



UNIVERSITÀ DI PISA
Dipartimento di Informatica
Corso di Laurea Magistrale in Data Science and Business Informatics

TESI DI LAUREA

**Fiori, reti e frattali:
un approccio nature-inspired per l'analisi di network**

RELATORE
Prof. Giulio Rossetti

CANDIDATA
Federica Trevisan

ANNO ACCADEMICO 2020-21

Riassunto

È possibile implementare un algoritmo partendo da un fiore? Questa tesi unisce tre filoni di ricerca: il *Natural Computing* (NC), la *Social Network Analysis* (SNA) e gli oggetti frattali intrecciandoli fra loro con l'obiettivo di sviluppare una procedura gerarchica per l'analisi di reti. Traendo ispirazione dalla struttura dell'infiorescenza di una pianta appartenente alla famiglia delle ombrellifere, la *Daucus Carota*, e declinando tali concetti in un contesto di tipo algoritmico, è stato implementato *Daucus Fractal Network Analyzer* (DFNA): una procedura che permette di comprimere e decomprimere i network facendo leva sull'assunzione che all'interno di essi siano presenti delle strutture frattali. Dalle sperimentazioni realizzate nasce così una procedura gerarchica che permette di modellare e semplificare network. DFNA identifica all'interno di una rete dei pattern precedentemente definiti, successivamente li comprime in un singolo nodo in maniera iterativa con tre diverse tecniche di compressione, fino a giungere ad una rete priva di pattern ripetuti al suo interno. È prevista anche la procedura DFNA inversa che ricostruisce un'approssimazione del network di partenza a partire da quello compresso. Sono stati effettuati esperimenti su un vasto repertorio di reti di natura differente: DFNA è stato applicato analizzando un totale di dodici reti suddivise in quattro gruppi differenti con dieci pattern diversi. Dagli esperimenti si conferma l'ipotesi che in gruppi di reti omogenee sono presenti pattern frattali.

*È un ritmo ignoto che ci spinge
alle spalle e non posso fare a meno
di pensare: i fiori continuano
implacabili
a non vederci.*

Chandra Livia Candiani

Indice

Riassunto

Acronimi	3
1 Introduzione	4
2 Stato dell'arte e letteratura	8
2.1 Natural Computing	8
2.1.1 Saga degli algoritmi <i>nature-inspired</i>	12
2.1.2 Botanica e descrizione della <i>Daucus Carota</i>	20
2.1.3 Invasive Weed Optimization	23
2.2 Frattali	25
2.2.1 Gli oggetti frattali	25
2.2.2 Generare frattali	30
2.3 Social Network Analysis	36
2.3.1 Network embeddings	41
2.3.2 Networks and Fractals	42
2.4 Conclusione stato dell'arte	44
3 Descrizione procedura	45
3.1 Il problema dell'isomorfismo	45
3.2 Daucus Fractal Network Analyzer	48
3.2.1 Rename	50
3.2.2 Identify	52
3.2.3 Comprime	53
3.2.4 Memorize	53
3.3 Demo DFNA	54

<i>INDICE</i>	2
3.3.1 Compressione singola	55
3.3.2 Compressione multipla	56
3.3.3 Compressione frattale	57
3.4 Commenti sulla complessità di DFNA	58
3.5 Demo DFNA inversa	59
4 Analisi e risultati	62
4.1 Organizzazione esperimenti	62
4.1.1 Reti analizzate	63
4.1.2 Scelta dei motif	70
4.2 Discussione dei risultati	71
4.2.1 Indici statistici	72
4.2.2 DFNA per estrazione di backbone	72
4.2.3 Commento sui risultati degli esperimenti	85
5 Conclusioni	86
5.1 Esperimenti futuri	87
Ringraziamenti	89
Bibliografia	91
Sitografia	95
Elenco delle figure	96
Elenco delle tabelle	102
List of Algorithms	103

Acronimi

ACO *Ant Colony Optimization.*

DC *Daucus Carota.*

DFNA *Daucus Fractal Network Analyzer.*

EC *Evolutionary Computing.*

IWO *Invasive Weed Optimization.*

NC *Natural Computing.*

PSO *Particle Swarm Optimization.*

SNA *Social Network Analysis.*

Capitolo 1

Introduzione

Di là dalle pianure olandesi tappezzate da tulipani, oltre i reticolati di canali e mulini, lungo le rive del fiume Reno che scorre ininterrottamente fin dalle Alpi e di là di queste, in tutti questi punti e in molti altri, dove nei campi incolti e dimenticati regna l'anarchia naturale, c'è una pianta infestante che insieme a molte altre da un campo abbandonato all'altro si espande inosservata: la *Daucus Carota*.

Una formica ha appena trovato una briciola di pane sotto una panchina. Sono seduta su questa panchina e guardo un fiore, una *Daucus Carota*. Mi concentro sulla sua struttura ramificata e ripetitiva. Dietro questo spettacolo di semplicità visiva difficile da spiegare c'è un meccanismo complesso che spazia dalla *Social Network Analysis* alla matematica dei frattali. Visto dall'alto: tanti puntini petalosi. Visto da di fronte: un insieme di ramificazioni. Visto dal basso: uno stelo che si dirama fino all'ombrella. Il tutto reso più affascinante dalla luce del crepuscolo alpino, nasce così l'ispirazione per questa tesi. Nel frattempo la colonia di formiche ha preso il sopravvento della panchina. Questa tesi parte proprio da qui.

Sono stati uniti e intrecciati tra loro tre settori di ricerca ben distinti: da un lato, per ragioni concettuali, il *Natural Computing* (NC) è stato scelto per prendere spunto da fenomeni naturali e studiarli dal punto di vista algoritmico; dall'altro la *Social Network Analysis* (SNA), per avere una ba-

se matematica su cui traslare e costruire *Daucus Fractal Network Analyzer* (DFNA), la procedura sviluppata in questa tesi; terzo la teoria dei frattali, per l'ipotesi dell'esistenza di pattern frattali all'interno di reti.

Dal triplice intreccio è stata sviluppata una procedura gerarchica per l'analisi di network che trae ispirazione dalla struttura di un fiore. Una delle ragioni per cui c'è una stretta corrispondenza tra l'algoritmo presentato e le ramificazioni di una *Daucus Carota* è che per ogni iterazione della procedura DFNA corrisponde una ramificazione del fiore. Con una certa dose

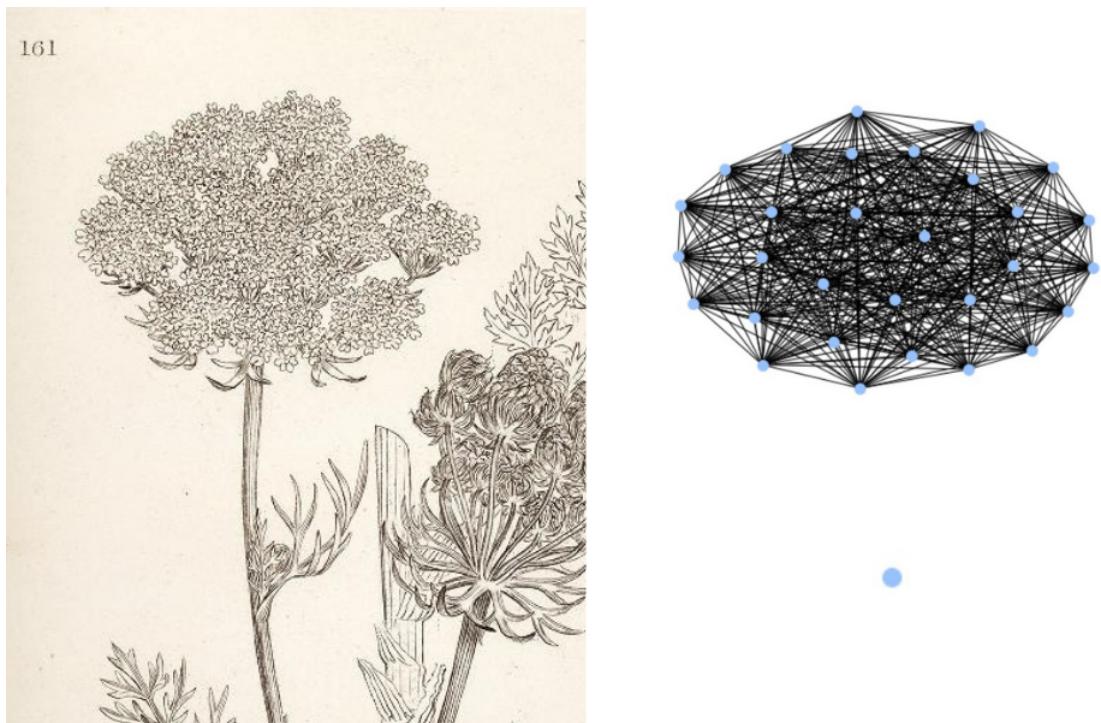


Figura 1.1: Immagine comparativa: a sinistra un'illustrazione di DC consultabile in Medical Botany (1832) di William Woodville ^a, a destra l'iterazione iniziale e finale della procedura DFNA su una rete reale.

^aUn ringraziamento speciale a Lloyd Library and Museum per aver fornito la digitalizzazione di questo raro libro.

di astrazione, si è appurato che ogni punto petaloso appartenente alla corolla del fiore può essere concettualizzato, tralasciando i florilegi, come un insieme di punti, una rete di nodi in qualche modo collegati tra loro, un network. Considerando dall'alto solo l'ombrella, il livello a cui si trovano i fiori, le connessioni tra questi nodi non si trovano però allo stesso livello, sono da estrarre poiché derivano dalle ramificazioni. Una delle ipotesi fon-

damentali di questa tesi è che in una rete che rappresenta fenomeni reali si possano trovare strutture di questo tipo: pattern gerarchici, ripetuti e frattali. Conseguentemente, una volta estratti da una rete, sono un'importante fonte d'informazione descrittiva di quest'ultima.

Il presente lavoro nasce con lo scopo di apportare un contributo unendo i suddetti indirizzi di ricerca, proponendo una nuova metodologia che comprime e decomprime network facendo leva sull'assunzione che all'interno di essi siano presenti strutture frattali. Dopo uno studio della letteratura nasce così un algoritmo per analizzare le componenti frattali ripetitive all'interno di reti. A grandi linee, i pattern ricorrenti vengono definiti ed identificati all'interno di una rete, la quale viene successivamente compressa ripetendo la procedura fino a che non compaiono più pattern. In questo modo, iterazione dopo iterazione, si ricostruisce la struttura ramificata del fiore DC. Nel compito d'identificazione ci troviamo davanti al problema dell'isomorfismo tra grafi, tale problema matematico è oggetto di studio in letteratura sia per il suo vasto coinvolgimento nella teoria dei grafi, sia per il suo essere un problema NP completo. Per le analisi sono stati presi in considerazione quattro gruppi contenenti tre reti di natura profondamente diversa: reti sociali di animali, neurali, ecologiche e chimiche sono state testate in maniera tale da analizzare la risposta di gruppi diversi all'applicazione della procedura DFNA.

Il resto del presente lavoro di tesi è organizzato come segue: il Capitolo 2, verte sull'analisi dello stato dell'arte. Caratterizzato da una forte interdisciplinarità, viene passata in rassegna la letteratura esistente relativa ai tre principali temi che s'intrecciano in questa tesi. Aprono le danze gli algoritmi *nature-inspired* e l'ambito del *Natural Computing*, con un excursus dei principali obiettivi e algoritmi. Segue lo studio dei frattali, con i concetti fondamentali e gli studi inerenti; vengono introdotte infine le nozioni principali di *Social Network Analysis* e di teoria dei grafi necessarie ai fini della comprensione del presente lavoro.

Il Capitolo 3 è strettamente legato alla fase d'implementazione, è qui che vie-

ne introdotto il tema centrale affrontato nel lavoro di tesi e frutto dello studio dell'attuale stato dell'arte. Viene definita la procedura algoritmica *Daucus Fractal Network Analyzer* (DFNA), descrivendo nel dettaglio le varie funzioni che la compongono.

Nel Capitolo 4 viene quindi discussa l'analisi sperimentale effettuata. Vengono qui descritte le quattro classi a cui appartengono le reti su cui è stato applicato il modello DFNA: reti sociali di animali, neurali, ecologiche e chimiche. Nell'analisi degli esperimenti svolti, vengono definite le metriche e gli indici statistici per l'analisi delle performance dell'algoritmo, vengono inoltre riportati e discussi i risultati ottenuti dalle simulazioni.

Chiudendo il cerchio, al postutto nel Capitolo 5 viene fatto il punto della situazione con una chiave di lettura globale, soffermandosi sulle criticità intercorse che hanno delineato il percorso e le decisioni di questo elaborato e presentando le possibili estensioni, integrazioni e sviluppi futuri di questo lavoro giungendo così alle conclusioni della nostra analisi.

Capitolo 2

Stato dell'arte e letteratura

Un algoritmo *nature-inspired* mima i fenomeni naturali, dal comportamento di una colonia di formiche a quello di uno sciame di uccelli; i frattali sono delle strutture matematiche ricorsive presenti di frequente in natura; i network sono delle strutture dati composte da nodi e archi. Per ognuno dei tre settori di ricerca viene dedicata una sezione, dando così un approccio multidisciplinare a questo capitolo e seguendo un filo logico che li connette tra loro e che porterà a definire la procedura algoritmica *Daucus Fractal Network Analyzer* (DFNA) nel Capitolo 3. Per ogni sezione è prevista una spiegazione con alcuni esempi concisi, volti a trasmettere l'idea di fondo. Questo capitolo presenta quindi una spiegazione dei principali algoritmi *nature-inspired*. Vengono poi presentate le strutture frattali e infine viene fatta una descrizione dei concetti e delle metodologie di *Social Network Analysis* (SNA) utili per la fase di sviluppo e analisi.

2.1 Natural Computing

La natura è presente e attiva nella risoluzione di problemi da milioni di anni. Si pensi che 13,5 miliardi di anni prima di oggi viene segnato l'inizio della fisica con l'apparizione di materia ed energia e l'inizio della chimica con l'apparizione di atomi e molecole, mentre viene collocata a 3,8 miliardi di anni fa l'inizio della biologia con la comparsa degli organismi, per passare a 2,5 milioni di anni fa per l'evoluzione del genere *Homo* in Africa. [1]

Nonostante ciò il *Natural Computing* (NC) è una scienza multidisciplinare recente e in espansione. In un primo avvicinamento a questa disciplina, termini come *nature*, *bio*, *genetic* ed *evolutionary* possono sembrare inter-scambiabili. Seguendo la tassonomia definita da De Castro in *Fundamentals of natural computing* [2] presentata nella Figura 2.1 vengono chiarite le differenze tra questi termini e i settori di ricerca che ne derivano.

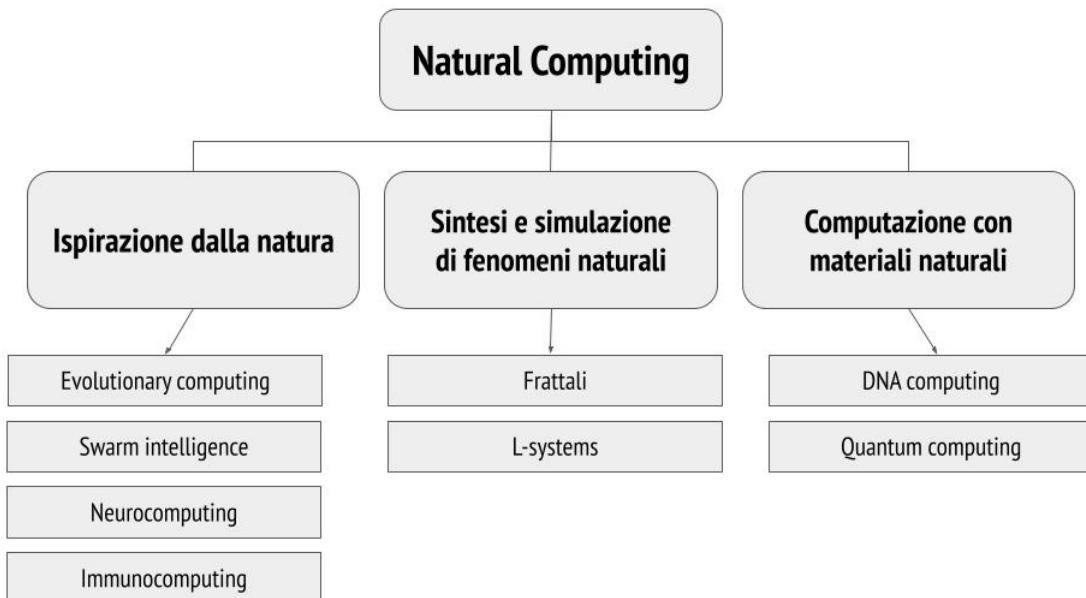


Figura 2.1: La tassonomia del *Natural Computing* (NC) secondo De Castro.

L'associazione dei due termini *natural* e *computing* lascia intendere che il tema in questione abbia a che fare con la programmazione in qualche modo naturale, inherente alla natura. Si scopre così che ci sono diverse sfaccette del rapporto interconnesso tra natura e programmazione.

Stando alla definizione di De Castro in *Fundamentals of natural computing*, [2] il *natural computing* nasce dall'assunzione che sia possibile interconnettere campi di ricerca completamente differenti. È una nuova scienza che si espande su molti campi di ricerca che tiene assieme materie che fino a prima si pensava fossero mutualmente esclusive l'una dall'altra: natura da una parte, informatica dall'altra. Esistono problemi che non possono essere risolti con le attuali tecniche di programmazione e *problem solving*, qui trovano spazio le euristiche e le soluzioni portate dal *natural computing*.

Il NC può essere visto come un interscambio tra informatica e natura da diversi punti di vista: è importante discernere che il NC può essere diviso in tre principali ramificazioni: ispirazione dalla natura, sintesi e simulazione di fenomeni naturali, e infine computazione con materiali naturali.

Ecco le tre aree d'investigazione:

- **Ispirazione dalla natura:** consiste nel partire dalla natura per arrivare alla soluzione di problemi complessi. La natura nel suo complesso viene vista come ispiratrice nello sviluppo e nella progettazione di nuove tecniche computazionali per la risoluzione dei suddetti problemi. Dallo studio e dall'osservazione della natura nascono quindi nuovi algoritmi. Fanno parte di questo paradigma di ricerca, chiamato anche *bio-inspired computing*: *evolutionary computing*, gli algoritmi genetici, la *swarm intelligence*, il *neurocomputing* e *immunocomputing*. Parte di questi algoritmi verranno ampiamente presentati e discussi.
- **Sintesi e simulazione di fenomeni naturali:** l'uso del calcolatore per ricreare, studiare e simulare organismi e fenomeni naturali. La possibilità di testare teorie biologiche che non sarebbero verificabili sperimentalmente, il processo di sintesi volto alla creazione di pattern, forme e comportamenti che assomigliano alla "life-as-we-know-it", come per esempio gli oggetti frattali osservabili nella struttura delle piante, nel profilo delle coste, nei polmoni e così via. I due approcci principali per la simulazione e l'emulazione della natura in computer sono: lo sviluppo di strumenti informatici per lo studio della geometria frattale e i metodi di *artificial life* o *L-Systems* (sistemi che prendono il nome dal loro creatore Lindenmayer), cioè il tentativo di riprodurre artificialmente le caratteristiche degli esseri viventi. Frattali e *L-Systems* vengono abbondantemente approfonditi nella Sezione 2.2 di questo capitolo.
- **Computazione con materiali naturali:** l'utilizzo di materiale naturale per la programmazione in quanto tale, in altre parole per la creazione di calcolatori e procedure per lo svolgimento di calcoli. Si forma così un

nuovo paradigma computazionale che va a sostituire o ad incrementare gli attuali computer basati sul silicio. [2] Sono qui compresi i temi del *DNA computing* e del *quantum computing* [38] che non verranno trattati in questa tesi.

Come idea di fondo rimane quella che dall'interazione con la natura ci sia un'ispirazione e una conseguente creazione di processi e teorie. Detto ciò, il NC può essere definito come il settore di ricerca che, basato o ispirato alla natura, permette lo sviluppo di nuovi strumenti computazionali per *problem solving*, conduce alla sintesi di pattern naturali, comportamenti e organismi, e potrebbe risultare nella progettazione di nuovi sistemi computazionali che utilizzano mezzi naturali per calcolare.

Tra le caratteristiche più speciali del NC nel suo complesso si trova sicuramente la multidisciplinarità; esso è il punto d'incontro tra informatica e scienze naturali, ma non solo. Gli algoritmi *nature-inspired* trovano applicazione anche in altre discipline: Brabazon e O'Neill in *Natural Computing in Computational Finance* [3] offrono una dettagliata analisi dell'applicazione di algoritmi *nature-inspired* in ambito finanziario.

E non è solo l'informatica ad imparare dalle scienze naturali, è vero anche il viceversa. Il NC, infatti, fornisce utili strumenti per una maggiore comprensione della natura, tramite la simulazione computerizzata di fenomeni naturali. Naturalmente, molti degli approcci di ricerca nel NC sono caratterizzati da un certo grado di semplificazione. Più si aggiungono dettagli e florilegi, più il costo computazionale aumenta.

2.1.1 Saga degli algoritmi *nature-inspired*

Andando ad approfondire la branca di ricerca del *nature-inspired* computing, s'illustriano i vari protagonisti che compongono la raccolta di algoritmi ispirati alla natura più pertinenti ed evocativi per questa tesi.

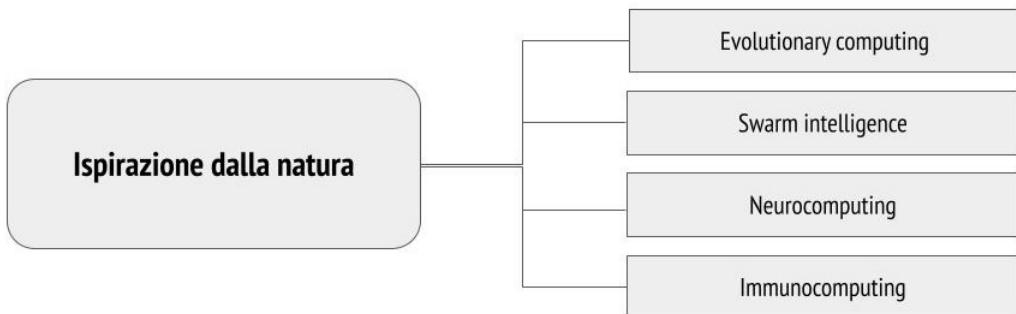


Figura 2.2: Tassonomia degli algoritmi *nature-inspired* secondo De Castro.

Evolutionary Computing

L'*Evolutionary Computing* (EC) è un'area di ricerca che trae ispirazione dai processi dell'evoluzione naturale e da cui deriva la famiglia degli algoritmi evolutivi. Non è sorprendente il fatto che sia stata scelta proprio l'evoluzione delle specie come fonte d'ispirazione: la potenza dell'evoluzione naturale è evidente nella diversità delle specie che abitano il pianeta, ognuna adattata a sopravvivere nella sua nicchia. La metafora sottesa nell'EC fa riferimento ad un particolare paradigma di *problem solving*: "trial-and-error", l'apprendimento per tentativi. [4] La maggior parte degli algoritmi evoluzionari mette le proprie radici proprio sulla teoria darwiniana dell'evoluzione. Darwin propose la teoria che una popolazione di individui capaci di riprodursi e soggetti a variazione genetica seguita da selezione sarebbe risultata in nuove popolazioni con un *fit* maggiore rispetto al loro ambiente circostante.

Per carpire il meccanismo degli algoritmi evolutivi consideriamo l'evoluzione naturale come segue. Un dato ambiente è riempito con una popolazione di individui che s'impegna per la sopravvivenza e la riproduzione. Il *fitness*, la capacità di adattamento degli individui, è determinato dall'ambiente e fa

riferimento alla capacità degli individui di raggiungere i loro obiettivi. In altre parole, rappresenta le loro chance di sopravvivenza e di moltiplicazione. Ricordandoci che siamo in un contesto di "trial-and-error", la collezione di possibili soluzioni sono proprio gli individui. Le loro qualità in termini di adattamento determinano le chance di essere mantenute e tramandate nella generazione di altre possibili soluzioni. La correlazione tra evoluzione e *problem solving* è lampante: l'ambiente rappresenta il problema, gli individui le possibili soluzioni mentre il fitness la qualità di queste soluzioni. [2] Darwin suggerisce quindi un semplice algoritmo basato sulla riproduzione, variazione e selezione naturale, e con questo riesce a riprodurre forme di vita complessa.

L'evoluzione quindi può essere vista come un processo algoritmico che permette la sopravvivenza degli organismi che si adattano meglio all'ambiente tramite una popolazione di individui che si riproduce con ereditarietà, variazione genetica e selezione naturale in un globale processo di adattamento. Un algoritmo evolutivo viene generalmente proposto con i seguenti passaggi:

Algorithm 1 Standard evolutionary algorithm

- 1: Initialise Population
 - 2: **while** Termination Criteria NOT MET **do**
 - 3: Selection
 - 4: Reproduction
 - 5: Replacement
 - 6: **end while**
-

L'algoritmo 1 rappresenta la struttura di base di un algoritmo evolutivo standard. La maggior parte degli algoritmi evolutivi può essere implementata utilizzando questa base con alcune differenze per quanto riguarda la selezione, riproduzione, variazione e l'ordine di questi operatori. Il criterio di uscita è in genere il massimo numero di generazioni, o un obiettivo prefissato.

Fanno parte della categoria degli algoritmi evolutivi gli algoritmi genetici, strategie evolutive, programmazione evolutiva, programmazione genetica. No-

nostante ci siano delle differenze tra i diversi approcci, hanno tutti in comune le caratteristiche di base della teoria darwiniana dell'evoluzione.

Gli **algoritmi genetici** sono i più noti tra gli algoritmi evoluzionari, utilizzano un vocabolario preso in prestito dalla genetica (gli operatori genetici sono *crossover*, *mutation*, *recombination*) e mimano il processo evolutivo delle specie biologiche per risolvere principalmente problemi di ottimizzazione.

[2] Negli algoritmi genetici, le soluzioni sono presentate come stringhe binarie composte da sequenze di 0 e 1 di lunghezza fissa che ricordano infatti un filamento di DNA, costituito da sequenze di A, C, G e T dalle quali è possibile ricostruire un essere vivente.

Nello Pseudocodice 2 di un algoritmo genetico standard si notano delle similità con lo pseudocodice per gli algoritmi evoluzionari.

Algorithm 2 Standard genetic algorithm

- 1: Initialise Population
 - 2: **while** Termination Criteria NOT MET **do**
 - 3: Selection
 - 4: Crossover
 - 5: Mutation
 - 6: **end while**
-

Swarm Intelligence

La *swarm intelligence* si riferisce all'intelligenza dello sciame. Il suo obiettivo è di sfruttare il principio, osservato nelle formiche, nei branchi di animali e negli stormi di uccelli, secondo cui dall'insieme di agenti di capacità ridotte emergono comportamenti intelligenti. Questa categoria di algoritmi ispirati alla natura prende spunto da gruppi della stessa specie, mettendo in risalto la loro orchestrazione per attività come la ricerca di cibo, la divisione del lavoro, la caccia e difesa dai predatori. La *swarm intelligence* è stata coniata nella fine degli anni 80 per fare riferimento alla proprietà dei sistemi formati da agenti con limitate capacità individuali che, collettivamente, esibiscono un comportamento intelligente. In altre parole la *swarm intelligence* è ogni

tentativo di progettazione di algoritmi ispirato al comportamento collettivo di insetti o di società di animali. A questo punto è noto che per *swarm intelligence* ci si riferisce allo sciame quando si hanno agenti che interagiscono tra loro in una collezione strutturata. Un esempio di sciame è lo sciame d'api, anche se la metafora dello sciame può essere estesa anche ad altri sistemi con una simile architettura. La colonia di formiche può essere vista come uno sciame i cui agenti individuali sono le formiche, uno stormo di uccelli rispettivamente è uno sciame i cui agenti sono uccelli, il traffico di punta in una metropoli uno sciame di macchine, una folla uno sciame di persone, un sistema immunitario uno sciame di cellule e molecole, un'economia uno sciame di agenti economici. È importante puntualizzare che per sciame non è interessante solamente il movimento collettivo sincronizzato nello spazio, ma anche il comportamento collettivo. Quindi si fa leva sulla capacità di interazione come gruppo, non sulle capacità del singolo in quanto tale. Un individuo viene visto come dotato di capacità d'interazione con altri individui della propria specie e con l'ambiente.

Una caratteristica fondamentale della *swarm intelligence* è la particolare forma di comunicazione indiretta impiegata dagli insetti sociali per coordinare le loro attività mediante la modifica dell'ambiente in cui operano. Questa caratteristica ha un nome preciso, si tratta di stigmergia; su tale meccanismo è incentrato un campo di ricerca di crescente interesse e applicazione. [39] S'intende con stigmergia l'intelligenza di sciame, la forma di intelligenza espressa da gruppi organizzati composti da individui cooperanti tra loro (non per una mera sommatoria delle intelligenze individuali). Il processo per cui il risultato dell'attività di un individuo agisce come stimolo per una successiva attività di un altro individuo.

Particle Swarm Optimization simula lo stormo degli uccelli e il banco di pesci nella ricerca di cibo per la soluzione di problemi continui di ottimizzazione. L'idea principale dietro al PSO è il comportamento delle particelle di uno sciame. Le particelle hanno una posizione nello spazio multidimensionale,

e si scambiano informazioni tra di loro. [5]

Ant colonies

Vi è un certo livello di comunicazione fra le formiche, sufficiente a evitare che il loro movimento sia completamente casuale. Con questa minima comunicazione esse si ricordano reciprocamente che non sono sole, ma che sono impegnate in un lavoro di gruppo. Per qualsiasi attività, come ad esempio per formare una colonna, sono sempre necessarie innumerevoli formiche tenute insieme da questa minima comunicazione reciproca. [6]

Questo passaggio è tratto da *Gödel, Escher, Bach* di Douglas Hofstader, il quale nel suo tomo cita spesso le formiche come sistema a cui fare riferimento quando si ritrova a spiegare i meccanismi della mente umana.

Le formiche sono un esempio fragrante di *swarm intelligence*. Quante volte ci si ritrova a lottare contro il loro algoritmo che le porta a fare intrusione tra le briciole della nostra cucina o nei posti più disparati.

Le formiche sono state studiate sotto vari aspetti, uno dei quali è la ricerca del cibo. Uno dei più esemplari algoritmi *nature-inspired* è l'*Ant Colony Optimization* (ACO), e ha come scopo problemi di ottimizzazione discreta. Dorigo, nel suo libro dedicato interamente alle formiche *Ant Colony Optimization* [7] illustra che le capacità visive sono solamente sviluppate in maniera rudimentale in alcune specie di formiche, per il resto sono completamente cieche. Questo fa capire che il comportamento tra formiche e con l'ambiente avviene tramite le sostanze chimiche rilasciate dalle formiche. Il materiale chimico si chiama feromone, la sua etimologia significa portare ormone. È il nome che viene dato alle sostanze specifiche emesse da organismi animali che, captate da altri della stessa specie, permettono il riconoscimento intra-specifico e sessuale degli individui, la trasmissione dei segnali d'allarme, e così via. [39] Ed è proprio il feromone che le formiche rilasciano sul suolo durante il loro passaggio in determinati casi, come nel cammino da una fonte di cibo al nido. Le altre formiche, sentendo il feromone si limiteranno a

seguirlo per raggiungere il cibo scoperto dalle loro compagne, rilasciandone ancora per avvisare altre formiche. Un processo collettivo in cui viene segnato un percorso e questo collettivamente viene seguito è l'idea principale alla base dell'algoritmo, che in versioni più dettagliate arriva a simulare l'evaporazione del feromone stesso dopo una soglia di tempo. Da questo algoritmo sono state derivate meta-euristiche per risolvere problemi di ottimizzazione combinatoria.

Andando ad analizzare più nel dettaglio una tecnica probabilistica dell'ACO, sappiamo che le formiche alla ricerca di cibo iniziano esplorando casualmente l'ambiente. Quando trovano delle risorse allora tornano alla colonia rilasciando sul suolo feromone nel percorso per il cibo. Il feromone aumenta la probabilità di indirizzare altre formiche a prendere lo stesso percorso per arrivare all'obiettivo. La sostanza è anche soggetta ad evaporazione, ciò significa che dopo un breve periodo di tempo le vecchie tracce cominciano ad evaporare nell'aria.

A una successiva iterazione esse tenderanno a seguire i percorsi già marcati, ovvero che assicurano di raggiungere il cibo. Tuttavia come già detto i feromoni evaporano dopo un certo lasso di tempo, quindi più breve è il percorso, più intensa è la traccia. Dopo un adeguato numero di iterazioni le formiche tenderanno a seguire unicamente il percorso più breve, reso attraente da una concentrazione di feromone più alta.

Il risultato finale dato dal rafforzamento della traccia feromonica è lo *shortest path* che viene preferito dall'intera colonia rispetto ai percorsi più lunghi, che poi evaporano.

Riassumendo l'ant colony optimization nello Pseudocodice 3 abbiamo i seguenti passaggi:

Algorithm 3 Ant Colony Optimization

```
1: procedure ANT COLONY OPTIMIZATION(ants)
2:   while Casual exploration of the environment do
3:     if An ant finds food then
4:       Ant comes back to the colony leaving a pheromone trace
5:     end if
6:   end while
7:   Other ants follow the pheromone trace by probability  $p$ 
8:   The pheromone trace becomes shorter and stronger
9:   The pheromone trace becomes the shortest path
10:  Return The shortest path dicovered by the ant colony
11: end procedure
```

La storia delle formiche raccontata in termini computazionali continua con altri due algoritmi altrettanto avvincenti; sono stati effettuati una serie di esperimenti che hanno portato alla luce un aspetto di organizzazione dei cimiteri da parte delle formiche. È stato osservato che mettendo in gruppi di formiche delle formiche morte, queste dopo qualche ora venivano raggruppate tra di loro, clusterizzate in un cimitero di formiche. Da questi esperimenti è stato derivato un algoritmo di *clustering* che lavora per similarità su una griglia bidimensionale.

Grey wolf optimizer Questo è un algoritmo che simula il comportamento dei lupi grigi durante la caccia. I lupi grigi vivono in gruppi con una stretta gerarchia tra di loro. La loro strategia di caccia è formata da tre fasi: circondare la preda, ruotarla attorno e cacciarla. Questo processo è stato adattato per la soluzione di problemi di ottimizzazione. [8]

Bat algorithm Come le formiche, anche i pipistrelli non sono dotati della vista. Il loro metodo per orientarsi negli spostamenti è l'ecolocazione: la capacità di emettere ultrasuoni e rilevare gli echi riflessi per localizzare la posizione di ostacoli o fonti di cibo. Dalla loro abilità di ecolocazione è stato

creato il *bat algorithm*. [9] L'algoritmo si basa appunto sull'abilità di individuare oggetti nel loro ambiente dalla percezione dell'eco di questi oggetti. Comportamento che dipende anche dal variare della pulsione dell'emissione e del volume circostante.

Neurocomputing e Immunocomputing

In questa categoria appartengono le reti neurali che si ispirano alla neuroanatomia. Sebbene le cose da dire siano molte non verranno trattate in questo contesto e si rimanda al testo principale per eventuali approfondimenti.

[2]

Abbiamo quindi visto come questi algoritmi facciano leva sulle proprietà uniche delle specie più varie, prese in singolo o nella loro forza di gruppo; caratteristiche che hanno sviluppato durante l'evoluzione per necessità di adattamento volendo citare Darwin o simulando ex novo un'evoluzione, prendendo alla lettera Darwin. Ora ci spostiamo nel campo della botanica, addentrandoci in una classe di algoritmi che più si avvicina allo scopo di questa tesi, pur rimanendo distante da quella che è l'intenzione.

2.1.2 Botanica e descrizione della *Daucus Carota*

Spostiamo gradualmente il focus dagli animali più disparati alle piante. Facciamo un passo avanti e introduciamo il soggetto misterioso che si cela dietro la parola "fiore" del titolo di questa tesi: la *Daucus Carota*.



Figura 2.3: Illustrazione di *Daucus Carota*, Medical Botany (1832) di William Woodville.

Botanicamente parlando, appartiene alla famiglia delle ombrellifere (o anticamente dette *apiaceae*), una famiglia vasta che si distingue sia per le caratteristiche infiorescenze a forma di ombrello, sia per le particolari sostanze prodotte che si manifestano nel sapore, nell'odore o anche nella tossicità di molti dei suoi membri. [10] È una pianta erbacea annuale, alta dai 50 cm fino ai 2 m, con ombrelle a molti raggi, fiori per lo più bianchi, spontanea e

nota comunemente come carota selvatica; infatti è la progenitrice della carota.

La pianta di *Daucus Carota* è stata scelta appunto per la sua particolare e ricorrente infiorescenza: i singoli peduncoli fiorali si originano alla stessa altezza sull'asse principale, ma hanno lunghezze diverse in modo da portare tutti i fiori allo stesso livello. Questa è una caratteristica importante per mettere le basi di uno studio algoritmico. L'ombrella possiede ramificazioni formate a loro volta da tante ombrelle semplici, dette ombrellette, che lo rendono un fiore canonicamente frattale. Il fiore delle ombrelle è quasi costantemente formato da cinque petali.

La *Daucus Carota* come altre numerose specie di ombrellifere erano conosciute fin dalle antiche civiltà. Si pensi che veniva usata fin dai tempi dei Faraoni essenzialmente come aromatizzante o come pianta medicinale per problemi digestivi o per favorire il concepimento, mentre Ippocrate la consigliava per i problemi respiratori. Si trovano tracce di *Daucus* fin da Teofrasto, che duemila anni fa nel trattato di storia naturale *Historia Plantarum* descrive molte varietà di piante, alberi e le loro caratteristiche. Per quanto riguarda il *Daucus*, oltre alle suddette caratteristiche, Teofrasto sosteneva che tenerne dei semi accanto al letto era foriero di dolci sogni, per il loro delicato profumo.

Non è una pianta nota soltanto nel bacino Mediterraneo, bensì in America. Nel 1914 Ada Georgia nel manuale delle piante erbacee *Manual of Weeds* cataloga e illustra le piante più diffuse nel continente nord-americano; nella famiglia delle ombrellifere ritroviamo la *Daucus Carota*.

Flowers clustered, in large, flat, compound umbels, white, except that there is usually one in the center of each umbel which is dark purple; rays of the umbel crowded, the inner ones shorter than the outer rows, all subtended by a whorl of green, finely cut, involucral bracts. [11]

Estremamente diffuso anche nel Nord America, la descrizione singolare che emerge è la clusterizzazione dei fiori in ombrelle larghe, piatte, composte e

bianche. Per la sua struttura è una pianta che ha affascinato molte persone. Viene chiamato anche *Queen Anne's Lace* perché sembra essere stato il fiore preferito della regina Anna di Danimarca e Scozia, tanto che era diventato il pattern preferito per le decorazioni dei suoi merletti. La leggenda continua dicendo che la regina si fosse punta con l'ago durante la cucitura di uno dei suoi merletti, ed ecco perché al centro di questo fiore spesso troviamo un fiore rosso scuro. Viene chiamato anche *Bird's Nest Weed* per la forma che prende quando la sera i rami rientrano verso il centro rendendo l'ombrella concava e facendolo sembrare proprio un nido d'uccello.

La carota è comunissima in tutti i prati e luoghi erbosi, si di collina che di pianura d'ogni provincia dell'Europa, della Crimea e del Caucaso, da dove passò nella China, e nell'America. In tutta l'Italia è pianta volgarissima, ed anche abbonda in moltissimi luoghi della Toscana, cosicché questo vegetale non ha fatto che passare dai luoghi salvatici ed incolti, negli orti, dove addomesticandosi, ha prodotto le molte varietà ora coltivate che se ne conoscono da molli anni indietro. [12]

Leggende a parte, la *Daucus Carota* nei fini pratici è una pianta molto diffusa perché estremamente infestante. A fine fioritura, l'intera ombrella trasforma i suoi fiori in semi resistenti, che diventeranno altre piante l'anno successivo. Un solo seme di *Daucus Carota* può infestare un campo per molti anni. Nel libro *Che cosa raccontano le erbe infestanti* [13] che fornisce consigli su come gestire in maniera casalinga le piante infestanti, Pfeiffer dice che la *Daucus Carota* indica la presenza di un suolo fertile, proporzionalmente alla grandezza della pianta. Ed è proprio da questa pianta semplice e diffusa che nasce l'ispirazione per questa tesi. Nel Capitolo 3 ci arrampicheremo metaoricamente sulla sua struttura trasformando i fiori in nodi di un network.

2.1.3 Invasive Weed Optimization

L'approccio che più si avvicina all'idea di questa tesi è stato trovato nel paper *Automatic clustering using nature-inspired metaheuristics*, [14] un sondaggio in cui vengono descritte e analizzate varie tecniche di clustering *nature-inspired*. Oltre alle varianti di clustering degli algoritmi descritti in precedenza, è stato introdotto l'*Invasive Weed Optimization* (IWO).

IWO è una tecnica metaeuristica relativamente recente proposta da Mehrabian e Lucas [15] che mima il comportamento delle colonie di erbe infestanti, la cui crescita vigorosa e specialmente invasiva rappresenta una seria minaccia per le piante coltivate, rendendole un pericolo per la produzione agricola. Per definizione, una pianta infestante è una pianta che cresce in un luogo in cui non è voluta. Un albero, un cespuglio o una pianta erbacea può essere qualificata come infestante, dipende dalla situazione; generalmente il termine viene riservato alle piante che sono vigorose, che hanno abitudini invasive di crescita e che pongono una seria minaccia per le piante che si desidera coltivare. Le erbe infestanti si riproducono rapidamente e così aumenta velocemente la loro popolazione. Il loro comportamento cambia col tempo: la colonia si addensa lasciando poche chance di sopravvivenza alle altre piante che non riescono ad adattarsi ad un ambiente ormai diventato ostile.

L'algoritmo IWO simula quindi il comportamento delle erbe infestanti quando colonizzano e trovano un posto adatto per la crescita e la riproduzione. In questo algoritmo vengono utilizzati i concetti base di semina, crescita e competizione, concetti prevalenti nelle piante infestanti e che mostrano che IWO è efficace nel convergere ad una soluzione ottima.

Il comportamento colonizzatore delle erbe infestanti viene simulato delineando le seguenti procedure:

- **Inizializzazione della popolazione:** questo passaggio ricorda gli algoritmi evolutivi; un insieme di soluzioni iniziali vengono disperse casualmente sullo spazio dimensionale di ricerca di d dimensioni. Queste

soluzioni iniziali sono analoghe ai semi nella colonia che rappresentano la prima generazione della popolazione.

- **Riproduzione:** una pianta membro della popolazione produce semi rispettivamente alla propria capacità di adattamento (*fitness*) in quanto individuo della popolazione. Più alto il fitness, maggiore il numero di semi prodotti. Questo tipo di riproduzione fornisce un'opportunità di crescita della popolazione.
- **Dispersione nello spazio:** i semi vengono distribuiti casualmente lungo lo spazio dimensionale di ricerca di d dimensioni da numeri casuali che hanno una distribuzione normale.
- **Esclusione competitiva:** una certa dose di competizione deve esistere per limitare il numero di piante in una colonia. Inizialmente tutte le piante si riproducono in velocità e tutte le piante vengono introdotte nella colonia, fino a che il numero di piante nella colonia raggiunge un valore massimo. D'ora in poi solo le piante con maggior livello di fit della colonia verranno selezionate per la generazione successiva.

Il primo algoritmo di clustering automatico basato su IWO è stato proposto da Chowdhury. [16] In questa procedura chiamata IWO clustering, è stata utilizzata una codifica della variabile lunghezza per denotare le stringhe di erbacce: in questo modo, ogni erba nella popolazione decodifica i centroidi del cluster. In una recente proposta di Razavi-Far e Palade [17] IWO è stato utilizzato in problemi di *pattern recognition* e per classificazione multiclasse.

È interessante trovare dei paper estremamente recenti e nello stesso ambito di questa tesi: c'è molto da imparare dalla natura e anche dalle piante infestanti.

2.2 Frattali

In questa sezione si transita dall'ultima ramificazione del NC, la sintesi e simulazione di fenomeni naturali, andando ad approfondire il tema degli oggetti frattali descrivendo le loro proprietà matematiche.

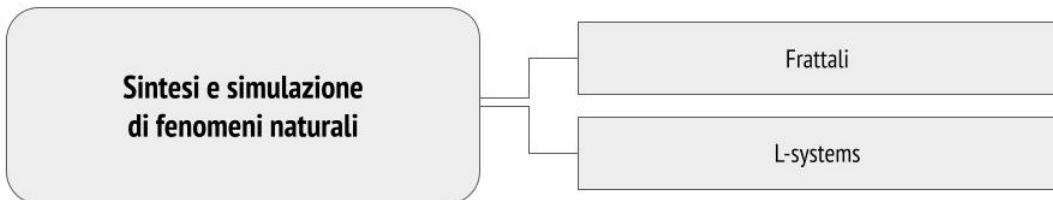


Figura 2.4: La tassonomia della sintesi e simulazione dei fenomeni naturali secondo De Castro.

2.2.1 Gli oggetti frattali

Perché la geometria viene spesso descritta come fredda e arida?

Una delle ragioni sta nella sua incapacità di descrivere la forma di una nuvola, di una montagna, di una costa o di un albero. Le nuvole non sono sfere, le montagne non sono coni, le coste non sono cerchi e le corteccce non sono lisce, nemmeno la luce viaggia in linea retta. L'esistenza di questi pattern ci lancia una sfida per studiare queste forme che Euclide aveva tralasciato definendole senza forma, per investigare la morfologia dell'amorfo. [18]

Nel 1982 Mandelbrot ha coniato il termine frattale per identificare una famiglia di forme che descrivono i pattern irregolari e frammentati che si trovano in natura, differenziandole dalle forme puramente geometriche euclidee. [19] La parola deriva dal latino *fractus*, corrispondente al verbo *frangere* che significa rompere, creare frammenti irregolari. La geometria frattale è la geometria delle forme irregolari che si trovano in natura, caratterizzate in generale da: infiniti dettagli e lunghezza, *self-similarity*, una determinata dimensione frattale e l'assenza di una derivata.

I frattali sono caratterizzati da un'apparente irregolarità: hanno lo stesso grado di irregolarità su tutte le scale. I frattali sono la stessa figura se vista da vicino o da lontano: sono *self-similar*. La natura presenta molti oggetti frattali: le coste, le catene montuose, i broccoli romani, gli alberi di sicomoro e le altre piante in cui ogni ramificazione corrisponde al tutto e la cui struttura ramificata viene ripetuta su molteplici scale. Lo stesso sistema vascolare di una foglia è frattale, come il sistema venoso e arterioso di ogni essere umano. In ognuno di questi casi, ogni piccola parte dell'oggetto è un'immagine ridotta dell'oggetto intero.

Self-similarity I frattali sono quindi oggetti matematici che hanno la proprietà di essere *self-similar*, ciò significa che sono composti da copie più piccole di loro stessi, in italiano definita proprietà di autosomiglianza. Un classico esempio di frattale si trova nella foglia di felce e nelle parti di cui è composta la foglia. Sono composte da copie più piccole di loro stesse, questa *self-similarity* si estende su scale di grandezza diverse. Ma se per anni i frattali non sono stati accettati all'unanimità dalla comunità scientifica, una delle ragioni è stata la loro dimensione. [18] Venivano definiti mostri matematici dallo stesso Mandelbrot appunto per la loro dimensione, spesso non intera. Lui stesso ha raggruppato i frattali come la curva di Koch, l'insieme di Cantor, la curva di Peano e il triangolo di Sierpiński all'interno di una Galleria di Mostri, e la decisione di nominarli frattali nasce dalla necessità di dare un nome a questi mostri matematici, i quali una volta ricevuto un nome, hanno acquistato esistenza: *nome est noumen* per queste forme di ordine dentro il caos. [19]

Da un punto di vista intuitivo, la *self-similarity dimension* per un oggetto frattale è la quantità di copie più piccole che stanno nella copia più grande. Per definirla in maniera più formale c'è bisogno della nozione di *magnification factor*, il fattore d'ingrandimento che permette di ingrandire una copia più piccola allungandola in una o più direzioni.

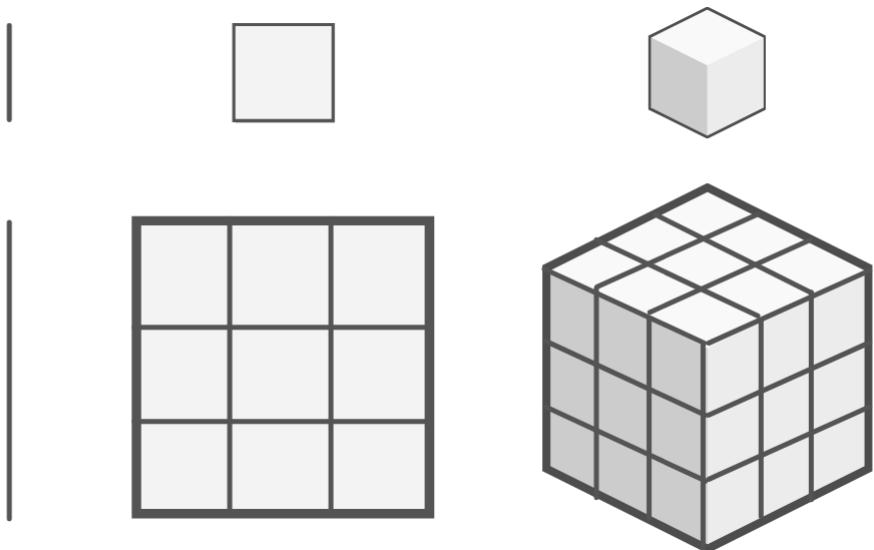


Figura 2.5: Linea, quadrato e cubo con *magnification factor* pari a tre.

Se per esempio una linea di lunghezza uno viene scalata per un fattore d'ingrandimento tre, la linea risultante avrà lunghezza tre. Se, come rappresentato nella Figura 2.5, un quadrato di area uno viene scalato per un fattore d'ingrandimento tre, il quadrato risultante sarà composto da nove quadrati, in quanto oggetto bidimensionale l'allungamento viene fatto in due dimensioni. Per un cubo tridimensionale, il *magnification factor* pari a tre porta ad un cubo composto da 27 cubi, come riassunto in Tabella 2.1.

Forma	Magnification factor	N. di copie piccole in copia grande	Dimensione
Linea	3	$3 = 3^1$	1
Quadrato	3	$9 = 3^2$	2
Cubo	3	$27 = 3^3$	3

Tabella 2.1: Calcolo della dimensione per linea, quadrato e cubo.

Il numero di copie presenti all'interno di un oggetto frattale si calcola quindi con la seguente formula:

$$\text{Number of small copies} = (\text{magnification factor})^D$$

Dove l'esponente D rappresenta la *self-similarity dimension*.

Esempi di oggetti frattali La curva di Koch è un frattale costruito come nella Figura 2.6. Ad ogni iterazione, ogni segmento viene sostituito con quattro

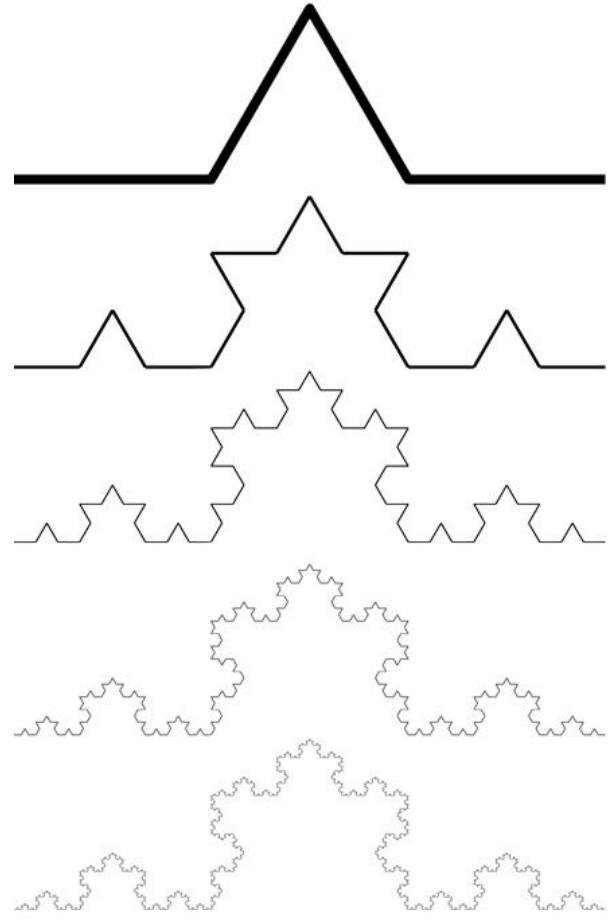


Figura 2.6: Iterazioni per costruzione della curva di Koch.

segmenti, ognuno dei quali è $1/3$ della lunghezza del segmento originale. La *self-similarity dimension* della curva di Koch corrisponde a $\log(4)/\log(3)$. Inoltre, un'altra proprietà interessante della curva di Koch è che la lunghezza è infinita, mentre l'area è finita. Gli oggetti frattali sono quindi una combinazione tra finito e infinito come si evince dal seguente passaggio di Mandelbrot:

La questione del finito e infinito è davvero al centro della geometria frattale. Si parte dalla formula che non solo è finita, ma anche estremamente corta, che, se ben scritta, occupa solo una riga di programma. [...] Questa formula molto lunga e complicata ha una sua struttura, ha un nucleo che è molto semplice, il resto è solo

ripetizione. [19]

Il triangolo di Sierpiński è un altro esempio lampante di oggetto frattale. Chiamato così per il matematico polacco che nel 1915 lo descrisse, la sua costruzione viene descritta nella Figura 2.7.

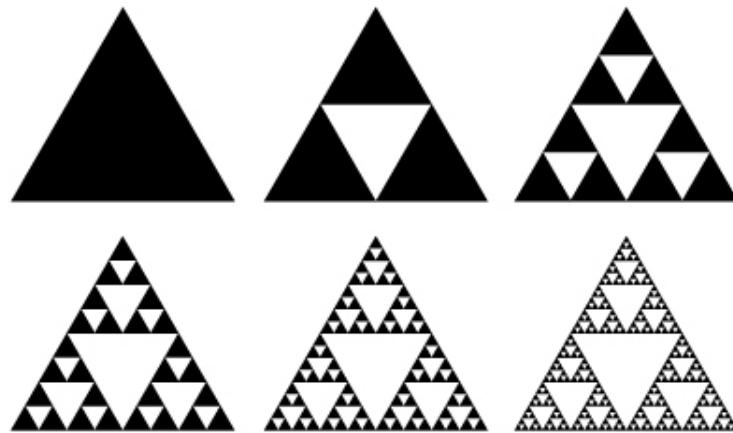


Figura 2.7: Iterazioni per costruzione del Triangolo di Sierpiński.

Per la costruzione del triangolo di Sierpiński si parte da un triangolo equilatero con lati di lunghezza uno. Nella prima iterazione si congiungono i punti di ciascun lato individuando quattro triangoli di lato $1/2$, di cui tre orientati nella stessa direzione di quello di partenza mentre uno capovolto, suddividendo quindi il triangolo principale in quattro triangoli equilateri più piccoli congruenti e rimuovendone quello centrale. All'iterazione successiva si ripete l'operazione precedente su ciascuno dei tre triangoli non capovolti, ottenendo così nove triangoli non capovolti di lato $1/4$. E così via, *ad infinitum*. Quando n tende ad infinito, l'area del triangolo di Sierpiński tende a zero. Per quanto concerne il perimetro, quando n tende ad numeri infinitamente grandi, il perimetro tende anch'esso ad infinito.

La dimensione corrisponde invece a $\log(3)/\log(2) = 1,585$. Che con qualche passaggio algebrico si può esprimere come:

$$2^D = 2^{1,585} \simeq 3$$

Ogni volta che si ingrandisce per un fattore di due, la massa o il volume

della forma diventa all'incirca tre. Riassumendo, la *self-similarity* è una proprietà cruciale per ogni oggetto frattale, un oggetto composto da copie più piccole di se stesso e uguale su diverse scale. Nella teoria matematica, più s'ingrandisce un oggetto frattale più si continua a vedere la stessa forma ripetersi.

Quando si rappresenta un fenomeno reale si fa un modello, poi lo si corregge ancora, lo si migliora costantemente, ma il modello in sé stesso non va mai confuso con la cosa reale, la realtà è sempre più complicata del modello. [19]

La natura che ci circonda non è composta da oggetti frattali, bensì da oggetti *fractal-like*, similmente frattali. In questa importante distinzione è giusto notare che un oggetto matematico frattale si ripete su tutte le scale all'infinito. Se avessimo la capacità computazionale di ingrandire un triangolo di Sierpiński all'infinito, vedremmo la stessa figura ripetersi, uguale a quella di partenza, ancora ed ancora. Per esempio, il corrispondente frattale di una costa frastagliata dell'Inghilterra viene rappresentato dalla curva di Koch, la cui *self-similarity* si ripete su una quantità limitata di scale, non infinita. Più formalmente quindi la nozione di frattale matematico consiste nell'astrazione, un ideale. Esiste nel mondo della matematica ma non esattamente nella realtà, e ciò vale per ogni geometria. In un certo senso, la dimensione che assume un valore compreso tra uno e due sta a significare che l'oggetto frattale in questione presenta proprietà di entrambe le dimensioni, che è sia unidimensionale che bidimensionale.

2.2.2 Generare frattali

Gli oggetti frattali sono quindi oggetti autosomiglianti attraverso molte scale, non sono oggetti descritti con la classica geometria euclidea, hanno una dimensione *self-similar* che è più grande della dimensione topologica. La dimensione topologica può essere descritta come la dimensione intuitiva; per esempio la curva di Koch essendo una linea la sua dimensione topologica è

pari a uno, mentre la dimensione *self-similar* è un valore tra uno e due.

Vengono ora presentati i meccanismi per generare frattali, meccanismi che fanno riferimento a semplici regole iterative che sono capaci di rappresentare interessanti e intriganti frattali.

Esistono due metodi usati nel campo di ricerca della computer graphic per la formalizzazione e rappresentazione degli oggetti frattali: Iterated Function Systems e L-systems. In ciò che segue verranno descritti brevemente entrambi.

Iterated Function Systems Sfruttano la matematica delle matrici per fare trasformazioni geometriche ripetute. Ogni trasformazione può essere descritta come un'operazione su matrici, come ingrandire, ridurre o ruotare.

L-systems Gli *L-systems* fanno parte di quella corrente del NC che simula graficamente fenomeni o eventi naturali. Sono un altro formalismo per rappresentare frattali, basato su un sistema di riscrittura di stringhe con un metodo algoritmico e flessibile per generare frattali realistici.

Il libro *The Algorithmic Beauty of Plants* [20] di Prusinkiewicz e Lindenmayer si focalizza appunto sulle simulazioni grafiche di certi pattern naturali. Vengono utilizzate le grammatiche di Chomsky e i sistemi di linguaggio formali per sviluppare *L-systems*, che prendono nome proprio dal loro creatore Aristid Lindenmayer, con il concetto di sostituire simboli secondo delle regole di produzione. In questo esempio le seguenti regole di produzione formalizzano la costruzione della curva di Koch:

Assioma : F

$p_1 : F \rightarrow F + F - -F + F$

$t_1 : + \rightarrow +$

$t_2 : - \rightarrow -$

Parametro : 60°

Tra i vari esempi di simulazioni grafiche presenti nel libro, trova posto anche il modello della *Daucus Carota*, di cui vengono descritte le varie componenti per una rappresentazione virtuale.

Come già visto, l'infiorescenza è caratterizzata da più internodi collegati ad un singolo nodo, che risulta nella tipica forma ad ombrella, dove il pattern ramificato si ripete ricorsivamente un determinato numero di volte. Il sistema L-system parziale per un'ombrella composta da una profondità di ricorsione pari a due è il seguente:

Assioma : A

$$p_1 : A \rightarrow I[IB]^k B$$

$$p_2 : B \rightarrow I[IC]^l C$$

$$p_3 : C \rightarrow I[IK]^m$$

Questa serie di regole di produzione porta al tipo di infiorescenza comunemente trovato nella famiglia delle ombrellifere, di cui la *Daucus Carota* ne è un esempio. Gli L-systems sono quindi formalismi per simulare lo sviluppo di organismi pluricellulari che generano immagini realistiche a partire da semplici regole geometriche. Immagini che sembrano complesse ma sono semplici da costruire, e che vengono utilizzate in diverse applicazioni principalmente decorative come *landscape design*, design ornamentale o illustrazioni botaniche. L'utilità degli L-systems si estende oltre lo scopo decorativo, come la ricostruzione di specie di piante estinte, il design di nuovi tipi di piante, identificazioni di risposte delle piante ad attacchi pesticidi, o addirittura come training dataset per una rete neurale, come spiegheremo nella Sezione 2.3.

Box-counting dimension Esponiamo ora un metodo per il calcolo della dimensione frattale diverso dalla *self-similarity dimension*: il *box counting*. È un metodo flessibile e performante che può essere applicato a forme e contesti diversi. La sua concretezza e facilità di applicazione dipendono dal suo

semplice utilizzo: consiste nel sovrapporre all'immagine dell'oggetto di cui si vuole calcolare la dimensione frattale una griglia quadrata e successivamente contare il numero di box utilizzati per riempire la figura al variare della dimensione della griglia. La sua flessibilità dipende anche dal fatto che sia possibile calcolare la dimensione di oggetti che non sono propriamente *self-similar*: gli oggetti *fractal-like*, i casi più disordinati, più reali.

La definizione di dimensione che si deriva da questo metodo e che si applica ad oggetti *fractal-like* è la seguente:

$$N(s) = c * \left(\frac{1}{s}\right)^D$$

Dove si assume che $c = 1$, mentre s tende a zero. L'equazione è vera solo se s assume valori sempre più piccoli, ciò significa che la dimensione dei quadrati nella griglia deve essere piccola, tendente a zero. Per valori grandi di s questa equazione non è valida. È opportuno notare che con questo metodo non si ottiene una misura precisa, dato che una fonte di errore dipende dal posizionamento stesso della griglia per il conteggio dei box. In casi reali quindi, ottenere una stima precisa per la dimensione *box counting* è alquanto difficile. *Box counting* con manipolazioni algebriche diventa:

$$\log(N(s)) = -D \log(s) + \log(c)$$

Che rappresenta l'equazione della linea retta. Si può stampare la dimensione di un oggetto puramente frattale che segue la linea retta, calcolandone quindi la dimensione come la pendenza della retta. Gli oggetti similmente frattali hanno un altro comportamento non propriamente rettilineo, che verrà descritto a breve.

Dimensione frattale del *Daucus Carota* Dopo la presentazione dei due metodi principali per il calcolo della dimensione frattale, *self-similarity dimension* e *box counting*, andiamo a calcolare la dimensione frattale dell'oggetto *fractal-*

like *Daucus Carota*. La pianta infestante precedentemente descritta può essere facilmente classificata come oggetto similmente frattale: la ripetizione di ombrellette all'interno dell'ombrella principale ne è un chiaro esempio di frattalità in natura. La stessa foglia di *Daucus Carota* è stata studiata da Hartvigsen per il calcolo della sua dimensione frattale tramite *box counting*. [21]

Nel suo studio ha calcolato la dimensione frattale delle foglie di *Daucus Carota* analizzando varie foglie lungo lo stelo verticale della pianta. Va sottolineato che la dimensione delle foglie diminuisce leggermente procedendo verticalmente lungo lo stelo. Questo ha portato ad un calcolo della dimensione della foglia risultante in un valore compreso tra 1,25 e 1,50. Essendo però il fiore l'oggetto di maggiore interesse di questa tesi, abbiamo calcolato separatamente la dimensione frattale dell'ombrella della *Daucus Carota* tramite *box counting*.

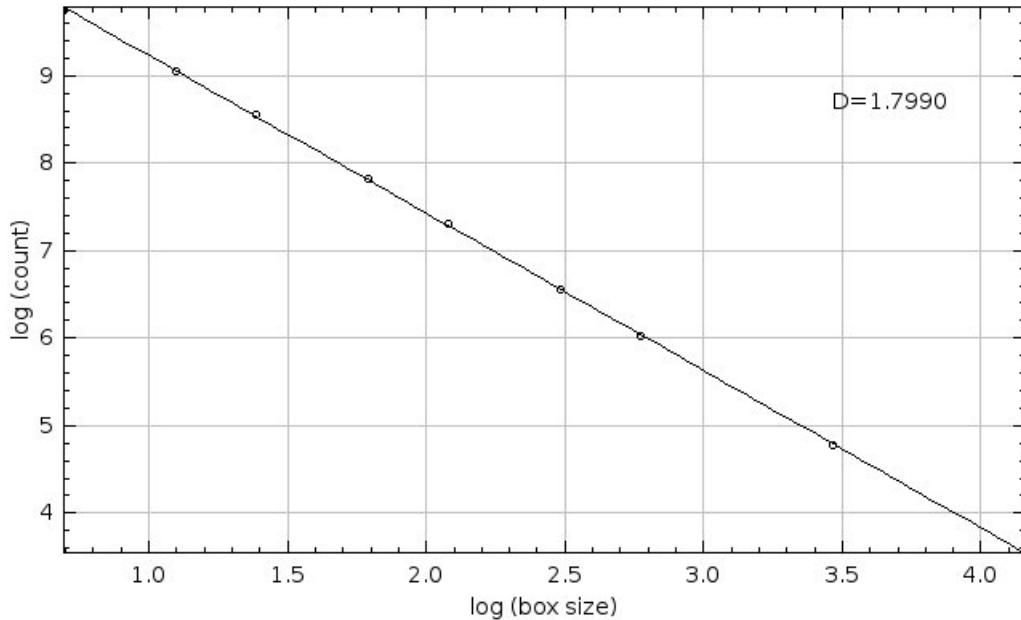


Figura 2.8: Dimensione *Daucus Carota* calcolata tramite box counting.

Il grafico log-log rappresentato nella Figura 2.8 rappresenta sulle ascisse il logaritmo del numero di box contate, sulle ordinate il logaritmo per ogni misurazione effettuata. La pendenza della linea equivale alla dimensione

frattale dell'oggetto. Nel caso in questione, la dimensione frattale del Daucus Carota è circa 1,80.

Power Law Nel metodo del box counting abbiamo quindi ottenuto una funzione che descrive la dimensione basandoci sull'equazione della retta. Con ulteriori passaggi algebrici si ottiene la formula che descrive il numero di box di lato s che servono per coprire una forma simil-frattale:

$$\begin{aligned} N(s) &= c \frac{1}{s}^D \\ &= cs^{-D} \end{aligned}$$

Dove D è il negativo della pendenza della retta del log-log plot. Questo tipo di funzione con questa forma $P(x) = Ax^{-\alpha}$ rappresenta la distribuzione power law. Una power law ha *long tail*, una coda lunga ed è *scale free* o *self-similar*, proprio come i frattali. È una funzione che decade in maniera attenuata, più lentamente di un'esponenziale. A differenza di una funzione esponenziale, se si scala una power law la curva rimane sempre la stessa, mentre per l'esponenziale questa proprietà non vale. Le power law sono le uniche curve ad avere la proprietà di essere *scale free*. In altre parole, se si vede un fenomeno *scale free* allora deve essere espresso da una power law. [40] In conclusione, l'immagine che emerge è che modelli e meccanismi che generano power law o che hanno un comportamento simile ad una power law sono definiti *scale free*.

Questa è una proprietà che verrà trattata anche per le reti, perché la distribuzione dei nodi di una rete si può descrivere con una power law.

2.3 Social Network Analysis

Se finora si è trattato di algoritmi ispirati alla natura, piante ombrellifere infestanti e frattali, presentiamo ora l'oggetto matematico su cui astrarre dal concetto di fiore e fare una sorta di metamorfosi da fiore a rete. Per fornire una base teorica allo sviluppo della procedura riportata nel Capitolo 3, vengono riportati i concetti fondamentali della *Social Network Analysis* (SNA). Innanzitutto viene fornita la rappresentazione delle reti come oggetto matematico facendo quindi riferimento alla teoria dei grafi, poi vengono date varie interpretazioni ed esempi di reti reali e del loro contenuto. Si introducono poi concetti e proprietà fondamentali di una rete, ovvero la grandezza, sparsità, small worlds e le reti *scale free*. Infine, vengono presentate le network embedding e dei lavori che uniscono network e frattali.

Teoria dei grafi Un *grafo* è una struttura discreta G , composta da nodi (altresì *vertices*), e archi (o *edges*) che connettono tra loro questi nodi. Quando tra due vertici esiste un arco i vertici vengono definiti vicini. Esistono diversi tipi di grafi a seconda delle proprietà presenti, come la direzione degli archi. Problemi di quasi ogni disciplina concepibile sono risolvibili grazie alla teoria dei grafi. Si utilizzano modelli di grafi quando si rappresenta una rete di trasporti, di interazioni chimiche o di comunicazione e sociali, e non solo. Dal punto di vista matematico un grafo viene definito come la coppia

$$G = (V, E)$$

che consiste in un insieme non vuoto di nodi $V = \{v_1, v_2, \dots, v_N\}$ e un insieme di archi $E = \{(u, v) | u, v \in V\}$. Ogni arco (u, v) connette tra loro due nodi. I grafi posso contemplare direzionalità o meno, nel primo caso gli archi che connettono i nodi tra loro contengono anche l'informazione della direzione. Dati due nodi $u, v \in V$, vengono chiamati grafi diretti o orientati se gli archi $(u, v) \in E$ si associano ad una coppia ordinata di vertici, ossia quando $(u, v) \neq (v, u)$ e si rappresentano graficamente con una freccia. Nel caso in

cui non è prevista direzionalità, se quindi $(u, v) = (v, u)$, sono *grafo indiretti* o non orientati. In questa tesi per ragioni pratiche è stato scelto di basarsi esclusivamente su grafi indiretti.

Un singolare tipo di grafo in cui ogni vertice è collegato a ciascuno degli altri da un arco viene definito *grafo completo* K_n , dove n è il numero di nodi. Un esempio di grafo completo è il triangolo, che verrà utilizzato come pattern da identificare in fase di sviluppo.

Metriche Alcune delle proprietà fondamentali dei grafi sono il numero di nodi e di archi. Se quindi un nodo è l'oggetto semplice che compone un grafo, allora il *numero totale di nodi*, la cardinalità dell'insieme N contenente i nodi presenti in una rete $\# \text{ nodi} = |N|$, rappresenta il numero di componenti del sistema, la dimensione della rete. Per distinguere un singolo nodo dall'altro, nella pratica in genere si rinominano con numeri interi. Simmetricamente, il *numero totale di archi* di una rete equivale alla cardinalità dell'insieme E : $\# \text{ archi} = |E|$ e rappresenta il numero totale di interazioni tra i nodi. Gli archi vengono raramente rinominati, poiché possono essere identificati attraverso i nodi che essi connettono. Il *grado* di un nodo è il numero di archi che connettono un nodo, il numero di altri nodi a cui esso è connesso. Indichiamo pertanto con k_i il grado di un i -esimo nodo. Un *nodo isolato* è un nodo con grado pari a zero. Partendo dalla nozione di grado di un nodo, possiamo definire il numero totale degli archi di un grafo indiretto come:

$$L = \frac{1}{2} \sum_{i=0}^N k_i$$

Dove il fattore $1/2$ sta a simboleggiare il fatto che nella sommatoria qui sopra ogni arco viene contato due volte, facendo emergere quindi la necessità di dimezzare il risultato finale.

Rete o network La rete o network si riferisce all'applicazione di un grafo in un determinato contesto. Un singolo grafo con la sua conformazione di nodi e archi può rappresentare tipi di reti diversi. Eccone alcuni esempi:

- *World Wide Web*: la topologia del WWW può essere modellata come una rete dove i nodi sono le pagine web mentre i link che rimandano da una pagina all'altra degli archi diretti.
- *Reti sociali*: i vertici di una rete sociale sono le persone, mentre un arco collega due persone se queste si conoscono. Più precisamente il mero fatto di conoscersi può identificarsi in una stretta di mano, nel sapere il nome dell'altra persona, nel salutarsi. Come già si evince dalla descrizione, la difficoltà principale è appunto quella di trovare delle misure coerenti di conoscenza tra persone.
- *Rete di attori*: una rete in cui i nodi sono gli attori e gli archi le collaborazioni lavorative tra di loro. Se due attori hanno recitato assieme in almeno un film, allora sono collegati tra loro tramite un arco.
- *Rete di paper*: un esempio tangibile di network utilizzato per lo studio della letteratura di questa tesi è rappresentato in Figura 2.9 tramite il sito *Connected Papers*. [41] Dopo aver selezionato sul sito una pubblicazione accademica, nel nostro caso *Ant Colony Optimization* di Dorigo, viene restituita una rete in cui ogni nodo rappresenta un paper accademico correlato alla pubblicazione cercata. I paper sono ordinati secondo la similarità tra loro, la grandezza dei nodi è variabile e dipende dal numero di citazioni, mentre il colore dei nodi cambia in base all'anno di pubblicazione.

Analisi svolte in precedenti lavori [42] hanno evidenziato che esistono delle caratteristiche peculiari condivise da buona parte delle reti. Alcune di queste sono: grandezza, sparsità, presenza di small words e *scale free*.

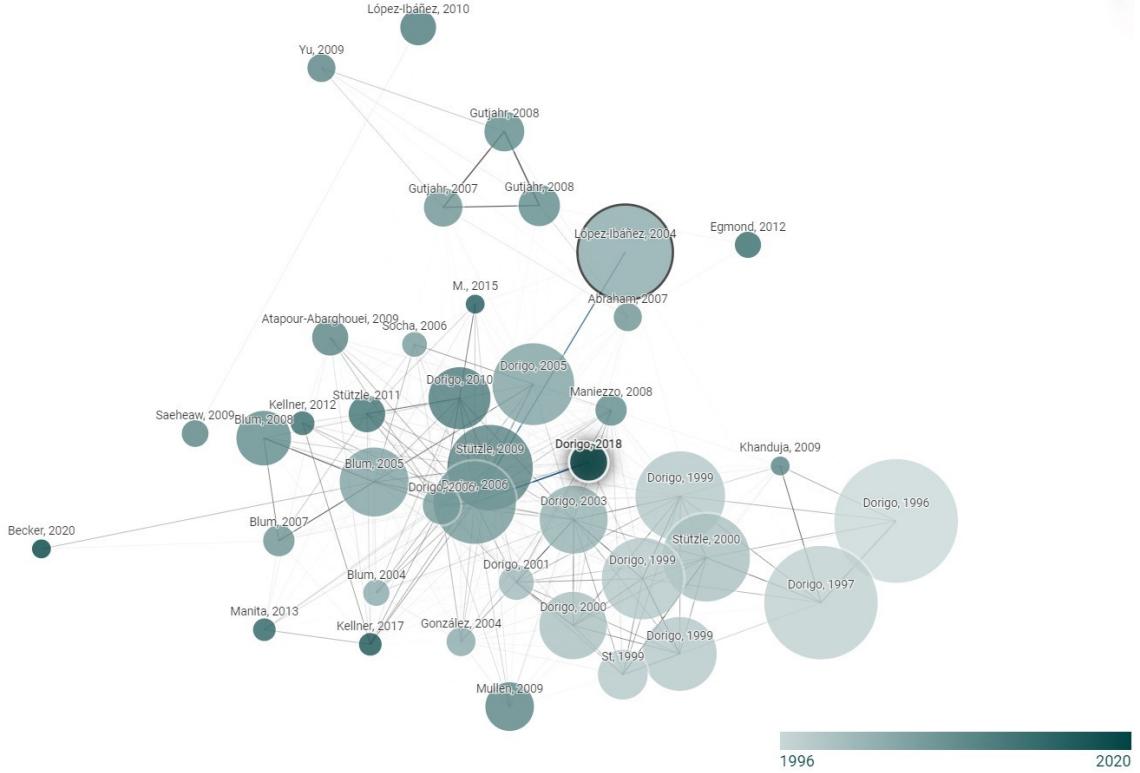


Figura 2.9: Paper connessi ad Ant Colony Optimization.

Grandezza Nelle reti reali il numero di nodi e archi è variabile, per esempio la rete neurale del verme *elegans* (l'unico sistema nervoso interamente mappato di un organismo vivente) consiste di 302 neuroni, quindi di 302 nodi collegati tra loro da sinapsi. Al contrario, le stime dei neuroni della mente umana sono di un ordine di grandezza ben maggiore: centinaia di bilioni di neuroni. [42] Generalmente i network hanno un alto numero di nodi, si pensi che nella rete WWW ci sono 5,27 bilioni di pagine. [43]

Sparsità I nodi hanno grado basso se comparati con la grandezza del grafo. In altre parole la sparsità dei network si può spiegare con il numero di Dunbar: una persona è in grado di mantenere approssimativamente 150 relazioni sociali, il che significa 150 archi circa per ogni nodo considerando una rete sociale. Se pensassimo a noi stessi come centro del nostro ego-network, la nostra rete di conoscenze personale, allora la nozione del "magico" numero di Dunbar può essere immaginata con dei cerchi concentrici: all'interno del primo cerchio ci sono le persone più vicine con cui comunichiamo più

frequentemente, in un altro cerchio esterno quelle con cui comuniciamo occasionalmente, in un altro quelle con cui non comuniciamo regolarmente. [44] Il numero di relazioni con persone che una persona in media può avere è quindi ben definito, comunque non tenendo conto dei rapporti digitali, che complicherebbero di gran lunga le cose.

Small worlds Il fenomeno dello *small world*, noto anche come i sei gradi di separazione, afferma che dati due individui, in qualsiasi posto del mondo, si riesce a trovare un cammino (una sequenza di nodi adiacenti collegati da archi) che li connette di al massimo sei conoscenze tra loro. L'effetto *small world* consiste quindi nel fatto che la tipica distanza tra vertici è piccola, che anche individui che si trovano ai due poli del pianeta possono essere connessi tra loro da poche persone. [42]

Scale free Uno *scale free* network ha la proprietà di avere la distribuzione dei nodi che segue una power law. Questo significa che quando il numero di nodi aumenta, la struttura sottostante rimane *scale free*. A questo si collega il modello del *preferential attachment*, che prevede che se un nuovo nodo entra a far parte della rete, si connette con i nodi dal grado più alto. È noto anche come il paradigma del "ricco diventa più ricco".

Nel modello del *preferential attachment*, si descrive la procedura per aumentare un network come segue: nuovi nodi vengono creati ad ogni iterazione, la probabilità che l'arco colleghi il nuovo nodo ad un nodo esistente è proporzionale al grado del nodo corrente, in altre parole i nodi con un grande valore di grado sono i preferiti per diventare più popolari. All'aumentare della rete, la frazione di nodi segue una power law.

Il punto iniziale di questo fenomeno è l'osservazione che la popolarità di molti avvenimenti è distribuita su una power law e che ha una coda lunga, basti pensare che nel World Wide Web tra i 5,27 bilioni di nodi che rappresentano siti web ci sono pochi siti web con molti click al giorno e molti con pochi click al giorno. Una ragione per questa discrepanza consiste sicuramente nella

qualità dei contenuti, ma potrebbe anche esserci una diversa dinamica in gioco, che è l'idea del *rich gets richer*. In termini generali è quindi l'idea è che in tutti sistemi ci sia dell'instabilità. Nell'esempio dei siti web, un botto casuale iniziale nella rete può crescere velocemente e portare a siti superstar; sono i più fortunati per aver ricevuto questo effetto randomico.

Andando a formalizzare il *preferential attachment model*, dato un grafo G che rappresenta siti e link, il modello descrive come cresce un network.

- Una nuova pagina si connette tramite link ad una pagina esistente.
- Con probabilità p , si connette ad una nuova pagina in maniera casuale.
- Con probabilità $1 - p$, si connette ad una pagina con probabilità proporzionale alle pagine con maggior numero di archi entranti.

In conclusione, quando un fenomeno viene descritto da una power law, fornisce l'informazione che il sistema in questione è *scale free*, che ha una *long tail*, che la probabilità di eventi grandi decade quindi in maniera lenta.

2.3.1 Network embeddings

Con *network embedding* si intendono le tecniche per mappare i nodi di un grafo all'interno di un vettore di numeri reali, infatti il termine *embedding* viene tradotto come incorporamento. Un buon *embedding* dovrebbe preservare la struttura del grafo. I vettori risultanti da questo approccio possono essere usati come input per vari modelli e compiti di analisi di network e grafi, come *link prediction*.

In *Network Embedding and overview* [22] Arsov e Mirceva descrivono questo metodo come un approccio per codificare i nodi di una rete in modo tale che la similarità nello spazio di *embedding* rifletta la similarità all'interno del network. Tra i quattro metodi descritti per sviluppare questa metodologia trova particolare interesse il node2vec per le sue performance ottimali. L'obiettivo è quello di mappare nodi in uno spazio vettoriale di caratteristiche in maniera tale da massimizzare la similarità preservando la vicinanza tra nodi del network. Le word *embedding* vengono utilizzate anche in *text*

analytics con lo scopo simile di rappresentare in uno spazio vettoriale le parole e i loro significati in modo tale che parole più vicine corrispondano a significati simili. Il principio del network embedding verrà utilizzato per la rappresentazione dei risultati della procedura *Daucus Fractal Network Analyzer* (DFNA).

2.3.2 Networks and Fractals

Nella recente pubblicazione *Pre-training without Natural Images*, [23] un team di ricercatori giapponesi ha utilizzato un dataset di immagini frattali per allenare una rete neurale per compiti di *computer vision*. In sintesi, hanno scelto di utilizzare immagini frattali come alternativa per addestrare una rete neurale per il motivo che i dataset attualmente disponibili contengono preconcetti (*bias*). [45] Per *bias* in questo caso s'intende il fatto che, per esempio, per addestrare un modello per il riconoscimento di immagini vengono fornite fotografie di oggetti d'uso comune. Con l'esposizione ripetuta, il modello riesce così ad identificare un tipo di oggetto dall'altro. I ricercatori giapponesi hanno quindi mostrato come addestrare questi modelli su immagini simulate graficamente. Vengono quindi utilizzati per generare immagini raffiguranti forme naturali: come già affermato in precedenza i frattali possono essere trovati negli alberi, nei fiori e persino nelle nuvole e nelle onde, sfruttando l'ipotesi che queste immagini generative possano prendere il posto delle immagini di oggetti reali e possano insegnare ad un sistema automatizzato le basi dell'*image recognition*. I risultati provano che questa ipotesi sia valida e funzioni correttamente, e che potrebbe portare ad un'inversione di tendenza. Ovvero l'utilizzo di FractalDB (il dataset creato generativamente) al posto dei dataset attuali.

Un'ipotesi nata durante questo lavoro di tesi è che all'interno di network siano presenti strutture frattali. Richard Dryden, in un titolo ambizioso come *Modelling Processes in a fractal network: a possible substructure for*

consciousness [24] propone il concetto che la conoscenza non emerge da un solo livello di organizzazione biologica (per esempio una rete neurale) bensì da una conseguenza di attività interdipendenti tra network a diversi livelli. Pile di frattali e intercomunicazione tra network sono i punti cardine della sua teoria. Ora, nel nostro abbraccio interdisciplinare non andremo nel dettaglio del suo esperire conoscenza, ma ci limiteremo a trovare punti in comune e d'ispirazione per il nostro approccio.

Da questo punto di vista, le organizzazioni biologiche appaiono come insieme annidati di reti, con unità a livelli sconnessi che formano altre reti, e così via. Un termine consono per questa configurazione è reti frattali che hanno la proprietà della *self-similarity*. È importante notare che il termine frattale qui non si riferisce soltanto ad un livello di organizzazione, bensì a più livelli connessi. Quindi i network in cui tutti i livelli sono interconnessi formano una struttura organizzativa continua.

Rete apollinea Sono una categoria di reti per definizione frattale. Prendono il nome da Apollonio di Perga, vissuto più di duemila anni fa, che aveva proposto il problema dell'incastonatura dei cerchi: dati tre cerchi a due a due tangenti, il problema richiede di tracciare un cerchio a loro tangente. La rete apollinea introdotta da Andrade, [25] prende spunto da questo antico problema matematico creando una categoria di reti frattali che hanno la proprietà di trovare utilizzo in svariate applicazioni. Una rete apollinea è quindi un grafo indiretto formato da un processo di suddivisione ricorsiva di un triangolo in tre triangoli più piccoli. Le reti apollinee sono *scale free*.

2.4 Conclusione stato dell'arte

In questo capitolo è stato trattato ampiamente il settore di ricerca del *Natural Computing*, sono stati presentati i principali obiettivi ed è stato fatto un excursus dei più famosi algoritmi *nature-inspired*. Dalla somma di tutti questi esempi si possono trarre alcune conclusioni. Innanzitutto, il NC non è semplicemente un'alternativa ai classici metodi attuali di *problem solving*, ma è una scienza in forte espansione che può trovare soluzioni per vari tipi di problemi. Problemi complessi caratterizzati da un grande numero di variabili e di possibili soluzioni; problemi per cui non è possibile garantire l'ottimalità di una soluzione, ma è possibile confrontare le soluzioni fra loro; problemi difficilmente modellabili, come ad esempio il riconoscimento di pattern e la classificazione di immagini; problemi per cui è necessario modellare fenomeni e forme della natura, per cui ad esempio la geometria euclidea non si rivela adeguata. Date le potenzialità di queste tecniche, il NC è un settore in continua espansione e possiamo aspettarci continui risultati e miglioramenti.

Conseguentemente è stata aperta una parentesi botanica in cui è stata introdotta la pianta di *Daucus Carota*, e la classe di algoritmi ispirati alle piante infestanti. Nella sezione inerente ai frattali sono invece stati definiti gli oggetti matematici frattali e i metodi per calcolare la dimensione frattale. Sono stati presentati degli oggetti frattali per poi verificare l'esistenza di oggetti similmente frattali, come appunto il *Daucus Carota*. È stata anche calcolata la dimensione frattale della pianta di *Daucus Carota*, che ha un valore non intero compreso tra uno e due. Abbiamo introdotto lo studio delle reti per mettere le basi per il prossimo capitolo in cui verrà descritta la procedura *Daucus Fractal Network Analyzer* (DFNA). Abbiamo inoltre trovato delle similitudini tra reti e frattali, con fenomeni che possono essere descritti da una distribuzione power law.

Capitolo 3

Descrizione procedura

Viene ora presentato il contributo principale di questo elaborato: la procedura nature-inspired per l’analisi di reti *Daucus Fractal Network Analyzer* (DFNA). L’implementazione è disponibile a questo indirizzo https://github.com/kdd-lab/2020_Trevisan.

Prima di ciò, in una breve ma necessaria interpolazione viene descritto il problema NP completo dell’isomorfismo tra grafi e il modo in cui è stato affrontato in questa tesi. Successivamente viene presentata la procedura DFNA descritta in Sezione 3.2 e le diverse funzioni di cui è composta che sono state sviluppate: *identify* per l’identificazione di un pattern all’interno di una rete, *comprime* per comprimere il pattern in un unico nodo con tre tecniche differenti di compressione. Inoltre, viene data la definizione della funzione inversa, DFNA-Reverse, ideata con lo scopo di ricostruire un’approssimazione del grafo di partenza dato il grafo finale e un dizionario di memorizzazioni. Infine, viene descritta l’orchestrazione delle suddette funzioni della procedura DFNA con dei pratici esempi.

3.1 Il problema dell’isomorfismo

L’etimologia del termine isomorfismo deriva dal greco, è composto da due parti: *isos* che significa uguale, e *morphe* che significa forma. Per isomorfismo s’intende una trasformazione che conserva l’informazione. Nel già citato

libro di Hofstadter Gödel, Escher, Bach oltre ai comportamenti delle formiche viene approfondito ampiamente il concetto di isomorfismo:

Si parla di "isomorfismo" quando due strutture complesse si possono applicare l'una sull'altra, cioè far corrispondere l'una sull'altra, in modo tale che per ogni parte di una delle strutture ci sia una parte corrispondente nell'altra struttura; in questo contesto diciamo che due parti sono "corrispondenti" se hanno un ruolo simile nelle rispettive strutture. [6]

Hofstadter continua dicendo che spesso scoprire un isomorfismo tra due strutture note porta felicità, provoca stupore come un fulmine a ciel sereno. Lascia trapelare la sua teoria che riconoscere un isomorfismo tra due strutture note faccia compiere un notevole passo avanti nella conoscenza; sostiene anche che la percezione stessa di isomorfismi (nella trasformazione di oggetti a simboli, per riassumere un libro di 800 pagine in poche parole) crea significati nella mente umana. Qui si è appena parlato di isomorfismi in un linguaggio figurato, di una parola che ha tutta la vaghezza usuale delle parole. Ci si può riferire ad isomorfismo in termini di linguaggio, a livello di simboli tra cervelli di persone con uno stile di pensiero simile che comunicano; ma anche tra delle ragnatele, per esempio se volessimo stabilire se due ragnatele sono state tessute da ragni appartenenti alla stessa specie. Per stabilirlo cercheremmo di sovrapporre le due ragnatele e controllare se i singoli vertici corrispondono esattamente, proiettando con precisione angolo su angolo, fibra su fibra. Abbastanza astruso da mettere in pratica, infatti quando si parla di isomorfismo tra ragnatele ci si riferisce alle loro proprietà globali, alla loro forma complessiva. Due ragnatele sono dette isomorfe se appartengono alla stessa specie, se conservano le proprietà globali.

Nel nostro caso ci soffermeremo sul problema di isomorfismo tra grafi, o *graph matching*, il problema di trovare similarità tra grafi. Anche in questo caso il meccanismo sottostante consiste nel combaciare completamente, come per le ragnatele, nodi e archi di due grafi. Quando il problema viene esteso a trovare la corrispondenza esatta di un sotto-grafo in un altro grafo,

allora viene chiamato isomorfismo tra sotto-grafi, (*subgraph matching*).

La procedura che verrà a breve approfondita si basa sul problema del *subgraph matching*; con l'obiettivo di trovare strutture frattali all'interno di una rete, si fa riferimento a questo problema. Il problema dell'isomorfismo tra grafi è un problema NP completo, già intuitivamente per la combinazione di mapping di nodi che facciano combaciare del tutto due grafi. Anche il suo sotto-problema dell'isomorfismo tra sotto-grafi è NP completo. Un metodo per la sua soluzione di questa categoria di problemi consiste nell'approccio a forza bruta per tentativi, che consiste nella generazione di tutte le possibili combinazioni e nella successiva verifica. Questo porta a dei risultati in tempi non accettabili computazionalmente. Nell'implementazione ci baseremo

Algorithm 4 Grandiso algorithm

```
1: procedure GRANDISO(H, M)
2:   Accept a motif M, and a host graph H.
3:   while  $H \neq \emptyset$  do
4:     Preprocessing
5:     Identify the most "interesting" node in M (degree, attributes)
6:     Motif search
7:     Reporting
8:   end while
9:   return result set
10: end procedure
```

sull'utilizzo della libreria *grandiso* [26] per superare questo scoglio computazionale. La libreria *grandiso* non risolve il problema dell'isomorfismo tra grafi, ma lo approssima per fornire un risultato in tempi accettabili. È un pacchetto che fa estrazione di sotto-grafi (o *motif*): specificando in input il *motif* che si vuole trovare in un grafo, in output restituisce la lista di sotto-grafi con i nodi che fanno parte di quel *motif*. La ricerca dei *motif* la attua in base ad una precedente valutazione del grafo in questione, definendo i nodi più interessanti in termini di grado e di altri attributi.

3.2 Daucus Fractal Network Analyzer

A questo punto descriviamo la parte centrale di questa tesi, la procedura *Daucus Fractal Network Analyzer* (DFNA); dove *Daucus* sta per la pianta a cui si è ispirata, *fractal* per il fatto che si vada a cercare pattern frattali all'interno di reti, e infine *network analyzer* appunto perché il suo scopo principale sta nell'apportare e fornire un'analisi ulteriore alle reti.

Partendo da un grafo indiretto G e da un sotto-grafo detto *motif*, DFNA restituisce un grafo semplificato. Si è voluto dare alla procedura un carattere gerarchico lavorando per livelli, in modo tale da riprendere la struttura ramificata e intrinseca della *Daucus Carota*. Ricordiamo infatti che l'infiorescenza della *Daucus Carota* ha la particolare caratteristica che fa sì che di singoli peduncoli fiorali si trovino alla stessa altezza pur avendo lunghezze diverse in modo da avere i fiori posti sullo stesso livello.

DFNA si presenta come una pipeline che semplifica un grafo identificandone pattern frattali al suo interno e comprimendoli ricorsivamente. Assume la forma di un procedimento per trovare pattern, strutture ripetitive e ricorsive nei grafi. Come si vede nella Figura 3.1 che rappresenta un grafo d'esempio per la spiegazione, il grafo G è composto da evidenti strutture ridondanti al suo interno, quattro triangoli. La scelta del motif da individuare al suo interno risulta ovvia, come si vede nella Figura 3.2 è infatti il triangolo che più s'addice a comprimere il grafo.

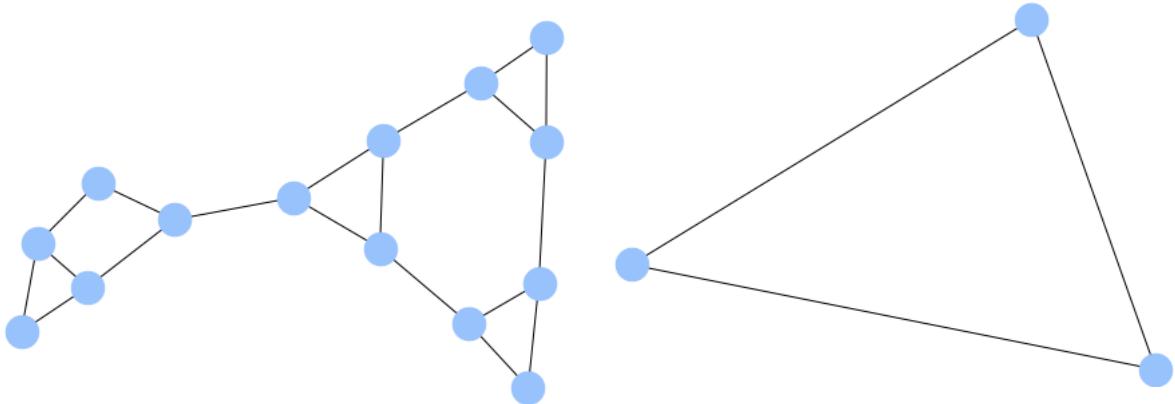


Figura 3.1: Grafo di partenza G .

Figura 3.2: Motif da identificare nel grafo G .

Si è deciso di sviluppare la procedura DFNA su grafi indiretti, i cui ar-

chi non sono né pesati né colorati. Ecco una descrizione ad alto livello dell'algoritmo, dato un grafo G e un sottografo *motif*:

Algorithm 5 Daucus Fractal Network Analyzer - DFNA

```
1: procedure DFNA( $G$ , motif)
2:   Rename( $G$ )
3:   while Identify  $\neq \emptyset$  do
4:     Identify( $G$ , motif)
5:     Comprime
6:     Update level
7:   end while
8:   return Result
9: end procedure
```

I dati che vengono forniti in input sono un grafo e un sotto-grafo *motif*, in output viene restituita una variante del grafo G , la sua versione ridotta e compressa nell'eventualità in cui uno o più sotto-grafi siano presenti in G . Come sintetizzato nello Pseudocodice 5 e rappresentato graficamente nella Figura 3.3 la procedura DFNA si sviluppa in un approccio gerarchico per livelli ed è composta da varie e differenti funzioni.

In estrema sintesi, DFNA semplifica il grafo trovando strutture ripetitive-ricorsive al suo interno. Una volta definito il grafo e il sotto-grafo, si passa alla fase di rinomina del grafo G . Il sotto-grafo viene quindi identificato all'interno del grafo, compresso e cercato nuovamente. Andiamo ora più nello specifico andando ad illustrare più nel dettaglio il funzionamento delle funzioni che compongono DFNA: *rename*, *identify*, *comprime* e *memorize* per il procedimento inverso.

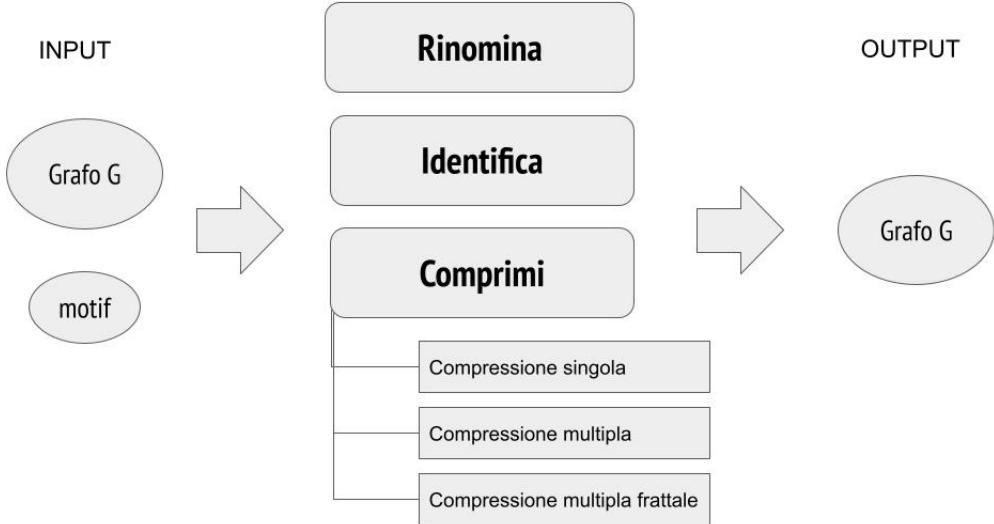


Figura 3.3: Diagramma *Daucus Fractal Network Analyzer*.

3.2.1 Rename

La necessità di una funzione che rinomini i nodi del grafo, generalmente rinominati come numeri interi, è emersa per vari motivi. Innanzitutto per l'ingresso in scena del concetto di **livello**: ad ogni iterazione dell'algoritmo il livello aumenta e i nodi cambiano. Metaforicamente parlando si potrebbe immaginare questa procedura come un ascensore all'interno di un ipotetico grattacielo. Ogni piano corrisponde ad ogni livello dell'iterazione, mentre l'ascensore è la procedura stessa, la quale compilando procede verticalmente lungo il grattacielo. Come nasce la necessità di tenere traccia del piano a cui l'ascensore si ferma, così anche la situazione all'interno del grattacielo va tracciata: i nodi vanno quindi rinominati in base al loro livello d'appartenenza.

Nel seguente Pseudocodice 6 si illustra nel dettaglio la funzione *rename*:

Algorithm 6 DFNA Rename

```
1: Dato un grafo G con i nodi rinominati con numeri interi (1, 2, 3, ..., N)
2: Rinomino l'insieme di nodi N con la tupla (livello, nodo)
3: procedure RENAME(G)
4:   level = 0
5:   nodi = list(nodi)
6:   for node  $\in= G$  do
7:     rename singolo nodo con tupla (livello, nodo)
8:   end for
9:   return G
10: end procedure
```

Il problema di trovare un modo ottimale per tenere conto della trasformazione dei nodi con l'aumentare dei livelli è stato risolto rinominando i nodi del grafo G aggiungendo l'informazione del livello a cui si trovano. Se quindi precedentemente un nodo veniva rinominato di default con un numero intero, ora tramite la funzione *rename* il nome del nodo consiste di una tupla $(livello, id)$ dove il primo elemento della tupla è il livello di aggregazione, il secondo è il nuovo id del nodo. Come primo passo della procedura, la funzione *rename* cambia il nome di tutti i nodi inizializzandoli a livello zero; prende il posto dell'intero la tupla con al primo elemento il numero del livello, al secondo l'id del nodo. È stato scelto questo stile di rinomina dei nodi per poter preservare le informazioni utili per la ricostruzione del nodo che verrà mostrata successivamente con la DFNA-Reverse, che ricostruisce il grafo di partenza grazie all'informazione cruciale del livello di appartenenza dei nodi. Sono state scartate altre possibili tecniche di regex per ragioni di semplicità computazionale.

Nodi default	Nodi post renaming
1, 2, 3	(0, 1), (0, 2), (0, 3)

Tabella 3.1: Effetto di rinomina dei nodi.

3.2.2 Identify

Come descritto in precedenza, il problema di identificare un sotto-grafo all'interno di un grafo è noto come *subgraph matching*, e in questo contesto includeremo la funzione *findmotif* della libreria *grandiso* per accelerare i tempi computazionali evitando operazioni greedy costose. Per i dettagli della funzione facciamo quindi riferimento allo Pseudocodice 4.

Spostiamo ora l'attenzione alla definizione dei pattern, i sotto-grafi da identificare all'interno delle reti. Oltre alla definizione di grafi completi composti da una piccola quantità di nodi, per la definizione dei pattern si è deciso di fare riferimento alla pubblicazione *Grasping frequent subgraph mining for bioinformatics applications*, [27] in cui vengono riassunte le principali tecniche di *subgraph mining* per l'applicazione in ambito medico.

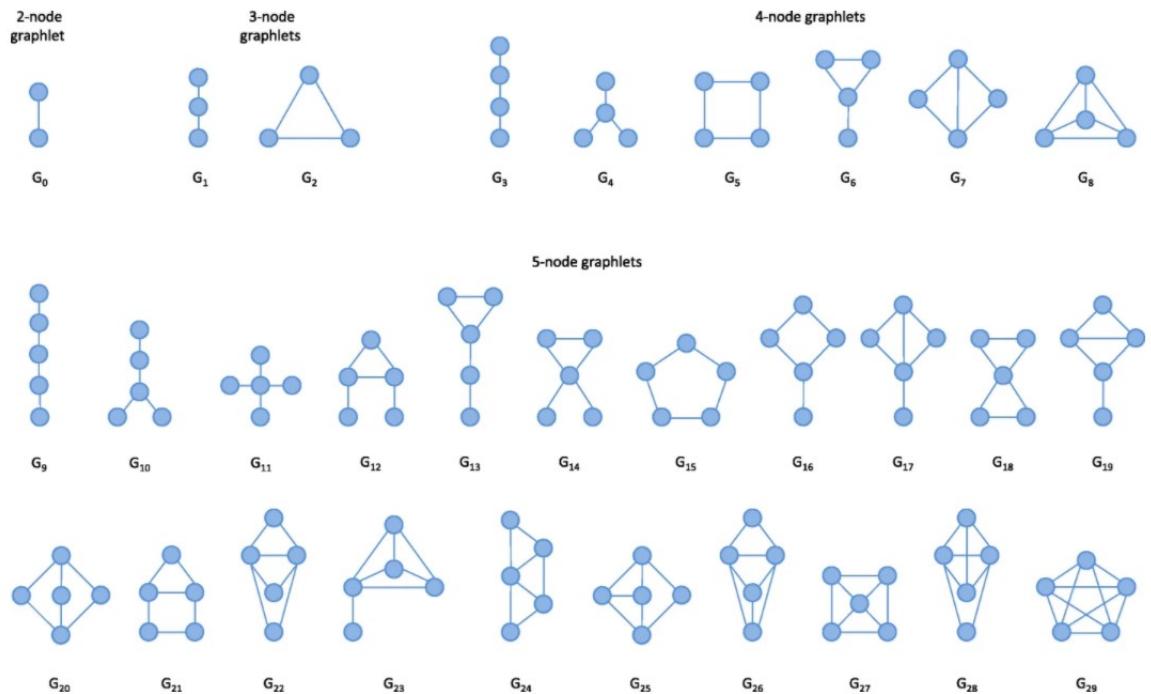


Figura 3.4: 30 possibili combinazioni di grafi di grandezza da due a cinque nodi.

Come rappresentato in Figura 3.4, i sotto-grafi sono tutte le combinazioni possibili da due a cinque, per un totale di 30 possibili sotto-grafi. Per gli esempi iniziali viene usato il grafo completo triangolare, per poi estendere nella fase di testing ad una decina tra le svariate configurazioni. Dalla lettura dei paper [27] e [28] emerge che i *motif* sono dei pattern ricorrenti che

appaiono più di frequente in reti reali che casuali e richiedono di trovare una determinata struttura in un grafo.

3.2.3 Comprime

Una volta identificato un motif all'interno del grafo, questo viene compresso in un singolo nodo. Sono state implementate tre varianti per la compressione che verranno discusse in seguito con un esempio pratico così da evidenziare le differenze tra loro.

3.2.4 Memorize

Oltre alle suddette funzioni, la procedura è stata sviluppata con un'ottica di futura creazione del suo procedimento inverso. È stato vitale per questo task memorizzare con il mapping quali nodi vengono compressi per riapplicare poi il procedimento al grafo risultante ricostruendolo a meno di un fattore d'errore.

Partendo da DFNA come approccio top-down, la funzione inversa poc'anzi descritta è bottom-up. Questi due sono infatti gli approcci principali quando si affrontano problemi riguardanti il funzionamento della natura. [2] La prima funziona dall'alto in basso poiché parte dall'assunzione che sia possibile trarre informazioni dall'osservazione dei pattern di una rete ad un macro livello; l'altra dal basso in alto poiché procede esattamente nel senso opposto, spostandosi a ritroso fino ad arrivare al network di partenza.

La funzione inversa prende in input il grafo compresso e un dizionario di mapping e inverte la procedura di compressione. La ricostruzione è parziale, si ottiene un'approssimazione del grafo di partenza. Non c'è una corrispondenza uno ad uno, ma si ottiene un grafo simile. Il grafo ottenuto avrà proprietà uguali a quello originale, sarà una sua versione anonimizzata e la sua topologia sarà lievemente diversa.

Algorithm 7 DFNA Reversed

```
1: procedure DFNA REVERSED(G, mapping)
2:   while node  $\in= G$  do
3:     if node in mapping then
4:       Expand node
5:     end if
6:   end while
7:   return G
8: end procedure
```

3.3 Demo DFNA

Viene ora fornita una spiegazione del procedimento passo per passo su un grafo d'esempio. Il grafo di riferimento è stato scelto per rendere chiari i passaggi dell'algoritmo e dettagliato al punto giusto per affrontare le casistiche descritte sopra. Una volta identificato un motif all'interno del grafo, questo viene compresso in un singolo nodo. In questa sezione inoltre discutiamo approfonditamente le diverse procedure che abbiamo utilizzato per gestire la compressione. Sono tre tecniche di compressione: singola, che comprime un motif alla volta; multipla, che unisce più motif contemporaneamente; frattale, che comprime solamente i motif che si trovano sullo stesso livello. Andiamo ora a descriverle più nel dettaglio.

Nel grafo di partenza G (Fig. 3.5), tutti i nodi sono allo stesso livello di partenza, inizializzato a zero. Il *motif* da trovare al suo interno è il triangolo. Ci sono quattro triangoli al suo interno che vengono rilevati dalla funzione *identify*: le triadi $[(0, 1), (0, 2), (0, 3)]$ in alto a destra, $[(0, 4), (0, 5), (0, 6)]$ in centro, $[(0, 7), (0, 8), (0, 9)]$ in basso a destra e $[(0, 10), (0, 11), (0, 12)]$ in basso a sinistra.

Ora che sono stati identificati i motif da comprimere, si analizzano le tre tecniche con cui farlo.

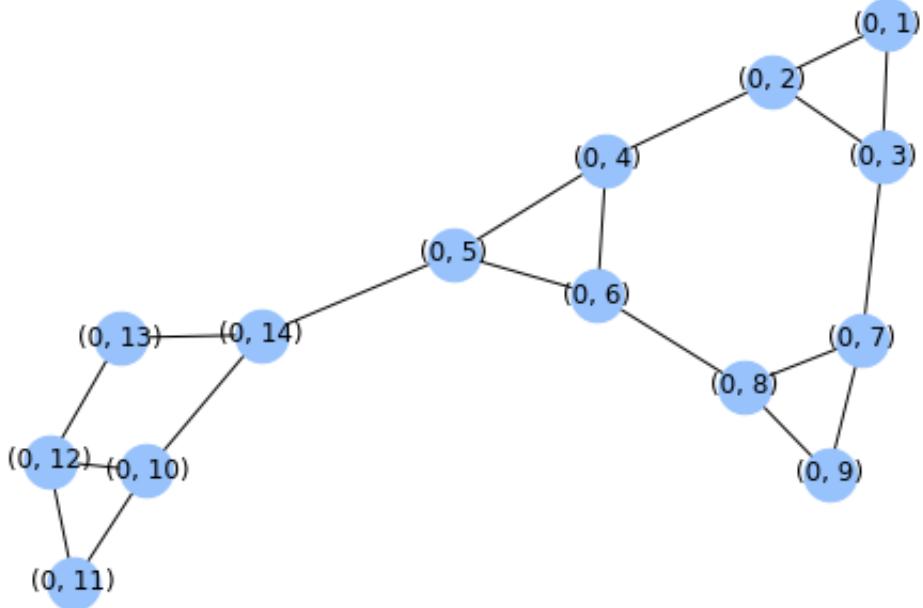


Figura 3.5: Grafo utilizzato per la demo

3.3.1 Compressione singola

Nella compressione singola per ogni livello viene fatta soltanto una compressione alla volta in maniera lineare. Questa compressione è stata sviluppata per assicurare che il motif identificato venga gestito e quindi compresso in un singolo nodo. Ciò significa che alla seconda compressione i nodi che non erano stati compressi al primo livello risultano già al secondo livello. È

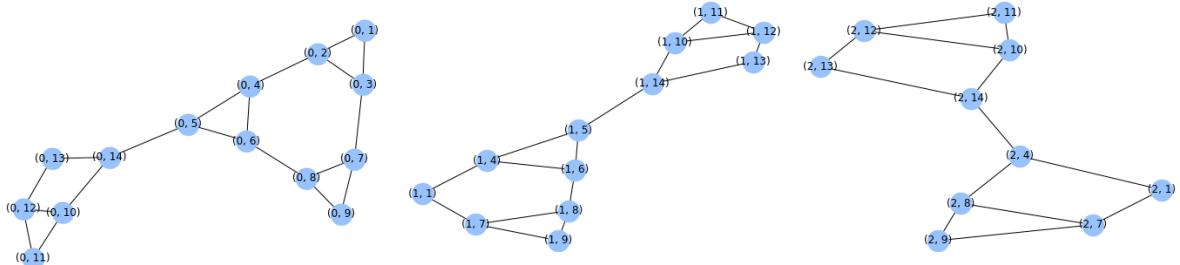


Figura 3.6: Grafo di partenza G .

Figura 3.7: Prima iterazione della compressione singola.

Figura 3.8: Seconda iterazione della compressione singola.

Figura 3.9: Demo compressione singola.

emerso quindi il problema dell'espansione dei livelli, che diventano troppi e ingestibili. Come si evince dalla Figura 3.9, in due iterazioni riesce a comprimere solo due dei quattro motif identificati. Se da una parte la compressione singola ha il vantaggio di tenere sotto controllo le varie compressioni effettuate.

tuate per un'eventuale ricostruzione, ha anche lo svantaggio di agire troppo lentamente nella compressione non tenendo conto del livello come "iterazione in cui più motif sono presenti". Si arriva ad avere un albero risultante profondo e non realistico. Nasce così la necessità di gestire la compressione in maniera diversa, perché in un livello potrebbero esserci due triangoli in due parti diverse del grafo, ed è necessario comprimerli contemporaneamente.

3.3.2 Compressione multipla

La compressione multipla su livelli mescolati comprime i *motif* senza controllare il loro livello d'appartenenza. Per esempio se in un triangolo identificato due nodi sono a livello zero e uno al livello uno, quando si comprimono il livello diventa due, ovvero il $\max+1$. In questo modo s'incrementa solo il livello dei nodi identificati. Questa tecnica restituisce una compressione massima perché mano a mano che trova comprime. Se comprimendo emergono delle strutture comprimibili indipendentemente dal livello, allora queste vengono compresse ugualmente. Ogni volta che si comprime si va semplicemente ad incrementare il livello del nodo compresso.

Come rappresentato nella Figura 3.13, in sole due iterazioni la compressio-

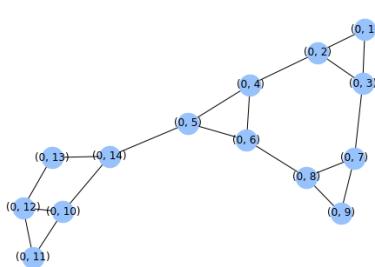


Figura 3.10: Grafo di partenza G .

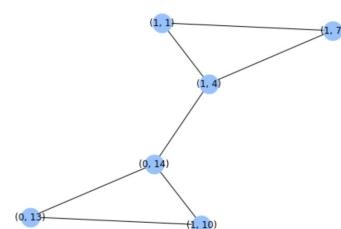


Figura 3.11: Prima iterazione compressione mista.

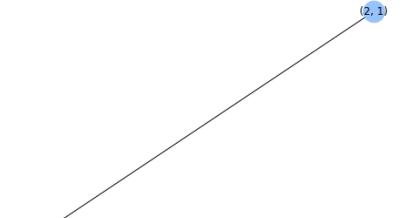


Figura 3.12: Seconda e ultima iterazione compressione mista.

Figura 3.13: Demo compressione mista.

ne mista porta il grafo G ad un grafo finale composto da soli due nodi. Come già detto si comprimono i motif appena vengono identificati, senza ulteriori controlli. Il motif $[(1, 10), (0, 13), (0, 14)]$ compresso dopo la prima iterazione

presente nella Figura 3.11 è composto da due nodi appartenenti al livello zero e uno al livello uno, e questo non genera problemi per questo tipo di compressione. La caratteristica principale di questa compressione, il fatto di essere mista e gestire più livelli contemporaneamente la rende svantaggiosa per un'eventuale ricostruzione. È invece vantaggiosa perché porta alla compressione totale di un grafo nel minor numero di iterazioni.

3.3.3 Compressione frattale

Una terza variante della funzione di compressione è quella frattale, sullo stesso livello. S'impone qui che i nodi siano allo stesso livello prima di comprimerli. A differenza della precedente è una procedura più scalabile e omogenea. La compressione frattale quindi comprime un *motif* solo se tutti i suoi nodi hanno lo stesso id temporale, lo stesso livello. Si comprime solo e finché si trovano delle strutture effettivamente frattali, dei *motif* indipendenti sullo stesso livello. La ricorsività trova la parte frattale e pulisce dal rumore di fondo togliendo i nodi traslati, i nodi non compressi. All'iterazione successiva identifica strutture solamente tra quelle già compresse. I nodi traslati non verranno più considerati.

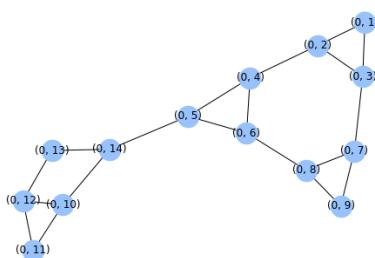


Figura 3.14: Grafo di partenza G .

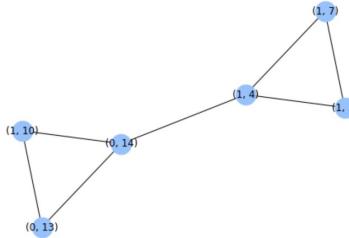


Figura 3.15: Prima iterazione compressione frattale.

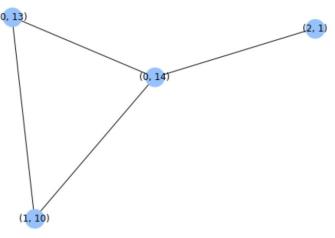


Figura 3.16: Seconda e ultima iterazione compressione frattale.

Figura 3.17: Demo compressione frattale.

Anche questo tipo di compressione è rapida: in due iterazioni porta alla conclusione della compressione del grafo G , come raffigurato in Fig. 3.17. Nonostante la sua rapidità, il vincolo che impone che un motif, affinché venga compresso, debba avere tutti i nodi sullo stesso livello, toglie l'occasione

a molti motif di essere compressi. La triade composta da due nodi appartenenti al livello 0 e un nodo appartenente al livello 1 [(1, 10), (0, 13), (0, 14)] presente nella Figura 3.15, pur essendo un grafo completo di tre nodi, non può essere compressa.

Potrebbero esserci *motif* adiacenti da comprimere, come due triangoli con un vertice in comune. La decisione arbitraria che è stata presa è di comprimere linearmente un *motif* alla volta, in questo caso se due triangoli sono adiacenti, allora ne viene compresso uno soltanto. La probabilità di incapparsi in questo caso è stata drasticamente diminuita con lo sviluppo della tecnica di compressione frattale e nell'inserimento della variabile di controllo (flag) che discrimini la modalità di compressione delle strutture identificate.

Si è notato da questi tre esempi che la compressione singola è completamente differente dalle altre due. Infatti ciò che cambia nelle ultime due tecniche è solo lo step finale. In conclusione, tra le tre tecniche di compressione elencate la compressione singola è la meno usata, a causa dell'espansione eccessiva nel risultato. Tra le due rimanenti, la compressione mista viene utilizzata maggiormente per i suoi minori vincoli alla compressione per il suo essere molto simile a quella frattale.

3.4 Commenti sulla complessità di DFNA

Per il calcolo della complessità di DFNA abbiamo testato la procedura calcolandone il runtime su un network Erdős-Rényi con grandezza che aumenta in maniera progressiva. Il modello Erdős-Rényi prende il nome dai due matematici che l'hanno inventato ed è il modello che più si avvicina alla generazione di un network casuale. Per questo è stato scelto per calcolare il tempo di esecuzione di DFNA.

I parametri scelti per la costruzione del modello Erdős-Rényi sono due: il numero di nodi n e la probabilità p che un arco sia incluso. Per il calcolo

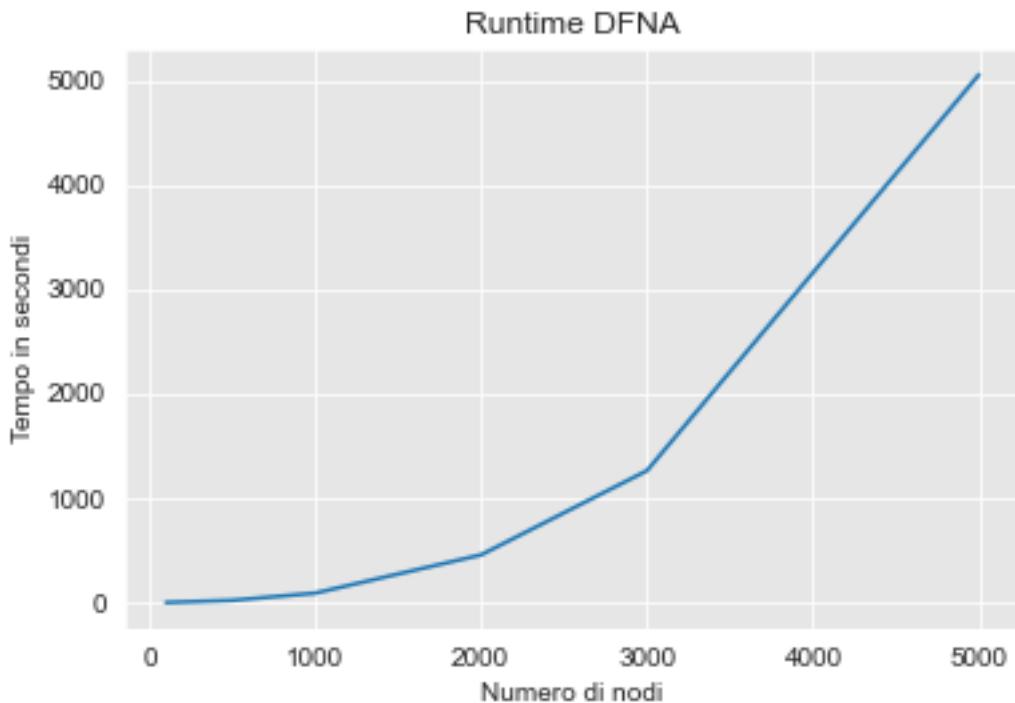


Figura 3.18: Tempi d'esecuzione DFNA

della complessità si è fissata p pari a 0.3 mentre il numero di nodi aumenta dell'ordine di grandezza delle migliaia ad ogni test. Dalla Figura 3.18 si nota che il tempo di esecuzione di DFNA aumenta con un andamento esponenziale all'incremento dei nodi della rete. Sull'asse delle ascisse sono rappresentati il numero di nodi che compongono la rete, 100, 500, 1000, 2000, 3000, fino ad un massimo di 5000 nodi. Sulle ordinate invece sono rappresentati i secondi. Ci sono margini per il miglioramento dell'efficienza di DFNA, anche se è bene ricordare che l'isomorfismo tra grafi che fa parte della procedura è un problema tuttora NP completo e che il linguaggio utilizzato, python, ha comunque dei limiti, specialmente per l'utilizzo di altre librerie che rallentano i tempi di esecuzione.

3.5 Demo DFNA inversa

Passiamo ora ad illustrare una demo della funzione DFNA inversa, *DFNA-Reversed*. È stato scelto come grafo da utilizzare per la ricostruzione il grafo risultante dalla compressione multipla, il grafo in Figura 3.19 composto da

due nodi, $(2, 1)$ e $(2, 10)$. I punti di partenza per la funzione DFNA inversa sono quindi il grafo finale appena descritto e la mappatura ricavata dalla funzione *memorize*, rappresentati nella Tabella 3.2.

Nodo	Memorizzazione
$(1, 1)$	$[(0, 1), (0, 7), (0, 8)]$
$(1, 4)$	$[(0, 4), (0, 5), (0, 6)]$
$(1, 7)$	$[(0, 7), (0, 8), (0, 9)]$
$(1, 10)$	$[(0, 10), (0, 11), (0, 12)]$
$(2, 10)$	$[(1, 10), (0, 13), (0, 14)]$
$(2, 1)$	$[(1, 1), (1, 4), (1, 7)]$

Tabella 3.2: Schema della memorizzazione dei nodi per la ricostruzione in DFNA inversa

Lo svolgimento della funzione DFNA inversa viene rappresentato nella Figura 3.21. In questo caso non ci sono i passaggi intermedi dell'iterazione, bensì il grafo finale e il grafo ricostruito.

È importante notare che la ricostruzione del grafo di partenza è corretta per

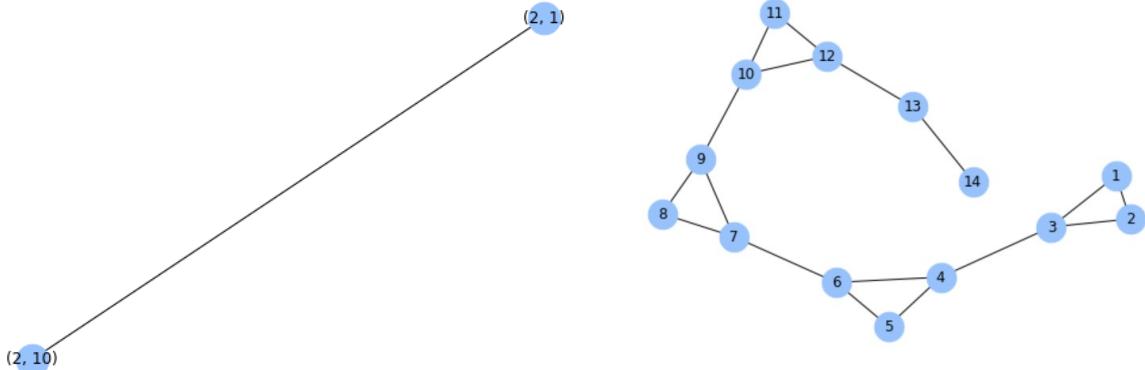


Figura 3.19: Grafo G finale

Figura 3.20: Ricostruzione dell'approssimazione del grafo G.

Figura 3.21: Demo DFNA inversa.

quanto riguarda la composizione dei nodi: i nodi avendo una memorizzazione all'interno della tabella non vengono rimossi. In questo modo si riesce a tenere traccia del tipo di nodi: nodi pieni da espandere e nodi vuoti da traslare al livello successivo. Espandendo i nodi pieni si ottiene un'approssimazione della struttura frattale di partenza. Tuttavia, l'informazione degli archi riesce a ricostruire un'approssimazione del grafo di partenza, perché l'assunzione di partenza prevede che, qualora un nodo pieno venga espanso, i nodi risultanti siano collegati tra loro. In questo modo però le ramificazioni

si diradano e si perde l'informazione che in un livello pattern diversi possano essere connessi tra loro.

Capitolo 4

Analisi e risultati

In questo capitolo sono riportati i risultati ottenuti a seguito delle sperimentazioni del modello DFNA su quattro classi di reti. Inizialmente viene fornita una breve spiegazione sulla configurazione dei diversi esperimenti. Successivamente vengono presentati e discussi i risultati ottenuti distinguendoli per le diverse classi, così da poter fornire un'analisi comparativa. Per alcune delle reti vengono anche forniti i risultati della procedura DFNA inversa dopo la ricostruzione della decompressione della rete precedentemente compressa.

4.1 Organizzazione esperimenti

Durante la fase sperimentale di questa tesi si è deciso di procedere individuando tre reti appartenenti a quattro classi diverse, in maniera tale da avere un corpus di reti vario e ampio. Dopo una descrizione delle classi e delle reti che le compongono, vengono esposti nella Sezione 4.2 i risultati prodotti dalla procedura DFNA sulle reti. Gli esperimenti sono basati sul confronto delle varie misure e statistiche sul grafo di partenza e sul grafo risultante.

4.1.1 Reti analizzate

Le quattro classi di reti reali analizzate sono state scelte da quattro gruppi differenti, con l'obiettivo di riuscire a trovare informazioni caratteristiche tipiche per ogni gruppo di reti estratte da *Network Repository* [46]. Le quattro classi, sintetizzate in Tabella 4.1, vengono ora presentate e descritte nel dettaglio, con l'analisi delle caratteristiche principali delle reti e degli indicatori per ogni classe. Dopo lo studio della complessità del capitolo precedente si è scelto di testare DFNA su reti dell'ordine di grandezza della centinaia di nodi, così da avere le migliori performance in termini di tempi d'esecuzione.

Animal Social Networks		
Insecta ant colony	Aves wild bird	Mammalia dolphin
Brain Networks		
Mouse brain	Mouse visual cortex	Macaque rhesus brain
Ecology Networks		
Mangwet	Foodweb	Everglades
Chemical Networks		
Elegans	CE-LC	CE-GT

Tabella 4.1: Schema riassuntivo delle dodici reti, divise in quattro classi, su cui sono stati effettuati gli esperimenti.

Animal Social Network Il primo gruppo di reti utilizzato per la fase di sperimentazione appartiene alla categoria delle reti sociali di animali. Queste reti reali rappresentano l'interazione tra animali e provengono da studi svolti con animali selvaggi, in cattività e addomesticati. Gli animali sono i nodi delle reti, nel nostro caso formiche, uccelli e delfini. Gli archi rappresentano l'interazione tra animali della stessa specie, sono indiretti e pesati, anche se il peso verrà tralasciato nella nostra analisi.

La prima rete, *insecta ant colony* in Figura 4.1, descrive la specie di formiche *Camponotus fellah*, nello specifico di una colonia di formiche al trentesimo giorno di osservazione. I dati sono stati raccolti tramite video e il criterio per definire l'interazione tra formiche è quello del contatto fisico: una coppia di formiche si considera in interazione quando l'estremità anteriore di una

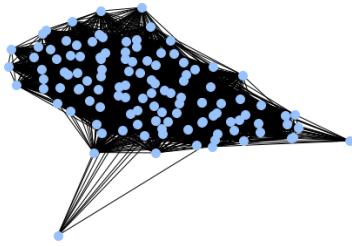


Figura 4.1: Rete insecta ant colony.

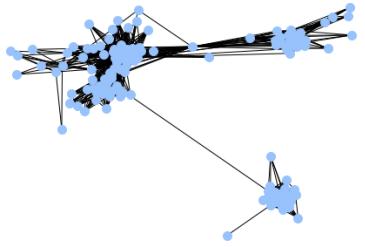


Figura 4.2: Rete aves wildbird.



Figura 4.3: Rete mammalia dolphin.

Figura 4.4: Classe di reti *Animal Social Network*.

formica è situata all'interno della forma trapezoidale che rappresenta l'altra formica. [29]

La seconda rete della classe di reti sociali animali ha come protagonista la specie degli uccelli *aves wildbird* e si trova in Figura 4.2. Ogni nodo rappresenta un uccello, mentre ogni arco l'interazione tra loro. I gruppi sono stati definiti come individui rilevati nello stesso nido durante lo stesso giorno e le interazioni rappresentano individui che si sono sovrapposti nei modelli di esplorazione del nido durante lo stesso giorno. [30]

La terza rete *mammalia dolphin* in Figura 4.3 descrive il delfino *Tursiops truncatus*. Ogni nodo rappresenta un delfino, mentre l'interazione è definita secondo prossimità spaziale. Essa è caratterizzata quindi da cattura di prede o persistenti episodi di ricerca di prede, come indicato da lunghe immersioni o comportamenti di alimentazione specializzati con cambiamenti di direzione tra gli affioramenti. [31] Per quanto riguarda la grandezza delle suddette reti: *insecta ant colony* contiene 118 nodi e 4709 archi, *aves wildbird* 131 nodi e 1444 archi, mentre *mammalia dolphin* 291 nodi e 3182 archi.

Nella figura 4.8 sono invece rappresentate le distribuzioni del grado delle tre reti. La prima rete *insecta ant colony* in Figura 4.5 ha una distribuzione di grado che propende per valori alti. Mentre per la rete *macaque rhesus* la distribuzione del grado dei nodi nella Figura 4.7 è più simile a quella del modello scale-free. In Figura 4.6 invece, la maggioranza dei nodi ha grado pari a venti.

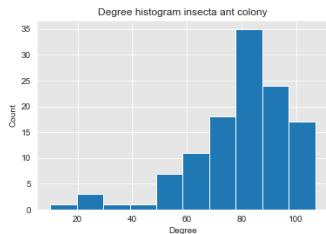


Figura 4.5: Istogramma insecta ant colony.

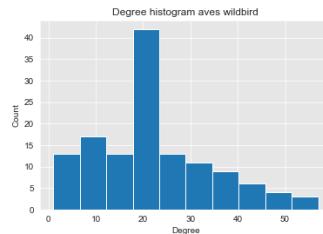


Figura 4.6: Istogramma aves wildbird.

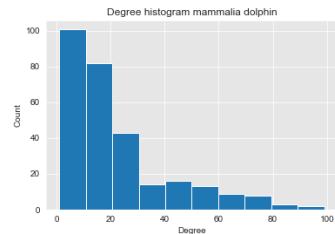


Figura 4.7: Istogramma mammalia dolphin.

Figura 4.8: Istogrammi di distribuzione del grado per la classe di reti *Animal Social Network*.

Le prime due reti sono composte da un'unica componente连通的, la terza invece, `mammalia dolphin`, è composta da tre componenti connesse: due delle quali un grafo completo stelliforme con $K = 5$ e una coppia di nodi. In Tabella 4.2 sono state svolte ulteriori analisi. La densità, il numero di archi presenti nella rete rispetto al numero totale di possibili archi, è maggiore per la rete `insecta ant colony`, anche perché è stato scelto il trentesimo giorno di osservazione della colonia.

Misura	Insecta ant colony	Aves wildbird	Mammalia dolphin
Densità	0.682	0.170	0.075
Cammino minimo medio	1.318	2.733	-
Coefficiente di clustering medio	0.781	0.803	0.682

Tabella 4.2: Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di animal social network.

Per la rete `mammalia dolphin` non è stato possibile calcolare il cammino minimo, essendo la rete composta da tre componenti connesse. Il coefficiente di clustering medio è simile per tutte e tre le reti.

Brain Network La seconda classe di reti presa in considerazione fa parte dello studio delle neuroscienze, nello specifico sono le reti neurali [32]. Sono reti complesse che rappresentano la connettività funzionale tra varie regioni del cervello e vengono estratte tramite risonanza magnetica. Le tre reti che fanno parte di questa classe sono: `mouse brain`, che rappresenta il connet-

toma di un topo; `mouse visual cortex`, che si riferisce anch'essa al cervello del topo andando nello specifico della corteccia visiva; `macaque rhesus brain`, il cervello della scimmia macaco *rhesus*. Tutte e tre le reti hanno in comune la caratteristica di essere sparse, gli archi sono indiretti e non pesati e rappresentano tratti di fibre che connettono un nodo all'altro.

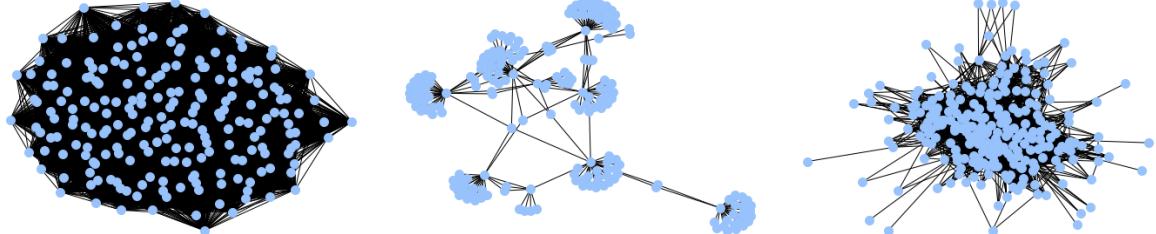


Figura 4.9: Rete `mouse brain`. Figura 4.10: Rete `mouse visual cortex`. Figura 4.11: Rete `macaque rhesus brain`.

Figura 4.12: Classe di reti *Brain Network*.

Come già anticipato, sono reti dell'ordine di grandezza delle centinaia, infatti la rete `mouse brain` contiene 213 nodi e 16242 archi, la rete `mouse visual cortex` 213 nodi e 214 archi, la rete `macaque rhesus brain` 242 nodi e 3054 archi. Come si evince dalla Figura 4.12, le tre reti hanno conformazioni diverse pur appartenendo alla stessa classe.

Nella figura 4.16 sono invece rappresentate le distribuzioni del grado delle tre reti. La prima rete `mouse brain` in Figura 4.13 ha una distribuzione di grado simile a quella di un modello Erdős-Rényi, random che si può approssimare alla distribuzione di Poisson. Mentre per la rete `macaque rhesus` la distribuzione del grado dei nodi nella Figura 4.15 è più similare a quella del modello scale-free, approssimabile ad una power law. In Figura 4.14 invece, la maggioranza dei nodi ha grado pari a uno.

Tutte e tre le reti sono composte da un'unica componente connessa. Inoltre, come si nota dalla Tabella 4.3, sono state svolte ulteriori analisi. La densità viene calcolata come il numero di archi presenti nella rete rispetto al numero totale di possibili archi. Sotto questo aspetto la rete `mouse brain` ha un maggiore valore di densità, pari a 0.719.

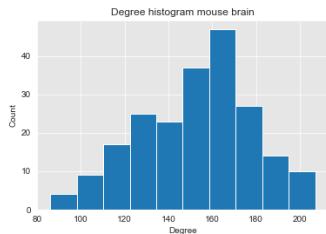


Figura 4.13: Istogramma mouse brain.

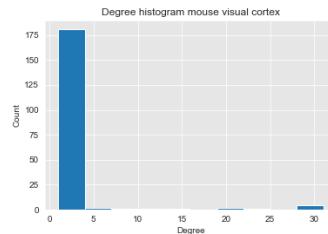


Figura 4.14: Istogramma mouse visual cortex.

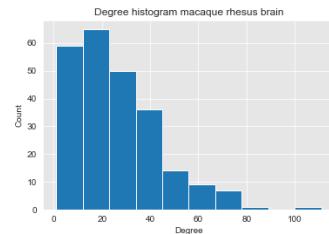


Figura 4.15: Istogramma macaque rhesus brain.

Figura 4.16: Istogrammi di distribuzione del grado per la classe di reti *Brain Network*.

La distanza nelle reti gioca un ruolo fondamentale nella determinazione del-

Misura	Mouse brain	Mouse visual cortex	Macaque rhesus brain
Densità	0.719	0.012	0.105
Cammino minimo medio	1.287	4.271	2.217
Coefficiente di clustering medio	0.758	0.021	0.450

Tabella 4.3: Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di brain network.

le interazioni tra componenti di un sistema. Il cammino viene definito come la sequenza di nodi aventi la proprietà che ogni coppia di nodi consecutivi è collegata da un arco. Per ogni rete è stato calcolato il cammino minimo medio e risulta essere più lungo per `mouse visual cortex`. Questo dato non è sorprendente, la distribuzione del grado in Figura 4.14 ha fatto emergere che i nodi hanno principalmente grado pari a uno. Di conseguenza aumenta così il cammino minimo medio da un nodo all'altro.

Per questa analisi preliminare è stato calcolato anche il coefficiente di clustering, che rappresenta il grado di clustering di una rete.

Ecology Network La terza classe di reti utilizzata per gli esperimenti di questa tesi è la classe di reti inerente all'ecologia. Le reti sono `mangwet` [33] composta da 97 nodi e 1446 archi, la rete `foodweb` [34] di 128 nodi e 2106 archi e `everglades` con 69 nodi e 885 archi. Come si evince dalla Figura 4.20, le tre reti hanno la stessa conformazione stelliforme.

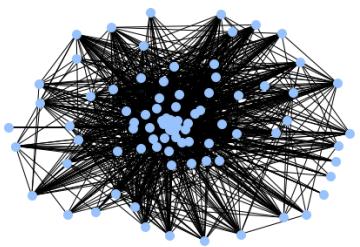


Figura 4.17: Rete mangwet.

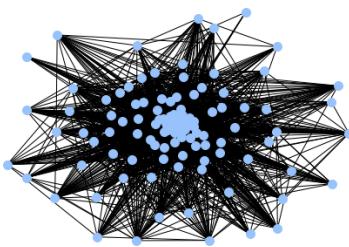


Figura 4.18: Rete foodweb.

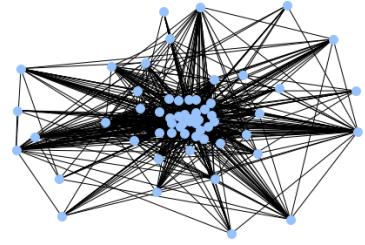


Figura 4.19: Rete everglades.

Figura 4.20: Classe di reti *Ecology Network*.

Nella figura 4.24 sono invece rappresentate le distribuzioni del grado delle tre reti, abbastanza simile tra loro. Nello specifico la rete foodweb in Figura 4.22 assomiglia a quella di un modello scale-free, che si può approssimare alla distribuzione di power law. Per quanto riguarda le altre due reti la maggioranza dei nodi ha grado inferiore a 50.

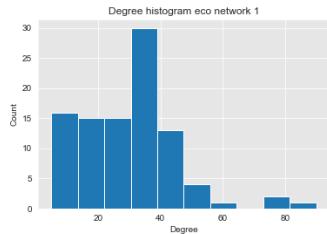


Figura 4.21: Istogramma mangwet.

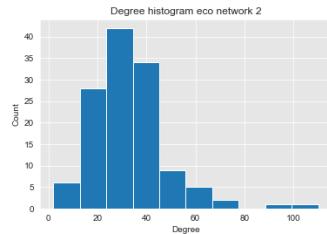


Figura 4.22: Istogramma foodweb.

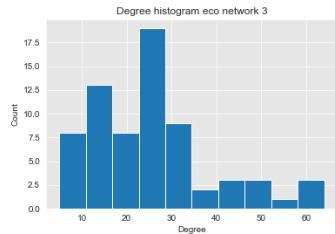


Figura 4.23: Istogramma everglades.

Figura 4.24: Istogrammi di distribuzione del grado per la classe di reti *Ecology Network*.

Tutte e tre le reti sono composte da un'unica componente连通. Inoltre, come si nota dalla Tabella 4.4, sono state svolte ulteriori analisi. La densità viene calcolata come il numero di archi presenti nella rete rispetto al numero totale di possibili archi. Sotto questo aspetto le tre reti hanno densità simile. Per ogni rete è stato calcolato il cammino minimo medio e risulta essere simile in tutte e tre le reti, pari a circa 1.7.

Classe di reti chimiche Il quarto e ultimo gruppo di reti analizzate per i test appartiene alle reti biologiche, tra le quali una è inerente ad interazioni me-

Misura	Mangwet	Foodweb	Everglades
Densità	0.311	0.259	0.377
Cammino minimo medio	1.693	1.772	1.636
Coefficiente di clustering medio	0.468	0.335	0.552

Tabella 4.4: Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di ecology network.

taboliche [35] e due di associazioni genetiche. [36]

La prima rete *elegans* si riferisce alla rete metabolica del verme *elegans*. Il già citato *elegans* è famoso per essere l'unico sistema nervoso interamente mappato di un organismo vivente. [37] Nel nostro caso ci soffermiamo ad analizzare il suo meccanismo metabolico, con una rete i cui vertici sono substrati, le molecole su cui agiscono gli enzimi, e gli archi sono le reazioni metaboliche, rappresentata in Figura 4.25.

Per le altre due reti in Figura 4.26 e 4.27, entrambe si riferiscono a asso-

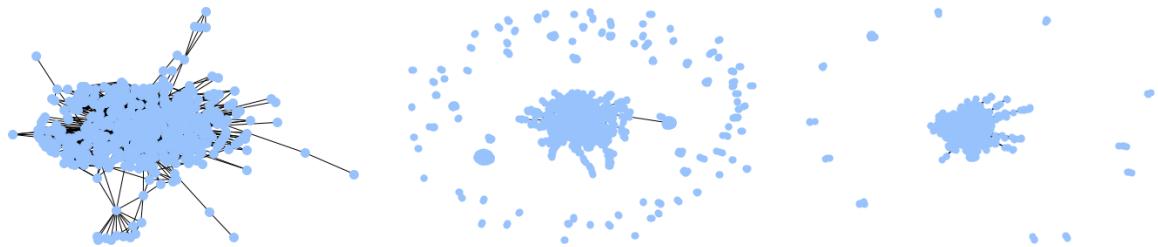


Figura 4.25: Rete *elegans*. Figura 4.26: Rete CE-LC. Figura 4.27: Rete CE-GT.

Figura 4.28: Classe di reti chimiche.

ciazioni funzionali genetiche studiate su vermi, quindi i vertici sono i geni e gli archi i collegamenti tra loro. Sono due reti diverse, CE-LC e CE-GT.

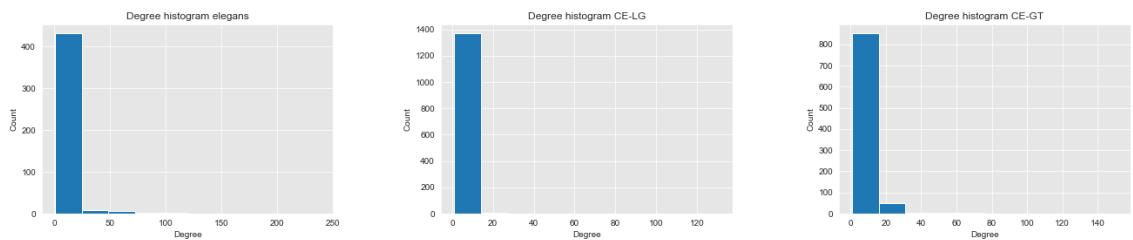


Figura 4.29: Istogramma della distribuzione del grado per la rete *elegans*. Figura 4.30: Istogramma della distribuzione del grado per la rete CE-LC. Figura 4.31: Istogramma della distribuzione del grado per la rete CE-GT.

Figura 4.32: Istogrammi di distribuzione del grado per la classe di reti chimiche.

Sono tre reti sparse, la distribuzione del grado dei nodi si assesta per tutte e tre le reti a valori molto bassi inferiori a venti, come emerge dai tre plot in Figura 4.32. La rete *elegans* contiene 453 nodi e 2040 archi, la rete CE-LC 1387 nodi e 1648 archi, la rete CE-GT 924 nodi e 3239 archi.

La rete *elegans* è composta da un'unica componente连通的, le reti CE-LC e CE-GT invece sono composte rispettivamente da 89 e 13 componenti connesse.

Misura	Elegans	CE-LC	CE-GT
Densità	0.020	0.002	0.008
Cammino minimo medio	2.66	-	-
Coefficiente di clustering medio	0.646	0.076	0.605

Tabella 4.5: Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di reti chimiche.

Come si evince dalla Tabella 4.5, la densità è bassa per tutte e tre le reti, il cammino minimo non è stato calcolato per le reti composte da più componenti connesse e il coefficiente di clustering medio si assesta attorno a 0.6 per la rete *elegans* e CE-GT.

4.1.2 Scelta dei motif

È stato appurato che le possibili combinazioni di grafi di grandezza da 2 a 5 nodi sono 30, rappresentate in Figura 3.4 nella Sezione 3.2.2 del capitolo precedente. Si è deciso di effettuare i test della procedura DFNA sulle quattro diverse classi di reti appena descritte con una varietà di pattern. Per ragioni di spazio è stato necessario scegliere un solo motif per rappresentare nel dettaglio questa serie di esperimenti. Il motif scelto è il triangolo, poiché il più facilmente interpretabile. Per la rete *insecta ant colony* viene ora mostrato il comportamento di DFNA in base a tipi diversi di motif. Utilizzando la coppia di nodi collegati da un arco, i tempi di esecuzione sono rapidi ma il risultato finale troppo semplificativo: ogni coppia di nodi viene iterativamente sostituita con un singolo nodo. Lo stesso vale per il motif composto da tre nodi collegati da due archi. Il quadrato invece aumenta i tempi di esecuzio-

ne rendendo più lunga l'identificazione di pattern quadrati all'interno di reti. Questo lascia supporre che il quadrato non è una struttura presente all'interno della rete. Viene rappresentata in Figura 4.1.2 la compressione al terzo livello della rete `insecta ant colony` e il rispettivo motif con cui è stata compressa. I risultati sono piuttosto simili, anche se per il motif composto

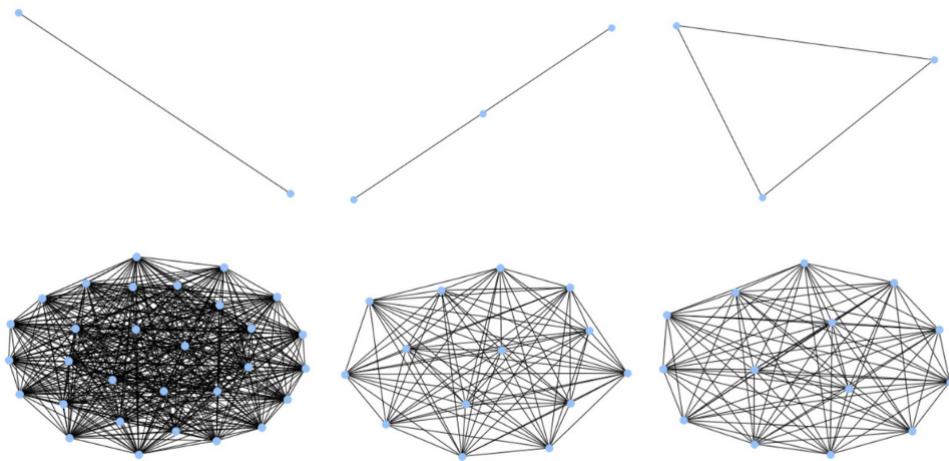


Figura 4.33: Motif e compressione su rete `insecta ant colony` al terzo livello

da due nodi si ha una rete ancora densa al terzo livello. La percentuale di perdita di nodi è al 83.90%, mentre per il motif composto da tre nodi e per il triangolo 98.3%. D'ora in poi lavoreremo solo con il triangolo come motif.

4.2 Discussione dei risultati

Andiamo ora ad esporre i risultati degli esperimenti ottenuti testando la procedura DFNA sulle 12 reti totali divise in quattro classi. Le analisi si svolgono confrontando varie misure tra cui la distribuzione del grado, la densità del grafo e altre caratteristiche di similarità presentate nella prossima sezione tra la rete originale e la rete risultante dopo l'applicazione della procedura DFNA.

4.2.1 Indici statistici

Sono state definite e implementate delle funzioni d'appoggio per questa fase sperimentale. Gli indici e le statistiche si dividono in globali, che descrivono il grafo interamente, e locali, più specifiche a caratteristiche specifiche.

Indici globali:

- **Grandezza**, ovvero il numero di nodi totali.
- **Distribuzione del grado**, calcola la distribuzione del numero di archi per ogni nodo del grafo.
- **Densità**, il numero di archi presenti nella rete rispetto al numero totale di possibili archi.
- **Grafo finale**, la struttura risultante estratta.
- **Decremento**, indice percentuale di perdita di nodi, ovvero la quantità di nodi che sono stati aggregati, compressi.
- **Nodi mantenuti invariati**, nodi traslati.

Statistiche per livello:

- Numero di livelli raggiungibili da ciascun pattern (motif, livello finale).
- Livello raggiunto dal pattern in compressione massima.

4.2.2 DFNA per estrazione di backbone

È stata applicata la procedura DFNA sui quattro gruppi di reti. Per omogeneità nell'esperimento iniziale illustriamo come le dodici reti rispondono, ognuna in maniera diversa, alla procedura DFNA settata con la modalità di compressione mista multi-livello e avente come motif da identificare il grafo completo di tre nodi, il triangolo.

La rete risultante dall'applicazione di DFNA viene definita come backbone di una rete, la sua struttura portante.

Applicazione DFNA su Animal Social Network Come dimostrato nella Sottosezione 4.1.2 per la rete insecta ant colony, il triangolo è stato scelto come motif per la sua facile interpretazione e modellazione. Nel seguente grafico (Fig. 4.37) sono rappresentate le tre fasi principali della procedura DFNA sulla rete insecta ant colony con il triangolo come motif e con la compressione mista. Per ragioni di spazio vengono raffigurati i momenti salienti dell'iterazione: la rete iniziale (Fig. 4.34), la rete intermedia (Fig. 4.35) alla terza iterazione, e la rete finale ottenuta applicando la funzione inversa in Figura 4.36.

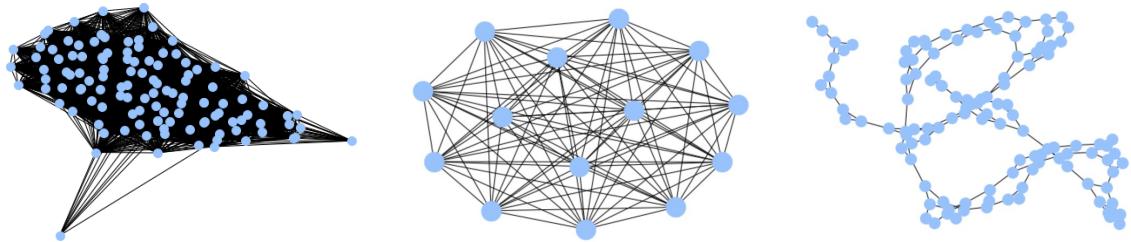


Figura 4.34: Rete insecta ant colony iniziale.

Figura 4.35: Rete insecta ant colony al terzo livello.

Figura 4.36: Rete insecta ant colony invertita.

Figura 4.37: Fasi salienti di DFNA su rete insecta ant colony con compressione mista.

Comprimendo la rete secondo triangoli e iterando con questo procedimento sulle reti risultanti si arriva al grafo finale composto da due nodi collegati da un arco, una coppia di nodi connessi. Come illustrato in Figura 4.36, la procedura DFNA si è snodata in cinque livelli e ha portato ad un decremento dei nodi da 118 a 2 nodi finali, del 98.30%. La rete ricostruita tramite procedura inversa (Figura 4.36) è composta dallo stesso numero di nodi ma da una conformazione di archi differente.

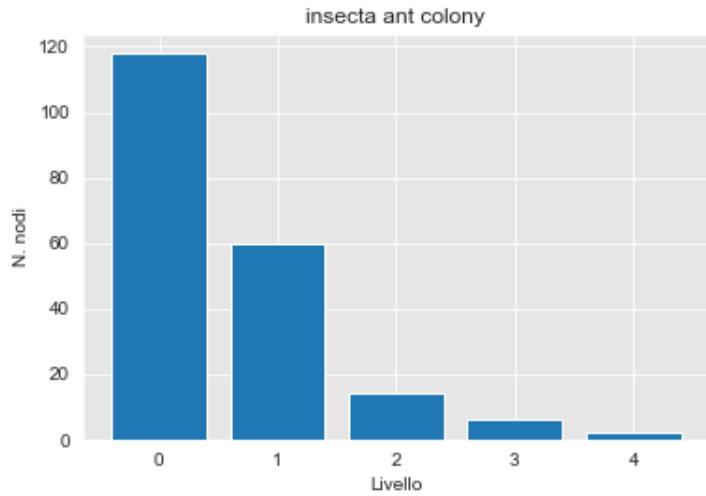


Figura 4.38: Grafico raffigurante il decremento dei nodi per livello dopo l'applicazione di DFNA su insecta ant colony.

Dagli esperimenti sulla rete aves wildbird il risultato rappresentato nella Figura 4.40 è una valida sintesi della rete iniziale rappresentata nella Figura 4.39. Vengono raffigurate due delle sei iterazioni totali, e il risultato inverso.

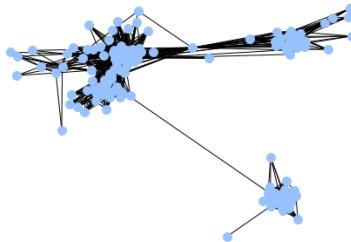


Figura 4.39: Rete aves wildbird iniziale.

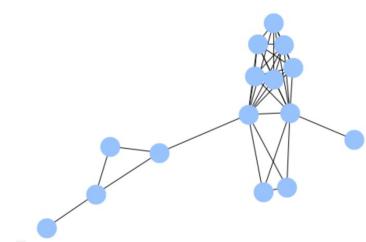


Figura 4.40: Rete aves wildbird al terzo livello.

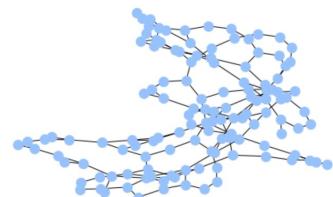


Figura 4.41: Rete aves wildbird invertita.

Figura 4.42: Fasi principali di DFNA su rete aves wildbird con compressione mista.

Come si evince dalla Figura 4.43, il decremento dei nodi è stato meno rapido rispetto alla rete insecta ant colony. In sei iterazioni si è passati da 131 nodi a 5 nodi finali, con una percentuale di perdita del 96.18%.

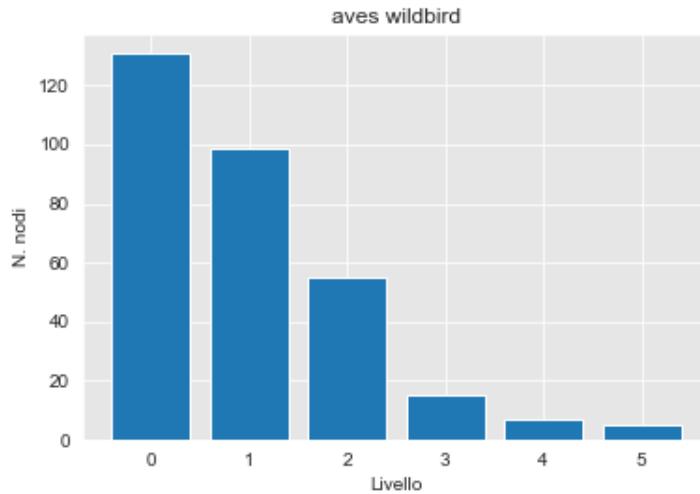


Figura 4.43: Grafico del decremento dei nodi per livello dopo l'applicazione di DFNA su *aves wildbird*.

Nella Figura 4.47 vengono riassunte le tre fasi dell'iterazione totale con la rete *mammalia dolphin*.

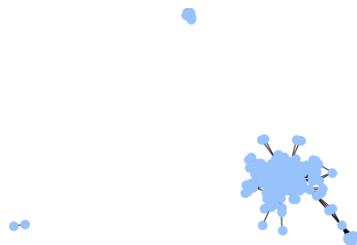


Figura 4.44: Rete *mammalia dolphin* iniziale.

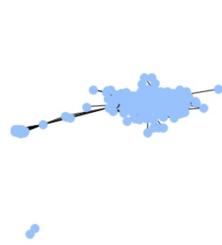


Figura 4.45: Rete *mammalia dolphin* al secondo livello.

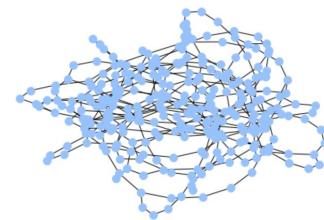


Figura 4.46: Rete *mammalia dolphin* invertita.

Figura 4.47: Fasi principali di DFNA su rete *mammalia dolphin* con compressione mista.

In questo caso DFNA il decremento dei nodi è del 68.04%, da 291 nodi della rete iniziale a 93 alla rete finale dopo la terza iterazione. La rete rimane composta da tre componenti connesse, una delle quali era inizialmente un pentagono e viene ridotta ad un singolo nodo. È stato calcolato il coefficiente medio di clustering della rete risultante, pari a 0.620. La densità della rete finale è 0.258, maggiore rispetto alla densità di partenza 0.075.

Nella Figura 4.48 è possibile notare che in tre livelli si è raggiunti ad un decremento di nodi e alla convergenza di DFNA, mentre nella Figura 4.50 viene raffigurata la distribuzione del grado della rete finale, diversa da quella

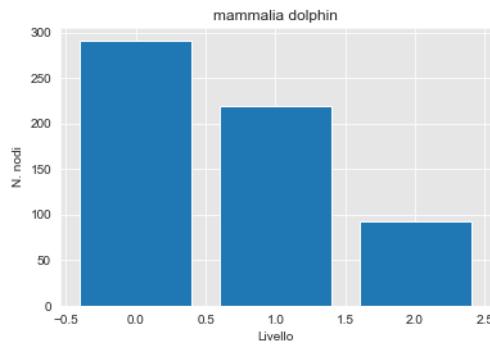


Figura 4.48: Grafico decremento nodi per livello dopo l'applicazione di DFNA su mammalia dolphin.

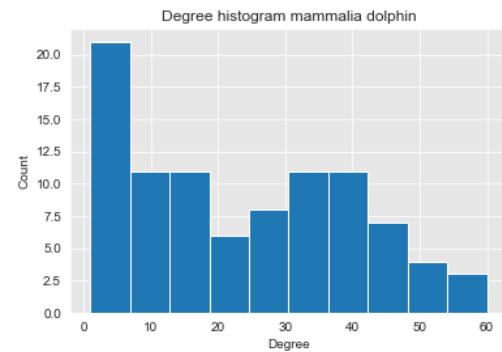


Figura 4.49: Distribuzione del grado della rete mammalia dolphin finale.

Figura 4.50: Statistiche per la rete mammalia dolphin.

iniziale in Figura 4.7.

Applicazione DFNA su Brain Network Nella figura 4.54 si vedono le tre fasi di DFNA sulla rete mouse brain.

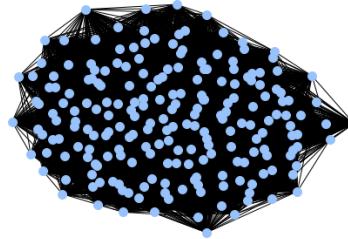


Figura 4.51: Rete mouse brain iniziale.

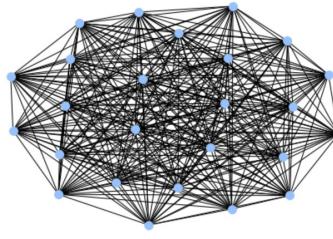


Figura 4.52: Rete mouse brain al terzo livello.

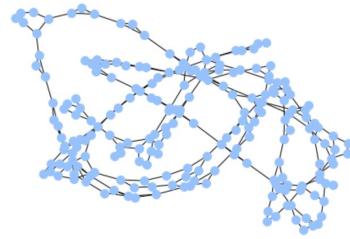


Figura 4.53: Rete mouse brain invertita.

Figura 4.54: Fasi principali di DFNA su rete mouse brain con compressione mista.

Nella Figura 4.55 è possibile osservare il decremento del 92.96% della grandezza della rete, partendo da 213 nodi di arriva alla quarta iterazione con 15 nodi finali che comunque descrivono la struttura sottostante. È stata calcolata la densità della rete finale e risulta essere 1.423.

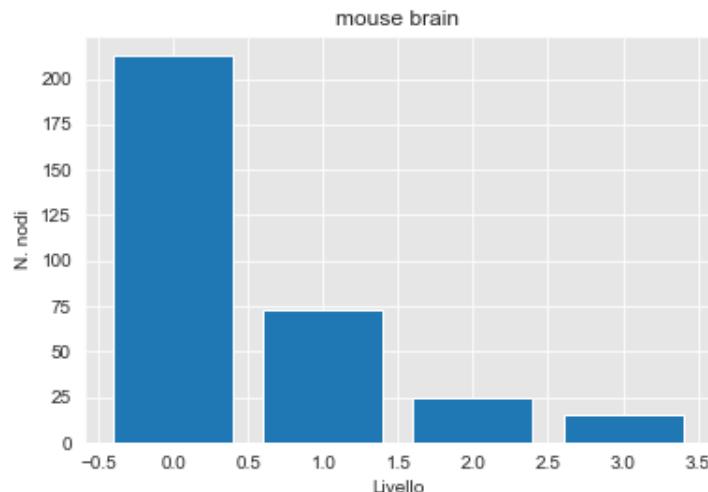


Figura 4.55: Grafico decremento nodi per livello dopo l'applicazione di DFNA su mouse brain.

La rete mouse visual cortex ha una forma rizomatica, che alla fine dell'interazione si addensa in tre principali rizomi (Fig. 4.58).

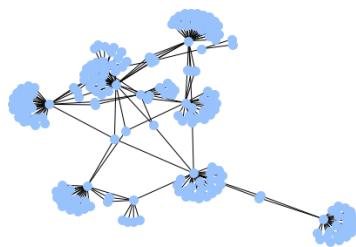


Figura 4.56: Rete mouse visual cortex iniziale.

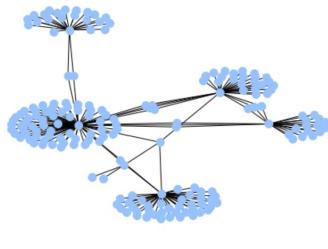


Figura 4.57: Rete mouse visual cortex al terzo livello.

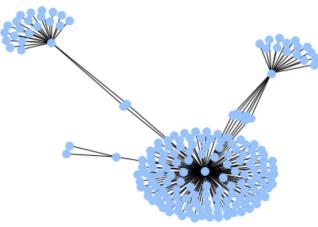


Figura 4.58: Rete mouse visual cortex finale.

Figura 4.59: Fasi principali di DFNA su rete mouse visual cortex con compressione mista.

Dalla Figura 4.60 è evidente il lieve decremento provocato da DFNA in questa rete, con solo 7.25% di nodi compressi, che ha comunque portato ad una riorganizzazione della rete. La densità finale è pari a 0.012 e la distribuzione del grado è rappresentata in Figura 4.61.

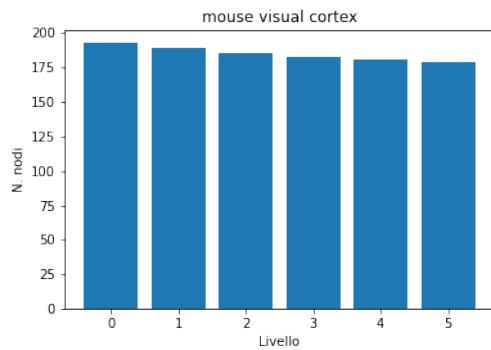


Figura 4.60: Grafico decreimento nodi per livello dopo l'applicazione di DFNA su mouse visual cortex.

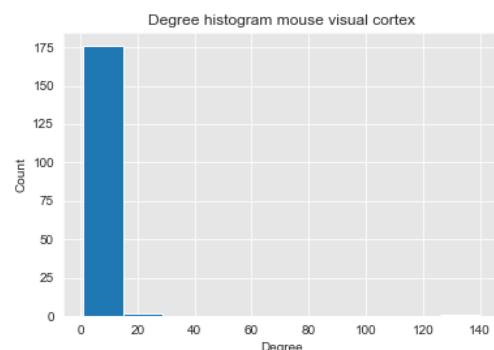


Figura 4.61: Distribuzione del grado della rete mouse visual cortex finale.

Figura 4.62: Statistiche per la rete mouse visual cortex.

Per la rete macaque rhesus brain vengono rappresentate tre delle nove iterazioni totali di DFNA. In nove livelli il decremento dei nodi è pari a 81.82%, da una grandezza di 198 nodi a 44.

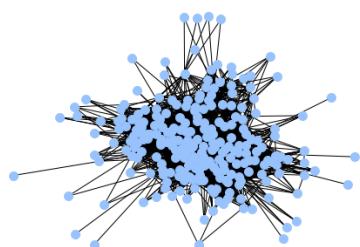


Figura 4.63: Rete macaque rhesus brain iniziale.

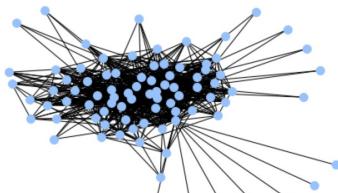


Figura 4.64: Rete macaque rhesus brain al terzo livello.

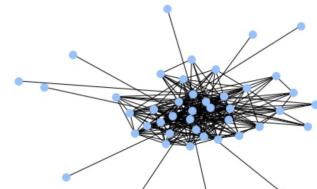


Figura 4.65: Rete macaque rhesus brain finale.

Figura 4.66: Fasi principali di DFNA su rete mouse visual cortex con compressione mista.

La densità finale calcolata è pari a 0.301. Il coefficiente di clustering finale è pari a 0.574.

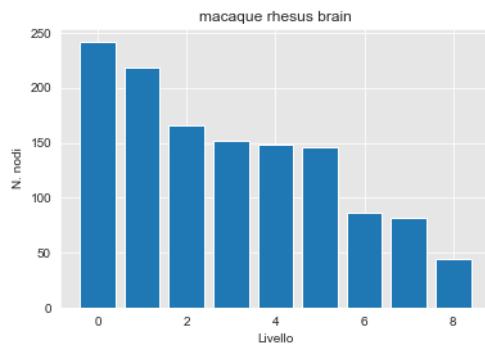


Figura 4.67: Grafico decreimento nodi per livello dopo l'applicazione di DFNA su macaque rhesus brain.

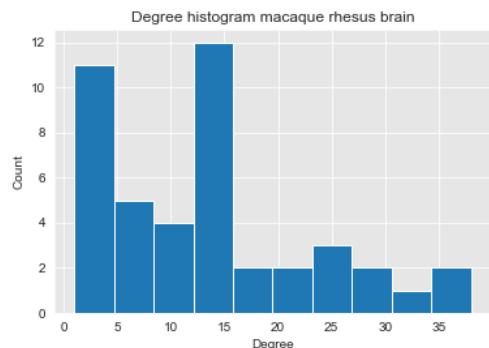


Figura 4.68: Distribuzione del grado della rete macaque rhesus brain finale.

Figura 4.69: Statistiche per la rete macaque rhesus brain.

Applicazione DFNA su Ecology Network Andando ad analizzare i risultati di DFNA sulla classe di reti ecologiche, notiamo come di base queste siano simili nella struttura. Partendo da mangwet nel giro di sette iterazioni è stata trasformata la rete in nodo singolo.

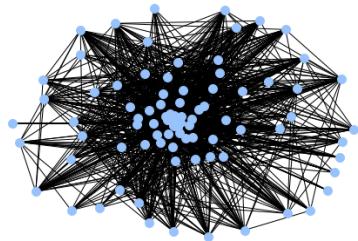


Figura 4.70: Rete mangwet iniziale.

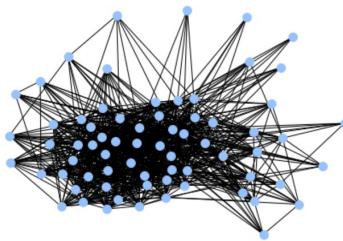


Figura 4.71: Rete mangwet al terzo livello.

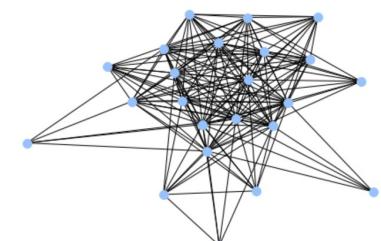


Figura 4.72: Rete mangwet al quarto livello.

Figura 4.73: Fasi principali di DFNA su rete mangwet con compressione mista.

Il decremento dei nodi è massimo, del 98.97% appunto perché partendo da 97 nodi siamo arrivati alla rete finale composta da un singolo nodo.

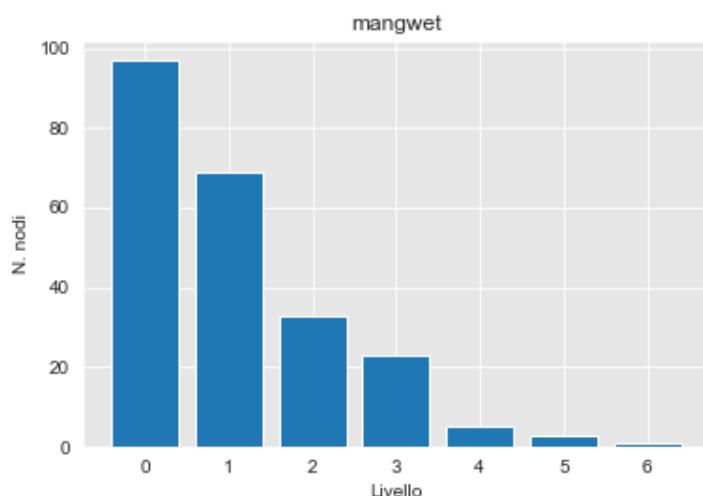


Figura 4.74: Grafico decreimento nodi per livello dopo l'applicazione di DFNA su mangwet.

Per la rete foodweb i risultati sono simile alla rete mangwet, con una rete finale composta da una coppia di nodi, decremento del 98.44% da 128 a 2 soli nodi in sei iterazioni, come si evince in Figura 4.79.

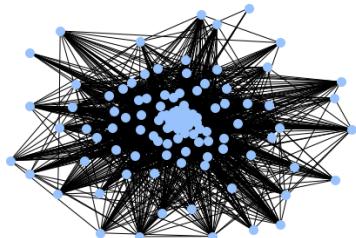


Figura 4.75: Rete foodweb iniziale.

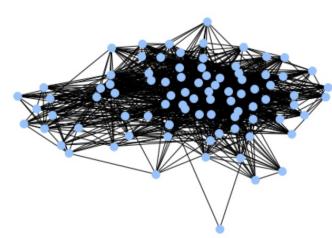


Figura 4.76: Rete foodweb al terzo livello.

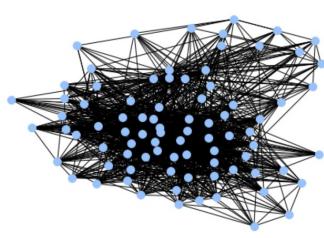


Figura 4.77: Rete foodweb al quarto livello.

Figura 4.78: Fasi principali di DFNA su rete foodweb con compressione mista.

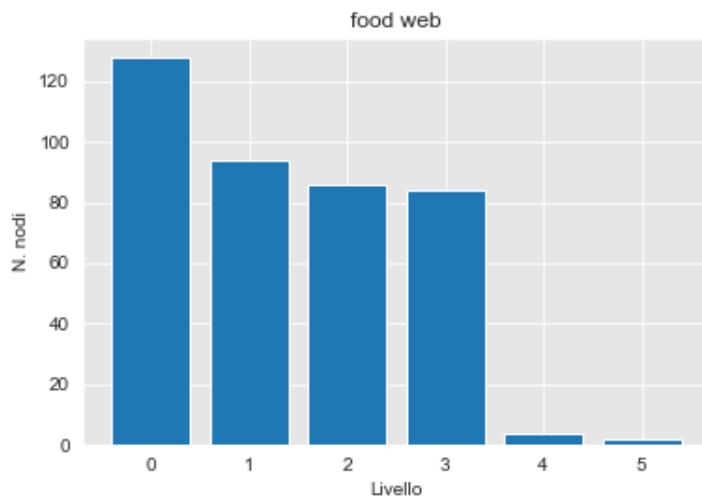


Figura 4.79: Grafico decreimento nodi per livello dopo l'applicazione di DFNA su foodweb.

La rete ecologica everglades ha anch'essa un comportamento simile alle precedenti: il decremento del 98.55% porta ad una rete finale di un singolo nodo.

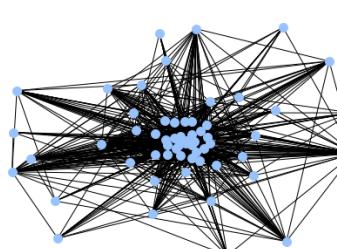


Figura 4.80: Rete everglades iniziale.

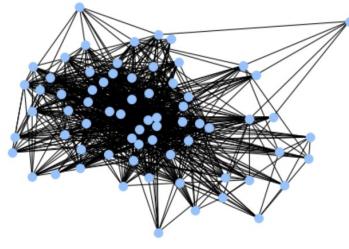


Figura 4.81: Rete everglades al secondo livello.

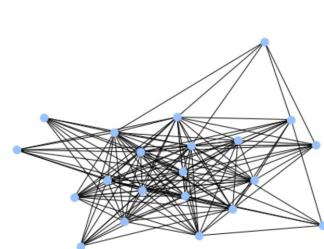


Figura 4.82: Rete everglades al terzo livello.

Figura 4.83: Fasi principali di DFNA su rete everglades con compressione mista.

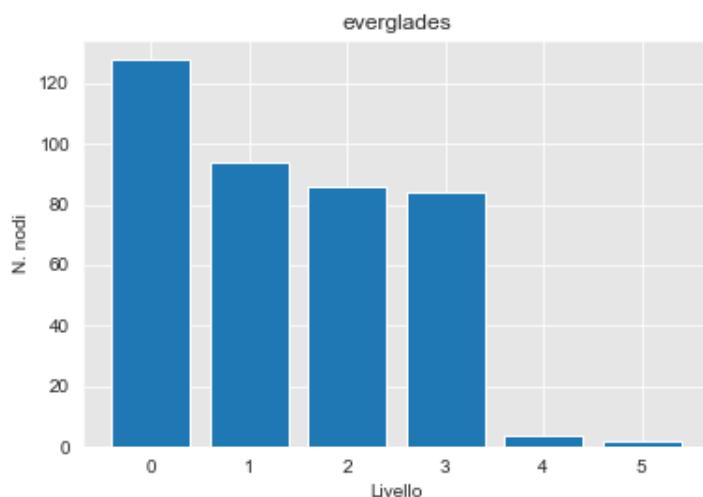


Figura 4.84: Grafico decreimento nodi per livello dopo l'applicazione di DFNA su everglades.

Applicazione DFNA su Chemical Network Per quest'ultima classe di reti chimiche i risultati sono stati differenti rispetto alla classe precedente. Per la rete elegans dopo 15 iterazioni il decremento è del 43.71%, che ha portato da 453 nodi a 255.

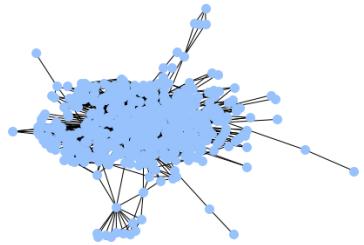


Figura 4.85: Rete elegans iniziale.

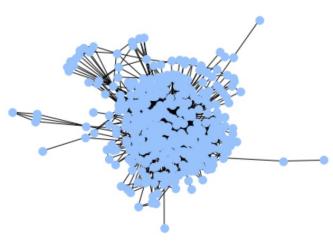


Figura 4.86: Rete elegans al terzo livello.

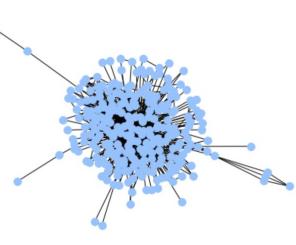


Figura 4.87: Rete elegans finale.

Figura 4.88: Fasi principali di DFNA su rete elegans con compressione mista.

Essendo la rete finale popolosa, sono stati calcolati anche densità, pari a 0.028 e coefficiente di clustering medio, pari a 0.688.

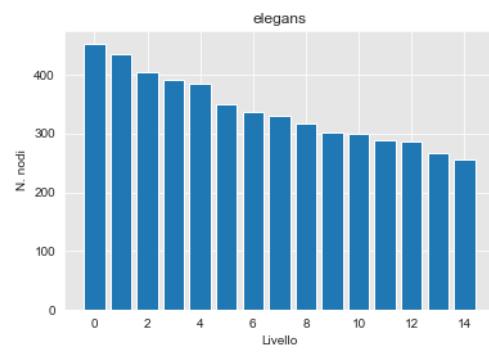


Figura 4.89: Grafico decreimento nodi per livello dopo l'applicazione di DFNA su elegans.

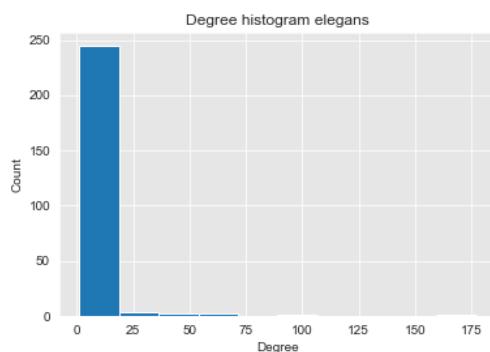


Figura 4.90: Distribuzione del grado della rete elegans finale.

Figura 4.91: Statistiche per la rete elegans.

Per la rete CE-LC DFNA al ventiduesimo livello converge con un indice di decremento pari a 14.28%, portando la rete da 1387 nodi a 1195.

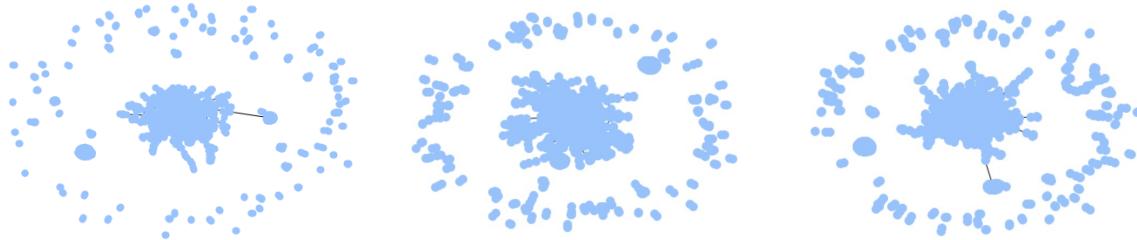


Figura 4.92: Rete CE-LC iniziale. Figura 4.93: Rete CE-LC al terzo livello. Figura 4.94: Rete CE-LC finale.

Figura 4.95: Fasi principali di DFNA su rete CE-LC con compressione mista.

La densità della rete finale è pari a 0.002 e il coefficiente di clustering medio 0.0016.

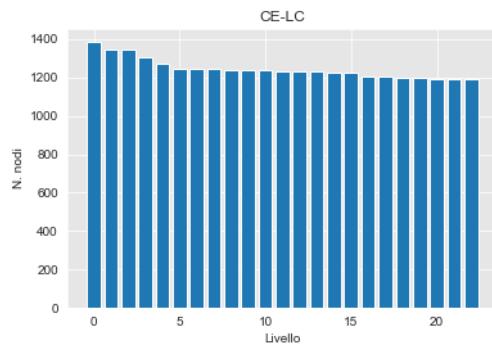


Figura 4.96: Grafico decremento nodi per livello dopo l'applicazione di DFNA su CE-LC.

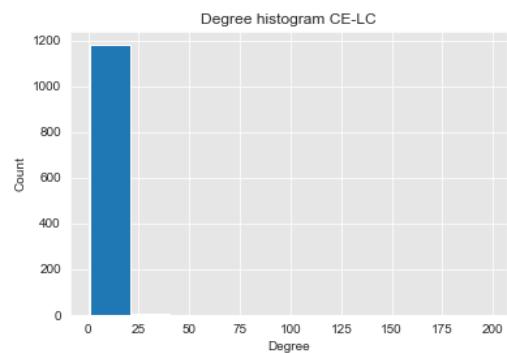


Figura 4.97: Distribuzione del grado della rete CE-LC finale.

Figura 4.98: Statistiche per la rete CE-LC.

Infine, la rete CE-GT in undici iterazioni diminuisce del 21.43%, da 924 nodi a 726. La densità finale è pari a 0.006 e il coefficiente di clustering medio 0.48.

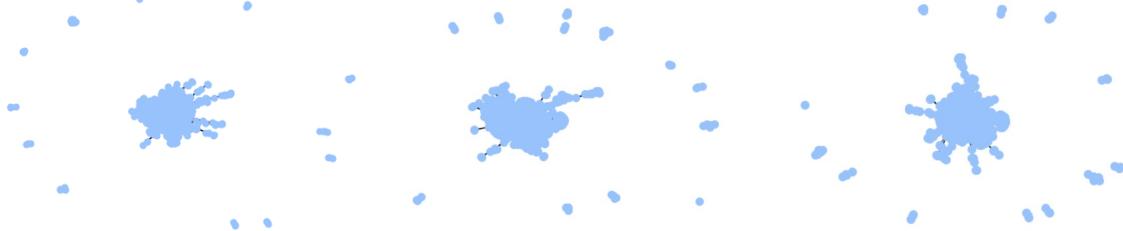


Figura 4.99: Rete CE-GT iniziale.
Figura 4.100: Rete CE-GT al terzo livello.
Figura 4.101: Rete CE-GT finale.

Figura 4.102: Fasi principali di DFNA su rete CE-GT con compressione mista.

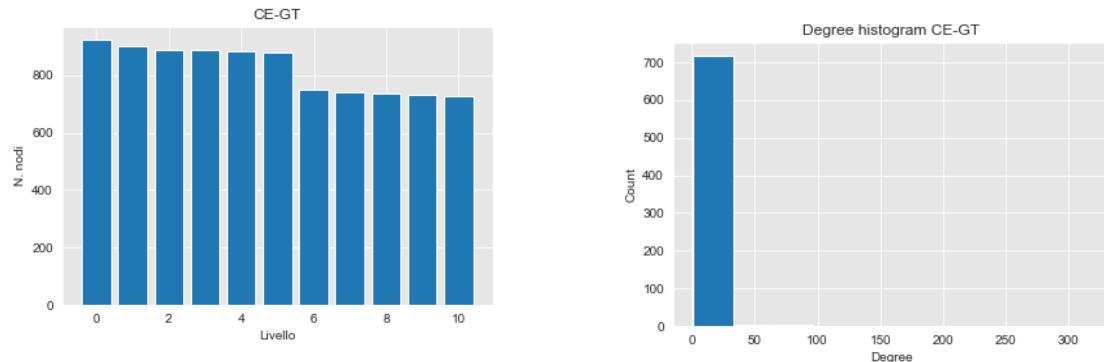


Figura 4.103: Grafico decremento nodi per livello dopo l'applicazione di DFNA su CE-GT.

Figura 4.104: Distribuzione del grado della rete CE-GT finale.

Figura 4.105: Statistiche per la rete CE-GT.

4.2.3 Commento sui risultati degli esperimenti

Sono stati eseguiti gli esperimenti su quattro classi di reti diverse e si è notato che la procedura DFNA restituisce risultati unici per ogni tipo di rete. Le reti ecologiche e chimiche hanno una simile risposta alla compressione, questo fa sì possano essere definite omogenee anche per le reti risultanti da DFNA. Le reti neurali rispondono anch'esse bene alla compressione con motif triangolare e sono più frattali delle altre reti testate. In particolare, nella rete mouse visual cortex, è stata scoperta una struttura ricorsiva rizomatica che persiste nonostante la compressione.

Capitolo 5

Conclusioni

In quest'ultimo capitolo presentiamo le conclusioni della nostra analisi, evidenziando i risultati ottenuti. Vengono inoltre proposti i possibili sviluppi futuri per il lavoro proposto in questa tesi. È stato proposto un algoritmo *nature-inspired* per l'analisi di network. Prima di ciò è stato necessario un esteso studio della letteratura sul fenomeno naturale che ha ispirato il nostro algoritmo e l'analisi dell'attuale stato dell'arte degli algoritmi ispirati alla natura. Lo studio dello stato dell'arte è stato inoltre approfondito con nozioni di teoria dei frattali e studio delle reti, utili per trarre un filo che colleghi i tre argomenti principali in questa tesi. È stata definita poi la procedura *Daucus Fractal Network Analyzer*, risultato dell'analisi teorica e dell'astrazione dal fenomeno naturale preso in considerazione. La metodologia proposta si è posta come obiettivo la semplificazione di reti con l'ipotesi di identificare pattern ricorsivi al loro interno.

DFNA è una procedura aperta e prototipale, volta a rappresentare il fatto che da un fenomeno presente in natura si sia arrivati ad un algoritmo. In questa tesi è stata presentata e sviluppata un'idea, in questo caso basata a ricostruire algoritmamente la struttura di un fiore. È stata sviluppata una procedura in versione prototipale che presenta delle criticità. Queste criticità sono dei punti di partenza per dei risvolti futuri, che possono essere estesi e ampliati in una versione più robusta e performante di DFNA.

Si è notato che la procedura DFNA inversa ricostruisce soltanto un'approssi-

mazione della rete originale. Questo può essere migliorato cambiando l'encoding dei livelli, in modo da mantenere più informazione per una futura ricostruzione. La decompressione giustifica in ogni caso la bontà della procedura: nonostante la perdita di informazione data dalla compressione si riesce a ricostruire un'approssimazione del grafo di partenza.

Inoltre tra i contributi è stata realizzata una dashboard ad hoc per la rappresentazione dei risultati. La dashboard è stata implementata con python e la libreria streamlit ed è reperibile a questo indirizzo: <https://share.streamlit.io/federikovi/tesi-dashboard/main/app.py>.

5.1 Esperimenti futuri

Dati botanici Un aspetto che non è stato approfondito in questa tesi è l'utilizzo di dati inerenti a fenomeni botanici. Inizialmente nel presente lavoro di tesi si era ipotizzato di fare uno studio analitico su dati rappresentanti caratteristiche di piante infestanti. Oltre al già noto iris dataset, dataset di piccole dimensioni utilizzato per scopi didattici, ad oggi non sono presenti dati che descrivano infiorescenze di piante come la *Daucus Carota*. L'assenza di questo tipo di dati botanici potrebbe essere colmata con l'utilizzo di tecniche di *Computer Vision* per avere una banca dati sull'infiorescenza delle piante ombrellifere.

DFNA per interscambio di dati La procedura DFNA può essere vista come mezzo per interscambio di dati, per il trasferimento di un grafo di grandi dimensioni o di dati sensibili tramite compressione e decompressione. Questo potrebbe essere uno sviluppo interessante del lavoro proposto finora, perché DFNA si può utilizzare per la compressione di grafi essendo invertibile. Riesplodendo il grafo con la numerazione fatta e sapendo il pattern, si riesce a ricostruire il grafo originale approssimato.

Pruning La procedura DFNA può essere utilizzata per il pruning di un grafo togliendo i nodi che sono rumore, cioè i nodi frattali.

Altri algoritmi nature-inspired La procedura DFNA può essere lo spunto iniziale per altri algoritmi ispirati alla natura. Prendendo come esempio la specie di uccelli *Himantopus himantopus*, l'uccello acquatico conosciuto anche come Cavaliere d'Italia, il suo comportamento per la ricerca di cibo lascia delle tracce non lineari e curiose. Le tracce lasciate sulla laguna barenile dai Cavalieri d'Italia alla ricerca di cibo potrebbero essere utilizzate come approssimazione di un comportamento di ricerca brute force.

Google Summer of Code Sono stata ammessa per il Google Summer of Code, un programma estivo remoto e globale in cui Google connette studenti e organizzazioni impegnate nell'open source. L'organizzazione con cui svolgo il progetto del GSoC è Open Bioinformatics Foundation (OBF), una fondazione che si occupa di promuovere lo sviluppo open source all'interno della community di ricerca biologica. Questa tesi ha sollevato dell'interesse dei membri di OBF che mi hanno invitata a esporre i risultati in una presentazione che avrà luogo il 9 Luglio 2021.

Ringraziamenti

Questa tesi è la continuazione di un viaggio. Senza il quale non avrei scoperto rami forti e robusti all'interno di me stessa su cui fare leva nei momenti di vacuità. Durante i 1500 km di pedalata da Delft a Saccagnana dopo il semestre Erasmus, nelle prime tre settimane di Agosto 2019, i fiori di Daucus Carota sparsi lungo la strada hanno catturato la mia attenzione. S'infiltravano misteriosamente gettando le basi per questo studio.

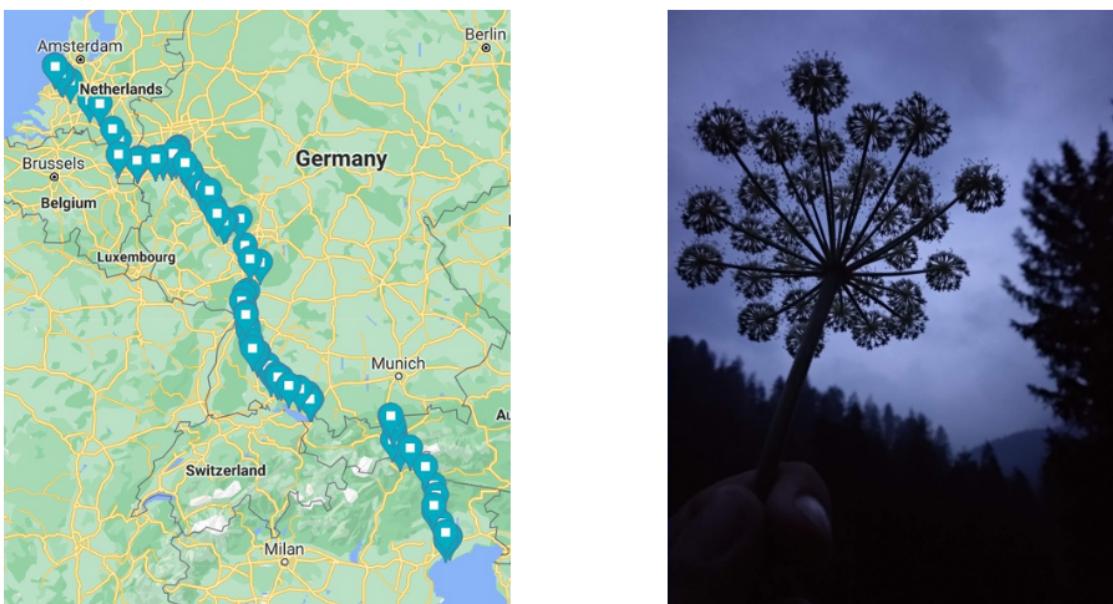


Figura 5.1: A sinistra, il tragitto in bici di 1500km da Delft a Saccagnana. A destra, un esemplare di Daucus Carota trovato nel Cadore, vicino alle Alpi Carniche.

Ringrazio mia sorella Sara per avermi proposto questa avventura e per aver attraversato le Alpi con me. Ringrazio mia sorella Alessia per averci aspettato e tranquillizzato i nostri genitori durante questa impresa. Ringrazio le mie colonne cariatidi, mia mamma e mio papà, per quando mi hanno accompagnata a Piazzale Roma dopo la Regata Storica nel 2017, poco prima

di iscrivermi a questa magistrale. E per tutto quello che fanno.

Un ringraziamento speciale va al professore Giulio Rossetti per essere stato il mio relatore, che ha accettato un'idea embrionale e mi ha aiutata a farla fiorire.

Come ha scritto Rovelli in Helgoland: *a tutte queste persone, che insieme sono la magica rete di relazioni, di cui questo libro è un filo.*

Elena per gli audio eterni in cui ripeto algoritmica. Chiara che si ricorda i miei impegni più di me. Massimo, un valido compagno da sempre, anche per questa fase finale della tesi passata in compagnia. Giulia e gli axolotl. Gabriele per il sostegno costante a colpi di trash, Silvia per le evasioni alla Biennale, Beatrice per i giri a Livorno e Andrea per i concerti memorabili.

Dedico questa tesi ai miei nipoti Aurelio, Cesare e Luca, alle ramificazioni, ai fili sottili e alle onde una a una. A tutto quello che verrà.

Venezia, giugno 2021.

Bibliografia

- [1] Yuval Noah Harari e Giuseppe Bernardi. *Da animali a dèi: breve storia dell'umanità*. Bompiani Milano, 2014.
- [2] Leandro Nunes de Castro. «Fundamentals of natural computing: an overview». In: *Physics of Life Reviews* 4.1 (2007), pp. 1–36.
- [3] Anthony Brabazon e Michael O'Neill. «Natural Computing in Computational Finance (Volume 2): Introduction». In: *Natural Computing in Computational Finance*. Springer, 2009, pp. 1–5.
- [4] Agoston E Eiben, James E Smith et al. *Introduction to evolutionary computing*. Vol. 53. Springer, 2003.
- [5] Pakize Erdogmus e Fatih Kayaalp. «Introductory Chapter: Clustering with Nature-Inspired Optimization Algorithms». In: *Introduction to Data Science and Machine Learning*. IntechOpen, 2020.
- [6] Douglas R Hofstadter et al. *Gödel, escher, bach*. Harvester press Hassocks, 1979.
- [7] Marco Dorigo, Mauro Birattari e Thomas Stutzle. «Ant colony optimization». In: *IEEE computational intelligence magazine* 1.4 (2006), pp. 28–39.
- [8] Seyedali Mirjalili, Seyed Mohammad Mirjalili e Andrew Lewis. «Grey wolf optimizer». In: *Advances in engineering software* 69 (2014), pp. 46–61.
- [9] Xin-She Yang e Xingshi He. «Bat algorithm: literature review and applications». In: *International Journal of Bio-inspired computation* 5.3 (2013), pp. 141–149.
- [10] M Maccioni. «Le ombrellifere: la carota-(*Daucus Carota L.*)» In: *Rivista della Società Toscana di Orticoltura* 31.1/2 (1946), pp. 20–28.
- [11] Ada Eljiva Georgia. *A Manual of Weeds: With Descriptions of All the Most Pernicious and Troublesome Plants in the United States and Canada, Their Habits of Growth and Distribution, with Methods of Control*. Macmillan, 1914.
- [12] Antonio Targioni-Tozzetti. *Cenni storici sulla introduzione di varie piante nell'agricoltura ed orticoltura toscana*. Tipografia M. Ricci, 1896.

- [13] Ehrenfried E Pfeiffer. *Che cosa raccontano le erbe infestanti*. Ed. Antroposofica, 1989.
- [14] Adàn José-García e Wilfrido Gòmez-Flores. «Automatic clustering using nature-inspired metaheuristics: A survey». In: *Applied Soft Computing* 41 (2016), pp. 192–213.
- [15] Ali Reza Mehrabian e Caro Lucas. «A novel numerical optimization algorithm inspired from weed colonization». In: *Ecological informatics* 1.4 (2006), pp. 355–366.
- [16] Aritra Chowdhury, Sandip Bose e Swagatam Das. «Automatic clustering based on invasive weed optimization algorithm». In: *International Conference on Swarm, Evolutionary, and Memetic Computing*. Springer. 2011, pp. 105–112.
- [17] Roozbeh Razavi-Far, Vasile Palade e Enrico Zio. «Invasive weed classification». In: *Neural Computing and Applications* 26.3 (2015), pp. 525–539.
- [18] Benoit B Mandelbrot e Benoit B Mandelbrot. *The fractal geometry of nature*. Vol. 1. WH freeman New York, 1982.
- [19] Benoit Mandelbrot e Roberto Pignoni. *Gli oggetti frattali: forma, caso e dimensione*. Einaudi, 1995.
- [20] Przemyslaw Prusinkiewicz e Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.
- [21] Gregg Hartvigsen. «The analysis of leaf shape using fractal geometry». In: *The American Biology Teacher* 62.9 (2000), pp. 664–669.
- [22] Nino Arsov e Georgina Mirceva. «Network Embedding: An Overview». In: *arXiv preprint arXiv:1911.11726* (2019).
- [23] Hirokatsu Kataoka et al. «Pre-training without Natural Images». In: *Proceedings of the Asian Conference on Computer Vision*. 2020.
- [24] Richard Dryden. «Modelling Processes in a fractal network: a possible substructure for consciousness.» In: () .
- [25] José S Andrade Jr et al. «Apollonian networks: Simultaneously scale-free, small world, Euclidean, space filling, and with matching graphs». In: *Physical review letters* 94.1 (2005), p. 018702.

- [26] Jordan K. Matelsky et al. «DotMotif: An open-source tool for connectome subgraph isomorphism search and graph queries». In: (2020). doi: 10.1101/2020.06.08.140533. URL: <http://dx.doi.org/10.1101/2020.06.08.140533>.
- [27] Aida Mrzic et al. «Grasping frequent subgraph mining for bioinformatics applications». In: *BioData mining* 11.1 (2018), pp. 1–24.
- [28] Rafael Espejo et al. «Exploiting graphlet decomposition to explain the structure of complex networks: the GHuST framework». In: *Scientific reports* 10.1 (2020), pp. 1–14.
- [29] Danielle P Mersch, Alessandro Crespi e Laurent Keller. «Tracking individuals shows spatial fidelity is a key regulator of ant social organization». In: *Science* 340.6136 (2013), pp. 1090–1093.
- [30] Josh A Firth e Ben C Sheldon. «Experimental manipulation of avian social structure reveals segregation is carried over across contexts». In: *Proceedings of the Royal Society B: Biological Sciences* 282.1802 (2015), p. 20142350.
- [31] Stefanie Gazda et al. «The importance of delineating networks by activity type in bottlenose dolphins (*Tursiops truncatus*) in Cedar Key, Florida». In: *Royal Society Open Science* 2.3 (2015), p. 140263.
- [32] Katrin Amunts et al. «BigBrain: An Ultrahigh-Resolution 3D Human Brain Model». In: *Science* 340.6139 (2013), pp. 1472–1475.
- [33] Robert E Ulanowicz e Donald L DeAngelis. «Network analysis of trophic dynamics in south florida ecosystems». In: *FY97: The Florida Bay Ecosystem* (1998), pp. 20688–20038.
- [34] Carlos J Melián e Jordi Bascompte. «Food web cohesion». In: *Ecology* 85.2 (2004), pp. 352–358.
- [35] J. Duch e A. Arenas. «Community identification using Extremal Optimization Phys». In: *Rev. E* 72 (2005), p. 027104.
- [36] Ara Cho et al. «WormNet v3: a network-assisted hypothesis-generating server for *Caenorhabditis elegans*». In: *Nucleic acids research* 42.W1 (2014), W76–W82.

[37] Sebastian Seung. *Connectome: How the brain's wiring makes us who we are.*
HMH, 2012.

Sitografia

- [38] *Introduzione al Natural Computing*. Visitato il 29-03-2021. URL: <https://leganerd.com/2016/10/16/introduzione-al-natural-computing/>.
- [39] *Vocabolario Treccani*. Visitato il 28-02-2021. URL: <https://www.treccani.it/vocabolario/>.
- [40] *Fractals and Scaling*. Visitato il 30-04-2021. URL: <https://www.complexityexplorer.org/courses/118-fractals-and-scaling>.
- [41] *Connected Papers*. Visitato il 12-04-2021. URL: <https://www.connectedpapers.com/>.
- [42] *Network Science*. Visitato il 12-04-2021. URL: <http://networksciencebook.com/>.
- [43] *World Wide Web size*. Visitato il 12-04-2021. URL: <https://www.worldwidewebsize.com/>.
- [44] *Dunbar's number: Why we can only maintain 150 relationships*. Visitato il 01-03-2021. URL: <https://www.bbc.com/future/article/20191001-dunbars-number-why-we-can-only-maintain-150-relationships>.
- [45] *Unlimited computer fractals can help train AI to see*. Visitato il 3-04-2021. URL: <https://www.technologyreview.com/2021/02/04/1017486/fractals-ai-learn-see-more-ethically-bias-imagenet-training/>.
- [46] Ryan A. Rossi e Nesreen K. Ahmed. *The Network Data Repository with Interactive Graph Analytics and Visualization*. Visitato il 30-05-2021. 2015. URL: <http://networkrepository.com>.

Elenco delle figure

1.1	Immagine comparativa: a sinistra un'illustrazione di DC consultabile in <i>Medical Botany</i> (1832) di William Woodville, a destra l'iterazione iniziale e finale della procedura DFNA su una rete reale.	5
2.1	La tassonomia del <i>Natural Computing</i> (NC) secondo De Castro.	9
2.2	Tassonomia degli algoritmi <i>nature-inspired</i> secondo De Castro.	12
2.3	Illustrazione di <i>Daucus Carota</i> , <i>Medical Botany</i> (1832) di William Woodville.	20
2.4	La tassonomia della sintesi e simulazione dei fenomeni naturali secondo De Castro.	25
2.5	Linea, quadrato e cubo con <i>magnification factor</i> pari a tre. .	27
2.6	Iterazioni per costruzione della curva di Koch.	28
2.7	Iterazioni per costruzione del Triangolo di Sierpiński.	29
2.8	Dimensione Daucus Carota calcolata tramite box counting.	34
2.9	Paper connessi ad Ant Colony Optimization.	39
3.1	Grafo di partenza G.	48
3.2	Motif da identificare nel grafo G.	48
3.3	Diagramma <i>Daucus Fractal Network Analyzer</i>	50
3.4	30 possibili combinazioni di grafi di grandezza da due a cinque nodi.	52
3.5	Grafo utilizzato per la demo	55
3.6	Grafo di partenza G.	55
3.7	Prima iterazione della compressione singola.	55
3.8	Seconda iterazione della compressione singola.	55

3.9	Demo compressione singola.	55
3.10	Grafo di partenza G.	56
3.11	Prima iterazione compressione mista.	56
3.12	Seconda e ultima iterazione compressione mista.	56
3.13	Demo compressione mista.	56
3.14	Grafo di partenza G.	57
3.15	Prima iterazione compressione frattale.	57
3.16	Seconda e ultima iterazione compressione frattale.	57
3.17	Demo compressione frattale.	57
3.18	Tempi d'esecuzione DFNA	59
3.19	Grafo G finale	60
3.20	Ricostruzione dell'approssimazione del grafo G.	60
3.21	Demo DFNA inversa.	60
4.1	Rete insecta ant colony.	64
4.2	Rete aves wildbird.	64
4.3	Rete mammalia dolphin.	64
4.4	Classe di reti <i>Animal Social Network</i>	64
4.5	Iistogramma insecta ant colony.	65
4.6	Iistogramma aves wildbird.	65
4.7	Iistogramma mammalia dolphin.	65
4.8	Iistogrammi di distribuzione del grado per la classe di reti <i>Animal Social Network</i>	65
4.9	Rete mouse brain.	66
4.10	Rete mouse visual cortex.	66
4.11	Rete macaque rhesus brain.	66
4.12	Classe di reti <i>Brain Network</i>	66
4.13	Iistogramma mouse brain.	67
4.14	Iistogramma mouse visual cortex.	67
4.15	Iistogramma macaque rhesus brain.	67
4.16	Iistogrammi di distribuzione del grado per la classe di reti <i>Brain Network</i>	67

4.17	Rete mangwet.	68
4.18	Rete foodweb.	68
4.19	Rete everglades.	68
4.20	Classe di reti <i>Ecology Network</i>	68
4.21	Iistogramma mangwet.	68
4.22	Iistogramma foodweb.	68
4.23	Iistogramma everglades.	68
4.24	Iistogrammi di distribuzione del grado per la classe di reti <i>Ecology Network</i>	68
4.25	Rete elegans.	69
4.26	Rete CE-LC.	69
4.27	Rete CE-GT.	69
4.28	Classe di reti chimiche.	69
4.29	Iistogramma elegans.	69
4.30	Iistogramma CE-LC.	69
4.31	Iistogramma CE-GT.	69
4.32	Iistogrammi di distribuzione del grado per la classe di reti chimiche.	69
4.33	Motif e compressione su rete insecta ant colony al terzo livello	71
4.34	Rete insecta ant colony iniziale.	73
4.35	Rete insecta ant colony al terzo livello.	73
4.36	Rete insecta ant colony invertita.	73
4.37	Fasi salienti di DFNA su rete insecta ant colony con compressione mista.	73
4.38	Grafico raffigurante il decremento dei nodi per livello dopo l'applicazione di DFNA su insecta ant colony.	74
4.39	Rete aves wildbird iniziale.	74
4.40	Rete aves wildbird al terzo livello.	74
4.41	Rete aves wildbird invertita.	74
4.42	Fasi principali di DFNA su rete aves wildbird con compressione mista.	74

4.43	Grafico del decremento dei nodi per livello dopo l'applicazione di DFNA su <i>aves wildbird</i>	75
4.44	Rete <i>mammalia dolphin</i> iniziale.	75
4.45	Rete <i>mammalia dolphin</i> al secondo livello.	75
4.46	Rete <i>mammalia dolphin</i> invertita.	75
4.47	Fasi principali di DFNA su rete <i>mammalia dolphin</i> con compressione mista.	75
4.48	Grafico decremento nodi per livello dopo l'applicazione di DFNA su <i>mammalia dolphin</i>	76
4.49	Distribuzione del grado della rete <i>mammalia dolphin</i> finale. .	76
4.50	Statistiche per la rete <i>mammalia dolphin</i>	76
4.51	Rete <i>mouse brain</i> iniziale.	77
4.52	Rete <i>mouse brain</i> al terzo livello.	77
4.53	Rete <i>mouse brain</i> invertita.	77
4.54	Fasi principali di DFNA su rete <i>mouse brain</i> con compressione mista.	77
4.55	Grafico decremento nodi per livello dopo l'applicazione di DFNA su <i>mouse brain</i>	77
4.56	Rete <i>mouse visual cortex</i> iniziale.	78
4.57	Rete <i>mouse visual cortex</i> al terzo livello.	78
4.58	Rete <i>mouse visual cortex</i> finale.	78
4.59	Fasi principali di DFNA su rete <i>mouse visual cortex</i> con compressione mista.	78
4.60	Grafico decremento nodi per livello dopo l'applicazione di DFNA su <i>mouse visual cortex</i>	78
4.61	Distribuzione del grado della rete <i>mouse visual cortex</i> finale. .	78
4.62	Statistiche per la rete <i>mouse visual cortex</i>	78
4.63	Rete <i>macaque rhesus brain</i> iniziale.	79
4.64	Rete <i>macaque rhesus brain</i> al terzo livello.	79
4.65	Rete <i>macaque rhesus brain</i> finale.	79

4.66	Fasi principali di DFNA su rete <code>mouse visual cortex</code> con compressione mista.	79
4.67	Grafico decremento nodi per livello dopo l'applicazione di DFNA su <code>macaque rhesus brain</code>	79
4.68	Distribuzione del grado della rete <code>macaque rhesus brain</code> finale.	79
4.69	Statistiche per la rete <code>macaque rhesus brain</code>	79
4.70	Rete <code>mangwet</code> iniziale.	80
4.71	Rete <code>mangwet</code> al terzo livello.	80
4.72	Rete <code>mangwet</code> al quarto livello.	80
4.73	Fasi principali di DFNA su rete <code>mangwet</code> con compressione mista.	80
4.74	Grafico decremento nodi per livello dopo l'applicazione di DFNA su <code>mangwet</code>	80
4.75	Rete <code>foodweb</code> iniziale.	81
4.76	Rete <code>foodweb</code> al terzo livello.	81
4.77	Rete <code>foodweb</code> al quarto livello.	81
4.78	Fasi principali di DFNA su rete <code>foodweb</code> con compressione mista.	81
4.79	Grafico decremento nodi per livello dopo l'applicazione di DFNA su <code>foodweb</code>	81
4.80	Rete <code>everglades</code> iniziale.	82
4.81	Rete <code>everglades</code> al secondo livello.	82
4.82	Rete <code>everglades</code> al terzo livello.	82
4.83	Fasi principali di DFNA su rete <code>everglades</code> con compressione mista.	82
4.84	Grafico decremento nodi per livello dopo l'applicazione di DFNA su <code>everglades</code>	82
4.85	Rete <code>elegans</code> iniziale.	83
4.86	Rete <code>elegans</code> al terzo livello.	83
4.87	Rete <code>elegans</code> finale.	83

4.88	Fasi principali di DFNA su rete elegans con compressione mista.	83
4.89	Grafico decremento nodi per livello dopo l'applicazione di DF-NA su elegans.	83
4.90	Distribuzione del grado della rete elegans finale.	83
4.91	Statistiche per la rete elegans.	83
4.92	Rete CE-LC iniziale.	84
4.93	Rete CE-LC al terzo livello.	84
4.94	Rete CE-LC finale.	84
4.95	Fasi principali di DFNA su rete CE-LC con compressione mista.	84
4.96	Grafico decremento nodi per livello dopo l'applicazione di DF-NA su CE-LC.	84
4.97	Distribuzione del grado della rete CE-LC finale.	84
4.98	Statistiche per la rete CE-LC.	84
4.99	Rete CE-GT iniziale.	85
4.100	Rete CE-GT al terzo livello.	85
4.101	Rete CE-GT finale.	85
4.102	Fasi principali di DFNA su rete CE-GT con compressione mista.	85
4.103	Grafico decremento nodi per livello dopo l'applicazione di DF-NA su CE-GT.	85
4.104	Distribuzione del grado della rete CE-GT finale.	85
4.105	Statistiche per la rete CE-GT.	85
5.1	A sinistra, il tragitto in bici di 1500km da Delft a Saccagnana. A destra, un esemplare di Daucus Carota trovato nel Cadore, vicino alle Alpi Carniche.	89

Elenco delle tabelle

2.1 Calcolo della dimensione per linea, quadrato e cubo.	27
3.1 Effetto di rinomina dei nodi.	51
3.2 Schema della memorizzazione dei nodi per la ricostruzione in DFNA inversa	60
4.1 Schema riassuntivo delle dodici reti, divise in quattro classi, su cui sono stati effettuati gli esperimenti.	63
4.2 Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di animal social network.	65
4.3 Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di brain network.	67
4.4 Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di ecology network.	69
4.5 Densità, cammino minimo medio e coefficiente di clustering medio per il gruppo di reti chimiche.	70

List of Algorithms

1	Standard evolutionary algorithm	13
2	Standard genetic algorithm	14
3	Ant Colony Optimization	18
4	Grandiso algorithm	47
5	Daucus Fractal Network Analyzer - DFNA	49
6	DFNA Rename	51
7	DFNA Reversed	54