A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

Università degli Studi di Perugia
Dipartimento di Ingegneria

Corso di Laurea Triennale in
Ingegneria Informatica ed Elettronica

Tesi di Laurea

Sviluppo di un'applicazione web per la visualizzazione di grafi completi con ridotto numero di incroci

Laureando:
Davide Sonno

Relatore:
Prof. Luca Grilli

Anno Accademico 2022/2023

Indice

Introduzione

L'ubiquità del concetto di grafo è da ricercarsi nella sua definizione: un grafo è l'unione tra un gruppo di elementi, vertici o nodi, e le coppie formate dagli elementi stessi, ad indicarne delle relazioni.

Questa struttura così generica e reticolare fa si che sia possibile ricondurvi gran parte degli elementi di molti ambiti di ricerca, come la biologia, l'urbanistica, l'informatica e le scienze in generale.

Concentrarsi quindi sullo studio di un determinato grafo, ottenuto a partire dalla realtà d'interesse, può risultare fondamentale e per ciò è di ausilio una visualizzazione opportuna di quella specifica situazione.

Visualizzare un grafo significa rappresentarne tutti i vertici e le varie coppie tra di questi. Alle seconde, viene generalmente conferita la struttura di un collegamento tra i due vertici, il cosiddetto arco, il tutto viene poi posto su di una superficie: in generale un piano, ma potrebbe anche essere una sfera o un cilindro.

Se rappresentati con questa notazione, una caratteristica molto forte che questi possono avere è la cosiddetta planarità, ovvero l'assenza di incroci tra gli archi del grafo su visualizzazione planare. Tale proprietà comporta una più facile visualizzazione ed interpretazione del grafo.

In realtà la maggior parte dei grafi risulta essere non planare: con l'aumento del numero di archi velocemente si incappa in rappresentazioni non realizzabili bidimensionalmente senza incorrere in intersezioni.

Risulta quindi necessario trovare una visualizzazione che minimizzi il numero di incroci, per rendere il sistema più facilmente analizzabile e visualizzabile. Allo scopo vi è stato molto lavoro a partire da mezzo secolo fa, con buoni risultati ma per i quali non si è ancora riusciti a dimostrarne l'universalità.

L'operazione che consiste nel ricercare la configurazione migliore per rappresentare un determinato grafo è però "NP-difficile", ovvero non se ne possono confrontare tutte le possibili realizzazioni per poi scegliere la migliore, ma ci si può solo limitare a trovarne l'approssimazione migliore.

In particolare in questo progetto di Tesi si considereranno i cosiddetti grafi completi, nei quali tutti gli elementi sono in relazione con tutti gli altri elementi. La rappresentazione di tali grafi può risultare poco comprensibile, se non addirittura caotica, per questo verranno proposte ed implementate particolari rappresentazioni che si congetturano essere ottime per questo problema, il tutto tramite l'applicazione web sviluppata.

Al fine di argomentare con accuratezza il contenuto del tema sopra citato, si è scelto di suddividere questa Tesi in diversi capitoli, così organizzati:

Capitolo 1 - ***Tecnologie e concetti preliminari*** Verranno descritte le tecnologie ed i concetti indispensabili per la realizzazione del software.

Capitolo 2 - ***Motivazione e Ispirazione*** Verranno qui esposti i fattori che hanno stimolato l'interesse per questo progetto di Tesi.

Capitolo 3 - ***Il problema affrontato*** Verranno qui descritti compiutamente quali sono gli obiettivi realizzativi dell'applicazione web.

Capitolo 4 - ***Sviluppo dell'applicazione*** Verrà qui descritto come è stato prodotto il programma dell'applicazione web, quali criticità si sono presentate e come sono state risolte, i suoi punti di forza e gli aspetti che sono stati implementati.

Capitolo 5 - ***Dimostrazione di uso per il fine proposto*** Verranno proposti degli esempi del funzionamento dell'applicazione web volti al dimostrarne la correttezza, l'utilità e gli eventuali limiti.

Conclusioni e sviluppi futuri - Nell'ultimo capitolo si trarranno le Conclusioni, seguite da alcune proposte per un possibile sviluppo futuro.

Capitolo 1

Tecnologie e concetti preliminari

Segue all'introduzione una fase di preambolo dedicata alle tecnologie impiegate per lo sviluppo dell'applicazione ed i concetti sui quali si basa il funzionamento della stessa, così da poter affrontare al meglio gli argomenti trattati in seguito, senza soffermarsi sugli aspetti tecnici.

1.1 Tecnologie

1.1.1 HyperText Markup Language (HTML)

HTML [1] è un linguaggio di markup sviluppato nel periodo immediatamente successivo alla nascita del web: gli anni novanta del XX secolo da Tim Berners-Lee [2] al CERN di Ginevra, assieme al protocollo HTTP dedicato al trasferimento di documenti in tale formato. HTML resta, ancora oggi, nelle sue versioni aggiornate, tra i più utilizzati per la strutturazione di interfacce grafiche.

Il markup è una sintassi volta alla marcatura degli elementi che compaiono in una pagina web, con cui si intende la descrizione delle modalità di impaginazione e visualizzazione degli stessi, esplicitando le relazioni tra di loro e/o con il layout. Sup-

porta l'inserimento di contenuti testuali e allo stesso tempo di oggetti multimediali quali immagini e filmati.

Non ricorrendo a variabili e proprie strutture dati, HTML è classificato come un linguaggio dichiarativo e non come linguaggio di programmazione, ma negli ultimi decenni la sua definizione, in parallelo all'esagerato numero di nuove esigenze legate al moderno uso del web, ha indotto il World Wide Web Consortium (W3C), una associazione non governativa che si occupa di implementare nuove funzioni per rendere il web sempre più libero e accessibile ed i suoi membri, ad apportare nuove modifiche che lo dotino di un carattere "applicativo".

Ne consegue la scelta di incapsulare dei propri plugin tra i supporti, che colloca HTML sul podio degli strumenti più richiesti per la strutturazione del contenuto e lo sviluppo di vere e proprie applicazioni. Come risultato si ottiene un linguaggio moderno e flessibile, finalizzato ad una revisione continua delle proprie regole di impaginazione grafica, all'inserimento di nuovi strumenti di struttura e all'assorbimento di nuove tecnologie di supporto web.

Per gli scopi di questo lavoro di Tesi si è utilizzato HTML per creare la struttura portante della applicazione web, si è poi reso necessario l'utilizzo di altri linguaggi, di seguito descritti, che consentono di scrivere pagine ricche e complete sotto ogni punto di vista, per strutturare compiutamente l'interfaccia.

Nel particolare verranno impiegati i seguenti elementi HTML:

- **canvas**: il canvas è un elemento HTML5 rappresentato dal tag <canvas>. Può essere inteso come il corrispettivo digitale di una tela trasparente: uno spazio all'interno di una pagina web sul quale insistere con specifiche API adatte a tracciare linee, cerchi, rettangoli, immagini e altro ancora;

- **div**: il tag è il più generico contenitore messo a disposizione da HTML. L'uso generico è di fungere da divisore tra le varie parti della pagina, ma può diventare un elemento a se stante se opportunamente stilizzato;

- **button**: come intuibile dal nome l'elemento button consiste in un vero e proprio pulsante cliccabile, dal quale possono scaturire azioni o comportamenti;

- **input**: gli elementi input sono numerosi e ricoprono il compito fondamentale di rendere interattiva la pagina web. Nel caso di questo progetto saranno impiegati come casella di testo, slider e casella di verifica, rispettivamente *number*, *range* e *checkbox*.

1.1.2 JavaScript (JS)

Javascript [3] è un linguaggio di programmazione orientato agli oggetti che gestisce la dinamicità di una pagina e rende possibile l'interazione tra server e client.

Occupa una posizione fondamentale nello sviluppo di applicativi web e ha la peculiarità di essere un linguaggio interpretato, leggero con funzioni di prima classe. È un linguaggio per cui è prevista la diretta esecuzione del codice e non è necessario ricorrere all'uso di un compilatore. JavaScript è un linguaggio dinamico multiparadigma con tipi e operatori, oggetti incorporati standard e metodi, inoltre vi è una scarsa tipizzazione delle variabili. La sua sintassi è basata sui linguaggi Java e C: molte strutture di questi linguaggi si applicano anche a JavaScript.

Le capacità dinamiche di JavaScript includono la costruzione di oggetti a tempo di esecuzione, elenchi di parametri variabili, variabili di funzione, creazione dinamica di script, introspezione di oggetti (tramite le utility *for...in* e *Object*) e recupero del codice sorgente (le funzioni JavaScript memorizzano il loro testo sorgente e possono essere recuperate tramite *toString()*).

Per potenziare la qualità della navigazione all'interno di una pagina, spesso le librerie vengono utilizzate in concomitanza con le *API*s, Application Programming Interfaces, dei costrutti che sintetizzano in poche righe di codice delle funzioni altrimenti complicate, che lo sviluppatore può usare per permettere lo scambio di

informazioni tra componenti di uno stesso browser o tra un browser e servizi di terzi.

1.1.3 Cascading Style Sheets (CSS)

Mentre HTML si preoccupa della struttura di una pagina Web, CSS [4] rivolge la propria attenzione al suo aspetto grafico. Tra le operazioni di base che permette di effettuare troviamo la formattazione del testo, la modifica delle dimensioni di un oggetto e la gestione delle distanze tra gli elementi della pagina.

Possiamo guardare a CSS come un linguaggio con una finalità complementare rispetto a quella di HTML, dove la pagina realizzata dal linguaggio di markup rappresenta uno scheletro che deve essere reso tramite CSS graficamente godibile per l'utente. Nonostante si posa usare CSS direttamente all'interno del file HTML, si può anche utilizzarlo tramite file esterni da importare, per permettere la gestione separata di struttura e stile.

1.1.4 GeoGebra

GeoGebra [5] è un software interattivo open source liberamente scaricabile per la matematica dinamica che comprende geometria, algebra e analisi. Ideato e sviluppato da Markus Hohenwarter nel 2001/2002 come parte della sua Tesi in Educazione alla Matematica e Informatica, presso l'Università di Salisburgo, GeoGebra offre principalmente due funzionalità:

- sul versante geometrico adotta l'approccio tipico dei programmi di geometria dinamica, consentendo la costruzione dei vari oggetti geometrici (punti, vettori, segmenti, rette, coniche, ma anche funzioni) e la possibilità di modificarli dinamicamente;

- sul versante algebrico e analitico GeoGebra consente di inserire direttamente da tastiera equazioni e coordinate, con un approccio molto simile ai programmi di tipo CAS (Computer Algebra System). Grazie a questa possibilità, GeoGebra consente di: trattare variabili numeriche, vettori e punti; calcolare derivate e integrali di funzioni; determinare il valore di una radice o un punto di estremo per una funzione assegnata.

Alle due funzionalità principali corrispondono due diverse visualizzazioni. Lo schermo di GeoGebra è suddiviso in due finestre con reciproco rimando: un'espressione nella finestra algebra corrisponde a un oggetto nella finestra geometria e viceversa. Ha come funzione di base la possibilità di costruire e gestire oggetti geometrici in modo interattivo, permettendo non solo di disegnare le figure ma anche di "manipolarle", sottoponendole mediante il mouse alle varie possibilità di trascinamento che il programma mette a disposizione.

Tale strumento sarà impiegato per la rappresentazione di simulazioni, necessarie alla realizzazione del progetto.

1.2 Concetti preliminari

Per la realizzazione di questo progetto di Tesi è stato fatto ampio ricorso al concetto di Grafo, di Geometria analitica ed a concetti di Algebra lineare; di seguito vengono descritti gli elementi principali utilizzati.

1.2.1 Grafi

In matematica un grafo è la configurazione formata da un insieme di punti (vertici o nodi) e un insieme di coppie di nodi: $G = (V, E)$.

Con questa notazione, V rappresenta l'insieme dei vertici e la sua dimensione sarà $|V| = n$.

L'insieme di queste coppie è E e la sua dimensione $|E| = m$. Essendo questo insieme formato da coppie di elementi contenuti in E, tutti gli elementi di E appartengono al prodotto cartesiano $V \times V$.

Una notazione usata per esprimere un grafo, non tramite immagini, prende il nome di matrice di adiacenza, una matrice $n \times n$ con valori 1 se l'arco esiste, 0 altrimenti.

È comodo rappresentare uno specifico grafo mediante un diagramma, nel quale i vertici sono punti in un piano, ed ogni arco è rappresentato da una linea continua (non necessariamente un segmento di retta) congiungente i due vertici che corrispondono agli estremi dell'arco.

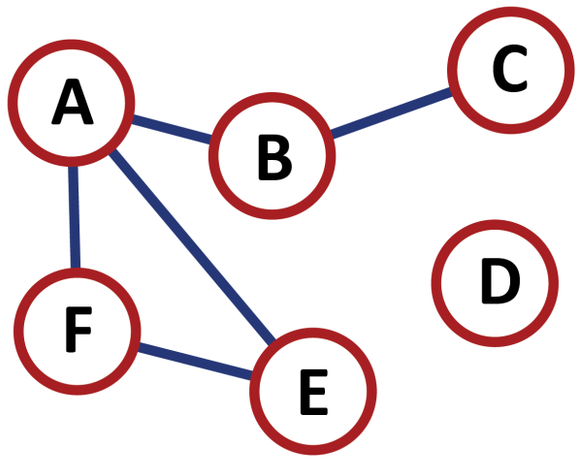

Figura 1.1: Esempio di rappresentazione di un grafo.
Immagine ispirata da Wikipedia

Nella teoria dei grafi un grafo completo è un grafo semplice (di grandezza finita e senza archi con entrambi gli estremi incidenti su uno stesso vertice) nel quale ogni vertice è collegato direttamente a tutti gli altri vertici. I grafi completi con n vertici sono tutti isomorfi.

Per grafo completo bipartito si intende un tipo speciale di grafo bipartito, ovvero composto di elementi suddivisi in due gruppi (partizioni), in cui ogni vertice del primo insieme è connesso ad ogni vertice del secondo insieme [6][7].

Nella teoria dei grafi, il numero minimo di incroci $cr(G)$ di un grafo G è il numero più basso di incroci di archi di un disegno piano del grafo G e prende il nome di *crossing number*. Ad esempio, un grafo è planare se e solo se il suo numero di incroci è zero. Nel grafo completo di n vertici, il minor numero di intersezioni dei suoi archi si denota con $Z(n)$.

La determinazione del numero di incroci continua a rivestire una grande importanza sia teorica che pratica nel disegno dei grafi [8][9].

1.2.2 Coordinate polari

Il sistema di coordinate polari è un sistema di coordinate bidimensionale nel quale ogni punto del piano è identificato da un angolo e da una distanza dall'origine. Il sistema di coordinate polari permette di esprimere le relazioni tra due punti in termini di angolo e di distanza.

In particolare, esprimere un punto in coordinate polari corrisponde al separarne le due componenti orizzontale e verticale, così da ottenere le coordinate nel piano cartesiano. La notazione di un punto in coordinate polari è $z = \rho(cos\theta + isin\theta)$ che corrisponde, in notazione esponenziale, a $z = \rho e^{i\theta}$. Le coordinate del punto risultante saranno date da:

$$\begin{cases} x = \rho \cdot cos(\theta) \\ y = \rho \cdot sin(\theta) \end{cases}$$

Tale notazione è stata impiegata nello sviluppo della applicazione web per posizionare i cerchi rispetto al centro con modulo e fase desiderati a partire dal centro del grafo.

1.2.3 Regola di Cramer

La regola di Cramer, o metodo di Cramer, è un teorema di algebra lineare, che prende il nome dal matematico Gabriel Cramer. Essa permette di calcolare la soluzione di un sistema lineare quando la matrice dei coefficienti A è una matrice quadrata e il suo determinante, $\det(A)$, è diverso da zero.

In particolare consideriamo un sistema di due equazioni e due incognite, nella forma

$$\begin{cases} xa_1 + yb_1 = c_1 \\ xa_2 + yb_2 = c_2 \end{cases}$$

Allora le soluzioni di tale sistema saranno date da:

$$det\ A = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix},\ det\ A_x = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix},\ det\ A_y = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}$$

$$x = \frac{det\ A_x}{det\ A},\ y = \frac{det\ A_y}{det\ A}$$

Questa regola è stata utilizzata nella scrittura del codice della applicazione web sviluppata per determinare il punto di intersezione dei segmenti.

Capitolo 2

Motivazione e Ispirazione

La nozione di grafo si presta ai più svariati utilizzi, costituendo un modo comodo e compatto per descrivere relazioni binarie in senso lato. Infatti, la rappresentazione di un grafo, con le sue diverse varianti, non solo fornisce uno schema visivo più o meno suggestivo, ma costituisce il naturale strumento matematico per rendere accessibile lo studio di problemi di svariata natura, da quelli sociologici a quelli ingegneristici.

Lo studio dei grafi tramite una loro visualizzazione risulta particolarmente complicato, in quanto la complessità delle rappresentazioni ed il numero di queste tendono ad aumentare esponenzialmente con la dimensioni stesse del grafo.

Data l'importanza assunta dai grafi, il problema degli incroci è un argomento di studio molto affrontato nell'ultimo secolo. Nonostante ciò non si è ancora riusciti a trovare delle soluzioni rigorose e valide per grafi di ogni dimensione. L'argomentazione più valida al momento è la congettura introdotta da Harary e Hill, dettagliata in seguito, che consta di grande approvazione ma che non si è riuscita a dimostrare, data la natura complessa del problema.

L'argomento fu inizialmente portato alla luce da Pàl Turán [10], quando, costretto a lavorare in una fabbrica di mattoni, si mise a riflettere su come trovare una soluzione ad un problema che di frequente si poneva trasportando carrelli carichi di

mattoni: il *Problema della Fabbrica di Mattoni di Turán* [11].

Questo consisteva nel percorso di binari sui quali scorrevano carrelli colmi di mattoni che dovevano essere trasportati dai vari forni ai rispettivi magazzini. Ogni qualvolta i binari si incrociavano i carrelli rischiavano di cadere, andando a creare un notevole disagio.

Turán si pose quindi il problema di come trovare configurazioni, e quindi percorsi per i carrelli, ottimali tali da minimizzare il numero di incroci dei binari.

Una versione più moderna del problema potrebbe essere introdotta nel campo dell'elettronica, in particolare nei circuiti elettronici stampati, nel quale vi sono un elevato numero di componenti, con i relativi collegamenti. Tali collegamenti necessariamente dovranno incrociarsi, ma ciò non può avvenire se non su livelli diversi, in quanto se si dovessero toccare vi sarebbero malfunzionamenti. La complessità della scheda è direttamente proporzionale al numero di livelli che si devono dunque realizzare. Lo studio attento del numero di incroci permette quindi di ottimizzare i percorsi delle piste nei circuiti elettronici stampati con evidente risparmio in materie prime, spazio e costi.

Un esempio più su larga scala potrebbe essere la rete stradale, urbana e non, nella quale ogni incrocio può essere gestito tramite intersezioni ma a scapito di sicurezza, o tramite cavalcavia/sottopassi ma a scapito di risorse. Per trovare il bilancio ottimale fra queste due soluzioni i grafi sono lo strumento di elezione.

Figura 2.1: Raccordo stradale su più livelli.
(Source:
https://www.infobuild.it/wp-content/uploads/Strade.jpg)

Questo lavoro di Tesi verte sull'implementare configurazioni di grafi completi e completi bipartiti, proposte da studiosi che hanno affrontato il tema in passato, su uno spazio bidimensionale con lo scopo di poter fornire uno strumento utile allo studio del problema degli incroci, nel suddetto caso particolare.

Tali configurazioni sono risultate ottime per grafi di ridotte dimensioni e si congettura lo siano per valori superiori. Nonostante gli sforzi profusi dalla comunità scientifica, non si è ancora proposto un approccio promettente alla dimostrazione della congettura sul numero minimo di incroci, problematica che potrebbe avvalersi dello strumento sviluppato in questo progetto di Tesi. Con le funzionalità dell'applicazione sarà possibile ricercare strutture particolari, andando ad esempio ad isolare dei vertici su azione dell'utente.

Capitolo 3

Il problema affrontato

L'obiettivo che si impone questo progetto è la realizzazione di un'applicazione web che possa generare in maniera attendibile grafi completi di dimensioni arbitrarie sulla base di visualizzazioni proposte in passato e congetturate essere ottime. Le visualizzazioni dovranno quindi rispettare dei criteri ben posti e garantire che il numero di intersezioni corrisponda a quello congetturato.

Sarà possibile interagire su di essi così da poterne studiare meglio le caratteristiche, nell'insieme o nel particolare. Inoltre sarà possibile isolare parte dei vertici per poterne analizzare nel dettaglio il loro comportamento e le loro intersezioni, così da facilitare lo studio di questo argomento.

3.1 Il problema degli incroci minimi

Come descritto precedentemente, non vi è una completa conoscenza della legge che regola quale sia il minor numero di incroci che si possano ottenere per un grado di una determinata dimensione. Tanto meno si è a conoscenza della rappresentazione perfetta per dimensioni sopra una certa soglia.

Il progetto è stato realizzato senza l'uso di librerie grafiche, così da poterlo rendere più leggero ed efficiente possibile. Questo anche a ragione del fatto che alcune delle operazioni eseguite dal programma sono computazionalmente dispendiose.

La seguente applicazione, dal punto di vista grafico, non ha come scopo quello di ricercare la rappresentazione ottima, quindi con minimo numero di incroci, per un grafo di una data dimensione.

3.2 Elenco dei requisiti

L'intero progetto verrà sviluppato all'interno di una pagina HTML, alla quale si appoggia uno script realizzato in JavaScript ed il tutto gestito graficamente con un file CSS. L'applicazione non disporrà di una parte di back-end, non essendoci la necessità di comunicare con API, database o altro. Il sistema è interamente funzionante localmente su browser e non è necessaria la predisposizione di eventuali configurazioni software.

Il sistema così descritto necessiterà, almeno, delle seguenti caratteristiche:

- Fornire la dimensione di un grafo completo.
- Fornire la dimensione della prima partizione di un grafo completo bipartito.
- Fornire la dimensione della seconda partizione di un grafo completo bipartito.
- Visualizzazione di grafi completi secondo la rappresentazione di Harary-Hill.
- Visualizzazione di grafi completi secondo la rappresentazione di Blažek-Koman.
- Visualizzazione di grafi completi bipartiti secondo la rappresentazione di Turán.
- Visualizzazione delle intersezioni tra gli archi nelle varie rappresentazioni.

- Possibilità di isolare parte dei vertici, evidenziando loro ed i collegamenti tra di essi, con le relative intersezioni.
- Possibilità di ridimensionare i grafi.
- Possibilità di ruotare i grafi.
- possibilità di attivare o disattivare la visualizzazione delle intersezioni.

Capitolo 4

Sviluppo dell'applicazione

In questo capitolo si andranno ad affrontare i vari aspetti che hanno portato allo sviluppo dell'applicazione, sia per quanto ne riguarda l'aspetto tecnologico ma anche relativamente alla geometria che è stata utilizzata.

L'argomento principale che si è affrontato è stato quello di implementare la rappresentazione dei grafi tramite tre visualizzazioni, la cui costruzione è nota. Tali rappresentazioni, due per grafi completi ed una per grafi completi bipartiti, forniscono un disegno congetturato essere ottimo.

In seguito, l'attenzione è stata rivolta nei confronti delle intersezioni tra i vari archi del grafo, affrontato in modo diverso sulla base degli elementi da intersecare. Questo aspetto è di natura prettamente algebrica e matematica, con delle accortezze relativamente al sistema di coordinate usato dalle pagine web e alla gestione delle variabili da parte di JavaScript.

Ultimo aspetto principale sarà l'interazione che l'utente avrà nei confronti dei grafi creati, potendone variare la dimensione, la rotazione e potendone selezionare alcuni vertici per osservarne nel dettaglio le caratteristiche.

4.1 Le diverse visualizzazioni

Vi sono varie metodologie per rappresentare un grafo completo con i suoi incroci, in questo lavoro di Tesi ne sono state utilizzate alcune, di seguito descritte.

In generale, rappresentando un grafo, si cerca di evitare che più archi si incrocino in uno stesso punto. Nel caso di questa applicazione la situazione citata può però accadere, a causa dell'algoritmo impiegato per la generazione, ma ciò non comporta errori nel calcolo delle intersezioni.

4.1.1 Harary-Hill

Alla fine degli anni 50, Anthony Hill [12] intraprese degli studi per trovare un metodo per poter disegnare un grafo completo con il numero minimo di intersezioni. Assieme a Frank Harary [13] propose un disegno con questa costruzione:

- preso un cilindro, si pongono metà dei punti sulla circonferenza di base e metà dei punti in quella di testa;
- i vertici dei singoli gruppi vengono collegati tra di loro con linee dritte;
- si collegano i vertici dei due gruppi lungo la superficie del cilindro con linee dritte.

Per poter disegnare su di un piano il grafo, è necessario porre le due circonferenze una dentro l'altra. I collegamenti nello spazio tra i due cerchi corrispondono a quelli punti sulla superficie del cilindro, ma non saranno più segmenti bensì archi o spirali. Il cerchio interno avrà linee dritte per collegarne i vertici. Per quanto riguarda la circonferenza esterna è necessario l'uso di curve opportune con incroci riconducibili a quelli che si avrebbero se i punti fossero sulla circonferenza di base del cilindro.

In caso di dimensione dispari, si possono porre i due gruppi di punti arbitrariamente, in quanto sarebbe come ribaltare il cilindro di partenza.

I due studiosi svilupparono inoltre una congettura che stabilisce un numero minimo di incroci $Z(n)$, che ogni gravo dovrà necessariamente avere. Tale limite inferiore si calcola come segue, in funzione della dimensione n del grafo:

$$Z(n) \coloneqq \tfrac{1}{4}\lfloor\tfrac{n}{2}\rfloor \ \lfloor\tfrac{n-1}{2}\rfloor \ \lfloor\tfrac{n-2}{2}\rfloor \ \lfloor\tfrac{n-3}{2}\rfloor$$

Non si è ancora a conoscenza di rappresentazioni di grafi con un numero di incroci che sia minore di quanto stabilito dalla congettura di cui sopra. Si suppone inoltre che questa rappresentazione dia luogo a disegni il cui numero delle intersezioni coincide con il *crossing number*, il minor numero possibile di incroci.

Vi sono, in compenso, alcune rappresentazioni in cui il numero di incroci coincide con quelli dettati Harary e Hill, come quella descritta successivamente.

Data la natura esponenziale del problema, è stato possibile dimostrarne la correttezza, tramite forza bruta, solo sino ad una dimensione dei grafi pari a 12 [14].

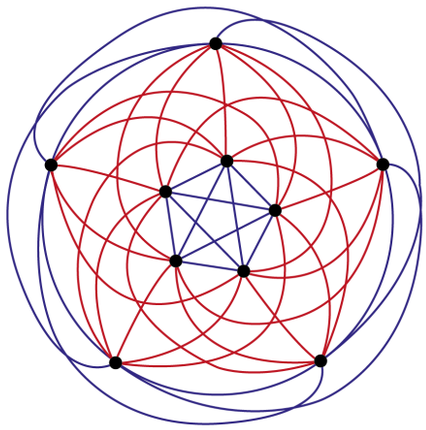

Figura 4.1: Rappresentazione di grafo Completo secondo Harary e Hill.
Immagine ispirata da SpringerLink

4.1.2 Blažek-Koman

Durante lo stesso periodo nel quale Harary e Hill studiarono l'argomento, anche altri studiosi diedero il loro contributo alla rappresentazione di un grafo completo con numero minimo di incroci. Tra di questi vi furono Blažek e Koman, i quali pubblicarono un articolo proponendo una loro rappresentazione alternativa [15].

In particolare tale rappresentazione viene realizzata come di seguito:

- si dispongono tutti i vertici del grafo su di una circonferenza;
- si collegano i vertici dove il segmento che li collega internamente ha una pendenza positiva;
- tutti i restanti collegamenti vengono effettuati esternamente, sempre in modo coerente al minimizzarne gli incroci.

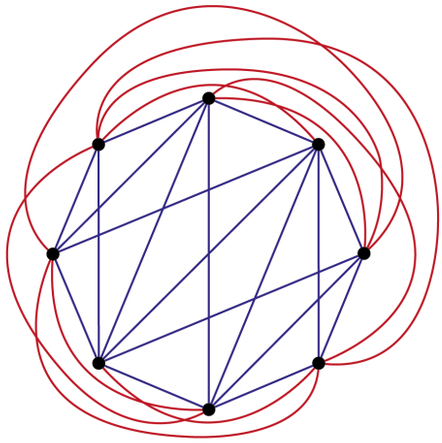

Figura 4.2: Rappresentazione di grafo Completo secondo Blažek e Koman. Immagine ispirata da SpringerLink

.

Come introdotto prima, anche questa rappresentazione è postulata essere ottima, producendo quindi disegni con numero di intersezioni uguale a quello derivante dalla congettura di Harary-Hill.

4.1.3 Grafo Bipartito

La realizzazione che verrà implementata per il caso di grafi completi bipartiti corrisponde alla soluzione proposta da Turán per il problema della fabbrica di mattoni.

Essa consiste nel porre tutti i punti della prima partizione orizzontalmente e porre gli elementi della seconda verticalmente a formare una croce. A questo punto sarà sufficiente collegare i vertici in modo tale che ogni elemento di una partizione raggiunga tutti gli elementi dell'altra partizione.

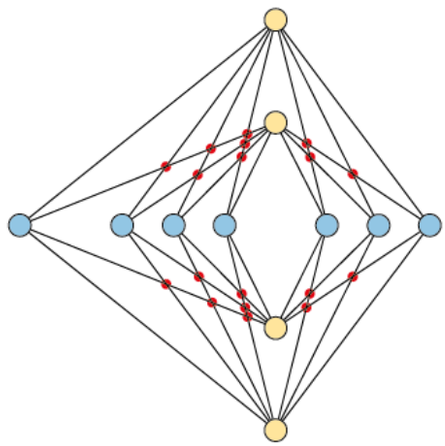

Figura 4.3: Rappresentazione di grafo Completo Bipartito. Si noti lo spazio vuoto al centro della croce.
Immagine ispirata da Wikipedia
.

In questa situazione di grafo completo bipartito, cercando di dimostrare la correttezza degli studi di Turán, lo studioso Kazimierz Zarankiewicz [16] fornì una formula che descrive un margine superiore al numero di incroci:

$$Z(n,m) \coloneqq \lfloor \tfrac{n}{2} \rfloor \lfloor \tfrac{n-1}{2} \rfloor \lfloor \tfrac{m}{2} \rfloor \lfloor \tfrac{m-1}{2} \rfloor$$

Nonostante non sia ancora stato possibile dimostrare la correttezza di questa formula, si congettura che sia proprio il numero minimo di incroci per un grafo completo bipartito e quindi il suo *crossing number*.

4.2 Struttura della pagina web

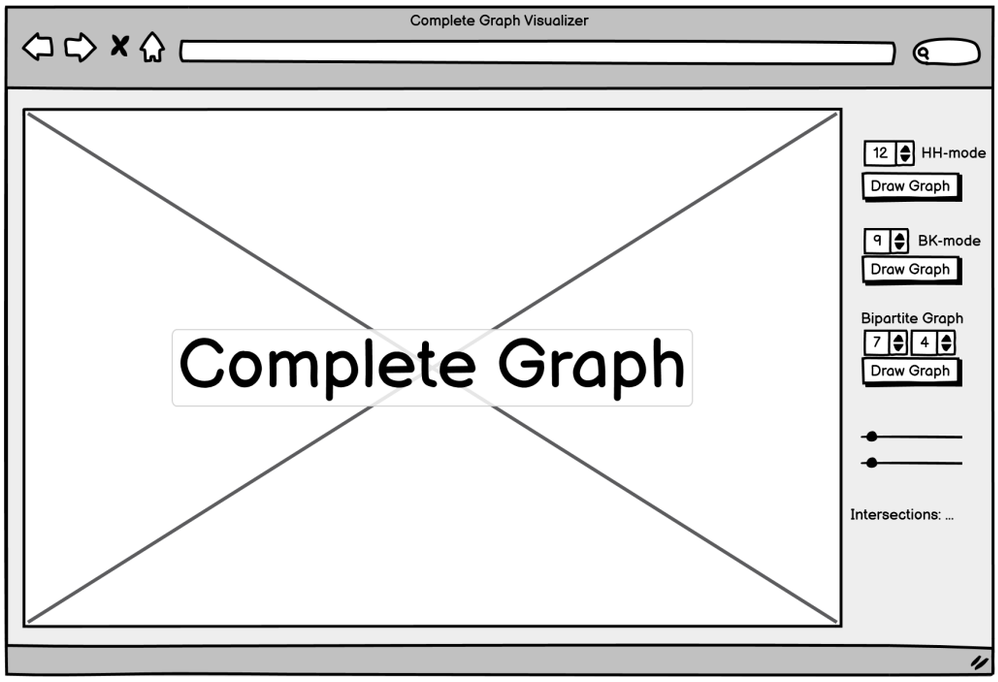

Figura 4.4: Elementi della Web App.

Gli elementi che compongono l'applicazione sono esplicitati nel file HTML "index.html". L'interezza della schermata è composta da un *div*, suddiviso a sua volta in altri due. Facendo riferimento alla figura 4.4, l'area riservata al disegno dei grafi è quella etichettata con *Complete Graph*, è costituita da uno dei due div principali, dentro il quale è posizionato un *canvas*.

Tutte le figure che appaiono all'interno del canvas lo fanno tramite metodi specifici di questo e sono nel dettaglio *arc([...])* per i cerchi assieme alla combinazione tra *moveTo(x,y)* e *lineTo(x,y)* per la creazione di segmenti. Di seguito il codice che li implementa:

```
function drawDirectLine(a, b, color, thickness, canvas) {
  canvas.lineWidth = thickness;
  canvas.strokeStyle = color;
  canvas.beginPath();
  canvas.moveTo(a.x, a.y);
  canvas.lineTo(b.x, b.y);
  canvas.stroke();
}
```

Figura 4.5: Funzione che consente di tracciare una linea retta tra i due punti a e b.

```
function drawCircle(x, y, radius, fill, stroke, thickness, canvas) {
  if (canvas.getContext) {
    ctx.beginPath()
    ctx.arc(x, y, radius, 0, 2 * Math.PI, false)
    if (fill) {
      ctx.fillStyle = fill
      ctx.fill()
    }
    if (stroke) {
      ctx.lineWidth = thickness
      ctx.strokeStyle = stroke
      ctx.stroke()
    }
  }
}
```

Figura 4.6: Funzione che consente di tracciare una circonferenza. Sono presenti anche condizioni per definirne il colore ed il riempimento.

La seconda parte della pagina si compone di tutti gli elementi necessari alla gestione degli input e degli output. In particolare, vi sono più campi di *input*, di

tipo numerico e non negativo, per poter inserire le dimensioni desiderate per il grafo e vi sono i pulsanti tramite i quali è possibile iniziare la creazione dei grafi.

Inoltre, sono posti qua degli *slider*, ovvero elementi input di HTML e di tipo *range* per gestire la dimensione del grafo e la sua rotazione. Un ultimo input, questa volta di tipo *checkbox*, è posto per poter disattivare a piacimento la visualizzazione delle intersezioni.

Vi sono infine degli elementi di testo per fornire informazioni riguardanti gli elementi e le intersezioni del grafo.

Va tenuto presente che il sistema cartesiano dei pixel della pagine non è tradizionale, infatti l'origine è posta sull'angolo in alto a sinistra della pagina, l'asse x è orientato verso destra mentre l'asse y verso il basso. Per il disegno dei grafi si andrà a calcolare le dimensioni del canvas, che variano in funzione della finestra del motore di ricerca, e con queste si individua il centro relativo del grafo.

4.3 Costruzione del grafo

4.3.1 Vertici

I vertici sono i primi elementi del canvas ad essere disegnati, tramite il metodo introdotto prima.

Il calcolo della posizione dei vertici si divide tra il caso di grafo bipartito e non. Per il grafo bipartito si suddividono l'altezza e la larghezza del canvas in tante parti tante quante il numero di vertici più uno, per lo spazio vuoto centrale. Sapendo quindi la distanza che intercorre tra i vari vertici di una partizione e le coordinate del centro della schermata, è possibile disegnare tutti i vertici.

Per i grafi non bipartiti occorre disporre i vertici circolarmente. Tramite una rappresentazione polare dello spazio è possibile generare i punti a partire dal centro

ed il raggio del grafo, dalla numerazione del vertice da disegnare e dall'angolo al quale porlo, quest'ultimo calcolato sulla base del numero di vertici da disegnare.

```
static generatePoint(cX, cY, r, number, delta, a0) {
    let angle = toRadians(number * delta + a0);
    let x = cX + r * Math.cos(angle);
    let y = cY - r * Math.sin(angle);
    return new Point(x, y);
}
```

Figura 4.7: Metodo per generare radialmente un vertice.

Questo metodo viene richiamato tante volte quante il numero di vertici da porre a quella distanza. Nel caso della visualizzazione di Harary-Hill, viene eseguito per entrambe le circonferenze, fornendo due raggi diversi: il cerchio interno è un terzo, in diametro, del cerchio esterno.

4.3.2 Archi lineari

Parte degli archi che compongono il grafo possono collegare i vertici sui quali incidono in modo diretto senza dover effettuare percorsi curvilinei.

In particolare ciò può essere eseguito su tutti gli archi della visualizzazione bipartita e sui segmenti che collegano internamente i vertici posti sulla circonferenza interna di Harary-Hill e sulla circonferenza di Blažek-Koman.

Per disegnarli è sufficiente usare la funzione drawDirectLine, fornendo gli opportuni vertici.

4.3.3 Scelta delle curve esterne

Le curve esterne sono il tema più delicato di questo progetto, in quanto trovare un buon sistema che sia indipendente dalla dimensione del grafo e che non abbia incroci doppi non è particolarmente facile.

Un primo approccio al problema impiegava delle curve di Bezièr [17], un particolare tipo di spline. Il problema sorto verteva sulla necessità di effettuare ridimensionamenti dipendenti dalla posizione dei vertici sul grafo e dal numero totale di vertici. Sarebbe quindi stata troppo complicata la ricerca di una funzione tale da poter ricevere i tre parametri in ingresso ed effettuare un calcolo idoneo, sebbene graficamente avrebbe potuto produrre risultati corretti.

Avendo scartato questa opzione un altro approccio abbastanza semplice è stato l'uso di circonferenze il cui centro viene posto nel punto medio del segmento che congiunge i due vertici. Con tale rappresentazione la figura era caratterizzata da spiccata simmetria, aspetto rilevante nel contesto di interesse in quanto garantisce una equa distribuzione degli archi con conseguente minor complessità.

La limitazione di questo approccio consisteva in vertici relativamente vicini connessi tramite un arco di circonferenza che tendeva ad essere molto esterno, mentre vertici più lontani venivano collegati con forme più schiacciate. Ma ciò costituisce un problema: se due vertici sono vicini è importante che l'arco congiungente rimanga più vicino possibile alla circonferenza sulla quale sono posizionati i vertici, in modo tale che curve che ne collegano di più lontani possano passare al di sopra di più archi possibile.

Detto ciò, la soluzione implementata in via definitiva consiste in un riadattamento dell'ultima metodologia descritta. Invece di porre il centro del cerchio a metà tra i due vertici, esso viene spostato radialmente rispetto al centro del grafo. Viene posto ad una distanza dal centro pari alla differenza che intercorre tra la dimensione del raggio e la distanza che il punto medio ha con il centro. Nel caso di punti diametralmente opposti, il centro del centro che comporrà l'arco esterno viene posto proprio sulla circonferenza dove sono posti i vertici.

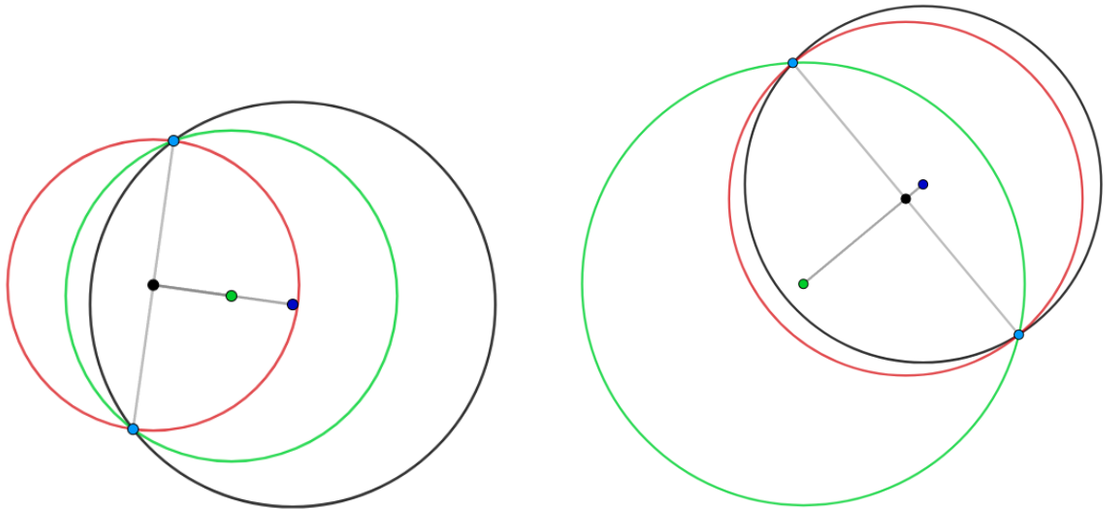

Figura 4.8: In nero la circonferenza sulla quale sono posti i vertici. In verde il centro e la circonferenza impiegata. In rosso la circonferenza se posta sul punto medio.
A sinistra il caso di vertici relativamente vicini, a destra il caso in cui sono posti ad una distanza maggiore.

Osservando la figura 4.8 si può effettivamente comprendere la necessità che vertici lontani siano collegati con un arco molto esterno e viceversa. La simulazione di tale comportamento è reperibile a questo link GeoGebra.

Operando in questo modo, è impossibile che due archi si incrocino due o più volte. Se si incontrassero due volte significa che possono in realtà essere disegnati senza alcuna intersezioni, cosa graficamente impossibile dato che il metodo impiegato costruisce archi tra vertici lontani al di sopra di tutti gli altri.

Di particolare complessità risulta essere la determinazione degli angoli ai quali sono posti i vertici. In base alla loro posizione, e più in particolare in base a dove viene posto il centro, viene stabilito se disegnare l'arco di circonferenza, sul quale i due vertici incidono, di lunghezza maggiore o minore. Nel dettaglio, se il centro è posto sulla circonferenza dei vertici, sarà sempre l'arco maggiore ad essere tracciato, mentre in tutti gli altri casi sarà l'arco minore.

È inoltre importante determinare il verso della rotazione, condizione necessaria al corretto funzionamento del metodo proprio del canvas per il disegno di archi, *drawArc([...])*.

Essendo l'asse della verticale ribaltato, non solo i coefficienti angolari dei segmenti sono in realtà da considerarsi opposti ma anche gli angoli non coincidono. È necessario ricavare l'angolo dapprima calcolando l'arcotangente del coefficiente angolare del segmento, per poi andare ad analizzare le coordinate del punto rispetto al centro del canvas, così da poter evincere il quadrante al quale appartiene il punto e modificare il valore dell'angolo di conseguenza.

```
function getAngleNewOrigin(newO, point) {
  let xO = newO.x;
  let yO = newO.y;
  let dX = point.x - xO;
  let dY = point.y - yO;
  let angle = Math.round(toDegrees(Math.atan(dY / dX)));

  if (dX >= 0 && dY >= 0) {
    angle = 360 - angle;
  } else if (dX < 0 && dY < 0) {
    angle += 180;
    angle -= 2 * (angle - 180);
  } else if (dX < 0 && dY >= 0) {
    angle += 180;
    angle += 2 * (180 - angle);
  } else if (dX >= 0 && dY < 0) {
    angle = -angle;
  }

  return angle;
}
```

Figura 4.9: Funzione che restituisce il valore dell'angolo tra due punti, uno rispetto all'altro.

Questa operazione viene fatta non solo relativamente ai vertici ed il centro, ma anche rispetto ad altri punti rispetto a punti arbitrari. La funzione implementata 4.9

funziona quindi per due punti qualsiasi, restituendone l'angolo che si avrebbe se il piano fosse cartesiano, ovvero l'angolo con il quale effettivamente si lavora.

4.3.4 Scelta delle curve interne nella visualizzazione di Harary-Hill

Relativamente alla visualizzazione di Harary-Hill, è necessario stabilire una metodologia di disegno per gli archi posti tra le due circonferenze che collegano i vertici esterni con quelli interni. Un primo approccio potrebbe suggerire l'uso di circonferenze, cosa difficilmente implementabile in quanto, per poter rappresentare la circonferenza, è necessario definirne il centro, da trovare rispetto alla posizione dei vertici. Considerando però che dati due punti vi sono infiniti cerchi che passano per entrambi, risulterebbe necessario trovare un ulteriore metodo per determinare quale tra di essi considerare. Inoltre, vi sono casi limite nei quali il numero di intersezioni aumenta a causa dell'angolo con il quale l'arco di circonferenza interseca i punti portando la circonferenza a passare attraverso la circonferenza sulla quale giacciono i vertici.

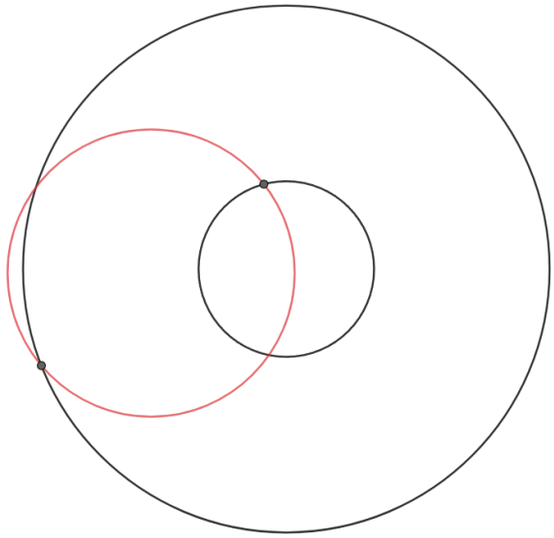

Figura 4.10: Esempio della problematica inerente all'utilizzo di semplici circonferenze.

Il passaggio successivo è stato effettuare una ricerca nei confronti di curve che potessero svolgere questa funzione in modo opportuno.

Tale ricerca ha portato all'implementazione di spirali di Archimede [18]. Una spirale di Archimede è una curva descrivibile in coordinate polari nella forma $R(\theta) = a + b\theta$, tale legge genera una spirale con intersezioni con gli assi poste a distanza uniforme.

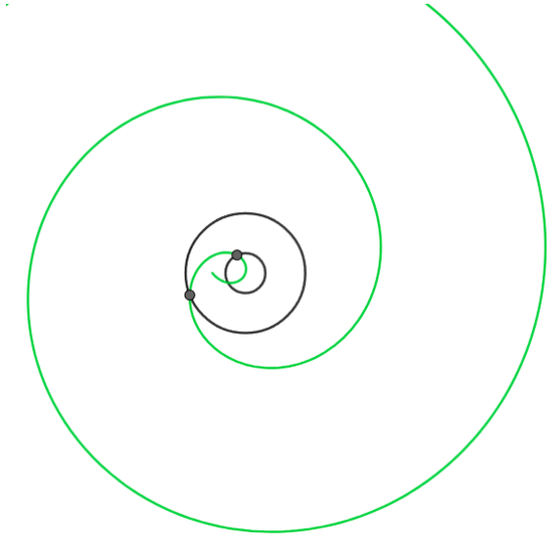

Figura 4.11: Spirale Archimedea passante per i due vertici dati.

In particolare, ai fini della visualizzazione verrà impiegato il solo tratto posto tra le due circonferenze.

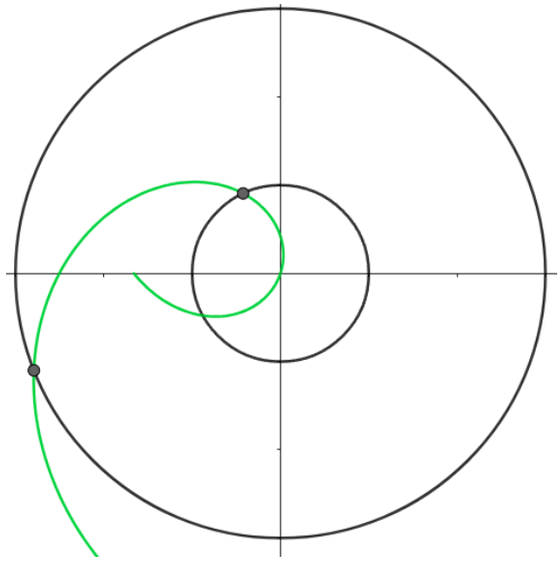

Figura 4.12: Segmento di interesse della spirale tra i due vertici.

La classe canvas di HTML non dispone dei mezzi necessari al disegno di spirali, è stato quindi necessario implementare una metodologia di disegno per le curve tramite JavaScript. La creazione dell'arco di spirale consiste in un elevato numero di segmenti molto piccoli e posti sequenzialmente, così da dare l'effetto di una curvatura continua. Partendo dall'angolo nel quale sono posti i due vertici, relativamente al centro, si procede andando ad individuare la direzione della spirale, il qui arco di interesse sarà posto nel settore circolare con ampiezza minore, piuttosto che quello maggiore, così che abbia un percorso di lunghezza inferiore e di conseguenza avvengano meno intersezioni con altre curve.

Anche qui i valori dell'angolo sono ricavati tramite la funzione 4.9.

Individuata la finestra angolare occupata dalla spirale, il suo valore angolare viene frazionato, così da ottenere l'incremento che ogni segmento ha nei confronti del precedente. Operazione analoga viene eseguita per determinare la distanza dal centro del punto che si sta generando, dividendo equamente la distanza che si interpone tra le due circonferenze.

Così facendo si ottiene un insieme di punti, che collegati vanno a formare l'arco ricercato.

Anche per questa curva è reperibile una simulazione, a questo link Geogebra.

4.4 Intersezioni

Il calcolo delle intersezioni si divide principalmente in due categorie: intersezioni tra linee ed intersezioni tra circonferenze.

Tutte le intersezioni vengono inserite in appositi array, così da non doverle ricalcolare laddove non fosse necessario, ovvero se l'utente vuole semplicemente nasconderle o palesarle nuovamente, oppure se alcuni vertici vengono selezionati: solamente le nuove intersezioni andranno ad essere calcolate. Questa accortezza comporta un

miglioramento delle prestazioni ed una maggiore fluidità del programma durante l'utilizzo.

4.4.1 Linee

Effettuare un'intersezione tra linee, e quindi segmenti, coincide con l'intersecare due rette e se vi è soluzione controllare che il punto di intersezione appartenga al dominio dei due segmenti.

Per poter trovare il punto di intersezione è necessario esprimere i segmenti come rette nella forma $ax + by = c$. Si può quindi notare che si è di fronte ad un sistema lineare a due variabili, x ed y e quindi:

$$\begin{cases} xa_1 + yb_1 = c_1 \\ xa_2 + yb_2 = c_2 \end{cases}$$

Per il calcolo di tali coefficienti è necessario ricondursi alla formula della retta per due punti. Svolgendo alcuni calcoli, è possibile esplicitare i coefficienti di x ed y.

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

$$(y_2 - y_1)(x - x_1) = (y - y_1)(x_2 - x_1)$$

$$x(y_2 - y_1) - x_1(y_2 - y_1) = y(x_2 - x_1) - y_1(x_2 - x_1)$$

$$x(y_2 - y_1) - y(x_2 - x_1) = x_1(y_2 - y_1) - y_1(x_2 - x_1)$$

$$a = y_2 - y_1,\ b = x_1 - x_2,\ c = x_1(y_2 - y_1) - y_1(x_2 - x_1)$$

Avendo ora un sistema lineare nella forma ricercata, possiamo applicare la regola di Cramer per ricavarne la soluzione, a patto che le rette non siano parallele.

Effettuato il calcolo è necessario confrontare i valori degli estremi dei due segmenti per stabilire se l'intersezione trovata appartenga o meno ad essi.

Si noti che se due linee sono uscenti da uno stesso vertice non la si considera essere un'intersezione, nonostante sia effettivamente una soluzione. Allo scopo si controlla che i due segmenti non siano consecutivi e che una soluzione esista, solo successivamente verrà calcolata e verificato il dominio.

```
static segmentIntersection(l1, l2) {
  if (Line.consecutive(l1, l2)) {
    return null;
  }

  let detA = l1.facX * l2.facY - l1.facY * l2.facX;
  if (detA == 0) {
    return null;
  }
  let x = (l1.c * l2.facY - l1.facY * l2.c) / detA;
  let y = (l1.facX * l2.c - l1.c * l2.facX) / detA;
  if (Line.checkDomain(l1, l2, new Point(x, y))) {
    return new Point(x, y);
  }
  return null;
}
```

Figura 4.13: Metodo statico della classe "Line" per trovare l'intersezione dei segmenti.

4.4.2 Archi di spirale

Come già detto gli archi di spirale sono in realtà composti da un insieme di segmenti. Non è possibile però utilizzare direttamente il metodo sviluppato per i segmenti normali perché per tutti i segmenti appartenenti allo stesso arco l'algoritmo andrebbe a cercare una soluzione, cosa ovviamente non di interesse. Per ovviare a ciò si è estesa la classe per i segmenti alla classe "ArcLine", con informazioni aggiuntive riguardanti l'arco di appartenenza e l'indice del segmento stesso all'interno dell'arco.

Con queste considerazioni il procedimento rimane invariato, a meno di qualche condizione aggiuntiva per minimizzare le iterazioni.

```
static segmentIntersection(l1, l2) {
  if (l1.index == 0 || l1.index == SEGMENTS - 1 ||
        l2.index == 0 || l2.index == SEGMENTS - 1) {
    return null;
  }
  if(l1.arcID==l2.arcID){
    return null;
  }
  let detA = l1.facX * l2.facY - l1.facY * l2.facX;
  if (detA == 0) {
    return null
  }
  let x = (l1.c * l2.facY - l1.facY * l2.c) / detA;
  let y = (l1.facX * l2.c - l1.c * l2.facX) / detA;
  if (Line.checkDomain(l1, l2, new Point(x, y))) {
    return new Point(x, y);
  }
  return null;
}
```

Figura 4.14: Metodo statico della classe "ArcLine" per trovare l'intersezione dei segmenti di arco.

In aggiunta a ciò si adopera una matrice di adiacenza nella quale registrare se due archi sono già stati intersecati tra di loro. Se non si tenesse conto di questa cosa ogni intersezione sarebbe ricercata più volte.

4.4.3 Circonferenze

L'intersezione tra le circonferenze risulta essere più semplice, in quanto per costruzione non possono verificarsi casi particolari, come circonferenze concentriche o addirittura coincidenti. Anche per le circonferenze si è creata una classe apposita, "Circle", con le sole informazioni relative al centro, al raggio e ai due vertici che

interseca. Per ricercare i punti di intersezione, in generale più di uno, si procede verificando che i due cerchi non siano più distanti di quanto lo siano i due raggi sommati. Successivamente viene messo in atto un procedimento geometrico mirato al trovare dei segmenti che, se analizzati, portano alla soluzione del problema.

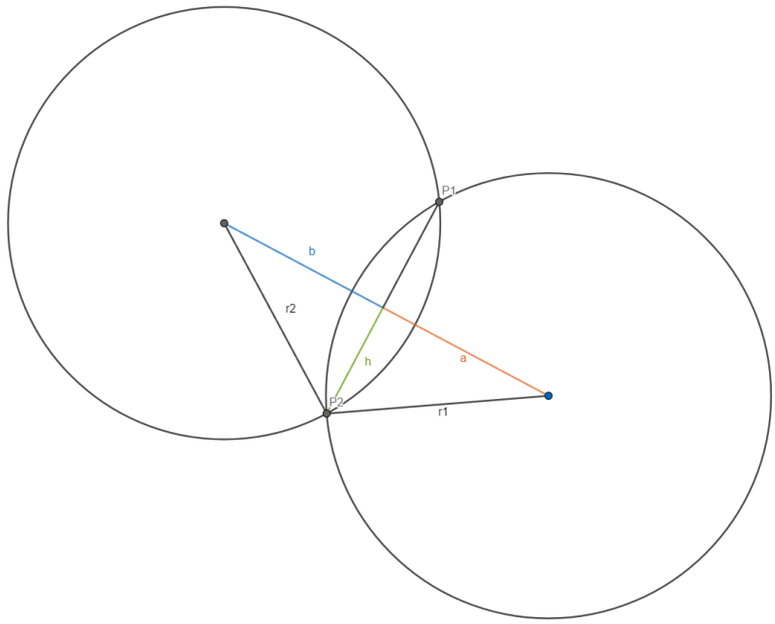

Figura 4.15: Impostazione geometrica per il problema delle intersezioni tra due circonferenze.

L'algoritmo consiste nel ricavare la distanza 'h' della figura 4.15 ed il punto di incrocio tra i due segmenti. Per fare ciò si utilizzano il teorema di Pitagora, la distanza tra i due cerchi e la posizione di essi. La spiegazione dettagliata dell'algoritmo può essere consultata a questo link.

```
static circleIntersection(circle1, circle2, domainRadius, domainCenter) {
    const x1 = circle1.center.x;
    const y1 = circle1.center.y;
    const r1 = circle1.radius;
    const x2 = circle2.center.x;
    const y2 = circle2.center.y;
    const r2 = circle2.radius;
    const d = Math.hypot(x2 - x1,y2 - y1);
    if (d > r1 + r2) {
      return null;
    }
    if (d < Math.abs(r1 - r2)) {
      return null;
    }
    if(Circle.startTogether(circle1,circle2)){
      return null;
    }
    const a = (r1*r1 - r2*r2 + d*d) / (2 * d);
    const h = Math.sqrt(r1*r1 - a*a);

    const x3 = x1 + (a * (x2 - x1)) / d;
    const y3 = y1 + (a * (y2 - y1)) / d;

    let s1 = new Point(x3 + (h * (y2 - y1)) / d
        ,y3 - (h * (x2 - x1)) / d)
    let s2 = new Point(x3 - (h * (y2 - y1)) / d
        ,y3 + (h * (x2 - x1)) / d)
    if (Point.distance(s1,domainCenter)>domainRadius){
      return s1;
    }
    if (Point.distance(s2,domainCenter)>domainRadius){
      return s2;
    }

    return null;
}
```

Figura 4.16: Metodo statico della classe "Circle" per trovare l'intersezione tra le circonferenze.

In particolare, il codice determina questo algoritmo:

- viene calcolata la distanza tra i due centri;
- se i cerchi sono troppo distanti o concentrici si restituisce *null*;
- se i cerchi disegnano archi che partono in uno stesso punto si restituisce *null*
- vengono calcolate le dimensioni dei segmenti, riferirsi alla figura;
- vengono creati i due punti: se sono all'interno del cerchio dei vertici allora l'intersezione non deve essere considerata in quanto quel segmento di arco non è effettivamente disegnato.

4.5 Gestione degli eventi

La maggior parte degli elementi che compongono la pagina possono dar luogo ad eventi, ad esempio i pulsanti ai quali è collegata la generazione del grafo o gli slider ai quali sono collegate le azioni di ridimensionamento e rotazione.

Inoltre il canvas stesso può generare un particolare evento: la selezione di un vertice. Questa è la funzionalità che permette di isolare uno o più vertici effettuando la combinazione del tasto *CTRL* ed il tasto sinistro del mouse.

L'operazione consiste nel far scaturire un evento se viene registrata la sequenza di tasti e ricavare la posizione del puntatore all'interno del canvas. Dopodiché sulla base di quale sia il tipo di visualizzazione corrente, si effettua una verifica relativamente alla posizione dell'evento rispetto ai vari vertici, andando in particolare a guardare se la differenza tra di essi risulta essere minore del raggio dei vertici. In caso affermativo l'indice di tale vertice viene aggiunto all'insieme dei vertici selezionati, che in partenza sarà quindi vuoto.

```
drawCanvas.addEventListener('click', (event) => {
  if (event.ctrlKey) {
    const board = drawCanvas.getBoundingClientRect();
    const xPos = event.clientX - board.left;
    const yPos = event.clientY - board.top;
    checkClickPosition(xPos, yPos);
  }
});
```

Figura 4.17: Listener per il canvas al click su schermo, da creare al lancio dell'applicazione.

Il disegno viene rigenerato colorando opportunamente tutti gli elementi che fanno parte di quelli selezionati dall'utente, quindi i vertici e gli eventuali archi che li collegano. Tutti i restanti elementi sono inoltre disegnati con un tratto meno spesso, così da esaltare ancora più quelli selezionati pur mantenendo il grafo completo.

4.6 Ruolo del CSS

Concettualmente, l'applicazione preserva le sue funzionalità anche senza le caratteristiche strutturali fornite dalla formattazione CSS. Ne rimane comunque importante l'utilizzo per rendere il tutto più "user-friendly", in particolare per organizzare meglio la sezione di input ed output e per poter gestire aspetti puramente estetici, come la colorazione degli elementi.

Figura 4.18: Un bottone prima e dopo essere stato stilizzato tramite CSS.

4.7 Aspetto finale dell'applicazione

Successivamente all'implementazione di tutte le funzionalità descritte nel corso del capitolo e ad una fase di stilizzazione dell'aspetto estetico tramite CSS, l'applicazione finale si presenta così:

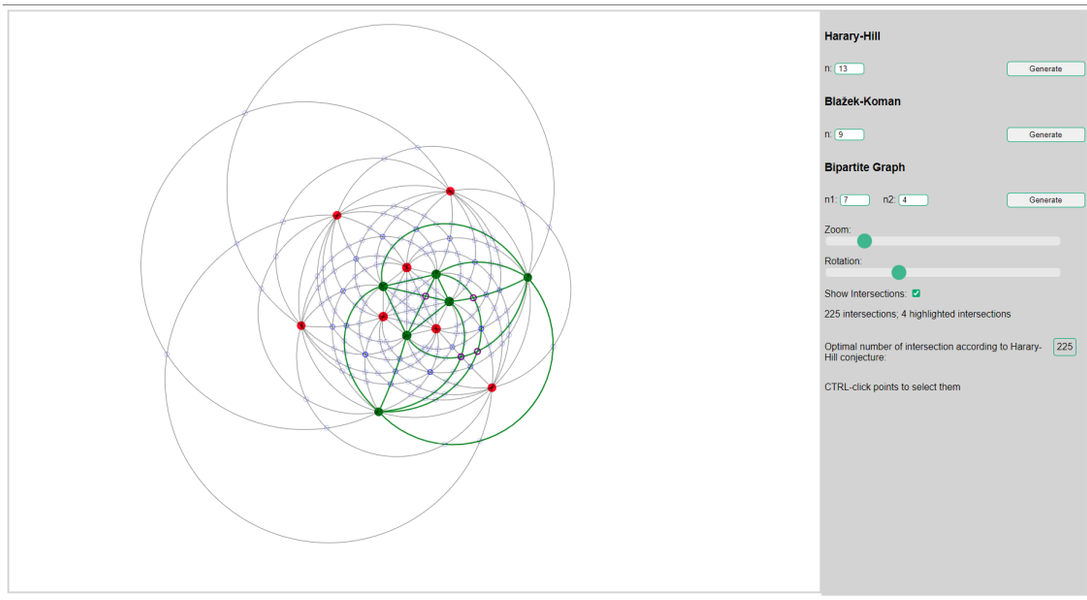

Figura 4.19: Aspetto dell'applicazione al suo stadio finale.

Capitolo 5

Dimostrazione di uso per il fine proposto

Verranno di seguito riportati alcuni risultati ottenuti dall'applicazione, per dimensioni diverse, visualizzazioni diverse e selezionando parte dei vertici dei grafi.

5.1 Esempio 1

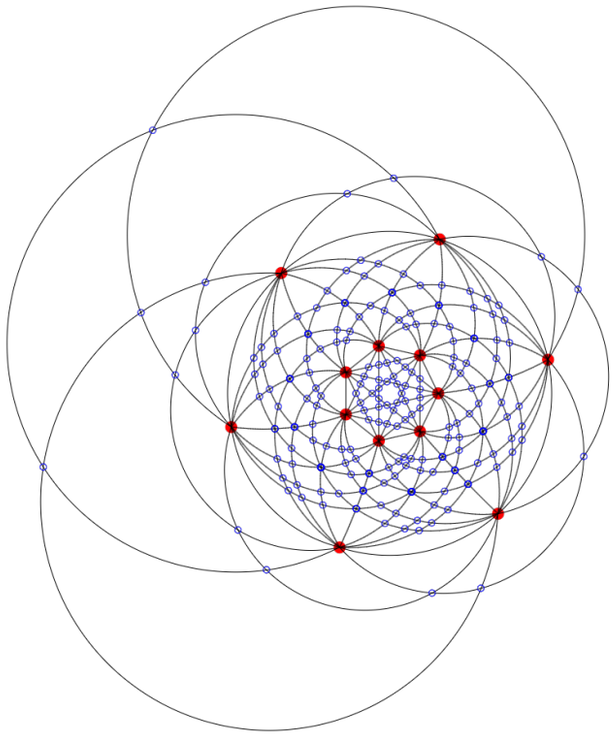

Figura 5.1: Numero di incroci: 225.
Incroci secondo la congettura: 225

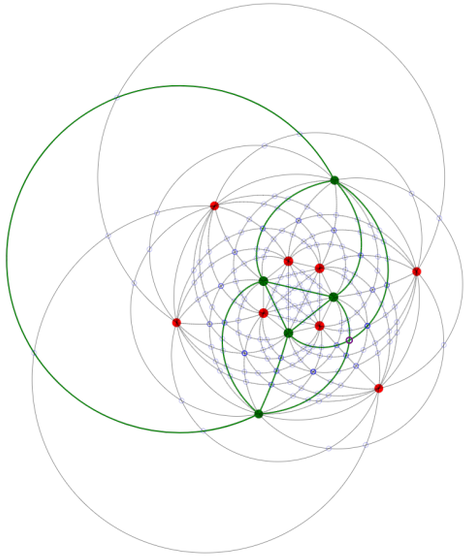

Figura 5.2: Con 5 vertici selezionati, 1 intersezioni tra di loro.

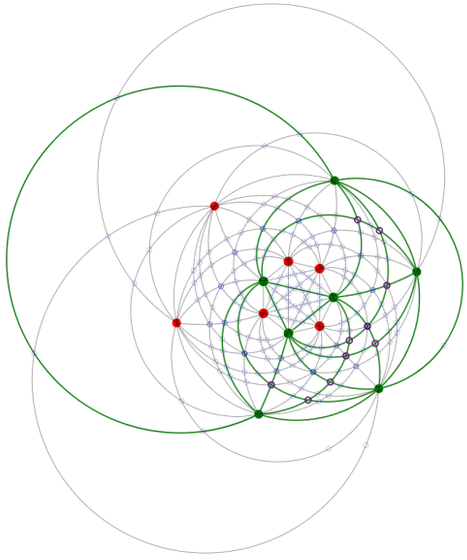

Figura 5.3: Con 7 vertici selezionati le intersezioni diventano 10.

5.2 Esempio 2

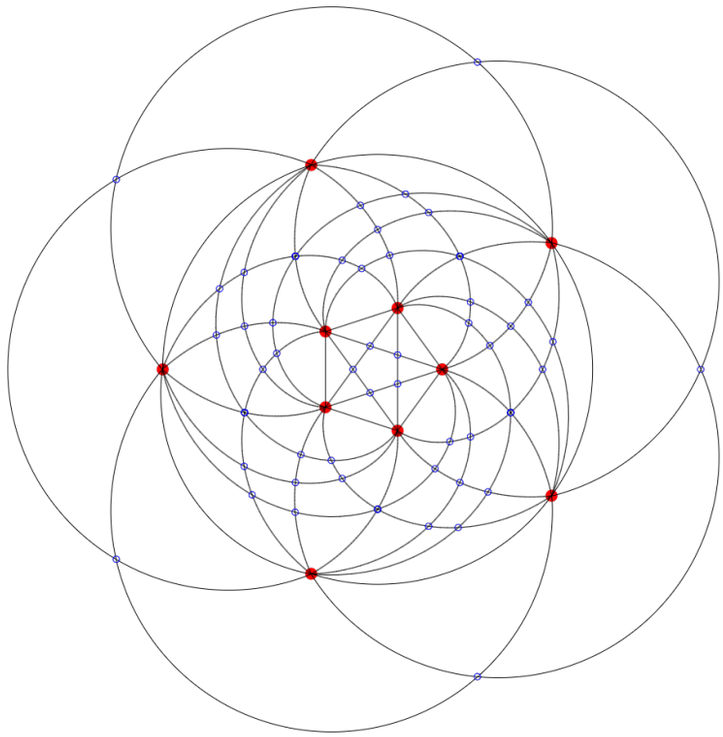

Figura 5.4: Vertici: 10, numero di incroci: 60.
Incroci secondo la congettura: 60

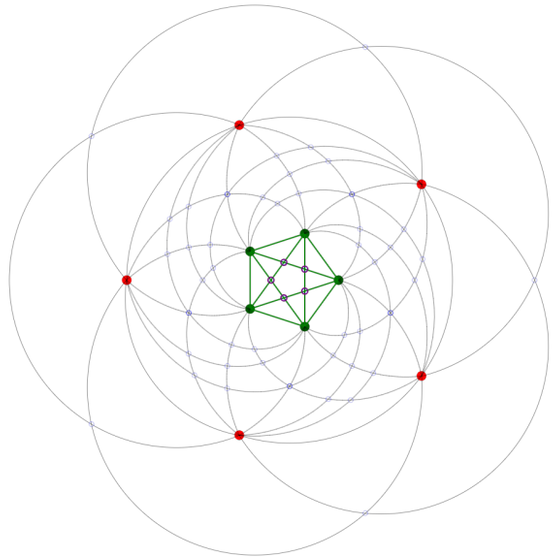

Figura 5.5: Con 5 vertici selezionati, 5 intersezione tra di loro.

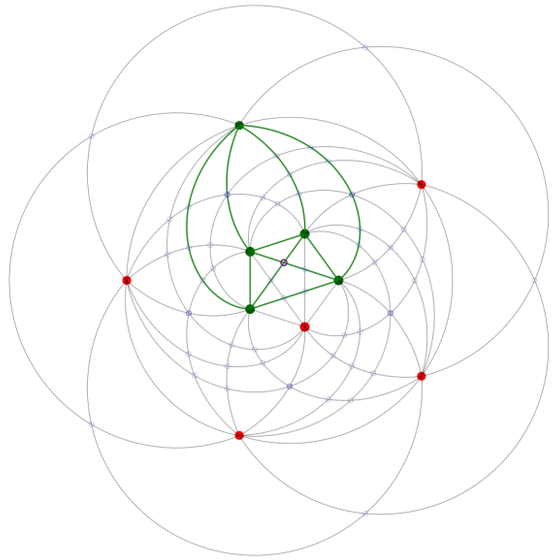

Figura 5.6: Con 5 vertici diversi l'intersezione può essere una sola.

5.3 Esempi 3 e 4

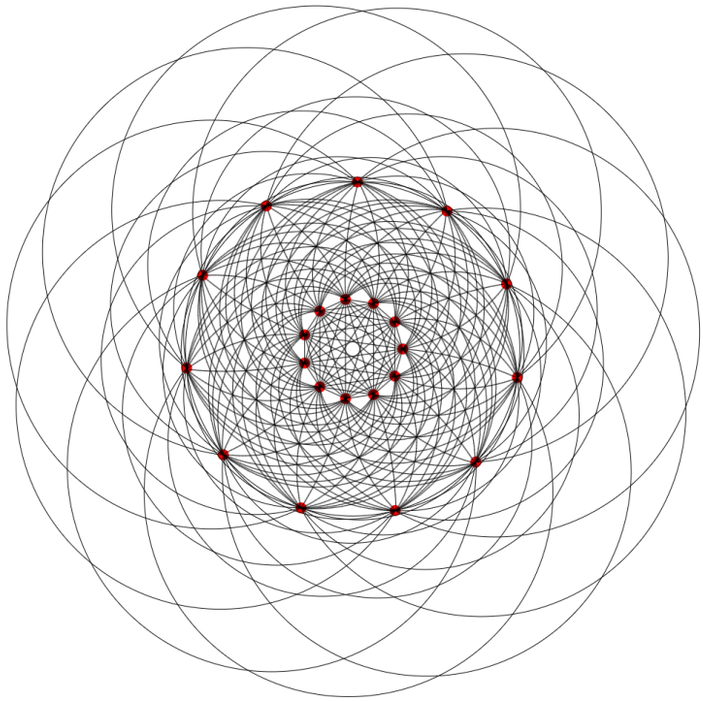

Figura 5.7: Rappresentazione di grafo completo con 22 vertici secondo il metodo di Harary e Hill

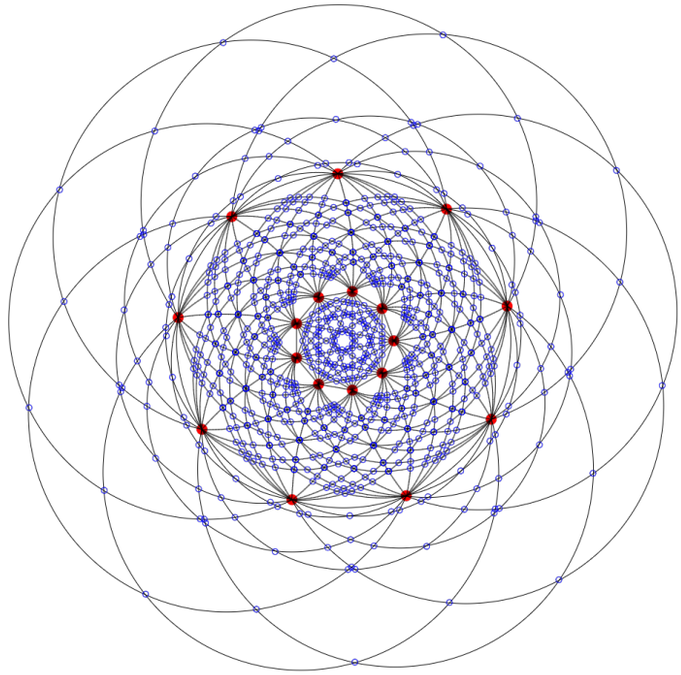

Figura 5.8: Vertici: 18, numero di incroci: 1008.
Incroci secondo la congettura: 1008

5.4 Esempio 5

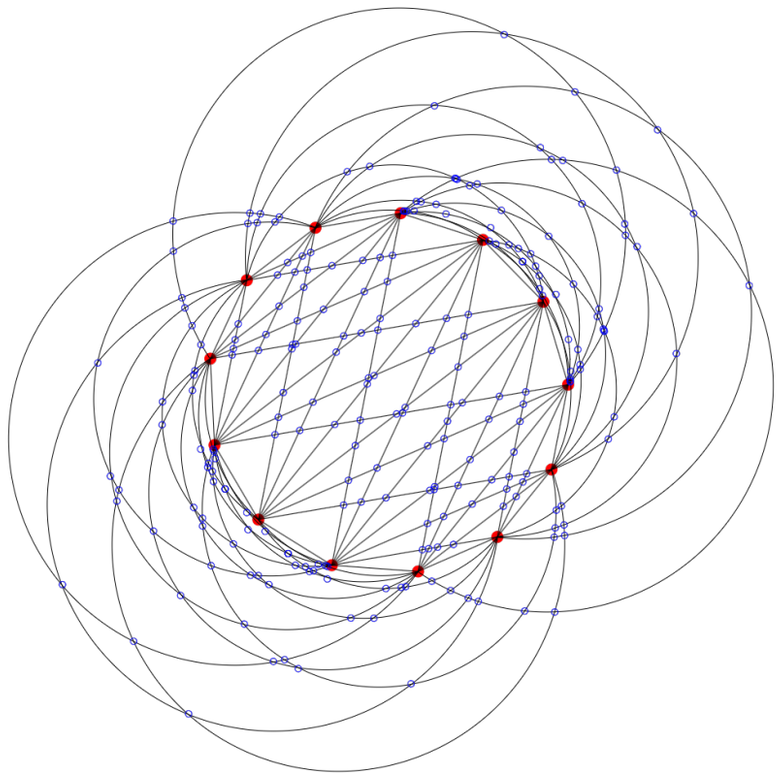

Figura 5.9: Vertici: 13, numero di incroci: 225.
Incroci secondo la congettura: 225

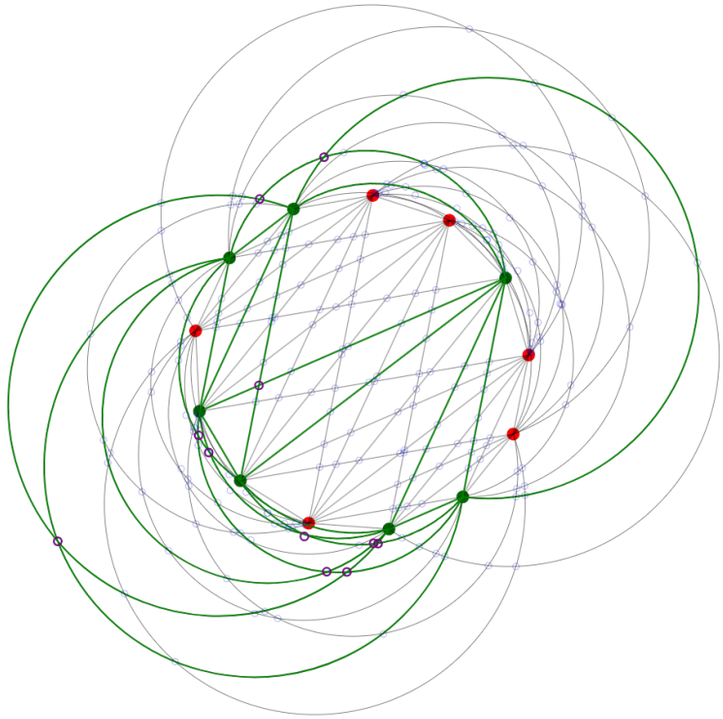

Figura 5.10: Con 7 vertici selezionati, 11 intersezioni tra di loro.

5.5 Esempio 6

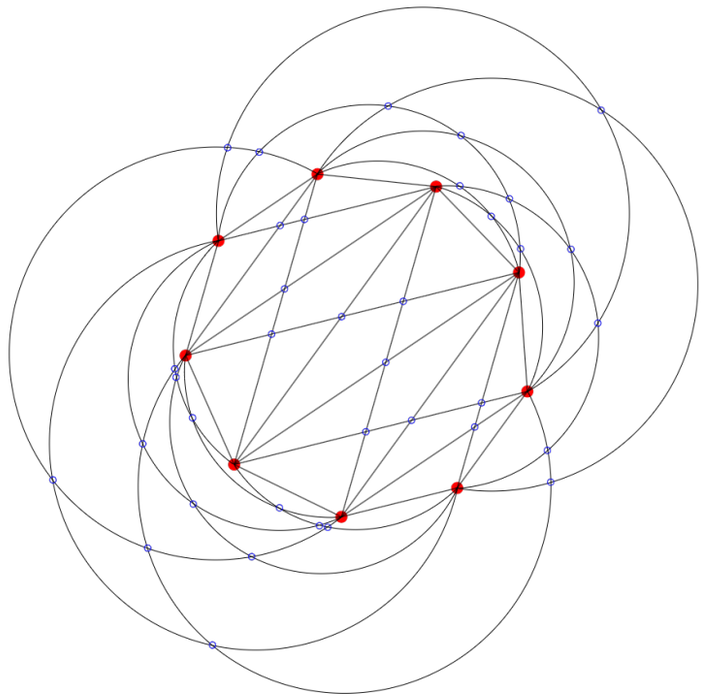

Figura 5.11: Vertici: 9, numero di incroci: 36.
Incroci secondo la congettura: 36

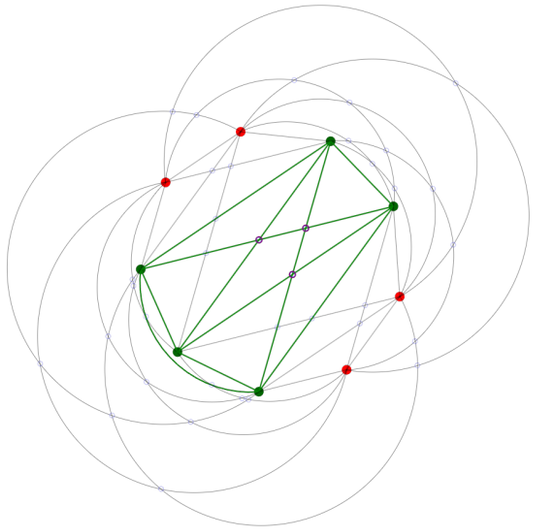

Figura 5.12: Con 5 vertici selezionati, 3 intersezioni tra di loro.

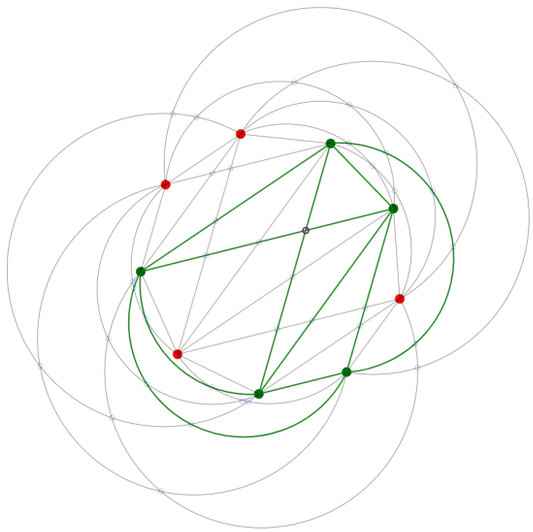

Figura 5.13: Con lo stesso numero di vertici selezionati,
si può avere una sola intersezione.

5.6 Esempio 7

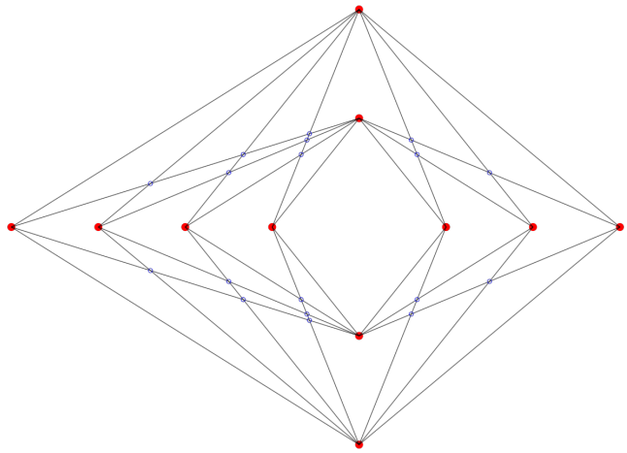

Figura 5.14: Partizioni di dimensioni 7 e 4: 18 incroci

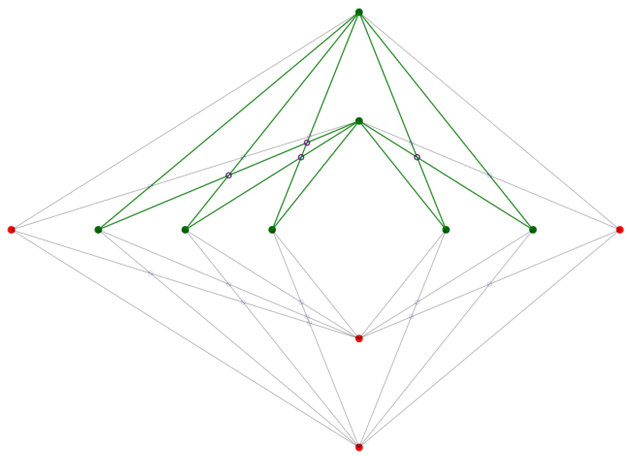

Figura 5.15: Selezionando sottopartizioni di 5 e 2 elementi
si ottengono 4 incroci in evidenza.

5.7 Esempio 8

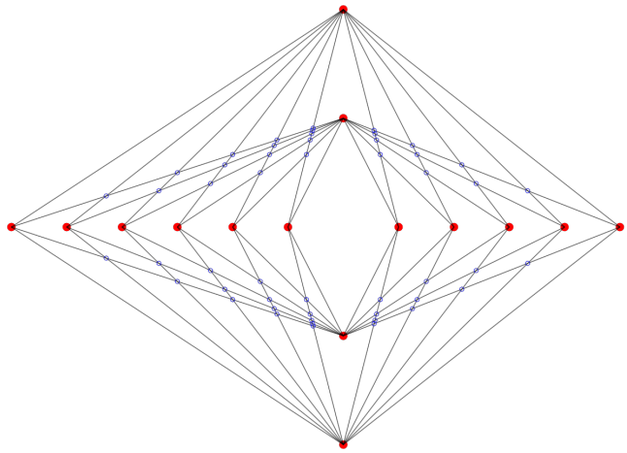

Figura 5.16: Grafo bipartito di dimensioni 11 e 4: 50 incroci.

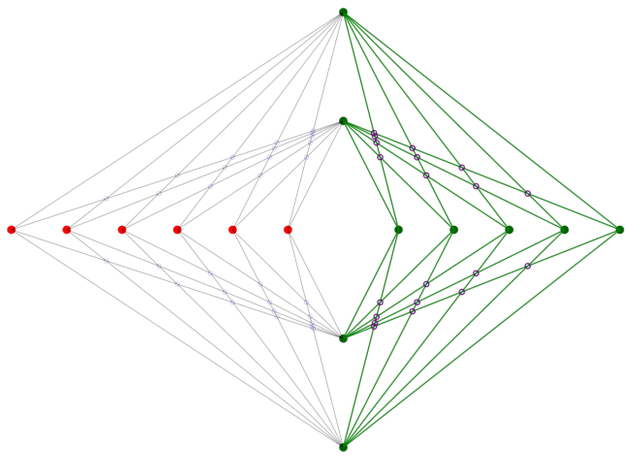

Figura 5.17: Selezionando la partizione più piccola e 5 elementi dell'altra si evidenziano 20 intersezioni

Conclusioni e sviluppi futuri

Conclusioni

Nell'introduzione si è discusso in merito all'utilità di disporre di un sistema software a supporto dello studio delle intersezioni di grafi completi, data l'elevata difficoltà del problema e l'importanza che assolve nel contesto dello studio di un problema tramite l'uso di grafi.

Per far ciò il progetto ha sviluppato un software per rappresentare grafi tramite tecniche di disegno introdotte da Harary-Hill, Blažek-Koman e Turán, trovando una metodologia che potesse rispettarne i criteri di costruzione ed andando poi a ricercare le intersezioni tra i loro archi.

L'applicazione sviluppata sembrerebbe riuscire ad assolvere le funzionalità imposte, rappresentando grafi completi e garantendo che il numero di incroci corrisponda a quello dettato dalla formula di Harary e Hill, confermando quindi la correttezza del disegno. Correttezza che risulta verificata anche per la costruzione di Blažek-Koman e per la costruzione bipartita che a loro volta verificano il valore della congettura.

Questo *tool* può prestarsi adeguatamente ad un uso nell'ambito della ricerca, costituendo un buono strumento per la visualizzazione dei grafi completi, procedura tediosa da riprodurre manualmente e che tende al non essere realizzabile per dimensioni del grafo elevate.

Data la correttezza delle rappresentazioni, si è poi sviluppata la funzionalità

effettiva da impiegare per lo studio dei grafi, ovvero il poterne selezionare dei vertici per osservarne nel dettaglio i comportamenti. Selezionando i vertici, infatti, questi verranno evidenziati assieme a tutti gli archi che li collegano così da poter rintracciare particolari sotto configurazioni utili ai fini della comprensione del problema.

Work In Progress e Future Implementazioni

Al momento il progetto può essere considerato come un primo prototipo, essendovi aspetti introducibili ed altri potenzialmente non perfetti. Una prima idea potrebbe essere implementare altri metodi di visualizzazione di grafi completi, ma potendosi anche estendere a grafi non completi. Sono in via di sviluppo anche funzionalità relative al front-end, per assicurarsi una migliore esperienza di uso.

Si può pensare di introdurre nuovi comandi, quali slider, per modificare singolarmente i raggi e la rotazione di entrambi i cerchi oppure per distribuire separatamente gli elementi dei grafi bipartiti. Può essere anche introdotto un sistema di modifica dei colori, ad esempio tramite un *color picker*, per modificare l'aspetto dei vertici e degli archi.

Implementare le cose sopra citate non comporta riscrittura del codice, in quanto le varie parti dell'applicazione sono ben segmentate e modulari. Sarà solamente necessario aggiornare il file HTML, aggiungere gli opportuni metodi, per poi effettuare una riformattazione della pagina tramite CSS.

Un aspetto più importante da poter considerare, è lo sviluppo di un sistema per poter esportare i disegni così generati in più formati, per poter essere utilizzati in altri applicativi, quali ad esempio IPE [19], un software molto usato nell'ambito della ricerca per il disegno digitale.

L'unico aspetto migliorabile relativo al grafo stesso potrebbe essere la ricerca di curve tali da rendere il disegno più compatto e meno esteso, almeno per dimensioni

relativamente piccole, come nella figura 5.1.

Questa miglioria potrebbe essere non banale, in quanto già solo l'ottenere la soluzione implementata ha necessitato di molta ricerca, incontrando anche problematiche abbastanza complicate. In particolare si fa riferimento allo sviluppo di una funzione che ridimensioni le curve coerentemente alla quantità e alla numerazione dei vertici.

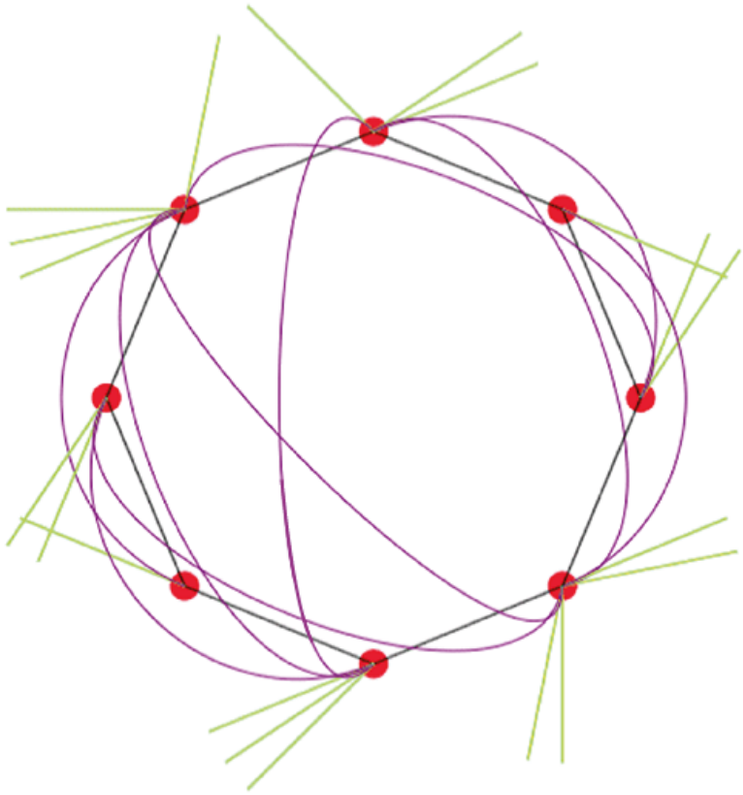

Figura 5.18: Potenzialità delle curve di Bèzier.

Nonostante le curve siano ovviamente non corrette, l'angolo con il quale esse partono dai vertici è molto promettente se si riuscisse a trovare un buon metodo per aumentarne la grandezza ma che possa funzionare indipendentemente dalla grandezza del grafo.

Bibliografia

[1] *Structuring the Web in HMTL*, `https://developer.mozilla.org/en-US/docs/Learn/HTML`

[2] *(Tim Berners-Lee e Daniel Connolly, Hypertext Markup Language (HTML) - A Representation of Textual Information and MetaInformation for Retrieval and Interchange (TXT), su World Wide Web Consortium, 1993*

[3] *JavaScript documentation by Mozilla*, `https://developer.mozilla.org/en-US/docs/Web/JavaScript`

[4] *(Mozilla.org, CSS: Cascading Style Sheets*, `https://developer.mozilla.org/en-US/docs/Web/CSS`

[5] *Geogebra, calcolatrice grafica*, `http://www.geogebra.at/`

[6] *Bondy, John Adrian; Murty, U. S. R. (1976), Graph Theory with Applications, North-Holland, p. 5, ISBN 0-444-19451-7*

[7] *Diestel, Reinhard (2005), Graph Theory (3rd ed.), Springer, ISBN 3-540-26182-6. Electronic edition*

[8] *Purchase, Helen C.; Cohen, Robert F.; James, Murray I. (1995). "Validating graph drawing aesthetics". In Brandenburg, Franz-Josef (ed.). Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22,*

1995, Proceedings. Lecture Notes in Computer Science. Vol. 1027. Springer. pp. 435–446.

[9] *Guy, R. K. (1960). "A combinatorial problem". Nabla (Bulletin of the Malayan Mathematical Society). 7: 68–72*

[10] *Pàl Turán*, https://en.wikipedia.org/wiki/P%C3%A1l_Tur%C3%A1n

[11] *The brick Factory Problem - Pàul Turán*, https://en.wikipedia.org/wiki/Tur%C3%A1n%27s_brick_factory_problem

[12] *Anthony Hill (artist)*, https://en.wikipedia.org/wiki/Anthony_Hill_(artist)

[13] *Frank Harary*, https://it.wikipedia.org/wiki/Frank_Harary

[14] *Shengjun Pan and R. Bruce Richter. The crossing number of K11 is 100, J. Graph Theory 56 (2007), 128–134.*, http://dx.doi.org/10.1002/jgt.20249

[15] *J. Blazek and M. Koman, A minimal problem concerning complete plane graphs, In: M. Fiedler, editor: Theory of graphs and its applications, Czech. Acad. of Sci. (1964) 113–117*

[16] *Zarankiewicz, K. (1954), "On a problem of P. Turan concerning graphs", Fundamenta Mathematicae, 41: 137–145*, doi:10.4064/fm-41-1-137-145

[17] *Curve di Bezièr*, https://it.wikipedia.org/wiki/Curva_di_B%C3%A9zier

[18] *Spirale Archimedea*, https://it.wikipedia.org/wiki/Spirale_archimedea

[19] *The IPE extensible drawing editor*, https://ipe.otfried.org/

[20] *Intersezioni tra circonferenze*, https://lucidar.me/en/mathematics/how-to-calculate-the-intersection-points-of-two-circles/

[21] *Simulazione archi esterni tramite circonferenze*, https://www.geogebra.org/calculator/cjqread7

[22] *Simulazione di spirale passante per due punti*, https://www.geogebra.org/calculator/ksdnqtyp

[23] *Shellable Drawings and the Cylindrical Crossing Number of* K_n, https://link.springer.com/article/10.1007/s00454-014-9635-0/figures/1

[24] *Grafo, Wikipedia*, https://it.wikipedia.org/wiki/Grafo#/media/File:6v-color-graph.svg

[25] *Grafo Bipartito, Wikipedia*, https://en.wikipedia.org/wiki/Crossing_number_(graph_theory)#/media/File:Zarankiewicz_K4_7.svg