Strategie risolutive e algoritmi per problemi di partizionamento ottimo di grafi

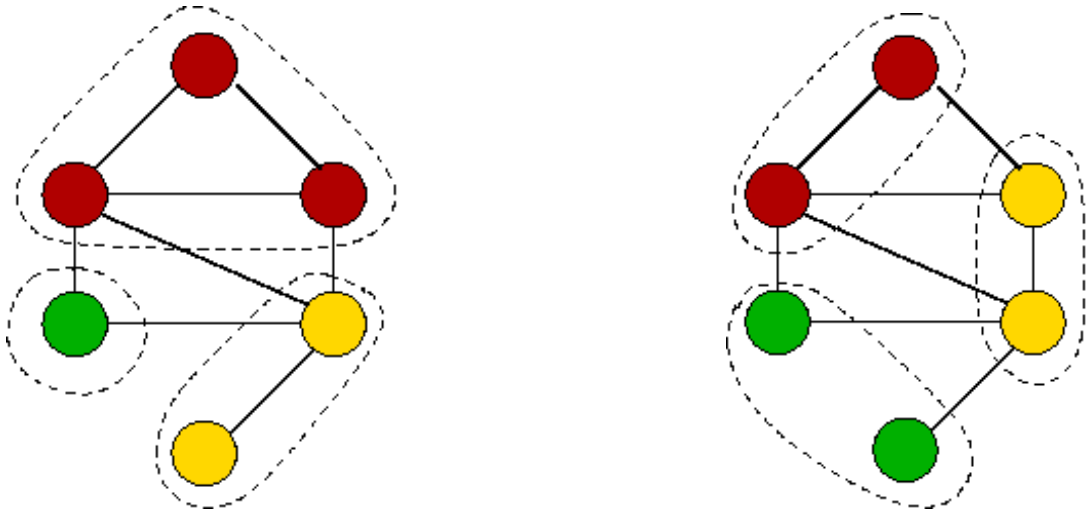
Natascia Piroso

12 luglio 2007

Definizione

*Dato un grafo non orientato $G = (V, E)$, una **partizione** di G è una suddivisione dei suoi vertici in insiemi a due a due disgiunti detti **componenti** della partizione.*

In particolare, siamo interessati a **partizioni connesse** del grafo, ovvero partizioni che inducono sottografi connessi.



Sia:

- p il numero di componenti della partizione
- $\Pi_p(G)$ l'insieme di tutte le possibili partizioni connesse di G con p componenti
- $\pi = (C_1, C_2, \ldots, C_p)$ una generica partizione in $\Pi_p(G)$

Sia:

- p il numero di componenti della partizione
- $\Pi_p(G)$ l'insieme di tutte le possibili partizioni connesse di G con p componenti
- $\pi = (C_1, C_2, \ldots, C_p)$ una generica partizione in $\Pi_p(G)$

Le grandezze caratterizzanti del problema concorrono alla definizione di una funzione della partizione stessa

$$f : \Pi_p(G) \rightarrow \mathbb{R}^+$$

che chiamiamo **funzione obiettivo**.

Massimizzandola o minimizzandola otteniamo la migliore partizione possibile conformemente all'obiettivo preposto.

Sia:

- p il numero di componenti della partizione
- $\Pi_p(G)$ l'insieme di tutte le possibili partizioni connesse di G con p componenti
- $\pi = (C_1, C_2, \ldots, C_p)$ una generica partizione in $\Pi_p(G)$

Le grandezze caratterizzanti del problema concorrono alla definizione di una funzione della partizione stessa

$$f : \Pi_p(G) \rightarrow \mathbb{R}^+$$

che chiamiamo **funzione obiettivo**.

Massimizzandola o minimizzandola otteniamo la migliore partizione possibile conformemente all'obiettivo preposto.

Dato un grafo G non orientato, trovare una partizione connessa di G in un numero prefissato p di classi, minimizzando (o massimizzando) una data funzione della partizione.

Le **applicazioni reali** riconducibili a problemi di partizionamento ottimo di grafi sono numerosissime e appartengono ad ambiti applicativi eterogenei.

- ingegneria
- ricerca operativa
- biologia, medicina, chimica
- finanza, economia
- scienze decisionali in generale

Esempi

- Progettazione di circuiti VLSI
- Uso razionale della memoria di un calcolatore parallelo
- Pianificazione urbana
- Visualizzazione di grafi di vaste dimensioni
- Scheduling di equipaggi
- Riproduzioni di immagini in qualità ridotta

Formalmente, si distinguono due principali classi di problemi di partizionamento ottimo di grafi:

- Problemi di **equipartizione**
- Problemi di **clustering**

Formalmente, si distinguono due principali classi di problemi di partizionamento ottimo di grafi:

- Problemi di **equipartizione**
- Problemi di **clustering**

Equipartizione

Ogni vertice $i \in V$ ha associato un peso non negativo e il peso di una componente si definisce come somma dei pesi dei vertici della componente.
Obiettivo ideale: Partizionare il grafo in p componenti tutte di egual peso μ, dove

$$\mu = W(V)/p$$

è il peso medio delle componenti della partizione.
Quasi mai tale configurazione rientra nello spazio delle partizioni ammissibili. Possiamo solo tendere a questo obiettivo ideale.

Esempi (Funzioni obiettivo per l'equipartizione)

norma L_1

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

norma L_2

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

norma L_∞

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

Esempi (Funzioni obiettivo per l'equipartizione)

norma L_1

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

norma L_2

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

norma L_∞

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

max-min

$$f(\pi) = \min_{1 \le k \le p} W(C_k)$$

min-max

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

Esempi (Funzioni obiettivo per l'equipartizione)

norma L_1

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

norma L_2

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

norma L_∞

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

max-min

$$f(\pi) = \min_{1 \le k \le p} W(C_k)$$

min-max

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

most uniform

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

Clusterig

Ad ogni coppia di vertici i e j del grafo è assegnato un indice d_{ij} della loro **dissimilarità**.

Obiettivo: Classificare i vertici del grafo in modo da favorire

l'omogeneità vertici in uno stesso cluster sono il più possibile simili tra loro

la separazione vertici appartenenti a cluster distinti sono il più possibile dissimili tra loro

Clusterig

Ad ogni coppia di vertici i e j del grafo è assegnato un indice d_{ij} della loro **dissimilarità**.

Obiettivo: Classificare i vertici del grafo in modo da favorire

l'omogeneità vertici in uno stesso cluster sono il più possibile simili tra loro

la separazione vertici appartenenti a cluster distinti sono il più possibile dissimili tra loro

Esempi

inner-dissimilarity

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

sum of splits

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

Principali tecniche di partizionamento

Programmazione dinamica

- Suddivisione del problema in sottoproblemi via via più piccoli
- Risoluzione dai problemi di dimensione minima ai sottoproblemi di dimensione maggiore
- Metodo di risoluzione tabulare
- Sottoproblemi comuni calcolati una sola volta

Principali tecniche di partizionamento

Programmazione dinamica

- Suddivisione del problema in sottoproblemi via via più piccoli
- Risoluzione dai problemi di dimensione minima ai sottoproblemi di dimensione maggiore
- Metodo di risoluzione tabulare
- Sottoproblemi comuni calcolati una sola volta

Strategie greedy

- Decisioni locali ottime
- Algoritmi di bassa complessità, semplici ed intuitivi

Principali tecniche di partizionamento

Migrazione di gruppi

- Generazione di una partizione iniziale
- Miglioramento della partizione attraverso lo scambio di un gruppo di vertici da una componente ad un'altra
- Selezione del gruppo di migrazione sulla base di scelte locali

Principali tecniche di partizionamento

Migrazione di gruppi

- Generazione di una partizione iniziale
- Miglioramento della partizione attraverso lo scambio di un gruppo di vertici da una componente ad un'altra
- Selezione del gruppo di migrazione sulla base di scelte locali

Simulated annealing

- Migrazione di gruppi con passi in cui il valore della funzione obiettivo può peggiorare
- Superamento di eventuali minimi (o massimi) locali
- Condizioni di arresto più forti

Complessità computazionale di un problema di partizionamento

- Grafi qualsiasi: problemi NP-completi
- Alberi: la complessità computazionale dipende dalla funzione obiettivo
- Cammini: algoritmi esatti di complessità polinomiale

Partizionamento di cammini

Aparo, Simeone - 1973

Il problema del p-partizionamento di un cammino definito dalle funzioni obiettivo

- norma L_1
- norma L_2
- norma L_∞

è risolvibile in tempo $O(pn^2)$ attraverso la programmazione dinamica .

Partizionamento di cammini

Aparo, Simeone - 1973

Il problema del p-partizionamento di un cammino definito dalle funzioni obiettivo

- norma L_1
- norma L_2
- norma L_∞

è risolvibile in tempo $O(pn^2)$ attraverso la programmazione dinamica .

Lo stesso procedimento può essere **generalizzato** per le funzioni obiettivo della forma

$$f(\pi) = \theta(C_1) * \theta(C_2) * \cdots * \theta(C_p)$$

con

- $*$ operazione binaria associativa e commutativa su $\mathbb{R}^+$
- θ applicazione che associa ad ogni possibile sottocammino di P un elemento in $\mathbb{R}^+$.

Partizione Most Uniform

Per alcune funzioni obiettivo, l'utilizzo di tecniche di partizionamento più sofisticate ha permesso di trovare algoritmi più efficienti.

Partizione Most Uniform

Per alcune funzioni obiettivo, l'utilizzo di tecniche di partizionamento più sofisticate ha permesso di trovare algoritmi più efficienti.

In molte applicazioni reali, la funzione obiettivo *most uniform* è ritenuta la più appropriata funzione per l'equipartizione di un grafo. Il relativo problema di partizionamento è

- NP-completo per grafi qualsiasi
- irrisolto nel caso di un albero
- risolvibile in modo esatto nel caso di un cammino

Partizione Most Uniform

Per alcune funzioni obiettivo, l'utilizzo di tecniche di partizionamento più sofisticate ha permesso di trovare algoritmi più efficienti.

In molte applicazioni reali, la funzione obiettivo *most uniform* è ritenuta la più appropriata funzione per l'equipartizione di un grafo. Il relativo problema di partizionamento è

- NP-completo per grafi qualsiasi
- irrisolto nel caso di un albero
- risolvibile in modo esatto nel caso di un cammino

Lucertini, Perl, Simeone - 1993

Il problema del p-partizionamento di un cammino definito dalla funzione *most uniform* è risolvibile con un algoritmo esatto di complessità computazionale $O(pn^2 \log p)$.

Partizione Most Uniform di cammini

L'algoritmo si basa sulla risoluzione ripetuta di un problema di partizionamento correlato, quello della ricerca di una partizione-(L, U) in p componenti.

Definizione

Una p-partizione-(L, U) di un grafo G pesato sui vertici è una partizione di G in p componenti di peso compreso tra i valori L ed U, con $L \leq U$.

Partizione Most Uniform di cammini

L'algoritmo si basa sulla risoluzione ripetuta di un problema di partizionamento correlato, quello della ricerca di una partizione-(L, U) in p componenti.

Definizione

Una p-partizione-(L, U) di un grafo G pesato sui vertici è una partizione di G in p componenti di peso compreso tra i valori L ed U, con $L \leq U$.

Trovare una p-partizione che ottimizzi la funzione *most uniform* equivale infatti a trovare una partizione-(L, U) in p componenti tale che la differenza $U - L$ sia minima.

La coppia (L, U) con differenza $U - L$ minima viene cercata in un'opportuna regione ammissibile del piano (L, U) con ricerche binarie che portano a definire la complessità $O(pn^2 \log n)$ dell'algoritmo.

Partizione Most Uniform di cammini

Il problema del p-partizionamento di un cammino può essere risolto direttamente, senza passare per problemi di partizionamento ausiliari.

L'estensione dell'algoritmo di programmazione dinamica di Aparo e Simeone a tutte funzioni obiettivo della forma

$$f(\pi) = \theta(C_1) * \theta(C_2) * \cdots * \theta(C_p)$$

può essere riadattata per risolvere il problema definito dalla funzione *most uniform*.

L'algoritmo

- Un generico passo dell'algoritmo determina la migliore distribuzione possibile di vertici tra le componenti

$$C_k, C_{k+1}, \ldots, C_p$$

  nei vari casi determinati dall'indice scelto come primo vertice di C_k
- Si procede da $k = p - 1$ a $k = 1$

L'algoritmo

- Un generico passo dell'algoritmo determina la migliore distribuzione possibile di vertici tra le componenti

$$C_k, C_{k+1}, \ldots, C_p$$

  nei vari casi determinati dall'indice scelto come primo vertice di C_k
- Si procede da $k = p - 1$ a $k = 1$

I risultati ottenuti ad ogni passo sono memorizzati in un tabella.
In particolare, viene memorizzato:

- Il valore della funzione obiettivo calcolato nella partizione ottima del sottocammino considerato
- Il peso della componente più pesante
- Il peso della componente più leggera

Possiamo così utilizzare le soluzioni dei sottoproblemi già considerati per costruire le soluzioni dei problemi di dimensione via via maggiore.

L'algoritmo

Quando $k = 1$, il primo vertice della nuova componente considerata non può essere che 1. La soluzione ottima del problema determinato da quest'unico caso determina la soluzione ottima del problema originario.

L'algoritmo

Quando $k = 1$, il primo vertice della nuova componente considerata non può essere che 1. La soluzione ottima del problema determinato da quest'unico caso determina la soluzione ottima del problema originario.

La complessità computazionale dell'algoritmo è $O(pn^2)$: si migliora, di un fattore $\log n$, la complessità calcolata con l'algoritmo di Lucertini, Perl e Simeone.

FINE