2005.

[8] Caterina De Simone, Mario Lucertini, Stefano Pallottino, Bruno Simeone, *Fair Disconnections of Spiders, Worms and Caterpillars*, Networks, Vol. 20, 1990, pp. 323-344.

[9] Michael R. Garey, David S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[10] Martin Grötschel, George L. Nemhauser, *A polynomial algorithm for the max-cut problem on graphs without long odd cycles*, Mathematical Programming, 29, 1984, pp. 28-40.

[11] Scott W. Hadley, *Approximation techniques for hypergraph partitioning problems*, Discrete Applied Mathematics, 59, 1995, pp. 115-127.

[12] S. Kundu, J. Misra, *A linear tree partitioning algorithm*, SIAM Journal of Computing, 6, 1977, pp. 151-154.

[13] Isabella Lari, *Partizioni ottime di grafi a griglia*, Tesi di Dottorato in Ricerca Operativa, 1994, Università di Roma "La Sapienza", Dipartimento di Statistica, Probabilità e Statistiche Applicate.

[14] Marco Liverani, Aurora Morgana, Bruno Simeone, Gianni Storchi, *Path equipartition in the Chebyshev norm*, European Journal of Operational Research, 123, 2000, pp. 428-436.

[15] Mario Lucertini, Yehoshua Perl, Bruno Simeone, *Most uniform path partitioning and its use in image processing*, Discrete Applied Mathematics, 42, 1993, pp. 227-256.

[16] Maurizio Maravalle, Rossella Naldini, Bruno Simeone, *Clustering on Trees*, preprint (successivo al 1985).

[17] Takao Nishizeki, Svatopluk Poljak, *k-Connectivity and decomposition of graphs into forests*, Discrete Applied Mathematics, 55, 1994, pp. 295-301.

[18] Yehoshua Perl, Stephen R. Shach, *Max-min tree partitioning*, Journal of the Association for Computing Machinery, Vol. 28, No. 1, January 1981, pp. 5-15.

[19] Yehoshua Perl, Uzi Vishkin, *Efficient implementation of a shifting algorithm*, Discrete Applied Mathematics, 12, 1985, pp. 71-80.

[20] Erich Prisner, *Clique covering and clique partition in generalizations of line graphs*, Discrete Applied Mathematics, 56 1995, pp. 93-98.

[21] M. R. Rao, *Cluster Analysis and Mathematical Programming*, Journal of the American Statistical Association, Vol. 66, N. 335, 1971, pp. 622-626.

[22] Sartaj Sahni, Teofilo Gonzales, *P-Complete Approximation Problems*, Journal of the Association for Computing Machinery, Vol. 23, No. 3, July 1976, pp. 555-565.

[23] Stephen W. Wharton, *A generalized histogram clustering scheme for multidimensional image data*, Pattern Recognition, Vol. 16, N. 2, 1983, pp. 193-199.

[24] Jing-Ho Yan, Gerard J. Chang, *The path-partition problem in block graphs*, Information Processing Letters, 52, 1994, pp. 317-322.