Università degli Studi Roma Tre
Facoltà di Scienze M.F.N.

Partizionamento ottimo di grafi ed applicazioni

Sintesi della tesi di Laurea in Matematica
di Massimiliano Lorenzoni

Relatore: Prof. Marco Liverani

a.a. 1999/2000 – Novembre 2000

L'obiettivo principale della tesi consiste nello studio e l'implementazione di alcuni algoritmi per partizionare in modo ottimale un cammino pesato sui nodi in modo da ridurre il numero di sfumature di grigio in un'immagine digitale senza, per questo, ridurne il contrasto e la riconoscibilità. Cercare una partizione del cammino quanto più equa possibile, equivale a distribuire equamente i colori che costituiscono l'immagine digitale. Per far questo abbiamo trasformato un'immagine di dimensione $n \times m$ in una matrice i cui elementi corrispondono al colore che un corrispondente pixel ha nell'immagine e successivamente abbiamo implementato i nostri algoritmi leggendo da file la matrice e convertendola in un cammino il cui nodo i ha peso uguale al numero di occorenze dei corrispondenti valori della matrice e quindi tale che i nodi sono posti in ordine crescente. Partizionare il cammino significa raggruppare un certo numero di nodi per ogni componente in modo che il peso di ognuna di esse sia quanto più possibile uguale al peso delle altre.

Abbiamo deciso di impostare la tesi partendo dal caso più generale dei reticoli o grafi a griglia dimostrando come in questi casi, e in generale per quasi tutti i grafi bi-dimensionali, non sia possibile trovare la soluzione al problema in tempo polinomiale. Abbiamo definito infatti, i così detti problemi *NP-completi*, una classe di problemi che contengono esempi troppo importanti perché si possano tralasciare, quali ad esempio i problemi di equipartizione di grafi completi o di ricerca della p-partizione (ovvero alla ricerca di una partizione in esattamente p componenti connesse) di un albero in modo da minimizzare alcune funzioni obiettivo.

Considerando il fatto che la scopo del lavoro e le nostre applicazioni non sono direttamente connesse alla partizione di grafi bi-dimensionali, abbiamo deciso di parlare solo brevemente del problema del partizionamento di tali grafi, cercando di sottolineare in modo particolare sia la loro NP-complessità, sia la possibilità di ottenere alcune risposte sulla loro reale complessità sfruttando alcuni particolari algoritmi che restituiscono soluzioni *quasi* ottime e che vanno sotto il nome di *algoritmi approssimanti*. Questi sono algoritmi il cui costo C della soluzione prodotta si discosta per un certo fattore dal costo C^* della soluzione ottima. Quanto più questo fattore è piccolo, tanto migliore risulta l'algoritmo appossimante. Abbiamo introdotto, allora, il problema del partizionamento di grafi a griglia prendendo spunto da un articolo di Conti, Malucelli, Nicoloso e Simeone [4] per dimostrare sia la NP-complessità di questo problema tramite alcune riduzioni ad altri problemi che si conoscono essere NP-completi, come il problema del SET COVERING [5], sia la possibilità di ridurre il problema ad altri simili, ma meno complicati, sfruttando delle euristiche. Alcune di queste sono molto semplici sia da definire che da implementare (anche se noi abbiamo preferito non farlo per non dilungarci troppo su un argomento non direttamente correlato alle applicazioni elaborate) visto che sfruttano alcune imposizioni paricolari sui tagli da effettuare (fissando ad esempio i tagli orizzontali o verticali). Altre sono un po' più complicate anche perché sfruttano, all'interno dell'algoritmo stesso, anche le euristiche di cui abbiamo parlato sopra rendendo lo

studio e l'implementazione più complicati. Abbiamo considerato, poi, il caso del p-partizionamento di grafi di tipo Max-Min studiato da Becker, Lari, Lucertini e Simeone [2], un particolare tipo di partizionamento che abbiamo studiato anche nel capitolo sugli alberi, e che divide il grafo a griglia in p componenti connesse con lo scopo di massimizzare il più possibile la componente di peso minore. Per la prima parte di questa sezione il risultato più interessante, a parte quelli per dimostrare la correttezza degli algoritmi, è il teorema:

Teorema 1. *Dati un grafo G e un intero K, sia P^2 il problema dell'esistenza di una bi-partizione di G che soddisfi la condizione $w(C_i) \geqslant K$, (i=1,2). Il problema P^2 per grafi di dimensione $M \times 3$ è NP-completo.*

Per concludere il capitolo dedicato ai grafi bi-dimensionali, abbiamo considerato due algoritmi approssimanti denominati SHIFT e SHIFTA che approssimano la soluzione ottima del problema e che hanno una complessità di $O(n^3p)$ e $O(n^3p^2)$ rispettivamente.

Una delle strategie adottate nell'implementazione degli algorimi è stata quella di considerare un algoritmo sviluppato per grafi più complicati dei cammini, quali sono gli alberi, ed applicarlo ai cammini per vedere se migliorava i risultati ottenuti dagli algoritmi studiati per essere implementati esclusivamente sui cammini stessi. Abbiamo studiato, allora, in modo più approfondito questi particolari grafi connessi, aciclici e non orientati (gli alberi) e in particolare due algoritmi denominati MIN-MAX e MAX-MIN che risolvono in tempo polinomiale il problema del partizionamento di un albero. Il primo, dovuto a Becker, Shach e Perl [3], sfrutta due tipi di spostamenti dei tagli (che all'inizio vengono posti tutti sull'arco incidente la radice dell'albero): il primo verso il basso DOWN SHIFT e il secondo lateralmente *side shift*. L'algoritmo termina non appena la componente che contiene la radice diventa la componente più pesante tra tutte quelle dell'albero e trova una soluzione con complessità computazionale dell'ordine $O(k^3h(T) + kn)$ (in cui con $h(T)$ abbiamo indicato l'altezza dell'albero, con k il numero dei tagli e con n i nodi che lo compongono). Il secondo, di Perl e Shach [10], sfrutta solo gli spostamenti dei tagli verso il basso e si blocca quando il peso della componente generata dallo spostamento di un taglio diventa minore del peso della componente più leggera del passo precedente e trova una soluzione con complessità computazionale dell'ordine $O(k^2h(T) + kn)$. Abbiamo deciso, anche perché ha una complessità inferiore, che fosse quest'ultimo l'algoritmo da implementare con il risultato (visibile dalle applicazioni nelle ultime pagine della tesi) che l'algoritmo non migliorando la qualità delle immagini (almeno per quanto riguarda quelle rstituite dall'algoritmo di Liverani, Morgna, Simeone e Storchi [8]) non si può ritenere preferibile rispetto agli altri.

L'ultima parte della tesi è dedicata allo studio approfondito dei cammini e degli algoritmi che abbiamo implementato. Il problema del partizionamento di cammini trova applicazioni in svariati campi tra i quali uno dei più interessanti riguarda l'elaborazione di immagini digitali. Spesso le immagini importate su un computer tramite scanner o ... sono rappresentate con una matrice di punti ognuno dei quali può essere rappresentato con un intero in cui il valore minore, lo 0, rappresenta un pixel nero, mentre il più grande (il numero 255) indica un pixel completamente bianco; i valori intermedi indicano quantità crescenti di luminosità in una "scala di grigio. Per vari problemi che possono sorgere che vanno dalla bassa risoluzione delle macchine adottate ai problemi legati all'occupazione di memoria di questo tipo di rappresentazioni, è spesso utile ridurre la scala di livelli di grigio fino ad appena 16 gradazioni diverse. Questo problema può essere ben rappresentato tramite i cammini in questo modo: ogni vertice del cammino rappresenta un colore e lo scopo è quello di partizionarlo in un numero di componenti fissato (16 per esempio). Il peso che associamo ad ogni vertice i rappresenta il numero di pixel aventi l'i-esimo livello di luminosità nell'immagine digitale. A questo punto è possibile partizionare

il cammino, pesato sui nodi nel modo che abbiamo appena detto, assumendo come funzione obiettivo una tra quelle associate alle varie norme: L_∞, L_1 e L_2 così definite:

- L_1: la funzione obiettivo è : $f(\pi) = \sum_{k=1}^{n} |W(C_k) - \mu|$
- L_2: la funzione obiettivo è : $f(\pi) = \sum_{k=1}^{n} (W(C_k) - \mu)^2$
- L_∞: la funzione obiettivo è : $f(\pi) = \max_k |W(C_k) - \mu|$

dove con μ si è indicato il peso medio di $\pi = \{C_1 \dots C_k\}$

$$\mu = \frac{W(C_1) + W(C_2) + \dots W(C_p)}{p} = \frac{W(V)}{p}$$

A questo proposito abbiamo ritenuto importante studiare alcuni algoritmi per il bilanciamento delle immagini digitali che tengono conto del minimo (L) e il massimo (U) numero di pixel all'interno di ogni componente e che cercano di risolvere il problema del partizionamento di un cammino pesato sui nodi trovando il più piccolo valore di $U - L$ oltre alla condizione che il peso di ogni componente debba rimanere nel range $[L, U]$.

Lucertini, Perl e Simeone [9] hanno dimostrato che il miglior contrasto ottico si ottiene tanto più quanto questa differenza è piccola. Il problema sopra descritto va sotto il nome di MUP (Most Uniform Partitioning). Oltre a questo caso particolare abbiamo considerato anche degli esempi meno restrittivi sulle condizioni, sui quali vengono implementati degli algoritmi che sfruttano una tecnica detta di *preprocessing*.

Per la parte più importante del lavoro, quella su cui abbiamo costruito le applicazioni presentate nell'ultima parte, abbiamo reputato interessante implementare l'algoritmo di Aparo e Simeone [1] con la norma L_1 e l'algoritmo di Liverani, Morgana, Siemone e Storchi [8] con la norma L_∞. Se tra tutte le partizioni che possono essere generate da un cammino dato in input, non consideriamo le partizioni improprie (ovvero tutte quelle partizioni che ammettono qualche componente nulla), le partizioni possibili le possiamo rappresentare con un grafo simile a quello di figura 1, in cui la partizione (a) rappresenta una partizione in cui le prime $p - 1$ componenti contengono un unico vertice, mentre l'ultima componente C_p contiene i rimanenti $n - p - 1$ vertici e la (b) rappresenta la partizione "opposta (in cui la prima componente C_1 contiene $n - p - 1$ vertici e gli altri vertici sono equamente distribuiti tra le rimanenti componenti).

Il primo algoritmo sfrutta un metodo di programmazione detto *programmazione dinamica* che risolve i problemi combinando la soluzione dei sottoproblemi comuni che calcola una sola volta memorizzando la loro soluzione per poterla riutilizzare (senza doverla ricalcolare) ogni volta che il sottoproblema si ripresenta. Il secondo, invece, fornisce un nuovo algoritmo per l'equipartizione di un cammino che migliora in modo consistente la complessità del problema, essendo caratterizzato da un'efficienza superiore a quella di altri algoritmi analoghi presenti in letteratura. Il metodo che sfrutta si basa su una tecnica detta *simulated annealing*, tecnica che differisce da molte altre per il criterio di accettazione della nuova partizione generata: vengono accettate infatti, anche partizioni che peggiorano il valore della funzione obiettivo, nella speranza che questo peggioramento permetta di superare eventuali minimi locali. Se osserviamo ancora la figura 1 possiamo sintetizzare i due algoritmi dicendo che, mentre l'algoritmo di Aparo e Simeone calcola tutte le possibili partizioni (proprie) scegliendo quelle con valore ottimo (la programmazione dinamica, infatti, non genera *una* soluzione con valore ottimo, ma diverse soluzioni ottime), l'altro algoritmo ne "salta alcune scegliendo di volta in volta (sotto alcune ipotesi) la soluzione che sembra essere la migliore.

Figura 1: Grafo di tutte le possibili p-partizioni di un cammino di n vertici

Riportiamo, infine, in figura , un elaborazione ottenuta con l'implementazione degli algoritmi presentati nella tesi. La prima figura (a) è l'immagine originale, in (b) abbiamo riportato l'immagine ottenuta con l'algoritmo MAX-MIN, (c) riporta l'immagine elaborata con l'algoritmo di Aparo e Simeone mentre in (d) abbiamo riportato quella ottenuta tramite l'algoritmo denominato PATH SHIFTING e dovuto a Liverani, Morgana, Simeone e Storchi. Abbiamo scelto l'elaborazione della raffigurazione de "Il bacio di Francesco Hayez perché ci sembra l'immagine che più mette in risalto la differenza tra gli algoritmi. In realtà, come è facile vedere, le immagini in (b) e (d) sono molto simili (anche se PATH SHIFTING sembra restituire un'immagine leggermente più definita soprattutto sul muro in secondo piano) mentre invece l'immagine che scaturisce dall'algoritmo di Aparo e Simeone è molto più "solarizzata e meno definita rispetto alle altre due. Anche grazie alle altre immagini considerate, alcune delle quali abbiamo riportato nell'appendice B della tesi, abbiamo potuto concludere che sia per via della migliore definizione, sia, soprattutto, per via della bassa complessità l'algoritmo di Liverani, Morgana, Simeone e Storchi è da preferire agli altri due.

Riferimenti bibliografici

[1] Enzo L. Aparo, B. Simeone, *Un algoritmo di equipartizione e il suo impiego in un problema di contrasto ottico*, Ricerca operativa n.6, 1973.

[2] R. I. Becker, I. Lari, M. Lucertini, B. Simeone, *Max-Min partitioning of grid graphs into connected components*, Networks 32, 1998, pp. 115-125.

[3] R. I. Becker, S. R. Scach, Y. Perl *A shifting algorithm for Min-Max tree partitioning*, Journal of the Association for Computing Machinery, Vol. 29, No. 1, gennaio 1982, pp. 58-67.

[4] F. Conti, F. Malucelli, S. Nicoloso, B. Simeone, *On a 2-dimensional equipartition problem*, preprint.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduzione agli algoritmi*, Jackson Libri, 1998, Vol. 1, (cap. 5), Vol. 2 (cap. 16-17), Vol. 3 (cap. 36-37).

[6] C. De Simone, M. Lucertini, S. Pallottino, B. Simeone, *Fair dissections of Spider, Worms and Caterpillars*, Networks, Vol. 20, 1990, pp. 323-344.

[7] M. R. Garey, D. S. Johnson, *Computers and Intractability. A guide to the theory of NP-completeness*, BT Laboratories, 1979.

[8] M. Liverani, A. Morgana, B. Simeone, G. Storchi, *Path equipartition in the Chebyshev norm*, European Journal of Operational Research, 123, 2000, pp. 428-436.

[9] M. Lucertini, Y. Perl, B. Simeone, *Most uniform path partitioning and its use in image processing*, Discrete Applied Mathematic 42, 1993, pp. 227-256.

[10] Y. Perl, S. R. Schach, *Max-Min tree partitioning*, Journal of the Association for Computing Machinery, Vol. 28, No. 1, gennaio 1981, pp. 5-15.

[11] M. Schroeder, *Balanced tree partitioning*, preprint, 2000.

Figura B.3: Elaborazione de “Il bacio”