UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI- CIVITA"
CORSO DI LAUREA MAGISTRALE IN MATEMATICA

Problemi decisionali sui grafi: l'approccio logico basato sul Teorema di Courcelle

Relatore:
Prof. Francesco Ciraulo

Laureando:
Alessandro Berardi
2006653

19 aprile 2024

Indice

Capitolo 1

Introduzione alla Teoria dei Grafi e Logica: Fondamenti per il Teorema di Courcelle

Questo capitolo serve a fornire un background teorico per i concetti chiave che andremo ad analizzare in questa tesi, a partire dalle definizioni e concetti fondamentali.

1.1 Grafi

1.1.1 Teoria dei grafi

Definizione 1.1.1. Sia V un insieme sia $E \subseteq V \times V$.
La coppia ordinata $G = (V, E)$ si dice grafo. V è l'insieme dei vertici (o nodi) di G mentre E è l'insieme dei suoi archi (o lati).
Due vertici u, v connessi da un arco $e = (u, v)$ prendono il nome di estremi dell'arco, in tale caso u e v si dicono vertici adiacenti.
Un grafo $G = (V, E)$ si dice indiretto (o non orientato) se $\forall u, v \in V$ si ha $(u, v) \in E \Leftrightarrow (v, u) \in E$, in tal caso si indica con $\{u, v\}$ l'arco di estremi u e v.
Se G non è indiretto si dice che è un grafo diretto (o orientato).

I grafi sono fondamentali nella rappresentazione di una vasta gamma di problemi. Ad esempio i grafi sono utilizzati per rappresentare reti complesse come reti di computer, reti di trasporto, reti elettriche, reti di telecomunicazioni. I nodi rappresentano dispositivi o punti di connessione, mentre gli archi rappresentano i collegamenti tra di essi. Questa rappresentazione è cruciale per la progettazione e l'ottimizzazione di queste reti nel mondo reale. Oppure vengono anche utilizzati in problemi di ottimizzazione come il problema del commesso viaggiatore, che richiede di trovare il percorso più breve che visita una serie di città. Questi problemi sono fondamentali nella ricerca operativa e nella teoria degli algoritmi.

Definizione 1.1.2. L'ordine di un grafo $G = (V, E)$ è il suo numero di vertici $|V|$. La dimensione di un grafo è il suo numero di archi $|E|$.
Si dice che un vertice $v \in V$ ha grado n se esistono esattamente n archi incidenti su

v. Se il grafo è diretto si può definire il *grado entrante* come il numero di archi che terminano su v e *grado uscente* come il numero di archi che partono da v.
Si definiscono il grado minimo e il grado massimo di G come, rispettivamente, il grado minore e maggiore dei vertici di G. Se tutti i vertici hanno lo stesso grado, il grafo si dice regolare.

Un multigrafo è una forma di generalizzazione di un grafo che permette ad archi differenti di avere la stessa coppia di estremi.

Definizione 1.1.3. Un multigrafo diretto (o orientato) $G = (V, E, \phi)$ è una terna che consiste di due insiemi, V l'insieme dei vertici ed E l'insieme degli archi, e di un'applicazione $\phi : E \to V \times V$. Se $e \in E, u, v \in V$ e $\phi(e) = (u, v)$, allora si dice che e è un arco orientato da u a v. Nel caso particolare in cui $u = v$ si ha che $\phi(e) = (u, u)$; in tal caso l'arco e si dice cappio.
Un multigrafo indiretto (o non orientato) $G = (V, E, \phi)$ è una terna che consiste di due insiemi, V l'insieme dei vertici ed E l'insieme degli archi, e di un'applicazione $\phi : E \to \mathcal{P}_2(V)$, dove $\mathcal{P}_2(V)$ è l'insieme di tutti i sottoinsiemi di V di cardinalità due. Se $e \in E, u, v \in V$ e $\phi(e) = \{u, v\}$, si dice che e è un arco da u a v.
Per dire che ci sono più lati aventi gli stessi estremi u e v si dice che c'è un arco multiplo tra u e v.

A differenza di un grafo semplice, in cui c'è al massimo un arco tra qualsiasi coppia di nodi, un multigrafo può avere duplicati di archi che collegano gli stessi nodi. Questo concetto permette di rappresentare in modo più accurato le situazioni del mondo reale in cui possono esistere molteplici relazioni o connessioni tra gli stessi oggetti.
I multigrafi sono utilizzati in diverse applicazioni pratiche, come le reti di comunicazione, le reti sociali online, i sistemi di trasporto pubblico e altri scenari in cui possono sorgere relazioni multiple tra gli stessi elementi. La possibilità di rappresentare queste relazioni complesse rende i multigrafi uno strumento utile nella modellazione di sistemi reali.

Definizione 1.1.4. Un cammino dal vertice u al vertice v in un grafo G è una successione finita $e_1 = (z_1, z_2), e_2 = (z_2, z_3), ..., e_n = (z_n, z_{n+1})$ di archi distinti di G tali che $u = z_1$ e $v = z_{n+1}$. In tal caso si dice che n è la lunghezza del cammino. In ogni cammino archi consecutivi sono incidenti.
Un circuito (o ciclo) è un qualunque cammino di lunghezza maggiore di zero da un vertice v allo stesso vertice v.
Un grafo $G = (V, E)$ si dice connesso se $\forall u, v \in V$ esiste un cammino da u a v. Un grafo che non è connesso si dice sconnesso.
Se $u \in V$, l'insieme C_u i cui elementi sono tutti i vertici $v \in V$ per i quali esiste un cammino da u a v è un sottoinsieme di V, chiamato la componente connessa di u.

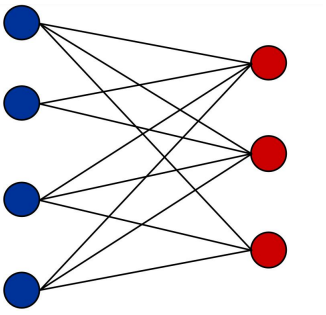
I cammini sono concetti fondamentali nella teoria dei grafi e sono ampiamente utilizzati per rappresentare percorsi o sequenze di eventi in una varietà di applicazioni, come reti di computer, trasporti, ottimizzazione delle rotte e molti altri contesti dove le connessioni tra i vertici sono importanti.

Definizione 1.1.5. Un grafo si dice completo se ogni coppia di vertici è collegata da un unico arco, cioè se tutti i suoi vertici sono a due a due adiacenti.

Per ogni numero intero $n \geq 1$ c'è un unico grafo completo con n vertici a meno di isomorfismi, il quale si indica con K_n (la lettera K sta per komplett, che significa completo in tedesco).
Un grafo completo K_n contiene $\frac{n(n-1)}{2}$ archi. Questo deriva dal fatto che ogni nodo è connesso agli altri $n-1$ nodi del grafo.
Il grafo K_n è un grafo connesso e regolare di grado $n-1$.
I grafi completi sono utilizzati come rappresentazione di situazioni in cui esiste una connessione diretta tra ogni coppia di oggetti o entità nel contesto del problema considerato.

Definizione 1.1.6. un grafo $G = (V, E)$ si dice bipartito se l'insieme dei vertici V può essere diviso in due sottoinsiemi disgiunti V_1 e V_2 tali che ogni arco in E collega un vertice in V_1 ad un vertice in V_2.

$\{V_1, V_2\}$ è una partizione di V.
In un grafo bipartito non ci sono archi tra vertici dello stesso insieme.



Esempio 1.1.7.

Esempio 1.1.8. Siano m, n interi positivi. Il grafo bipartito completo $K_{n,m} = (V, E)$ è il grafo tale che

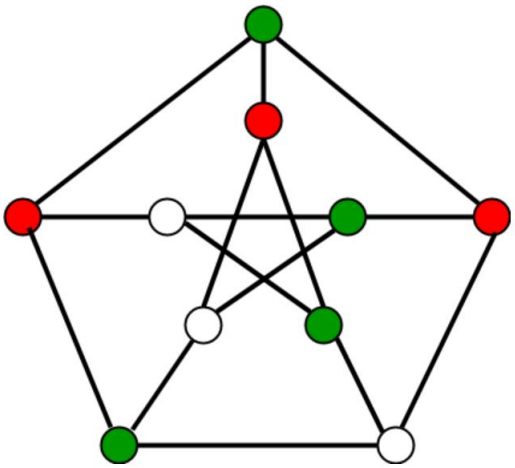
$$V = \{u_1, u_2, ..., u_n, v_1, v_2, ..., v_m\},$$
$$E = \{e_{i,j} | i = 1, 2, ..., n,\ j = 1, 2, ..., m\}$$

e $e_{i,j} = \{u_i, v_j\}$. Quindi $K_{n,m}$ è il grafo bipartito con $m + n$ vertici in cui, se $V_1 = \{u_1, u_2, ..., u_n\}$ e $V_2 = \{v_1, v_2, ..., v_n\}$, ogni vertice di V_1 è adiacente ad ogni vertice di V_2.

Questo tipo di grafo rappresenta una situazione in cui ci sono relazioni complete tra due gruppi distinti di oggetti o entità, ed è un concetto fondamentale nella teoria dei grafi bipartiti.

Definizione 1.1.9. La *colorazione di un grafo* è un aspetto della teoria dei grafi che coinvolge l'assegnazione di colori ai nodi di un grafo in modo tale che nodi adiacenti abbiano colori diversi, questa assegnazione di colori è chiamata *colorazione*.
Una colorazione che usa al massimo k colori è chiamata $k-$ *colorazione*, il numero minimo di colori richiesti per colorare un grafo $G = (V, E)$ è chiamato il suo *numero cromatico*.

Esempio 1.1.10. La seguente figura mostra l'esempio di una 3-colorazione di un grafo, con numero cromatico 3, in quanto il grafo non è 2-colorabile.



Esempio 1.1.11. I grafi bipartiti hanno numero cromatico 2.

La colorazione dei grafi è un argomento ampiamente studiato e ha applicazioni pratiche in diversi contesti; essa rappresenta un problema classico nell'ambito della teoria dei grafi. Si veda ad esempio [13].

1.1.2 Alberi

Definizione 1.1.12. Un albero è un grafo non orientato connesso privo di circuiti.

Ogni coppia di nodi in un albero è collegata da un unico cammino. Non ci sono nodi isolati o nodi non raggiungibili in un albero.
Un albero con n nodi ha esattamente $n - 1$ archi. Questo numero minimo di archi collega tutti i nodi senza formare cicli.

Definizione 1.1.13. Una foresta è un grafo privo di circuiti.

Una foresta è costituita da più componenti connesse, ognuna delle quali è un albero. In particolare ogni foresta risulta essere unione disgiunta di più alberi.

Proposizione 1.1.14. *Un grafo G è un albero se e solo se per ogni coppia di nodi distinti di G c'è un unico cammino che li unisce.*

Dimostrazione. Supponendo che G sia un albero. Siano u e v vertici distinti. Bisogna dimostrare due cose per mostrare che c'è un cammino unico tra u e v: che esiste un cammino e che non ce ne sia più di uno.
La prima di queste è automatica, poiché G è un albero, è connesso, quindi c'è un cammino tra ogni coppia di vertici. Per mostrare che è unico, si supponga che ci siano due cammini distinti tra u e v. I due cammini potrebbero iniziare allo stesso modo, ma poiché sono diversi, c'è qualche primo vertice u' dopo il quale i due cammini divergono. Tuttavia, poiché i due cammini terminano entrambi in v, c'è un primo vertice dopo u' che hanno in comune, chiamato v'. Ora considerando i due cammini

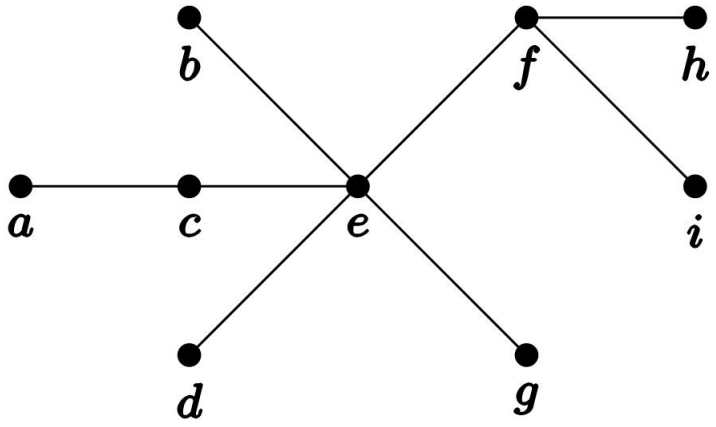
da u' a v'. Presi insieme, questi formano un ciclo, che contraddice la supposizione che G sia un albero.
Ora si supponga che tra ogni coppia di vertici distinti di G c'è un unico cammino. Per dimostrare che G è un albero, si deve mostrare che è connesso e non contiene cicli.
G è connesso, perché c'è un cammino tra ogni coppia di vertici. Per mostrare che G non ha cicli, si supponga per assurdo che ce ne sia uno. Siano u e v due vertici distinti in un ciclo di G. Poiché si può andare da u a v in senso orario o antiorario intorno al ciclo, ci sono due percorsi da u a v, contraddicendo l'ipotesi. □

Leggendo attentamente la dimostrazione sopra, si può notare che entrambe le direzioni sono costituite da due parti: l'esistenza di cammini e l'unicità dei cammini (correlata al fatto che non ci siano cicli). In questo caso, queste due parti sono effettivamente separate. Infatti, se si considerassero solo grafi senza cicli, cioè foreste, si potrebbero comunque eseguire le parti della dimostrazione che esplorano l'unicità dei cammini tra i vertici.

Corollario 1.1.15. *Un grafo G è una foresta se e solo se per ogni coppia di nodi distinti di G c'è al più un cammino che li unisce.*

Definizione 1.1.16. In un albero, un vertice di grado uno viene chiamato foglia.

Ogni albero con almeno due vertici ha almeno due foglie.
Risulta essere particolarmente utile aggiungere una struttura aggiuntiva agli alberi che aiuterà a risolvere diversi tipi di problemi.
Questo tipo speciale di albero viene chiamato albero radicato (o albero con radice), in cui viene designato un nodo come radice, e ogni altro nodo ha un padre unico, formando una struttura gerarchica con una direzione definita. Questo funziona perché c'è un cammino unico tra ogni coppia di vertici in un albero. Quindi, da qualsiasi vertice, si può tornare alla radice in esattamente un modo. Questo consente anche di descrivere come sono correlati vertici distinti in un albero radicato.
Se due vertici sono adiacenti, si dice che uno di essi è il genitore mentre l'altro è chiamato figlio. Dei due, il genitore è il vertice più vicino alla radice. Quindi, la radice di un albero è un genitore, ma non è il figlio di nessun vertice.
In generale, si dice che un vertice v è un discendente di un vertice u a condizione che u sia un vertice nel cammino da v alla radice. Si dice che u è un antenato di v.
Per la maggior parte degli alberi, ci sono coppie di vertici in cui nessuno dei due è un discendente dell'altro: vengono chiamati cugini o fratelli. In particolare, due vertici u e v sono chiamati fratelli a condizione che abbiano lo stesso genitore.



Esempio 1.1.17.
Se nell'esempio di questo albero radicato il nodo f viene designato come radice, si ha che i nodi e, h e i sono tutti figli di f e fratelli tra loro. Il nodo a è il figlio di c e discendente di e. Mentre i nodi b, c, d e g sono tutti fratelli avendo il genitore comune e.
Se, al contrario, viene designato come radice il nodo a, c risulta essere l'unico figlio di a, il quale a sua volta ha un unico figlio e. In questo caso f è un discendente di a invece di essere antenato. f e g ora sono fratelli.

Questo linguaggio aiuta a descrivere come muoversi attraverso un albero: visitare i vertici in un certo ordine è un passo fondamentale in molti algoritmi. Ad esempio gli alberi radicati sono comunemente utilizzati per rappresentare la struttura delle directory e dei file in un sistema operativo. La radice rappresenta la directory principale, i nodi interni rappresentano le sottodirectory e le foglie rappresentano i file.

1.1.3 Decomposizioni ad albero

Definizione 1.1.18. Una decomposizione ad albero di un grafo $G = (V, E)$ è una coppia (T, X), dove $T = (I, F)$ è un albero radicato e $X : I \to 2^V$ è una funzione tale che

1. Per ogni nodo $v \in V$ esiste un $i \in I$ con $v \in X(i)$.

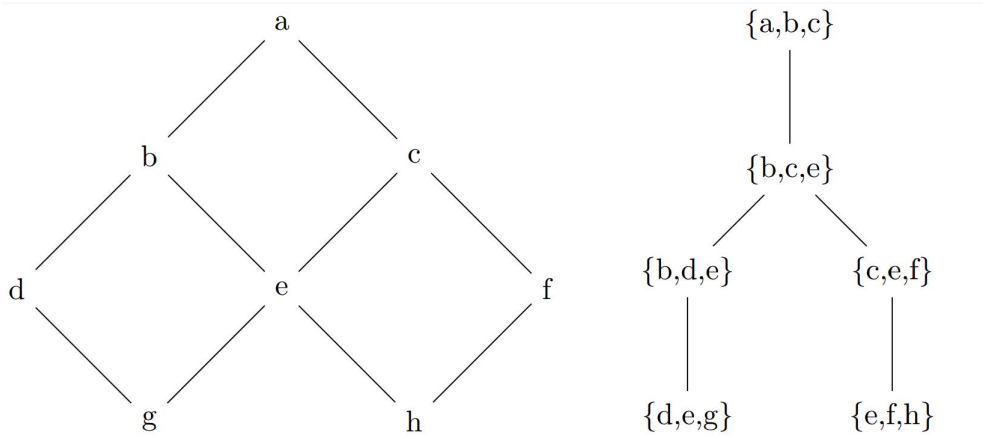
2. Per ogni arco $\{u, v\} \in E$ esiste un $i \in I$ con $\{u, v\} \subset X(i)$.

3. Per ogni nodo $v \in V$, il sottografo indotto da $\{i \in I | v \in X(i)\}$ è un albero.

Gli insiemi $X(i)$ sono chiamati *borse*.

La decomposizione ad albero è spesso utilizzata in contesti in cui è vantaggioso risolvere un problema su un grafo più semplice, come un albero, e poi combinare le soluzioni parziali per ottenere la soluzione al problema originale.

Esempio 1.1.19. Viene ora fornito un esempio di una decomposizione ad albero per un grafo. Le tre proprietà sono facilmente verificate.

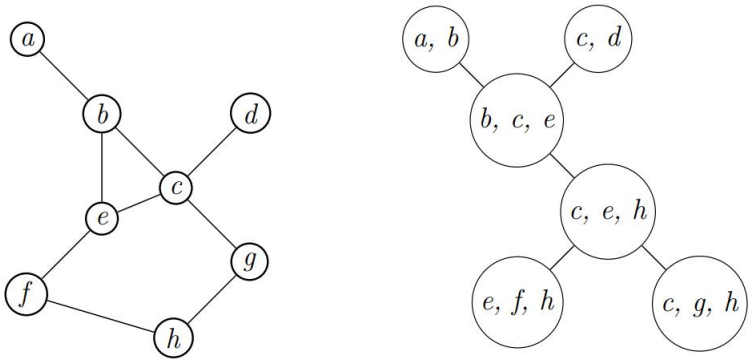


Si può notare che la struttura dell'albero ricorda visivamente quella del grafo corrispondente. Si può anche notare che la decomposizione ad albero non è unica.

Ora è utile introdurre il concetto di *larghezza dell'albero* di un grafo, un numero intero che descrive quanto un grafo è lontano dall'essere un albero.

Definizione 1.1.20. La *larghezza* di una decomposizione ad albero (T, X) è data dalla dimensione della borsa con cardinalità più grande meno uno, cioè $\max_{i \in I} |X(i)| - 1$.
La *larghezza dell'albero* di un grafo $G = (V, E)$ è la minima larghezza tra tutte le sue possibili decomposizioni.

Esempio 1.1.21. Nella seguente figura vengono mostrati un grafo con larghezza dell'albero pari a 2 e una sua corrispettiva decomposizione ad albero di larghezza 2:



La più piccola *larghezza dell'albero* possibile è 1, questi grafi sono esattamente gli alberi e le foreste.
La *larghezza* è un parametro fondamentale perché influisce sull'efficienza degli algoritmi che utilizzano la decomposizione ad albero. Larghezze più piccole generalmente portano ad algoritmi più efficienti.

Esempio 1.1.22. Ogni grafo completo K_n ha larghezza dell'albero $n - 1$.

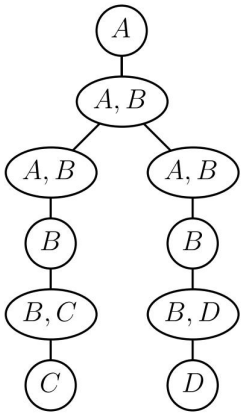
Definizione 1.1.23. Si dice che una classe di grafi $\mathcal{C}$ ha una *larghezza dell'albero limitata* se esiste $k \in \mathbb{N}$ tale che per ogni $G \in \mathcal{C}$, la larghezza dell'albero di G è minore o uguale a k.

Definizione 1.1.24. Una decomposizione ad albero (T, X) si dice ottimale se

1. ogni $i \in I$ ha al massimo due figli,
2. se $i \in I$ ha esattamente un figlio $j \in I$, allora $X(i)$ e $X(j)$ differiscono di esattamente un nodo,
3. se $i \in I$ ha due figli $l, r \in I$, allora $X(i) = X(l) = X(r)$.

Gli algoritmi progettati con decomposizioni ad albero ottimali spesso traggono vantaggio dalla regolarità e dalla semplicità della decomposizione, portando a un miglioramento dei tempi di esecuzione e delle garanzie sulle prestazioni.

Esempio 1.1.25. Una decomposizione ad albero ottimale



1.2 Logica

1.2.1 Logica del primo ordine

Una logica del primo ordine è un sistema logico che estende la logica proposizionale introducendo concetti di variabili, quantificatori e predicati. Questa logica è utilizzata per formulare e ragionare su affermazioni che coinvolgono oggetti specifici, relazioni tra oggetti e proprietà degli oggetti stessi.
Ci sono due tipi di espressioni in una logica del primo ordine:

1. *termini*, che rappresentano gli oggetti,
2. *formule*, che rappresentano affermazioni che possono essere vere o false.

I termini e formule di una logica del primo ordine sono sequenze di *simboli*, in cui tutti i simboli del linguaggio formano l'alfabeto.
Questi ultimi si possono dividere in simboli logici e simboli non logici.
I simboli logici sono rappresentati dai seguenti:

1. *quantificatori*: il quantificatore universale $\forall$ e il quantificatore esistenziale $\exists$,

2. *connettivi logici*: congiunzione $\wedge$, disgiunzione $\vee$, implicazione $\Rightarrow$, doppia implicazione $\Leftrightarrow$ e negazione $\neg$,

3. *parentesi e altri simboli di punteggiatura*,

4. *variabili*: generalmente indicate con le ultime lettere dell'alfabero $x, y, z...$,

5. *il simbolo di uguaglianza* $=$.

I simboli non logici sono rappresentati dai seguenti:

1. *costanti e funzioni*: si possono utilizzare costanti per rappresentare oggetti specifici e funzioni per descrivere relazioni tra oggetti. Ad esempio, $f(x)$ potrebbe rappresentare un'azione o una relazione che coinvolge l'oggetto x,

2. *predicati*: relazioni che prendono come argomento uno o più oggetti e restituiscono un valore booleano.

I predicati sono fondamentali nella logica del primo ordine poiché consentono di esprimere proprietà e relazioni su oggetti specifici, contribuendo così a formulare affermazioni più complesse e generiche all'interno di un sistema logico.

Definizione 1.2.1. Il numero di argomenti di un predicato o funzione viene chiamato *arietà*.

Esempio 1.2.2. :

1. $P(x)$ è una relazione 1-aria (o unaria),

2. $Q(x, y)$ è una relazione 2-aria (o binaria),

3. $R(x_1, ..., x_n)$ è una relazione n-aria.

La logica del primo ordine fornisce un linguaggio espressivo per formulare affermazioni e ragionare su relazioni e proprietà all'interno di un dominio specifico.
Si possono differenziare due tipi di formule:

1. *atomiche*, ad esempio $Q(x, y)$,

2. *composte*, ad esempio $P(x) \vee Q(y, z)$.

Esempio 1.2.3 (Teoria dei gruppi)**.** Sia $(G, \cdot, 1)$ un gruppo. Nella logica del primo ordine associata a G

1. $x, y, 1 \in G$, $x \cdot y, x^{-1} \in G$ sono *termini*,

2. se $x, y, z \in G$, $x \cdot y = 1$ e $x \cdot z = y^{-1}$ sono *formule atomiche*,

3. $\forall x, y \in G : x \cdot y = y \cdot x$ è un esempio di *formula composta*, essa indica che il gruppo preso in considerazione è abeliano.

Esempio 1.2.4 (Aritmetica di Peano)**.** Sia $(\mathbb{N}, s, 0, +, \cdot)$ l'aritmetica di Peano.

1. $x, y, 0, s(0), s(s(x \cdot y)), s(x + 0) \in \mathbb{N}$ sono *termini*,

2. $s(x) = s(y), s(s(0)) = x$ sono *formule atomiche*,

3. $\forall n, m \in \mathbb{N}, \exists k \in \mathbb{N} : n + k = m$ è un esempio di *formula composta*.

Esempio 1.2.5. Sia $(G, \cdot, 1)$ un gruppo. L'elemento neutro 1 rappresenta l'unica costante mentre l'operazione $\cdot : G \times G \to G$ è una funzione binaria.
Possono essere utilizzate formule di una logica del primo ordine per formalizzare gli assiomi della teoria dei gruppi:

$$\forall x \forall y \forall z (x \cdot (y \cdot z) = (x \cdot y) \cdot z) \tag{1.1}$$

$$\forall x (x \cdot 1 = x \wedge 1 \cdot x = x) \tag{1.2}$$

$$\forall x \exists y (x \cdot y = 1 \wedge y \cdot x = 1) \tag{1.3}$$

La formula (1.1) indica la proprietà associativa, la formula (1.2) indica l'elemento neutro mentre la formula (1.3) indica l'esistenza dell'inverso.

Esempio 1.2.6. Sia $(\mathbb{N}, s, 0, +, \cdot)$ l'aritmetica di Peano. Il numero 0 è l'unica costante, la funzione successore s è una funzione unaria mentre le funzioni di somma e prodotto, $+$ e $\cdot$, sono funzioni binarie.
Possono essere usate formule del primo ordine per esprimere gli assiomi dell'aritmetica

$$\forall x \neg (s(x) = 0) \tag{1.4}$$

$$\forall x \forall y (s(x) = s(y) \Rightarrow x = y) \tag{1.5}$$

$$\forall x (x + 0 = x) \tag{1.6}$$

$$\forall x \forall y (x + s(y) = s(x + y)) \tag{1.7}$$

$$\forall x (x \cdot 0 = 0) \tag{1.8}$$

$$\forall x \forall y (x \cdot s(y) = (x \cdot y) + x) \tag{1.9}$$

Si consideri ora il seguente predicato $P(x)$

$$\exists y (x = y \cdot s(s(0))) \tag{1.10}$$

Questo predicato indica se il numero x preso in argomento è pari espresso in logica del primo ordine.

1.2.2 Logica del secondo ordine

Una logica del secondo ordine è un sistema logico che estende la logica del primo ordine introducendo variabili quantificate che possono riferirsi a insiemi di elementi, funzioni e relazioni generiche piuttosto che solo ad elementi individuali. Questo permette di esprimere proprietà non esprimibili nella logica del primo ordine.

Esempio 1.2.7. Consideriamo le seguenti formule

$$\phi := \forall x \exists y [P(x,y)] \wedge \forall x \forall y \forall z [P(x,y) \wedge P(x,z) \Rightarrow y = z]$$

$$\psi := \forall y \exists x [P(x,y)]$$

$$\rho := \forall x \forall y \forall z [P(x,z) \wedge P(y,z) \Rightarrow x = y]$$

La formula ϕ esprime che il predicato binario P è una funzione, ψ dice che P è suriettiva e ρ dice che P è iniettiva. Consideriamo ora

$$\forall P[(\phi \wedge \psi) \Rightarrow \rho] \tag{1.11}$$

La formula del secondo ordine (1.11) afferma che se una funzione da un insieme a se stesso è suriettiva, allora è anche iniettiva, e questo accade se, e solo se, il dominio è finito.

1.2.3 Logica dei grafi

Descrivere un grafo con una struttura logica permette di formulare le proprietà dei grafi con formule logiche, che saranno fondamentali per il teorema di Courcelle.
La logica del primo ordine dei grafi considera formule in cui le variabili e i predicati riguardano singoli vertici e archi di un grafo, mentre la logica del secondo ordine dei grafi consente la quantificazione su insiemi di vertici o archi. Se $G = (V, E)$ è un grafo, è definita una relazione binaria $adj_G(x, y)$, chiamata relazione di adiacenza, sui vertici di G nel seguente modo:

$$adj_G(x,y) \Leftrightarrow \textit{esiste un arco in } G \textit{ da } x \textit{ a } y. \tag{1.12}$$

Può inoltre essere definita una relazione ternaria edg_G tra un arco e due vertici di G come segue

$$edg_G(e,x,y) \Leftrightarrow e \textit{ è un arco da } x \textit{ a } y. \tag{1.13}$$

1.2.4 Logica dei grafi del primo ordine

La logica del primo ordine applicata ai grafi permette di quantificare su nodi e archi che rappresentano i termini della teoria.
Si possono formulare affermazioni e teoremi riguardanti proprietà specifiche dei grafi e delle loro strutture.

Esempio 1.2.8. La condizione per cui un grafo $G = (V, E)$ non abbia vertici isolati può essere espressa con le seguenti formule:

$$\forall x \exists y (adj_G(x,y) \vee adj_G(y,x)), \tag{1.14}$$

o equivalentemente

$$\forall x \exists y \exists e (edg_G(e,x,y) \vee edg_G(e,y,x)). \tag{1.15}$$

Esempio 1.2.9. Si consideri il seguente predicato $P(x)$:

$$\forall y_1 \forall y_2 \forall y_3 (adj_G(x, y_1) \wedge adj_G(x, y_2) \wedge adj_G(x, y_3) \Rightarrow (y_1 = y_2 \vee y_1 = y_3 \vee y_2 = y_3)). \tag{1.16}$$

$P(x)$ esprime che il vertice x del grafo G preso in considerazione ha grado uscente al più due.
La formula

$$\forall x P(x), \tag{1.17}$$

dice che il grado uscente massimo di G è al più due.

Esempio 1.2.10 (grafo indiretto)**.** Si può esprimere l'assioma dei grafi indiretti nel seguente modo

$$\forall x \forall y (adj_G(x, y) \Rightarrow adj_G(y, x)). \tag{1.18}$$

Esempio 1.2.11. La seguente formula esprime che ogni vertice di G ha almeno un arco uscente e un arco entrante

$$\forall x \exists y \exists z (adj_G(y, x) \wedge adj_G(x, z)). \tag{1.19}$$

Oltre a quelle mostrate possono essere espresse tante altre proprietà dei grafi utilizzando la logica del primo ordine.

1.2.5 Logica monadica del secondo ordine

La logica monadica del secondo ordine (MSO) è una restrizione della logica del secondo ordine nella quale i quantificatori del secondo ordine sono limitati a quantificare solo su predicati monadici (unari).
La logica monadica del secondo ordine è spesso utilizzata in contesti specifici dove la restrizione a insiemi monadici è sufficiente per formalizzare i concetti desiderati senza introdurre la complessità aggiuntiva associata all'utilizzo di insiemi più generici.

Esempio 1.2.12.

$$\exists X \forall Y \forall x (Y(x) \Rightarrow X(x)), \tag{1.20}$$

questa formula afferma che esiste un insieme X tale che, per ogni insieme Y, tutti gli elementi di Y sono anche elementi di X, in questo caso X coincide con tutto il dominio e la formula può essere espressa come $\forall x : X(x)$.

1. $\exists X$ denota il quantificatore esistenziale su insiemi,
2. $\forall Y$ denota il quantificatore universale su insiemi,
3. $\forall x$ denota il quantificatore universale su individui.

Questa rappresentazione logica può essere utilizzata per esprimere concetti come l'esistenza di un insieme che contiene tutti gli elementi di altri insiemi.

Nell'ambito della teoria dei grafi la logica monadica del secondo ordine si presenta in due varianti:

1. MSO_1, che consente la quantificazione solo su vertici e insiemi di vertici,
2. MSO_2, che consente la quantificazione su vertici, archi, insiemi di vertici e insiemi di archi.

Esempio 1.2.13. La formula che esprime la connessione di un grafo può essere scritta in MSO_1 nel seguente modo:

$$\forall S(\forall x(x \in S) \vee \forall y(\neg(y \in S)) \vee \exists x \exists y(x \in S \wedge \neg(y \in S) \wedge adj_G(x,y))). \tag{1.21}$$

Esempio 1.2.14 (3-colorazione)**.** Sia $Part(X,Y,Z)$ la formula che esprime che gli insiemi X, Y, Z siano una partizione del dominio, definita come segue:

$$\begin{aligned} \forall x\big((x \in X \vee x \in Y \vee x \in Z) \wedge (\neg(x \in X \wedge x \in Y) \wedge \\ \neg(x \in Y \wedge x \in Z) \wedge \neg(x \in X \wedge x \in Z))\big) \end{aligned} \tag{1.22}$$

La formula che esprime la 3-colorazione di un grafo G può essere scritta nel seguente modo:

$$\begin{aligned} \exists X \exists Y \exists Z\big(&Part(X,Y,Z) \wedge \\ &\forall x \forall y(adj_G(x,y) \wedge \neg(x = y) \Rightarrow \neg(x \in X \wedge y \in X) \wedge \\ &\neg(x \in Y \wedge y \in Y) \wedge \neg(x \in Z \wedge y \in Z))\big). \end{aligned} \tag{1.23}$$

Per un numero intero generico k si possono costruire formule simili a questa per esprimere la k-colorazione di un grafo. Formule di questo genere non possono essere scritte in logica del primo ordine.

Viene ora mostrato un esempio di una prorpietà esprimibile in MSO_2 ma non in MSO_1.

Esempio 1.2.15. Sia $G = (V,E)$ un grafo diretto; viene scritta in MSO_2 la proprietà che un insieme di archi X sia un cammino diretto da un vertice x a un vertice y, con $x \neq y$. Per ogni $U \subset E$ sia $V(U)$ l'insieme di vertici che siano estremi di almeno un arco di U. Sia $\theta(x,y,Y)$ la formula del primo ordine che esprime le seguenti condizioni:

1. x e y sono vertici e $x \neq y$,
2. Y è un insieme di archi,
3. x è estremo di esattamente un arco in Y ed è l'estremo di partenza dell'arco; y è estremo di esattamente un arco in Y ed è l'estremo di arrivo dell'arco,
4. ogni vertice $z \in V(Y) \setminus \{x,y\}$ è estremo di esattamente due archi in Y, di cui è l'estremo di partenza di uno e l'estremo di arrivo dell'altro.

Questa formula dice che Y è l'unione dell'insieme X di archi di un cammino da x a y e l'insieme di archi di qualche circuito disconnesso da questo cammino.
Quindi X è caratterizzato come il più piccolo sottoinsieme Z di Y, tale che valga $\theta(x,y,Z)$. Segue che la formula MSO_2 desiderata é definita come segue:

$$\theta(x,y,X) \wedge \forall Z(Z \subset X \Rightarrow \neg\theta(x,y,Z)). \tag{1.24}$$

Nella quale $Z \subset X$ sostituisce

$$\forall z(z \in Z \Rightarrow z \in X) \wedge \exists x(x \in X \wedge \neg(x \in Z)).$$

Non esiste una variante MSO_1 della formula (1.24).

Capitolo 2

Teorema di Courcelle

2.1 Enunciato

Il teorema di Courcelle è un risultato fondamentale nella teoria dei grafi e nella teoria degli algoritmi. Il teorema stabilisce condizioni sotto le quali proprietà dei grafi, espresse in logica monadica del secondo ordine (MSO_2), sono decidibili in tempo lineare rispetto alla dimensione del grafo.

Definizione 2.1.1. Siano $G = (V, E)$ un grafo e ϕ una formula in un linguaggio logico, la relazione di soddisfacibilità $G \models \phi$ indica se la formula ϕ è vera nel grafo G.

Esempio 2.1.2. Sia $\phi : \exists p \in Path(G), p(u, v)$ una formula del primo ordine, dove $Path(G)$ è il predicato che indica l'insieme dei cammini nel grafo G e $p(u, v)$ è una relazione binaria che indica che esiste un cammino tra i vertici u e v. La relazione di soddisfacibilità $G \models \phi$ indica se questa formula è vera nel grafo G, ovvero se effettivamente esiste un cammino tra quei due vertici.

La relazione di soddisfacibilità nel contesto di una struttura grafo consiste nel valutare formule logiche basate sulle proprietà e le relazioni all'interno del grafo.

Definizione 2.1.3. Sia ϕ una formula espressa in logica del secondo ordine, la *dimensione* $|\phi|$ è definita induttivamente come segue:

1. $|\phi| = 1$ se ϕ è una formula atomica,
2. $|\neg\phi| = |\phi| + 1$,
3. $|\phi \wedge \psi| = |\phi \vee \psi| = |\phi \Rightarrow \psi| = |\phi \Leftrightarrow \psi| = |\phi| + |\psi| + 1$,
4. $|\exists X.\phi| = |\forall X.\phi| = |\exists x.\phi| = |\forall x.\phi| = |\phi| + 1$.

La dimensione di una formula espressa in logica del secondo ordine è una misura della complessità della formula stessa.
Si può ora enunciare il seguente.

Teorema 2.1.4 (Teorema di Courcelle)**.** *Sia $G = (V, E)$ un grafo con larghezza d'albero w e sia ϕ una formula in MSO_2 allora $G \models \phi$ può essere decisa in tempo $O(f(|\phi|, w) \cdot |G|)$, per qualche funzione computabile f.*

La funzione f dipende dalla formula ϕ presa in considerazione e dalla larghezza dell'albero.
L'approccio tipico per dimostrare il teorema di Courcelle coinvolge la costruzione di un automa ad albero finito bottom-up che agisce sulle decomposizioni ad albero del grafo dato.
Il processo bottom-up consiste nell'iniziare il processo dalle foglie (parte inferiore) dell'albero e procedere verso la radice. Nel contesto di un automa ad albero finito bottom-up, l'automa elabora le etichette dei nodi partendo dalle foglie e muovendosi verso l'alto.

2.2 Automi ad albero

Viene ora connessa la logica monadica del secondo ordine con la nozione di automi ad albero. Essi svolgono un ruolo importante in molte applicazioni, tra cui i sistemi di riscrittura e la dimostrazione automatica di teoremi.
Un automa ad albero è una tipologia di macchina a stati finiti che tratta strutture ad albero invece delle stringhe tipiche delle macchine a stati più convenzionali.
Un automa ad albero è una macchina a stati finiti utilizzata per riconoscere linguaggi ad albero. Un linguaggio ad albero è l'insieme di tutti gli alberi i cui vertici sono etichettati da un insieme fisso Σ e soddisfano alcune proprietà richieste.

Esempio 2.2.1. Sia $\Sigma = \{a, b\}$ e sia $\mathcal{T} = \{T | T \text{ è un albero etichettato da } \Sigma\}$, un esempio di linguaggio ad albero è

$$L = \{T \in \mathcal{T} | \text{il numero totale di nodi etichettati da } a \text{ è} \leq 5\}. \tag{2.1}$$

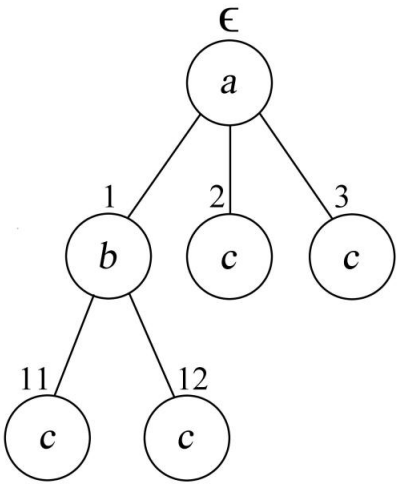
Definizione 2.2.2. Un automa ad albero a stati finiti è una tupla $A = (Q, \Sigma, q_0, F, \delta)$ dove:

1. Q è un insieme finito di stati,
2. Σ è l'alfabeto utilizzato per etichettare i nodi dell'albero,
3. $q_0 \in Q$ è lo stato iniziale,
4. $F \subseteq Q$ è l'insieme degli stati che possono essere accettati,
5. δ è la funzione di transizione $\delta : Q \times Q \times \Sigma \to Q$.

L'automa inizia ad associare i nodi dell'albero di input con alcuni stati a partire dalle foglie e infine l'albero di input viene accettato se lo stato associato al nodo radice è in F.

Definizione 2.2.3. Sia $T = (V, E)$ un albero etichettato da un insieme Σ, si definisce *funzione di etichettatura* la funzione $h : V \to \Sigma$ che associa ad ogni nodo $v \in V$ la corrispondente etichetta in Σ.

Esempio 2.2.4. Si consideri il seguente albero etichettato $T = (V, E)$, in cui $V = \{\epsilon, 1, 2, 3, 11, 12\}$ e $\Sigma = \{a, b, c\}$.



La funzione di etichettatura h risulta essere
$h(\epsilon) = a$
$h(1) = b$
$h(2) = h(3) = h(11) = h(12) = c$.

Definizione 2.2.5. Sia $A = (Q, \Sigma, q_0, F, \delta)$ un automa ad albero a stati finiti e sia $T = (V; E)$ un albero con la corrispondente funzione di etichettatura $h : V \to \Sigma$. Un'esecuzione di A su T è una funzione $r : V \to Q$ tale che

1. se s è una foglia etichettata da a, allora $r(s) = \delta(q_0, q_0, a)$,

2. siano s_1 ed s_2 figli di s, se $r(s_1) = q$, $r(s_2) = q'$ e $h(s) = a$, allora $r(s) = \delta(q, q', a)$.

Un'esecuzione r è di successo se $r(\epsilon) \in F$, dove ϵ è la radice dell'albero.

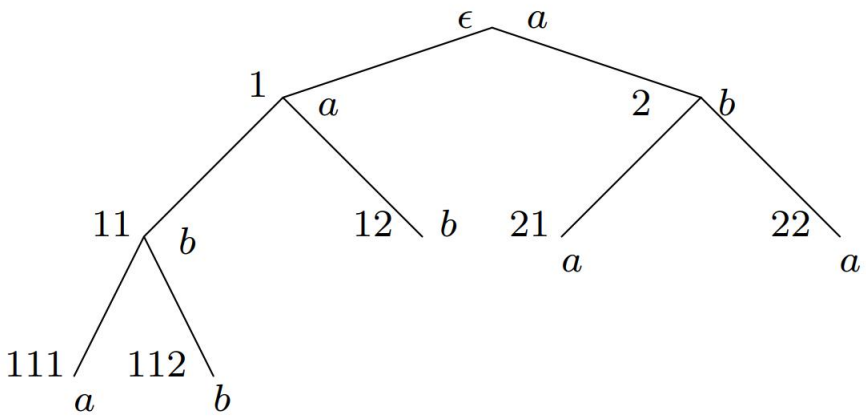
Esempio 2.2.6. Si consideri l'automa ad albero a stati finiti $A = (Q, \Sigma, q_0, F, \delta)$ definito nel seguente modo

1. $Q = \{q_0, q_a, q_b, q, q'\}$

2. $\Sigma = \{a, b\}$

3. $F = \{q'\}$.

Con funzione di transizione δ avente le seguenti proprietà:
$\delta(q_0, q_0, a) = q_a$
$\delta(q_0, q_0, b) = q_b$
$\delta(q_a, q_b, b) = q$
$\delta(q_a, q_a, b) = q'$
$\delta(q, q_b, a) = q$
$\delta(q, q', a) = q'$.
Sia ora $T = (V, E)$ il seguente albero etichettato da Σ



Se $r : V \to Q$ è un'esecuzione di A su T allora si ottiene:

1. per le foglie $r(111) = r(21) = r(22) = q_a$ e $r(112) = r(12) = q_b$
2. $r(11) = \delta(q_a, q_b, b) = q$
3. $r(1) = \delta(q, q_b, a) = q$
4. $r(2) = \delta(q_a, q_a, b) = q'$
5. $r(\epsilon) = \delta(q, q', a) = q'$

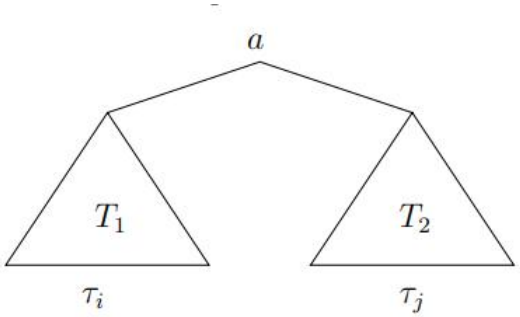
Dato che $r(\epsilon) = q' \in F$ l'esecuzione è di successo e l'albero viene accettato dall'automa.

Definizione 2.2.7. Un linguaggio ad albero L si dice regolare se esiste un automa ad albero a stati finiti A che accetta L.

Viene ora presentato il seguente teorema, il quale mostra che i linguaggi ad albero regolari sono esattamente quelli definibili in logica monadica del secondo ordine.

Teorema 2.2.8. *Un linguaggio ad albero L è definibile in MSO se, e solo se, L è regolare.*

Dimostrazione. $(\Leftarrow)$
Sia L un linguaggio ad albero accettato da un automa ad albero a stati finiti A. Per ogni stato q, sia X_q l'insieme di nodi nei quali l'esecuzione di A è nello stato q. In seguito si controlla, in primo ordine, che ogni foglia etichettata da a sia in X_q per qualche $q \in \delta(q_0, q_0, a)$, che le funzioni di transizione siano costruite correttamente e che la radice sia in uno degli stati accettati. Questa è una formula *MSO*.
$(\Rightarrow)$
Per l'altro verso dell'implicazione bisogna mostrare che formule *MSO* definiscono solo linguaggi regolari. Sia Φ una formula *MSO* di rango k. Siano $\tau_0, ..., \tau_m$ tutti i tipi *MSO* di rango k (si veda [10] per la definizione di tipi *MSO* di rango k), in cui τ_0 è il tipo dell'albero vuoto e $\{\tau_0, ..., \tau_m\}$ è l'insieme degli stati di un automa A_Φ. Sia $\tau_l \in \delta(\tau_i, \tau_j, a)$, esistono degli alberi T_1 e T_2 i cui tipi *MSO* di rango k siano rispettivamente τ_i e τ_j tali che il tipo *MSO* dell'albero ottenuto congiungendo T_1 e T_2 come figli di un nodo radice etichettato da a sia τ_l.



Si può mostrare che A_Φ è un automa ad albero a stati finiti che accetta il linguaggio ad albero $\{T|T \models \Phi\}$. □

2.3 Dimostrazione del teorema di Courcelle

Viene descritto un algoritmo che prende in input un grafo $G = (V, E)$ e un formula ϕ in *MSO* e decide se $G \models \phi$.
Input: Grafo $G = (V, E)$ **di larghezza dell'albero** w **e una formula** ϕ **espressa in** *MSO*.
Output: Se $G \models \phi$ **o** $G \not\models \phi$.
Passo 1: si costruisce T_G, la decomposizione ad albero di G.
Passo 2: si etichettano i vertici di T_G utilizzando un insieme Σ adeguato, il nuovo albero etichettato verrà indicato con $\mathcal{T}$.
Passo 3: si costruisce una formula ϕ' espressa in *MSO* tale che $G \models \phi \Leftrightarrow \mathcal{T} \models \phi'$.
Passo 4: si verifica se $\mathcal{T} \models \phi'$ e ritorna *si* o *no* di conseguenza.
Bisognerà dimostrare che il tempo di esecuzione dell'algoritmo è $O(f(|\phi|, w) \cdot |G|)$.

2.3.1 Passo 1

Nel primo passo viene costruita la decomposizione ad albero di un grafo dato G. Utilizzando un opportuno algoritmo si può costruire una decomposizione ad albero di G di larghezza w in tempo $O(2^{w^2} \cdot |G|)$.
Nel passo 3 si costruisce un automa ad albero A per ϕ' e si controlla se A accetta $\mathcal{T}$. Costruire A richiede tempo $O(f(|\phi'|))$ per qualche funzione f computabile e controllare se A accetta $\mathcal{T}$ richiede tempo $O(|G|)$.
Se i passi 2 e 3 possono essere completati in tempo $O(f(|\phi|, w) \cdot |G|)$ la dimostrazione è conclusa.

Teorema 2.3.1. *Per ogni $w \in \mathbb{N}$ esiste un algoritmo in tempo lineare che controlla se un grafo $G = (V, E)$ ha larghezza al più w, in caso affermativo ritorna una decomposizione ad albero di G con larghezza al più w.*

Si veda [7] per una dimostrazione.
L'idea principale di questo algoritmo che serve ad ottenere una decomposizione ad albero di G è la seguente: i vertici vengono suddivisi in due insiemi, uno con vertici di 'basso grado', e uno con vertici di 'alto grado'. Si può dimostrare che per grafi con larghezza dell'albero al più w, ci sono solo 'pochi' vertici di alto grado.

Vengono distinti due casi.

1. Un numero "sufficientemente elevato" di vertici a basso grado è adiacente a uno o più altri vertici di basso grado. Si costruisce il grafo G' ottenuto contraendo tutti gli archi in un arco di grado massimale. Ricorsivamente, si calcola una decomposizione ad albero con larghezza al più w di G', oppure concludiamo che la larghezza dell'albero di G', e quindi la larghezza di G è maggiore di w. Da questa decomposizione ad albero, si può facilmente costruire una decomposizione ad albero di G con larghezza di albero al più $2w + 1$.

2. "Solo pochi" vertici di basso grado sono adiacenti a uno o più altri vertici di basso grado. Si dimostra che una certa collezione di archi può essere aggiunta a G senza aumentare la larghezza dell'albero da un numero al più w a un numero maggiore di w. Ricorsivamente, viene calcolata una decomposizione ad albero con larghezza dell'albero al più w di G', ottenuta rimuovendo tutti i vertici I-simpliciali dal grafo migliorato di G. Data tale decomposizione ad albero di G', si calcola facilmente una decomposizione ad albero di G con larghezza dell'albero al più w.

In entrambi i casi, il tempo di esecuzione dei passaggi non ricorsivi è lineare, e ciascun G' ha dimensione al più una frazione costante della dimensione di G. Segue che l'algoritmo utilizza tempo lineare.

2.3.2 Passo 2

Senza perdere di generalità si può assumere che T_G sia un albero binario. Ogni $S \subseteq V$ di dimensione al più $w + 1$ può essere codificato da una stringa di lunghezza $w + 1$−la lista degli elementi di S, e ripetuta in modo da assicurare che la lista degli elementi della stringa sia esattamente $w + 1$.
Se $T_G = (V_T, E_T)$ si ha quindi una decomposizione ad albero (T_G, B_T) di larghezza w di un grafo $G = (V, E)$ tale che $B_T : V_T \to V^{w+1}$. Si può notare che $|V_T| = O(|V|)$.
Non si può utilizzare X_T per etichettare i vertici di T_G in quanto B_T dipende dalla dimensione di G, per etichettare i vertici di un albero è necessario avere un insieme Σ fissato.

Si costruisce l'albero etichettato $\mathcal{T} = (T_G, L_T)$ utilizzando un opportuno Σ.
L_T è definito come segue:

$$\forall u \in V_T, \quad L(u) = (\lambda_1, \lambda_2, \lambda_3)$$

dove

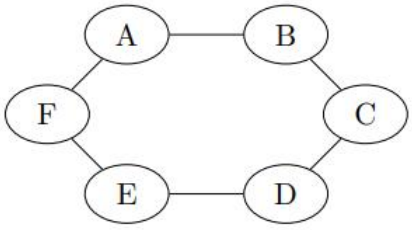
1. $\lambda_i \subseteq \{1, 2, ..., w + 1\} \times \{1, 2, ..., w + 1\}$, *per* $i = 1, 2, 3$.

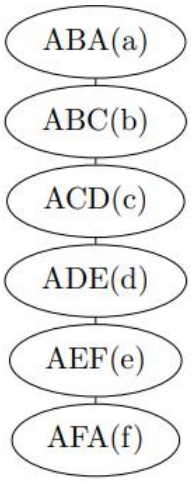
2. $\lambda_1 = \{(i, j) | l'i - esimo\ vertice\ in\ B_T(u)\ e\ il\ j - esimo\ vertice\ in\ B_T(u)$ *hanno un arco che li unisce in* $G\}$

3. $\lambda_2 = \{(i,j)|l'i - esimo\ vertice\ in\ B_T(u)\ e\ il\ j - esimo\ vertice\ in\ B_T(u)\ coincidono\}$

4. $\lambda_2 = \{(i,j)|l'i - esimo\ vertice\ in\ B_T(u)\ e\ il\ j - esimo\ vertice\ in\ B_T(v)\ coincidono,\ in\ cui\ v\ è\ genitore\ di\ u\ in\ T_G\}$.

Esempio 2.3.2. Si consideri il seguente grafo e la sua corrispondente decomposizione ad albero di larghezza $w = 2$.





La corrispondente etichettatura è la seguente.

1. $L_T(a) = \{(12,21,23,32)(13,31)()\}$
2. $L_T(b) = \{(12,21,23,32)()(11,13)\}$
3. $L_T(c) = \{(23,32)()(11,23)\}$
4. $L_T(d) = \{(23,32)()(11,23)\}$
5. $L_T(e) = \{(13,31,23,32)()(11,23)\}$
6. $L_T(f) = \{(12,21,23,32)(13,31)(11,31,23)\}$.

Questa etichettatura dipende solo da w e non dalla dimensione di G e quindi ha un insieme Σ costante.
Questa etichettatura può essere computata in tempo $O(|V_T|) = O(|V|) = O(|G|)$.

2.3.3 Passo 3

In questa sezione bisogna costruire ϕ' tale che $G \models \phi \Leftrightarrow \mathcal{T} \models \phi'$. Per fare ciò bisogna rappresentare i vertici di G come tuple di vertici di $\mathcal{T}$.

Ogni $S \subseteq V$ è codificato da una stringa $Tree(S) = (U_1, U_2, ..., U_{w+1})$, dove $U_i \subseteq V_T$ tale che

$$U_i = \{v \in V_T | l'i - esimo\ valore\ in\ B_T(v) \in S\}$$

Nel caso precedente si ha ad esempio
$Tree(\{A, C, E\}) = (\{a, b, c, d, e, f\}, \{c, e\}, \{a, b, d, f\})$
$Tree(\{A\}) = (\{a, b, c, d, e, f\}, \{\}, \{a, f\})$
$Tree(\{B, F\}) = (\{\}, \{a, b, f\}, \{e\})$
$Tree(\{C, D, E\}) = (\{\}, \{c, d, e\}, \{b, c, d\})$
Si può notare che non tutte le tuple di vertici dell'albero corrispondono ad un sottoinsieme dei vertici del grafo. Nel precedente esempio $(\{a\}, \{\}, \{\}) \neq Tree(S)$ per ogni $S \subseteq V$.

Lemma 2.3.3. *Una tupla di insiemi di nodi di $\mathcal{T}$, $\overline{X} = (X_1, X_2, ..., X_{w+1})$ è rappresentata da $Tree(S)$ per qualche S con $|S| \geq 1$ se, e solo se valgono i seguenti.*

(i) Se $(i, j) \in \lambda_2(u)$ allora $u \in X_i \Leftrightarrow u \in X_J$.

(ii) Se $(i, j) \in \lambda_3(u)$ allora $u \in X_i \Leftrightarrow v \in X_J$, dove v è genitore di u.

Inoltre si ha che overline$X = Tree(S)$ per qualche S tale per cui $|S| = 1$ se, e solo se valgono le due affermazioni precedenti e

(iii) se $u \in X_i$ e $u \in X_j$ allora $(i, j) \in \lambda_2(u)$.

(iv) Se $u \in X_i$ e $v \in X_j$, dove v è genitore di u allora $(i, j) \in \lambda_3(u)$.

(v) $\bigcup_{i=1}^{w+1} X_i$ forma una componente connessa di T_G.

Dimostrazione. $\Rightarrow$ è facile da vedere in entrambi i casi.
Bisogna dimostrare $\Leftarrow$

Caso $|S| \geq 1$
Sia $\overline{X} = (X_1, X_2, ..., X_{w+1})$ una tupla che soddisfi (i) e (ii). Sia

$$Set(\overline{X}) := \{x \in V | \exists i \exists u, u \in X_i \text{ e } l'i - esima \text{ posizione in } B_T(u) = x\}.$$

Bisogna provare che $Tree(Set(\overline{X})) = \overline{X}$.
Supponendo per assurdo che $Tree(Set(\overline{X})) \neq \overline{X}$, allora esistono u_1 e u_2 tali per cui l'i-esima posizione in $B_T(u_1)$ = j-esima posizione in $B_T(u_2) = x$, tali che $u_1 \in X_i$ e $u_2 \notin X_i$.

1. Dato che $x \in B_T(u_1)$ e $x \in B_T(u_2)$, x compare nelle etichette di ogni vertice nel cammino che va da u_1 a u_2.

2. Senza perdere di generalità si può assumere $u_1 = u_2$ o u_1 genitore di u_2.

3. Ma si ottiene che $\overline{X}$ contraddice (i) se $u_1 = u_2$ e $\overline{X}$ se u_1 è genitore di u_2.

Questo porta ad un assurdo.

Caso $|S| = 1$.
Sia $\overline{X}$ una tupla che soddisfi (i), (ii), (iii), (iv) e (v). Si consideri $S = Set(\overline{X})$.

Dato che $\overline{X}$ soddisfa (i) e (ii), dal precedente risultato $\overline{X} = Tree(S)$.
Bisogna provare che $|S| = 1$.
Supponendo per assurdo che $|S| > 1$, allora ci sono due vertici $u_1 \in X_i$ e $u_2 \in X_j$ tali per cui l'i-esima posizione in $B_T(u_1)$ uguaglia la j-esima posizione in $B_T(u_2)$.

1. Dato che $\overline{X}$ soddisfa (v), senza perdere di generalità, si ha $u_1 = u_2$ o $u1$ genitore di u_2.

2. Questo contraddice (iii) o (iv).

□

Una volta provata la validità di queste condizioni, si possono scrivere in formule *MSO*.
Siano ϕ_i, ϕ_{ii}, ϕ_{iii}, ϕ_{iv} e ϕ_v le formule MSO di (i), (ii), (iii), (iv) e (v) definite nel seguente modo.

$$\phi_i(X_1, ..., X_{w+1}) = \forall x \bigwedge_{i,j \in \{1,2,...,w+1\}} \bigwedge_{a:(i,j) \in \lambda_2(a)} (X_i x \Leftrightarrow X_j x) \tag{2.2}$$

$$\phi_{ii}(X_1, ..., X_{w+1}) = \forall x \forall y \bigwedge_{i,j \in \{1,2,...,w+1\}} \bigwedge_{a:(i,j) \in \lambda_3(a)} E_{T_G} xy \Rightarrow (X_i x \Leftrightarrow X_j x) \tag{2.3}$$

$$\phi_{iii}(X_1, ..., X_{w+1}) = \forall x (X_i x \wedge X_j x) \Rightarrow \bigvee_{a:(i,j) \in \lambda_2(a)} Q_a x \tag{2.4}$$

$$\phi_{iv}(X_1, ..., X_{w+1}) = \forall x \forall y (X_i x \wedge X_j y \wedge E_{T_G} yx) \Rightarrow \bigvee_{a:(i,j) \in \lambda_3(a)} Q_a x \tag{2.5}$$

$$\phi_{iv}(X_1, ..., X_{w+1}) = Connected(\bigcup_{i=1}^{w+1} X_i). \tag{2.6}$$

Dove la formula *Connected* è definita nel seguente modo.

$$\begin{aligned} Connected(Y) :=& \forall Y_1 \forall Y_2 ((Y_1 \cup Y_2 = Y) \wedge (Y_1 \cap Y_2 = \Phi) \wedge Y_1 \neq \Phi \wedge Y_2 \neq \Phi) \\ & \Rightarrow (\exists x \exists y Y_1 x \wedge Y_2 y \wedge E_{T_G} xy) \end{aligned}$$

Data la formula $\phi(x_1, x_2, ..., x_n, Y_1, Y_2, ..., Y_n)$ si può costruire la formula $\phi'(\overline{X_1}, \overline{X_2}, ..., \overline{X_n}, \overline{Y_1}, \overline{Y_2}, ..., \overline{Y_n})$ tale che

$$G \models \phi(x_1, x_2, ..., x_n, Y_1, Y_2, ..., Y_n) \Leftrightarrow \mathcal{T} \models \phi'(\overline{X_1}, \overline{X_2}, ..., \overline{X_n}, \overline{Y_1}, \overline{Y_2}, ..., \overline{Y_n}). \tag{2.7}$$

ϕ' è costruita induttivamente:

Caso base

1. Se $\phi : (x_1 = x_2)$ allora $\phi' : \bigwedge_{i=1}^{w+1} X_{1_i} = X_{2_i}$.

2. Se $\phi : E_G xy$ allora $\phi' : \exists u \bigvee_{a:(i,j) \in \lambda_2(a)} X_{a_i} u \wedge X_{a_j} u \wedge Q_a u$.

3. Se $\phi : Yx$ allora $\phi' : \bigwedge_{i=1}^{w+1} X_i \subseteq Y_i$.

Passo induttivo

1. Se $\phi : \phi_1 \wedge \phi_2$ allora $\phi' : \phi'_1 \wedge \phi'_2$.
2. Se $\phi : \phi_1 \vee \phi_2$ allora $\phi' : \phi'_1 \vee \phi'_2$.
3. Se $\phi : \neg\phi_1$ allora $\phi' : \neg\phi'_1$.
4. Se $\phi : \exists Y \phi_1$ allora $\phi' : \exists \overline{Y}(\phi_i(\overline{Y}) \wedge \phi_{ii}(\overline{Y}) \wedge \phi'_1)$.
5. Se $\phi : \exists x \phi_1$ allora $\phi' : \exists \overline{X}(\phi_i(\overline{X}) \wedge \phi_{ii}(\overline{X}) \wedge \phi_{iii}(\overline{X}) \wedge \phi_{iv}(\overline{X}) \wedge \phi_v(\overline{X}) \wedge \phi'_1)$.

Capitolo 3

Computazione del numero di incroci in tempo quadratico

Il teorema di Courcelle è un potente strumento per la risoluzione di problemi decisionali su grafi, consentendo la definizione di algoritmi efficienti per problemi che altrimenti sarebbero computazionalmente complessi. In questo capitolo viene presentata una delle sue applicazioni.
Viene mostrato che per ogni $k \geq 0$ fissato, esiste un algoritmo che decide in tempo quadratico se un grafo dato ha un numero di incroci al più k. Se questo è il caso, computa una raffigurazione del grafo con al più k incroci.

3.1 Contesto teorico

I grafi presentati in questo capitolo sono indiretti e senza cicli, ma possono avere archi multipli.

Definizione 3.1.1 (raffigurazione e numero di incroci)**.** Una raffigurazione di un grafo $G = (V, E)$ è una funzione Δ che associa ad ogni vertice $v \in V$ un punto $\Delta(v) \in \mathbb{R}^2$ e ad ogni arco $e \in E$ una curva semplice $\Delta(e)$ in $\mathbb{R}^2$ tali che:

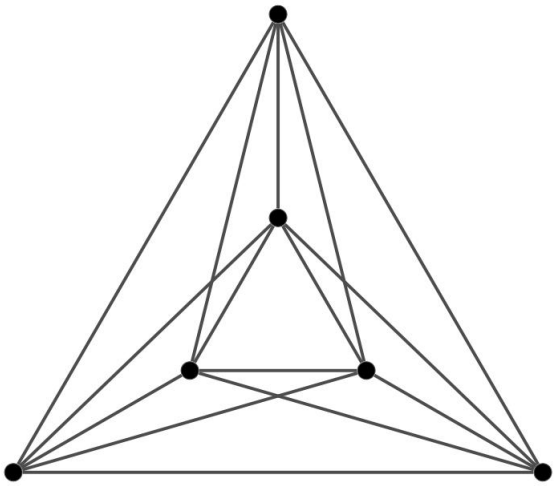
1. Per due vertici distinti $v, w \in V$, punti $\Delta(v)$ e $\Delta(w)$ sono distinti in $\mathbb{R}^2$.
2. Per due archi distinti $e, f \in E$, le curve $\Delta(e)$ e $\Delta(f)$ hanno al più un punto interno in comune (eventualmente anche i loro estremi).
3. Per ogni arco $e \in E$ di estremi v e w, i due estremi della curva $\Delta(e)$ sono $\Delta(v)$ e $\Delta(w)$, si ha inoltre che $\Delta(u) \notin \Delta(e)$ per ogni $u \in V \setminus \{v, w\}$.
4. In ogni punto si intersecano al più due archi. Più precisamente $|\{e \in E | x \in \Delta(e)\}| \leq 2$ per ogni $x \in \mathbb{R}^2 \setminus \Delta(V)$.

Si definisce $\Delta(G) := \Delta(V) \cup \bigcup_{e \in E} \Delta(e)$.
Un punto $x \in \mathbb{R}^2 \setminus \Delta(V)$ con $|\{e \in E | x \in \Delta(e)\}| = 2$ è chiamato un incrocio di Δ. La cardinalità dell'insieme che contiene tutti gli incroci di Δ è definita come numero di incroci di Δ. Il *numero di incroci* di G è il minimo tra tutti i numeri di incroci presi

tra tutte le raffigurazioni di G. Una raffigurazione di un grafo è detta *planare* se il suo numero di incroci è zero.

Il numero di incroci di un grafo è una misura della complessità della sua rappresentazione su un piano. In un grafo, un incrocio si verifica quando due archi si intersecano. Il numero di incroci di un grafo è quindi il numero minimo di intersezioni al variare dei possibili modi di disegnarlo che si verificano quando il grafo è disegnato su un piano senza archi che si sovrappongono. Questa misura è significativa in numerosi contesti come nell'analisi di algoritmi di disegno grafico. Un grafo con un numero basso di incroci è generalmente più facile da interpretare e visualizzare rispetto a uno con un numero elevato di incroci.

Esempio 3.1.2 (numero di incroci di un grafo completo)**.** Si consideri il grafo completo con sei vertici $G = K_6$.
La seguente è una raffigurazione di G con tre incroci.



Il numero di incroci di G sono rappresentati graficamente dal numero di intersezioni degli archi di G. Quindi si ha che il numero di incroci è ≤ 3. Per provare che esso è esattamente uguale a 3, si assuma per assurdo che esista una raffigurazione di G con due incroci, si ottiene che:

1. Entrambi gli incroci coinvolgono quattro vertici distinti, cioè sono i 4 vertici due archi che si incrociano.

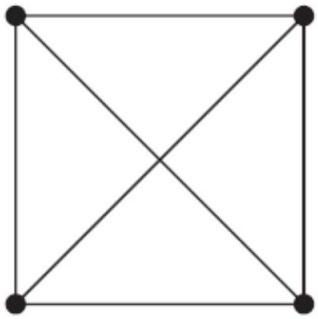
2. Dato che G ha 6 vertici, ne esiste uno v in entrambi gli incroci.

3. Se vengono cancellati il vertice v e gli archi aventi come estremo v, il grafo ottenuto risulterebbe avere zero incroci.

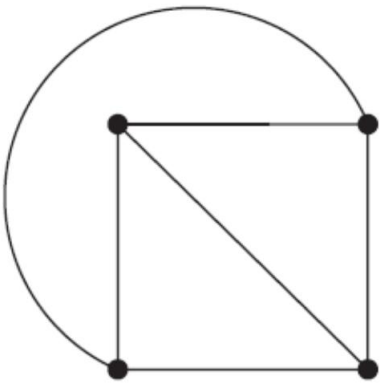
4. Si sarebbe così ottenuta una raffigurazione planare di K_5, ma questo è assurdo in quanto un grafo completo con 5 vertici non può presentare zero incroci.

Segue che il numero di incroci di K_6 è uguale a 3.

Esempio 3.1.3 (grafo planare)**.** Si consideri ad esempio il grafo completo K_4.

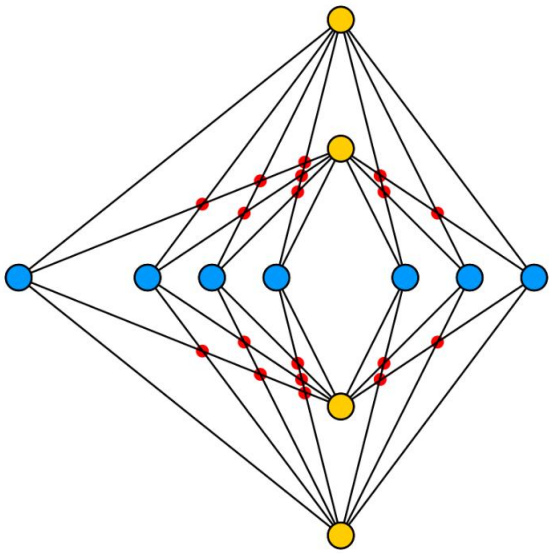


Esso può essere raffigurato senza archi che si intersecano, spostando uno degli archi dati da una diagonale al di fuori del perimetro del quadrato nel seguente modo:



Di conseguenza si deduce che K_4 è un grafo planare.

Esempio 3.1.4. Si consideri il grafo bipartito completo $K_{4,7}$, la seguente è una sua raffigurazione con 18 incroci (rappresentati dai punti rossi nel disegno).



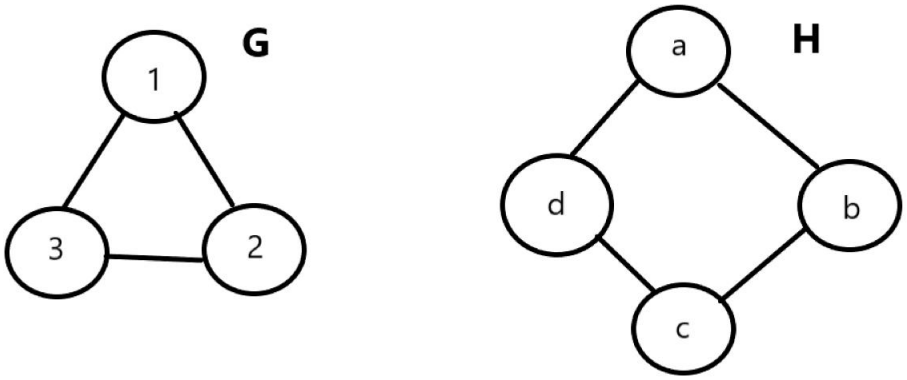
Definizione 3.1.5. Un'immersione topologica di un grafo $G = (V, E)$ in un grafo $H = (V', E')$ è una funzione h che associa ad ogni vertice $v \in V$ un vertice $h(v) \in V'$ e ad ogni arco $e \in E$ un cammino $h(e)$ in H tali che:

1. Per due vertici distinti $v, w \in V$, i vertici $h(v)$ e $h(w)$ sono distinti.
2. Per due archi distinti $e, f \in E$, i cammini $h(e)$ e $h(f)$ sono internamente disgiunti, cioè possono avere al massimo gli estremi in comune, ma non i vertici interni ai cammini.
3. Per ogni arco $e \in E$ di estremi v e w, i due estremi del cammino $h(e)$ sono $h(v)$ e $h(w)$; si ha inoltre che $h(u) \notin V^{h(e)}$ per ogni $u \in V \setminus \{v, w\}$, dove $V^{h(e)}$ rappresenta l'insieme dei vertici del cammino $h(e)$.

Si definisce $h(G) := h(V) \cup \bigcup_{e \in E} h(e)$.

Un'immersione topologica preserva la struttura del grafo originale nel grafo di destinazione, senza alterare la connettività o l'ordinamento dei vertici e degli archi.

Esempio 3.1.6. Si considerino i seguenti grafi G e H:

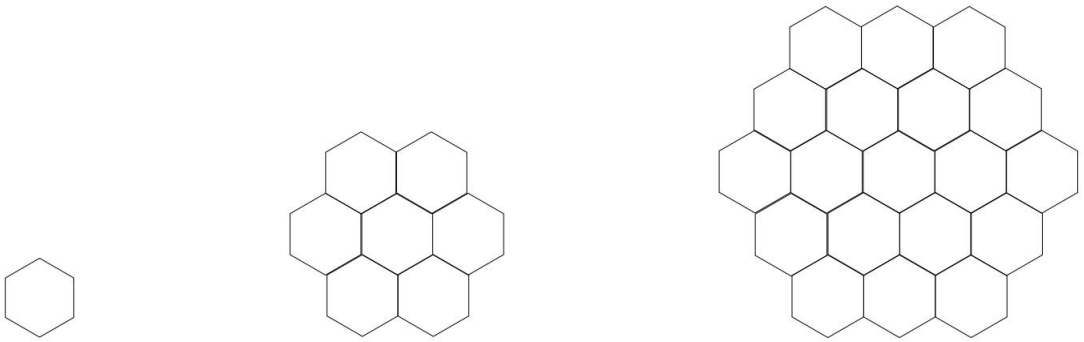


Un esempio di immersione topologica di G in H è dato dalla funzione h definita nel seguente modo:

1. $h(1) = a$;
2. $h(2) = b$;
3. $h(3) = c$;
4. $h(\{1,2\}) = cammino(a, b)$;
5. $h(\{2,3\}) = cammino(b, c)$;
6. $h(\{3,1\}) = cammino(c, a)$.

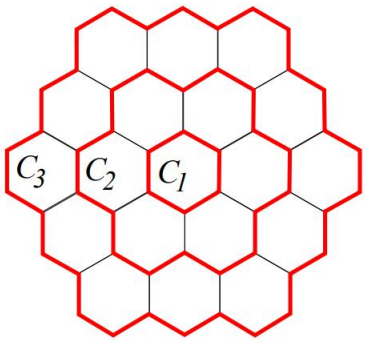
I vertici e gli archi di G vengono mappati in vertici e cammini di H rispettando le relazioni di adiacenza.

Definizione 3.1.7. Dato $r \geq 1$, sia H_r la *griglia esagonale di raggio r*. Invece di fornire una propria definizione, viene mostrata nella figura seguente cosa si intende per griglia esagonale di raggio r.



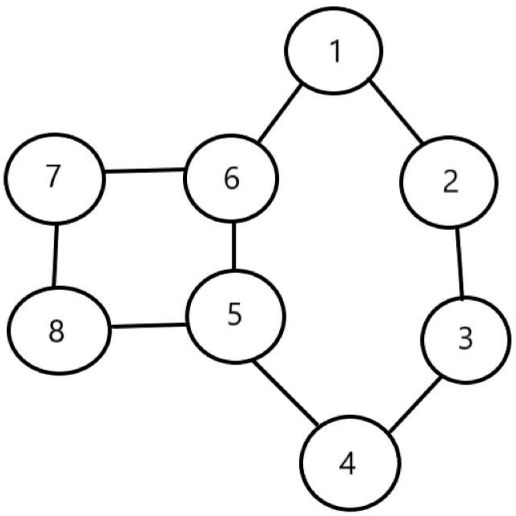
Esse rappresentano nell'ordine, le griglie esagonali H_1, H_2 e H_3.
Si definiscono *cicli principali* $C_1, ..., C_r$ di H_r i cicli concentrici numerati dall'interno all'esterno.

Esempio 3.1.8. La figura seguente mostra i cicli principali di H_3.



Definizione 3.1.9. Siano $G = (V, E)$ e $H = (V', E')$ due grafi tali che $H \subseteq G$, una *H-componente* di G è l'unione di una componente connessa C di $G \setminus H$ e di tutti gli archi che connettono C con H insieme ai loro estremi in H, oppure è un arco in $E \setminus E'$ i cui estremi sono entrambi in H.

Esempio 3.1.10. Sia G il seguente grafo:



Si consideri il sottografo H di G formato dai vertici $V' = \{5, 6\}$ e l'arco che li unisce. Un esempio di componente connessa C di $G \setminus H$ può essere formata dal sottografo di G contenente i vertici $\{1, 2, 3\}$. In questo caso una *H-componente* di G risulta essere il sottografo di G formato dai vertici $\{1, 2, 3, 6\}$ e dagli archi che uniscono questi vertici. Considerando gli stessi G e H. un altro esempio di componente connessa C' è il sottografo di G formato dai vertici $\{7, 8\}$ e l'arco che li unisce; in questo caso una *H-componente* di G risulta essere il grafo formato dai vertici $\{5, 6, 7, 8\}$ e dagli archi che uniscono questi vertici, con l'esclusione dell'arco che unisce i vertici 5 e 6.

Definizione 3.1.11. Sia $G = (V, E)$ un grafo e $h : H_r \to G$ un'immersione topologica. Si definisce *interno di* $h(H_r)$ il sottografo $h(H_r \setminus C_r)$, in cui C_r è il ciclo principale più esterno di H_r.
Si definiscono *collegamento di* $h(H_r)$ le $h(H_r)$-componenti di G che non hanno intersezione vuota con l'interno di $h(H_r)$.
L'immersione topologica h è *piana* se l'unione di $h(H_r)$ con tutti i sui collegamenti è planare.

Esempio 3.1.12. Si considerino le griglie esagonali H_2 e H_3, si consideri l'immersione topologica $h : H_2 \to H_3$ che manda ogni vertice ed ogni arco in se stesso. L'*interno* di $h(H_2)$ è proprio il sottografo H_1.

Teorema 3.1.13. *Per ogni $k, r \geq 1$ esiste un $s \geq 1$ tale che vale la seguente: se $G = (V, E)$ è un grafo con numero di incroci al più k e $h : H_s \to G$ è un'immersione topologica, allora esiste una sottogriglia $H_r \subseteq H_s$ tale che la restrizione $h|_{H_r}$ di h su H_r è piana.*

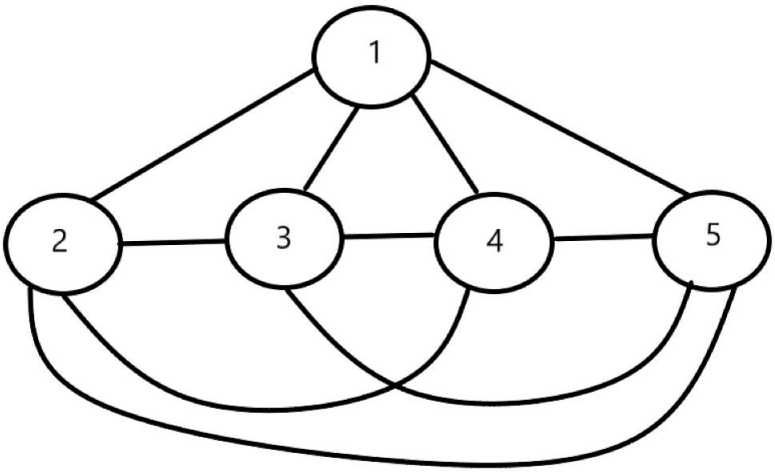
Si veda [14] per una dimostrazione del precedente teorema.

Teorema 3.1.14. *Sia $r \geq 1$, esiste $w \geq 1$ e un algoritmo in tempo lineare che, dato un grafo $G = (V, E)$, riconosce se G ha larghezza d'albero al più w o computa un'immersione topologica $h : H_r \to G$.*

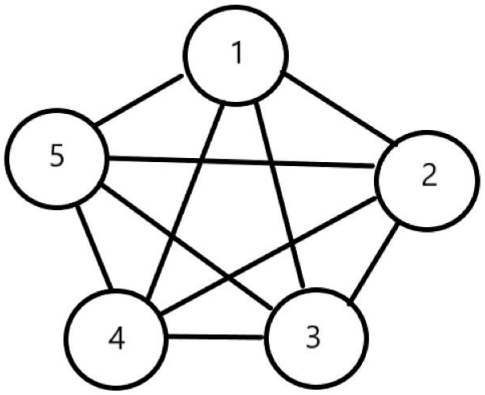
Si veda [15] per una dimostrazione del precedente teorema.

Definizione 3.1.15. Siano $l \geq 1$, $G = (V, E)$ un grafo e $F \subseteq E$ un sottoinsieme degli archi di G. Una *l-raffigurazione di G rispetto a F* è una raffigurazione Δ di G con numero di incroci al più l tale che nessun elemento di F sia coinvolto in alcun incrocio, cioè per ogni incrocio $x \in \Delta(e) \cap \Delta(F)$ di Δ si ha $e, f \in E \setminus F$.

Esempio 3.1.16. Sia $G = (V, E)$ il grafo completo K_5 e sia
$F := \{\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,5\}\} \subseteq E$. La seguente è una *1-raffigurazione di G rispetto a F*:



Se invece $F' := \{\{1,2\}, \{2,3\}, \{3,4\}, \{4,5\}\{5,1\}\}$, la seguente è una *5-raffigurazione di G rispetto a F'*:



3.2 Algoritmo

Sia $k \geq 1$, l'obiettivo è descrivere un algoritmo che risolva il seguente *problema del k-numero di incroci generalizzato* in tempo quadratico.
Input: Grafo $G = (V, E)$ **e un sottoinsieme** $F \subseteq E$.
Output: Decidere se G **ammette una k-raffigurazione rispetto a** F.

In seguito verrà presentato un algoritmo che computa un k-raffigurazione di G, se ne esiste una.
Questo algoritmo opera in due fasi:
Passo1: riduce iterativamente le dimensioni del grafo in input finché non ottiene un grafo la cui larghezza è limitata da una costante che dipende solo da k.
Passo 2: risolve il problema in questo grafo di larghezza limitata.

3.2.1 Passo 1

Sia $r := 2k + 2$, si sceglie un numero s sufficientemente grande tale che per ogni grafo $G = (V, E)$ con numero di incroci al più k e ogni immersione topologica $h : H_s \to G$ esista una sottogriglia $H_r \subseteq H_s$ tale che la restrizione $h|_{H_r}$ di h su H_r sia piana. Un tale s esiste per il teorema (3.1.13). In seguito viene scelto w rispetto ad s in accordo con il teorema (3.1.14), cioè che esista un algoritmo in tempo lineare tale per cui, dato un grafo di larghezza almeno w, si trovi un'immersione topologica $h : H_s \to G$.
r, s e w sono fissati in seguito in questa sezione.

Lemma 3.2.1. *Dato un grafo G, esiste un algoritmo in tempo lineare che riconosce se il numero di incroci di G è maggiore di k, oppure riconosce se la larghezza di G è al più w, oppure computa un'immersione topologica piana $h : H_r \to G$.*

Dimostrazione. Applicando l'algoritmo del teorema (3.1.14), se l'algoritmo riconosce che la larghezza del grafo G è al più w, la dimostrazione è conclusa. Altrimenti computa un'immersione topologica $h : H_s \to G$. Dalla scelta di s, si ha che il numero di incroci di G è maggiore di k oppure esiste una sottogriglia $H_r \subseteq H_s$ tale che la restrizione di h su H_r sia piana.
Per ogni $H_r \subseteq H_s$ si può decidere se $h|_{H_r}$ è piana in tempo lineare. L'algoritmo riconosce se h_{H_r} è piana per ogni $H_r \subseteq H_s$. Quindi trova una h_{H_r} piana oppura trova che il numero di incroci di G è maggiore di k.
Dato che s è una costante fissata, il tempo di esecuzione è lineare. □

Sia $G = (V, E)$ un grafo e $h : H_r \to G$ un'immersione topologica piana. Per $2 \leq i \leq r$, sia H^i la sottogriglia di H_r limitata dall'i-esimo ciclo principale C_i. Sia K_i il sottografo di G costituito da $h(H^i)$ e da tutti i collegamenti di $h(H_r)$ che intersecano $h(H^i \setminus C_i)$. Inoltre, sia F_i l'insieme di tutti gli archi di K_i che hanno almeno un estremo in $h(C_i)$. Dato che h è piana, risulta che gli insiemi F_i sono disgiunti per $2 \leq i \leq r$.
Sia Δ una k-raffigurazione di G con numero di incroci minimo. Dato che $r = 2k + 2$ esiste almeno un $i, 2 \leq i \leq r$ tale che nessuno degli archi in F_i sia coinvolto in nessun incrocio di Δ. Sia i_0 il minimo i ad avere questa proprietà.
Sia $C := h(C_{i_0}), K := K_{i_0}$ e $I := K \setminus C$. Allora K e I sono entrambi grafi planari connessi.

Teorema 3.2.2. *La restrizione di Δ su K è una raffigurazione planare.*

Dimostrazione. Si supponga per assurdo che la restrizione di Δ su K non sia una raffigurazione planare. Sia Π una raffigurazione planare di K, allora $\Pi(C)$ è una curva chiusa semplice del piano e, senza perdere di generalità, si può assumere che $\Pi(I)$ sia interamente contenuto nell'interno di $\mathbb{R}^2 \setminus \Pi(C)$. Sia ora Δ' una raffigurazione di G che è identica a Δ su $G \setminus I$ e omeomorfa a Π su K. Dato che nessun arco in F_i è

coinvolto in nessun incrocio di Δ allora nessuno degli archi in F_i è coinvolto in nessun incrocio di Δ'. Segue che il numero di incroci di Δ' è minore del numero di incroci di Δ. Questo contraddice la minimalità del numero di incroci di Δ. □

Il precedente teorema implica in particolare che nessuno degli archi di F_2 sia coinvolto in nessun incrocio di Δ. Dalla minimalità di i_0 si ha quindi che $i_0 = 2$. Quindi i_0 è indipendente dalla raffigurazione Δ.
Sia $G' = (V', E')$ il grafo ottenuto partendo da G e contraendo il sottografo connesso I ad un singolo vertice v_I, come in figura.

Sia F' l'unione di F con tutti gli archi di $h(C)$ e con tutti gli archi incidenti nel vertice v_I. Allora G ha una k-raffigurazione rispetto a F se, e solo se, G' ha una k-raffigurazione rispetto a F'.
Dati G, F e h, il grafo G' e l'insieme di archi F' sono computabili in tempo lineare. Inoltre $|V'| < |V|$. Combinando questo fatto con il lemma (3.2.1) si ottiene il seguente.

Lemma 3.2.3. *Dato un grafo $G = (V, E)$, esiste un algoritmo in tempo lineare che riconosce se il numero di incroci di G è maggiore di k oppure riconosce se la larghezza di G è al più k oppure computa un grafo $G' = (V', E')$ e un insieme di archi $F' \subseteq E'$ con $|V'| < |V|$ tale che G ha una k-raffigurazione rispetto a F se, e solo se, G' ha una k-raffigurazione rispetto a F'.*

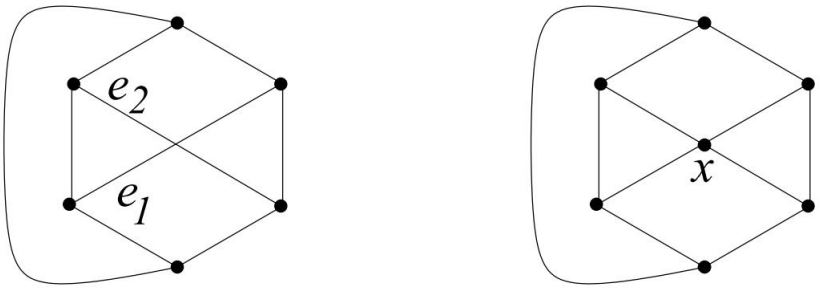
Reiterando l'algoritmo del precedente lemma si ottiene.

Corollario 3.2.4. *Dato un grafo $G = (V, E)$, esiste un algoritmo in tempo quadratico che riconosce se il numero di incroci di G è maggiore di k oppure computa un grafo $G' = (V', E')$ e un insieme di archi $F' \subseteq E'$ tale che la larghezza dell'albero di G' sia al più w e G ha una k-raffigurazione rispetto a F se, e solo se, G' ha una k-raffigurazione rispetto a F'.*

3.2.2 Passo 2

Se l'algoritmo non ha trovato che il grafo $G = (V, E)$ ha numero di incroci maggiore di k nel passo 1, allora ha prodotto un grafo $G' = (V', E')$ di larghezza al più k e un insieme $F' \subseteq E'$ tale che G ha una k-raffigurazione rispetto a F se, e solo se, G' ha una k-raffigurazione rispetto a F'. Nel passo 2 l'algoritmo deve decidere, usando il teorema di Courcelle, se G' ammette una k-raffigurazione rispetto a F.
Bisogna quindi trovare una formula MSO $\phi(X)$ tale che per ogni grafo $G = (V, E)$ ed ogni insieme $F \subseteq E$ si ha $G \models \phi(F)$ se, e solo se, G ammette una k-raffigurazione rispetto a F. È risaputo che esiste una formula MSO $\phi_{planare}$ che dichiara quando un

grafo è planare.
Sia $G = (V, E)$ un grafo e siano $e_1, e_2 \in E$ due archi distinti. Sia $G^{e_1 \times e_2}$ il grafo ottenuto partendo da G, rimuovendo gli archi e_1 ed e_2 e aggiungendo un nuovo vertice x e quattro archi che connettono x con gli estremi di e_1 ed e_2. Come in figura.



Si può osservare che per ogni $l \geq 1$ un grafo $G = (V, E)$ ammette una l'raffigurazione rispetto ad un insieme $F \subseteq E$ se, e solo se, esistono due archi distinti $e_1, e_2 \in E \setminus F$ tale che $G^{e_1 \times e_2}$ ammetta una (l-1)-raffigurazione rispetto a F.
Si ha il seguente lemma.

Lemma 3.2.5. *Per ogni formula MSO $\phi(Y)$ esiste una formula MSO $\phi^*(x_1, x_2, Y)$ tale che per ogni grafo $G = (V, E)$, insieme di archi $F \subseteq E$ e archi distinti $e_1, e_2 \in E \setminus F$ si ha*

$$G \models \phi^*(e_1, e_2, F) \Leftrightarrow G^{e_1 \times e_2} \models \phi(F). \tag{3.1}$$

Usando questo lemma, si possono definire induttivamente, per ogni $l \geq 1$, due formule $\phi_l(Y)$ e $\psi_l(x_1, x_2, Y)$ tali che per ogni grafo $G = (V, E)$ e insieme di archi $F \subseteq E$ si ha

$$G \models \phi_l(F) \Leftrightarrow G \textit{ ammette una } l - raffigurazione \textit{ rispetto a } F, \tag{3.2}$$

inoltre per ogni $e_1, e_2 \in E \setminus F$ si ha

$$G \models \psi_l(e_1, e_2, F) \Leftrightarrow G^{e_1 \times e_2} \textit{ ammette una } (l-1) - raffigurazione \textit{ rispetto a } F. \tag{3.3}$$

Sia

$$\psi_1(x_1, x_2, Y) := \phi^*_{planare}(x_1, x_2) \tag{3.4}$$

e per ogni $l \geq 1$ siano

$$\phi_l(Y) := \exists x_1 \exists x_2 (x_1 \neq x_2 \wedge \neg Y x_1 \wedge \neg Y x_2 \wedge \psi_l(x_1, x_2, Y)), \tag{3.5}$$

$$\psi_{l+1}(x_1, x_2, Y) := \phi^*_l(x_1, x_2, Y). \tag{3.6}$$

3.2.3 Computazione di una raffigurazione

Si può modificare il precedente algoritmo in modo che computi una raffigurazione del grafo G. Nel passo 1, se si ha una raffigurazione di G' rispetto a F' allora si può facilmente costruire una raffigurazione di G rispetto a F.
Nel passo 2 invece si può definire per induzione su $l \geq 0$ un algoritmo in tempo lineare $DRAW_l$ che, dato un grafo $G = (V, E)$ di larghezza dell'albero al più w e un sottoinsieme $F \subseteq G$, computa una l-raffigurazione di G rispetto a F, se ne esiste una. $DRAW_0$ computa una raffigurazione planare di G.
Per $l \geq 1$ si può applicare il teorema di Courcelle alla formula MSO

$$\chi_l(x_1, x_2, Y) := x_1 \neq x_2 \wedge \neg Y x_1 \wedge \neg Y x_1 \wedge \psi_l(x_1, x_2, Y). \tag{3.7}$$

Si ottiene quindi un algoritmo in tempo lineare che, dato un grafo G e $F \subseteq E$, computa due archi $e_1, e_2 \in E \setminus F$ tale che $G \models \chi_l(e_1, e_2, F)$. Segue dalla definizione di ψ_l che $G \models \chi_l(e_1, e_2, F)$ se, e solo se, $G^{e_1 \times e_2}$ ammette una l-raffigurazione rispetto a F.
Dati G e F, l'algoritmo $DRAW_l$ applica l'algoritmo precedente per computare $e_1, e_2 \in E \setminus F$ tale che $G \models \chi_l(e_1, e_2, F)$. Quindi si applica $DRAW_{l-1}$ al grafo $G^{e_1 \times e_2}$ per computare una (l-1)-raffigurazione del grafo $G^{e_1 \times e_2}$ rispetto a F. L'algoritmo modifica questa raffigurazione in modo da ottenere una l-raffigurazione di G rispetto a F.

Bibliografia

[1] Bruno Courcelle, Joost Engelfriet, Graph structure and monadic second-order logic. A language-theoretic approach, *Cambridge University Press, pp.728, 2012, Encyclopedia of Mathematics and its applications*.

[2] Joachim Kneis, Alexander Langer, A Practical Approach to Courcelle's Theorem, *Dept. of Computer Science, RWTH Aachen University, 2009*.

[3] Michael Sipser, Introduction to the Theory of Computation, *massachusetts institute of technology, 2012*.

[4] Alberto Facchini, Algebra e matematica discreta, *Zanichelli, 2000*.

[5] Oscar Levin, Descrete mathematics, an open introduction, *School of Mathematical Science, University of Northern Colorado, 2015*.

[6] Charu C. Aggarwal, Artificial Intelligence: A Textbook, *IBM T. J. Watson Research Center Yorktown Heights, New York, 2021*.

[7] Hans L. Bodlaender, a linear-time algorithm for finding tree-decompositions of small treewidth, *Department of computer science, Utrecht University, 1992*.

[8] Anantha Padmanabha MS, Advanced Graph Algorithms, Courcelle's Theorem, *Institute of Mathematical Sciences, Chennai, 2014*.

[9] Samuel Frederic Barr, Courcelle's Theor s Theorem: Overview and Applications, *Oberlin College, 2020*.

[10] Leonid Libkin, Elements of Finite Model Theory, *Springer, 2012*.

[11] Leonid Libkin, Towards A Formal Verification of Courcelle's Theorem, *Wesleyan University, 2017*.

[12] Martin Grohe, Computing Crossing Numbers in Quadratic Time, *University of Illinois at Chicago, 2018*.

[13] R.M.R. Lewis, Guide to Graph Colouring, *Texts in Computer Science, 2021*.

[14] C. Thomassen, A simpler proof of the excluded minor theorem for higher surfaces, *Journal of Combinatorial Theory, 1997*.

[15] N. Robertson, P.D. Seymour, Graph minors XIII. The disjoint paths problem, *Journal of Combinatorial Theory, 1995*.

[16] H.-D. Ebbinghaus, J. Flum, W. Thomas, Mathematical Logic, *Springer-Verlag, second edition, 1994*.