UNIVERSITÀ CA' FOSCARI – VENEZIA
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

Laureando: Andrea Albarelli

Una Formulazione Continua per il Matching di Grafi

Relatore: Chiar.mo prof. Marcello Pelillo
Correlatore: Prof. Andrea Torsello

Anno Accademico 2005/06

Abstract

Il graph matching, ovvero la ricerca di corrispondenze fra elementi di strutture relazionali, è un tema centrale di molti ambiti matematici ed informatici, in quanto un gran numero di problemi può essere ricondotto ad esso. Per questo motivo risulta essere anche un problema estremamente studiato per le cui istanze sono state proposte una moltitudine di tecniche risolutive sia esatte che approssimate, queste ultime essendo spesso inevitabili dal punto di vista pratico a causa della natura non polinomiale della formulazione generale del problema.

In questa tesi ci occupiamo della versione più generale del problema, ovvero della ricerca di match fra sottografi di grafi dotati di attributi e di misure di similarità fra di essi. Dopo aver fornito una panoramica delle tecniche di maggior successo proposte in letteratura presenteremo una formulazione continua riducendo il problema alla ricerca di un sottografo completo di massima cardinalità (max-clique) in un grafo ausiliario mediante la massimizzazione di una funzione quadratica. L'approccio continuo a max-clique utilizzato è stato proposto per la prima volta da Motzkin e Straus[1], recentemente esteso da Gibbons al caso di grafi con vertici pesati [2] e da Pelillo che propone una generalizzazione a grafi con archi pesati del concetto di max-clique [3].

Il problema di massimizzazione poi viene risolto in modo approssimato mediante l'uso di replicatori dinamici sviluppati nell'ambito della biologia computazionale.

I risultati ottenuti mediante la nostra formulazione verranno infine confrontati con la tecnica basata sul Graduated Assignment [4] permettendo di evidenziarne pregi e difetti che saranno discussi assieme alle ipotesi di lavoro futuro.

Indice

Elenco delle figure

Elenco delle tabelle

Notazioni

$\mathbb{R}$	insieme dei numeri reali
$A \times B$	prodotto cartesiano degli insiemi A e B
$\|A\|$	cardinalità di un insieme A
$f : A \to B$	funzione f con dominio A e codominio B
$\mathbf{x}, \mathbf{y}, \mathbf{z}, \cdots$	lettere in grassetto che rappresentano vettori
Δ	Il simplesso standard in n dimensioni (dove n è generalmente chiaro dal contesto)
$\mathbf{e}$	Vettore con tutti gli elementi pari a 1
I	matrice identica
$\sigma(\mathbf{x})$	supporto del vettore $\mathbf{x}$
$\mathbf{x}'$	trasposizione di un vettore $\mathbf{x}$
A'	trasposizione di una matrice A

Capitolo 1

Introduzione

1.1 Il problema del graph matching

Un grafo è una struttura relazionale costituita da un insieme di nodi (o vertici) che possono essere collegati fra di loro da archi[1]. Nella figura 1.1 vediamo un esempio di un semplice grafo con 4 vertici e 4 archi che collegano alcuni dei vertici.

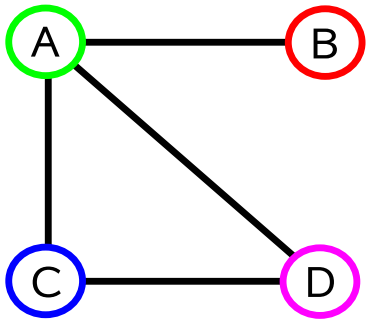

Figura 1.1: Esempio di semplice grafo

Due grafi si dicono isomorfi se è possibile definire una biiezione fra i loro vertici, detta appunto isomorfismo, che preservi la relazione di adiacenza: ovvero per ogni coppia di vertici che dal primo grafo vengono proiettati nel secondo, questi devono essere collegati da un arco nel grafo di partenza se e solo se anche le loro immagini sono collegate da un arco.

[1]Si veda il paragrafo 3.1 per una definizione più formale di grafo

Il problema del massimo sottografo isomorfo comune (MCS), che per brevità chiameremo semplicemente graph matching[2], è un problema decisionale NP completo che richiede, dati due grafi G_1 e G_2, di individuare il più grande sottografo di G_1 isomorfo ad un sottografo di G_2.

In questa semplice formulazione il concetto di massimalità si intende relativo alla cardinalità dell'isomorfismo, ovvero al numero di vertici dei quali è composto ciascun sottografo coinvolto. Nell'esempio in figura 1.2 vediamo un isomorfismo di massima cardinalità fra sottografi di due semplici grafi.

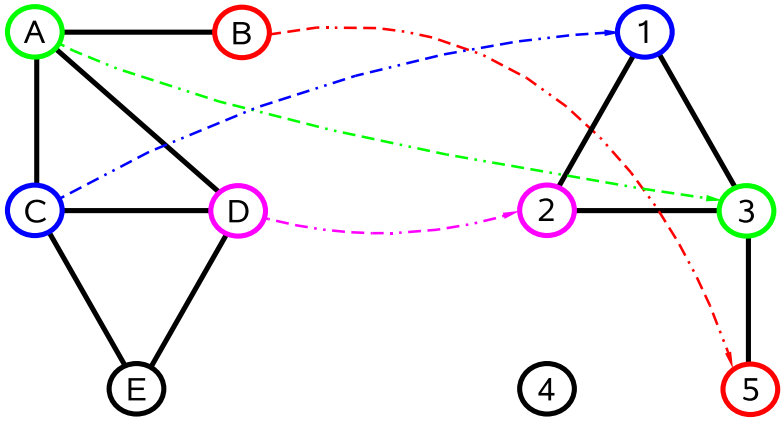

Figura 1.2: Esempio di massimo sottografo isomorfo comune.

Nel caso di figura 1.2 la cardinalità dell'isomorfismo trovato risulta essere pari a 4 ed è facile verificare che non è possibile trovare isomorfismi di cardinalità superiore.

Se aggiungiamo degli attributi ai vertici dei due grafi e definiamo una misura di similarità fra tali attributi possiamo riformulare il problema dalla ricerca di un isomorfismo di massima cardinalità a quella di una biiezione isomorfa fra i vertici di G_1 e G_2 tale da massimizzare la somma delle similarità fra i vertici coinvolti in G_1 e le loro immagini in G_2. Nell'esempio di figura 1.3 abbiamo associato a ciascun vertice un attributo naturale e abbiamo definito una funzione di similarità $S : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+$ come $S(a, b) = e^{-|a-b|}$.

Vediamo che in questo caso l'isomorfismo presentato in figura 1.2 non soddisfa più la nuova definizione di massimalità, la quale viene invece soddisfatta da un isomorfismo di cardinalità inferiore ma di peso totale massimo. Infatti il peso del nuovo isomorfismo risulta essere $e^{-1} + e^{-1} + e^{-1} = \frac{3}{e}$, mentre il peso totale dell'isomorfismo presentato in figura 1.2 risulta essere $e^{-1} + e^{-1} + e^{-2} + e^{-4} < \frac{3}{e}$.

[2]Il termine *graph matching* è piuttosto generale, ma in questo contesto lo applicheremo sempre a MCS e a sue generalizzazioni

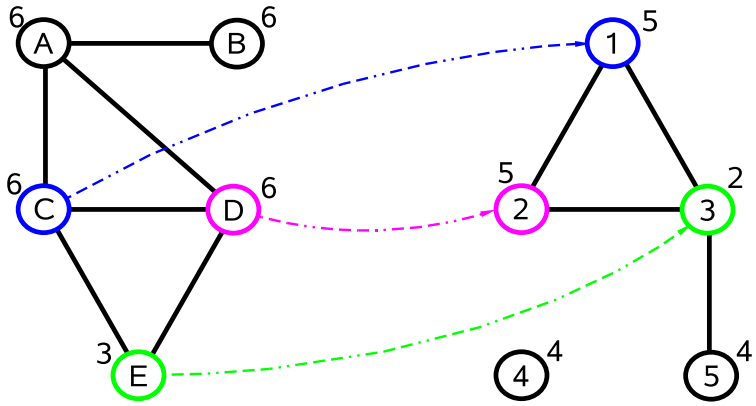

Figura 1.3: Esempio di sottografo isomorfo comune di massima similarità.

Se oltre ad aggiungere attributi ai vertici ne aggiungiamo anche agli archi la nostra definizione di massimalità cambia ulteriormente.

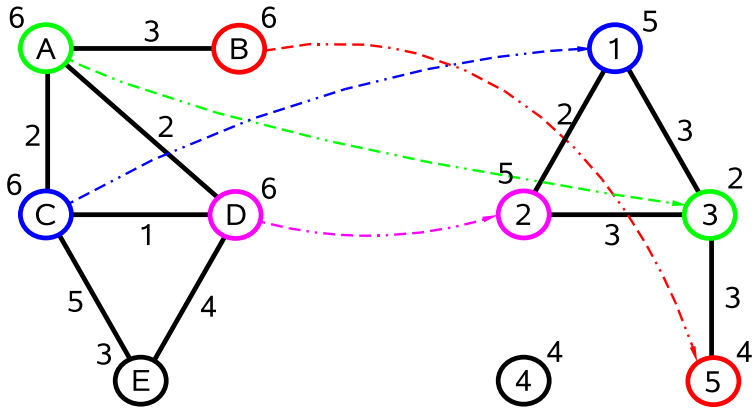

Figura 1.4: Esempio di sottografo isomorfo comune di massima similarità fra grafi con attributi su vertici ed archi.

Come si può vedere in figura 1.4, cercando di massimizzare la somma delle similarità dovute sia ai vertici che agli archi (entrambe espresse dalla medesima funzione $S(a, b) = e^{-|a-b|}$) l'isomorfismo massimo cambia ancora.

In generale non è necessario che gli attributi siano naturali, ne' che il loro dominio sia lo stesso sia per quelli associati ai vertici che per quelli associati agli archi. Infatti possiamo immaginare di associare a ciascun vertice ed arco un vettore di attributi che possono essere sia quantitativi che qualitativi, purchè sia sempre possibile definire fra tali vettori delle funzioni di similarità atte a determinare il peso complessivo di un match.

1.2 Applicazioni del graph matching

In letteratura è possibile trovare un grandissimo numero di applicazioni del graph matching, le quali spaziano in molti diversi ambiti scientifici. Nel 1979 Rouvray e Balaban individuano una delle prime applicazioni nel campo dell'analisi delle strutture chimiche [5]. Più di recente il graph matching è stato applicato al case-based reasoning (CBR), una tecnica per risolvere problemi basandosi sulla soluzione di casi simili già risolti [6], [7]. Altre applicazioni includono il machine learning [8], [9], [10], la rappresentazione della conoscenza nelle reti semantiche [11], grafi concettuali e sistemi esperti [12] e supervisione delle reti di calcolatori [13].

Le applicazioni che ci interessano di più sono comunque quelle inerenti al campo della visione artificiale e del riconoscimento di pattern. In quest'ambito possiamo trovare il riconoscimento di simboli grafici [14], [15], il riconoscimento di caratteri [16], [17], l'analisi delle forme e degli oggetti bidimensionali [18], [19], [20] e tridimensionali [21].

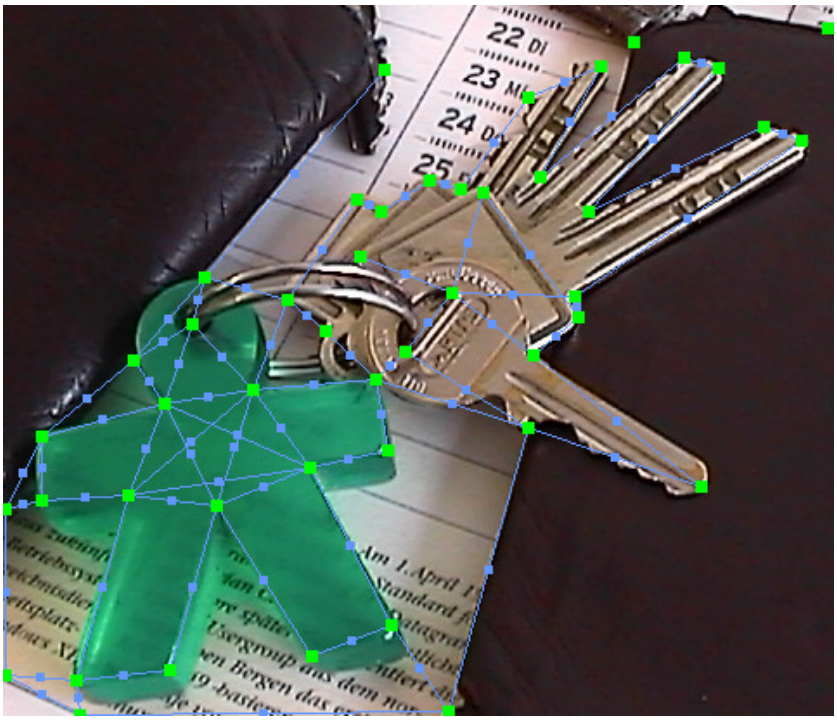

Figura 1.5: Feature graph generato da edge detection su scena

In generale, per quanto riguarda quest'ultimo ambito, possiamo ricondurre alla ricerca di un match ottimale fra grafi una vasta gamma di problemi di visione artificiale.

L'organizzazione del grafo, la relazione fra i vertici e gli attributi da assegnare ad archi e vertici dipendono dalla specifica applicazione e dal modo in cui utilizziamo dei grafi con attributi per rappresentare gli aspetti salienti delle immagini esaminate che che ci interessano.

Nell'esempio di figura 1.5 vediamo un grafo di feature (indicate dai vertici) individuate mediante l'applicazione di un algoritmo di edge detection. Alcuni di questi vertici sono marcati, mediante un altro algoritmo con attributi che indicano che si tratta di spigoli della figura.

Figura 1.6: Feature graph generato da edge detection su oggetto

Applicando lo stesso processo ad un'immagine di un singolo oggetto contenuto nella scena originale (eventualmente con prospettiva e scale diverse) possiamo ottenere un grafo rappresentativo delle feature di tale oggetto (figura 1.6).

A questo punto è possibile applicare un algoritmo di graph matching per localizzare la presenza (o meno) dell'oggetto rappresentato in figura 1.6 nella scena originale.

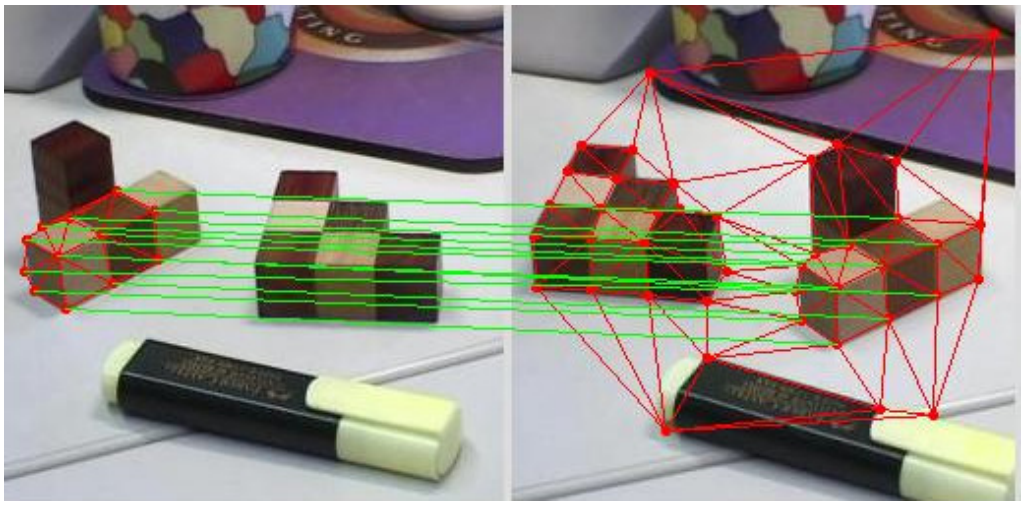

Figura 1.7: Match fra grafi con feature indicate manualmente

In alcuni casi le feature possono essere indicate anche manualmente da un operatore. In figura 1.7 vediamo il match fra un grafo di feature indicato manualmente in una scena ed un grafo più grande associato ad una scena diversa nella quale la disposizione degli elementi è stata cambiata.

Una tecnica completamente differente che permette di associare ad un'immagine un grafo che ne catturi le caratteristiche è quella degli shock tree, basata sull'individuazione di singolarità (shock) in un processo di evoluzione di curve all'interno di semplici silhouette di oggetti costituite da curve chiuse [22].

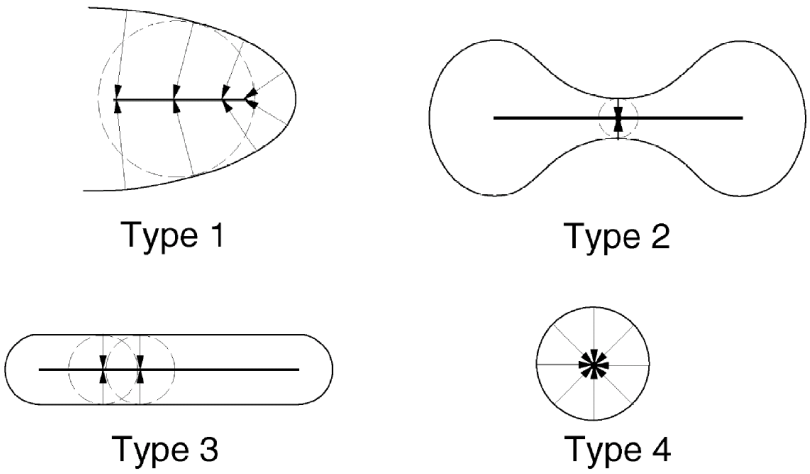

Figura 1.8: Tipologie di shock lungo l'asse mediano di una figura piana

Si possono classificare quattro diversi tipi di shock in base all'andamento della funzione raggio lungo l'asse mediano considerato della figura piana (vedi figura 1.8).

Gli shock di tipo 1 derivano da una protrusione e disegnano tipicamente un segmento di curva formato da una serie di shock di tipo 1 adiacenti lungo il quale la funzione raggio diminuisce monotonicamente. Uno shock di tipo 2 corrisponde ad uno strozzamento, ovvero ad una zona dove la funzione raggio trova un minimo locale. Il tipo 3 è associato a un segmento di curva dove la funzione raggio risulta costante e il tipo 4 a punti dove questa raggiunge un massimo locale.

Un insieme di shock corrispondenti ad una figura può essere riformulato in uno shock graph, all'interno del quale i vertici vengono etichettati con un numero corrispondente al loro tipo di shock e gli archi vengono creati in base al tempo di formazione dei singoli shock. Un insieme di regole permette la riduzione di questi grafi ad alberi radicati [23], [24].

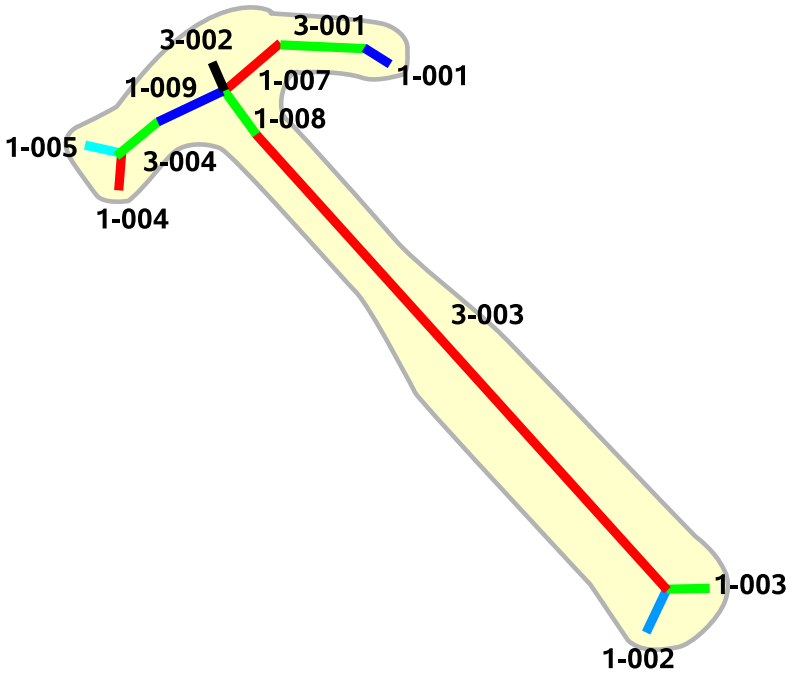

Figura 1.9: Generazione di shock tree sulla silhouette di un martello

Nella figura 1.9 vediamo una marcatura degli shock generati all'interno della silhouette di un martello e nella figura 1.10 possiamo vedere lo shock tree corrispondente. Dal match di shock graph prodotti da diverse figure piane

è possibile determinare la loro similarità ed eventualmente raggrupparle in classi di equivalenza flat o gerarchiche.

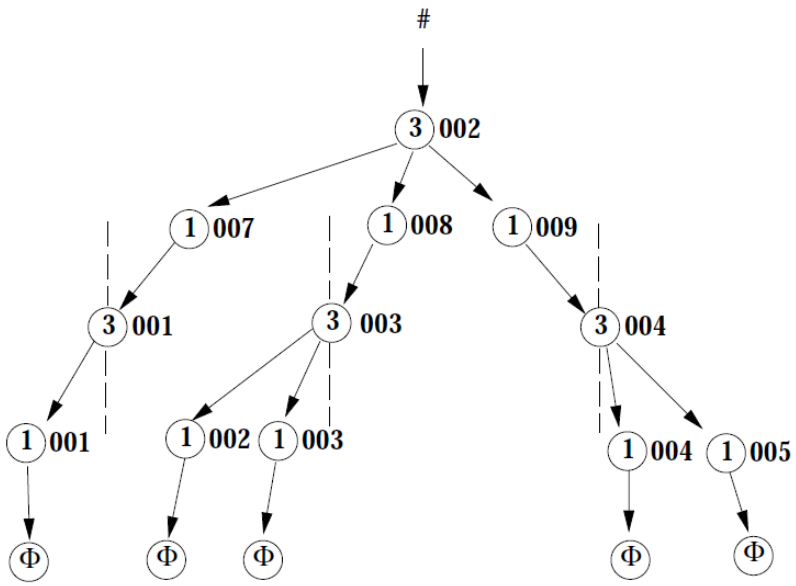

Figura 1.10: Shock tree generato dalla silhouette di figura 1.9

Un ulteriore metodo per rappresentare le caratteristiche discriminanti di oggetti rappresentati all'interno di immagini sotto forma di grafi è quello del saliency graph map.

Si tratta di una rappresentazione delle informazioni caratterizzanti le immagini basata su una parametrizzazione numerica indipendente dalla scala e non fondata sull'estrazione di caratteristiche di alto livello quali linee, curve o regioni uniformi. Inoltre tale rappresentazione gode di una località spaziale tale da attenuare fortemente le perturbazioni dovute alle occlusioni parziali degli oggetti o a leggere deformazioni degli stessi o cambiamenti di illuminazione.

Questa tecnica prevede l'applicazione di una trasformata wavelet all'immagine, ovvero di un'operatore in grado di scomporre il segnale sia nel dominio della frequenza che del tempo. In base a tale scomposizione viene definita una mappa gerarchica che individua, a differenti scale di risoluzione, le regioni salienti dell'immagine (figura 1.11). A loro volta tali regioni vengono mappate come nodi in un grafo diretto e aciclico (figura 1.12) dove gli archi uniscono regioni ampie a regioni più fini ogni volta che il centro dell'area di estensione di una regione fine cade all'interno di quella di una regione ampia. La struttura gerarchica che ne risulta, chiamata appunto saliency map gra-

Figura 1.11: Esempio di saliency map generata su un'immagine

ph, cattura sia le informazioni topologiche che geometriche contenute nella saliency map.

Dettagli su questo tipo di rappresentazione, incluse le tecniche per calcolarla e le sue proprietà invarianti, si possono trovare in [25], [26], [27].

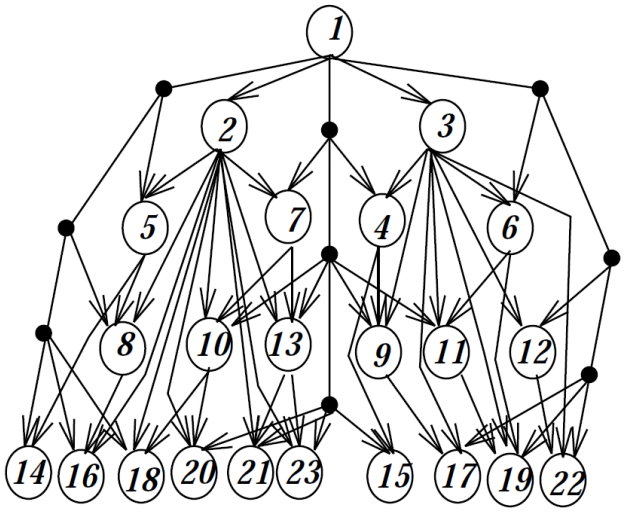

Figura 1.12: Saliency map graph corrispondente all'immagine di figura 1.11

1.3 Obiettivi e struttura della tesi

In questa tesi ci occupiamo di metodologie approssimate per il matching di grafi con attributi. Tale problema è di natura combinatoria ed NP completo (almeno nella nostra formulazione), pertanto ogni tentativo di aggredirlo con algoritmi che permettano di ottenere una soluzione esatta è uno sforzo del tutto velleitario.

Poichè il graph matching ha importanti applicazioni pratiche in molteplici ambiti (visione artificiale, ottimizzazione,etc.) il problema è stato analizzato per lungo tempo e nel corso degli anni sono state proposte varie euristiche per affrontarlo: nel capitolo 2 prendiamo in esame le più importanti di esse all'interno di una breve review di letteratura.

All'interno del capitolo 3 tratteremo il problema del matching mediante ricerca di clique massima all'interno di un grafo ausiliario detto grafo di associazione. Dopo aver illustrato una formulazione continua classica per questo sottoproblema valida nel caso di grafi non pesati [1] introdurremo una sua estensione al caso di grafi pesati [3] e ne proporremo l'applicazione ad un grafo di associazione pesato derivato dal problema di matching originale. Infine proporremo un approccio basato su una diverso grafo di associazione alla quale applicheremo un'ulteriore formulazione continua [2].

Il capitolo 4 illustra un procedimento effettivo per cercare in modo approssimato una clique massima all'interno dei grafi di associazione proposti. Tale tecnica, basata sulle dinamiche di replicazione utilizzate in biologia computazionale, non garantisce l'individuazione di una clique massima (e quindi di un isomorfismo massimo nel match associato), tuttavia viene garantita comunque la massimalità della soluzione e, come vedremo, spesso permette di trovare approssimazioni molto valide.

Nel capitolo 5 viene presentata una serie di dati sperimentali che mettono a confronto i risultati ottenuti mediante applicazione delle dinamiche di replicazione ai grafi di associazione da noi proposti nel capitolo 3 con quelli ottenibili da un algoritmo molto noto in letteratura [4]. Infine nel capitolo 6 vengono tratte le conclusioni sui risultati ottenuti e si propone un elenco di lavori futuri e approfondimenti che vengono ritenuti degni di ulteriore indagine.

Capitolo 2

Stato dell'arte nel Graph Matching

Data la sua importanza dal punto di vista teorico e applicativo, il problema del graph matching è stato affrontato da un consistente numero di ricercatori e aggredito da diverse angolazioni. In generale si possono individuare due classi principali di tecniche: gli algoritmi esatti, che usano costose tecniche di ricerca deterministiche (come ad esempio visita di alberi con backtracking [28],[29],[30],[31]) per esplorare le possibili soluzioni del problema, e algoritmi per match inesatto che individuano euristiche che permettano di affrontare la ricerca della soluzione in modo approssimato impiegando un tempo polinomiale nella dimensione dei grafi coinvolti. Come vedremo tale obiettivo è ottenibile con approcci anche molto diversi fra loro, ciascuno dei quali ha caratteristiche peculiari in termini di qualità delle soluzioni trovate, tempi di convergenza e condizioni di applicazione ottimali.

2.1 Graduated Assignment

Gold e Rangarajan in [4] presentano un algoritmo per il matching di grafi pesati mediante ottimizzazione di una funzione non lineare basato su tre concetti fondamentali: i rispetto dei vincoli bidirezionali negli assegnamenti, ottenuto mediante il softassign, la capacità di sfuggire ai minimi locali, mediante lo sfruttamento della nonconvessità graduata, e l'utilizzo della sparsità, una tecnica derivante dal relaxation labeling framework, per migliorare la velocità di convergenza.

Questo metodo non cerca all'interno dello spazio degli stati, come fanno gli algoritmi di ricerca di soluzioni esatti o altre euristiche. Piuttosto punta alla minimizzazione, nell'ambito del continuo, di una funzione non lineare

della matrice di match fra i grafi pesati G_1 e G_2, ovvero di una matrice $M = (M_{ij})$ così definita:

$$M_{ai} = \begin{cases} 1 & \text{se il nodo } a \text{ in } G_1 \text{ è associato al nodo } i \text{ in } G_2 \\ 0 & \text{altrimenti} \end{cases} \tag{2.1}$$

La funzione da minimizzare è soggetta a vincoli ed un volta minimizzata, dopo opportune procedure di pulizia, la matrice M indicherà le coppie di vertici associati, ovvero il match trovato.

Per definire la funzione obiettivo abbiamo innanzitutto bisogno di definire una misura di similarità fra gli archi dei due grafi. Sia $G_1(ab)$ il peso (o gli attributi) dell'arco del grafo G_1 che collega il vertice a al vertice b (o NULL nel caso tale arco non sia presente) e sia analogamente $G_2(ij)$ il peso (o gli attributi) dell'arco del grafo G_2 che collega il vertice i al vertice j (o NULL), la matrice di compatibilità fra gli archi $C = (C_{aibj})$ (ovvero la compatibilità fra l'arco ab in G_1 e ij in G_2) è così definita:

$$C_{aibj} = \begin{cases} 0 & \text{se } G_1(ab) \text{ o } G_2(ij) \text{ sono NULL} \\ \gamma(G_1(ab), G_2(ij)) & \text{altrimenti} \end{cases} \tag{2.2}$$

Dove γ è una qualsiasi misura di similarità fra i pesi o i vettori. In questa forma la matrice di compatibilità assegna peso minimo alle associazioni indotte fra archi assenti (o incompatibili) e un'appropriata misura di similarità negli altri casi.

A questo punto possiamo definire la funzione obiettivo da minimizzare che risulta essere:

$$E(M) = -\frac{1}{2} \sum_{a=1}^{n} \sum_{i=1}^{m} \sum_{b=1}^{n} \sum_{j=1}^{m} M_{ai} M_{bj} C_{aibj} \tag{2.3}$$

Soggetta a

$$\forall a \sum_{i=1}^{m} M_{ai} \leq 1 \text{ con } M_{ai} \in \{0, 1\} \tag{2.4}$$

e

$$\forall i \sum_{a=1}^{n} M_{ai} \leq 1 \text{ con } M_{ai} \in \{0, 1\} \tag{2.5}$$

Ovvero le condizioni che rendono M una matrice di permutazione corretta. In aggiunta, poichè è possibile che i grafi abbiano dimensioni diverse o che comunque alcuni nodi di ciascun grafo non possano essere mappati, è necessario modificare la matrice M aggiungendo una riga ed una colonna extra (dette di *slack*) allo scopo di accomodare eventuali vertici non assegnati senza alterare la possibilità di soddisfare i vincoli.

Il problema 2.3 è un problema quadratico di assegnazione, che al contrario del problema di assegnazione classico, che è risolvibile in tempo polinomiale, è noto essere NP-completo. Gold e Rangarajan propongono quindi una tecnica per risolvere 2.3 in modo approssimato mediante la risoluzione successiva di una successione di problemi di assignment semplici.

Il problema di assignment, nella sua forma tradizionale, chiede dato un insieme di variabili reali X_{ai} ed una matrice doppiamente stocastica M, ovvero soggetta alle condizioni 2.4 e 2.5, di trovare il valore di M che massimizza

$$E_{ass}(M) = \sum_{a=1}^{n} \sum_{i=1}^{m} M_{ai} X_{ai} \tag{2.6}$$

Questo problema (si veda anche [32]) può essere risolto in passi successivi applicando delle tecniche di softassign per variare i valori di M opportunamente e poi ottenere ad ogni passo una matrice doppiamente stocastica applicando un importante risultato dovuto a Sinkhorn [33] il quale dimostra che qualsiasi matrice quadrata ad elementi positivi può convergere ad una matrice doppiamente stocastica mediante un processo iterativo di normalizzazione alternata di righe e colonne.

L'algoritmo proposto da Gold e Rangarajan (si veda [4] per i dettagli) parte da una matrice M iniziale e ripete iterativamente un ciclo dove calcola l'espansione di Taylor al primo ordine per 2.3, la cui minimizzazione corrisponde alla massimizzazione di un problema di assignment semplice nella forma 2.6, massimizza lo stesso mediante softassign, aggiorna la matrice M (dopo aver effettuato la normalizzazione di Sinkhorn) e prosegue fino al verificarsi di una condizione di arresto.

Le condizioni di arresto dell'algoritmo hanno a che vedere con la convergenza della matrice M e con il raggiungimento di determinati valori da parte di parametri di controllo.

Una volta terminato l'algoritmo è necessario un ulteriore step di clean-up in quanto la matrice ottenuta non è sempre una matrice di permutazione (anche se generalmente risulta dominante per righe).

In [4] l'algoritmo viene testato con grafi derivanti da oggetti e scene con feature marcate manualmente, come quelli di figura 2.1 e con grafi gene-

rati casualmente da 100 nodi e soggetti a vari tipi di rumore strutturale e perturbazione sugli attributi.

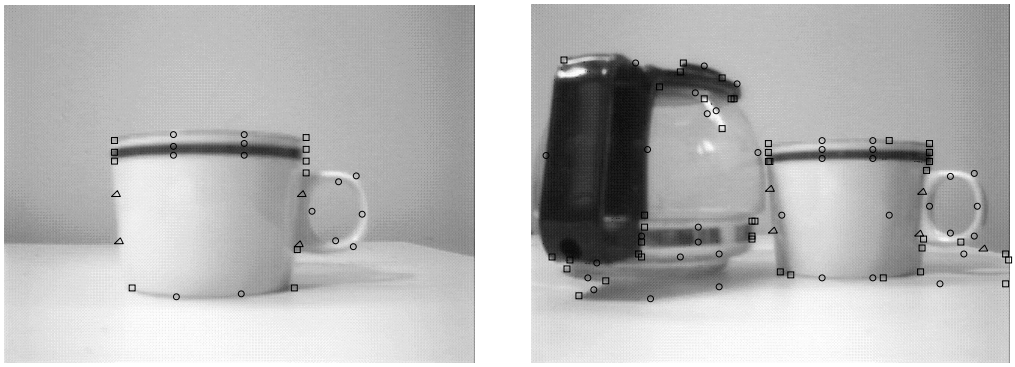

Figura 2.1: Scene con feature inserite manualmente

I risultati ottenuti dall'algoritmo sono piuttosto buoni, con un'alta percentuale di match corretti, in particolar modo per quanto riguarda grafi sottoposti a rumore strutturale e perturbazioni basse o nulle. Inoltre l'algoritmo sembra funzionare in modo non ottimale per grafi a basso grado di connettività. Ad ogni modo Graduated Assignment risulta uno dei migliori algoritmi di graph matching per precisione e velocità e per questo risulta essere uno dei benchmark di riferimento in letteratura.

2.2 Alcune tecniche probabilistiche per il graph matching

Possiamo trovare in letteratura molti esempi di tecniche per il graph matching basate sull'applicazione della teoria delle probabilità. I primi lavori proposti in questo senso ([34],[35]) adottano un approccio iterativo utilizzando un metodo chiamato rilassamento probabilistico (probabilistic relaxation), considerando solamente attributi associati agli archi e assumendo la presenza di rumore gaussiano. In pubblicazioni seguenti il medesimo approccio verrà arricchito utilizzando il framework Bayesano e sarà esteso per trattare sia attributi sui vertici che sugli archi delle strutture confrontate ([36],[37]).

Più recentemente, in [38], viene presentato uno studio comparativo di diverse strategie di ricerca deterministica in uno spazio discreto basata sulla misura Bayesana di consistenza precedentemente proposta in [37]. In questo studio inoltre viene proposto Tabu search come algoritmo per il graph

matching (si veda [39],[40],[41],[42]). Infine il framework è stato ulteriormente esteso per permettere il matching di strutture relazionali su modelli gerarchici[43]. Tutte queste tecniche si basano in genere sul miglioramento iterativo di una soluzione iniziale e dipendono in modo piuttosto pesante da come tale punto di partenza viene scelto e dalla sua qualità: per questo motivo sono particolarmente indicate qualora la formulazione del problema che si intende aggredire sia tale da permettere di individuare semplicemente una soluzione iniziale non ottimale, ma ragionevole, viceversa si rivelano poco pratiche quando si è costretti a partire da punti arbitrari.

Un altro importante approccio probabilistico è quello basato sull'algoritmo EM (Expectation Maximization), utilizzato normalmente per trovare stime di massima verosimiglianza di parametri in modelli probabilistici. Troviamo due applicazioni di tale approccio in [44] e [45], dove sono presentati due differenti framework EM per due diversi problemi di matching. Un algoritmo per il match approssimato che si focalizza unicamente sulla struttura connettiva dei grafi e non prende in considerazione attributi su archi o vertici viene presentato in [46] ed esteso in [47] . In [48] viene illustrata un'applicazione di EM applicata al riconoscimento di caratteri scritti a mano libera mediante la loro rappresentazione attraverso grafi gerarchici.

Altri esempi di tecniche per il match di grafi pesati o meno basate su un approccio probabilistico si possono trovare nel campo del riconoscimento di regioni facciali in [49], dell'autenticazione biometrica mediante rilassamento probabilistico in [50] e [51], e della predizione della formazione dei ponti disolfurici nel folding proteico in [52].

2.3 Altri approcci al graph matching

In [53] viene presentata un'altra tecnica per il matching fra grafi pesati che segue una via analitica anzichè combinatoria. Il problema viene risolto efficientemente tramite la decomposizione spettrale delle matrici di adiacenza, nel caso di grafi non orientati, e delle matrici Hermitiane derivate dalle matrici di adiacenza, nel caso di ricerca di match di peso massimo fra grafi orientati. Dai risultati sperimentali presentati sembra che questo algoritmo funzioni adeguatamente a patto che i due grafi sui quali opera siano sufficientemente simili fra di loro. Inoltre, una pesante limitazione di questo approccio è che, poichè ha bisogno di lavorare su matrici di adiacenza quadrate, può essere applicato unicamente a problemi di match fra grafi delle medesime dimensioni.

In [54] viene infine proposto un approccio per il matching fra grafi pesati basato sulla programmazione lineare. Il problema di ricerca del match di

peso massimo viene in un primo passo formulato come un problema di ottimizzazione quadratica, il quale a sua volta viene trasformato in un problema di programmazione lineare. Quest'ultimo problema viene risolto mediante un algoritmo basato sulla tecnica del simplesso. Una volta massimizzato il problema lineare la soluzione reale trovata viene discretizzata portando i suoi elementi nell'insieme $\{0, 1\}$ attraverso il metodo Ungherese, un noto algoritmo di ottimizzazione combinatoria per il problema dell'assegnamento proposto da Kuhn [55]. Complessivamente la tecnica presentata ha una complessità polinomiale, anche se piuttosto alta, in quanto risulta essere nell'ordine di $O(n^6 L)$ dove n è la dimensione dei grafi per i quali si cerca un match e L la dimensione del problema di ottimizzazione lineare. L'algoritmo viene confrontato sperimentalmente sia con una tecnica basata sulle trasformate polinomiali simmetriche (SPT) [56] precedentemente proposta dallo stesso autore, che con la tecnica basata sulla decomposizione spettrale illustrata in [53]. In entrambi i casi questo algoritmo si rivela superiore in termini di peso totale del match trovato.

Capitolo 3

Una formulazione continua

La ricerca di un match massimo fra due grafi può essere ricondotta alla ricerca di una clique massima all'interno di un grafo derivato da entrambi detto *grafo di associazione*. Questo tipo di riduzione è particolarmente interessante in quanto, pur appartenendo alla classe di complessità NP, la ricerca della clique massima in un grafo è un problema estremamente studiato, sia nell'abito dei grafi pesati che non pesati.

In questo capitolo vedremo come costruire un grafo di associazione e come cercare in esso una clique massima mediante massimizzazione di un problema quadratico. Successivamente vedremo come estendere questo framework a grafi con vertici pesati, passando quindi da una definizione di massimalità della clique legata alla sua cardinalità ad una basata sul peso complessivo dei vertici coinvolti.

Dopo aver visto la naturale estensione del concetto di clique massima a grafi con archi pesati vedremo come applicarla al nostro problema, proponendo una formulazione del grafo di associazione che tenga conto delle similarità di vertici ed archi.

Infine proporremo un'ulteriore formulazione basata sulla creazione di un grafo di associazione fra gli archi anzichè fra i vertici.

3.1 Definizioni

Un grafo con attributi è una quadrupla $G = \{V, E, \alpha, \beta\}$ dove $V = \{1, 2...n\}$ è l'insieme dei vertici, $E \subseteq V \times V$ l'insieme degli archi, $\alpha : V \to C_\alpha$ è una funzione che associa a ciascun vertice in V un vettore in un determinato spazio vettoriale C_α ed infine $\beta : E \to C_\beta$ è una funzione che associa a ciascun arco in E un vettore in un determinato spazio vettoriale C_β.

Dati due grafi con attributi $G_1 = \{V_1, E_1, \alpha_1, \beta_1\}$ e $G_2 = \{V_2, E_2, \alpha_2, \beta_2\}$

possiamo definire una funzione $\sigma : C_{\alpha 1} \times C_{\alpha 2} \rightarrow \mathbb{R}^+$ che associa ad ogni coppia di vettori di attributi nel codominio di α_1 e α_2 un valore positivo di similarità. Allo stesso modo possiamo definire una funzione $\rho : C_{\beta 1} \times C_{\beta 2} \rightarrow \mathbb{R}^+$ che associa ad ogni coppia di vettori di attributi nel codominio di β_1 e β_2 un valore positivo di similarità.

Dati due sottoinsiemi di vertici $H_1 \subset V_1$ e $H_2 \subset V_2$ qualsiasi biiezione $\phi : H_1 \rightarrow H_2$ è un isomorfismo fra sottografi se e solo se conserva l'adiacenza dei vertici, ovvero $\forall (u, v) \in H_1 \times H_2, (u, v) \in E_1 \Leftrightarrow (\phi(u), \phi(v)) \in E_2$.

Dato un isomorfismo fra sottografi di grafi con attributi $\phi : H_1 \rightarrow H_2$ e le due funzioni di similarità fra attributi σ e ρ possiamo definire una funzione $\omega : H_1 \times H_2 \rightarrow \mathbb{R}^+$ che misuri la similarità fra due vertici associati da ϕ come

$$\omega(u_1, u_2) = \sigma(\alpha_1(u_1), \alpha_2(u_2)) \tag{3.1}$$

inoltre, poichè ogni isomorfismo guidato dai vertici ne induce uno sugli archi, possiamo definire anche una funzione $\gamma : (H_1 \times H_2) \times (H_1 \times H_2) \rightarrow \Re^+$ che misuri la similarità fra due archi associati da ϕ come

$$\gamma((u_1, u_2), (v_1, v_2)) = \begin{cases} \rho(\beta_1(u_1, v_1), \beta_2(u_2, v_2)) & \text{se } (u_1, v_1) \in E_1, (u_2, v_2) \in E_2 \\ 0 & \text{se } (u_1, v_1) \notin E_1, (u_2, v_2) \notin E_2 \\ \text{indefinita} & \text{altrimenti} \end{cases} \tag{3.2}$$

Si noti che in pratica, se ϕ è un isomorfismo, la funzione γ non sarà mai indefinita in quanto un isomorfismo è tale proprio se conserva la relazione di adiacenza fra i vertici coinvolti.

A questo punto per ogni isomorfismo $\phi : H_1 \rightarrow H_2$ possiamo definire una misura di similarità complessiva fra archi e vertici come

$$S(\phi) = \sum_{u \in H_1} \omega(u, \phi(u)) + \sum_{u \in H_1} \sum_{v \in H_1} \gamma((u, \phi(u)), (v, \phi(v))) \tag{3.3}$$

Nel seguito quando parleremo di match di similarità massima intenderemo un isomorfismo ϕ che rende massima (3.3).

3.2 Costruzione di un grafo di associazione

L'idea del grafo di associazione applicata al graph matching è stata introdotta per la prima volta da Ambler in [57].

Dati due grafi $G_1 = \{V_1, E_1\}$ e $G_2 = \{V_2, E_2\}$ con $V_1 = \{1, 2...n\}$, $V_2 = \{1, 2...m\}$ e $E_1 \subseteq V_1 \times V_1$, $E_2 \subseteq V_2 \times V_2$, un grafo di associazione è un grafo

ausiliario $G_a = \{V_a, E_a\}$ dove $V_a = V_1 \times V_2$ e $E_a \subset V_a \times V_a$. In particolare dati $u_1, v_1 \in V_1$ e $u_2, v_2 \in V_2$ si ha che

$$((u_1, u_2), (v_1, v_2)) \in E_a \iff \begin{cases} u_1 \neq v_1 \\ u_2 \neq v_2 \\ (u_1, v_1) \in E_1 \Leftrightarrow (u_2, v_2) \in E_2 \end{cases} \tag{3.4}$$

Dal punto di vista intuitivo l'insieme di vertici del grafo di associazione V_a rappresenta tutte le possibili associazioni dei vertici di G_1 nei vertici di G_2, quindi qualunque suo sottoinsieme può rappresentare un match fra G_1 e G_2. Di tutti i possibili match che si possono costruire (che sono il numero di sottoinsiemi di V_a, cioè $2^{|V_a|}$) sono di nostro interesse solamente quelli che rappresentano effettivamente isomorfismi fra sottografi, ovvero che proiettano ogni vertice di G_1 in al più un vertice di G_2, che non proiettano mai due vertici di G_1 nello stesso vertice di G_2 e che conservano l'adiacenza dei vertici.

Tali condizioni sono rappresentate esattamente dalle tre condizioni poste perchè due vertici di V_a possano appartenere a E_a. In effetti due vertici di V_a che rispettano le condizioni suddette sono detti *compatibili* ed il grafo G_a presenta archi che connettono associazioni a due a due compatibili.

In questo senso qualunque clique in G_a raccoglie un insieme di associazioni tutte a due a due compatibili e pertanto rappresenta un corretto isomorfismo fra G_1 e G_2. Rispettivamente le clique massimali rappresenteranno isomorfismi massimali e le clique massime isomorfismi massimi. Sfortunatamente la ricerca di clique massimali è un problema in classe di complessità NP, pertanto è necessario individuare delle strategie approssimate per risolverlo: a questo scopo potrà essere utile dare una formulazione continua del medesimo problema, rendendoci così in grado di utilizzare tecniche di ottimizzazione continue.

3.3 Formulazione continua di Motzkin-Straus

Motzkin e Straus in [1] propongono una riduzione del problema della clique massima, di natura discreta, a quello della massimizzazione di una funzione quadratica in un dominio continuo.

Sia $G = \{V, E\}$ un grafo non pesato e non diretto, ovvero per il quale vale la proprietà $\forall u, v \in V, (u, v) \in E \Leftrightarrow (v, u) \in E$. Sia A la matrice di adiacenza di G, ovvero una matrice quadrata di lato $|V|$ dove $a_{uv} = 1 \Leftrightarrow (u, v) \in E$ (si noti che tale matrice risulta essere simmetrica in base alla proprietà che rende il grafo non diretto). Sia $\mathbf{x}$ un generico vettore di dimensione $|V|$

appartenente al simplesso standard nello spazio euclideo a $|V|$ dimensioni $\mathbb{R}^{|V|}$, ovvero all'insieme

$$\Delta = \{\mathbf{x} \in \mathbb{R}^{|V|} | \sum_{i=1}^{|V|} x_i = 1\} \tag{3.5}$$

Il teorema di Motzkin-Straus prova che se $\mathbf{x}^*$ è un massimizzatore globale della seguente funzione quadratica

$$g(\mathbf{x}) = \mathbf{x}^t A \mathbf{x} \tag{3.6}$$

il numero di clique ω del grafo G è legato ad esso tramite l'equazione

$$\omega = \frac{1}{1 - g(\mathbf{x}^*)} \tag{3.7}$$

Inoltre sottoinsieme S di vertici di G rappresenta una clique massima se e solo se il suo vettore caratteristico $\mathbf{C}$ è un massimizzatore globale di 3.6 in Δ, dove il vettore caratteristico di un insieme di vertici $S \in G$ è definito come

$$C_i = \begin{cases} 0 & \text{se } i \notin S \\ \frac{1}{|S|} & \text{se} i \in S \end{cases} \tag{3.8}$$

Purtroppo esistono anche massimizzatori di 3.6 che non sono vettori caratteristici e pertanto non possono essere utilizzati direttamente per ricavare informazioni sulle clique massime, o comunque non sui vertici in esse coinvolte. Tali massimizzatori, definiti *soluzioni spurie* sono stati osservati empiricamente da Pardalos e Phillips [58] e recentemente formalizzati da Pelillo e Jagota [59].

Il problema delle soluzioni spurie è stato recentemente risolto da Bomze [60] proponendo la seguente versione regolarizzata della 3.6:

$$\breve{g}(\mathbf{x}) = \mathbf{x}^t A \mathbf{x} + \frac{1}{2}\mathbf{x}^t \mathbf{x} \tag{3.9}$$

La quale è ricavabile dalla 3.6 sostituendo la matrice di adiacenza A di G con

$$\breve{A} = A + \frac{1}{2} I \tag{3.10}$$

Dove I è la matrice identità, ovvero una matrice quadrata dello stesso lato di A con tutti gli elementi in diagonale principale ad 1 e tutti gli elementi al di fuori di essa a 0.

Questa nuova formulazione della funzione obiettivo $\check{g}(x)$ mantiene la caratteristica principale della formulazione di Motzkin-Straus, ovvero che ogni sottoinsieme S di vertici di G rappresenta una clique massima se e solo se il suo vettore caratteristico $\mathbf{C}$ è un massimizzatore globale di $\check{g}(x)$ in Δ (anche se in questo caso per ovvi motivi $\omega = \frac{1}{2(1-\check{g}(C)}$).

Inoltre in questo caso si dimostra anche che tutti i massimizzatori di $\check{g}(x)$ in Δ sono nella forma di qualche vettore caratteristico di un certo sottoinsieme S di V e questo vale sia per i massimizzatori globali, che individuano clique massime, sia per quelli locali, che individuano semplicemente clique massimali, eliminando quindi la possibilità di avere soluzioni spurie.

Si noti che date le regole di adiacenza dei vertici del grafo di associazione fornite al paragrafo precedente questo avrà sempre la forma di un grafo non diretto, di conseguenza il teorema di Motzkin-Straus, nella sua forma libera da soluzioni spurie, potrà essere utilizzato per ridurre la ricerca di una clique massima a quella di un massimizzatore in 3.9: come vedremo esistono alcune tecniche continue piuttosto efficienti per raggiungere tale scopo in un tempo accettabile.

3.4 Estensione a grafi con vertici pesati

Il teorema di Motzkin-Straus presentato al paragrafo 3.3 mette in relazione clique di cardinalità massima (o massimale) con massimi globali (o locali) della funzione obiettivo proposta. Un recente lavoro di Gibbons [2] generalizza il framework al caso dei grafi con vertici pesati, definendo una formulazione estesa che permette di associare clique di peso massimo (o massimale) a minimizzatori globali (o locali) di una nuova funzione obiettivo.

Il primo passo è la trasposizione del problema quadratico di Motzkin-Straus sotto forma di un nuovo problema di minimizzazione, ovvero, dato un generico grafo $G = (V, E)$ e considerando la funzione

$$f(x) = \mathbf{x}^t(I + A_{\check{G}})\mathbf{x} \tag{3.11}$$

dove $A_{\check{G}}$ è la matrice di adiacenza del grafo complementare $\check{G}$, è facile dimostrare che se x^* è un minimizzatore globale di 3.11 in Δ abbiamo che

$$\omega(G) = \frac{1}{f(\mathbf{x}^*)} \tag{3.12}$$

In quanto 3.11 non è altro che una diversa formulazione del problema quadratico originale.

Consideriamo ora la seguente classi di matrici quadrate $n \times n$ generate dal grafo $G = (V, E)$:

$$\mathbb{M}(G) = \{(b_{ij})i, j \in V : b_{ij} \geq \frac{b_{ii} + b_{jj}}{2} \text{ se } (i,j) \notin E,\ b_{ij} = 0 \text{ altrimenti } \} \tag{3.13}$$

Aggiungiamo ora un vettore di pesi w al grafo G, dove $w_i \in \mathbb{R}$ è il peso del vertice i e consideriamo la nuova classe di matrici quadrate $n \times n$ generate dal grafo pesato $G = (V, E, w)$:

$$\mathbb{M}(G, w) = \{B = (b_{ij})i, j \in V \in \mathbb{M}(\mathbb{G}) : b_{ii} = \frac{1}{w_i} \text{ e } b_{ij} = b_{ji} \text{ per ogni } i, j\} \tag{3.14}$$

A questo punto, presa una qualunque matrice $B \in \mathbb{M}(G, w)$ e la nuova funzione quadratica

$$f(\mathbf{x}) = \mathbf{x}^t B \mathbf{x} \tag{3.15}$$

è possibile dimostrare, utilizzando una tecnica suggerita da Lovász in [61], che per qualsiasi matrice $B \in \mathbb{M}(G, w)$ abbiamo che

$$\omega(G, w) = \frac{1}{f(\mathbf{x}^*)} \tag{3.16}$$

dove $\mathbf{x}^* \in \Delta$ è un minimizzatore globale della funzione 3.15 e $\omega(G, w)$ è il peso della clique di peso massimo contenuta in G.

Inoltre si può osservare che un sottoinsieme di vertici $S \subset V$ del grafo pesato originale è una clique di peso massimo se e solo se il suo vettore caratteristico pesato $\mathbf{x}^{S,w}$ è un minimizzatore globale di 3.15, dove $\mathbf{x}^{S,w}$ è il vettore la cui i-esima componente è calcolata come

$$\mathbf{x}_i^{S,w} = \frac{w_i}{\sum_{j \in S} w_j} \tag{3.17}$$

Infine se osserviamo nuovamente la funzione 3.11 notiamo che la matrice $I + A_{\check{G}}$ appartiene in effetti a $\mathbb{M}(G, \mathbf{e})$ dove $\mathbf{e}$ è un vettore con tutti gli elementi a 1, quindi il teorema di Motzkin-Straus risulta essere un caso particolare del risultato precedente.

Come accadeva per il caso non pesato anche in queste condizioni è possibile individuare minimizzatori di 3.15 che non sono nella forma di vettori caratteristici e che quindi non possono essere utilizzati per risalire ad una clique di peso massimo. Tali soluzioni spurie sono state recentemente studiate e caratterizzate da Bomze [62] che ha introdotto una versione regolarizzata per sfuggire al problema esattamente come nel caso non pesato [63].

Anzichè utilizzare la classe $\mathbb{M}(G, w)$ per scegliere la matrice B nel problema 3.15 si definisce una nuova classe di matrici $n \times n$ generate dal generico grafo $G = (V, E)$

$$\mathbb{C}(G) = \{(c_{ij})i, j \in V : c_{ij} \geq b_{ii} + b_{jj} \text{ se } (i, j) \notin E,\ c_{ij} = 0 \text{ altrimenti } \} \tag{3.18}$$

Che aggiungendo il vettore dei pesi w diventano la classe

$$\mathbb{C}(G, w) = \{C = (c_{ij})i, j \in V \in \mathbb{C}(\mathbb{G}) : c_{ii} = \frac{1}{2w_i} \text{ e } c_{ij} = c_{ji} \text{ per ogni } i, j\} \tag{3.19}$$

In queste condizioni ogni vettore $\mathbf{x}^*$ appartenente al simplesso standard Δ è minimizzatore globale (o locale) di 3.15 se e solo se è nella forma $\mathbf{x}^* = \mathbf{x}^{S,w}$ con $S \in E$ clique massima (o massimale) di G. Inoltre ogni minimizzatore globale o locale di 3.15 è stretto.

La classe $\mathbb{C}(G, w)$, detta classe di Comtet, è isomorfa all'ortante positivo nello spazio euclideo in $\binom{n}{2} - |E|$ dimensioni. Tale classe rappresenta un cono con apice corrispondente alla matrice $C(G, w)$ i cui elementi sono

$$c_{ij}(w) = \begin{cases} \frac{1}{2w_i} & \text{se } i = j \\ \frac{1}{2w_i} + \frac{1}{2w_j} & \text{se } i \neq j \text{ and } (i, j) \notin E \\ 0 & \text{altrimenti} \end{cases} \tag{3.20}$$

3.5 Estensione a grafi con archi pesati

Recentemente Pelillo e Pavan hanno proposto una generalizzazione di max-clique ai grafi con archi pesati con applicazioni ai problemi di clustering [64]. In termini intuitivi un cluster può essere definito come un sottoinsieme di un insieme più grande che raggruppa elementi che presentano un'alta similarità fra di loro ed una bassa similarità con tutti gli altri elementi dell'insieme esterni al cluster. Rappresentando gli elementi degli insiemi come vertici e le loro reciproche similarità come archi pesati otteniamo grafi completi come quelli portati ad esempio in figura 3.1.

In questo caso le proprietà del cluster possono essere interpretate come il fatto che ogni cluster dovrebbe essere composto da vertici collegati fra di loro con archi di peso alto e collegati con tutti gli altri vertici con archi di peso basso.

Per dare una definizione formale di questo concetto abbiamo bisogno di proiettare in qualche modo il peso associato a ciascun arco sui vertici che

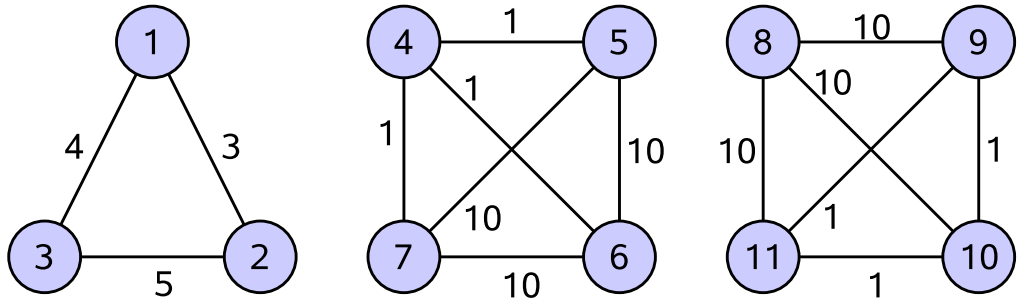

Figura 3.1: Alcuni grafi con archi pesati

faranno parte dei vari cluster in modo da poter determinare come individuare cluster che massimizzino il peso associato dei vertici in essi contenuti e minimizzino quello dei vertici esclusi.

Consideriamo un grafo $G = (V, E, w)$, privo di self-loop, dove in questo caso $w : E \to \mathbb{R}^+$ è una funzione che assegna un peso reale a ciascun arco e definiamo la sua matrice di adiacenza $A_G = (a_{ij})$ come

$$a_{ij} = \begin{cases} w(i,j) & \text{se } (i,j) \in E \\ 0 & \text{altrimenti} \end{cases} \tag{3.21}$$

Prima di tutto per ogni sottoinsieme di vertici $S \subset V$ ed ogni vertice $i \in V$ definiamo il grado medio pesato di i rispetto ad S come

$$awdeg_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij} \tag{3.22}$$

Notiamo che a causa dell'assenza di self-loop $awdeg_{\{i\}} = 0$ per ogni $i \in S$. Inoltre definiamo per ogni $j \notin S$ la seguente misura di similarità

$$\phi_S(i,j) = a_{ij} - awdeg_S(i) \tag{3.23}$$

Notiamo che $\phi_{\{i\}}(i,j) = a_{ij}$ per ogni $i, j \in V$ con $i \neq j$. Intuitivamente $\phi_S(i,j)$ misura la similarità fra i nodi j e i relativamente alla similarità media fra il nodo i e gli altri nodi contenuti in S. Si noti che tale similarità può assumere sia valori positivi che negativi.

A questo punto siamo nelle condizioni di formalizzare l'idea di "proiezione" dei pesi dagli archi ai vertici, mediante la seguente definizione ricorsiva

$$w_S(i) = \begin{cases} 1 & \text{se } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(i,j) w_{S \setminus \{i\}}(j) & \text{altrimenti} \end{cases} \tag{3.24}$$

Dove $W_S(i)$ è il peso di $i \in S$ rispetto ad un sottoinsieme di vertici non vuoto $S \subset V$. Inoltre il peso totale di S viene definito come

$$W(S) = \sum_{j \in S} w_S(i) \tag{3.25}$$

Si noti che $w_{\{i,j\}}(i) = w_{\{i,j\}}(j) = a_{ij}$ per tutti gli $i, j \in V$ con $i \neq j$. Si noti anche che $w_S(i)$ può essere calcolata semplicemente come funzione degli archi del sottografo indotto da S. Inoltre i pesi così definiti risultano coerenti con la nozione di cluster finora utilizzata: per esempio, riferendoci al grafo in figura 3.1, vediamo che $w_{\{1,2,3\}}(1) < w_{\{1,2,3\}}(2) < w_{\{1,2,3\}}(3)$. Questo perchè la similarità totale del vertice 1 rispetto a quella complessiva dell'insieme S è inferiore a quella del vertice 2 che a sua volta è inferiore a quella del vertice 3. Notiamo ancora che $w_{\{4,5,6,7\}}(4) < 0$ mentre $w_{\{8,9,10,11\}}(8) > 0$, questo perchè il vertice 4 ha una similarità totale con gli altri vertici del gruppo inferiore alla media del gruppo stesso, mentre il vertice 8, rispetto agli altri vertici del suo gruppo è nella condizione opposta.

A questo punto possiamo definire un Dominant Set (che nella nostra formulazione corrisponde all'idea di cluster) come un sottoinsieme non vuoto di V $S \subset V$ tale da rispettare le seguenti condizioni

- $w_S(i) > 0, \text{per ogni} i \in S$
- $w_S(i) < 0, \text{per ogni} i \notin S$

Inoltre è necessario che sia vero che $W(T) > 0$ per ogni sottoinsieme T di S non vuoto (quest'ultima condizione è un requisito tecnico spiegato in dettaglio in [65]).

Le due condizioni illustrate corrispondono alle due proprietà caratterizzanti un clustering: la prima riguarda l'omogeneità interna, la seconda la disomogeneità esterna.

Per chiarire meglio l'idea, vediamo un esempio illustrato in figura 3.2: in questo caso vediamo che l'insieme $\{1, 2, 3\}$ è dominante, infatti il peso degli archi interni a tale insieme è più grande di quello degli archi che connettono vertici interni a vertici esterni, mentre l'inclusione di altri vertici, collegati da pesi molto inferiori, romperebbe questa coerenza. E' facile infine dimostrare che la nozione di Dominant Set quando portata su grafi con similarità fra i vertici in $\{0, 1\}$ corrisponde alla nozione di clique massima: per questo motivo consideriamo in qualche modo i Dominant Set come una generalizzazione di max-clique.

Come per tutti i problemi combinatori, trovare un Dominant Set in un grafo pesato sugli archi è un problema combinatorio, e pertanto non è pos-

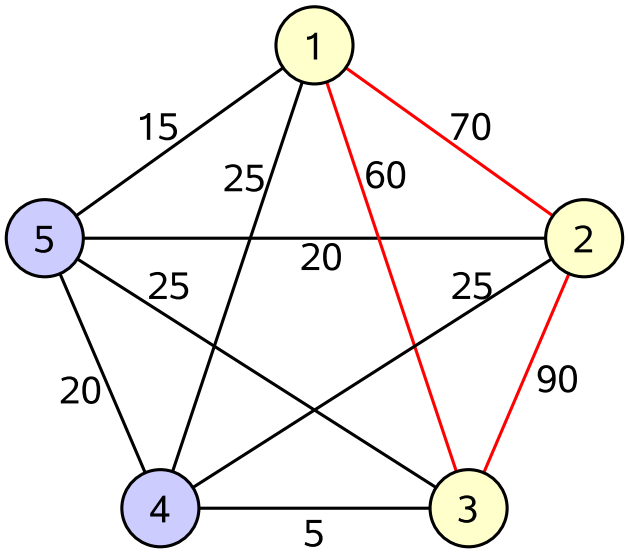

Figura 3.2: Un semplice Dominant Set

sibile generalmente risolverlo in tempo polinomiale. Fortunatamente anche in questo caso è possibile trovare una generalizzazione del teorema di Motzkin-Straus che ci permette di spostare il problema nell'ambito del continuo (e nel capitolo 4 vedremo come trovare eventuali massimizzatori in modo approssimato).

In particolare diamo una nuova definizione di vettore caratteristico, ovvero diciamo che un vettore $\mathbf{x}^S$ appartenente a Δ è un vettore caratteristico del Dominant Set S se il suo peso totale $W(S)$ non è nullo e se le sue componenti assumono la seguente forma

$$\mathbf{x}_i^S = \begin{cases} 0 & \text{se } i \notin S \\ \frac{w_s(i)}{W(S)} & \text{se} i \in S \end{cases} \tag{3.26}$$

Con $\mathbf{x}^S$ appartenente al simplesso standard Δ. E diamo una nuova funzione obiettivo

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} \tag{3.27}$$

Dove A è la matrice di adiacenza del grafo con pesi sugli archi $G = (V, E, w)$. In queste condizioni si dimostra che se S è un insieme dominate di vertici allora il suo vettore caratteristico $\mathbf{x}^S$ è un massimizzatore locale di 3.27. Viceversa se $\mathbf{x}^S$ è un massimizzatore locale di 3.27 allora il suo supporto $\sigma(\mathbf{x}^S)$ è un insieme dominante, a patto che $w_{\sigma(\mathbf{x}^S)\cup\{i\}}(i) = 0$ per ogni $i \notin \sigma(\mathbf{x}^S)$.

Quest'ultima condizione garantisce che si riescano ad evitare anche in questo caso soluzioni spurie.

3.6 Una formulazione basata sui Dominant Set

Nel paragrafo 3.2 abbiamo visto come sia possibile trovare un match fra grafi mediante la ricerca di una clique massima in un grafo di associazione. In 3.4 abbiamo visto come trovare una clique di peso massimo in un grafo pesato sui vertici, quindi in caso di ricerca di match fra grafi pesati sui vertici possiamo semplicemente ricondurre il problema alla ricerca di una clique di peso massimo in un grafo di associazione il cui peso dei vertici sia funzione della similarità fra i vertici associati nel grafo originale. Rimane tuttavia da chiarire come è possibile cercare match in modo efficiente fra grafi con attributi (e quindi con misure di similarità) sia su vertici che su archi.

Nel paragrafo 3.5 abbiamo visto una generalizzazione di max-clique al caso di grafi con archi pesati, che va a corrispondere con la comune nozione di cluster. Una possibile idea potrebbe essere quella di costruire un grafo di associazione pesato sugli archi e cercare in esso un Dominant Set. In questo caso rimarrebbe comunque da capire come introdurre l'informazione relativa alla similarità fra i vertici più altri dettagli che vedremo in seguito.

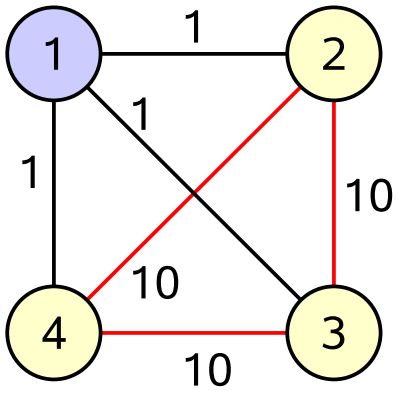

Figura 3.3: Esempio di Dominant Set non corrispondente a clique di peso massimo

Prima di proseguire è però necessario fare un'osservazione: nella formulazione del grafo di associazione che andremo a proporre in qualche modo un match di peso massimo fra i grafi originali corrisponderà ad una clique di

peso massimo (sugli archi) del grafo di associazione. Tuttavia i Dominant Set non hanno come obiettivo individuare insiemi di vertici che massimizzano il peso totale degli archi coinvolti, ma piuttosto che rendano grande la coerenza interna a dispetto di quella esterna all'insieme.

Questo è in qualche modo in contrasto con il nostro obiettivo: infatti se osserviamo l'esempio di figura 3.3 notiamo che l'insieme $\{2, 3, 4\}$ è dominante, tuttavia è un sottoinsieme della clique di massimo peso complessivo, che corrisponde all'insieme $\{1, 2, 3, 4\}$. Questa osservazione rende in qualche modo l'approccio basato sui Dominant Set meno adatto a risolvere il nostro problema, tuttavia è anche necessario considerare che quando si cerca un match fra grafi che hanno effettivamente un sottografo in comune questo presenterà un'ottima coerenza interna fra i pesi di vertici ed archi associati, anche in presenza di blando o moderato rumore. In questo senso la nozione di Dominant Set, anche se non corrisponde perfettamente in linea teorica con il nostro obiettivo, vi si avvicina molto ed è quindi giustificato fare delle verifiche sperimentali per vedere se, nei casi pratici, si tratti effettivamente di un'euristica efficace. Vediamo quindi come applicare questo framework al nostro problema.

Ricordando le definizioni di grafi con attributi dati in 3.1 e la caratterizzazione dell'insieme degli archi del grafo di associazione data in 3.2, possiamo pensare di costruire un grafo di associazione i cui elementi della matrice di adiacenza $A = (a_{ij})$ abbiano i seguenti valori

$$a_{(u_1,u_2)(v_1,v_2)} = \begin{cases} \lambda_1 \omega(u_1, u_2) & \text{se } (u_1, u_2) = (v_1, v_2) \\ \gamma((u_1, u_2), (v_1, v_2)) & \text{se } (u_1, v_1) \in E_1, (u_2, v_2) \in E_2 \\ \lambda_2 & \text{se } (u_1, v_1) \notin E_1, (u_2, v_2) \notin E_2 \\ 0 & \text{altrimenti} \end{cases} \tag{3.28}$$

Dove $u_1, v_1 \in V_1$ e $u_2, v_2 \in V_2$ ed il grafo di associazione è il consueto grafo con insiemi dei vertici $V \subset V_1 \times V_2$ e insieme degli archi $E \subset V \times V$.

Una volta creata la matrice A il nostro obiettivo sarà quello di trovare in vettore $\mathbf{x}$ in Δ che massimizzi la funzione quadratica

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A \mathbf{x} \tag{3.29}$$

Vediamo in dettaglio il significato delle varie condizioni che costruiscono gli elementi della matrice di adiacenza e di conseguenza il peso degli archi nel grafo di associazione.

- Le ultime tre condizioni hanno a che fare con l'esistenza o meno di archi nel grafo di associazione: come nella formulazione classica gli archi

esistono (ovvero hanno un peso non nullo) se e solo se le coppie di vertici dei grafi originali associate sono compatibili, ovvero sono entrambe adiacenti o non adiacenti nei rispettivi grafi originali. Distinguiamo però i due casi;

- La seconda condizione si riferisce al caso in cui le coppie di vertici associati siano adiacenti: in questo caso il peso associato all'arco nel grafo di associazione è pari alla similarità dei due archi, secondo le definizioni di similarità date in 3.1;

- La terza condizione copre il caso in cui i vertici associati siano compatibili, ma non adiacenti: in questo caso non è possibile calcolare una similarità in quanto gli archi non sono presenti. La soluzione è quella di assegnargli un certo valore λ_2. Tale valore non deve essere troppo basso, in quanto viceversa il framework dei Dominant Set tenderà ad escludere insiemi che contengono vertici non adiacenti (che avranno bassa compatibilità nel grafo di associazione). L'ideale sarà determinare sperimentalmente il valore ottimale di tale parametro, come vedremo nel capitolo 5;

- La prima condizione serve a considerare anche la similarità fra i vertici coinvolti nel match: infatti nella formulazione classica il grafo di associazione non presenta self-loop, anche perchè valori non nulli sulla diagonale tenderebbero ad individuare massimizzatori del problema quadratico erroneamente spostati verso i vertici del simplesso. Proprio per evitare questo fenomeno, ma nel contempo non rinunciare ad introdurre un fattore che tenga conto dei vertici, è stato introdotto un fattore di scala λ_1 che dovrà essere determinato sperimentalmente individuando un valore che tenga sufficientemente conto della similarità fra i vertici senza per questo forzare le soluzioni verso i bordi del simplesso.

Al contrario delle altre formulazioni questo approccio basato sui Dominant Set non garantisce che la massimizzazione della funzione obiettivo corrisponda ad un vettore il cui supporto corrisponda effettivamente ad una clique di peso massimo nel grafo di associazione e di conseguenza ad un match di peso massimo, tuttavia non è escluso che fornisca un'euristica sufficientemente buona da permettere di individuare, in determinate condizioni, soluzioni di una certa qualità.

Questa osservazione ha particolarmente senso se consideriamo che, almeno nei nostri test, andremo a cercare match fra grafi uguali che hanno subito delezioni e leggere perturbazioni, per i quali cioè esiste effettivamente un

match con similarità marcate fra vertici ed archi coinvolti. In questo senso l'approccio basato sui Dominant Set potrebbe trovare con relativa facilità questi assegnamenti fortemente coerenti in quanto sarebbero, proprio in virtù della loro coerenza interna, anche insiemi dominati.

C'e' infine da notare che non esiste una naturale "gerarchia" nei Dominant Set, di conseguenza non vi è nemmeno garanzia che, applicando opportune tecniche di ricerca nel simplesso, si trovi in qualche modo un insieme dominante "massimale", ma al più se ne troverà quello massimamente "coerente".

3.7 Una formulazione basata sul grafo di associazione fra gli archi

La formulazione basata sui Dominant Set presentata nel paragrafo 3.6 mette in lassa correlazione gli insiemi dominanti in un grafo di associazione con i match di peso massimo fra i grafi originali. Come abbiamo notato la formulazione non è in sè esatta, ma potrebbe dimostrarsi un'approssimazione di buona qualità.

Esiste tuttavia un approccio diverso che garantisce la corrispondenza fra i massimizzatori del problema quadratico associato e i match di peso massimo fra i grafi considerati. Questo purtroppo avviene al prezzo di un notevole aumento della complessità del problema in quanto, come vedremo, la matrice del problema quadratico passa da un lato di $O(n^2)$ con n la cardinalità massima dei due grafi, ad un lato di dimensioni $O(n^4)$, portando la complessità totale di un'interazione al ragguardevole traguardo di $O(n^8)$.

Questo approccio, molto semplicemente, si basa sulla costruzione di un grafo di associazione fra gli archi, anzichè fra i vertici dei due grafi originali, da cui la maggiore complessità computazionale, ricordando che in generale gli archi di un grafo possono essere anche il quadrato del numero dei vertici (considerando grafi orientati completi e con self-loop).

Il punto chiave è che tale grafo di associazione sarà dotato di pesi solo sui suoi vertici, essendo ciascuno pesato in base alla similarità degli archi o dei vertici che associa. Di conseguenza sarà possibile applicare il framework presentato nel paragrafo 3.4, il quale associa in modo esatto i minimizzatori globali (e locali) della sua funzione obiettivo alle clique di peso massimo (e massimale) di un grafo a vertici pesati. Ricordando che le clique di peso massimo (o massimale) nel grafo di associazione corrispondono a match di peso massimo (o massimale) fra i grafi originali, questa tecnica ci permetterà

di ottenere una formulazione che associ quindi minimizzatori della funzione obiettivo proposta a match massimi fra i grafi originali.

Naturalmente vale la pena di osservare che il fatto di aver prodotto una formulazione con tali caratteristiche non garantisce affatto che tali minimizzatori (e di conseguenza i match di peso massimo) possano essere trovati in modo esatto rapidamente, anzi se questo accadesse in qualche modo si dimostrerebbe che il problema del graph matching può essere risolto in tempo polinomiale, il che sarebbe quanto meno sorprendente.

Vediamo in dettaglio come è definito il nostro grafo di associazione. Come anticipato avremo un grafo pesato sui vertici $G_a = (V_a, E_a, w)$ con $w : V_a \to \mathbb{R}$ la funzione peso.

L'insieme dei vertici V_a sarà costituito sia dalle associazioni fra i vertici dei due grafi originali, in questo caso rappresentate da self-loop introdotti artificialmente con attributi pari a quelli dei vertici originali stessi[1], sia dalle associazioni fra gli archi, ottenendo:

$$V_a = E_1 \times E_2 \cup \{(u,v) \in V_1 \times V_1 | u = v\} \times \{(u,v) \in V_2 \times V_2 | u = v\} \quad (3.30)$$

L'insieme degli archi, come al solito è un sottoinsieme di $V_a \times V_a$ dove due vertici di V_a sono adiacenti (ovvero rappresentati da un elemento in E_a se e solo se sono compatibili. Nel caso classico la compatibilità di due vertici del grafo di associazione è data semplicemente dalla compatibilità dell'adiacenza dei vertici associati nei grafi originali, in questo caso abbiamo bisogno di una nozione di compatibilità un po' più estesa. Definiamo quindi E_a come

$$\begin{aligned} E_a = \quad \{ \quad & (((u_1,v_1),(u_2,v_2)),((w_1,z_1),(w_2,z_2))) \in V_a \times V_a | \\ & u_1 = w_1 \Leftrightarrow u_2 = w_2, \\ & v_1 = z_1 \Leftrightarrow v_2 = z_2, \\ & u_1 = z_1 \Leftrightarrow u_2 = z_2, \\ & u_1 = v_1 \Leftrightarrow w_2 = z_2, \\ & u_1 = v_1 \wedge w_1 = z_1 \Rightarrow \\ & (u_1,w_1) \in E_1 = (u_2,w_2) \in E_2 \wedge (w_1,u_1) \in E_1 = (w_2,u_2) \in E_2 \\ \} & \end{aligned} \quad (3.31)$$

Vediamo in dettaglio il significato dei vincoli che abbiamo posto perchè un elemento di $V_a \times V_a$ appartenga ad E_a:

[1] Ricordiamo che i grafi originali sono privi di self-loop, quindi questa operazione è lecita e non confligge con la struttura dei grafi originali

- I primi quattro vincoli corrispondono alle condizioni di compatibilità fra gli archi. Ovvero consideriamo due associazioni fra archi compatibili, cioè connessi nel grafo di associazione solo se vengono garantite le seguenti condizioni:
  - Due archi che hanno lo stesso vertice di origine in G_1 devono essere mappati in archi con lo stesso vertice di origine in G_2
  - Due archi che hanno lo stesso vertice di destinazione in G_1 devono essere mappati in archi con lo stesso vertice di destinazione in G_2
  - Due archi il cui vertice di destinazione del primo corrisponde al vertice di origine del secondo in G_1 devono essere mappati in archi per i quali valga la stessa proprietà in in G_2
  - Due archi il cui vertice di destinazione del secondo corrisponde al vertice di origine del primo in G_1 devono essere mappati in archi per i quali valga la stessa proprietà in in G_2
- Il quinto vincolo corrisponde alle condizioni di compatibilità fra vertici, ed infatti viene applicato quando si mappa un vertice, associato ad un self-loop implicito in un altro vertice. Si richiede, come nella formulazione classica, che vertici adiacenti siano mappati in vertici adiacenti e vertici non adiacenti in vertici non adiacenti.

Si noti che la combinazione delle condizioni di vincolo appena espresse è sufficiente ad impedire che vengano mappati vertici in archi e viceversa.

La funzione che assegna un peso ad ogni vertice del grafo di associazione è piuttosto semplice. Ricordando le funzioni di similarità fra vertici e archi di grafi con attributi introdotte in 3.1, possiamo definire la seguente funzione peso $w : V_a \to \mathbb{R}$:

$$w((u_1, v_1), (u_2, v_2)) = \begin{cases} \omega(u_1, u_2) & \text{se } u_1 = v_1 \wedge u_2 = v_2 \\ \gamma((u_1, u_2), (v_1, v_2)) & \text{altrimenti} \end{cases} \tag{3.32}$$

Che significa in pratica assegnare un peso pari alla similarità fra i vertici dei grafi originali per i vertici del grafo di associazione che rappresentano associazioni fra vertici ed un peso pari alla similarità fra gli archi dei grafi originali per i vertici del grafo di associazione che rappresentano associazioni fra archi. Si noti che per come sono definiti V_a ed E_a rispettivamente in 3.30 e 3.31 non avremo mai associazioni fra archi assenti e quindi, a differenza di quanto accadeva nella formulazione basata sui Dominant Set, non avremo bisogno di definire un valore di similarità fra di essi. Inoltre questo fatto

garantisce che γ sia sempre definita quando applicata a vertici del nostro grafo di associazione.

A questo punto siamo in grado di applicare i risultati presentati al paragrafo 3.4. Prima di tutto dobbiamo definire una matrice di Comtet che possa essere utilizzata per formulare il problema quadratico da minimizzare. Dato il grafo di associazione descritto ed i pesi definiti, la nostra matrice potrà essere $A = (a_{((u_1,v_1),(u_2,v_2))((w_1,z_1),(w_2,z_2))})$ così definita:

$$a_{((u_1,v_1),(u_2,v_2))((w_1,z_1),(w_2,z_2))} =$$

$$= \begin{cases} \frac{1}{2\omega(u_1,u_2)} & \text{se } u_1 = v_1 = w_1 = z_1 \wedge u_2 = v_2 = w_2 = z_2 \\ \frac{1}{2\gamma((u_1,u_2),(v_1,v_2))} & \text{se } u_1 = w_1, v_1 = z_1, u_2 = w_2, v_2 = z_2 \\ \frac{1}{2w((u_1,v_1),(u_2,v_2))} + \frac{1}{2w((w_1,z_1),(w_2,z_2))} & \text{se } ((u_1,v_1),(u_2,v_2))((w_1,z_1),(w_2,z_2)) \notin E_a \\ 0 & \text{altrimenti} \end{cases} \tag{3.33}$$

Il che, in accordo con quanto osservato nel paragrafo 3.4, corrisponde ad assegnare il reciproco del doppio del peso assegnato al vertice nel caso dei vertici che associano archi o vertici dei grafi originali, un peso nullo in corrispondenza degli archi presenti nel grafo di associazione e una penalità appropriata in corrispondenza delle coppie di vertici non adiacenti nel grafo di associazione.

Una volta definita la matrice A, il problema di trovare il match di peso massimo fra i grafi originali si riduce a quello di trovare il vettore $\mathbf{x}^*$ nel simplesso Δ che sia il minimo globale per la seguente funzione quadratica

$$f(\mathbf{x}) = \mathbf{x}^t A \mathbf{x} \tag{3.34}$$

Come osservato già più volte questo non è un compito banale, in particolare se si è interessati proprio al minimo globale e non a uno dei possibili minimi locali. Nel prossimo capitolo vedremo una tecnica piuttosto utilizzata per trovare massimi in problemi quadratici e scopriremo come applicarla per realizzare un'implementazione pratica delle tecniche di ricerca di match fra grafi presentate in questi due ultimi paragrafi.

Un'ultima nota riguarda la forma della funzione obiettivo rappresentata dall'equazione 3.34. Si tratta in fatti di una formulazione, fedele a quella originariamente presentata nel paragrafo 3.4, che cerca vettori in grado di minimizzare la funzione proposta. In effetti vedremo che le tecniche proposte nel prossimo capitolo sono tecniche di massimizzazione. Questo fatto, non rappresenta certo un impedimento per la loro applicazione al caso appena illustrato: infatti si consideri λ il massimo valore contenuto nella matrice A

i cui elementi sono definiti mediante la formula 3.33 e si calcoli la matrice B nel seguente modo

$$B = \mathbf{e}^t \mathbf{e} \lambda - A \tag{3.35}$$

Dove $\mathbf{e}$ è un vettore con tutti gli elementi unitari.

A questo punto il problema di minimizzazione della funzione rappresentata in 3.34 è equivalente al problema di massimizzazione, sempre all'interno del simplesso della seguente funzione

$$f(\mathbf{x}) = \mathbf{x}^t B \mathbf{x} \tag{3.36}$$

E' facile verificare che ogni $\mathbf{x} \in \Delta$ è un minimizzatore della 3.34 se e solo se è un massimizzatore della 3.35.

Capitolo 4

Dinamiche di replicazione

Nel capitolo 3 abbiamo visto come ridurre alcuni problemi combinatori alla ricerca di vettori in Δ che costituiscono massimi globali per una funzione quadratica. Questa ricerca non è né banale, né immune da possibili errori dovuti all'individuazione di massimi locali i quali, in assenza di altre informazioni, non sono in effetti riconoscibili come tali.

In questo capitolo vedremo una tecnica, presa a prestito dal campo della biologia computazionale e dalla teoria dei giochi, che si è rivelata molto adatta per la massimizzazione di problemi quadratici.

4.1 Popolazioni, strategie e dinamiche

La teoria dei giochi evoluzionistici opera in uno modello astratto dove si immagina una grande popolazione dove coppie di individui sono ripetutamente estratti casualmente per confrontarsi in un gioco simmetrico. Differentemente dai modelli tradizionali nella teoria dei giochi, in questo caso ai giocatori non viene attribuito alcun comportamento razionale o alcuna conoscenza ad alto livello dei dettagli del gioco. I giocatori infatti agiscono in base a uno schema comportamentale predeterminato, detto strategia pura, e si immagina che un processo evolutivo agisca nel tempo cambiando la distribuzione nella popolazione di tali strategie (si vedano per approfondimenti in questo campo [66] e [67]).

Sia $J = \{1, ..., n\}$ l'insieme delle strategie pure disponibili e, per ogni $i \in J$, sia $x_i(t)$ la porzione di individui nella popolazione che giocano la strategia i al tempo t. Lo stato della popolazione in un dato istante di tempo sia il vettore $\mathbf{x} = (x_1, ..., x_n)^t$, il quale ovviamente apparterrà al simplesso standard Δ in quanto il totale della popolazione dovrà essere 1 e nessuna strategia può essere attuata da una proporzione negativa di individui.

Definiamo il supporto di uno stato della popolazione $\mathbf{x} \in \Delta$, indicato da $\sigma(\mathbf{x})$ come l'insieme degli indici corrispondenti a elementi non nulli, ovvero a strategie non estinte:

$$\sigma(\mathbf{x}) = \{i \in J | x_i > 0\} \tag{4.1}$$

Considerando un sottoinsieme di strategie $S \subset J$, un insieme di qualsiasi strategie tale che tutte quelle non incluse in esso sono estinte, il quale corrisponde ad una faccia di Δ, può essere definito come

$$\Delta_S = \{\mathbf{x} \in \Delta | \sigma(\mathbf{x}) \subset S\} \tag{4.2}$$

Ed il suo interno relativo come

$$int(\Delta_S) = \{\mathbf{x} \in \Delta | \sigma(\mathbf{x}) = S\} \tag{4.3}$$

Ovviamente abbiamo $\Delta_J = \Delta$ e per questo potremmo scrivere $int(\Delta)$ anzichè $int(\Delta_J)$.

Definiamo una matrice $A = (a_{ij})$ di dimensioni $n \times n$ come la matrice di payoff (o di guadagno). Per ogni coppia di strategie $i, j \in J$, l'elemento a_{ij} rappresenterà il guadagno[1] di un individuo che applica la strategia i contro un avversario che applica la strategia j. Nel contesto biologico i payoff corrispondono al fitness Darwiniano o al successo riproduttivo, mente in ambito economico possono rappresentare un guadagno in termini economici o l'acquisizione di clienti.

Quando la popolazione si trova nello stato $\mathbf{x}$ il payoff medio atteso da un giocatore che applica la strategia i risulta essere

$$\pi_i(\mathbf{x}) = \sum_{j=1}^{n} a_{ij} x_j = (A\mathbf{x})_i \tag{4.4}$$

Mentre il payoff medio su tutta la popolazione sarà

$$\pi(\mathbf{x}) = \sum_{i=1}^{n} x_i \pi_j(\mathbf{x}) = \mathbf{x}^t A \mathbf{x} \tag{4.5}$$

Nella teoria dei giochi evoluzionistici si assume che il gioco sia giocato di continuo, generazione dopo generazione, e che l'azione della selezione naturale risulti nell'evoluzione delle strategie di maggior successo. Se le generazioni che si susseguono vengono ibridate fra di loro, il comportamento fenotipico

[1] In questo contesto il termine "guadagno" indica un generico parametro di merito che indica quanto beneficio si ottiene

può essere descritto da un insieme di equazioni differenziali ordinarie una cui classe generale può essere data da

$$\dot{x}_i = x_i g_i(\mathbf{x}) \tag{4.6}$$

dove un punto sopra alla variabile denota una derivata prima rispetto al tempo e $g = (g_1, ..., g_n)$ è un vettore di funzioni il cui dominio contiene Δ. In questo caso la funzione $g_i (i \in J)$ indica la velocità con la quale la strategia i si replica quando la popolazione si trova nello stato $\mathbf{x}$. Solitamente è richiesto che la funzioni di crescita g siano regolari, cioè che appartengano a C^1 e che $g(\mathbf{x})$ sia sempre ortogonale a $\mathbf{x}$, ovvero che sia sempre vero che $g(\mathbf{x})^t\mathbf{x} = 0$. La prima condizione garantisce che il sistema di equazioni differenziali 4.6 abbia un'unica soluzione per qualsiasi stato iniziale della popolazione. La seconda, invece, assicura che il simplesso Δ sia invariante nel sistema 4.6, ovvero che ogni traiettoria iniziata all'interno di Δ vi rimanga.

Un punto $\mathbf{x}$ si dice punto stazionario (o di equilibrio) del nostro sistema dinamico se $\dot{x}_i = 0$ per tutti gli $i \in J$. Un punto stazionario $\mathbf{x}$ è detto *stabile secondo Liapunov* (o più semplicemente *stabile*) se per ogni intorno U di $\mathbf{x}$ esiste un intorno W di $\mathbf{x}$ tale che ogni traiettoria che parte in W rimanga in U. Inoltre è detto *asintoticamente stabile* se tale traiettoria converge a $\mathbf{x}$.

In questo contesto le dinamiche a payoff monotonico rappresentano un'ampia classe di dinamiche di selezione naturale regolari per le quali valgono delle importanti proprietà. Dal punto di vista intuitivo una dinamica a payoff monotonico è una dinamica in cui a payoff più alti corrispondono velocità di crescita nella popolazione della strategia relativa più alte.

Formalmente una dinamica nella forma 4.6 è detta a payoff monotonico se per ogni $\mathbf{x} \in \Delta$ vale

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \Leftrightarrow \pi_i(\mathbf{x}) > \pi_j(\mathbf{x}) \tag{4.7}$$

Nonostante questa classe contenga diverse dinamiche, si verifica che queste condividono molte proprietà comuni, come ad esempio avere lo stesso insieme di punti stazionari.

Weibull, in [67], dimostra che un punto $\mathbf{x} \in \Delta$ è stazionario sotto qualsiasi dinamica a payoff monotonico se e solo se $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ per ogni $i \in \sigma(\mathbf{x})$.

Una classe piuttosto nota di dinamiche a payoff monotonico è la seguente

$$\dot{x}_i = x_i \left(\phi(\pi_i(\mathbf{x})) - \sum_{j=1}^{n} x_j \phi(\pi_j(\mathbf{x})) \right) \tag{4.8}$$

Dove $\phi(u)$ è una funzione crescente di u. Questi modelli emergono nella

modellazione dei comportamenti evolutivi per imitazione, dove i giocatori hanno l'opportunità di cambiare occasionalmente strategia [68].

Quando ϕ è la funzione identità, cioè $\phi(u) = u$, otteniamo le dinamiche di replicazione standard

$$\dot{x}_i = x_i \left(\pi_i(\mathbf{x}) - \sum_{j=1}^{n} x_j \pi_j(\mathbf{x}) \right) \tag{4.9}$$

La cui idea alla base è quella che la velocità media di crescita $\dot{x}_i/x_i$ è uguale alla differenza fra il fitness medio della strategia i e il fitness medio dell'intera popolazione.

Un altro modello diffuso emerge quanto ϕ è una funzione esponenziale, ovvero quando $\phi(u) = e^{ku}$, ottenendo quindi

$$\dot{x}_i = x_i \left(e^{k\pi_i(\mathbf{x})} - \sum_{j=1}^{n} x_j e^{k\pi_j(\mathbf{x})} \right) \tag{4.10}$$

Dove k è una costante positiva. Quando k tende a 0, l'orbita di questa dinamica approssima quella della dinamica standard, rallentata di un fattore k.

4.2 Applicazione ai problemi quadratici

Consideriamo il seguente problema quadratico standard

$$\begin{array}{ll} \text{massimizzare} & \pi(\mathbf{x}) = \mathbf{x}^t A \mathbf{x} \\ \text{soggetto a} & \mathbf{x} \in \Delta \end{array} \tag{4.11}$$

dove A è una arbitraria matrice $n \times n$ simmetrica. Notiamo che queste sono le condizioni sia della matrice della funzione obiettivo per la tecnica dei Dominant Set proposta in 3.28 che della matrice legata alla tecnica basata sul grafo di associazione fra archi proposta in 3.33 e della matrice del problema di massimizzazione associato calcolata con 3.35. Nel contesto della teoria dei giochi evoluzionistici matrici di payoff simmetriche di questo tipo emergono nell'ambito dei giochi doppiamente simmetrici (o di partnership) dove gli interessi di entrambi i giocatori coincidono.

Un punto $\mathbf{x}^* \in \Delta$ è detto una soluzione globale di 4.11 se $\pi(\mathbf{x}^*) \geq \pi(\mathbf{x})$ per ogni $\mathbf{x} \in \Delta$. Viene detto una soluzione locale se esiste un $\epsilon > 0$ tale che $\pi(\mathbf{x}^*) \geq \pi(\mathbf{x})$ per ogni $\mathbf{x} \in \Delta$ la cui distanza da $\mathbf{x}^*$ sia minore di ϵ, e se $\pi(\mathbf{x}^*) = \pi(\mathbf{x}) \Leftrightarrow \mathbf{x}^* = \mathbf{x}$, allora $\mathbf{x}^*$ è detto una soluzione locale stretta. si noti che la soluzione di 4.11 rimane invariata se la matrice A viene sostituita

con la matrice $A + k\mathbf{e}\mathbf{e}^t$, dove k è una costante arbitraria. Inoltre si osservi che massimizzare l'espressione quadratica non omogenea $\mathbf{x}^t Q\mathbf{x} + 2\mathbf{c}^t\mathbf{x}$ su Δ è equivalente a risolvere il problema quadratico 4.11 con $A = Q + \mathbf{e}\mathbf{c}^t + \mathbf{c}\mathbf{e}^t$.

Un punto $\mathbf{x}^* \in \Delta$ soddisfa le condizioni di Karush-Kuhn-Tucker (KTT), cioè le condizioni al primo ordine per l'ottimalità, per il problema 4.11 se esistono $n+1$ costanti $\mu_1, ..., \mu_n$ e λ, con $\mu_i \geq 0$ per ogni $i \in \{1, ..., n\}$ tali che

$$(A\mathbf{x})_i - \lambda + \mu_i = 0 \tag{4.12}$$

per ogni $i \in \{1, ..., n\}$, e

$$\sum_{i=1}^{n} x_i \mu_i = 0 \tag{4.13}$$

Si noti che, poichè sia x_i che μ_i sono non negative per ogni $i \in \{1, ..., n\}$, l'ultima condizione è equivalente ad affermare che $i \in \sigma(\mathbf{x})$ implica $\mu_i = 0$. Quindi le condizioni KTT possono essere riscritte come:

$$(A\mathbf{x})_i \begin{cases} = \lambda & \text{se } i \in \sigma(\mathbf{x}) \\ \leq \lambda & \text{se } i \notin \sigma(\mathbf{x}) \end{cases} \tag{4.14}$$

per qualche costante reale λ. D'altra parte è immediato verificare che $\lambda = \mathbf{x}^t A\mathbf{x}$. D'ora in poi ci riferiremo ad un punto che soddisfa la 4.14 come ad un punto KTT.

Poichè come abbiamo visto nel paragrafo 4.1 un punto $\mathbf{x} \in \Delta$ è stazionario sotto qualsiasi dinamica a payoff monotonico se e solo se $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ per ogni $i \in \sigma(\mathbf{x})$, è facile vedere che se $\mathbf{x} \in \Delta$ è un punto KTT per 4.11 allora è un punto stazionario sotto qualsiasi dinamica a payoff monotonico. Inoltre se $\mathbf{x} \in int(\Delta)$ vale anche il viceversa.

Chiaramente non tutti i punti stazionari corrispondono a punti KTT (per esempio i vertici di Δ, tuttavia si può dimostrare che questo è vero quando il punto stazionario viene raggiunto da una traiettoria partita dall'interno di Δ.

Hofbauer, in [68], dimostra inoltre, generalizzando il famoso teorema fondamentale della selezione naturale [9], che per matrici di payoff simmetriche il payoff medio della popolazione è una funzione strettamente crescente lungo ogni traiettoria generata da una dinamica a payoff monotonico, ovvero:

Teorema 1 *Se la matrice di payoff A è simmetrica, allora $\pi(\mathbf{x}) = \mathbf{x}^t A\mathbf{x}$ è strettamente crescente lungo ogni traiettoria non costante guidata da una dinamica a payoff monotonico. Ovvero $\dot{\pi}(\mathbf{x}(t))) \geq 0$ per ogni t, con eguaglianza se e solo se $\mathbf{x} = \mathbf{x}(t)$ è un punto stazionario.*

Prima di tutto questo risultato fornisce una funzione di Liapunov stretta per le dinamiche a payoff monotonico ed esclude la presenza di evoluzioni non costanti come, ad esempio cicli. Inoltre permette di dimostrare una forte connessione fra le proprietà di stabilità delle dinamiche e le soluzioni del nostro problema quadratico originale 4.11 (si veda [69] per una trattazione approfondita):

Teorema 2 *Un punto* $\mathbf{x} \in \Delta$ *è una soluzione locale stretta del problema quadratico 4.11 se e solo se è asintoticamente stabile sotto qualsiasi dinamica di replicazione a payoff monotonico.*

Quest'ultimo risultato ci permette di applicare le dinamiche di replicazione presentate in 4.9 e 4.10 per cercare soluzioni (locali) di problemi quadratici a matrice dei pesi simmetrica, ed in particolare dei due problemi quadratici 3.28 e 3.33 che abbiamo proposto come differenti riduzioni dei nostri problemi di graph matching.

In termini pratici avremo bisogno di trovare una formulazione discreta nel tempo della 4.9 e della 4.10 in modo da poterle applicare semplicemente ad un algoritmo che faccia evolvere, passo dopo passo, la traiettoria del vettore soluzione in Δ. Di questo ce ne occuperemo nel paragrafo 4.3.

Infine osserviamo che questa formulazione garantisce solo la convergenza asintotica e non da un certo tempo in poi. Poichè ovviamente le nostre implementazioni dovranno terminare in un tempo finito, sarà necessario determinare dei criteri pratici di arresto e delle tecniche per estrapolare correttamente il supporto dal vettore soluzione. Si troveranno maggiori informazioni circa questi dettagli implementativi all'interno del capitolo 5.

4.3 Dinamiche a tempo discreto

Come abbiamo visto nel paragrafo 4.2 le dinamiche di replicazione possono essere utilizzate con successo per trovare massimizzatori locali (e sperabilmente globali) di problemi quadratici.

Perchè sia possibile però realizzare semplicemente un'implementazione pratica di tali dinamiche sotto forma di un algoritmo sarebbe molto utile poter discretizzare tale modello nei confronti del tempo. Una tecnica classica per ottenere questo risultato è utilizzare le seguenti discretizzazioni, rispettivamente per la dinamica standard presentata nella formula 4.9:

$$x_i(t+1) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^{n} x_j(t)\pi_j(t)} \tag{4.15}$$

e per la dinamica esponenziale presentata nella formula 4.10:

$$x_i(t+1) = \frac{x_i(t)e^{k\pi_i(t)}}{\sum_{j=1}^{n} x_j(t)e^{k\pi_j(t)}} \tag{4.16}$$

Il modello presentato nella formula 4.15 è la dinamica standard a tempo discreto ed è stata sperimentata con successo nella ricerca di clique massime e problemi simili, dimostrando di essere competitiva con euristiche molto più sofisticate basate su reti neurali (si veda [60], [70], [62], [59], [20], [71]).

Il modello presentato in 4.16 è stato utilizzato in [71] e [72] come euristica per il problema della ricerca di un isomorfismo massimo fra alberi.

Queste dinamiche, come le loro controparti continue, sono a payoff monotonico, ovvero

$$\frac{x_i(t+1) - x_i(t)}{x_i(t)} > \frac{x_j(t+1) - x_j(t)}{x_j(t)} \Leftrightarrow \pi_i(t) > \pi_j(t) \tag{4.17}$$

É un noto risultato della teoria dei giochi evoluzionistici ([67], [66]) che il teorema fondamentale della selezione naturale (1) vale anche per la dinamica discreta standard 4.15. Per la precisione che per matrici A simmetriche $\mathbf{x}^t A\mathbf{x}$ è una funzione di Liapunov stretta, ovvero

$$\mathbf{c}(t)^t A\mathbf{x}(t) < \mathbf{c}(t+1)^t A\mathbf{x}(t+1) \tag{4.18}$$

A meno che $\mathbf{x}(t)$ non sia un punto stazionario. Sfortunatamente, a differenza del caso continuo, non si può dire lo stesso per la dinamica esponenziale discreta espressa dalla formula 4.16. Cioè non ci sono garanzie che per qualsiasi valore del parametro k la dinamica faccia crescere il valore di $\mathbf{x}^t A\mathbf{x}$. Addirittura in [71] si dimostra che per alti valori di tale parametro la dinamica assume un comportamento oscillatorio. Tuttavia un recente risultato di Bomze ([73]) ci permette di definire un approccio adattativo che garantisce il raggiungimento di un massimizzatore (locale) per $\mathbf{x}^t A\mathbf{x}$.

Questo risultato dimostra, per ogni problema quadratico con matrice dei pesi A simmetrica e per ogni punto $\mathbf{x} \in \Delta$, l'esistenza di un determinato valore di k_a tale che per ogni $k < k_a$ l'applicazione di una singola iterazione della 4.16 fa crescere il valore di $\mathbf{x}^t A\mathbf{x}$. L'esistenza di tale k_a è di per se sufficiente a garantire l'esistenza di una funzione di Liapunov associata alla nostra dinamica e pertanto a garantirne la convergenza.

A questo punto, dal punto di vista pratico è necessario definire una tecnica per trovare il valore di k da utilizzare per ciascuna iterazione della dinamica esponenziale discreta.

Sulla base di quanto suggerito in [69] noi abbiamo applicato una tecnica auto adattativa che assegna inizialmente a k un valore alto e applica un'iterazione della dinamica 4.16. Se questa iterazione porta ad un aumento di $\mathbf{x}^t A\mathbf{x}$ il valore di k viene lasciato invariato, se invece questo non avviene k viene ripetutamente dimezzato fino a quando un'iterazione della dinamica non fa crescere la funzione obiettivo.

Capitolo 5

Risultati sperimentali

Per verificare le prestazioni (in termini di qualità del match) dei metodi illustrati nei paragrafi 3.6 e 3.7 abbiamo deciso di implementare due algoritmi che utilizzano le dinamiche di replicazione adattative trattate nel capitolo 4 per individuare i massimizzatori (globali o locali) dei rispettivi problemi quadratici.

I due algoritmi sono stati confrontati fra di loro e comparati con un'implementazione della tecnica del Graduated Assignment [4]. Tale confronto è stato fatto su grafi con diversi gradi di connettività e sottoposti a diversi livelli di rumore strutturale (cancellazione casuale di una certa percentuale di vertici) e di perturbazione gaussiana degli attributi.

5.1 Condizioni sperimentali generali

Gli algoritmi sottoposti a test sono stati realizzati in linguaggio C e le prove sono state effettuate su un sistema AMD Opteron a 64bit con frequenza di clock 2Ghz e memoria ram 1Gb. I problemi sono stati dimensionati in modo tale da non utilizzare memoria swap.

Sono stati eseguiti test su tre algoritmi, denominati:

- *DominantSets*: utilizza le dinamiche di replicazione adattative per massimizzare il problema quadratico illustrato nel paragrafo 3.6. Come vedremo tale algoritmo sarà tarato determinando il miglior fattore di scala per i vertici e la migliore similarità da attribuire alle coppie di archi assenti associati;

- *EdgesAssociationGraph*: utilizza le dinamiche di replicazione adattative per massimizzare il problema quadratico illustrato nel paragrafo

3.6. Questo algoritmo presenta al momento una complessità computazionale e dei requisiti di memoria molto elevati, vedremo nel capitolo 6 alcune proposte per ridurle;

- *GraduatedAssignment*: utilizza la tecnica di match proposta da Gold e Rangarajan in [4]. E' uno dei migliori algoritmi per il graph matching in termini di prestazioni e velocità, tuttavia non trova sempre isomorfismi;

Tutti i test sono stati effettuati su grafi da 30 nodi, sottoposti a diversi tipi di perturbazione e con diversi gradi di connettività. A causa dell'attuale complessità di EdgesAssociationGraph e dei suoi requisiti di memoria non è stato possibile eseguire test con grafi di dimensioni maggiori, tuttavia a questo livello grafi di 30 nodi dovrebbero essere di dimensione sufficiente per confrontare gli algoritmi.

Ogni test prevede la generazione di due grafi e l'esecuzione degli algoritmi per la ricerca del match ottimale fra di essi. Il primo grafo viene generato casualmente mentre il secondo viene inizialmente copiato dal primo e successivamente perturbato mediante:

- *Perturbazioni strutturali*: viene eliminata una certa percentuale di nodi dal grafo in modo tale da ridurre la cardinalità dell'isomorfismo massimo individuabile;

- *Rumore gaussiano*: il valore di tutti gli attributi, sia dei vertici che degli archi, viene alterato sommando un errore estratto da una distribuzione gaussiana di media 0 e con diversi valori di σ.;

I grafi presentano attributi su vertici e archi sotto forma di valori reali compresi fra 0.1 e 1.0, estremi inclusi. Siano i grafi definiti come specificato nel paragrafo 3.1: in questo caso le funzioni α_1,α_2,β_1 e β_2 che associano a vertici ed archi dei due grafici attributi hanno tutte codominio $\mathbb{R}$. Le funzioni di similarità fra due attributi σ e ρ, dati due attributi in $\mathbb{R}$ nominati a_1 e a_2 sono definite semplicemente come

$$\sigma(a_1, a_2) = \rho(a_1, a_2) = e^{|a_1 - a_2|} \tag{5.1}$$

Per quanto riguarda la similarità fra i vertici associati da un match è possibile utilizzare la funzione ω descritta in 3.1, tuttavia la funzione γ deve essere modificata in

$$\gamma((u_1,u_2),(v_1,v_2)) = \begin{cases} \rho(\beta_1(u_1,v_1),\beta_2(u_2,v_2)) & \text{se } (u_1,v_1)\in E_1,(u_2,v_2)\in E_2 \\ \rho(0,\beta_2(u_2,v_2)) & \text{se } (u_1,v_1)\notin E_1,(u_2,v_2)\in E_2 \\ \rho(\beta_1(u_1,v_1),0) & \text{se } (u_1,v_1)\in E_1,(u_2,v_2)\notin E_2 \\ 1 & \text{se } (u_1,v_1)\notin E_1,(u_2,v_2)\notin E_2 \end{cases} \tag{5.2}$$

Il motivo di questa scelta è da ricondurre alla necessità di confrontare fra di loro i tre algoritmi sottoposti a test. Infatti GraduatedAssignment non garantisce che il match individuato sia un isomorfismo, pertanto utilizzando la versione di γ 3.2 il peso totale del match calcolato potrebbe non essere calcolabile in quanto quest'ultimo potrebbe non conservare l'adiacenza dei vertici.

La 5.2 non solo è calcolabile anche nel caso il match non sia un isomorfismo, ma ha anche un significato che in qualche modo si avvicina a quello della 3.2: in effetti all'interno del dominio degli isomorfismi quello di peso massimo massimizza la funzione di peso totale 3.3 sia utilizzando come misura di similarità fra gli archi la 3.2 che la 5.2. Naturalmente se si esce dal dominio degli isomorfismi è possibile trovare, in generale, match di peso anche superiore a quello dell'isomorfismo di peso massimo, tuttavia nel nostro caso questo non accade in quanto si cercano match fra grafi di cui uno è la copia dell'altro alla quale vengono applicate perturbazioni e delezioni, di conseguenza dovrebbe sempre esistere un isomorfismo di peso massimo che coinvolge tutti i vertici del secondo grafo[1].

É necessario anche notare che il fatto che GraduatedAssignment possa produrre anche match non isomorfi può essere in qualche modo fuorviante per i confronti, per questo motivo, oltre a confrontare il peso del match ottenuto dai vari algoritmi ed i loro tempi di esecuzione, abbiamo ritenuto utile riportare degli istogrammi che indichino nelle varie condizioni la percentuale di isomorfismi trovati da GraduatedAssignment (gli altri due algoritmi naturalmente trovano sempre isomorfismi).

5.2 Alcuni dettagli implementativi

Per quanto riguarda GraduatedAssignment in [4] viene lasciata una certa libertà per quanto riguarda la procedura di clean-up della matrice risultato. In questa implementazione, pur accettando la possibilità di trovare match

[1] In realtà con grosse perturbazioni potrebbe crearsi un isomorfismo di peso massimo, ma di cardinalità inferiore a quella del secondo grafo, ma non dovrebbe essere il nostro caso

non isomorfi, abbiamo optato per una soluzione che garantisca almeno il fatto che ogni match sia una funzione fra i due insiemi di vertici, ovvero che ogni vertice del primo grafo che partecipa al match sia mappato in un solo vertice del secondo e che nessun vertice del secondo sia immagine di più vertici del primo. Per ottenere questo risultato abbiamo applicato prima una selezione del massimo per colonne, come suggerito in [4] e successivamente, in caso fossero stati trovati due massimi sulla stessa riga, selezionato il più grande fra di essi. Da prove effettuate, ed in accordo con quanto illustrato in [4] tali casi sono piuttosto rari e si verificano soprattutto per basse connettività o alti livelli di rumore strutturale.

L'algoritmo di clean-up utilizzato segue il seguente pseudo codice

---

CleanUpMatrix(m), con m matrice finale a convergenza

1. Per tutte le colonne trova l'elemento massimo e azzera gli altri
2. Per tutte le righe trova l'elemento massimo e azzera gli altri
3. Elimina le variabili slack dalla matrice

Tabella 5.1: Pseudo codice dell'algoritmo di clean-up GraduatedAssignment

Per quanto riguarda i parametri iniziali e le condizioni di convergenza di questo algoritmo sono stati utilizzati esattamente i valori suggeriti in [4].

Nell'implementazione degli algoritmi DominantSets e EdgesAssociation-Graph abbiamo applicato le dinamiche esponenziali adattative presentate in [69] e una volta arrivati a convergenza, anzichè fissare una soglia di cut-off per selezionare le coppie di vertici partecipanti al match abbiamo applicato un algoritmo greedy che funziona nel seguente modo

---

GreedyMatchFromVector($\mathbf{x}$), con $\mathbf{x}$ vettore finale a convergenza

1. Trova l'elemento massimo di $\mathbf{x}$
2. assegna al match M la coppia ad esso associato e azzeralo
3. se M non è più un isomorfismo rimuovi la coppia
4. torna a 1 finchè vi sono elementi in $\mathbf{x}$

Tabella 5.2: Pseudo codice dell'algoritmo estrazione greedy dei match

5.3 Calibrazione del metodo basato sui Dominant Set

Prima di cominciare gli esperimenti veri e propri e confrontare i diversi algoritmi proposti vi è ancora un passo da effettuare per garantire un confronto appropriato.

L'algoritmo DominantSets infatti necessita del tuning di almeno due parametri fondamentali, ovvero il fattore di scala da applicare alle similarità fra i vertici (per evitare che il sistema converga verso un vertice del simplesso) e la similarità assegnata alle coppie di archi non presenti in entrambi i grafi associate dall'isomorfismo indotto dai vertici (per regolare il loro contributo nel peso totale del match).

Poichè si tratta di due parametri sostanzialmente da fissare sperimentalmente, abbiamo deciso di eseguire una serie di prove su un grafo campione con diverse condizioni di perturbazione sugli attributi e rumore strutturale. Sulla base di tali prove abbiamo poi cercato di individuare un valore dei parametri che dia il miglior risultato nella maggior parte degli esperimenti: in questo senso come parametro di valutazione della bontà del risultato ottenuto è stato scelto il rapporto fra il peso del match individuato con l'uso dei parametri candidati e il massimo peso di match trovato.

Come vedremo dai risultati degli esperimenti la scelta è caduta su un fattore di scala delle similarità dei vertici di 0.3 e un valori di similarità fra coppie di archi non presenti di 1.0, ovvero il massimo valore possibile in base alla funzione di similarità 5.1.

Sono state fatte due classi di esperimenti: la prima valuta l'effetto di diversi livelli di rumore strutturale per diversi gradi di connettività del grafo, la seconda esplora invece gli effetti della perturbazione degli attributi mediante rumore gaussiano con diversi valori di σ. Ogni esperimento è stato eseguito per valori dei parametri da ottimizzare che vanno da 0.1 a 1.0 e vengono riportati gli andamenti del peso del match trovato. Alla fine di ogni serie viene riportato l'andamento del rapporto fra il valore del match trovato con i parametri candidati e il massimo valore del peso all'interno di quell'esperimento.

In linea di massima sarebbe stato opportuno eseguire gli esperimenti su un certo numero di grafi generati casualmente e riportare il peso medio dei match trovati: purtroppo a causa del limitato tempo a disposizione e dell'alto numero di test necessari (100 per orni esperimento) è stato possibile testare un unico grafo per ogni prova. Tale scelta non è comunque un limite particolarmente gravoso, infatti i risultati ottenuti sono sufficienti per indicare un comportamento generale dell'algoritmo al variare dei parametri euristici.

5.3.1 Variazione del rumore strutturale con connettività 0.01

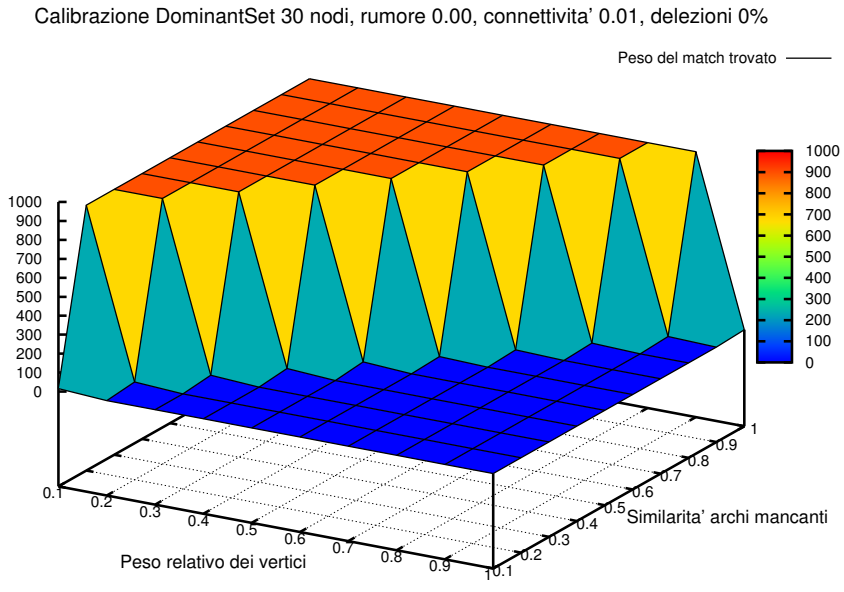

Figura 5.1: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.01, delezioni 0%

Applicando DominantSet a due grafi perfettamente isomorfi con bassa connettività vediamo (in figura 5.1) che viene trovato un match adeguato solamente se il fattore di scala associato ai vertici è inferiore al peso associato agli archi assenti.

Questo comportamento è coerente con le nostre aspettative: infatti con grafi così poco densi ogni isomorfismo sarà costituito soprattutto da vertici non connessi, pertanto il peso associato alle coppie di archi non presenti è estremamente determinante nel match finale, ne segue che se il fattore di scala associato ai vertici è maggiore di esso le dinamiche di replicazione tenderanno a convergere su un vertice del simplesso ed in generale verso il vertice di peso massimo, producendo quindi un isomorfismo di un solo vertice di peso 1.

In generale, e come vedremo nei test successivi, è ragionevole aspettarsi che il fattore di scala associato ai vertici dovrà essere piuttosto piccolo, ma avremo indizi maggiori su un valore adeguato in seguito.

In presenza di poche delezioni, come vediamo in figura 5.2, il comportamento non cambia di molto: il grafo continua ad essere poco connesso e quindi gli isomorfismi sono costituiti soprattutto da vertici non connessi.

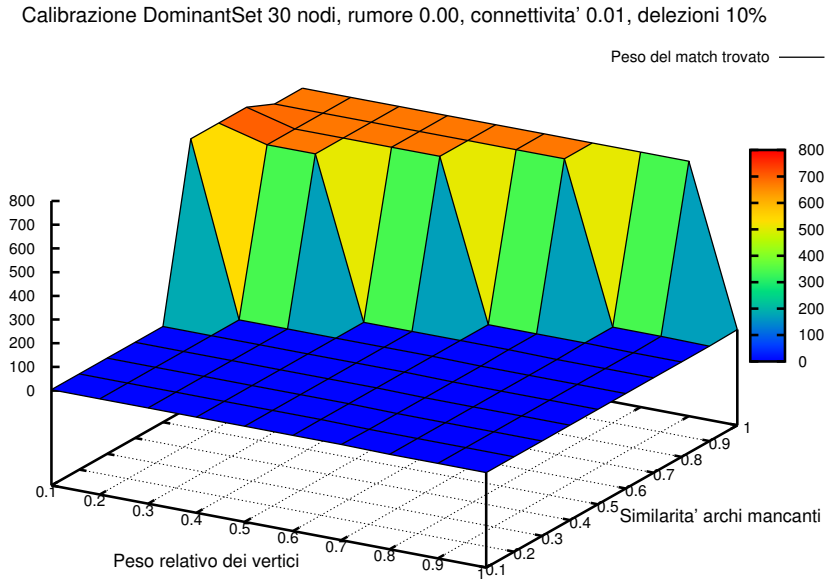

Figura 5.2: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.01, delezioni 10%

In questo caso notiamo anche che il massimo assoluto non si trova in corrispondenza del peso associato alle coppie di archi pari a 1.0. É comunque necessario ricordare che si tratta di test effettuati su un solo grafo e quindi facilmente soggetti a condizioni particolari non sfumate attraverso la media.

Il medesimo comportamento viene mantenuto ancora in figura 5.3 aumentando il numero di delezioni. Notiamo che in ogni caso, a patto che il peso associato alle coppie di archi non presenti sia alto (apparentemente vale la pena di portarlo a 1.0), il fattore di scala associato ai vertici diventa poco rilevante.

Infine, anche portando le delezioni al 50% come si può vedere in figura 5.4, confermiamo il principio generale che lega la valorizzazione dei due parametri.

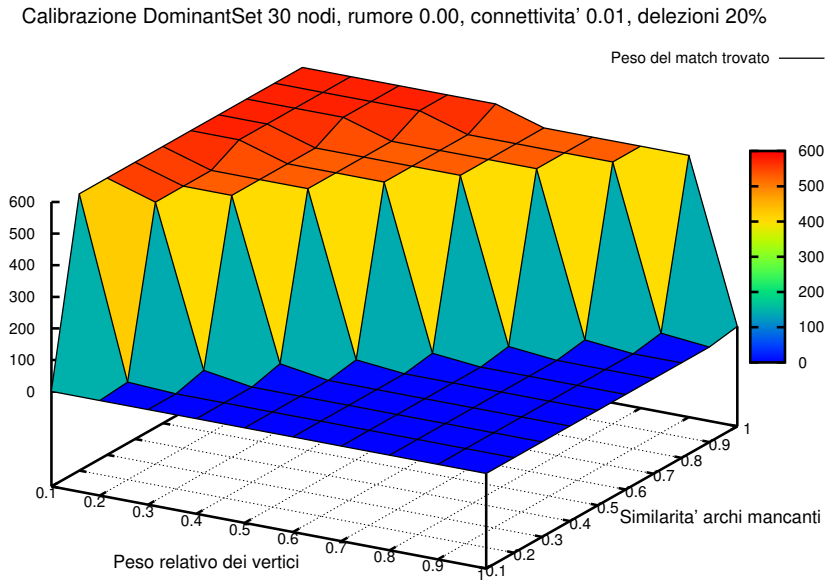

Figura 5.3: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.01, delezioni 20%

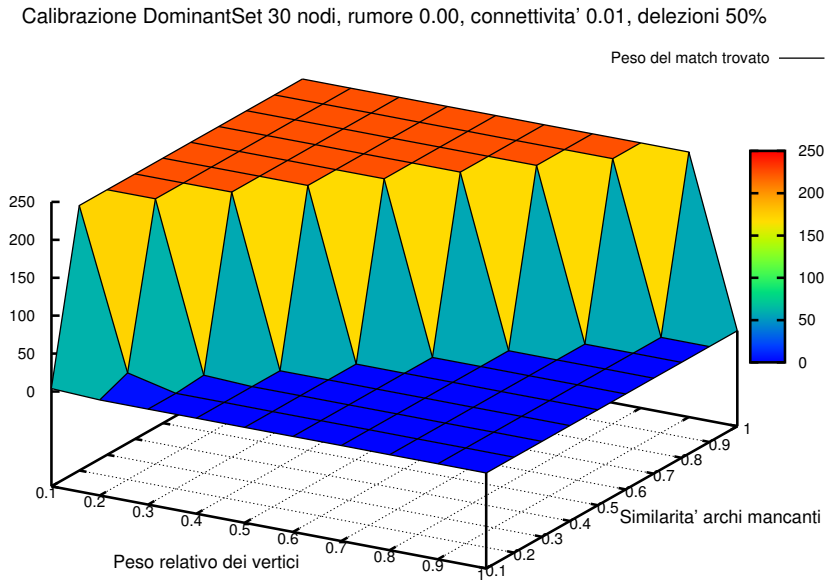

Figura 5.4: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.01, delezioni 50%

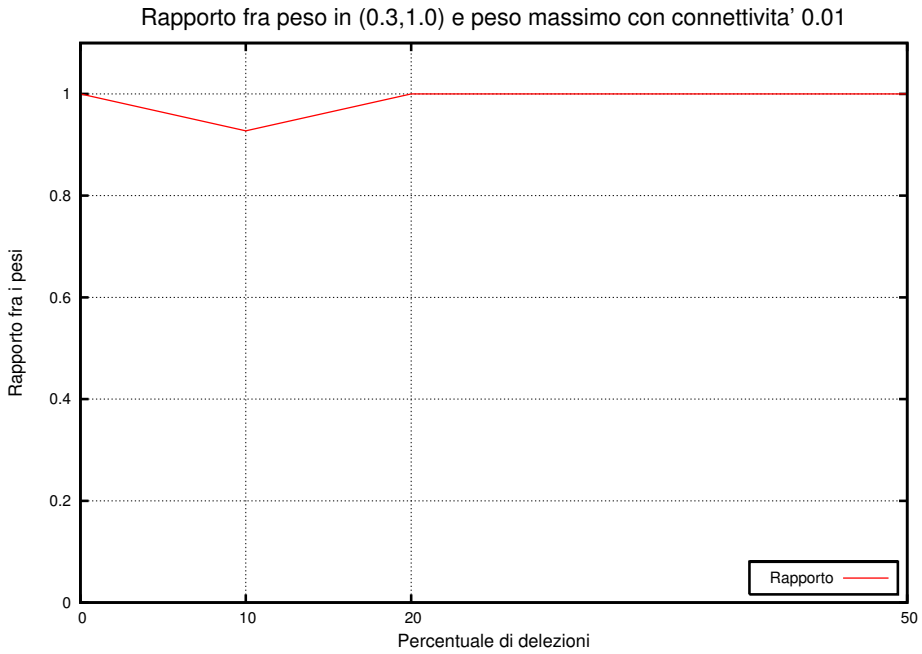

Figura 5.5: Peso relativo ottenuto con i parametri candidati senza rumore con connettività 0.01

A conclusione di questa serie di prove riportiamo l'andamento del rapporto fra il peso dell'isomorfismo trovato con i valori candidati 0.3 e 1.0 dei parametri e il peso massimo trovato in ciascuna prova. Anche se i parametri candidati non trovano sempre l'isomorfismo di peso massimo sembra che raggiungano sempre un buone risultato e, come vedremo analizzando le prove successive, tale comportamento si conserva anche nelle altre condizioni sperimentali.

5.3.2 Variazione del rumore strutturale con connettività 0.10

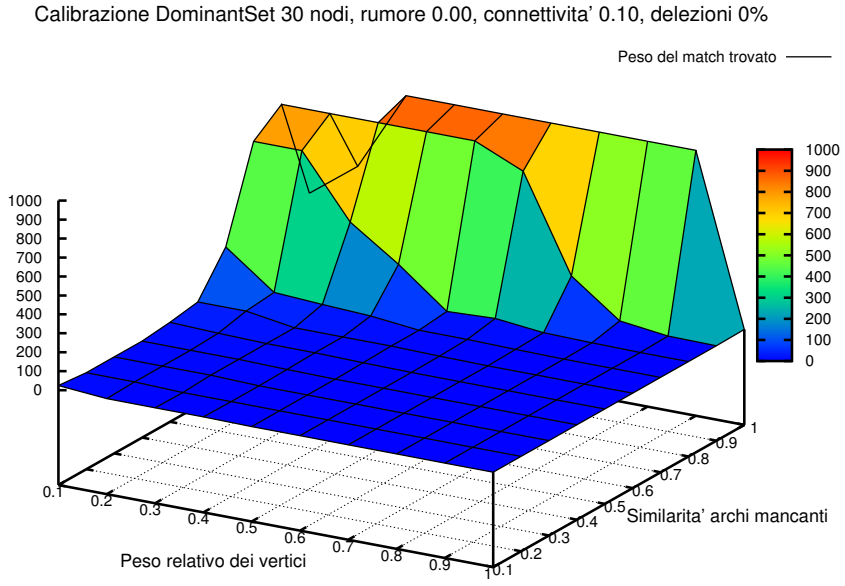

Figura 5.6: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.10, delezioni 0%

Gli esperimenti della serie precedente sono stati ripetuti aumentando la connettività dei grafi di test al 10% per vedere come varia il comportamento dell'algoritmo DominantSet in questo caso e verificare se la scelta dei parametri candidati fosse ancora adeguata.

Già dall'osservazione della figura 5.6 notiamo un andamento più irregolare, dovuto probabilmente al fatto che i grafi hanno una maggiore connettività, di conseguenza esistono maggiori condizioni di incompatibilià fra le associazioni di vertici e determinati valori dei parametri possono far tendere più facilmente l'algoritmo verso isomorfismi non ottimali. In generale comunque vale la considerazione che il fattore di scala associato ai vertici deve essere piccolo, mentre il peso associato alle coppie di archi non presenti alto.

In questo caso specifico un fattore di scala associato ai vertici troppo piccolo, in particolare in corrispondenza del valore massimo del peso associato alle coppie di archi non presenti, non dà risultati ottimali. Pur tenendo conto della limitatezza della prova, effettuata su un solo grafo, questo dato può in qualche modo essere indicativo del fatto che un fattore di scala troppo piccolo sui vertici tende a limitare la loro rilevanza numerica.

Anche nel test effettuato introducendo un po' di delezioni, riportato in figura 5.7, notiamo che anche al valore massimo del peso associato alle coppie di archi non presenti valori troppo piccoli del fattore di scala sui vertici tendono a dare risultati non ottimali. Come nel caso precedente e nei successivi che vedremo, il valore minimo accettabile per tale fattore di scala risulta essere intorno a 0.3 (sempre tenendo conto che stiamo facendo una valutazione empirica basata su test limitati).

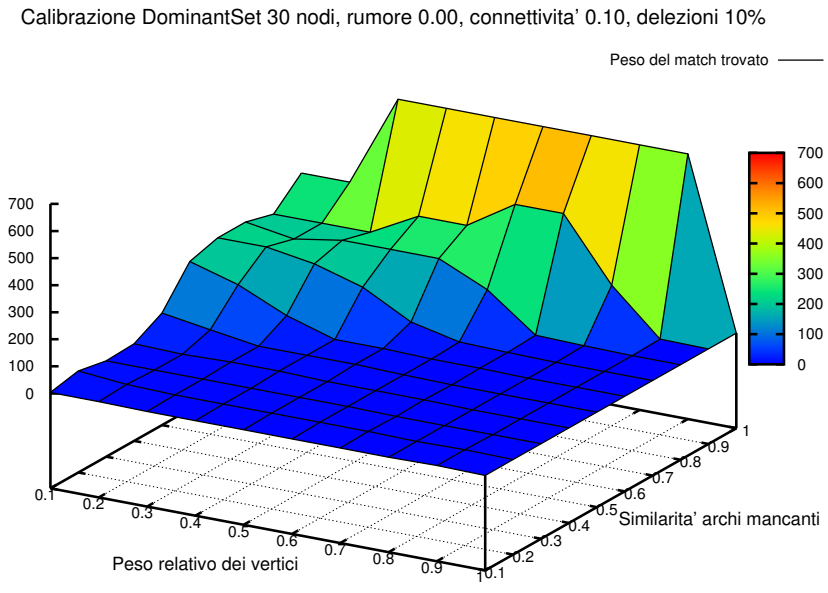

Figura 5.7: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.10, delezioni 10%

Risultati analoghi vengono evidenziati anche dagli esperimenti rappresentati nelle figure 5.8 e 5.9: nella prima l'andamento non è molto diverso da quello trovato in figura 5.7, mentre la seconda è più irregolare e presenta alcuni massimi relativi in posizioni inattese.

É comunque necessario osservare che nel caso con il 50% di delezioni riportato in figura 5.9 si ottengono isomorfismi di peso in generale piuttosto basso rispetto agli altri esperimenti, quindi probabilmente ci troviamo di fronte ad una coppia di grafi random ostici per l'algoritmo o per la dinamica.

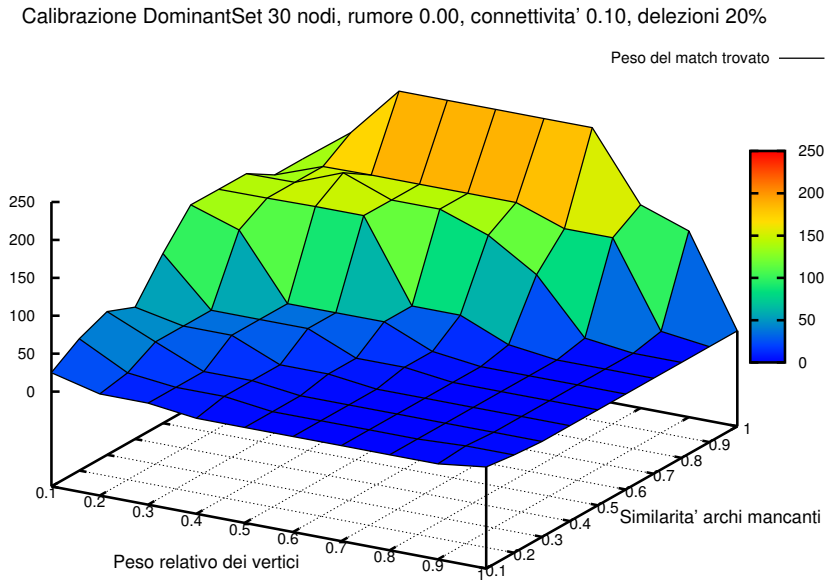

Figura 5.8: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.10, delezioni 20%

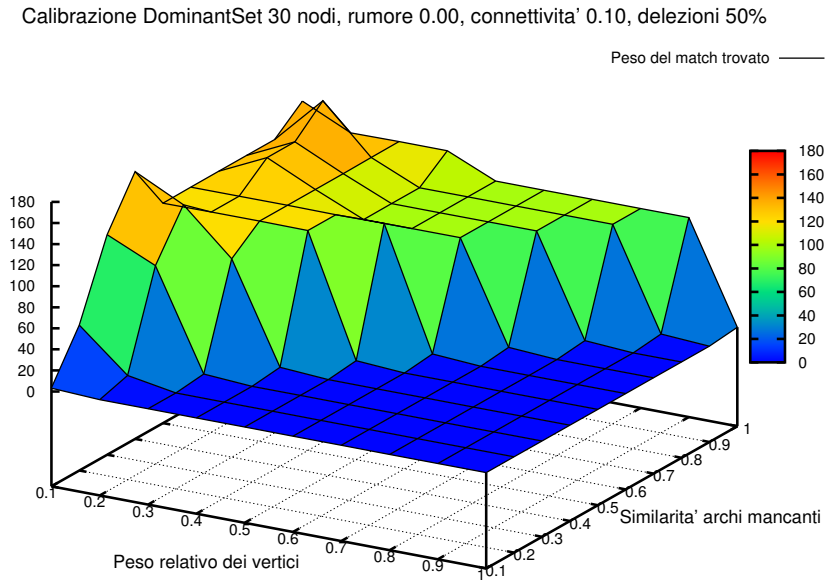

Figura 5.9: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.10, delezioni 50%

Figura 5.10: Peso relativo ottenuto con i parametri candidati senza rumore con connettività 0.10

Riportando anche in questo caso l'andamento del rapporto fra il peso dell'isomorfismo trovato con i valori candidati 0.3 e 1.0 dei parametri e il peso massimo trovato in ciascuna prova notiamo che nel caso della connettività al 50% il risultato non è estremamente buono. Tuttavia, come spiegato precedentemente, questo particolare caso è legato ad un grafo probabilmente difficile da aggredire per questa tecnica in quanto tutti i valori di peso dei match trovati sono risultati piuttosto bassi.

Probabilmente una serie di test fatti su un numero maggiore di grafi darebbe indizi più precisi sul comportamento, ad ogni modo dobbiamo ricordare che il nostro obiettivo è solo quello di trovare una parametrizzazione ragionevole, la quale non può comunque essere fissata in termini assoluti in quanto la stessa natura della tecnica alla quale applicheremo tale parametrizzazione è approssimata ed il suo comportamento dipende da fattori intrinseci alla topologia delle singole coppie di grafi e alle dinamiche applicate.

5.3.3 Variazione del rumore strutturale con connettività 0.50

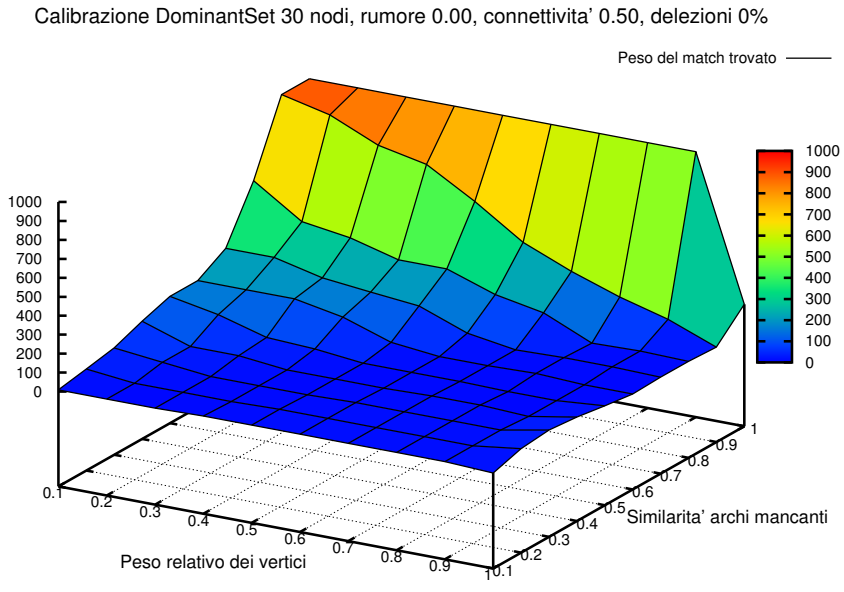

Figura 5.11: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.50, delezioni 0%

La serie di esperimenti per valutare la bontà dei parametri candidati al variare del numero di delezioni si conclude verificando il comportamento di DominantSet con grafi random con grado di connettività 0.5 e vari livelli di rumore strutturale.

L'andamento rappresentato nella figura 5.11 evidenzia, in questo caso, anche una maggiore necessità di mantenere alto il peso associato alle coppie di archi non presenti e ancora una volta il valore massimo, cioè 1.0, sembra essere la scelta ottimale. In queste condizioni il valore del fattore di scala associato ai vertici sembra non essere influente, ma in ogni caso, anche al ridursi del peso associato alle coppie di archi non presenti si nota in generale un maggiore valore del peso complessivo del match in prossimità di bassi valori del fattore di scala associato ai vertici.

Con grafi di questa densità introdurre delle delezioni non ha molta influenza sul comportamento dell'algoritmo. Infatti in figura 5.12 vediamo un risultato piuttosto simile a quello già visto nell'esperimento precedente, cioè la necessità di un valore molto alto per il peso associato alle coppie di archi non presenti ed un scarsa influenza del fattore di scala associato ai vertici.

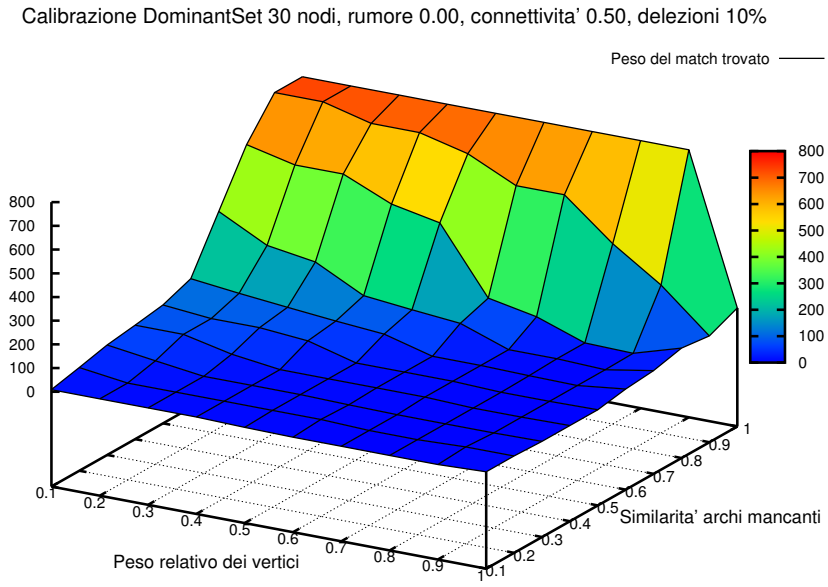

Figura 5.12: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.50, delezioni 10%

Gli stessi risultati si mantengono per quanto riguarda gli esperimenti effettuati con una percentuale di delezione via via più alta, come vediamo nelle figure 5.13 e 5.14.

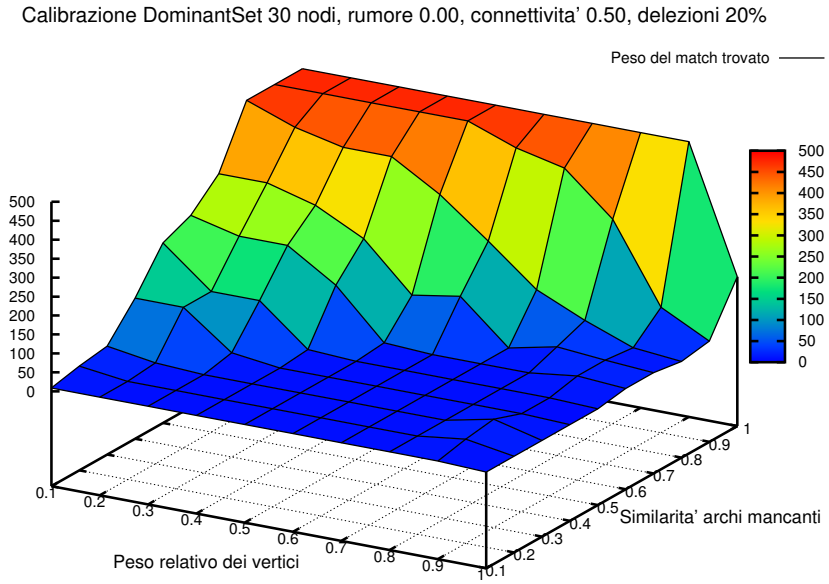

Figura 5.13: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.50, delezioni 20%

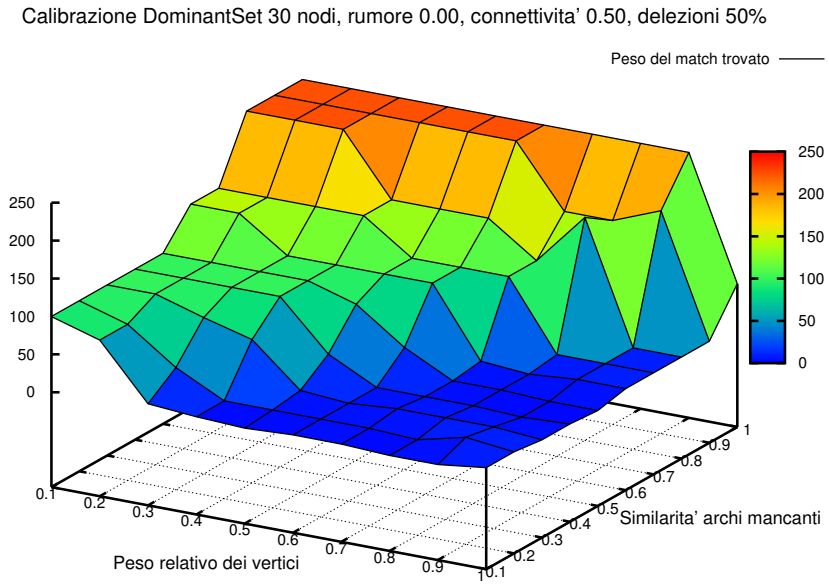

Figura 5.14: Calibrazione DominantSet 30 nodi, rumore 0.0, connettività 0.50, delezioni 50%

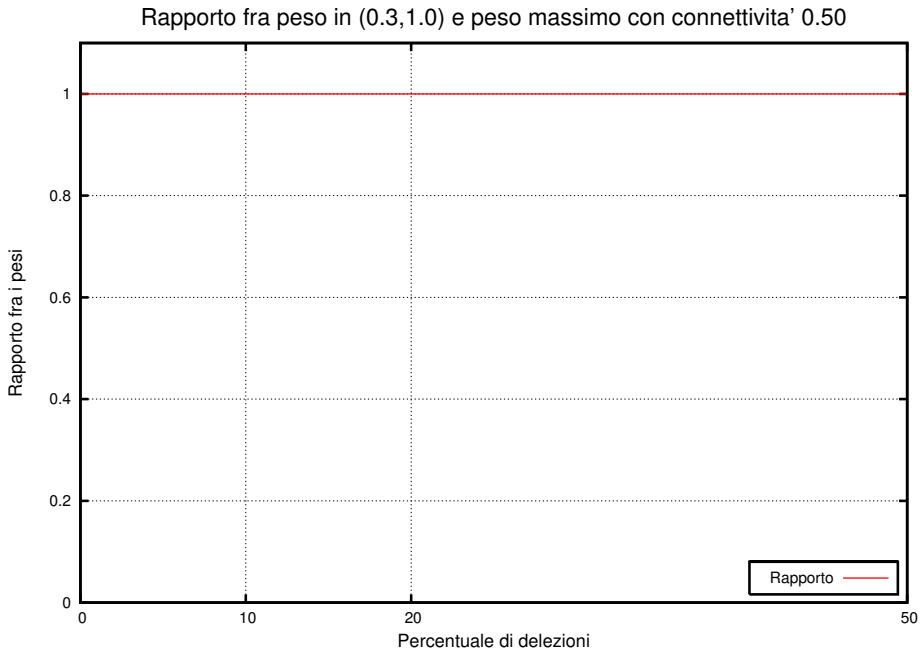

Figura 5.15: Peso relativo ottenuto con i parametri candidati senza rumore con connettività 0.50

Riassumendo i risultati ottenuti con connettività del 50% notiamo che il rapporto fra il peso dell'isomorfismo ottenuto con i valori candidati 0.3 e 1.0 dei parametri e il peso massimo trovato è sempre ottimale, ovvero l'algoritmo è sempre in gradi di individuare l'isomorfismo di peso massimo.

5.3.4 Variazione del grado di connettività con piccole perturbazioni

Eseguiamo ora una serie di test diversi: anzichè studiare il comportamento dell'algoritmo al variare dei parametri per determinate percentuali di delezioni, indagheremo su cosa accade in assenza di delezioni e con vari livelli di perturbazione introdotta sugli attributi di vertici ed archi.

Al solito produrremo delle perturbazioni gaussiane di media 0 e *sigma* rispettivamente 0.05 e 0.1. Considerando che i valori degli attributi variano da 0.1 a 1.0 tali valori introducono rispettivamente una perturbazione bassa e media senza però alterare radicalmente la somiglianza fra i due grafi.

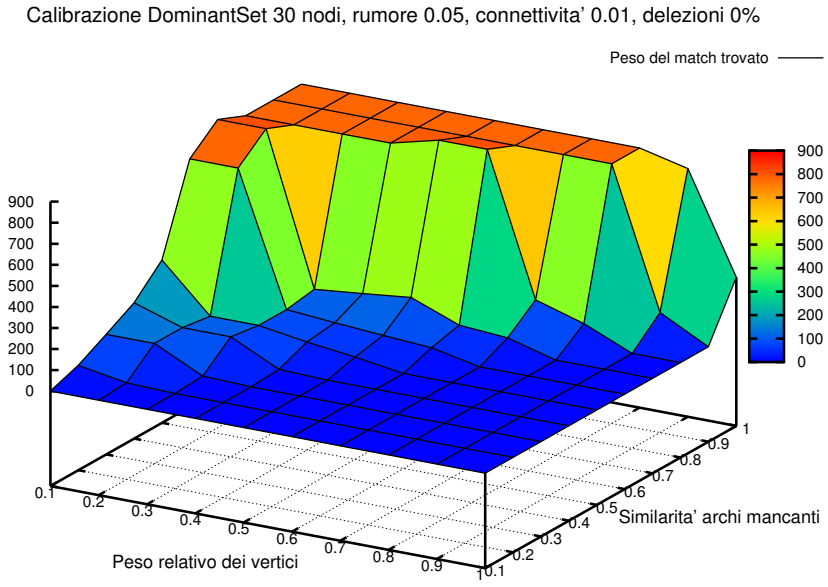

Figura 5.16: Calibrazione DominantSet 30 nodi, rumore 0.05, connettività 0.01, delezioni 0%

Dalla figura 5.16 possiamo notare che con bassa connettività il comportamento non è molto dissimile da quanto finora osservato e le stesse considerazioni valgono per i test fatti con connettività 10% e 50% rispettivamente nelle figure 5.17 e 5.18. L'esperimento di figura 5.17 presenta maggiore irregolarità, ma al solito fissando i parametri a 0.3 e 1.0 otteniamo un buon risultato.

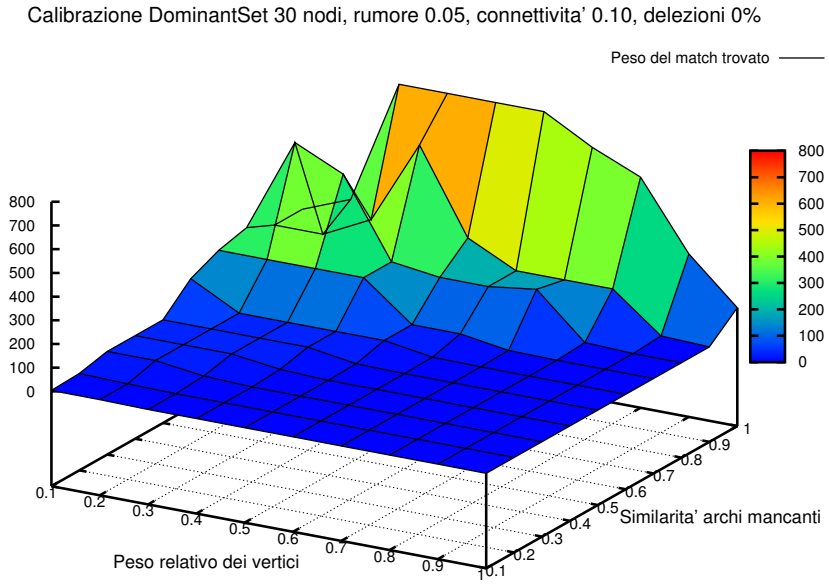

Figura 5.17: Calibrazione DominantSet 30 nodi, rumore 0.05, connettività 0.10, delezioni 0%

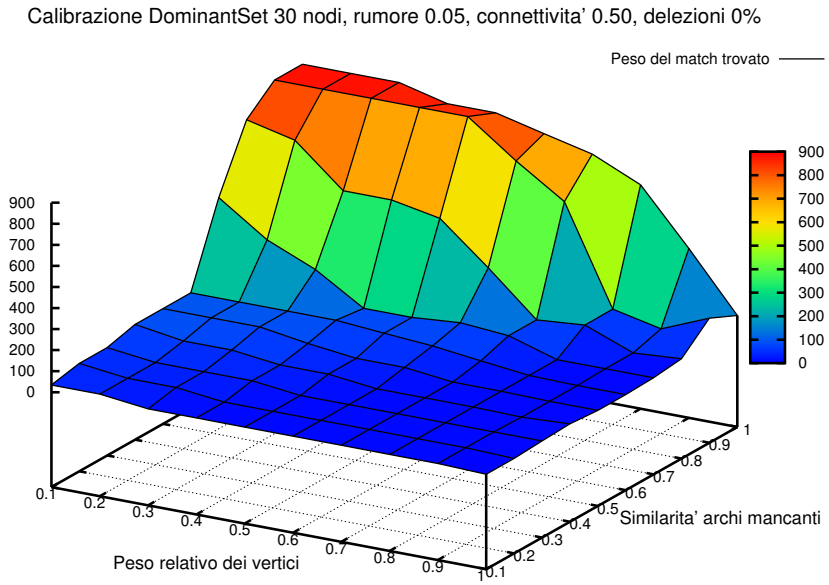

Figura 5.18: Calibrazione DominantSet 30 nodi, rumore 0.05, connettività 0.50, delezioni 0%

Figura 5.19: Peso relativo ottenuto con i parametri candidati senza delezioni e con rumore 0.05

Anche dopo questa serie di prove riportiamo l'andamento del rapporto fra il peso dell'isomorfismo trovato con i valori candidati 0.3 e 1.0 dei parametri e il peso massimo trovato in ciascuna prova. I parametri candidati non trovano sempre l'isomorfismo di peso massimo sembra che raggiungano sempre un risultato discreto e pertanto riteniamo di poterli considerare ancora accettabili.

5.3.5 Variazione del grado di connettività con perturbazioni medie

L'ultimo test in assoluto viene eseguito aumentando l'intensità delle perturbazioni sugli attributi di vertici ed archi raggiungendo un valore della deviazione standard di 0.1.

In queste condizioni il grafo di figura 5.20 mostra un massimo non coincidente con il valore ottenuto utilizzando i parametri candidati, tuttavia quest'ultimo non è di molto inferiore al massimo globale.

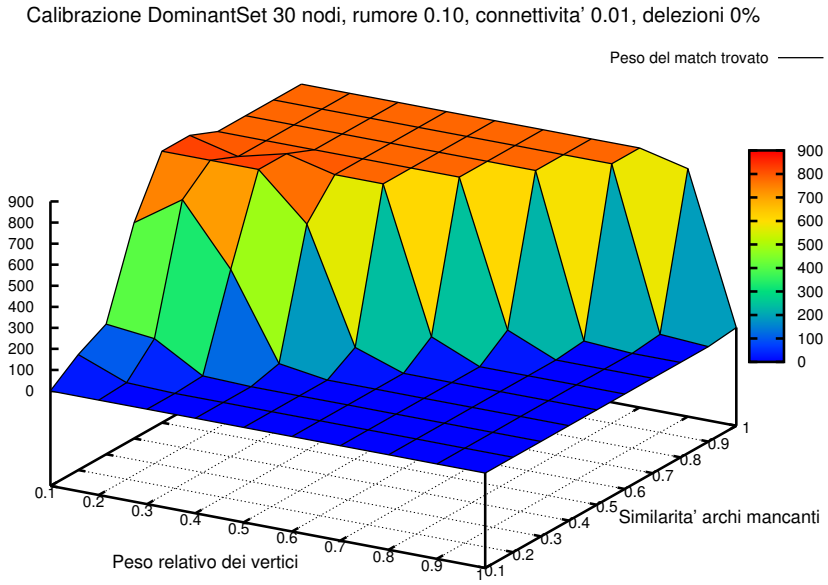

Figura 5.20: Calibrazione DominantSet 30 nodi, rumore 0.10, connettività 0.01, delezioni 0%

Nel grafico di figura 5.21 troviamo una situazione molto meno definita, come al solito probabilmente a causa dell'utilizzo di un unico campione. La figura 5.22 presenta invece una situazione più regolare ed in linea con quanto osservato finora.

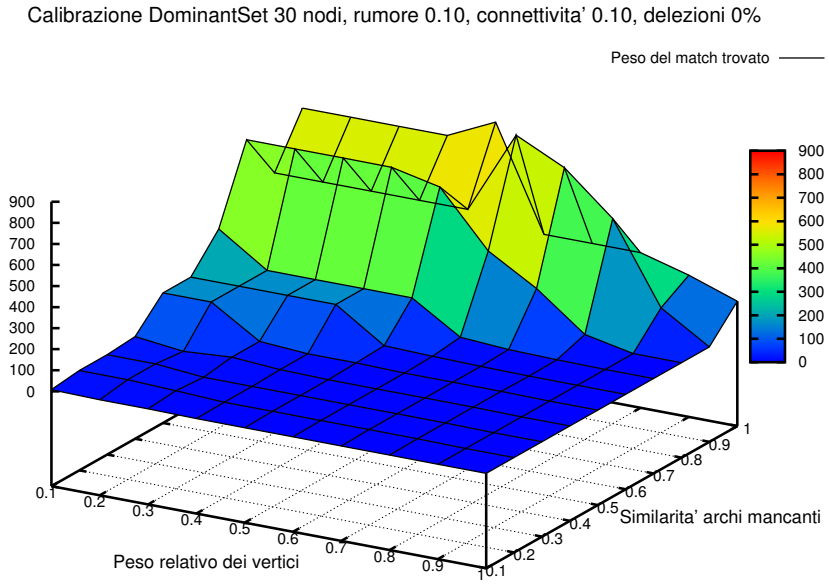

Figura 5.21: Calibrazione DominantSet 30 nodi, rumore 0.10, connettività 0.10, delezioni 0%

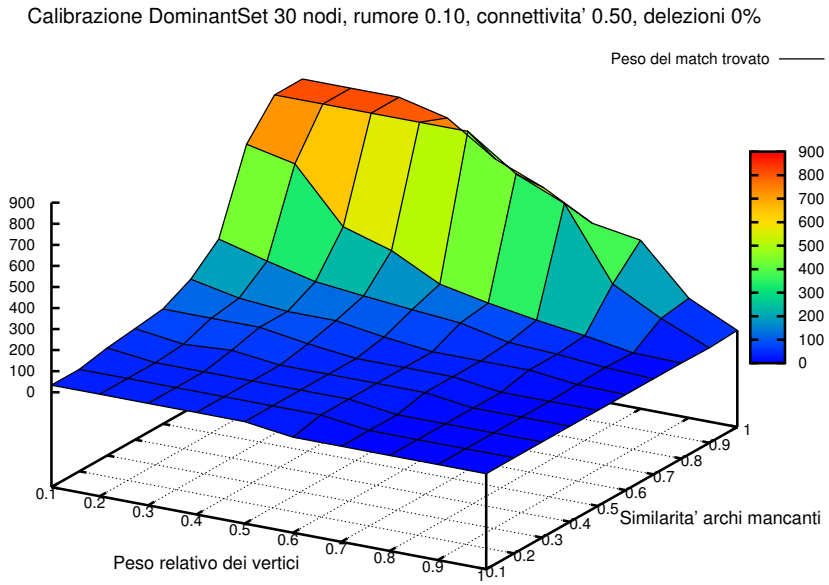

Figura 5.22: Calibrazione DominantSet 30 nodi, rumore 0.10, connettività 0.50, delezioni 0%

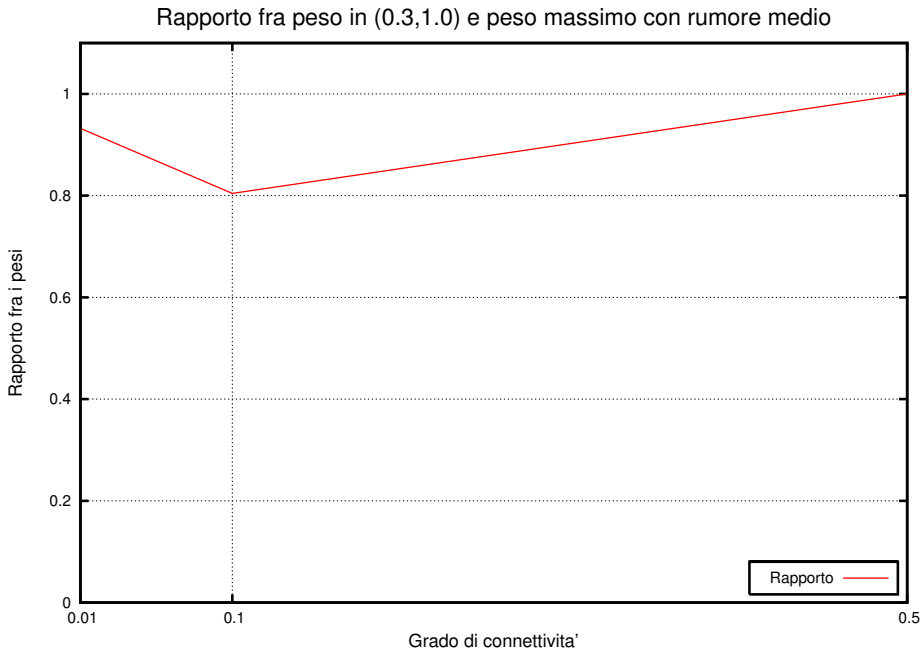

Figura 5.23: Peso relativo ottenuto con i parametri candidati senza delezioni e con rumore 0.10

Quest'ultima serie di prove presenta un andamento complessivo del rapporto fra il peso dell'isomorfismo trovato con i valori candidati 0.3 e 1.0 dei parametri e il peso massimo trovato in ciascuna prova non completamente soddisfacente. Tuttavia riteniamo questa condizione sufficientemente accettabile, soprattutto tenendo conto di tutte le prove fatte.

5.4 Confronto fra gli algoritmi di match

Dopo aver determinato i parametri empirici necessari per far operare al meglio l'algoritmo DominantSets possiamo procedere ad eseguire gli esperimenti centrali di questa tesi, ovvero il confronto fra i due algoritmi proposti e Graduated Assignment.

Gli algoritmi sono stati fatti eseguire su coppie di grafi di 30 nodi con diverse condizioni di connettività, rumore strutturale e perturbazione degli attributi. Poichè in questo caso la precisione è molto importante, ciascun test è stato effettuato su 10 coppie di grafi e qui riportiamo i valori medi corredati delle relative barre di errore (di raggio pari alla varianza dei campioni).

Per ciascun esperimento vengono riportati tre grafici

- *Peso relativo del match trovato*: viene riportato il rapporto percentuale fra il valore totale di similarità del match trovato e quello del match ottimale, misurati secondo la funzione peso descritta nel paragrafo 5.1. Il match ottimale viene determinato come il match fra i nodi corrispondenti fra i due grafi prima delle delezioni e dell'introduzione delle perturbazioni sugli attributi;

- *Percentuale di isomorfismi*: Poichè GraduatedAssignment non restituisce sempre isomorfismi, per un corretto paragone degli algoritmi è stato necessario riportare anche questo istogramma che indica nelle diverse condizioni qual'è la percentuale di isomorfismi corretti trovati da GraduatedAssignment;

- *Tempo di convergenza*: viene riportato il tempo di convergenza (in millisecondi) dei diversi metodi;

Per quanto riguarda DominantSets e EdgesAssociationGraph il metodo è stato fatto terminare una volta che la norma assoluta fra due soluzioni successive fosse risultata inferiore ad un ϵ fissato per questi esperimenti nel valore 10^{-6}.

A causa della complessità computazionale e i requisiti di memoria largamente superiori rispetto alle altre due tecniche di EdgesAssociatonGraph, quest'ultima è stata testata solo fino a un grado di connettività di 0.1: tale test dovrebbe comunque essere sufficiente ad evidenziare il suo comportamento. Si noti anche che mentre i primi due grafi di ciascun esperimento hanno il solo asse x in scala logaritmica, il terzo presenta questa scala in entrambi gli assi, proprio per permettere di rappresentare correttamente i tempi di esecuzione di ordini di grandezza superiori di EdgesAssociationGraph.

5.4.1 Grafi privi di perturbazione degli attributi e di rumore strutturale

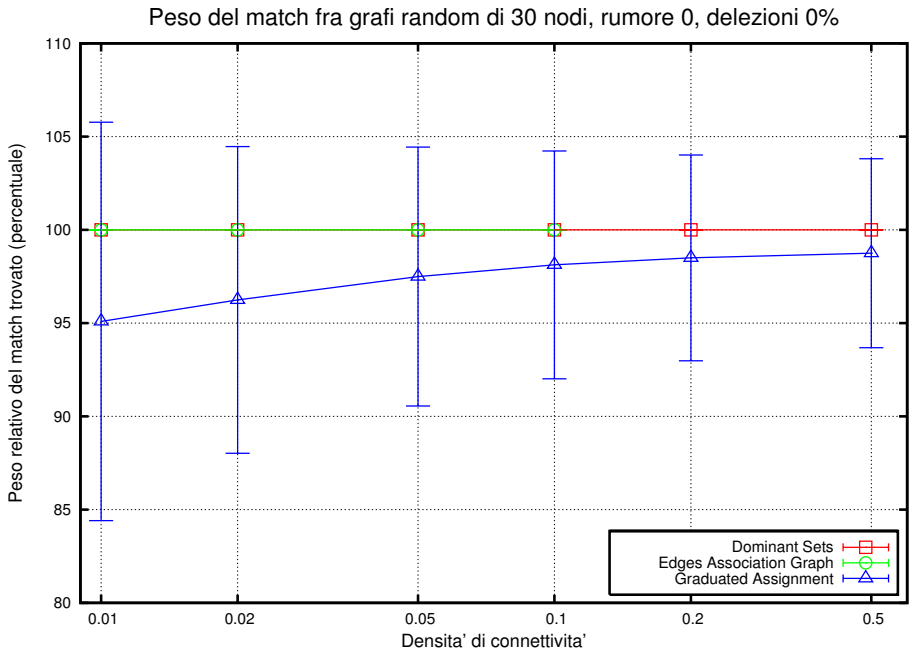

Figura 5.24: Peso del match fra grafi di 30 nodi, rumore 0.0, delezioni 0%

Quando confrontati su grafi identici DominatSets e EdgesAssociationGraph trovano sempre il match ottimo, mentre GraduatedAssignment produce risultati solo leggermente meno soddisfacenti, specialmente a bassi gradi di connettività, la qual cosa è in qualche modo prevedibile in quanto tale algoritmo è noto per funzionare al meglio ad alte connettività. Per contro, come mostrato nell'istogramma di figura 5.25, tutti i match trovati da GraduatedAssignment in queste condizioni sono anche isomorfismi.

Dal grafo di figura 5.26, che riporta il tempo necessario per la convergenza di ciascun algoritmo, notiamo che GraduatedAssignment è di gran lunga la tecnica più rapida, grazie al suo basso costo computazionale e al piccolo numero di iterazioni che deve compiere per convergere. Come atteso DominantSet risulta più lento e EdgesAssociationGraph estremamente lento, in virtù della complessità computazionale polinomiale di ordine particolarmente elevato che lo caratterizza.

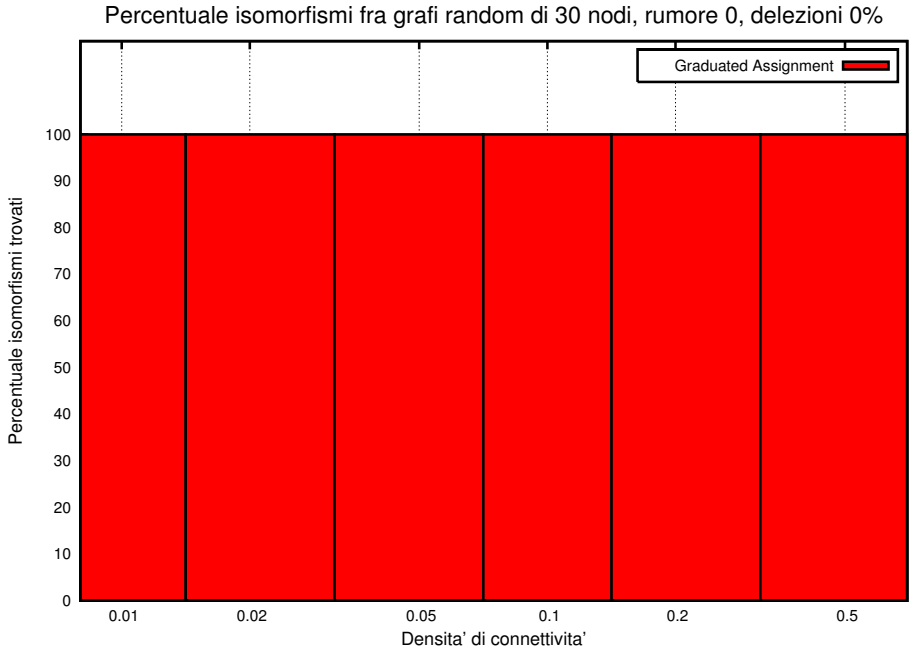

Figura 5.25: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.0, delezioni 0%

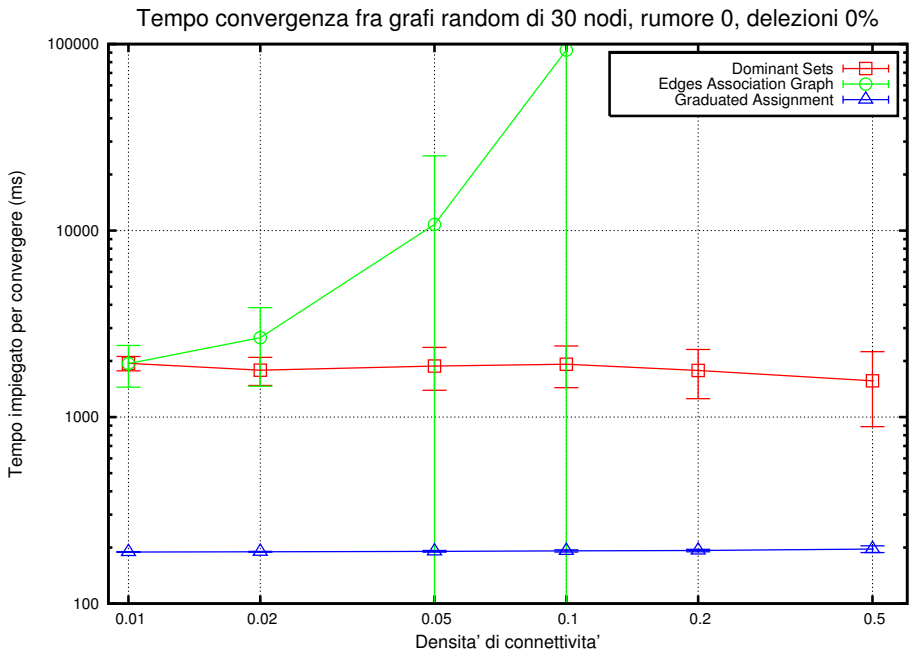

Figura 5.26: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.0, delezioni 0%

5.4.2 Grafi privi di perturbazione degli attributi e con rumore strutturale 10%

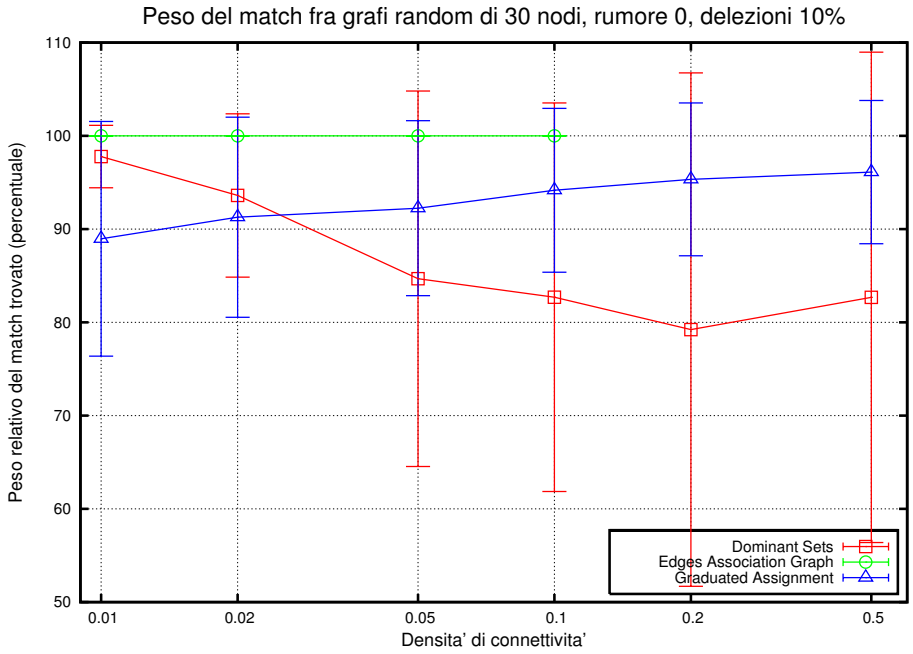

Figura 5.27: Peso del match fra grafi di 30 nodi, rumore 0.0, delezioni 10%

Con l'aggiunta di rumore strutturale lo scenario comincia ad acquisire un nuovo aspetto: il comportamento di EdgesAssociationGraph e di GraduatedAssignment non si modifica drasticamente [2], ma DominantSets, pur presentando buone prestazioni a basse connettività, comincia a degradare rapidamente a gradi di connettività più alti. Questo fenomeno è probabilmente legato alla differenza che sussiste fra una clique di massimo peso ed un Dominant Set (ovvero una clique di massima coerenza) nel grafo di associazione: l'aggiunta di rumore strutturale potrebbe creare dei Dominant Set che non corrispondono a clique di peso massimo, ma che vengono individuati dall'algoritmo (come accennato nel paragrafo 3.6).

Per quanto riguarda GraduatedAssignment notiamo in figura 5.28 che a basse connettività comincia a trovare qualche match che non rappresenta un isomorfismo. I tempi di convergenza risultano sostanzialmente nella stessa relazione del test precedente, come prevedibile.

[2] non deve trarre in inganno il minor valore del peso massimo, che è dovuto semplicemente al minore numero di vertici coinvolti nel match a causa delle delezioni

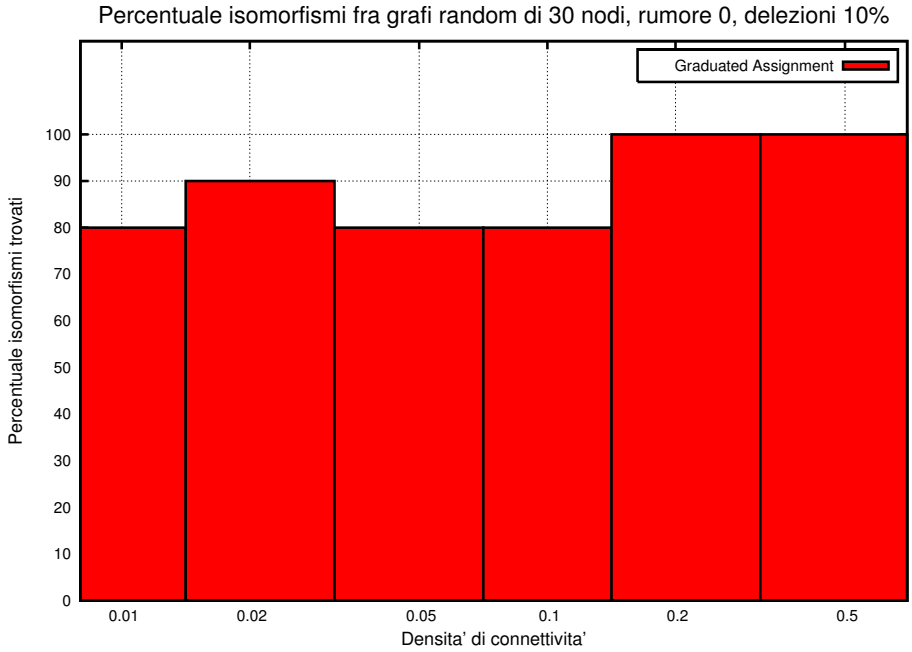

Figura 5.28: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.0, delezioni 10%

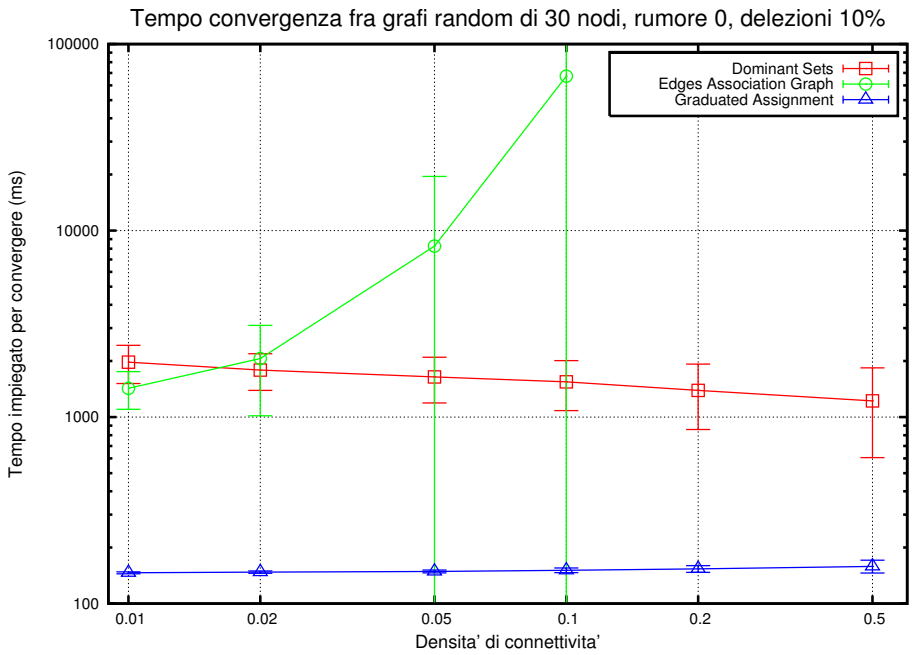

Figura 5.29: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.0, delezioni 10%

5.4.3 Grafi privi di perturbazione degli attributi e con rumore strutturale 20%

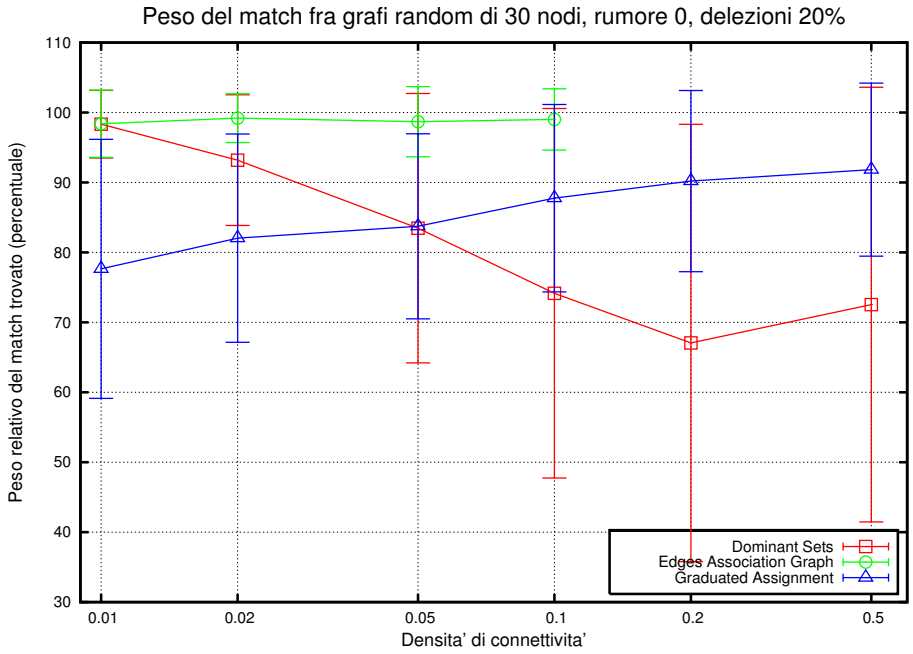

Figura 5.30: Peso del match fra grafi di 30 nodi, rumore 0.0, delezioni 20%

Aumentando la percentuale di delezioni il confronto fra gli algoritmi proposti non cambia in modo apprezzabile, se non forse per DominantSets il cui peggioramento all'aumentare del grado di connettività del grafo risulta leggermente più marcato.

Per quanto riguarda GraduatedAssignment non notiamo delle modifiche particolari al comportamento dal punto di vista del peso complessivo del match trovato, tuttavia se osserviamo l'istogramma in figura 5.31 ci accorgiamo che, soprattutto a basse connettività, l'algoritmo riesce a trovare molti meno match che corrispondono ad isomorfisi. A seconda dell'applicazione questo può o meno essere un fattore vincolante.

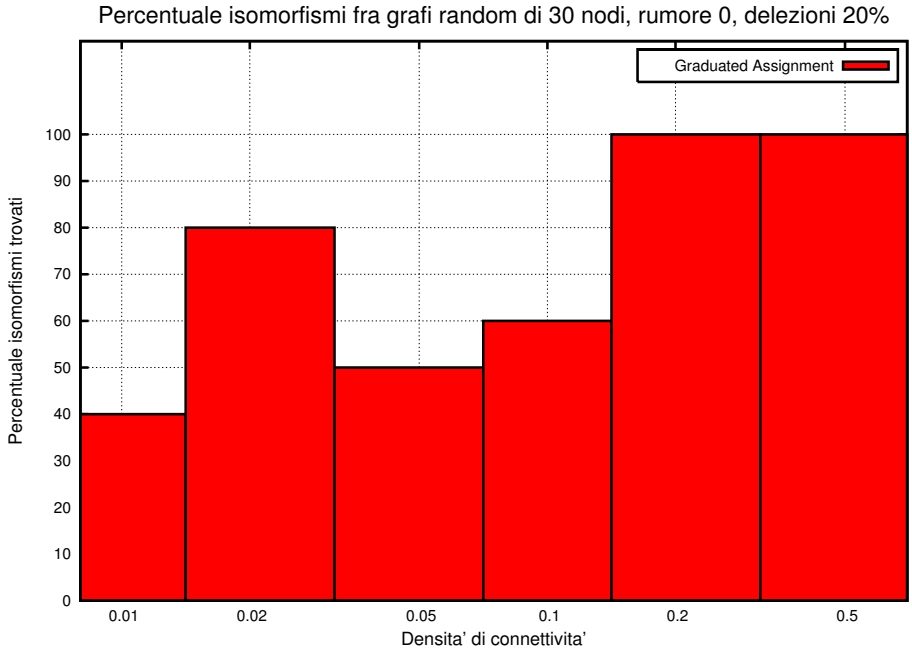

Figura 5.31: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.0, delezioni 20%

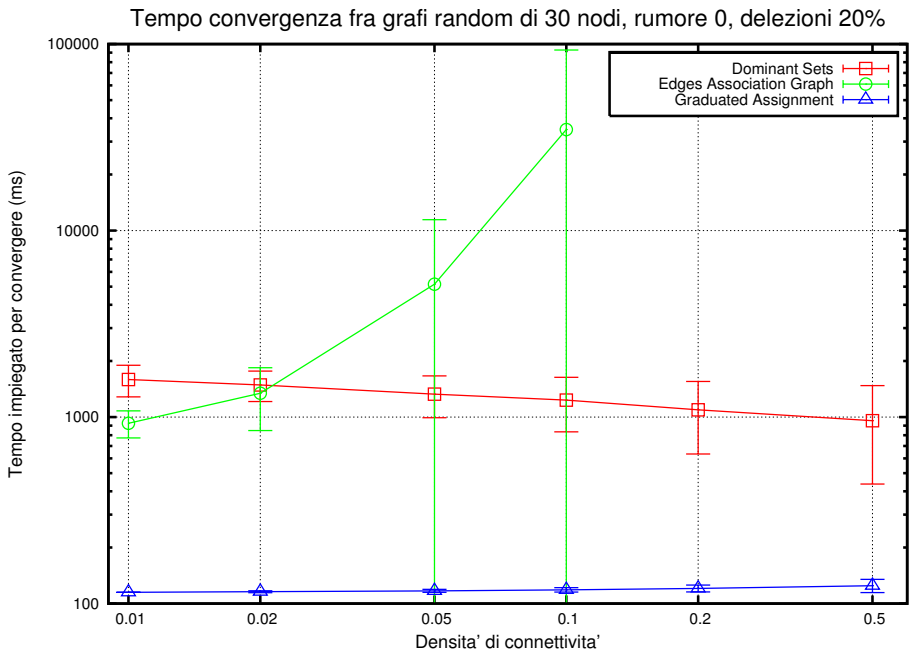

Figura 5.32: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.0, delezioni 20%

5.4.4 Grafi privi di perturbazione degli attributi e con rumore strutturale 40%

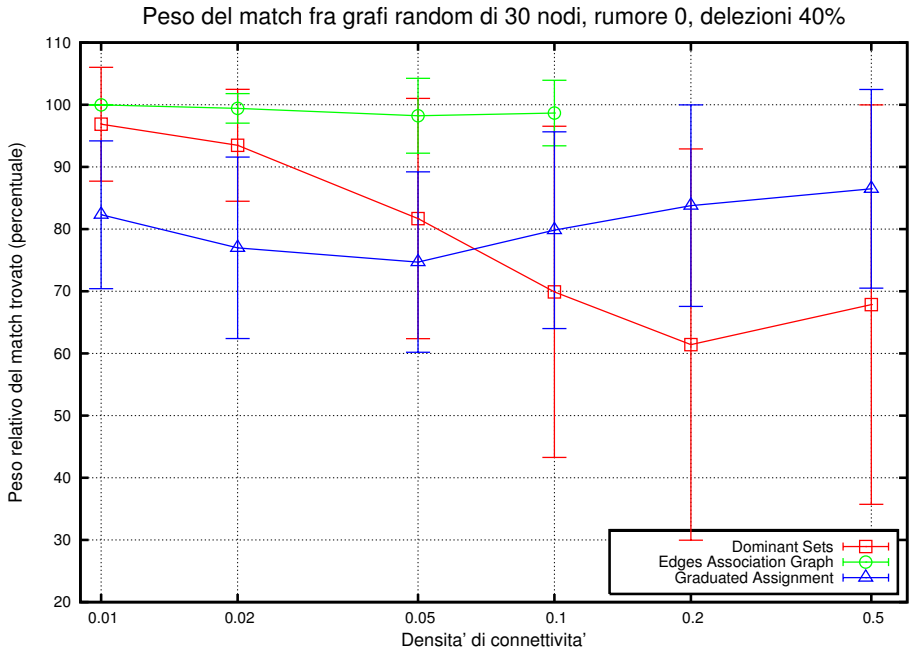

Figura 5.33: Peso del match fra grafi di 30 nodi, rumore 0.0, delezioni 40%

Quando il numero di vertici eliminati comincia a diventare piuttosto alto non notiamo ancora variazioni nel comportamento dei singoli algoritmi, almeno dal punto di vista del peso totale del match trovato da ciascuno. In questo caso però, osservando l'istogramma in figura 5.34 notiamo un drastico peggioramento per quanto riguarda la capacità di GraduatedAssignment di trovare isomorfismi validi, richiedendo addirittura una connettività del 50% prima di poter trovare sempre isomorfismi nei nostri test. Questo comportamento era atteso, infatti viene evidenziato anche nell'articolo originale di Gold e Rangarajan [4].

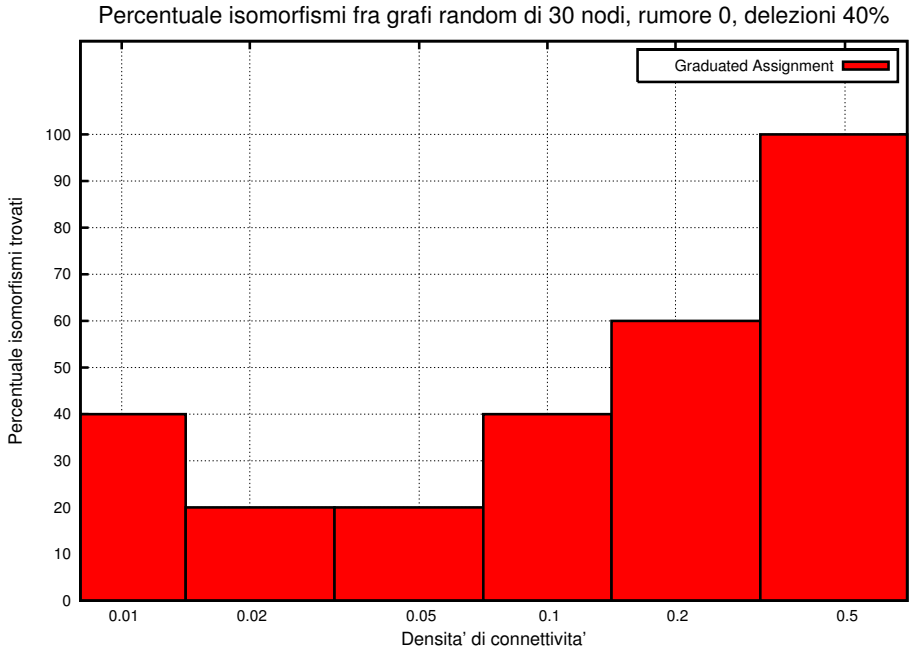

Figura 5.34: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.0, delezioni 40%

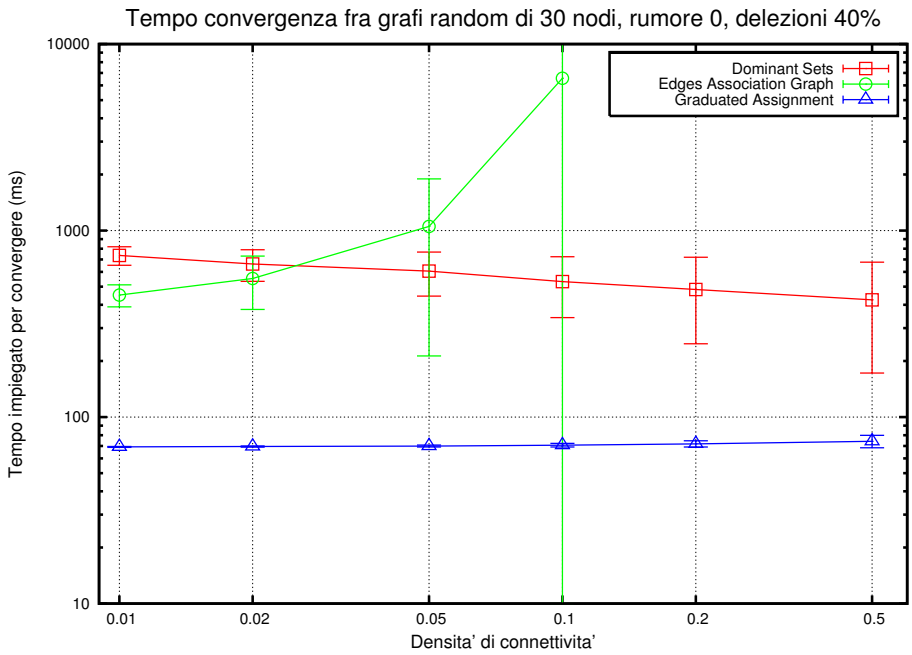

Figura 5.35: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.0, delezioni 40%

5.4.5 Grafi con rumore di deviazione standard 0.05 e senza rumore strutturale

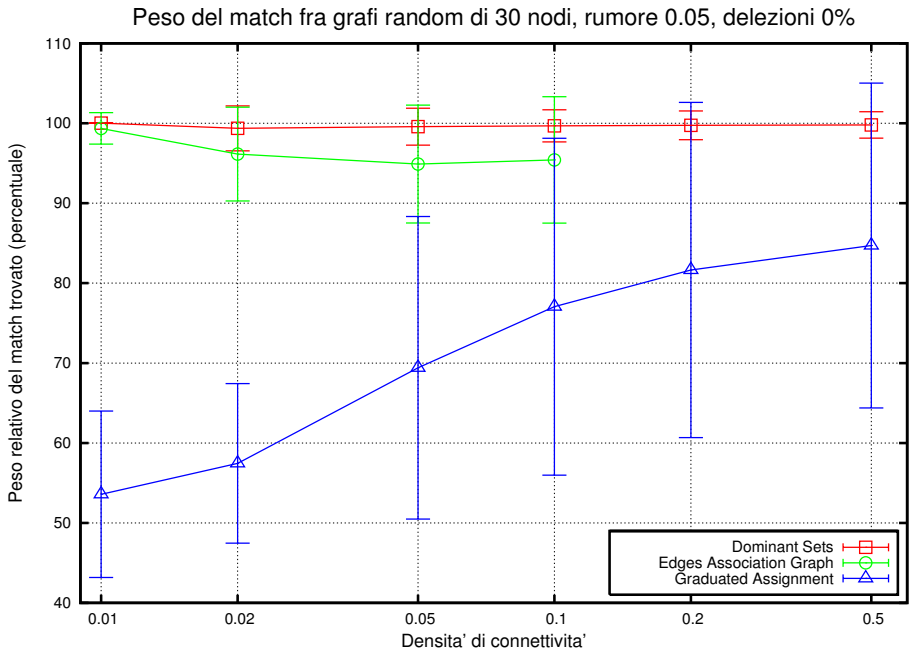

Figura 5.36: Peso del match fra grafi di 30 nodi, rumore 0.05, delezioni 0%

Eseguendo i test senza rumore strutturale e con delle perturbazioni gaussiane sugli attributi con una deviazione standard di 0.5 notiamo che DominantSets performa molto bene, addirittura meglio di EdgesAssociationGraph. Da questo fatto possiamo dedurre che l'algoritmo risulta molto più sensibile alle delezioni che non alle perturbazioni sugli attributi (almeno per piccole perturbazioni) questo probabilmente perchè essendo comunque i valori degli attributi piuttosto vicini ed il rumore a media 0 le clique del grafo di associazione restano composte da archi di peso vicino, favorendo clique coerenti derivate da associazione di vertici originariamente uguali.

Per quanto riguarda GraduatedAssignment non solo notiamo una performance iniziale leggermente inferiore al caso di grafi senza delezioni e senza perturbazioni, ma anche osservando l'istogramma 5.37 appare evidente come a basse connettività ed in presenza di rumore sia impossibile o molto difficile per questo algoritmo trovare isomorfismi validi.

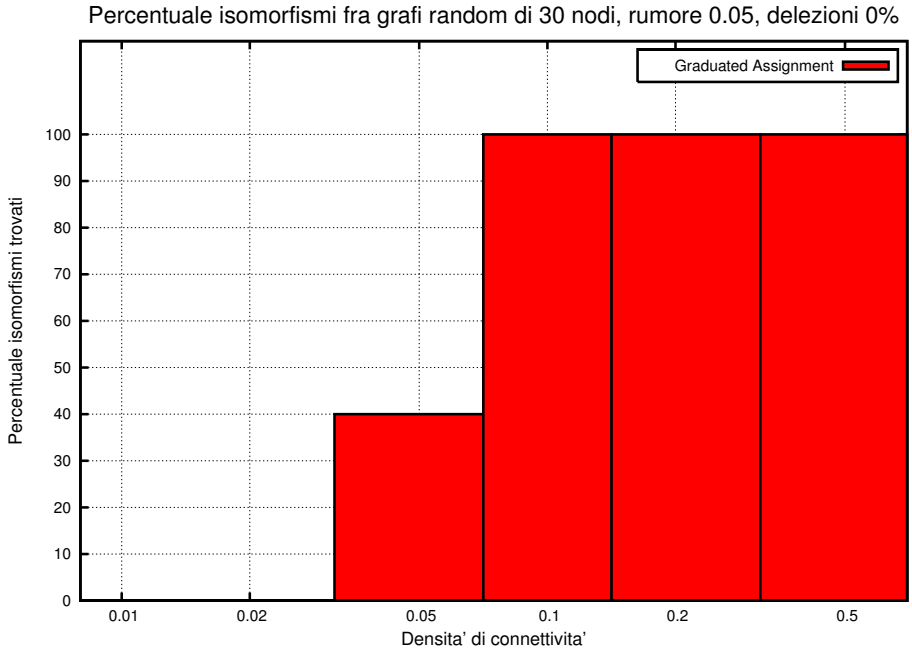

Figura 5.37: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.05, delezioni 0%

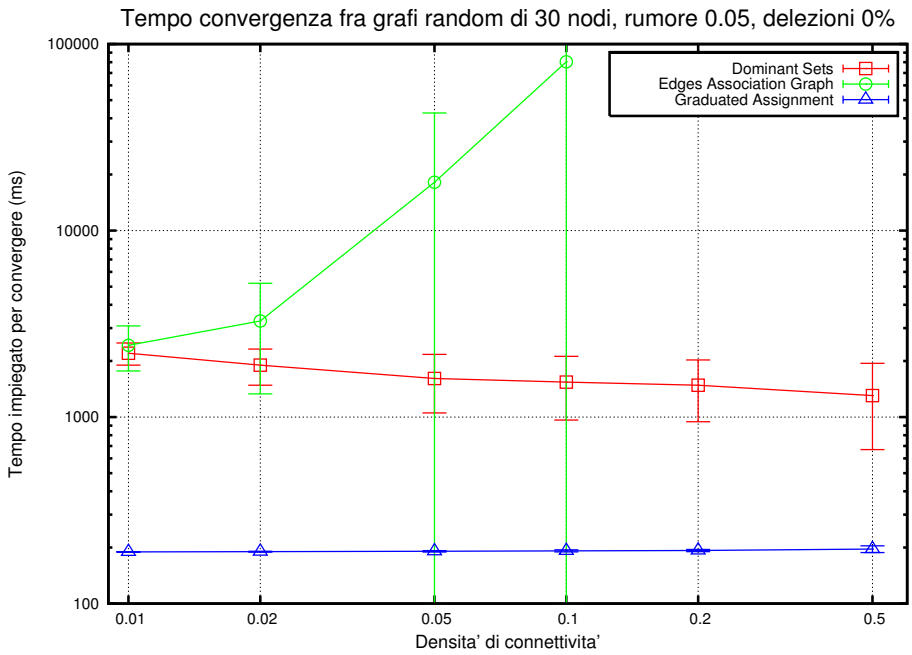

Figura 5.38: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.05, delezioni 0%

5.4.6 Grafi con rumore di deviazione standard 0.05 e rumore strutturale 10%

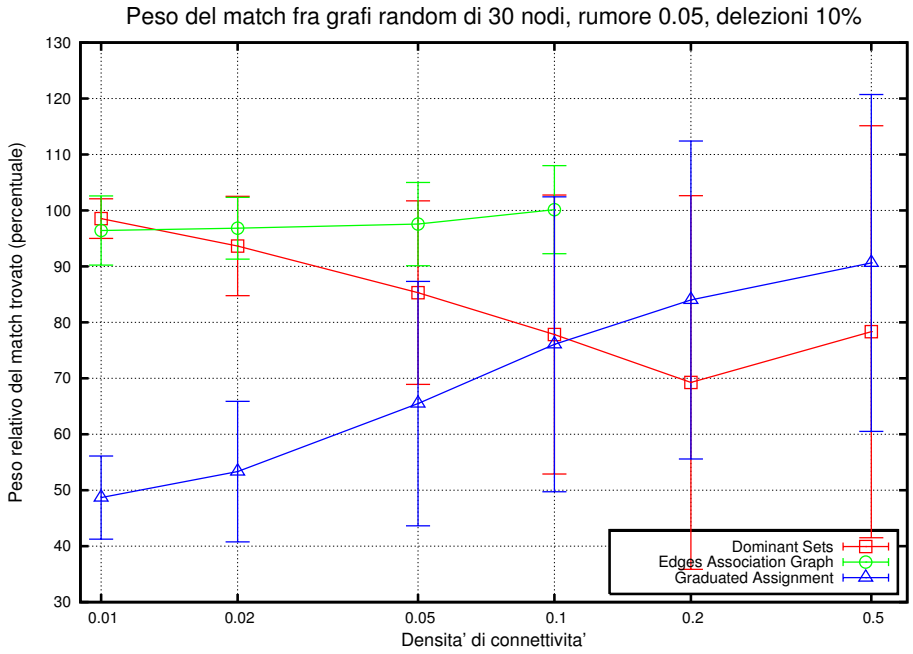

Figura 5.39: Peso del match fra grafi di 30 nodi, rumore 0.05, delezioni 10%

Appena introduciamo le delezioni, come previsto, DominantSets peggiora rapidamente con l'aumentare della connettività del grafo. Possiamo comune notare come, rispetto al caso senza rumore, GraduatedAssignment richieda una connettività leggermente più alta prima di diventare migliore di DominantSets. La capacità di GraduatedAssignment di trovare isomorfismi è sempre molto bassa a basse connettività.

La qualità di EdgesAssociationGraph rimane invece sorprendentemente alta, almeno fino al grado di connettività che ci è stato possibile testare. Si noti che anche il suo errore, rappresentato da barre con raggio pari alla deviazione standard del campione, è relativamente contenuto.

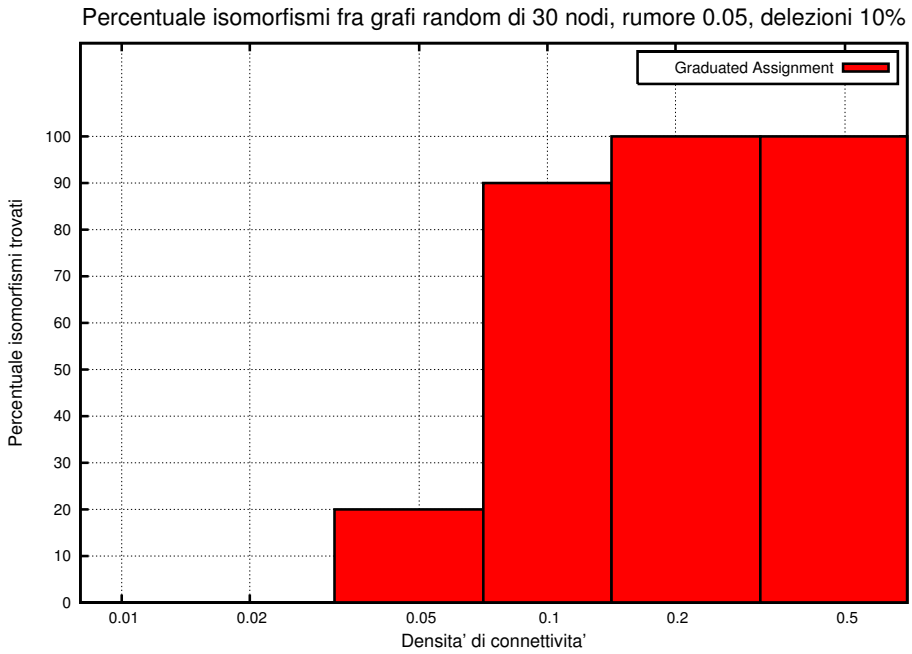

Figura 5.40: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.05, delezioni 10%

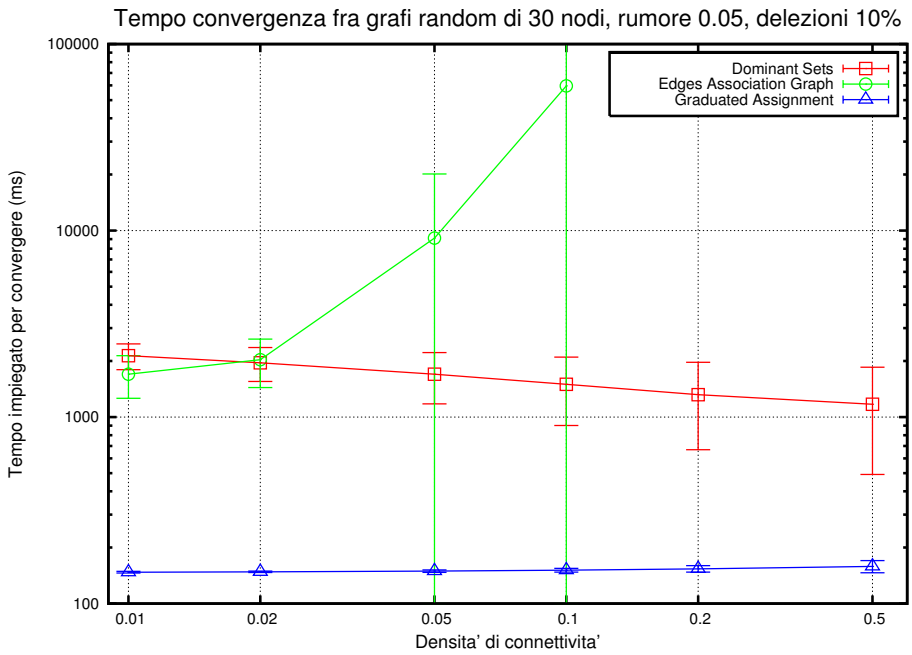

Figura 5.41: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.05, delezioni 10%

5.4.7 Grafi con rumore di deviazione standard 0.05 e rumore strutturale 20%

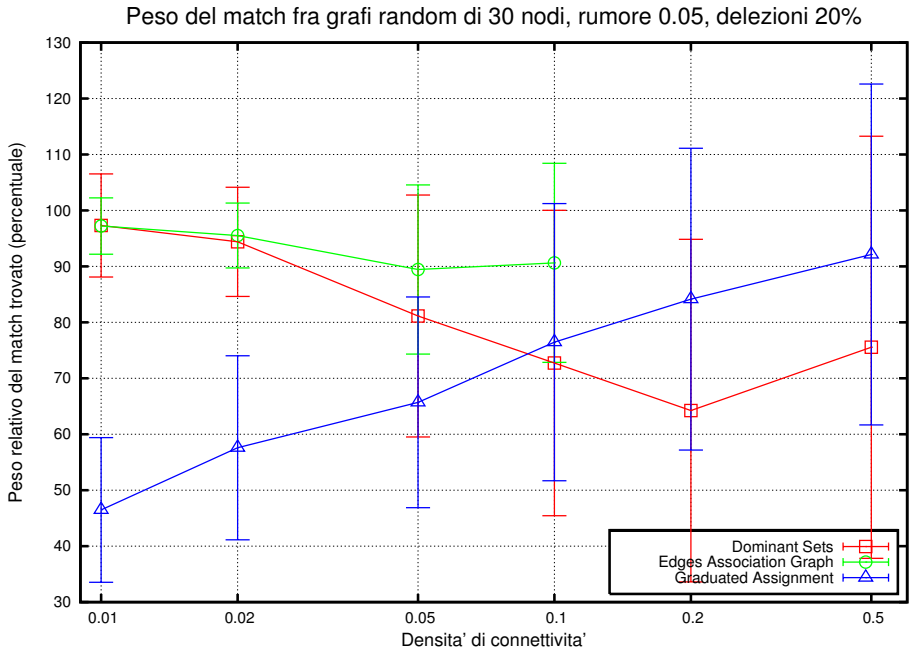

Figura 5.42: Peso del match fra grafi di 30 nodi, rumore 0.05, delezioni 20%

Aumentando il numero di delezioni al 20% per la prima volta EdgesAssociationGraph comincia a peggiorare le prestazioni all'aumentare della connettività, rimanendo comunque superiore sia a DominantSets che a GraduatedAssignment.

Quest'ultimo come al solito ha un comportamento piuttosto cattivo a basse connettività, riuscendo raramente a individuare isomorfismi corretti.

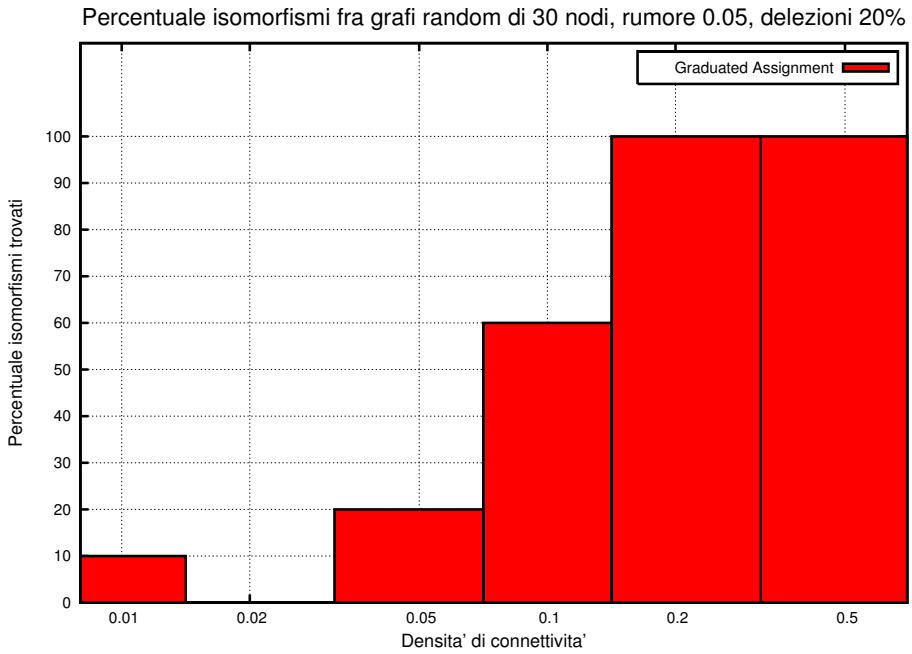

Figura 5.43: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.05, delezioni 20%

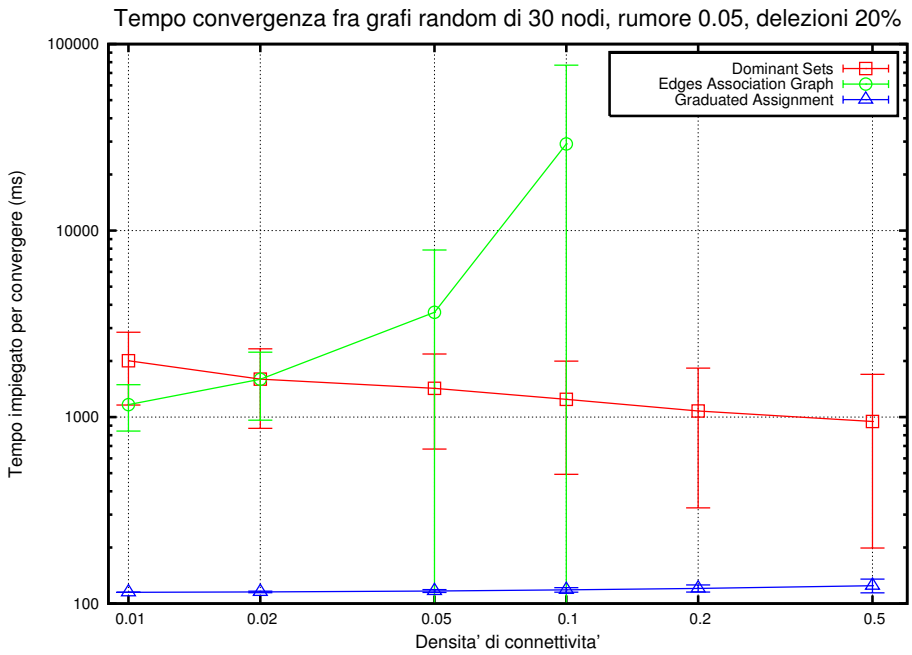

Figura 5.44: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.05, delezioni 20%

5.4.8 Grafi con rumore di deviazione standard 0.05 e rumore strutturale 40%

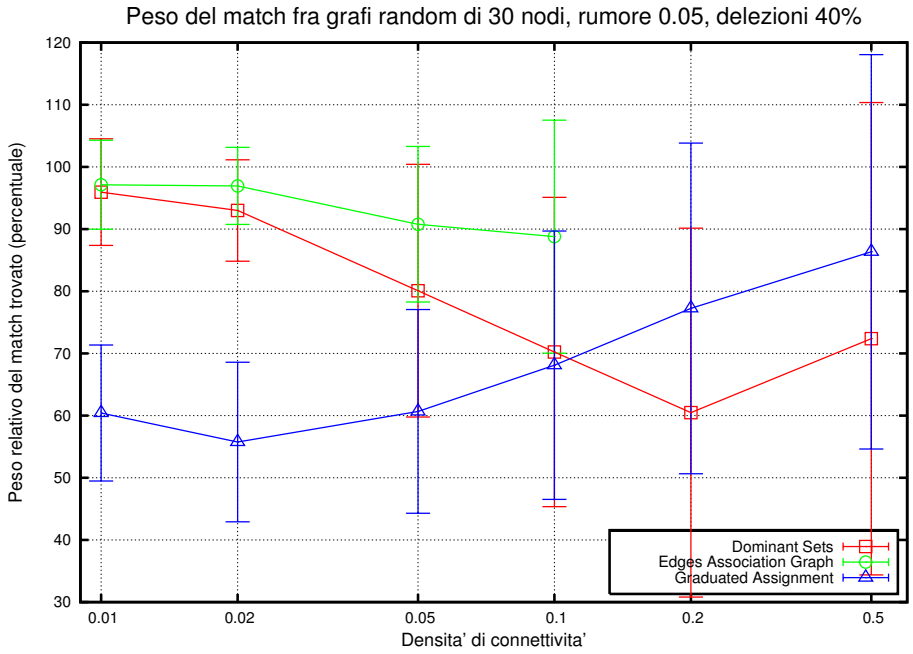

Figura 5.45: Peso del match fra grafi di 30 nodi, rumore 0.05, delezioni 40%

Portando al 40% il numero di delezioni lo scenario rimane sostanzialmente invariato, fatta salva l'osservazione di un ulteriore peggioramento nella capacità di GraduatedAssignment di trovare isomorfismi corretti, come del resto accadeva nello stesso test in assenza di rumore.

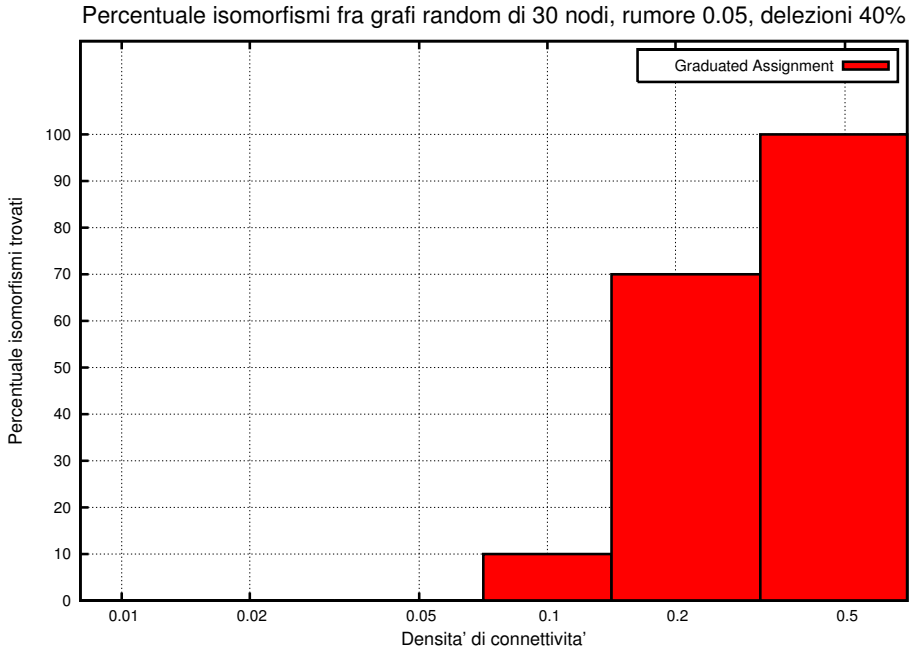

Figura 5.46: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.05, delezioni 40%

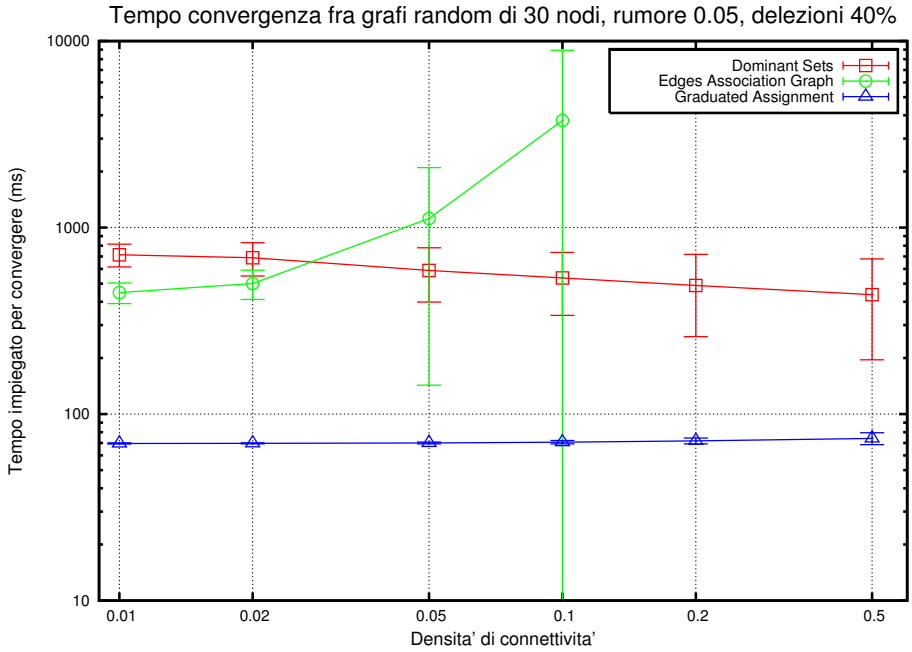

Figura 5.47: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.05, delezioni 40%

5.4.9 Grafi con rumore di deviazione standard 0.1 e senza rumore strutturale

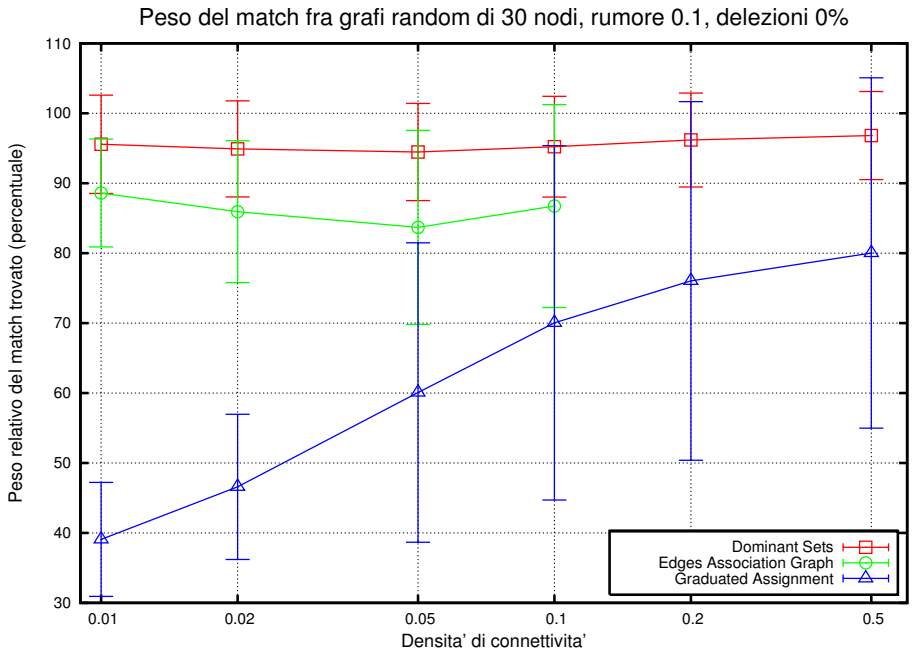

Figura 5.48: Peso del match fra grafi di 30 nodi, rumore 0.1, delezioni 0%

Applicando una perturbazione gaussiana sugli attributi con deviazione standard 0.1 notiamo che per la prima volta DominantSets performa meglio di EdgesAssociationGraph, confermando l'idea che questo algoritmo sia più sensibile alle delezioni che alle perturbazioni degli attributi. Questo comportamento è evidentemente inverso per EdgesAssociationGraph la cui qualità comincia a decadere con l'aumento del rumore.

Si può ipotizzare che questo sia dovuto ad una maggiore tendenza, anche a causa della dimensione maggiore del simplesso utilizzato, delle dinamiche di replicazione di individuare soluzioni locali.

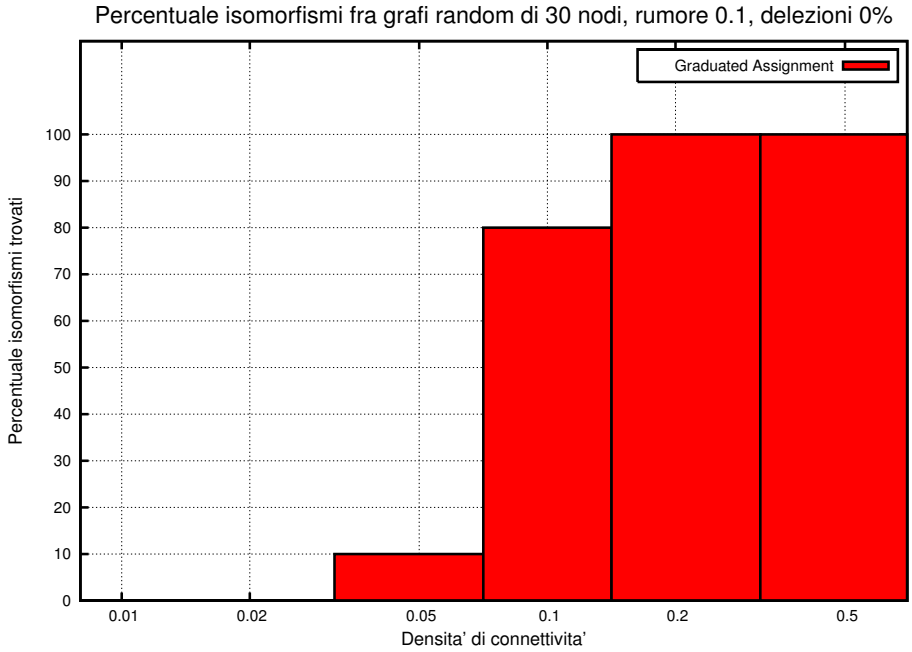

Figura 5.49: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.1, delezioni 0%

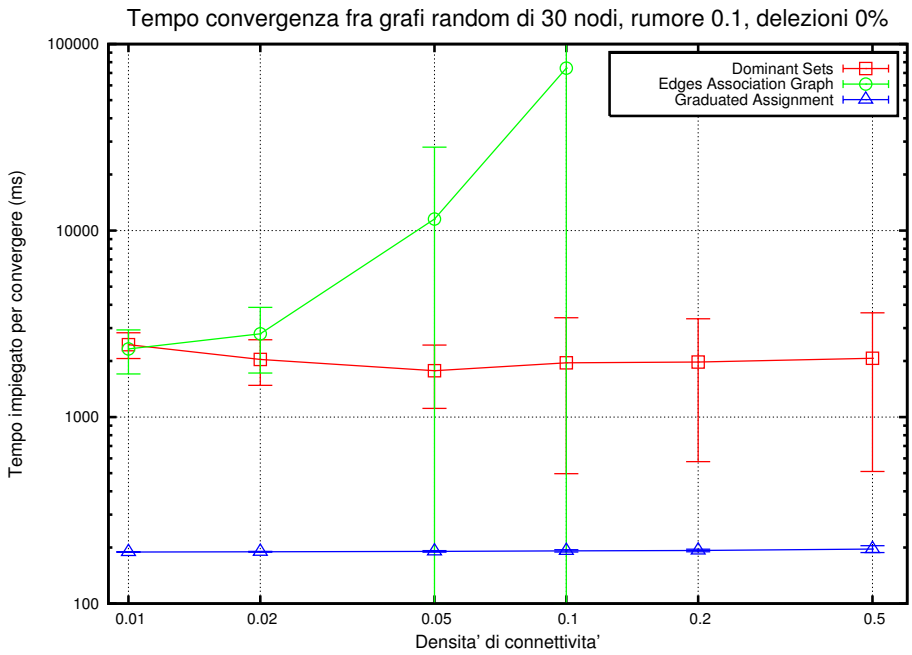

Figura 5.50: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.1, delezioni 0%

5.4.10 Grafi con rumore di deviazione standard 0.1 e rumore strutturale 10%

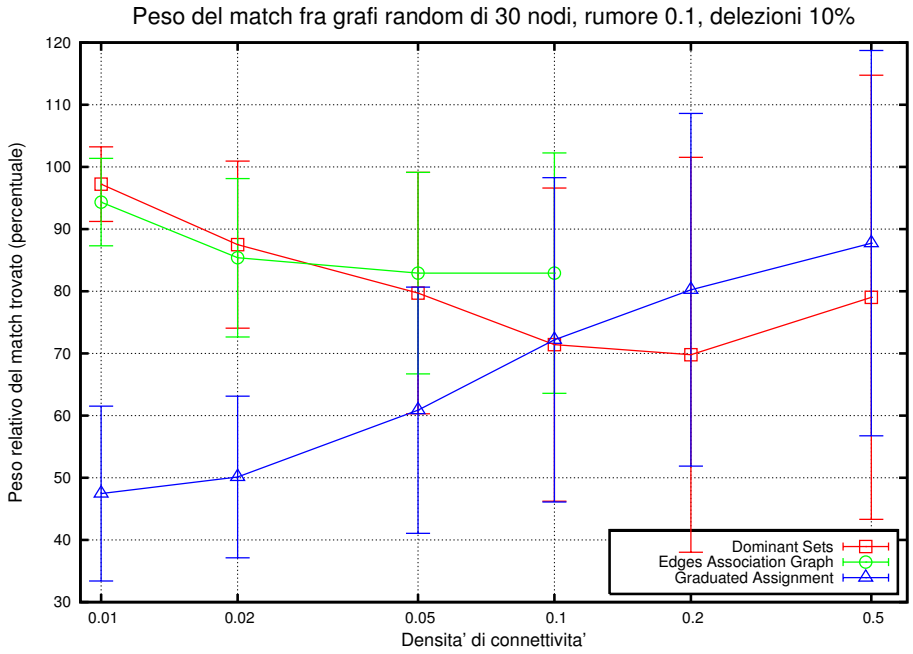

Figura 5.51: Peso del match fra grafi di 30 nodi, rumore 0.05, delezioni 10%

Aggiungendo delezioni in questa condizione il peggioramento di DominantSets lo rende sostanzialmente comparabile con EdgesAssociationGraph, almeno fino ad un certo grado di connettività dove questo torna ad essere migliore.

GraduatedAssignment mantiene il medesimo comportamento, sia in termini di peso totale del match trovato, sia della capacità di trovare isomorfismi.

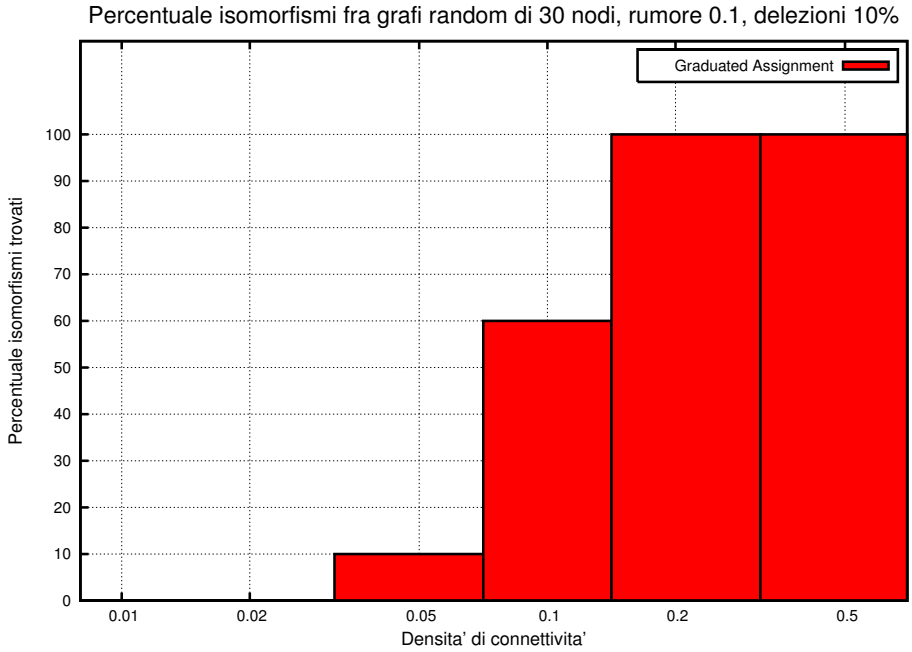

Figura 5.52: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.05, delezioni 10%

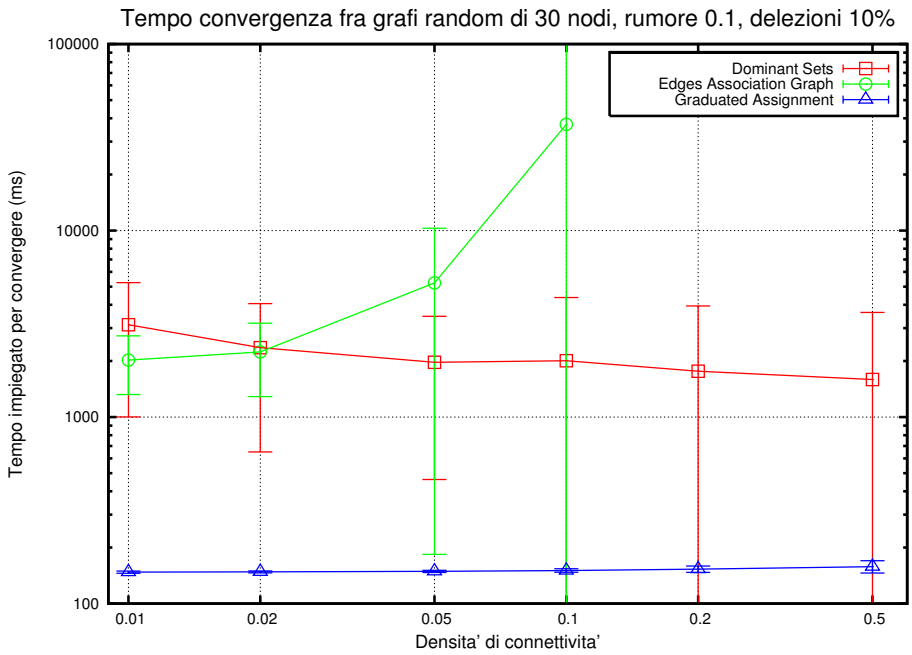

Figura 5.53: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.05, delezioni 10%

5.4.11 Grafi con rumore di deviazione standard 0.1 e rumore strutturale 20%

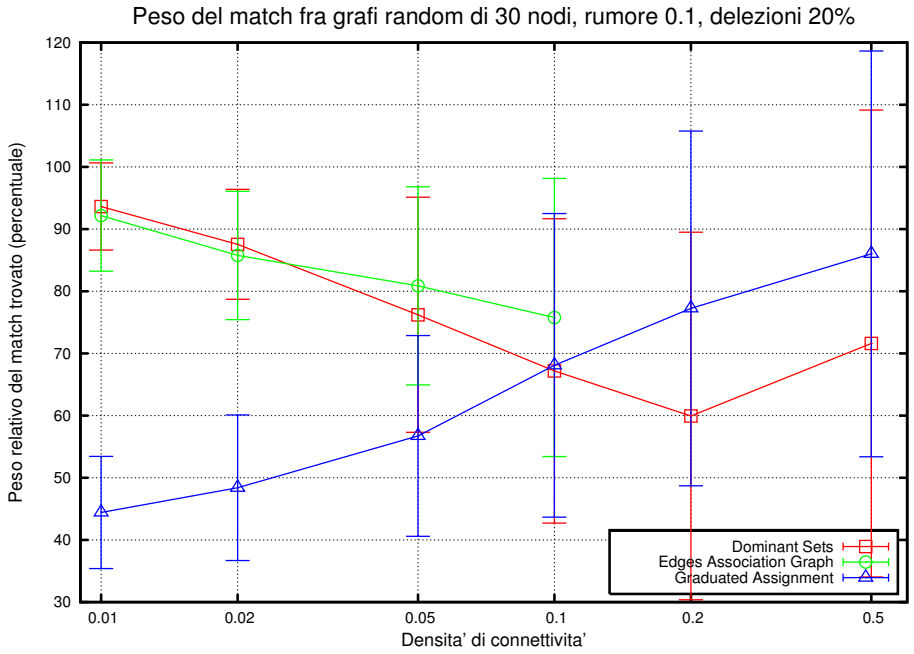

Figura 5.54: Peso del match fra grafi di 30 nodi, rumore 0.05, delezioni 20%

Aumentando ancora le delezioni EdgesAssociationGraph peggiora leggermente e comincia a dare l'impressione di poter diventare peggiore di GraduatedAssignment per grandi valori di connettività. Il suo comportamento rimane comunque ottimo a basse connettività, anche se comparabile con DominatSets.

La capacità di GraduatedAssigment di trovare isomorfismi esatti a basse connettività diventa ancora più bassa.

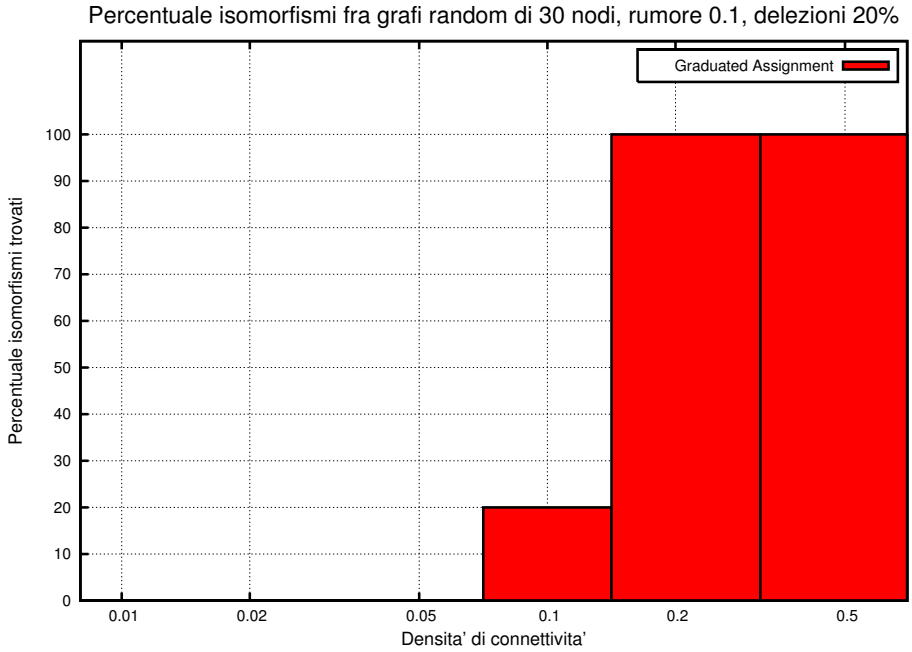

Figura 5.55: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.05, delezioni 20%

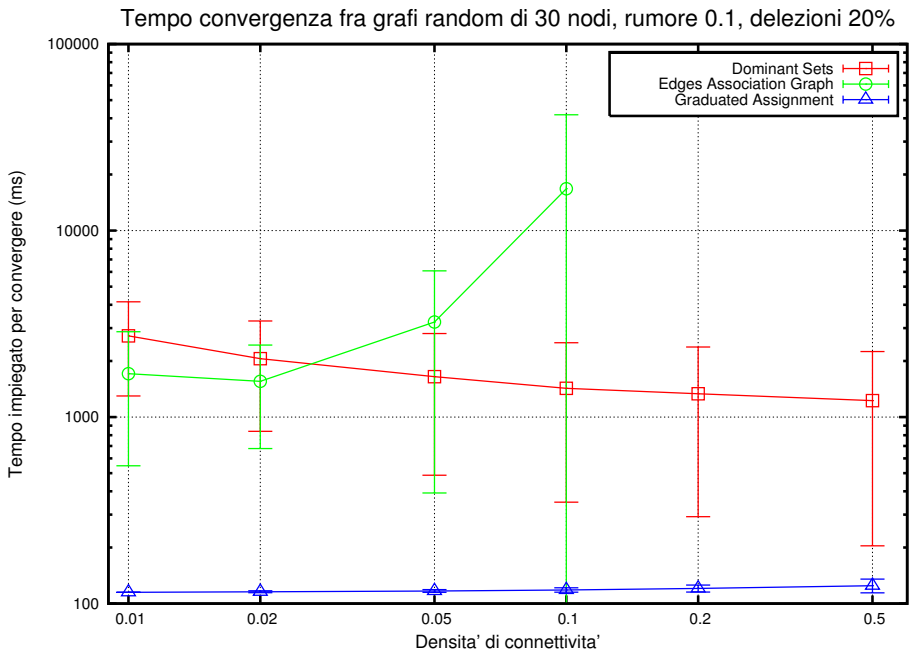

Figura 5.56: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.05, delezioni 20%

5.4.12 Grafi con rumore di deviazione standard 0.1 e rumore strutturale 40%

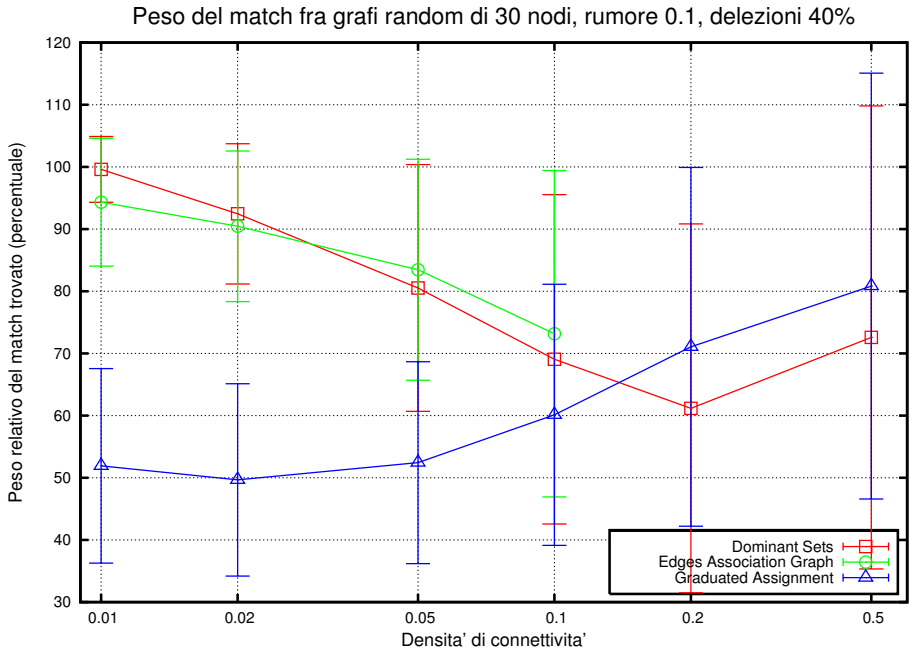

Figura 5.57: Peso del match fra grafi di 30 nodi, rumore 0.05, delezioni 40%

Con una percentuale di delezioni del 40% ed un rumore gaussiano con deviazione standard di 0.1 DominantSets e EdgesAssociationGraph diventano comparabili.

GraduatedAssignment resta poco performante a basse connettività, mentre tende a superare gli altri due a connettività alte. A basse connettività la sua capacità di trovare isomorfismi validi è comunque praticamente nulla.

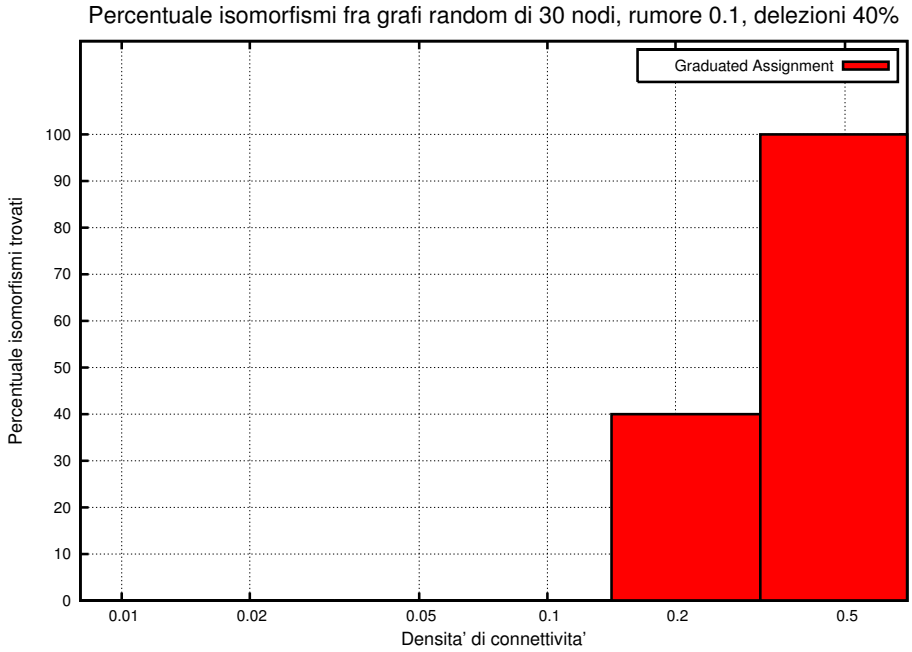

Figura 5.58: Percentuale di isomorfismi trovati da GraduatedAssignment fra grafi di 30 nodi, rumore 0.05, delezioni 40%

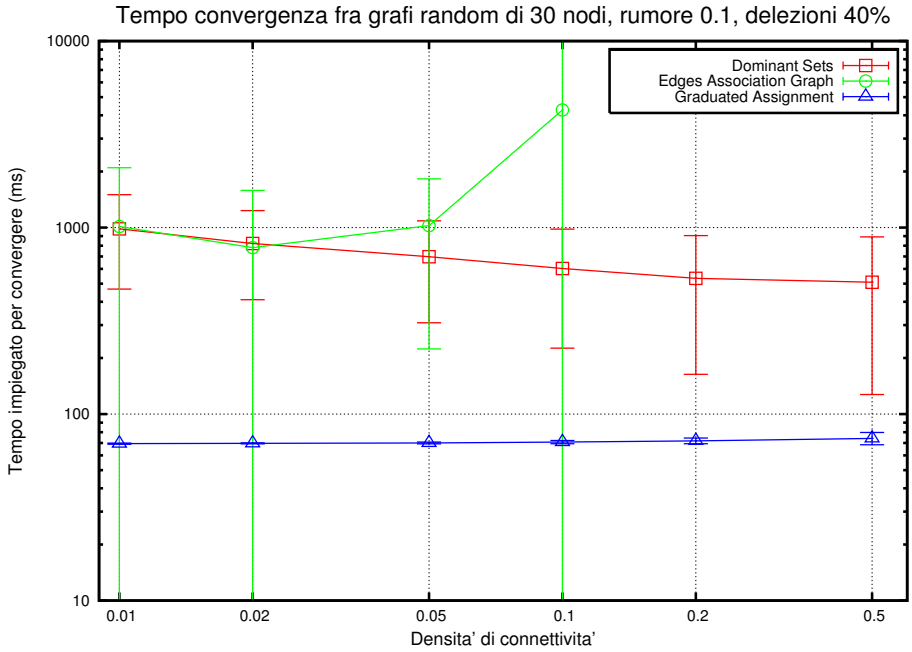

Figura 5.59: Tempi di convergenza degli algoritmi su grafi di 30 nodi, rumore 0.05, delezioni 40%

Capitolo 6

Conclusioni

6.1 Valutazione delle tecniche proposte

Dai risultati degli esperimenti effettuati riusciamo a formulare un quadro generale delle prestazioni dei diversi algoritmi testati.

6.1.1 Tecnica basata sui Dominant Set

Per quanto riguarda l'approccio basato sui Dominant Set notiamo che si comporta sempre bene solo a bassissimi livelli di connettività, ovvero intorno ad 1%. Negli altri casi l'algoritmo continua a comportarsi molto bene in presenza di perturbazioni gaussiane sugli attributi ed in assenza di delezioni. Infatti notiamo che cercando match fra grafi privi di rumore strutturale l'algoritmo fornisce risultati uguali (nel caso privo di perturbazioni sugli attributi) o addirittura migliori degli altri due, sia nel caso di rumore gaussiano basso (con σ pari a 0.05) che medio (con σ pari a 0.1).

É interessante notare che tale comportamento è consistente a tutti i livelli di connettività, ovvero la qualità del match non decade mano a mano che questo viene testato su grafi più densi e il peso del match trovato è sempre molto vicino al 100%. Questa osservazione, assieme ai tempi di convergenza sempre piuttosto rapidi, permette di affermare che la tecnica basata sui Dominant Set è certamente da preferire alle altre, compreso Graduated Assignment, per grafi di qualunque densità soggetti a sole perturbazioni sul valore degli attributi.

Purtroppo, quando si introduce anche del rumore strutturale, ovvero delle delezioni di vertici, la qualità del match offerto comincia a decadere rapidamente con l'aumentare del grado di connessione del grafo, e questo in modo relativamente indipendente dal livello di perturbazione sugli attributi introdotta. Ad ogni modo per bassi livelli di connettività (generalmente inferiori

al 10%) questo algoritmo si mantiene più efficace di Graduated Assignment, fatto particolarmente importante se consideriamo che quest'ultimo, a basse connettività e soprattutto in presenza di rumore strutturale e perturbazioni, raramente è in grado di trovare isomorfismi validi. Sospettiamo che il motivo del forte peggioramento delle prestazioni di questa tecnica in presenza di delezioni possa essere imputato al crearsi nel grafo di associazione di insiemi dominanti diversi dalle clique di peso massimo, oltre che dall'eventuale individuazione di minimi locali da parte delle dinamiche di replicazione.

Per quanto riguarda i tempi di convergenza, pur essendo questi maggiori di quelli di Graduated Assignment, rimangono sempre nell'ordine di grandezza del secondo, rendendo l'uso questa tecnica appetibile nelle condizioni in cui è in grado di trovare buoni match.

6.1.2 Tecnica basata sul grafo di associazione fra gli archi

L'approccio basato sul grafo di associazione fra archi fornisce risultati straordinariamente buoni per grafi privi di perturbazioni gaussiane sugli attributi. In particolare, a tutti i livelli di connettività del grafo testati e anche in presenza di forte rumore strutturale (fino al 40% dei vertici eliminati nel secondo grafo), permette di trovare match in media estremamente vicini all'ottimale ed in genere superiori al 98%.

Il suo comportamento comincia ad essere leggermente meno soddisfacente quando si introducono perturbazioni sugli attributi. Applicando una perturbazione gaussiana con σ pari a 0.05 i risultati continuano a mantenersi buoni: in particolare rimangono prossimi al 100% per bassi livelli di connettività e dacadono leggermente mano a mano che si presentano grafi più densi, senza però cadere mai sotto al 90%. In generale l'andamento sembra solo leggermente influenzato dal numero di delezioni effettuate sul secondo grafo. Applicando una perturbazione maggiore il suo comportamento si avvicina molto a quello della tecnica basata sui Dominant Set, in particolare con l'aumento delle delezioni, rendendo quindi preferibile quest'ultima in quanto convergente in modo più veloce. É da notare che con perturbazioni ed in assenza di delezioni, pur offrendo questa tecnica dei risultati di ottimo livello, la tecnica basata sui Dominant Set risulta preferibile in quanto è quasi sempre in grado di trovare match ottimali o comunque migliori.

In qualche modo quindi, anche se generalmente migliore, questa tecnica può essere quasi considerata complementare a quella basata sui Dominant Set, in quanto funziona bene in presenza di delezioni, mentre è soggetta a

peggioramenti dovuti alle perturbazioni degli attributi, viceversa a quanto accade per quest'ultima.

Per quanto riguarda il confronto con Graduated Assignment questa tecnica è risultata migliore in tutte le condizioni, anche se ad alti livelli di perturbazione strutturale e rumore sugli attributi sembra avvicinarvisi molto e con un andamento che lascia pensare un possibile superamento in qualità di quest'ultima ad alti livelli di connettività (per ora non testati a causa della complessità computazionale). É comunque necessario ricordare che nelle condizioni in cui questa tecnica fornisce match di qualità meno buona, Graduated Assignment, soprattutto a bassi livelli di connettività non è in grado di trovare isomorfismi validi o li trova molto raramente.

Come anticipato nella trattazione teorica, i risultati sperimentali hanno evidenziato per questa tecnica dei tempi di convergenza estremamente alti, tanto da rendere necessario per il confronto l'adozione di una scala logaritmica sull'asse dei tempi. Questo è dovuto alla dimensione delle matrici coinvolte nel problema quadratico: poichè tali matrici sembrano essere sparse è immaginabile che sia possibile trovare una formulazione dell'algoritmo in grado di abbassare tale complessità, evitando calcoli inutili. Tale possibilità sarà eventualmente oggetto di studio futuro.

6.1.3 Tecnica basata sul Graduated Assignment

Il comportamento di Graduated Assignment è piuttosto noto e ben descritto in [4]. La nostra implementazione ha prestazioni che corrispondono a quelle attese: ovvero notiamo in generale un funzionamento piuttosto insoddisfacente con grafi sparsi, ma con un miglioramento via via progressivo mano a mano che aumenta il grado di connettività, fino a raggiungere livelli di peso percentuale superiori a 80/90% per connettività vicine al 50%. I risultati sono tanto migliori quanto meno perturbati sono i grafi, anche se ad alti livelli di connettività le perturbazioni sono meno influenti sulla qualità dei match trovati.

In base a queste considerazioni le tecniche da noi proposte risultano in qualche modo andare a sopperire i difetti di Graduated Assignment in un dominio per esso particolarmente ostico. Infatti sembra che sia proprio nelle condizioni in cui questo algoritmo funziona in modo meno efficace (bassi livelli di connettività) che i nostri approcci danno i risultati migliori. Questa osservazione, inoltre, è ulteriormente rafforzata dal fatto che in tali condizioni Graduated Assignment, oltre a trovare match di peso non ottimale, raramente individua isomorfismi validi.

Per esempio nella figure 6.1 e 6.2 vediamo due match trovati su grafi di 8 nodi, senza delezioni, con connettività 20% e rumore gaussiano sugli attributi

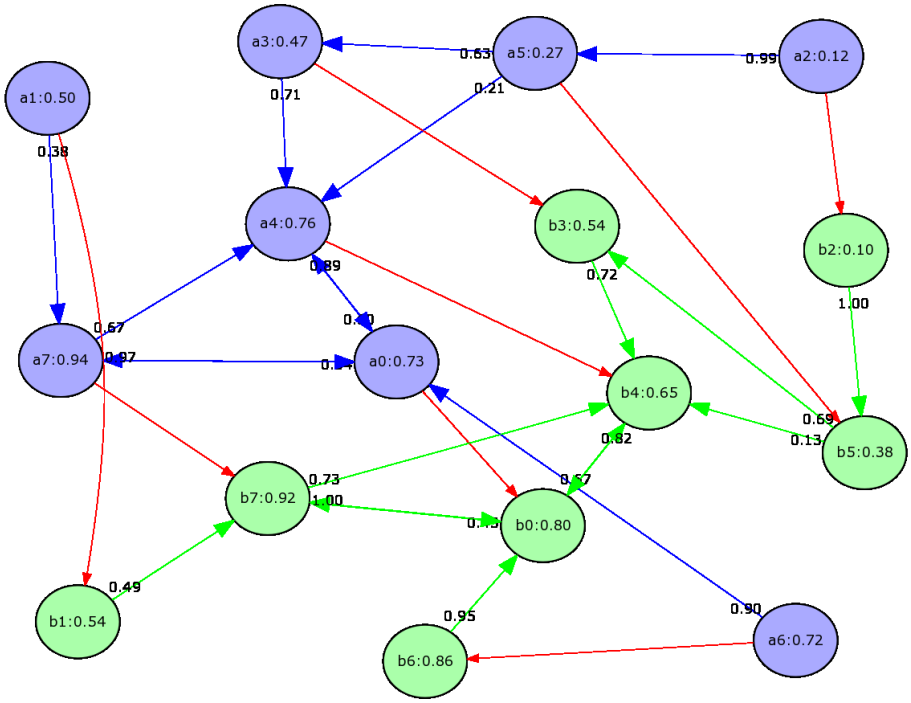

Figura 6.1: Match isomorfo trovato con EdgesAssociationGraph

con σ pari a 0.1 rispettivamente mediante la tecnica del grafo di associazione bastato sugli archi e del Graduated Assignment.

Mentre il match di figura 6.1 è effettivamente un isomorfismo, notiamo che quello ottenuto attraverso Graduated Assignment in figura 6.2 non lo è: infatti si può osservare, ad esempio, come le associazioni di $a5$ in $b5$ e di $a6$ in $b4$ non siano compatibili, in quanto $a5$ e $a6$ non risultano essere adiacenti, mentre $b5$ e $b4$ lo sono.

Dal punto di vista della velocità di convergenza invece Graduated Assignment è di gran lunga l'algoritmo più performante, infatti risulta convergere con poche iterazioni e sempre con tempi nell'ordine dei decimi di secondo.

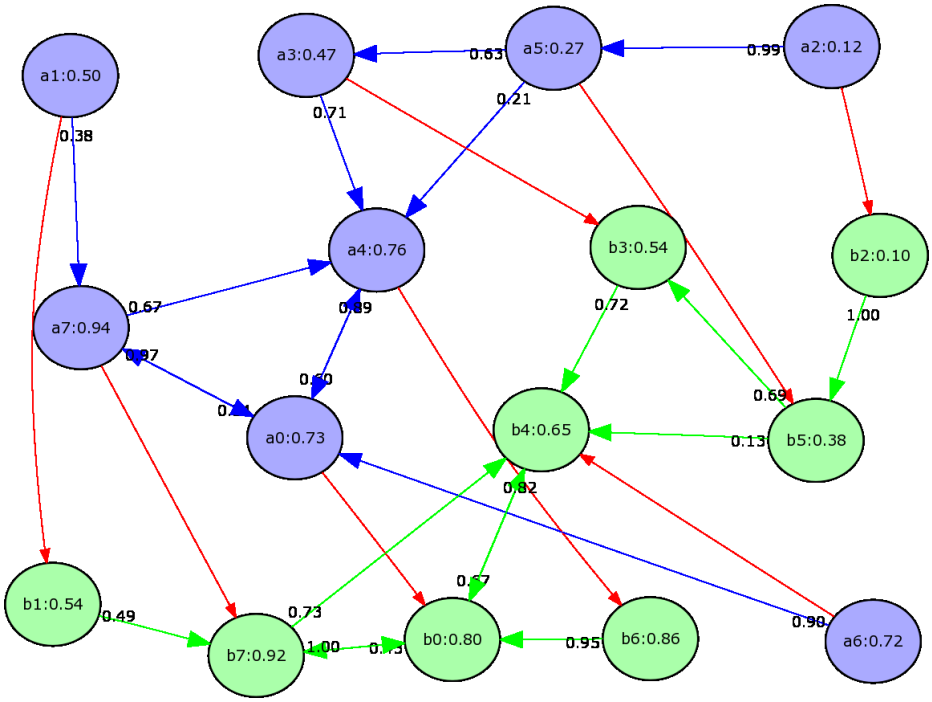

Figura 6.2: Match non isomorfo trovato con GraduatedAssignment

6.2 Sviluppi futuri

Uno dei principali problemi associati alla tecnica basata sul grafo di associazione fra archi è quello della sua elevata complessità computazionale, nonchè dei suoi ragguardevoli requisiti in termini di memoria, anche per grafi relativamente piccoli. Poichè, come precedentemente accennato, la matrice del problema quadratico che risolviamo è sparsa, potrebbe essere interessante indagare circa le possibili ottimizzazioni algoritmiche applicabili per ridurre l'ordine del numero complessivo di calcoli da effettuare.

Allo stesso modo sarà necessario studiare anche tecniche di memorizzazione parziale o di produzione dei valori della matrice solo al momento dell'uso.

Per facilitare queste operazioni un primo passo potrebbe essere quello di riformulare gli elementi della matrice, anzichè utilizzando la formulazione 3.33, con la seguente forma alternativa

$$a_{((u_1,v_1),(u_2,v_2))((w_1,z_1),(w_2,z_2))} =$$

$$= \begin{cases} \frac{1}{2\omega(u_1,u_2)} & \text{se } u_1 = v_1 = w_1 = z_1 \wedge u_2 = v_2 = w_2 = z_2 \\ \frac{1}{2\gamma((u_1,u_2),(v_1,v_2))} & \text{se } u_1 = w_1, v_1 = z_1, u_2 = w_2, v_2 = z_2 \\ 2\lambda & \text{se } ((u_1,v_1),(u_2,v_2))((w_1,z_1),(w_2,z_2)) \notin E_a \\ 0 & \text{altrimenti} \end{cases} \tag{6.1}$$

Dove λ è il massimo valore contenuto nella matrice.

In questo modo otteniamo comunque una matrice di Comtet e quindi adatta a risolvere il nostro problema, tuttavia in questo caso tutti gli elementi al di fuori della diagonale possono assumere solo due valori (cioè 0 o 2λ) di conseguenza probabilmente sarà più facile trovare formulazioni compatte per la memorizzazione della matrice e per eseguire le computazioni su di essa.

Una volta ottenuta una formulazione maggiormente trattabile, sarà opportuno ripetere gli esperimenti su un dominio più ampio: in particolare con grafi dotati di un maggior numero di nodi e valutando le prestazioni della tecnica anche con connettività superiori al 10%.

Infine sarebbe certamente interessante, oltre che eseguire test su grafi generati casualmente, individuare qualche applicazione che utilizzi dati provenienti da strutture reali che possano essere rappresentate mediante grafi con attributi. Ad esempio un'applicazione ideale, anche considerando la sparsità dei grafi associati, potrebbe essere quella di individuare sottostrutture all'interno di grafi stradali, i quali rappresentano ciascun tratto percorribile con archi dotati di attributi che indicano la dimensione della strada, la velocità media e il verso di percorrenza e ciascun incrocio mediante vertici ai quali sono associate eventuali informazioni relative a restrizioni di percorribilità, come ad esempio divieti o obblighi di svolta.

Bibliografia

[1] T. S. Motzkin and E. G. Straus, "Maxima for graphs and a new proof of a theorem of Turán," *Canad. J. Math.*, vol. 17, pp. 533–540, 1965.

[2] L. E. Gibbons, D. W. Hearn, P. M. Paradalos, and M. V. Ramana, "Continuous characterizations of the maximum clique problem," *Math. Oper. Res.*, vol. 22, pp. 754–768, 1997.

[3] M. Pelillo and M. Pavan, "A new framework for pairwise clustering and segmentation," 2003.

[4] S. Gold and A. Rangarajan, "A graduated assignment algorithm fro graph matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 377–388, 1996.

[5] D. Rouvray and A. Balaban, *Applications of graph theory*, ch. Chemical application of graph theory, pp. 177–221. Academic Press, 1979.

[6] K. Börner, E. Pipping, E. Tammer, and C. Coulon, *Advances in Case-based Reasoning*, ch. Structural similarity and adaption, pp. 58–75. Springer, 1996.

[7] J. Poole, "Similarity in legal case based reasoning as degree of matching conceptual graphs," in *First European Workshop on Case-Based Reasoning*, pp. 54–58, 1993.

[8] D. Cook and L. Holder, "Substructure discovery using minimum description length and background knowledge," *Journal of Artificial Intelligence Research*, pp. 231–255, 1994.

[9] D. Fisher, *Readings in Machine Learning*, ch. Knowledge acquisition via incremental conceptual clustering, pp. 267–286. Morgan Kaufmann, 1990.

[10] B. Messmer and H. Bunke, *Graphics Recognition, Lecture Notes in Computer Science*, ch. Automatic learning and recognition of graphical symboles in engineering drawings, pp. 123–134. Springer Verlag, 1996.

[11] H. Ehrig, "Introduction to graph grammars whith applications to semantic networks," *Computers and Mathematics with Applications*, vol. 23, pp. 557–572, 1992.

[12] P. Maher, "A similarity measure for conceptual graphs," *Int. Journal of Intelligent Systems*, vol. 8, pp. 819–837, 1993.

[13] P. Shoubridge, M. Krarne, and D. Ray, "Detection of abnormal change in dynamic networks," in *IDC*, pp. 557–562, 1999.

[14] X. Jiang, A. Münger, and H. Bunke, "Synthesis of representative graphical symbols by generalized median graph computation," in *IAPR Workshop on Graphics Recognition*, 1999.

[15] S. Lee, J. Kim, and F. Groen, "Translation-rotation-and scale invariant recognition of hand-drawn symbols in schematic diagrams," *Journal of Pattern Recognition and Artificial Intelligence*, vol. 4, pp. 1–15, 1990.

[16] S. Lu, Y. Ren, and C. Suen, "Hierarchical attributed graph representation and recognition of handwritten chinese characters," *Pattern Recognition*, vol. 24, pp. 617–632, 1991.

[17] J. Rocha and T. Pavlidis, "A shape analysis model with applications to a character recognition system," *IEEE Trans. PAMI*, pp. 393–404, 1994.

[18] V. e. a. Cantoni, "2-d object recognition by multiscale tree matching," *Pattern Recognition*, vol. 31, pp. 1443–1455, 1998.

[19] T. Lourens, "A biologically plausible model for corner-based object recognition from color images," tech. rep., University of Groningen, The Netherlands, 1998.

[20] M. Pelillo, K. Siddiqi, and S. Zucker, "Matching hierarchical structures using associated graphs," *IEEE Trans. PAMI*, vol. 21, pp. 1105–1120, 1999.

[21] E. Wong, *Syntactic and Structural Pattern Recognition - Theory and Applications*, ch. Three-dimensional object recognition by attributed graphs, pp. 381–414. World Scientific, 1990.

[22] B. Kimia, A. Tannenbaum, and S. Zucker, "Shape, shocks and deformations i: The components of two-dimensional shape and the reaction-diffusion space," *International Journal of Computer Vision*, vol. 15, pp. 189–224, 1995.

[23] K. Siddiqui, A. Shokoufandeh, S. Dickinson, and S. Zucker, "Shock graphs and shape matching," *International Journal of Computer Vision*, vol. 30, pp. 1–22, 1999.

[24] K. Siddiqui, A. Shokoufandeh, S. Dickinson, and S. Zucker, "Shock graphs and shape matching," in *IEEE International Conference on Computer Vision, Bombay*, pp. 222–229, 1998.

[25] A. Shokoufandeh, I. Marsic, and S. Dickinson, "Saliency regions as a basis for object recognition," in *Third International Workshop on Visual Form, Capri, Italy*, 1997.

[26] A. Shokoufandeh, I. Marsic, and S. Dickinson, "View-based object matching," in *Third International Workshop on Visual Form, Capri, Italy*, pp. 588–595, 1998.

[27] A. Shokoufandeh, I. Marsic, and S. Dickinson, "View-based object recognition using saliency maps," *Image and Vision Computing*, vol. 27, pp. 445–460, 1999.

[28] J. Ullmann, "An algorithm for subgraph isomorfism," *Journal of the Association for Computing Machinery*, vol. 23, pp. 31–42, 1976.

[29] J. McGregor, "Backtrack search algorithms and the maximal common subgraph problem," *Software-Practice and Experience*, vol. 12, pp. 23–34, 1982.

[30] G. Levi, "A note on the derivation of maximal common subgraphs of two directed or undirected graphs," *Calcolo*, vol. 9, pp. 341–354, 1972.

[31] A. Pearce, T. Caelli, and F. Bischof, "Rulegraphs for graph matching in pattern recognition," *Pattern Recognition*, vol. 27, pp. 1231–1247, 1994.

[32] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*. Prentice-Hall, 1982.

[33] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *Ann. Math. Statistics*, vol. 35, pp. 876–879, 1964.

[34] E. Hancock and J. Kittler, "Edge-labeling using dictionary-based relaxation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 165–181, 1990.

[35] J. Kittler, W. Christmas, and M. Petrou, "Probabilistic relaxation for matching problems in computer vision," in *IEEE Proceedings of the International Conference on Computer Vision (ICCV93)*, 1993.

[36] W. J. Christmas, J. Kittler, and P. M., "Structural matching in computer vision using probabilistic relaxation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 749–764, 1995.

[37] R. C. Wilson and H. E. R., "Structural matching by discrete relaxation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 634–648, 1997.

[38] M. Williams, R. Wilson, and E. Hancock, "Deterministic search for relational graph matching," *Pattern Recognition*, vol. 32, pp. 1255–1271, 1999.

[39] E. Rolland, H. Pirkul, and F. Glover, "Tabu search for graph partitioning," *Ann. Oper. Res.*, vol. 63, pp. 200–232, 1996.

[40] F. Glover, "Ejection chains, reference structures and alternating path methods for traveling salesman problems," *Discrete Appl. Math.*, vol. 65, pp. 223–253, 1996.

[41] F. Glover, "Genetic algorithms and tabu search - hybrids for optimisation," *Discrete Appl. Math.*, vol. 49, pp. 111–134, 1995.

[42] F. Glover, "Tabu search for nonlinear and parametric optimisation (with links to genetic algorithms)," *Discrete Appl. Math.*, vol. 49, pp. 231–255, 1995.

[43] R. C. Wilson and H. E. R., "Graph matching with hierarchical discrete relaxation," *Pattern Recognition Letters*, vol. 20, pp. 1041–1052, 1999.

[44] A. Cross and Hancock, "Graph matching with a dual-step em algorithm," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 1236–1253, 1998.

[45] A. Finch, R. C. Wilson, and H. E. R., "Symbolic graph matching with the em algorithm," *Pattern Recognition*, vol. 31, pp. 1777–1790, 1998.

[46] B. Luo and E. Hancock, "A robust eigendecomposition framework for inexact graph matching," in *Proceedings 11th International Conference on Image Analysis and Processing*, 2001.

[47] B. Luo and E. Hancock, "Structural graph matching using the em algorithm and singular value decomposition," *IEEE Transactions on Pattern and Machine Intelligence*, vol. 23, pp. 1120–1136, 2001.

[48] H. Kim and J. Kim, "Hierarchical random graph representation of handwritten characters and its application to hangul recognition," *Pattern Recognition*, vol. 34, pp. 187–201, 2001.

[49] R. Herpers and G. Sommer, "iscrimination of facial regions based on dynamic grids of point representations," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 12, pp. 381–405, 1998.

[50] B. Duc, E. Bigun, J. Bigun, G. Maitre, and S. Fisher, "Fusion of audio and video information for multi modal person authentication," *Pattern Recognition Letters*, vol. 18, pp. 835–843, 1997.

[51] R. Mariani, "Face learning using a sequence of images," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, pp. 631–648, 2000.

[52] P. Fariselli and R. Casadio, "Prediction of disulfide connectivity in proteins," *Bioinformatics*, vol. 17, pp. 957–964, 2001.

[53] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 695–703, 1988.

[54] H. A. Almohamad and S. O. Duffaa, "A linear programming approach for the weighted graph matching problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 522–525, 1993.

[55] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[56] H. Almohamad, "A polynomial transform for matching pairs of weighted graphs," *J. Applied Math. Modeling*, vol. 15, 1991.

[57] A. P. Ambler, H. G. Barrow, C. M. Brown, R. M. Burstall, and R. J. Popplestone, "A versatile computer-controlled assembly system.," in *IJCAI*, pp. 298–307, 1973.

[58] P. Pardalos and A. Phillips, "A global optimization approach for solving the maximum clique problem," *Int. J. Comput. Math.*, vol. 33, pp. 209–216, 1990.

[59] M. Pelillo and A. Jagota, "Feasible and infeasible maxima in quadratic program for maximum clique," *J. Artif. Neural Networks*, vol. 2, pp. 411–420, 1995.

[60] I. M. Bomze, "Evolution towards the maximum clique," *J. Global Optim.*, vol. 10, pp. 143–164, 1997.

[61] L. Lovász and A. Schrijver, "Cones of matrices and set-functions and 0-1 optimization," *SIAM j. Optim*, vol. 1, pp. 166–190, 1991.

[62] I. M. Bomze, M. Pelillo, and V. Stix, "Approximating the maximum weight clique: An evolutionary game theory approach," *manuscript in preparation*, 1998.

[63] I. Bomze, "On standard quadratic optimization problems," *J. Glob. Optim*, vol. 13, pp. 369–387, 1998.

[64] M. Pelillo and M. Pavan, "A new graph-theoretic approach to clustering and segmentation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[65] M. Pavan, M. Pelillo, and E. Jabara, "On the combinatories of standard quadratic optimization." Forthcoming, 2003.

[66] J. Hofbauer and K. Sigmund, *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[67] J. Weibull, *Evolutionary Game Theory*. MIT Press, 1995.

[68] J. Hofbauer, *Imitation synamics for games*. Collegium Budapest, 1995.

[69] M. Pelillo and A. Torsello, "Payoff-monotonic game dynamics and the maximum clique problem," tech. rep., Università Ca' Foscari di Venezia, 2005.

[70] I. Bomze, M. Pelillo, and R. Giacomini, *Developments in Global Optimization*, ch. Evolutionary approach to the maximum clique problem: Empirical evidence on a larger scale, pp. 95–108. Dordrecht, 1997.

[71] M. Pelillo, "Replicator equations, maximal cliques and graph isomorphism," *Neural Computation*, vol. 11, pp. 2023–2045, 1999.

[72] M. Pelillo, "Matching free trees, maximal cliques, and monotone game dynamics," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 1535–1541, 2002.

[73] I. Bomze, "Portfolio selection via replication dynamics and projection of indefinite estimated covariances," *Dynamics of Continuous, Discrete and Impulsive Systems*, 2005.

Ringraziamenti

Un ringraziamento al mio relatore prof. Marcello Pelillo ed al mio correlatore prof. Andrea Torsello per la disponibilità e il prezioso aiuto che mi hanno dato nel redigere questa tesi.

Un grazie va a tutta la mia famiglia che mi ha sostenuto durante tutto il percorso universitario.

Un profondo ringraziamento anche ai miei colleghi, per aver reso possibile dedicare parte del tempo lavorativo all'impegno accademico.