Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Implementazione e analisi di un modello di traffico per lo studio della dinamica veicolare in città

Relatore:
Prof. Armando Bazzani

Correlatore:
Dott. Lorenzo Di Meco

Presentata da:
Lorenzo Rizzi

Anno Accademico 2023/2024

Sommario

Obiettivo di questo lavoro è la realizzazione e successiva analisi di un modello dinamico che simuli il traffico e la formazione di congestioni stradali all'interno di un città omogenea. Il modello, implementato tramite codice C++, si basa sul paradigma OD (Origine-Destinazione), in cui si assumono casualmente le origini e le destinazioni per ogni veicolo che popola il network urbano. Gli utenti della strada si portano da un estremo all'altro del loro tragitto prescelto seguendo un percorso che minimizzi il tempo totale del viaggio e che eviti, laddove possibile, code e ingorghi stradali. La formazione di congestioni è fortemente legata alle interazioni esistenti fra veicoli. Per rendere conto di questa influenza reciproca, il modello prescrive l'aggiunta di meccanismi di non-linearità alla dinamica fondamentale degli agenti. Ad esempio, ogni strada ammette una capienza massima, superata la quale non permette l'accesso ad altri veicoli, e possiede un limite massimo al flusso in uscita. La parte di analisi dati si concentra sulle condizioni necessarie perché il sistema subisca una transizione di fase da uno stato stazionario di traffico scorrevole (*free-flow*) ad uno di congestione della rete (*gridlock*), in cui ingorghi e code impediscono il movimento degli agenti. Verranno inoltre approfondite le proprietà del sistema nel regime di free-flow e in intorni del punto critico, ad esempio la distribuzione delle lunghezze delle code nelle strade, il flusso totale del network e il tempo medio di percorrenza.

Indice

Introduzione

Definire con precisione cosa sia un *sistema complesso* è un'operazione tutt'altro che facile ed immediata, come forse già suggerisce il nome. Giorgio Parisi dice che un sistema complesso *"è un sistema del quale si può parlare a lungo"* [1], un sistema la cui evoluzione non è affatto banale e difficile da predire in anticipo. Per essere più specifici, possiamo definire un sistema complesso come un sistema a più agenti la cui evoluzione temporale dipende principalmente dalla struttura del network di interazione tra le particelle e non dalla natura delle stesse o dal particolare tipo di interazione.

Il concetto di *complessità* si oppone ad uno delle speranze più sacrosante e radicate nella mente di molti fisici, e cioè il riduzionismo. Si tratta dell'assunto teorico secondo il quale il comportamento di un sistema composto da molti elementi è interamente noto e compreso quando è conosciuto il comportamento dei suoi costituenti fondamentali. Un diavoletto di Laplace che conoscesse lo stato dinamico di ogni particella di fluido in un recipiente e possedesse una capacità di calcolo senza precedenti, avrebbe in sè la conoscenza certa ed inequivocabile sull'evoluzione della massa di fluido in esame.

Sfortunatamente, questo non è sempre possibile: spesso il tutto è molto di più rispetto alla semplice somma delle sue parti. Le interazioni non lineari, a lungo ignorate ed accantonante dalla fisica tradizionale alle quali la scienza della complessità ha saputo ridare importanza e centralità, legano i costituenti essenziali di un sistema e producono, a scala macroscopica, quella straordinaria (ma complessa) ricchezza fenomenologica che un sistema lineare "semplice" non potrà mai vantare. Un sistema complesso è dunque spesso caratterizzato dal possedere *proprietà emergenti* ([2],[3]), ovvero comportamenti caratteristici che non possono essere facilmente derivati a scala microscopica.

La fisica dei sistemi complessi si ripropone di analizzare questi sistemi utilizzando i metodi della meccanica statistica, della teoria dei sistemi dinamici e di tanti altri framework teorici già ben assodati, ricorrendo al formalismo proprio della matematica. È una disciplina molto trasversale, poiché origina dalla fisica ma si applica con facilità a problemi provenienti da tanti altri ambiti disciplinari accomunati. Non dobbiamo dunque stupirci del fatto che esista una ricca letteratura scientifica dedicata allo studio del traffico veicolare urbano da un punto di vista fisico-statistico ([4],[5],[6],[7]). Il traffico, inteso come insieme di veicoli in reciproca interazione, è un chiaro esempio di sistema complesso. In un contesto urbano, gli autoveicoli interagiscono fra di loro in maniera altamente non-

lineare e non-newtoniana: questo rende incredibilmente difficile integrare analiticamente il flusso globale del sistema-città partendo dalla dinamica microscopica delle singole vetture [8]. Proprio come in termodinamica, è impossibile ottenere predizioni globali su di un sistema partendo dalle leggi del moto esatte dei singoli 10^{23} atomi che lo compongono. Si potrebbe quindi fare affidamento, continuando l'analogia con la termodinamica, ai metodi della meccanica statistica, che valutano proprietà medie del sistema. Tuttavia una città, per quanto grande, è composta da un numero finito di veicoli e finita è anche la lunghezza delle strade, e questo porta con sè alcune implicazioni sconosciute alla meccanica statistica classica dove il ricorso al limite termodinamico elimina ogni possibile rumore statistico. Tutto ciò pone nuovi problemi allo studio degli equilibri stazionari di un processo stocastico o dinamico. Un'imprevedibile piccola perturbazione nel traffico locale potrebbe espandersi a macchia d'olio all'interno del tessuto cittadino, mettendo in ginocchio l'intera rete stradale. Proprietà, queste, che dunque emergono in maniera non banale dall'interazione fra automobilisti e che sono analiticamente difficili da isolare.

In questo lavoro, dunque, ci riproponiamo di studiare la dinamica di traffico veicolare in un network urbano realistico attraverso simulazione numerica, creando un modello adeguato implementato in codice C++ che tenga conto delle proprietà salienti relative alla mobilità urbana. Il modello che viene impiegato a tal fine è detto *modello OD (origin-destination)*. Si parte innanzitutto dal creare la rete urbana su cui si muoveranno i veicoli che si intende simulare (*gli agenti*). Una volta istanziato correttamente il network, tramite codice, viene creato un quantitativo costante N di agenti all'interno del network scegliendo casualmente per ciascuno di essi una strada di origine (dove nascono) e una strada di destinazione (dove vogliono arrivare). Partendo dall'origine, dunque, gli agenti si muovono nel network seguendo una dinamica ben precisa con l'obiettivo di raggiungere la loro destinazione. Tale dinamica è regolata da tre meccanismi principali:

- Gli agenti viaggiano dall'origine alla destinazione scegliendo un percorso che minimizzi una funzione di costo, che in questo specifico caso è interpretata come il tempo di percorrenza. Ogni strada ha un suo tempo di percorrenza proprio (dovuto alla sua lunghezza intrinseca) a cui si aggiunge dinamicamente un malus dovuto al congestionamento del nodo. In altre parole, più una strada ospita agenti, minore sarà la velocità media degli agenti che la percorrono, maggiore sarà il suo tempo di percorrenza ([9]). Gli agenti tengono conto di queste considerazioni all'atto della loro creazione e scelgono un percorso dentro il network che colleghi l'origine con la destinazione che minimizzi il tempo di percorrenza

- La simulazione avviene a tempo discreto. Per ogni passo di integrazione Δt, solo una certo numero costante di veicoli riesce a transitare da una strada all'altra (esiste un flusso massimo). Soprattutto, non risolviamo la dinamica del veicolo oltre tale scala temporale.

- Ogni strada può ospitare veicoli fino ad una certa soglia. Quando una strada arriva a tale limite, essa non può più far entrare altri agenti, i quali rimangono nelle loro strade di provenienza finché il nodo non viene decongestionato.

Maggiori dettagli saranno forniti nel capitolo pertinente.

A questo punto, una volta che il modello è stato costruito correttamente, è possibile girare varie simulazione modificando a piacere i parametri del modello, primo fra tutti indubbiamente il *carico di rete* N, cioè il numero di agenti in circolazione. D'altronde, è molto interessante cercare di individuare le condizioni di densità veicolare nella rete che scatenino la formazione di ingorghi ([10]). Ed infatti, uno degli obiettivi principali della simulazione è proprio quello di impiegare il modello OD per caratterizzare e studiare approfonditamente la formazione di congestioni stradali all'interno del network al variare del carico di rete N. Ci si aspetta che, per carichi sufficientemente bassi, il flusso globale rimanga abbastanza scorrevole ed indisturbato e i veicoli tendano a non interagire con gli altri: questo stato macroscopico del sistema è chiamato, nelle letteratura scientifica, *free-flow* ([7]). All'aumentare del numero di agenti in circolazione, le interazioni fra veicoli cominciano ad essere importanti e non possono essere trascurate. Per carichi alti, si formano facilmente congestioni nella rete urbana in cui i veicoli sono completamente bloccati nel traffico. Per come è stato descritto, questo processo di formazione di congestioni stradali ricorda da vicino una transizione di fase fisica.

Una prima sezione del documento è dedicata ad una breve discussione sui metodi matematici utili a descrivere formalmente il problema, e cioè elementi di teoria dei grafi e algoritmi legati a tale struttura dati.

Nel secondo capitolo verrà finalmente introdotto il modello implementato tramite codice C++ spiegandone i dettagli ed approfondendo la dinamica fondamentale.

Nel terzo capitolo, infine, vengono presentati i dati raccolti effettuando svariate simulazioni sul modello con l'obiettivo di caratterizzare la transizione di fase da stato di free-flow a stato congestionato del sistema. Inoltre sarà approfondito il comportamento della rete in entrambi i regimi

Capitolo 1

Elementi di teoria dei grafi

Nell'affrontare un problema fisico, il metodo scientifico prescrive l'utilizzo di modelli matematici per rappresentare gli elementi fisici caratteristici del problema. Per descrivere la dinamica di traffico in una rete urbana più o meno estesa, utilizzeremo alcuni concetti matematici propri della teoria dei grafi. Per questa ragione, dedichiamo questo capitolo ad un'essenziale esposizione di alcuni degli elementi fondamentali di questa teoria matematica.

1.1 Teoria dei grafi

La fisica dei sistemi complessi si ritrova spesso a fare i conti con sistemi a molte componenti (*agenti*) in mutua interazione reciproca. Questa situazioni è ben rappresentabile in termini matematici dal concetto di *grafo*. Formalmente parlando [11], [12]

Definizione 1.1 *Consideriamo un insieme finito V formato da elementi distinti, che possiamo chiamare per convenzione vertici o nodi:*

$$V = \{a, b, c, ...\}$$

Consideriamo poi un insieme E i cui elementi siano coppie di elementi arbitrari di V escludendo però self-loop:

$$E = \{(u, v) : u, v \in V, u \neq v\}$$

tali coppie possono anche essere chiamate archi. Si definisce grafo G la coppia ordinata $G = (V, E)$.

Un grafo è quindi, nella sua forma più essenziale, un insieme di vertici con l'informazione aggiuntiva delle relative connessioni, specificate dall'insieme degli archi E. È chiaro, perciò, che i grafi siano lo strumento matematico d'elezione per tutti quei problemi

pratici in cui compaiano molte componenti diverse fra di loro in un regime di interazione reciproca. Ad ogni componente del sistema è associato un vertice: se due componenti entrano in interazione, allora i due vertici vengono collegati da un arco. Il grafo, nella sua semplicità, permette di rappresentare matematicamente la struttura statica delle interazioni di un sistema a molti agenti.

Un grafo del tipo descritto in Def.1.1, in cui l'insieme degli archi $E(G)$ è a tutti gli effetti un insieme matematico (ogni elemento compare al più una volta) è noto, nel gergo della teoria, col nome di *grafo semplice* o *grafo non orientato*. Rilassando la condizione che gli archi debbano essere unici o che non ne possano esistere di connettenti lo stesso vertice, si ottengono strutture matematiche più complicate, come quella di *multigrafo*, che però non tratteremo in questo documento poiché non sono di utilità pratica nel modello che costruiremo.

Un'ulteriore distinzione che, invece, possiamo fare riguarda la simmetria degli archi. In alcune situazioni pratiche, infatti, le interazioni fra agenti non sono simmetriche rispetto allo scambio di questi ultimi. Ad esempio, possiamo rappresentare con un vertice v una persona e con un arco $v - w$ la relazione binaria "v conosce w": in tal caso, niente garantisce che anche w conosca v. L'interazione, che in questo caso specifico è una relazione di conoscenza, è fortemente direzionale e questa proprietà si riflette anche negli archi che popolano l'insieme E. Un grafo che gode di questa proprietà è detto *grafo orientato* o *digrafo*[1]:

Definizione 1.2 *Un grafo G è detto digrafo o grafo orientato se, nel definire l'insieme E, conta l'ordine con il quale compaiono i vertici:*

$$(u, v) \neq (v, u)$$

Esempi di grafi orientati e non orientati sono mostrati in Fig.1.1. Diamo poi qualche altra definizione per i digrafi che ci tornerà utile in futuro

Definizione 1.3

- *Dato un arco $e = (u, v)$ di un digrafo, allora il vertice di origine u è detto sorgente di e mentre il vertice di arrivo v è detto target di e.*
- *Due vertici v e w sono adiacenti se esiste un arco $e \in E$ che li connetta, $e = (u, v)$ o $e = (v, u)$.*
- *Due archi e e f sono incidenti se il target dell'uno coincide con la sorgente dell'altro: $\exists u, v, w \in V : e = (u, v), f = (v, w)$. In altre parole, posso passare dall'arco e all'arco f sfruttando un solo vertice comune.*

[1] Dall'inglese *directed graph*, contratto in *digraph*

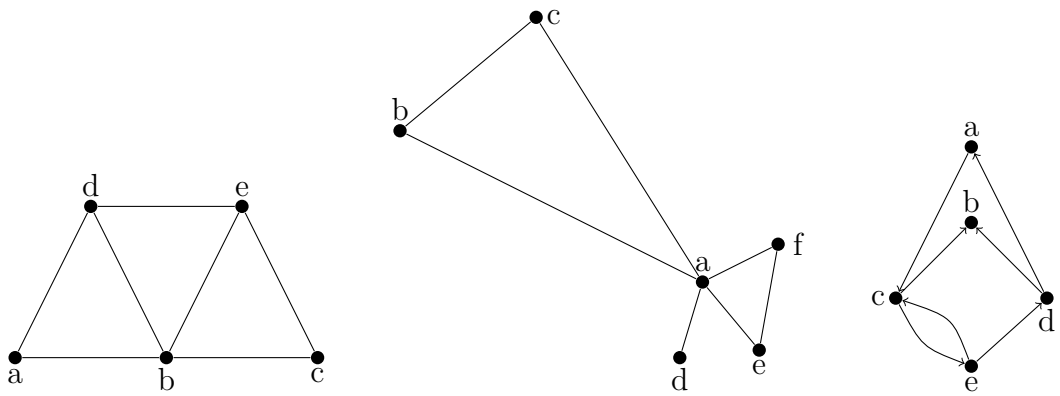

Figura 1.1: *Qualche esempio di grafo. I primi due da sinistra sono grafi non orientati, dunque gli archi connettono simmetricamente due vertici, mentre l'ultimo è orientato e la direzione del legame è specificata dal verso della freccia*

- *Scelto un generico vertice $v \in V$, definiamo il grado in uscita (o outdegree) come il numero di archi la cui sorgente è precisamente v*
- *Analogamente, definiamo il grado in ingresso (o indegree) di un vertice v come il numero di archi il cui target è precisamente v*

1.1.1 Cammini su grafi e algoritmo di Dijkstra

Consideriamo un generico grafo semplice[2] $G = (V, E)$. Definiamo un *cammino* $\{v_1, v_2, \ldots, v_k\}$ come:

Definizione 1.4 *Un cammino nel grafo $G = (V, E)$ è una generica sequenza di vertici a due a due adiacenti $\{v_1, v_2, \ldots, v_k\}$ con $v_i \in V$. Se chiamiamo v_1 origine e v_k destinazione, allora il cammino $\{v_1, v_2, \ldots, v_k\}$ non è altro se non un modo per raggiungere v_k da v_1 sfruttando le connessioni del grafo.*

Dato il concetto di cammino, possiamo a questo punto formalizzare una qualche nozione di connettività. Realizziamo una relazione di equivalenza $\sim$ fra vertici di un grafo G tale per cui $v \sim w$ se e solo se esiste un cammino interno a G con v origine e w destinazione.

Definizione 1.5 *Dato un grafo $G = (V, E)$, le classi di equivalenza secondo la relazione $\sim$ definita in precedenza formano insiemi chiamati componenti connesse di G. Se G è composto da un'unica componente connessa, allora diremo che G è connesso.*

In altre parole, G è connesso se da ogni nodo del grafo è possibile raggiungere un qualsiasi altro vertice rimanendo nel grafo stesso (Fig. 1.2).

[2]In questa sottosezione, considereremo solo grafi semplici non orientati. Tuttavia, i concetti elaborati possono facilmente essere estesi anche ai digrafi, avendo cura di considerare l'orientamento degli archi

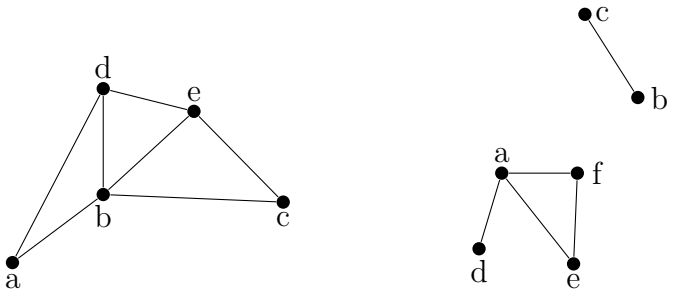

Figura 1.2: *A sinistra, grafo semplice non orientato connesso. A destra, grafo semplice non orientato formato da due componenti connesse,* $\{a, d, e, f\}$ *e* $\{c, b\}$

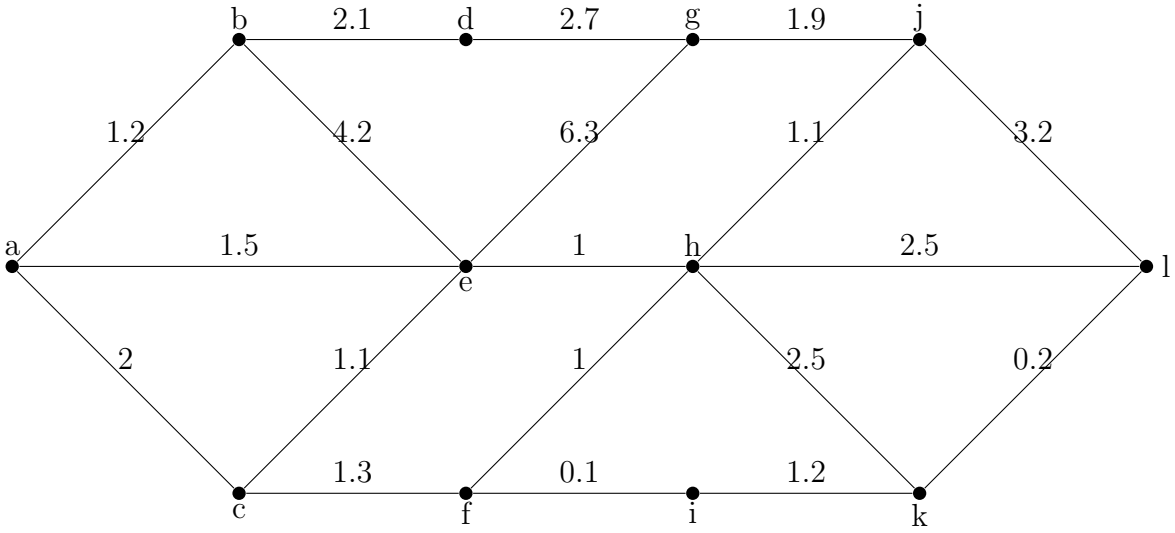

Figura 1.3: *Esempio di grafo pesato su cui eseguire l'algoritmo di Dijkstra, ripreso da [11]*

La *lunghezza semplice* di un cammino $\{v_1, v_2, \ldots, v_k\}$ coincide col numero di vertici attraversati durante il percorso, dunque k. In certe situazioni fisiche, tuttavia, non tutti gli archi sono equivalenti e non tutte le transizione da un vertice all'altro richiedono la stessa "lunghezza". A volte, infatti, è necessario associare una metrica agli archi che indichi la *lunghezza propria* del collegamento fra vertici adiacenti: in tal caso, si parla di *grafi pesati*. Ad ogni arco e, si associa un *peso* $w_e \geq 0$ che rappresenta il peso della connessione. Un esempio di grafo semplice pesato è proposto in Fig.1.3

Proponiamoci ora di calcolare, dati due vertici di origine e di destinazione v, w, il cammino $\{v, \ldots, w\}$ interno al grafo che minimizzi la lunghezza totale del percorso calcolata grazie ai pesi degli archi (ad esempio, in Fig.1.3 per passare da a, il nodo di origine, a l il nodo di termine). A tale scopo, esiste un semplice ed efficace algoritmo,

detto *algoritmo di Dijkstra*, che, dato un grafo pesato con pesi non negativi, permette di calcolare la distanza minima e il percorso associato dall'origine a verso qualsiasi altro nodo del grafo. Gli step dell'algoritmo sono i seguenti [13]:

- Ad ogni nodo, viene associato un'etichetta d_i che rappresenta la distanza minima rispetto al nodo di origine. All'inizio, per tutti i nodi tranne l'origine $d_i = \infty$, poiché l'algoritmo ancora non ha elaborato tutto il network. I nodi del grafo vengono poi divisi in tre categorie: i visitati (V), i nodi di frontiera (F) e gli sconosciuti (S). I primi sono i nodi per i quali l'algoritmo ha già fornito una risposta definitiva circa il cammino minimo e il valore d_i è effettivamente la distanza minima. I nodi di frontiera sono i nodi su cui iterativamente l'algoritmo lavora.

- Si procede selezionando fra i nodi in F quello con la distanza minima, ad esempio v, e lo si etichetta come visitato (V). Si aggiungono all'insieme F i nodi adiacenti a v e, per ciascuno di loro, si aggiorna l'etichetta d_i selezionando il percorso più breve. Ad esempio, se w è adiacente a v, allora la sua etichetta d_w si aggiorna secondo:
$$d_w^{new} = min\{d_w^{old}, d_v + e_{v,w}\} \tag{1.1}$$
dove $e_{v,w}$ è il peso dell'arco $v - w$.

- Si procede così fino ad esaurire i nodi sconosciuti e finché tutta la rete è nell'insieme dei visitati V

Capitolo 2

Modello dinamico di traffico su grafo urbano

In questo capitolo verrà mostrata la struttura e il funzionamento generale del modello implementato in codice C++ grazie al quale vengono effettuate le simulazioni e raccolti i dati necessari per le analisi successive. Il codice è liberamente disponibile presso la repository github `https://github.com/lorenzorizzi17/urban_network.git`. Per una breve e più tecnica spiegazione del codice, si faccia riferimento ad Appendice A.

2.1 Struttura statica del modello

Consideriamo un piccolo reticolo urbano, fatto di strade e incroci, come quello di Fig. 2.1. Possiamo rappresentare la struttura topologica di questo reticolo tramite un grafo convenientemente definito in cui ogni strada è un arco e ogni incrocio è un vertice. Nel creare il modello, faremo alcune assunzioni ragionevoli che riguardano tale struttura statica:

- Tutte le strade possiedono entrambi i sensi di marcia e sono simmetriche rispetto a tale proprietà. In altre parole, il grafo che costruiamo è quindi un digrafo e, per ogni arco (u, v) esiste anche l'arco (v, u). Questo per assicurare omogeneità nel grafo e garantire che ogni strada sia raggiungibile da altri nodi del network (un'unica componente connessa). D'ora in poi, quando parleremo di *strada* o nodo intenderemo un singolo senso di marcia, cioè un preciso arco del grafo
- La dinamica agli incroci verrà discussa dopo in maniera più approfondita. Per ora, menzioniamo solo il fatto che non sono contemplati meccanismi semaforici (semplici incroci stradali, dunque)
- Non esistono cavalcavia o ponti. Il grafo deve essere planare

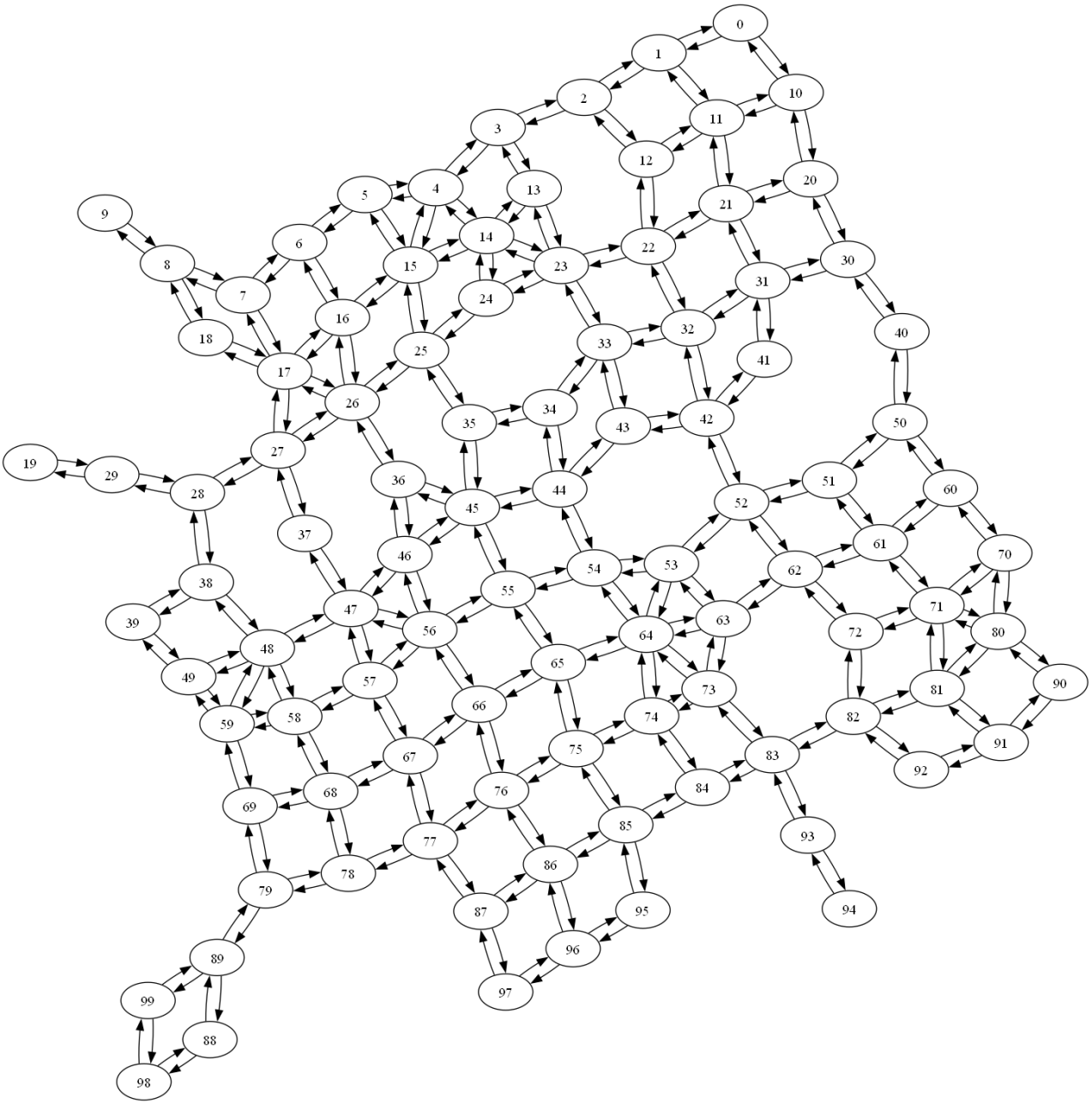

Figura 2.1: *Esempio di grafo che rappresenta il reticolo urbano di strade ed incroci (rappresentazione fisica)*

Ma lo stesso reticolo urbano ammette anche un'altra modellizzazione matematica che ci tornerà molto comoda a seguire. In Fig. 2.1 ogni strada è un arco e ogni incrocio è un nodo; tuttavia la dinamica che ci interessa si svolge principalmente sulle strade. È proprio nelle strade che gli agenti possono rimanere bloccati a causa del traffico ed è qui che transitano per raggiungere la loro destinazione. Essendo le strade dunque i protagonisti della dinamica che vogliamo simulare, conviene rappresentare la cartina stradale associando ad ogni strada un vertice e non un arco. In questa *rappresentazione*, due nodi (strade) v e w sono adiacenti se, nel reticolo urbano fisico, è possibile passare dalla strada v alla strada w tramite un incrocio (Fig. 2.2). Definiamo quindi due diverse *rappresentazioni* per lo stesso modello concreto: quella *fisica* (strada = arco) e quella *duale* (strada = vertice) ([14]). La simulazione viene interamente eseguita lavorando sulla rappresentazione duale, che è il vero protagonista del modello, ma per creare tale struttura è prima necessario costruire la rappresentazione fisica e poi tradurla nella duale. In fondo, è molto più semplice lavorare su tale rappresentazione per inizializzare correttamente le strutture dati utili ai fini della simulazioni. Quello che osserviamo sperimentalmente, in fondo, è il reticolo urbano cittadino ed è più istintivo codificare una strada con un arco: su questa rappresentazione è più naturale porre vincoli di ragionevolezza nel costruire il reticolo (ad esempio, la richiesta di planarità, che viene rotta in maniera non banale nella rappresentazione duale). In altre parole, opereremo la seguente divisione logica:

- La rappresentazione fisica viene impiegata per creare correttamente il network urbano secondo le prescrizione imposte in precedenza. Non solo: proprio perché è la rappresentazione più naturale, viene anche usata per visualizzare l'evoluzione temporale del sistema
- La rappresentazione duale è impiegata ai fini della simulazione. La dinamica che definiremo è interamente operata su questo tipo di grafo

2.1.1 Creazione grafo e modello OD

Si parte, come prima cosa, dalla costruzione del network su cui la simulazione potrà girare. Il programma si occupa di realizzare un grafo planare quadrato perfettamente regolare, stile Manhattan, che codifica un'iniziale modellizzazione del grafo urbano in rappresentazione fisica. Per rendere più realistico il modello, si è ritenuto sufficiente aggiungere casualmente strade diagonali (preservando la planarità del grafo) e rimuovendo interi incroci (come in Fig. 2.1). A questo punto, viene estratta la rappresentazione duale dal grafo randomizzato appena creato, che è il network protagonista del modello che stiamo costruendo (un esempio è riportato in Fig. 2.2). D'ora in poi, faremo riferimento prevalentemente a questo grafo, in cui un vertice è una strada.

Il modello dinamico oggetto di studio in questo documento è un modello di tipo OD (Origin-Destination). Il network appena costruito è popolato di *agenti* (cioè veicoli) ed è proprio dalla loro interazione reciproca e dalla loro dinamica microscopica che emergono

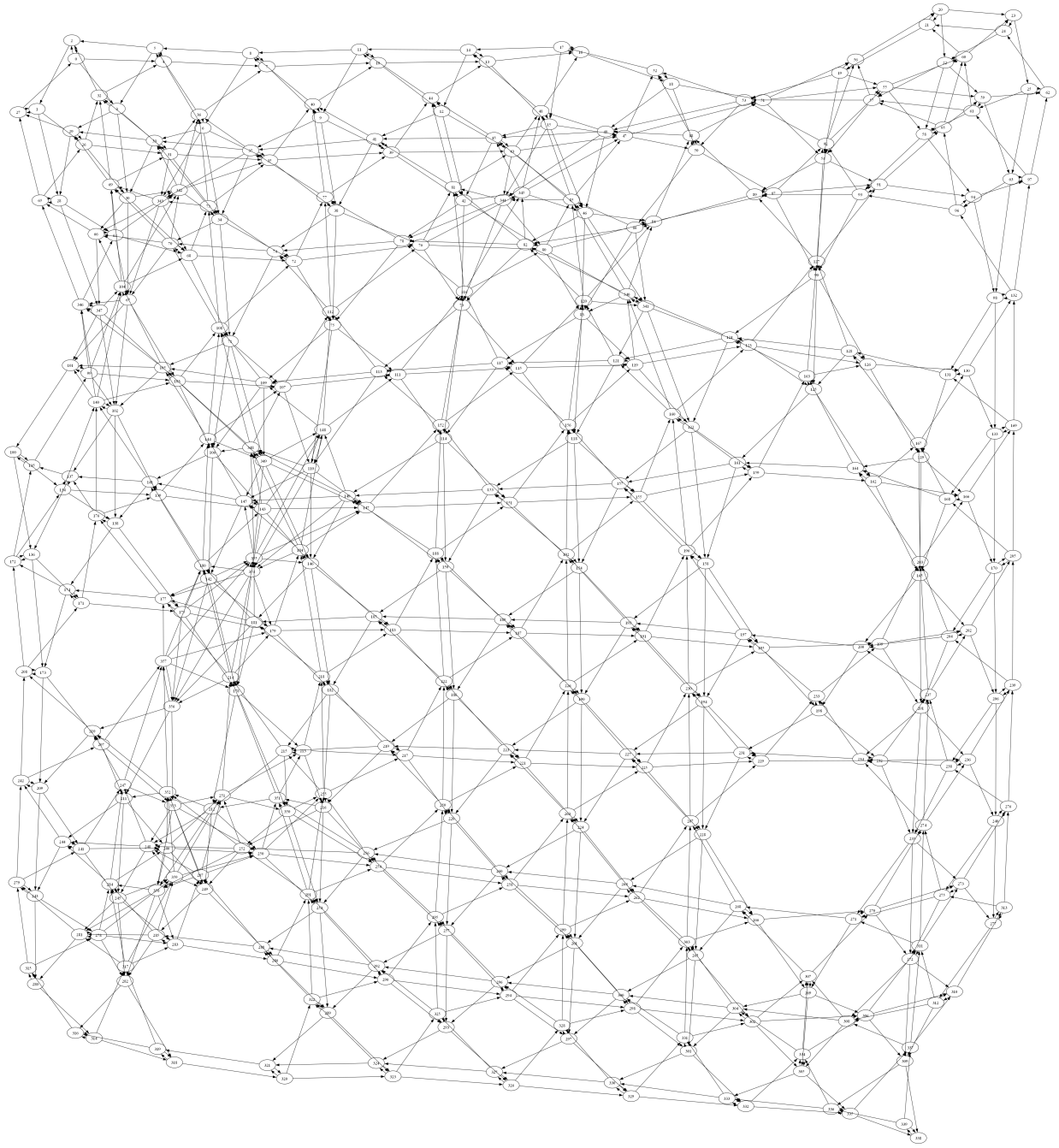

Figura 2.2: *Rappresentazione duale dello stesso reticolo urbano di Fig.2.1. Qui, ogni strada è interpretata come vertice e due vertici sono adiacenti se le sue strade sono in comunicazione (ossia sono incidenti nella rappresentazione fisica). Ogni strada è associata ad un indice, che in questa figura compare al centro dei nodi*

i fenomeni collettivi che osserviamo su scala macroscopica, congestione in primis. Il modello OD per il traffico, di per sè, prescrive che ogni agente nasca in un strada d'origine scelta casualmente all'interno del grafo e che sia diretto verso una destinazione similmente estratta a sorte. L'unico vincolo che poniamo circa queste estrazioni casuali è che origine e destinazione siano sufficientemente distanti, di modo che il viaggio medio che un agente deve percorrere sia superiore ad una certa soglia parametro della simulazione (si è deciso di impostare una distanza minima di 5 nodi, dunque per passare dall'origine alla destinazione devono servire almeno 5 passaggi verso nodi terzi). A questo punto, l'agente si costruisce il percorso nel network che ritiene il migliore (secondo una certa metrica, ad esempio il tempo di percorrenza) e lo percorre fino ad arrivare a destinazione, dove verrà distrutto e lascerà spazio ad un nuovo agente, creato con lo stesso meccanismo.

2.1.2 Assegnazione dinamica dei pesi

Gli agenti si muovono nel grafo seguendo un certo criterio ragionevole, e cioè cercando di minimizzare il tempo totale del loro tragitto. Ad ogni strada è quindi necessario associare un peso w_i che servirà agli agenti per decidere il percorso più vantaggioso dati due punti nel grafo. In particolare, richiediamo che tale peso dipenda dallo stato del network (e quindi dal tempo) secondo:

$$w_i(t) = w_i^0 + f(n_i(t)) \tag{2.1}$$

dove $f(n_i(t))$ è una funzione abbastanza regolare nel numero di agenti che al tempo t stazionano nella strada i e w_i^0 è una quantità costante che rappresenta il peso intrinseco della strada (interpretabile come il peso della strada in condizioni di percorribilità massima, senza cioè alcuna vettura sulla strada: $w_i^0 = w_i(n_i = 0)$). Il peso intrinseco w_i^0 per ogni strada viene impostato seguendo una distribuzione uniforme centrata in 1, $U(0.9, 1.1)$. La funzione $f(n_i(t))$ rappresenta quindi un malus di percorrenza dovuta alla presenza di altre vetture nella stessa strada. Nel modello in esame, si è scelta la funzione:

$$f(n_i) = \frac{1}{1000}(n_i - 1)^4 \tag{2.2}$$

che ha un andamento simile a quello riportato in Fig. 2.3. Notiamo innanzitutto che tale funzione non è lineare nel suo argomento: se n_i è piccolo (circa compreso fra 1 e 7), la percezione che un conducente ha sulla percorribilità della strada non cambia di molto rispetto alla situazione in cui $n_i = 0$. Quando invece n_i supera una certa soglia, la percezione del grado di congestione della strada aumenta vertiginosamente in funzione dell'occupazione di quest'ultima. Quando $n_i = 15$, che è numericamente uguale alla capacità massima di ogni strada, allora la strada è fortemente penalizzata e gli agenti cercheranno di evitarla. Tale non-linearità nei processi umani di decisione è riconosciuta in letteratura scientifica ([15])

A questo punto, dato un generico stato della rete $\vec{n}(t) = (n_1(t), n_2(t), \ldots, n_M(t))$ con n_i occupazione del nodo i-esimo, è noto anche lo stato dei pesi del grafo $\vec{w} =$

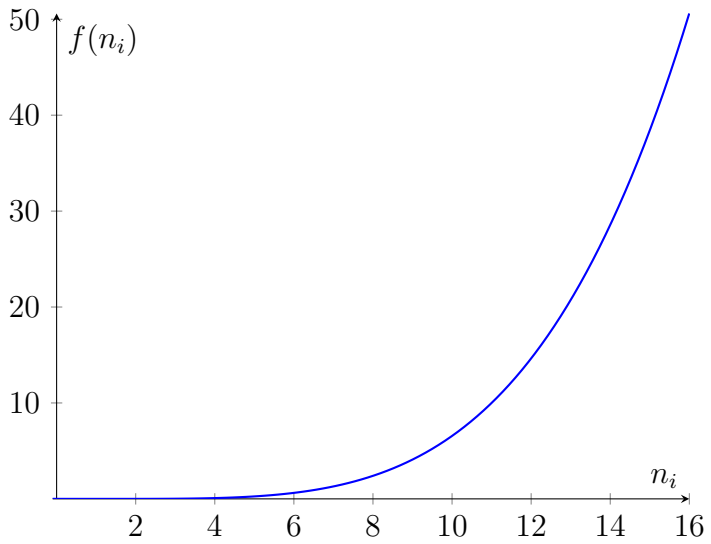

Figura 2.3: *Andamento grafico della funzione di costo $f(n_i)$ in relazione al numero di agenti di un vertice*

$(w_1(t), w_2(t), \dots) = f(\vec{n})$. Questo set di pesi dinamicamente definiti è fondamentale perché gli agenti possano ricostruire il percorso che compiranno per raggiungere la loro destinazione. All'atto della loro creazione, e solo in questo istante, ogni agente impiega l'algoritmo di Dijkstra, di cui abbiamo parlato in precedenza, per ricostruire il tragitto più conveniente che minimizzi il peso dei nodi percorsi.

È necessario sottolineare un aspetto molto importante del modello così costruito: gli agenti scelgono la strada che più conviene loro *solo nell'istante della loro creazione*. Gli agenti possiedono dunque un certo livello di intelligenza e conoscenza globale dello stato della rete, poiché nel momento della loro partenza decidono la strada migliore anche in base alle condizioni istantanee del traffico. Il loro meccanismo di scelta, seppur globale, non è però variabile nel tempo: una volta selezionato un percorso che si ritiene preferibile, questo non può più essere modificato. Se, nel frattempo, il cammino prescelto diventa improvvisamente congestionato, gli agenti non impiegheranno meccanismi di ricalcolo del tragitto

2.2 Dinamica modello OD

Procediamo ora a stabilire la regola dinamica che governa il comportamento degli agenti nel network.

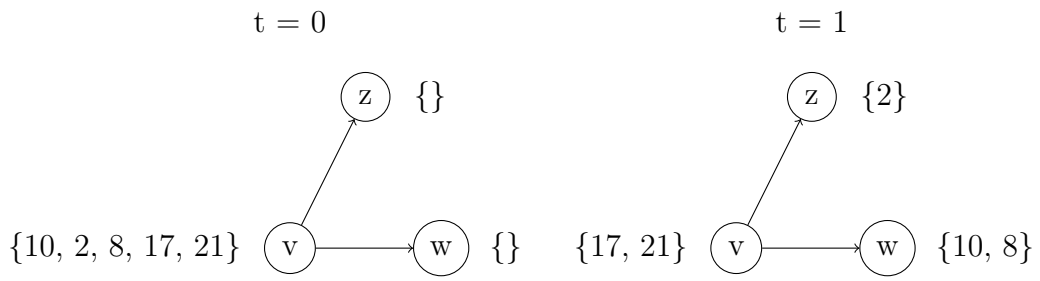

Figura 2.4: *Esempio di funzionamento del meccanismo di flusso massimo con parametro* $\Phi = 3$. *Al tempo* $t = 1$, *nel nodo* v *ci sono* 5 *agenti indicizzati dal loro identificativo. Al prossimo istante temporale, solo* $\Phi = 3$ *dei* 5 *veicoli è riuscito a fluire verso gli altri nodi (logica FIFO).*

2.2.1 Flusso massimo in uscita

La simulazione procede per intervalli temporali discretizzati $\Delta t = 1$ (unità arbitrarie). Ad ogni step temporale, gli agenti cercando di avanzare nel reticolo urbano seguendo la lista di vertici stabilita inizialmente tramite Dijkstra. Tuttavia, ci si aspetta che se un certo nodo è particolarmente congestionato, allora gli agenti che vi abitano siano più lenti e potrebbero impiegare più tempo per svoltare verso il prossimo vertice. Per ricreare questo effetto, si è deciso di imporre un limite massimo Φ al flusso in uscita per ogni nodo (che, per quasi tutte le simulazioni eseguite, è pari a $\Phi = 3$). Questo significa che:

- Se in un certo nodo al tempo intero t è presente un numero di agenti $n_i(t) \leq \Phi$, allora nell'istante temporale successivo $t + \Delta t$ tutti gli agenti riescono a fluire uscendo dal nodo e possono continuare nel loro tragitto inserendosi in nuovi vertici.
- Se, invece, al tempo t vale che $n_i(t) > \Phi$, allora solo Φ agenti riescono ad abbandonare effettivamente il nodo e ad entrare in altri vertici. I restanti $n_i(t) - \Phi$ veicolo rimangono nel nodo e dovranno aspettare il prossimo step temporale per continuare il loro percorso. Vale la pena ricordare che le code nelle strade sono ragionevolmente gestite secondo un meccanismo FIFO (first in, first out), dunque agenti appena entrati in un nodo saranno gli ultimi ad uscirne (Fig. 2.4).

2.2.2 Capienza massima e meccanismi multi-corsia in ingresso

Specifichiamo ora come gli agenti gestiscono l'ingresso verso nuovi vertici. Impostiamo il parametro N_{max} che rappresenta la capacità massima di un nodo: quando un vertice supera tale soglia $n_i \geq N_{max}$, allora tale nodo risulta saturo e non potrà far entrare altri agenti al suo interno i quali dovranno dunque pazientare nei loro nodi di provenienza.

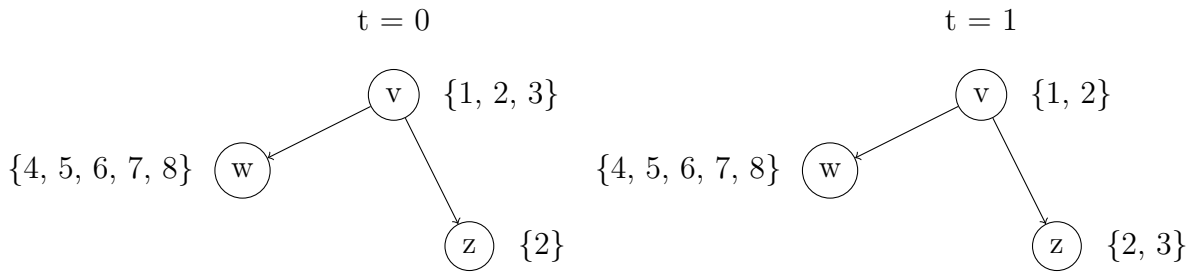

Figura 2.5: *Esempio di funzionamento del meccanismo di capienza massima e di multi-corsia per l'afflusso di veicoli nel network ($N_{max} = 5$).*

Per semplicità, imponiamo che tutti i nodi abbiano la stessa capienza massima fissata (in quasi tutte le simulazioni, salvo indicazione contraria, poniamo $N_{max} = 15$, quindi al massimo 15 veicoli).

La dinamica di movimento degli agenti è svolta in maniera *sincrona*. In altre parole, ad ogni step temporale Δt tutti gli agenti che possono muoversi considerato il particolare stato della rete $\vec{n}(t)$ lo faranno in maniera simultanea. Questo vuol dire, ad esempio, che possono crearsi situazioni in cui $n_i > N_{max}$. Immaginiamo, ad esempio, che nell'istante temporale t il nodo i-esimo non sia ancora saturo ma sia molto vicino alla soglia massima. Due altri nodi puntano verso tale vertice ed ospitano agenti che, nel prossimo step temporale, vorrebbero transitare proprio per i. Poiché i non è saturo al tempo t, al tempo $t + \Delta t$ tutti gli agenti che vogliono accedere a tale nodo potranno farlo, nonostante alla fine dello step evolutivo l'occupazione del nodo superi la soglia N_{max}. Al prossimo intervallo temporale, il nodo verrà considerato saturo e non potrà far entrare nuovi agenti, al massimo farne fluire di interni. Non esiste, dunque, una dinamica interna ai singoli Δt e gli agenti decidono se possono o meno accedere ad un nodo semplicemente valutando lo stato della rete al tempo t, senza considerare le intenzioni di altri veicoli.

Nel modello in esame, implementiamo anche una dinamica multi-corsia. Per spiegare tale meccanismo, facciamo riferimento a Fig. 2.5. Nel nodo v, sono presenti 3 agenti, di cui i primi due hanno intenzione di proseguire il loro tragitto verso il nodo w e l'ultimo verso il nodo z. Se impostiamo, ad esempio, $N_{max} = 5$, allora il nodo w è da considerare saturo e nessun altro agente può transitarvi. Gli agenti 1 e 2, di conseguenza, devono rimanere nel nodo v finché w non verrà decongestionato. Sotto queste condizioni, l'agente 3 può sorpassare i veicoli che lo precedono in coda e raggiungere tranquillamente il nodo z, che invece non è pieno.

Capitolo 3

Analisi dei risultati ottenuti

Dopo aver approfondito nel capitolo precedente il funzionamento e la struttura del modello di traffico in esame, lasciamo che la simulazione giri in autonomia e operiamo raccogliendo dati sul comportamento del sistema. Come per ogni modello, la dipendenza in input è data dai parametri della simulazione, e cioè dalle quantità numeriche che impostano una specifica regola dinamica. Nel codice realizzato, vi sono molti parametri che controllano svariati aspetti della simulazione, ma il più importante dal punto di vista fisico è indubbiamente il *carico di rete* N, ossia il numero di utenti che popolano il network urbano.

3.1 Comportamento macroscopico del sistema: regime di free-flow

Fissiamo alcuni parametri secondari della simulazione a valori costanti (ad esempio, il flusso massimo $\Phi = 3$ e la capienza massima $N_{max} = 15$) e andiamo ad osservare come si comporta il sistema variando il carico di rete N. Riconosciamo, in base al solo comportamento macroscopico del sistema, due diversi regimi o *fasi*. Il primo, per valori di N abbastanza bassi, almeno fino a $N < 2000$, è detto di *free-flow*: il sistema, dopo un iniziale transiente dovuto alla configurazione iniziale degli agenti[1], si porta verso uno stato di equilibrio dinamico. Ogni agente riesce a raggiungere la sua destinazione finale e, anche qualora un nodo venga saturato con $n_i = N_{max}$, tale condizione non dura a lungo e la congestione viene prontamente riassorbita dalla rete. Possiamo raccogliere i dati dell'occupazione di un vertice scelto casualmente nel grafo in funzione del tempo e vedere più da vicino cosa accade durante la simulazione in questo regime. A fini rappresentativi, sono stati selezionati 6 nodi/strade diversi all'interno del network urbano (evidenziate in rosso come archi in Fig. 3.1), cercando di includere sia nodi sufficientemente centrali

[1]Stabilita casualmente ad ogni run

sia nodi marginali e presumibilmente meno trafficati. Tali grafici delle lunghezze delle code in funzione del tempo, per una specifica run della simulazione con $N = 1900$, sono riassunti in Fig. 3.2. Notiamo subito un aspetto fondamentale: i valori di $n_i(t)$ non tendono ad appiattirsi ad una quantità costante ma sono anzi fluttuanti nel tempo in maniera abbastanza caotica. L'ampiezza di tali fluttuazioni dipende indubbiamente dalla centralità del nodo all'interno della rete: nodi periferici (come quello della strada $0 - 1$) sono poco trafficati, mentre nei nodi più centrali (e.g. la strada $52 - 43$) si osservano fluttuazioni maggiori. A volte può succedere che la lunghezza della coda in una strada superi temporaneamente la capienza massima fissata $N_{max} = 15$, ma finché il sistema rimane nel regime di free-flow, come già accennato, tale eccesso viene riassorbito (come è facile vedere dai grafici). La congestione, dunque, può insorgere localmente ma non riesce a diffondere lungo tutto il network.

Distribuzione delle code e trattazione analitica In Fig. 3.3 sono riportate le stesse informazioni ma sotto forma di istogramma delle occorrenze. In Fig. 3.4, gli stessi istogrammi per una run di simulazione con carico diverso ed inferiore, $N = 500$. Ci si accorge che, per carichi alti o per nodi particolarmente trafficati, la distribuzione delle lunghezze delle code nei nodi appare bimodale, con due picchi ben distinti, uno verso occupazioni basse e uno quasi a ridosso delle capienza massima N_{max}. Questo fenomeno è facilmente spiegabile facendo riferimento ad una trattazione analitica del modello un poco semplificata che sfrutta la teoria dei processi markoviani. Possiamo, infatti, immaginarci che l'evoluzione del modello OD sia paragonabile ad un ensemble di random walkers che si muovono sul grafo seguendo certe probabilità di transizione π_{ij}, che possono ad esempio essere misurate da simulazione valutando quanti agenti svoltano verso un certo nodo i partendo da un nodo j, senza meccanismi di calcolo intelligente del tragitto. Per comprendere meglio l'insorgere della bimodalità, costruiamo momentaneamente un modello alternativo in cui

- Esistono N random walkers sullo stesso grafo che effettuano svolte probabilistiche in base alla matrice di transizione π_{ij} fissata. Tale matrice può, ad esempio, essere misurata dal modello OD (si veda, a tal proposito, [16])
- La dinamica è asincrona o *one-step*, quindi l'evoluzione avviene agente dopo agente. Nel modello OD di cui si è parlato in precedenza la dinamica era sincrona, ma per ora non ce ne preoccupiamo
- Gli agenti/random walkers sono sempre soggetti agli stessi vincoli di flusso massimo e di capienza massima dei nodi

Un tale processo di Markov non lineare può essere, almeno in parte, affrontato analiticamente. Lo stato dinamico del sistema è specificato quando è noto $\vec{n}(t)$, cioè

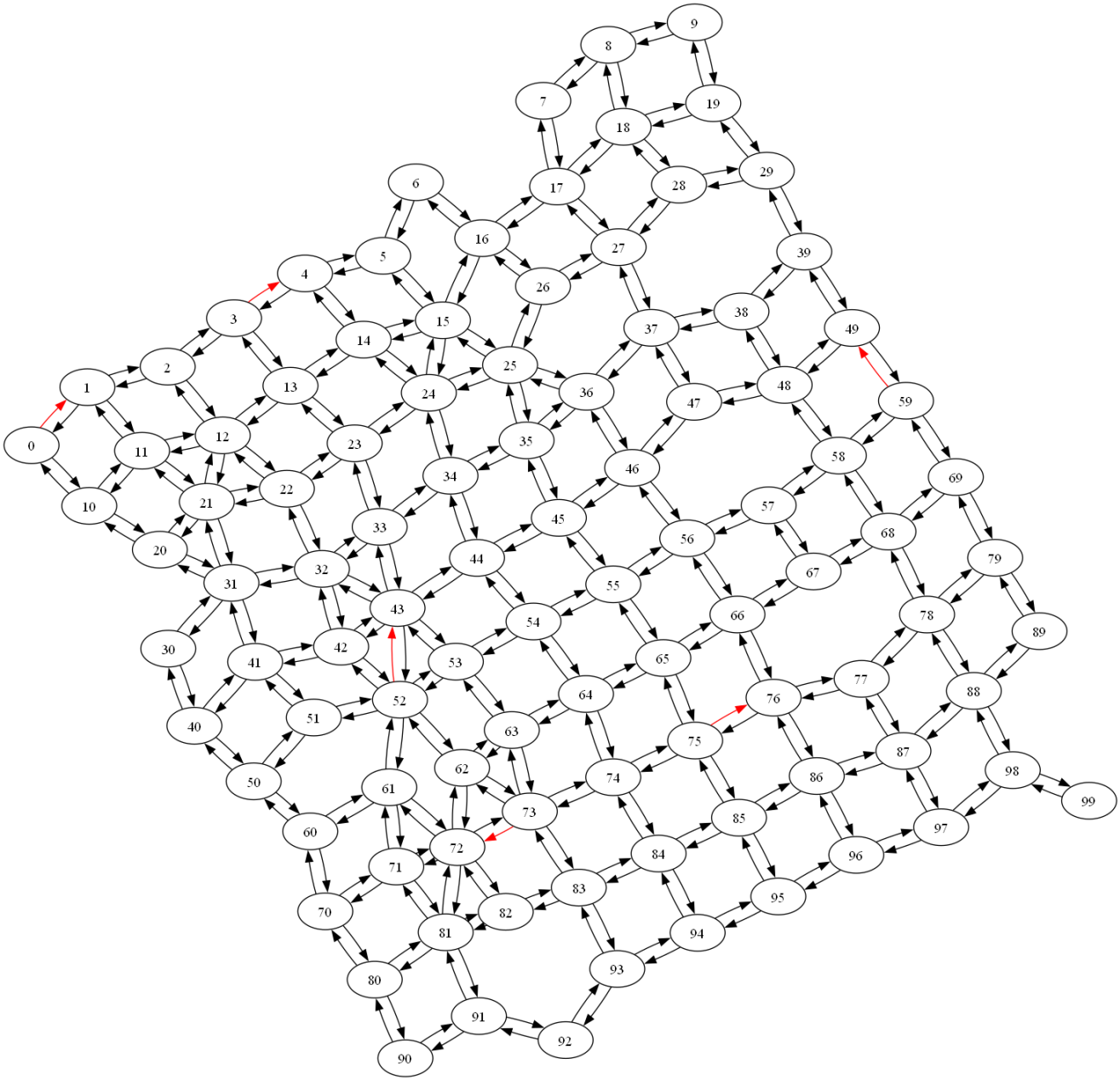

Figura 3.1: *Grafo in rappresentazione fisica (strada = arco) dove sono state evidenziate in rosso le strade impiegate per l'analisi delle code. Alcune sono particolarmente centrali, altre invece sono più marginali ed esterne.*

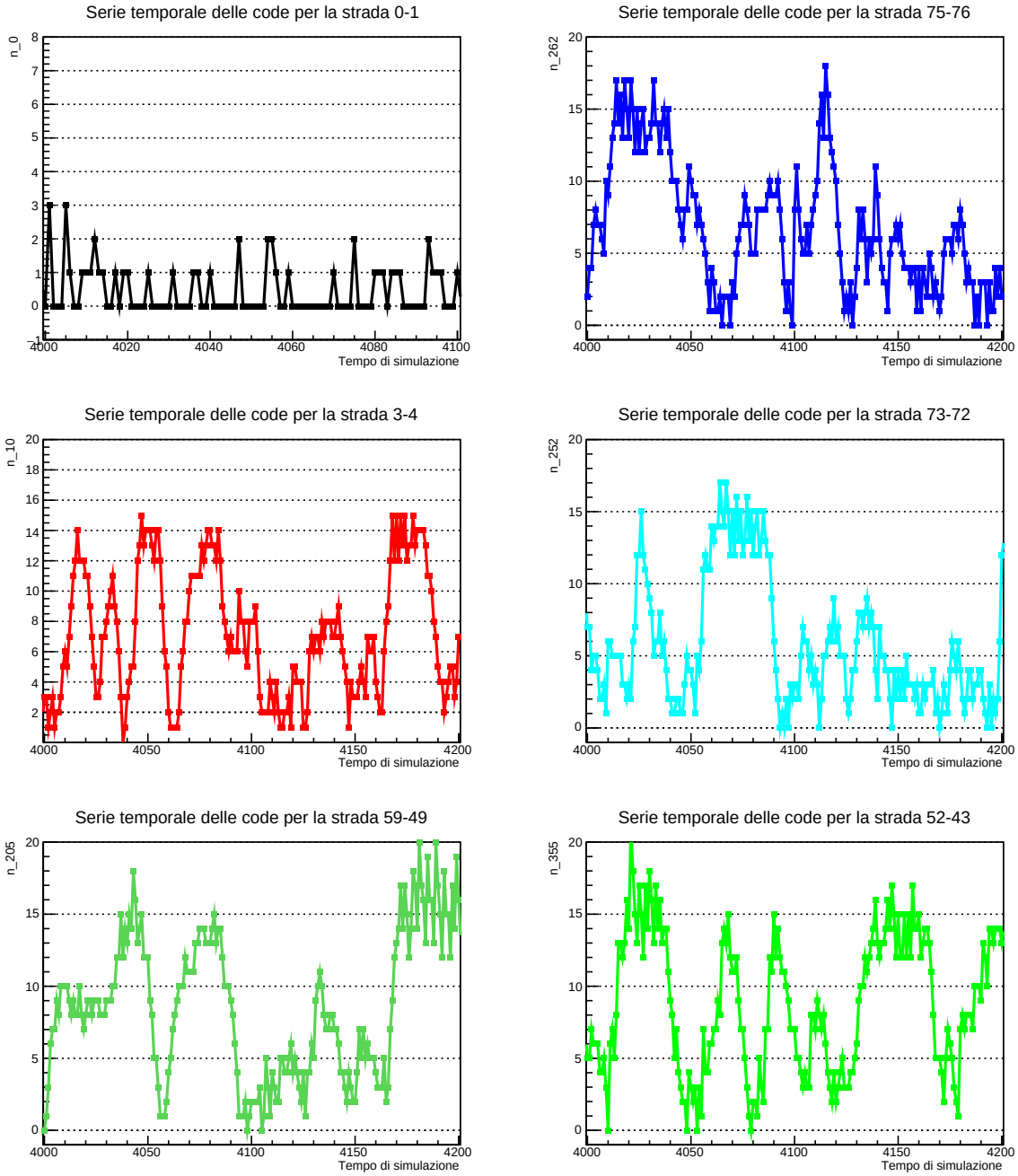

Figura 3.2: *Dati raccolti da una singola run con carico di rete* $N = 1900$. *I sei grafici rappresentano l'evoluzione nel tempo dell'occupazione di un singolo vertice nel grafo duale,* $n_i(t)$. *I sei vertici sono stati scelti per rappresentatività nel duale. È facile notare l'andamento fluttuante aperiodico dell'occupazione per tutti e sei i casi*

l'occupazione di ogni nodo ad un dato istante temporale, col vincolo che:

$$\sum_i n_i = N \tag{3.1}$$

Detto $\rho(\vec{n}, t)$ la probabilità che, al tempo t, lo stato della rete sia proprio $\vec{n} = (n_1, n_2, \dots)$, allora possiamo scrivere [17] la *one-step master equation* relativa al processo di Markov:

$$\rho(\vec{n}, t + \Delta t) - \rho(\vec{n}, t) = \Delta t \sum_{i,j} [E_i^- E_j^+ \varphi_{ij}(n_i, n_j) - \varphi_{ji}(n_j, n_i)]\rho(\vec{n}, t) \tag{3.2}$$

dove $\varphi_{ij}(n_i, n_j)$ definisce il flusso in ingresso e (in uscita) dai nodi e E^+, E^- sono chiamati *operatori di scaletta o di creazione/distruzione* [18] che agiscono su di una generica funzione dello stato $f(\vec{n})$:

$$E_i^+ f_i = f_{i+1} \qquad\qquad E_i^- f_i = f_{i-1} \tag{3.3}$$

Per tenere conto degli effetti di trasporto massimo e di capienza massima, possiamo riscrivere:

$$\varphi_{ij}(n_i, n_j) = \pi_{ij} \cdot \Phi_{max} \cdot \phi(n_j) \cdot c(n_i/N_{max}) \tag{3.4}$$

dove π_{ij} è la probabilità intrinseca della svolta $j - i$, mentre gli altri due termini rappresentano i vincoli non lineari. In particolare, $\Phi_{max} \cdot \phi(n_j)$ è una funzione monotona con valore massimo Φ_{max} (quindi $\phi(n) \in [0, 1]$) che rappresenta la capacità del nodo di far fluire agenti in relazione alla sua occupazione. Al contrario, la funzione $c(n_i/N_{max})$ codifica la capacità del nodo target di ospitare agenti in relazione al suo carico. Possiamo supporre, per fissare le idee:

$$\phi(n_j) = \theta(n_j) \tag{3.5}$$

$$c(n_i/N_{max}) = \theta(N_{max} - n_i) \tag{3.6}$$

con $\theta(x)$ funzione a gradino di Heaviside. Per comodità, lavoriamo nell'ipotesi $\Phi_{max} = 1$. In altre parole, il flusso in uscita da un nodo è massimo a pari a 1 quando nel nodo c'è almeno un agente. Il suo flusso in ingresso, cioè la sua capacità di far entrare nuovi agenti, è inalterato finché non si raggiunge la soglia N_{max}: superata tale soglia, $c(n_i/N_{max}) = 0$ e $\varphi_{ij} = 0$

Fatte queste premesse, possiamo riscrivere la (3.2) come:

$$\frac{\partial}{\partial t}\rho(\vec{n}, t) = \frac{1}{N} \sum_{i,j} [E_i^- E_j^+ \pi_{ij} \theta(n_j)\theta(N_{max} - n_i) - \pi_{ji}\theta(n_i)\theta(N_{max} - n_j)]\rho(\vec{n}, t) \tag{3.7}$$

Individuare la soluzione stazionaria a partire da questa equazione non è affatto banale. Si può dimostrare [19] che l'effetto delle due funzioni non lineari di Heaviside che regolano la dinamica, se prese singolarmente, è quello di creare una distribuzione delle code di tipo

circa esponenziale piccata o sull'origine ($n = 0$) o sul valore di capacità massima ($n = N_{max}$). L'effetto congiunto dei due effetti, quindi, è quello di generare una distribuzione con due picchi, dunque bimodale, esattamente come osservato sperimentalmente in Fig. 3.3. Ne possiamo concludere, quindi, che lo stato vuoto $n = 0$ e lo stato congestionato $n = N_{max}$ sono stati attrattivi per la dinamica del traffico. Tale affermazione, confermata dai dati, è effettivamente ragionevole: nel caso di stato vuoto, un nodo senza agenti ha minor flusso uscente e quindi tende a rendere vuoti i nodi a valle. Al contrario, un nodo congestionato non riceve flusso poiché saturo e quindi aumenta la probabilità di avere nodi vicini a monte congestionati.

Comportamento asintotico Cerchiamo ora di studiare l'andamento asintotico di tale distribuzioni per n_i "grandi" (intendendo con ciò la parte finale della distribuzione). Dalle considerazioni fatte in precedenza, ci aspettiamo che la distribuzione osservata sia di tipo approssimativamente esponenziale. Tuttavia, è fondamentale ricordare che i risultati ottenuti nel paragrafo precedente sono validi per una dinamica asincrona e di tipo markoviano, caratteristiche che non sono esattamente proprie del modello OD in esame, dunque non possiamo aspettarci una perfetta aderenza con una distribuzione esponenziale.

Possiamo ipotizzare che l'andamento asintotico degli istogrammi segua una distribuzione Gamma indicizzata dai parametri k, θ con probability density function:

$$f(x; k, \theta) = \frac{x^{k-1} e^{-x/\theta}}{\Gamma(k)\theta^k} \tag{3.8}$$

e quindi una miscela di legge di potenza e di esponenziale. In effetti, questa ipotesi si rivela quella più adatta: in Fig. 3.5 sono riportati i fit operati da ROOT tramite distribuzione Gamma su alcuni degli istogrammi proposti in precedenza. In tutti i casi, i valori del χ^2 è molto buono. Inoltre, la distribuzione Gamma tende ad una distribuzione esponenziale per valori elevati dell'argomento, in accordo con i risultati teorici già discussi.

3.2 Transizione allo stato congestionato

Aumentando il carico di rete, per la prima volta intorno circa a $N = 2000$, accade qualcosa di nuovo: il sistema subisce una vera e propria transizione di fase. Per i primi istanti temporali dopo il lancio, la simulazione procede come nel regime precedente e gli agenti si distribuiscono nel network a partire dalla configurazione iniziale. C'è però un effetto particolare: se in precedenza ogni agente riusciva a giungere a destinazione muovendosi più o meno agevolmente all'interno del network, superata una certa soglia in N può succedere che, ad un certo punto della simulazione, si formino in tempi molto rapidi dei cluster di nodi *completamente congestionati*. Per capire bene la natura di questi

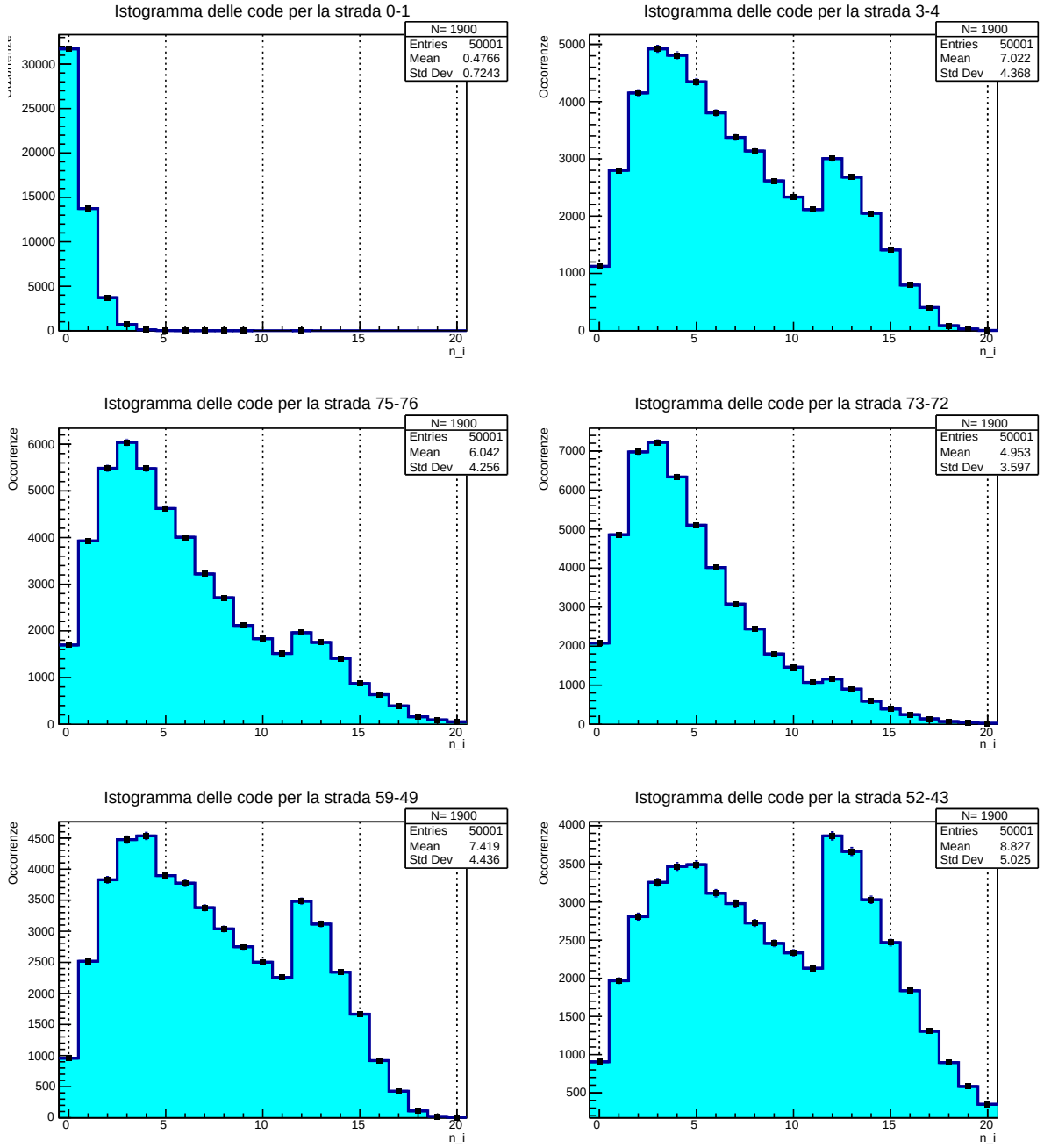

Figura 3.3: *Istogramma delle occupazioni di sei nodi scelti nel network, come già mostrato in Fig.3.2, raccolti su un periodo di osservazione del modello* $T = 50000$*, carico* $N = 1900$*. Gli errori, valutati come radice quadrata del contenuto del bin, sono molto piccoli e quindi difficilmente visibili, considerata la grande mole di eventi raccolti. La distribuzione appare in certi casi bimodale*

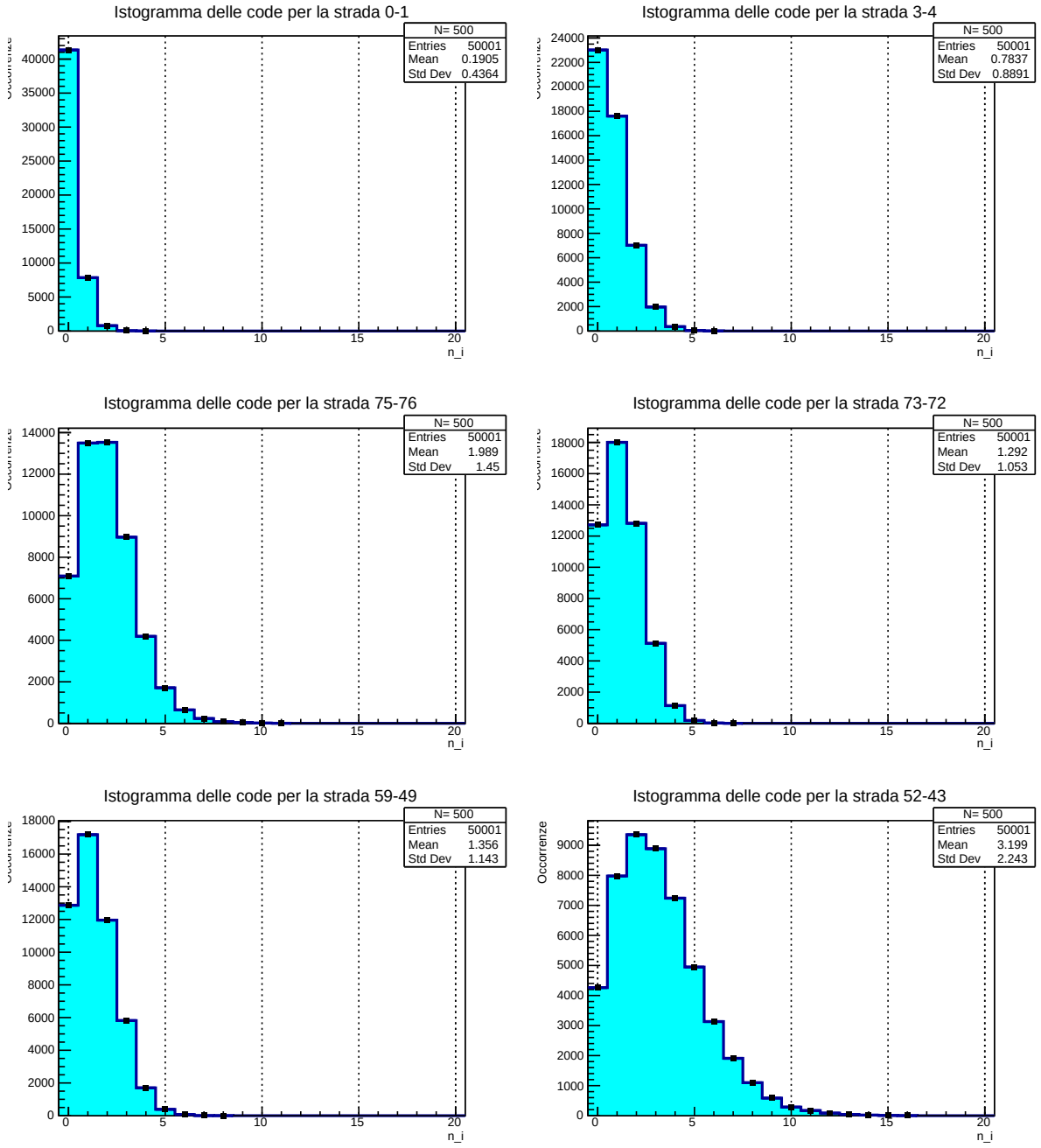

Figura 3.4: *Come in Fig.3.3, ma stavolta con un carico di rete inferiore* $N = 500$.

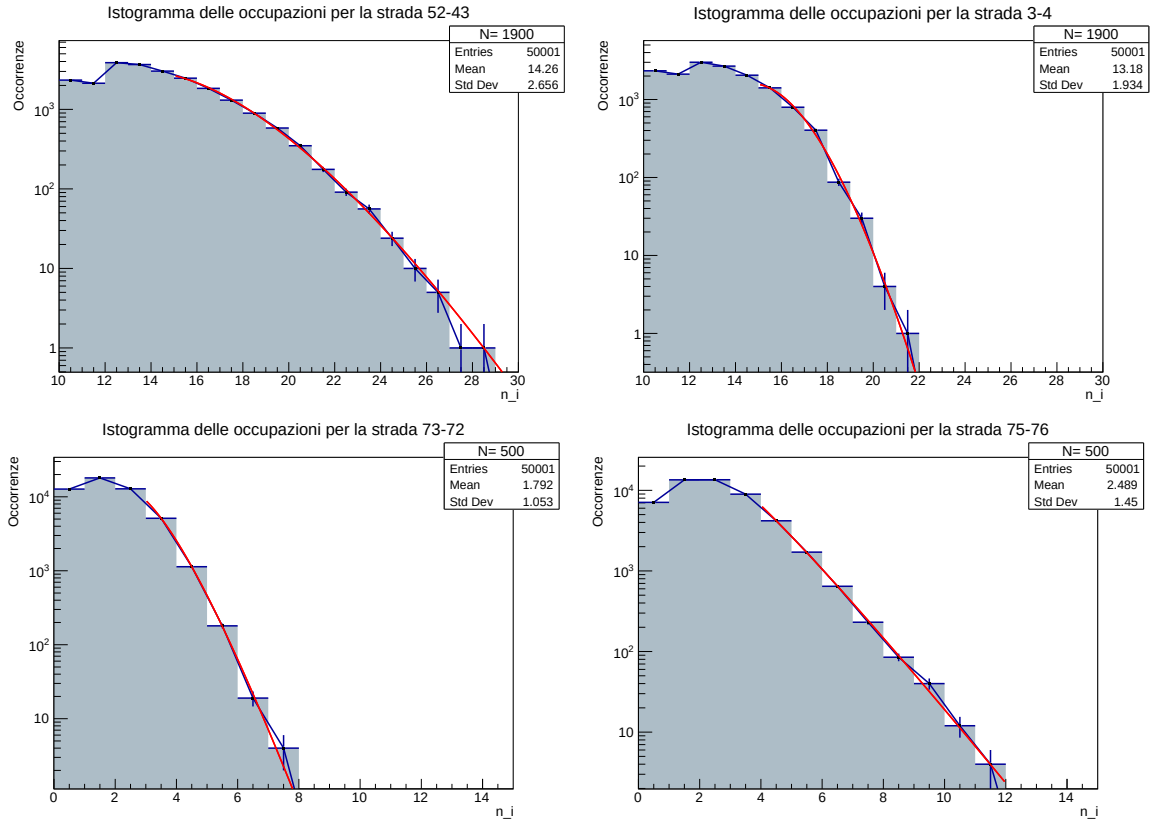

Figura 3.5: *Alcuni istogrammi scelti fra quelli di Fig.3.3 e Fig. 3.4, mostrati per comodità in scala semi-logaritmica. In rosso, la linea di best fit secondo una distribuzione di probabilità Gamma operata dal software ROOT. In tutti e quattro i casi, il fit è visivamente buono e l'accordo coi dati, misurato tramite* χ^2*, è ottimo.*

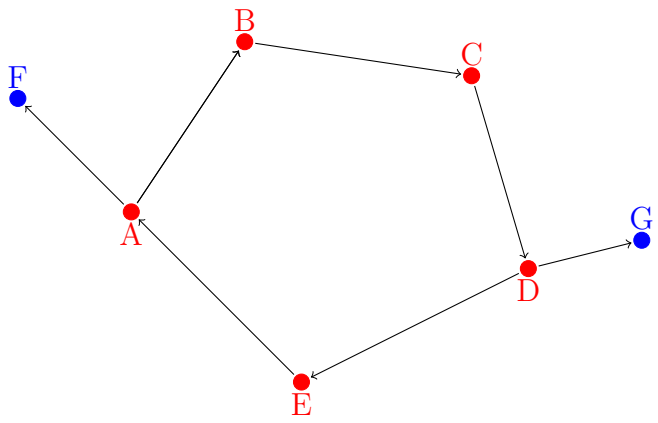

Figura 3.6: *Esempio di cluster di gridlock parziale*

cluster congestionati, facciamo riferimento a Fig. 3.6. I nodi rossi (A,B,C,D,E,F) sono completamente pieni, cioè ospitano un quantitativo di agenti superiori a $N_{max} = 15$: per quanto detto nei capitoli precedenti, dunque, ogni agente che voglia transitare per questi nodi dovrà prima aspettare che la coda si riduca. I nodi in blu, F e G, sono invece non saturi, dunque possono far entrare ancora veicoli. Immaginiamo che in ogni nodo rosso vi siano agenti il cui prossimo vertice da attraversare per giungere alla loro destinazione è un vertice rosso, dunque temporaneamente impercorribile. Si viene dunque a creare un loop (o meglio un *grafo ciclico*) in cui, ad esempio, ogni agente in A vuole andare in B e ogni agente in B vuole andare in C e così via. A rigore, gli agenti in D potrebbero passare a G, che è libero, ma se nell'atto della loro creazione hanno deciso che il nodo da occupare dopo D è precisamente E, allora non hanno modo di liberarsi dal loop: si viene a creare un *gridlock* della rete, cioè una struttura ciclica interna al grafo stesso definita dinamicamente dalle intenzioni degli agenti, *uno stato stazionario congestionato del sistema*. Tale struttura non è semplicemente definita dai nodi che hanno raggiunto una popolazione pari o superiore a N_{max} (condizione necessaria ma non sufficiente). I nodi devono essere *completamente congestionati*, cioè è necessario che tutti gli agenti che vivono in un siffatto nodo si ritrovino bloccati poiché vogliono viaggiare verso un terzo nodo anch'esso completamente congestionato (definizione dunque ricorsiva). Un esempio tratto da una simulazione vera è riportato in Fig. 3.7. La loro presenza è indicativa del fatto che il sistema ha raggiunto la soglia critica di carico e che una congestione nel network si è formata: il sistema ha cambiato fase. Nella vita reale, chiaramente, tali blocchi stradali si assorbono nel tempo e si risolvono; nella simulazione questo non accade perché l'algoritmo con cui gli agenti decidono il percorso da intraprendere si basa su di un'informazione circa lo stato globale del sistema *nell'istante della loro partenza*. Se le condizioni di traffico cambiano, gli agenti non sanno aggiornare il percorso e rischiano di finire dentro un gridlock che si è formato dopo la loro creazione. A prescindere tuttavia

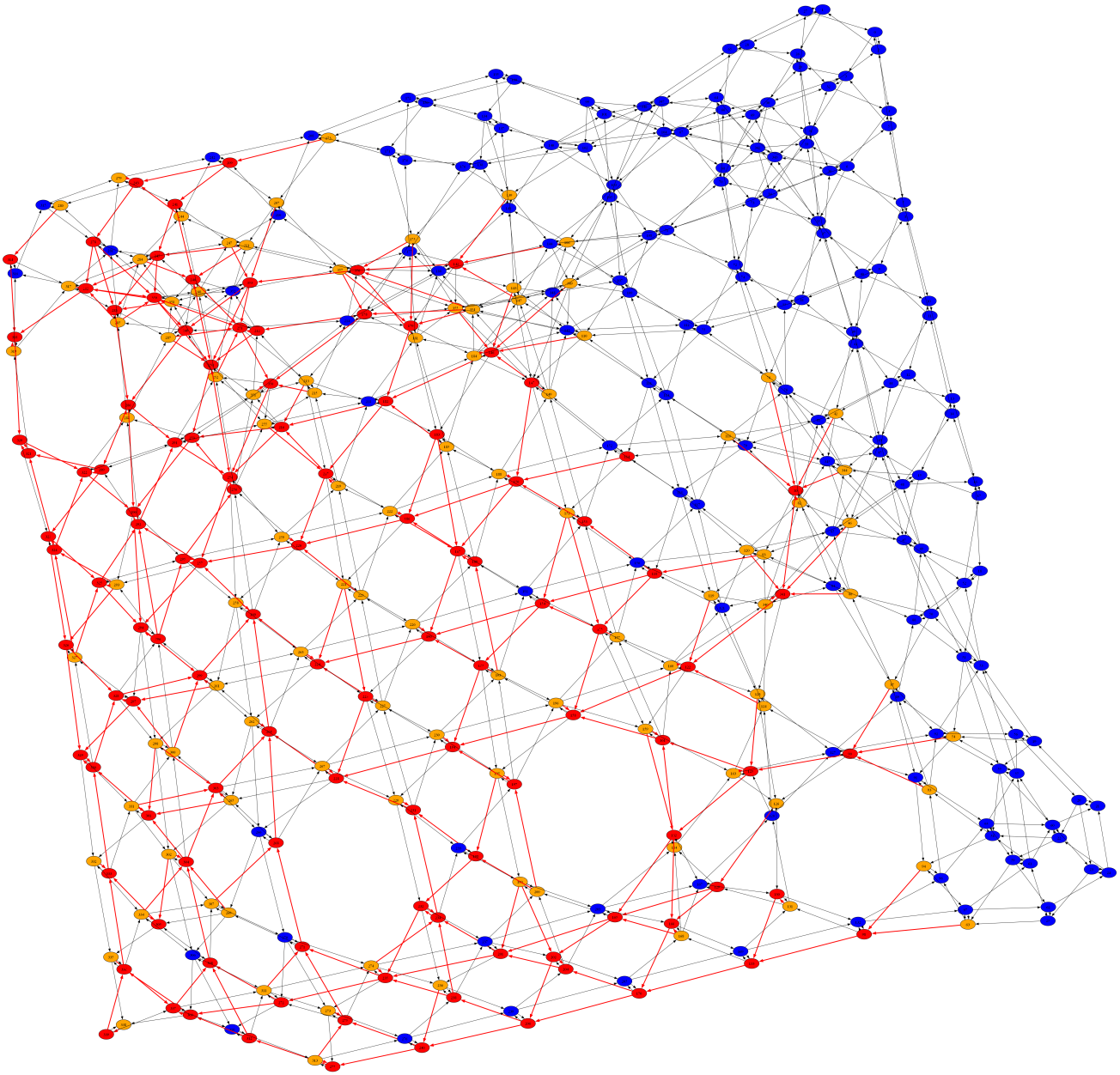

Figura 3.7: *Grafo (duale) ottenuto da una run terminata nel gridlock totale di rete. I nodi rossi sono i nodi saturi ($n_i \geq N_{max}$) mentre i nodi arancioni sono popolati da agenti ma non saturi. Gli archi evidenziati in rosso rappresentano le intenzioni di svolta degli agenti all'interno dei nodi da cui dipartono. Come si nota ad occhio, ogni arco rosso punta verso un nodo saturo, dunque non può esserci mobilità: il sistema è collassato nel gridlock totale di rete. I nodi arancioni e i nodi rossi sono dunque nodi completamente congestionati, poiché gli agenti che vivono in suddetti nodi vogliono passare verso un nodo (arco rosso) saturo.*

dall'evoluzione del gridlock in sè, per lo studio della quale il modello non è adatto, quello che è interessante studiare è *sotto quali condizioni* tali cluster completamente congestionati si formano nel network, poiché tali condizioni ci permettono di predire in anticipo se, sotto determinate condizioni iniziali, si potrà formare una congestione nel sistema

3.2.1 Probabilità di congestione p_g

Finché il carico di rete N rimane sufficientemente basso, il gridlock non si presenta mai ed il sistema si porta sempre nell'equilibrio dinamico di cui abbiamo già parlato (stato di free-flow). Per N abbastanza alti, invece, il sistema si porta sempre in un gridlock totale di rete. Per comprendere meglio tale dinamica, procediamo nella maniera seguente.

Fissiamo un carico di rete N e facciamo girare per R volte la simulazione da condizioni iniziali diverse scelte randomicamente[2] per un tempo di osservazione sufficientemente alto, $T = 20000$ step temporali. Se il sistema si porta nel gridlock, il programma se ne accorge: ripetendo il processo molte volte, è possibile calcolare $p_g(N)$, ossia la probabilità che, dato un carico N, il sistema evolva verso la congestione

In Fig. 3.8 sono riportate le osservazioni circa la probabilità che il sistema collassi nel gridlock entro il tempo $T = 20000$ per vari carichi N. Gli errori sono stati calcolati con la regola del conteggio, assumendo una probabilità poissoniana (quindi se su R ripetizioni si osservano αR scenari in cui il modello si porta nel gridlock, l'errore associato è $\sqrt{\alpha R}$).

È facile individuare nel grafico tre regioni distinte che segnalano abbastanza bene le due fasi che il sistema assume al variare del parametro di controllo N. Finché il carico è sufficientemente basso, indicativamente $N < 2000$, il sistema raggiunge sempre un equilibrio dinamico in cui il traffico scorre e gli agenti raggiungono sempre la destinazione, eventualmente un poco rallentati. Per carichi molti alti, $N > 3000$ circa, il sistema evolve in fretta verso la congestione totale di rete in tutte le diverse run simulate su macchina: il gridlock è garantito, sempre. Per valori di N intermedi, esiste invece una *zona di transizione* in cui l'apparire del cluster di gridlock è questione di probabilità e dipende presumibilmente dalle condizioni iniziali di rete (e.g. distribuzione iniziale degli agenti). Tale andamento è sintomo di una vera e propria *transizione di fase* della rete urbana, da un regime di scorrimento agevole verso una fase completamente congestionata ed intasata.

In genere, in una transizione di fase ci si aspetta un cambiamento brusco nelle proprietà del sistema che si sta studiando: il gridlock dovrebbe comparire d'improvviso, superato un certo carico critico N_c. Questo non accade nella simulazione proposta poiché il sistema non è nel limite termodinamico, e cioè il grafo su cui si esplica la dinamica ha proprietà finite (le fluttuazioni di traffico sulle strade non vanno a zero, ad esempio). La

[2]Per condizioni iniziali diverse si intende l'iniziale distribuzione degli agenti nella rete. La struttura statica del grafo, cioè la topologia delle connessioni fra nodi e il peso w_i^0, invece, è costante

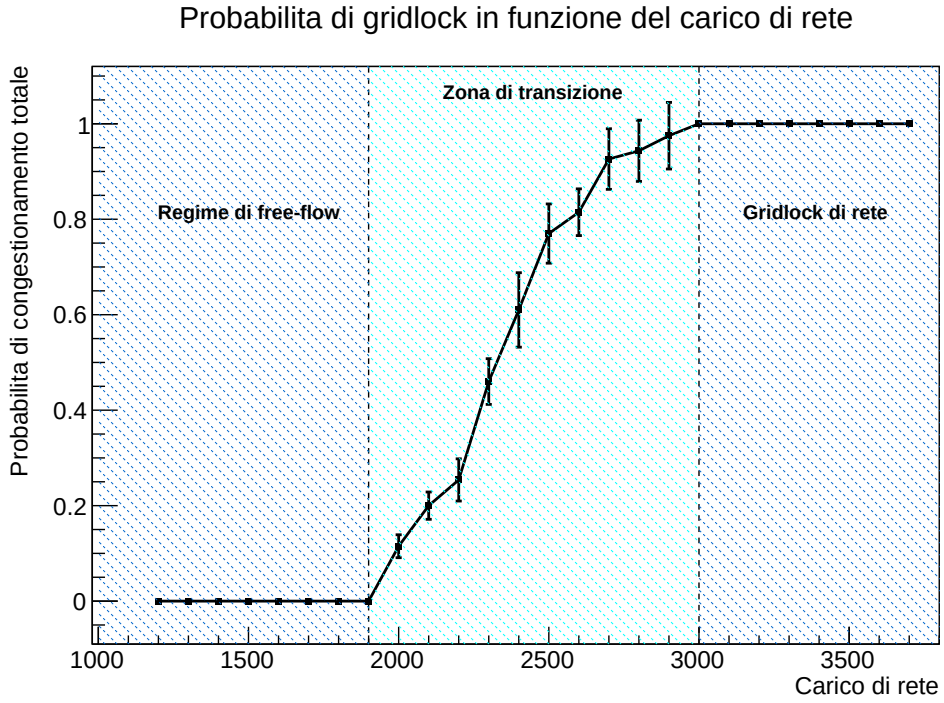

Figura 3.8: *Probabilità che il sistema si porti nel gridlock di rete entro un tempo di osservazione ragionevolmente lungo* $T = 20000$ *in funzione del numero di agenti* N*. È possibile apprezzare tre regioni distinte: una regione di free-flow a sinistra in cui il sistema raggiunge sempre l'equilibrio dinamico, una regione a destra in cui il numero elevato di agenti nella rete scatena sempre il congestionamento e una regione di transizione fra i due regimi.*

transizione è quindi leggermente più smooth e il cambiamento brusco in p_g è smussato ed assume la forma sigmoidale tipica di una transizione di fase al finito.

Possiamo caratterizzare la transizione di fase verso lo stato congestionato anche studiando un'altra proprietà caratteristica del sistema, e cioè la dimensione tipica del cluster completamente congestionato

3.2.2 Grandezza media dei cluster di gridlock

Definiamo la *grandezza caratteristica* S di un gridlock come la frazione di vertici completamente congestionati che appartengono a tale cluster (come già definito in precedenza):

$$S = \frac{\#\{v \in V |\ v \text{ è completamente congestionato }\}}{\#V} \tag{3.9}$$

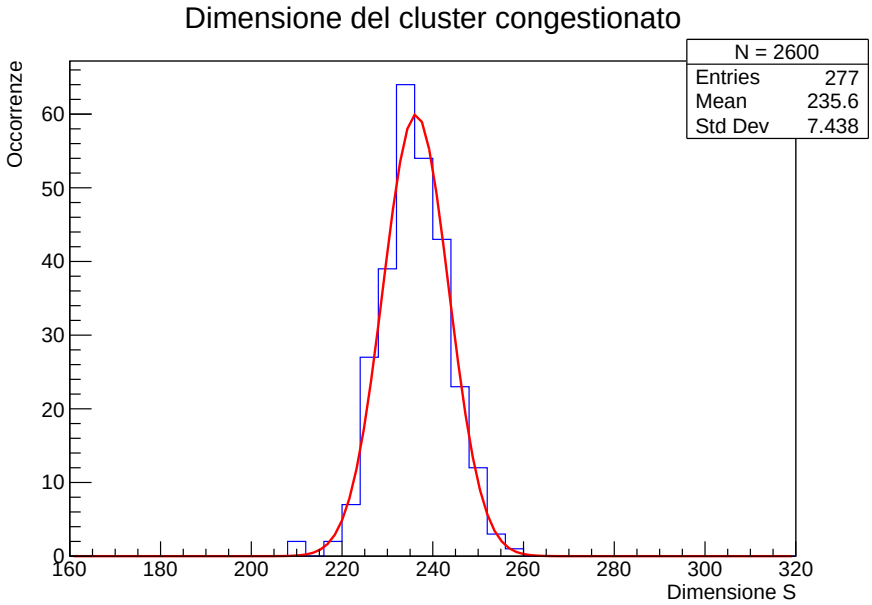

Figura 3.9: *Distribuzione della dimensione del cluster congestionato per più run operate con* $N = 2600, T = 20000$. *In rosso, fit con distribuzione normale operato da ROOT*

dove $\#V$ è il numero di vertici, e cioè il numero totale di strade (nel caso considerato, $V = 360$).

L'obiettivo è dunque quello di individuare $S(N)$, cioè la dimensione media del cluster di gridlock quando il carico di rete è precisamente N. Poiché il sistema non è nel limite termodinamico ($V \to \infty$), non possiamo aspettarci di ottenere un valore numerico preciso, anzi ad ogni run della simulazione otterremo un valore un po' differente. Procediamo come già fatto per individuare p_g, e cioè ripetendo R volte la simulazione con N fissato da condizioni iniziali diverse, selezionando solo le run in cui il sistema collassa nel gridlock. Un esempio per un valore del carico ben preciso $N = 3000$ è riportato in Fig. 3.9: la distribuzione di S misurata per diverse run della simulazione è ben approssimata da una gaussiana, per cui valutiamo l'errore come errore della media gaussiana:

$$S(2600) = 235.6 \pm 0.5 \tag{3.10}$$

dove, nuovamente, l'errore sulla misura è ottenuto dividendo la stima della deviazione standard per la radice quadrata degli eventi. Ripetendo questa procedura per diversi valori del carico di rete N, costruiamo quindi il grafico riportato in Fig. 3.10

Anche da questa figura, è facile confermare la presenza di un punto critico del sistema. Nel regime sotto-critico, il sistema rimane sempre in condizioni di free-flow e, chiaramente, $S = 0$. Quando N comincia a crescere e supera $N \approx 2000$, il sistema ha una certa probabilità di collassare nel gridlock (si veda Fig. 3.8). Quando tale blocco si

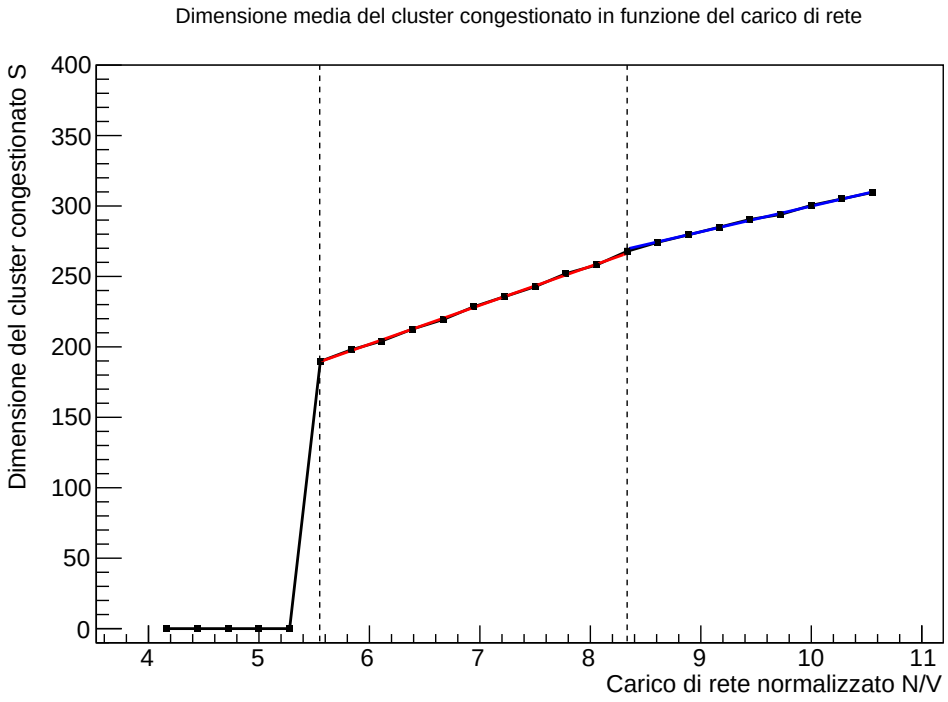

Figura 3.10: *Dimensione media S del cluster congestionato in funzione del carico di rete normalizzato N/V, con V numero totale di vertici nel grafo. Gli errori sono raccolti come errore della media di un processo gaussiano e sono abbastanza piccoli perché si vedano bene nell'immagine. I dati raccolti da simulazione sono segnati in nero, la linea rossa è il fit lineare proposto da ROOT*

forma, l'estensione di quest'ultimo, riportato in Fig. 3.10, sembra crescere linearmente con il carico di rete N: un fit con ROOT conferma l'andamento rettilineo con un $\chi^2_{rid} \approx 1$ e parametri:

$$S(n) = (27.7 \pm 0.3)n + (36 \pm 2) \tag{3.11}$$

Quando si entra nel regime di gridlock sicuro, circa $N > 3000$, l'estensione S continua comunque a crescere linearmente ma cambiando, per qualche ragione, pendenza della retta:

$$S(n) = (18.2 \pm 0.4)n + (117 \pm 4) \tag{3.12}$$

3.3 Caratterizzazione regime di free-flow

Studiamo il sistema in regime di free-flow, dunque con carichi di rete N bassi (intendendo con ciò $N < 2000$)

3.3.1 Flusso di veicoli in relazione al carico

La simulazione costruita procede per intervalli temporali discreti $\Delta t = 1$. Ad ogni step temporale, un certo numero $x(t_i)$ di agenti riescono a transitare da una strada all'altra: definiamo quindi il *flusso istantaneo veicolare* ϕ come numero di agenti che si muovono nell'unità di tempo:

$$\phi(t_i) = \frac{x(t_i)}{\Delta t} = x(t_i) \tag{3.13}$$

Operativamente parlando, è sufficiente contare quanti agenti si spostano ad ogni ciclo di simulazione. I dati raccolti per una singola run rappresentativa con parametri $T = 10000$, $N = 1000$ è mostrata in forma di grafico dipendente dal tempo e di istogramma delle occorrenze in Fig. 3.11. Per estrarre un valore significativo di ϕ in funzione del carico di rete, basterà prendere la media della distribuzione misurata, poiché l'accordo con una gaussiana è visivamente ottimo. A titolo d'esempio, dunque:

$$\phi(1000) = 659.49 \pm 0.16 \tag{3.14}$$

dove l'errore è stato valutato come errore sulla media, cioè

$$\sigma_{\bar{\phi}} = \frac{\sigma_\phi}{\sqrt{n}} \quad \text{con } \sigma_\phi = 15.62 \tag{3.15}$$

con n numero di eventi ($n \approx 20000$) e σ_ϕ deviazione standard della distribuzione. Procediamo quindi a raccogliere i valori medi delle distribuzioni di flusso per vari carichi di rete N: otterremo un grafico riportato in Fig. 3.12. Inizialmente la curva $\phi(N)$ assomiglia ad una retta di pendenza unitaria. Questa proprietà è ragionevole poiché, a carichi bassi, gli agenti si comportano come fossero un gas rarefatto, interagendo dunque poco con gli

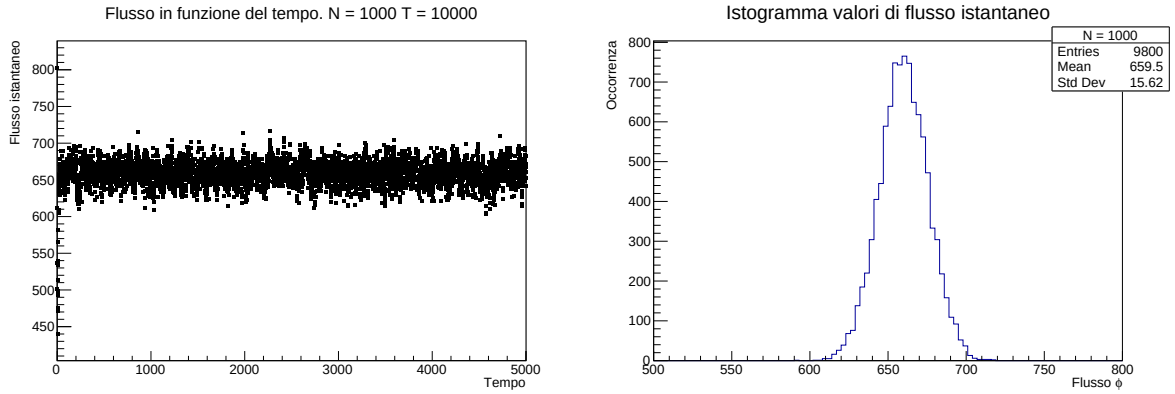

Figura 3.11: *Dati di flusso istantaneo raccolti durante una run del modello con* $T = 10000$ *e* $N = 1000$*. A sinistra, i dati sono visualizzati nella loro dipendenza temporale. Dopo un primo transiente abbastanza breve, i valori di flusso misurati sembrano assestarsi attorno ad un valore medio e fluttuarci attorno. A sinistra, gli stessi dati ma organizzati in un istogramma. È ben evidente il comportamento gaussiano della distribuzione piccato attorno ad un valore ben definito*

altri veicoli nella rete (la velocità di movimento rimane circa costante). Di conseguenza, ad ogni ciclo di simulazione tutti gli agenti riusciranno a spostarsi da una strada all'altra e il flusso istantaneo sarà numericamente uguale al carico di rete. Questa tendenza è vera fino a $N = 300$ circa, poiché nel modello si è scelto $\Phi = 3$, e quindi finché in ogni nodo convivono al massimo 3 agenti le interazioni reciproche non dovrebbero farsi sentire. Da quel punto in poi, la curva del flusso aumenta col traffico poiché, banalmente, aumentano gli agenti nella rete; tuttavia la velocità di crescita diminuisce a causa delle interazioni reciproche. Nella regione $N \approx 2000$, la funzione tende ad appiattirsi e assestarsi attorno ad un plateau costante: la rete ha raggiunto la sua capacità massima.

Il grafico di Fig. 3.12 ricorda da vicino quello che in letteratura scientifica è noto col nome di *Fundamental Diagram of Traffic Flow* o *Macroscopical Fundamental Diagram* ([7], [10]), riproposto in Fig. 3.13

3.3.2 Tempo medio di percorrenza

Fissiamo nuovamente un carico di rete N in regime di free-flow. Quando un agente giunge a destinazione, misuriamo il tempo che gli è servito per arrivarci partendo dalla sua origine, contando quanti cicli di simulazione t sono trascorsi dalla sua nascita (*lifespan*). Un esempio delle distribuzioni di lifespan è riportato, per valori crescenti del carico di rete, in Fig. 3.14, in scala normale e in scala semi-logaritmica. Notiamo subito che la distribuzione non è perfettamente simmetrica e tende a diventare *fat-tailed* aumentando il carico di rete: tutto ciò è ragionevole, poiché è più probabile che gli agenti rimangano

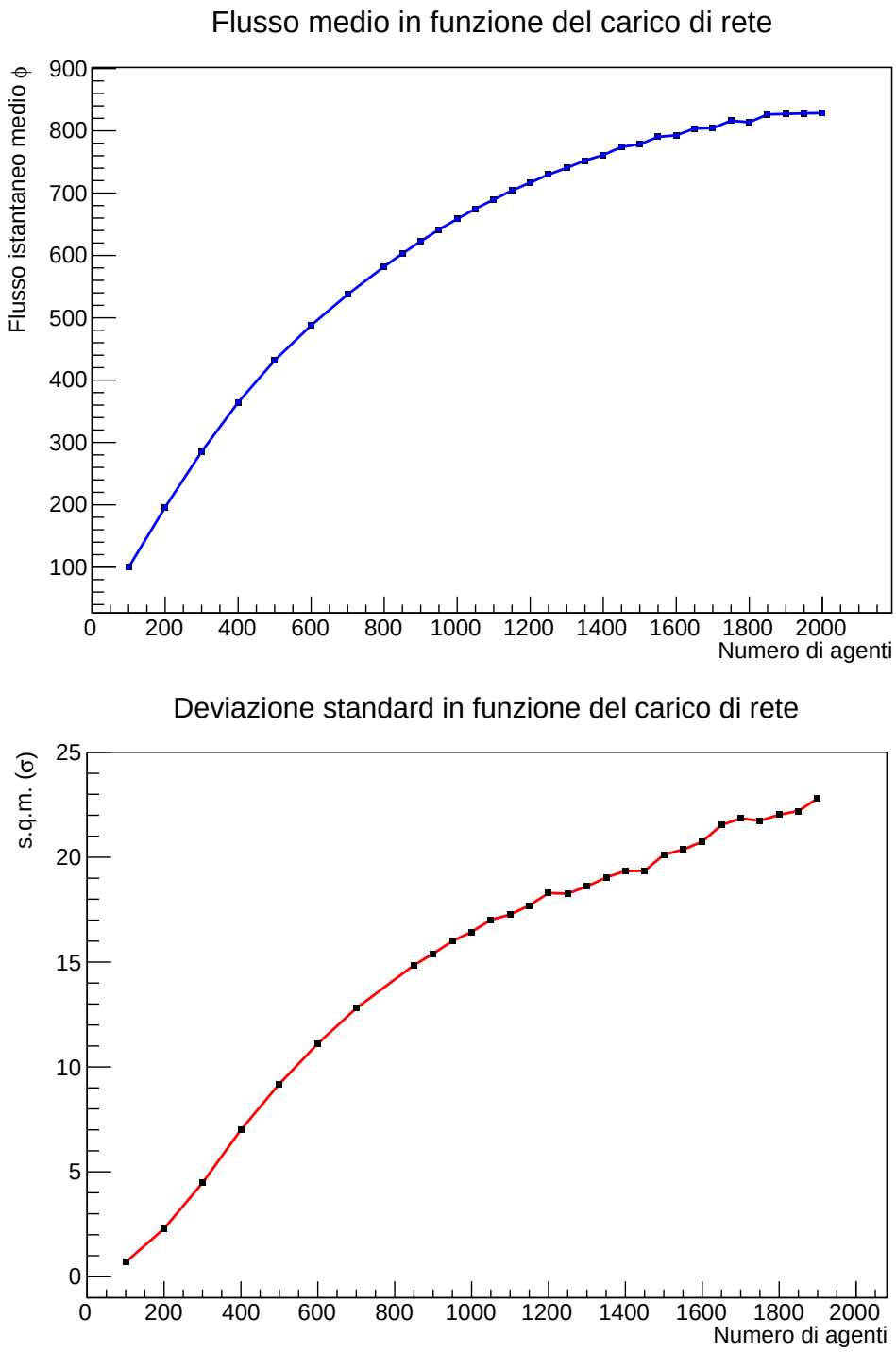

Figura 3.12: *In alto, flusso medio in funzione del carico di rete N. Come ci si aspetta, il flusso aumenta fino ad un certo carico critico, oltre il quale tende ad assestarsi. In basso, deviazione standard della distribuzione del flusso. Per valori più elevati di carico, anche la s.d. sembra aumentare*

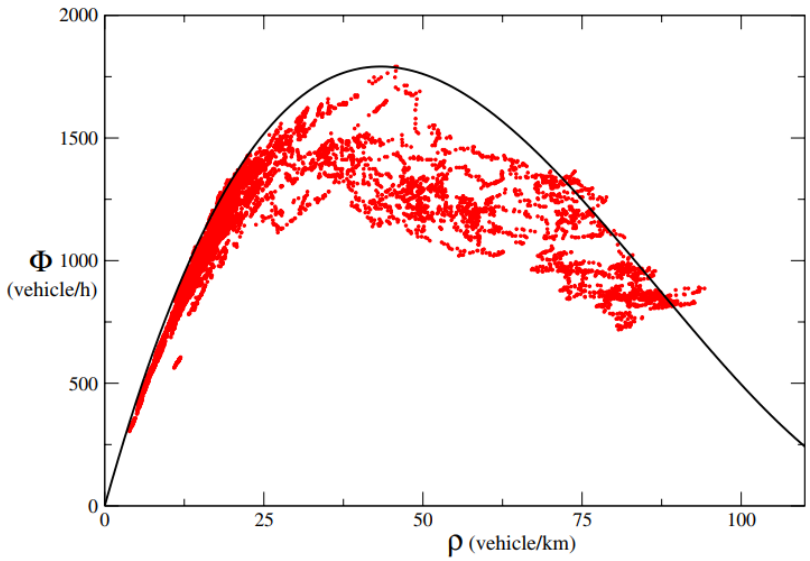

Figura 3.13: *Grafico di MFD come riportato in [14]. In rosso, i dati empirici misurati dal sistema MTS Emilia-Romagna*

incastrati nelle strade e che dunque allunghino la loro vita media se la densità veicolare è alta. Questo comportamento è evidente nella figura in scala semi-logaritmica, dove le piccole variazioni sono meglio apprezzabili. Cerchiamo ora di comprendere l'andamento asintotico delle code delle distribuzioni mostrate in Fig. 3.14. Osservando il plot in scala semi-logaritmica, sembra a primo sguardo che le code tendano a distribuirsi circa secondo una tendenza esponenziale (linea retta nel grafico semi-logaritmico). In realtà, effettuando svariati fit con ROOT ci si convince che una semplice distribuzione esponenziale non riesce e interpolare correttamente i dati di coda delle distribuzioni, se non per t abbastanza alti. Riproviamo dunque a fittare i dati raccolti con una distribuzione Gamma parametrizzata a (k, θ) (che, infatti, per t grandi si comporta circa come un esponenziale). Prima di procedere, ricordiamo alcune proprietà della distribuzione Gamma che ci tornano utili per interpretare i risultati:

Definizione 3.1 *La distribuzione Gamma $\Gamma(x; k, \theta)$ è la distribuzione di probabilità che si ottiene sommando fra di loro k variabili casuali esponenziali con parametro θ ($\exp(-x/\theta)$). La sua funzione di densità è, come già accennato:*

$$f(x; k, \theta) = \frac{x^{k-1} e^{-x/\theta}}{\Gamma(k)\theta^k} \tag{3.16}$$

Il parametro k è chiamato parametro di forma, mentre θ è detto parametro di scala.

Scegliamo quindi alcuni valori di N e operiamo un fit gamma (Fig. 3.15). In generale,

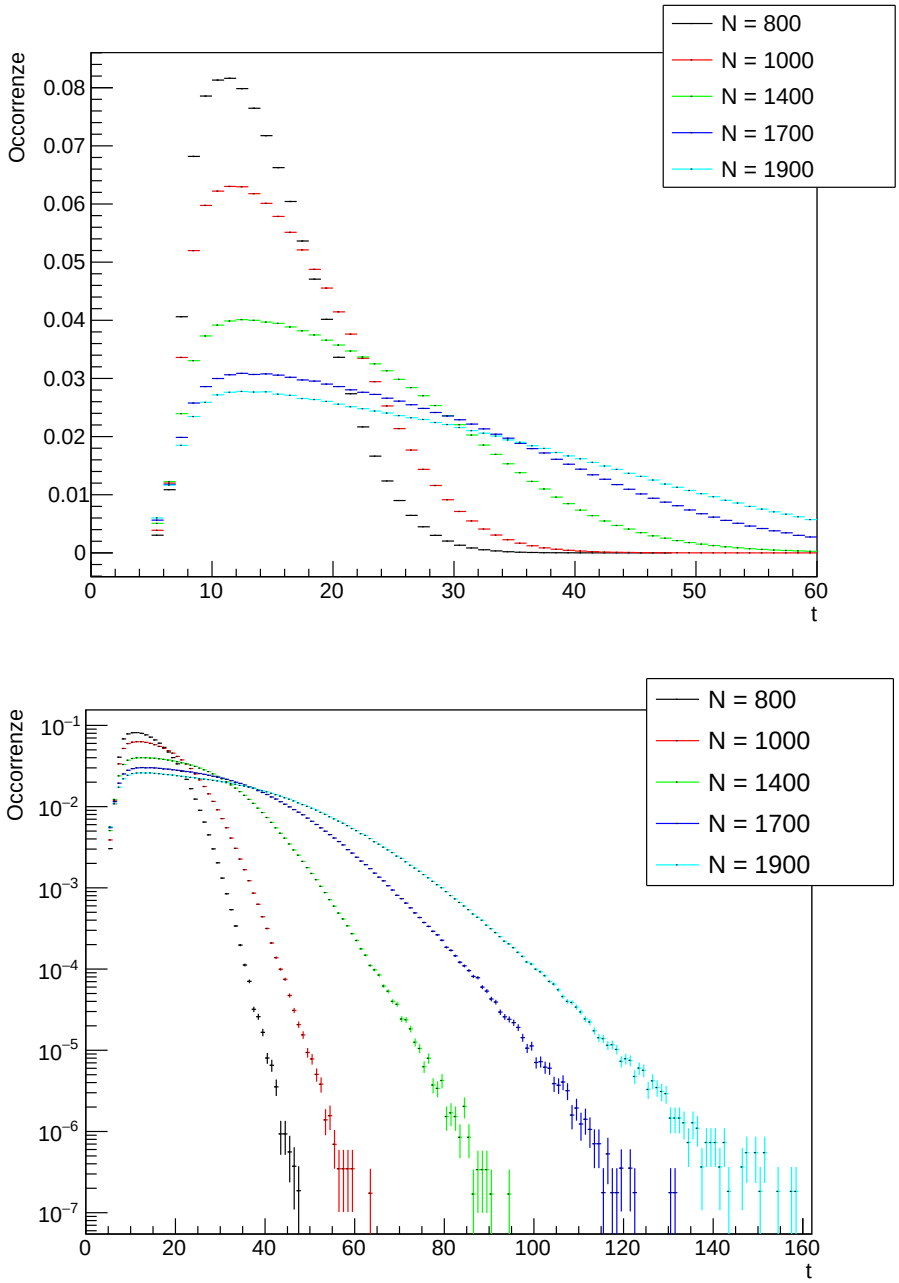

Figura 3.14: *Distribuzione dei lifespan degli agenti per cinque carichi di rete diversi* ($N = 800, 1000, 1400, 1700, 1900$)*,* $T = 10000$*.*

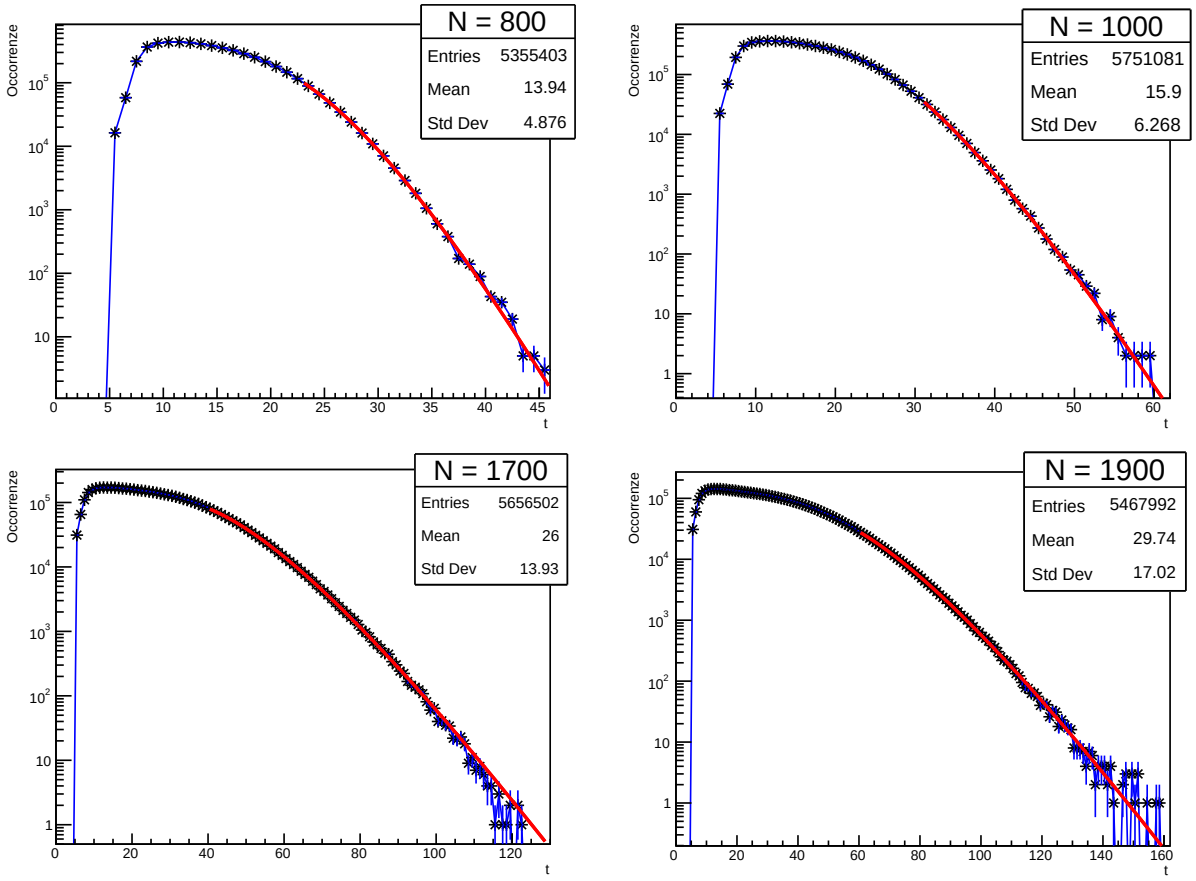

Figura 3.15: *Distribuzione del lifespan t per quattro carichi specifici, $N = 800, 1000, 1700, 1900$, con fit gamma in linea rossa proposto da ROOT. I valori dei bin sono evidenziati da stelline e riportati con la loro barra di errore*

l'accordo con il fit di ROOT valutato dal valore del χ^2 ridotto è abbastanza buono per tutti i valori di N in regime sottocritico. Ad esempio, prendendo il caso $N = 1700$, otteniamo come stima dei parametri di intesse:

$$k = 8.29 \pm 0.14, \quad \theta = 4.32 \pm 0.04, \tag{3.17}$$

$$\chi^2_{rid} = 1.15 \tag{3.18}$$

Possiamo interpretare i risultati nel seguente modo. Al carico $N = 1700$, per lifespan t abbastanza grandi (coda delle distribuzioni), gli agenti si muovono mediamente di $k \approx 8.3$ vertici. In ciascun vertice, il tempo medio di attesa per gli agenti è distribuito come una variabile esponenziale con vita media sul singolo nodo $\theta \approx 4.3$. Così facendo, l'attesa totale per il singolo agente è la somma di k variabili aleatorie esponenzialmente distribuite, esattamente come prescrive la distribuzione Gamma (si veda, in tal senso, [20] che ripropone la stessa tesi).

Momento primo distribuzione Ora cerchiamo di essere più quantitativi e valutiamo un'altra proprietà caratteristica delle distribuzioni dei lifespan t: il momento primo della distribuzione. Per ciascuna di esse, prendiamo il valore di aspettazione $\mathbb{E}[t] = \bar{t}$ (riportato nel box delle statistiche in Fig. 3.15, ad esempio) e realizziamo un grafico di quest'ultimo in funzione del carico di rete. Per valutare le incertezze da associare alla misura del valore medio $\bar{t}$ ripetiamo la run più volte con lo stesso carico ed estraiamo da queste misure la deviazione standard che fungerà da errore. I risultati ottenuti sono mostrati in Fig. 3.16 nella regione di free-flow. Notiamo che l'aumento del valore medio $\bar{t}$ sembra esponenziale nel numero di agenti N e tale ipotesi è ancora più evidente se il grafico viene mostrato in scala semi-logaritmica (riquadro in basso di Fig. 3.16). Effettuiamo quindi un fit tramite ROOT impostando una funzione del tipo $\bar{t}(N) = \exp(N/\alpha)$ ottenendo

$$\alpha = 1383 \pm 5 \tag{3.19}$$

Possiamo dunque concludere che il tempo medio di attesa dei veicoli cresce esponenzialmente col numero di agenti che popolano il network. In particolare, il parametro $\alpha \approx 1400$ ci dà un'idea approssimativa del carico di rete a partire dal quale gli effetti di rallentamento cominciano ad emergere.

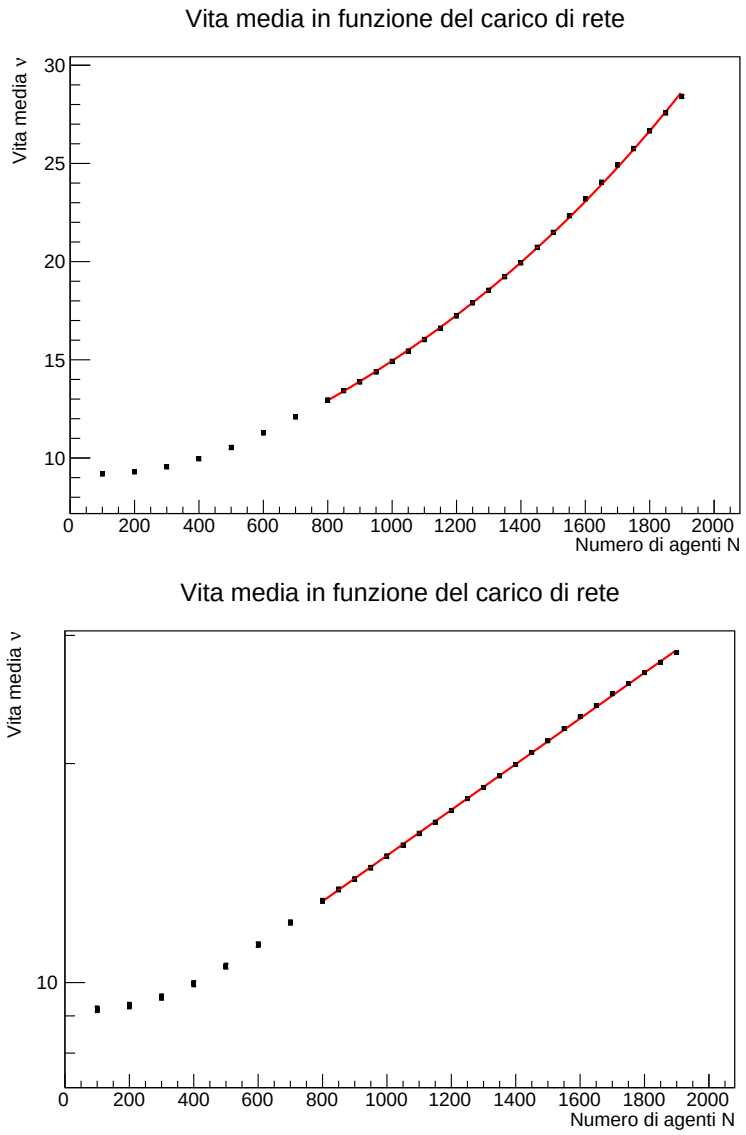

Figura 3.16: *In alto, valore medio $\bar{t}$ del tempo di attesa per gli agenti misurata tramite simulazione in funzione del numero di agenti nel grafo. Plottando la stessa curva ma in scala semi-logaritmica, in basso, è abbastanza chiaro che per N grandi il comportamento è di crescita esponenziale (retta in scala semi-logaritmica)*

Capitolo 4

Conclusioni

Il modello di tipo OD realizzato per questo lavoro, di cui si è parlato abbondantemente in Cap. 2, ci permette di approfondire e studiare in maniera dettagliata le proprietà del traffico veicolare in un contesto urbano omogeneo. Gli agenti nascono con un'origine e una destinazione casualmente definite e viaggiano nel grafo cercando di minimizzare il tempo di percorrenza. La dinamica fondamentale prevede alcuni meccanismi non-lineari, primi fra tutti la capienza massima dei nodi e il flusso massimo in uscita.

Le interazioni fra veicoli che popolano il network ne determinano lo stato stazionario ed è possibile riconoscere due regimi differenti (Cap. 3): uno di *free-flow*, in cui il traffico è al massimo leggermente rallentato ma mai congelato, e uno in cui compaiono nel sistema blocchi di vertici completamente congestionati (*gridlock*), in cui la circolazione veicolare è interamente interrotta. L'apparire di tali cluster segnala che il sistema sta subendo una transizione di fase.

È stata poi studiata la distribuzione delle lunghezze delle code nei nodi del sistema al variare del carico di rete N. Per valori di N sufficientemente bassi (o per nodi più periferici), le distribuzioni appaiono monomodali e a prima vista di tipo esponenziale. Aumentando N (o passando verso nodi più centrali), la distribuzione diventa bimodale e presenta due picchi apprezzabili, il primo vicino allo stato vuoto ($n_i = 0$), il secondo verso lo stato di massima capienza raggiunta ($n_i = N_{max}$). Questo fenomeno è effettivamente spiegabile ricorrendo ad un'analogia con un processo di Markov e un insieme di random walkers sullo stesso grafo.

A questo punto, si è passati ad individuare il punto critico del sistema, ossia il valore (o l'intervallo di valori) di N tale per cui il sistema si porta in uno stato stazionario completamente congestionato. Poiché il sistema non è al limite termodinamico (lunghezza strade finite, fluttuazioni nelle distribuzioni che non vanno a zero) possiamo solo stabilire un intervallo di valori che segnalano la transizione di fase (all'incirca $2000 < N < 3000$). Per valori di $N < 2000$, la rete rilassa verso un equilibrio dinamico e le fluttuazioni locali che potrebbero intasare i nodi vengono riassorbite prima che possano causare effetti su larga scala. Per $N > 3000$, il sistema collassa inevitabilmente nel gridlock di rete. Per

valori intermedi, l'apparire del cluster congestionato, almeno nel periodo di osservazione impiegato nella simulazione, non è sempre garantito ed è possibile affrontare la questione da un punto di vista probabilistico. È stata anche valutata la dimensione del blocco congestionato in relazione alla densità veicolare immessa in rete.

Si è, infine, passati ad analizzare il sistema in regime di free-flow (quindi $N < 2000$). In primis, è stato misurato il flusso medio di veicoli nell'unità di tempo discreta in relazione al numero di agenti nella rete. Per valori bassi di N, gli agenti si comportano come se fossero isolati poiché hanno basse possibilità di interagire con altri veicoli: il flusso del sistema è quindi pari precisamente a N poiché tutti gli agenti possono fluire, e questo coincide con le osservazioni effettuate. Aumentando il carico, aumentano le probabilità di interazione e la velocità media diminuisce: il flusso medio continua a crescere con N ma diminuisce la velocità di incremento, fino ad arrivare un plateau nella curva del flusso in prossimità del valore $N \approx 2000$. Tale curva è effettivamente osservata anche nella realtà e prende il nome, in letteratura scientifica, di *fundamental diagram of traffic flow*.

Infine, si è misurato il lifespan degli agenti in funzione del carico N, e cioè il tempo necessario a completare il tragitto origine-destinazione. La distribuzione di questa grandezza è ben descrivibile da una distribuzione Gamma. Inoltre, valutando il valor medio del lifespan $\bar{t}$ per ogni carico N, si è scoperto che il tempo medio di attesa cresce esponenzialmente con il numero di agenti. Tramite fit di ROOT, si è estratto come argomento dell'esponenziale $\alpha \approx 1400$, valore che può fornire un'idea indicativa del carico di rete oltre il quale gli effetti di interazione fra agenti sono tali da creare seri rallentamenti.

Appendice A

Implementazione tecnica del codice

Discutiamo brevemente ma in maniera esaustiva i dettagli più tecnici del codice.

Il programma utilizza librerie terze per il suo funzionamento. In particolare:

- La Boost Graph Library (BGL), per la gestione dei grafi e l'accesso ad algoritmi utili su di essi (ad esempio, Dijkstra). Questa è totalmente *headers-only*, dunque non richiede linking a librerie statiche/dinamiche
- Libreria SFML per la creazione e manipolazione di finestre grafiche in modo portabile

Il grafo La BGL consente di gestire e manipolare facilmente strutture dati che, per loro natura, si prestano ad una rappresentazione tramite grafi. Attraverso l'estensivo ricorso al *generic programming* [21] e ai *template*, la libreria consente di personalizzare la classe che incapsula il grafo dotandola di tutte le proprietà che l'utente desidera avere. In particolare, la BGL implementa due classi grafo primitive: `boost::adjacency_list_` e `boost::adjacency_matrix`. La distinzione fra le due riguarda il metodo interno con il quale i grafi vengono immagazzinati in memoria. Se un grafo è sparso, cioè ogni vertice è connesso a pochi altri nodi, allora è più conveniente optare per una rappresentazione tramite lista d'adiacenza (complessità che va come $O(|V| + |E|)$. La ricerca dei nodi adiacenti ad un dato nodo è molto efficiente con questa rappresentazione). Al contrario, se il grafo è particolarmente denso, allora la matrice di adiacenza è la scelta ottimale ($O(|V|^2)$). Poiché nel network che vogliamo costruire ogni vertice ha un numero abbastanza uniforme di vertici adiacenti e comunque nessun nodo supera il grado 4, allora scegliamo la classe primitiva `boost::adjacency_list_`. La classe che rappresenterà i grafi nella simulazione, sotto l'alias `Graph`, è definita come:

```
typedef boost::adjacency_list<boost::vecS, boost::vecS, boost::directedS,
VertexProperty, boost::property<boost::edge_weight_t, double, EdgeProperty>
```

e si tratta di un grafo orientato. La BGL permette poi di associare ai vertici e agli archi delle *bundled properties*, cioè proprietà che appartengono allo specifico vertice/arco e che lo contraddistinguono. Ricordando che un vertice è, nel modello che costruiamo, equivalente ad una strada, allora ci conviene associare a ciascuno di essi alcune informazioni riassunte e raccolte nello struct:

```
VertexProperty{
 int index;
 std::list<std::shared_pointer<Agent>> queue;
 bool full = false;
}
```

ossia un indice rappresentativo del nodo, una coda (cioè una lista di agenti che vivono nel nodo, di fatto rappresentano le vetture in coda sulla strada relativa) e una variabile booleana che indica se la strada ha raggiunto la capacità massima o meno.

L'inizializzazione e la gestione della simulazione viene affidata alla classe `Simulation`, che fra i suoi membri privati principali contiene due oggetti `Graph`: `m_graph` e `m_dual`. Il primo grafo codifica la rappresentazione fisica del network, il secondo quella duale.

Il network di base viene generato casualmente dal programma stesso secondo certi criteri. Il costruttutore di `Simulation` invoca la funzione `build_graph(int N)` che si occupa di costruire correttamente la rappresentazione fisica del grafo urbano in funzione del numero N di incroci che si vogliono realizzare. Il network di partenza è quindi una griglia $N \times N$ perfettamente regolare, in cui ogni nodo è un incrocio. Per rendere più realistico tale reticolo, sono state implementate due funzioni `add_diagonal_roads(..., int D, ...)` e `remove_random_edge(..., int R, ...)`. La prima funzione aggiunge randomicamente D strade diagonali pescando uniformemente un nodo nel grafo e una delle 4 possibili direzioni della strada diagonale che da quel nodo diparte. Un meccanismo interno di protezione impedisce che vengano aggiunti doppi edges e che l'aggiunta di troppe strade diagonali rompa la condizione di planarità che imponiamo sulla rappresentazione fisica del grafo. La seconda funzione rimuove casualmente R degli archi nel grafo (dunque rimuove un certo quantitativo di strade).

Una volta che è stato inizializzata la rappresentazione fisica `m_graph`, possiamo procedere ad estrarne la versione duale. A questa funzione è preposta la funzione `make_dual_graph (Graph& g, std::map<Vertex, Edge>& dual_map)` che ritorna la rappresentazione duale del grafo g salvandola nella variabile `m_dual`. Inoltre, passando una mappa vuota come parametro della funzione, questa la inizializza di modo che ad ogni vertice del duale sia associato il relativo arco nella rappresentazione fisica. Tale mappa diventa molto utile e tiene traccia della connessione fra i due oggetti grafo.

In alternativa, il programma offre la possibilità di caricare in memoria un grafo preesistente formattato secondo lo standard `.dot`. Così facendo, la simulazione viene girata sempre sulla stessa struttura di network e i risultati possono essere comparati riproducibilmente. Caricare un grafo già esistente, inoltre, richiede un tempo di esecuzione

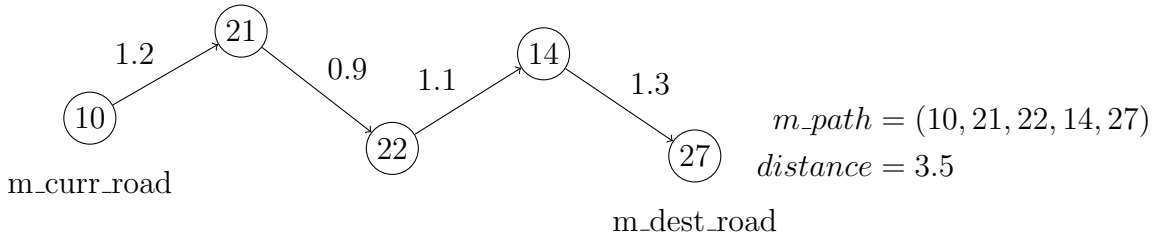

Figura A.1: *Esempio di costruzione della sequenza di vertici* `m_path` *avendo fissato un origine ed una destinazione.*

decisamente più basso rispetto alla sua creazione da zero (con conseguente creazione duale).

Gli agenti Nel modello che vogliamo costruire, gli *agenti* hanno un ruolo fondamentale. Dal punto di vista pratico, ogni agente viene rappresentato da un'istanza di una classe `Agent`:

```
class Agent {
 private:
 Vertex m_curr_road;
 Vertex m_dest_road;
 std::vector<Vertex> m_path;
 Graph* m_dual;
 ...
}
```

Quando un'istanza di `Agent` viene creata, il costruttore della classe si occupa di scegliere casualmente un vertice all'interno del grafo duale (quindi una strada), il cui valore viene passato a `m_curr_road`, nodo che rappresenterà l'origine dell'agente. Sempre in fase di costruzione, viene decisa anche la destinazione dell'agente. Tramite una chiamata alla funzione `get_vertex_based_on_dijkstra_shortest_path(Vertex const vi, Graph const &g)`, alla variabile interna `m_dest_road` viene assegnato il valore di un vertice nel grafo la cui distanza dal nodo di origine d sia almeno superiore ad una soglia, parametro della simulazione (`MIN_DIST_DIJKSTRA` $= 5$, quindi almeno 5 strade di distanza). Contestualmente, la funzione ricostruisce anche la sequenza di vertici `m_path` che l'agente dovrà percorrere per giungere a destinazione minimizzando la funzione di costo (Fig. A.1)

La dinamica Ad ogni ciclo di simulazione, gli agenti si muovono cercando di seguire il cammino individuato dalla variabile privata `m_path` seguendo le regole di dinamica esposte nel Cap. 2.

Una free function, `flow(Vertex v, ...)` ha il compito di gestire il flusso in uscita di agenti che risiedono nel vertice v. Impostando i parametri della simulazione `MAX_FLOW` e `MAX_CAP` (chiamati Φ e N_{max} in Cap. 2), tale funzione si assicura che il flusso massimo in uscita sia proprio `MAX_FLOW` e consente l'uscita di agenti solo se il nodo che vogliono raggiungere non è saturo. Dopo aver chiamato la funzione `flow` iterando sui vertici del grafo, viene chiamata la funzione `erase_agents(Vertex v, ...)` che si occupa di distruggere gli agenti arrivati a destinazione (valutando quindi se `m_curr_road == m_dest_road`). Inserendo tale algoritmo in un ciclo `while (t < TIME_MAX){}`, si potrà studiare il comportamento del sistema fino al tempo di osservazione $TIME\_MAX$ (chiamato T nel corpo del documento).

Bibliografia

[1] Giorgio Parisi. La fisica dei sistemi complessi. *Un mondo diverso: la fisica dei sistemi complessi*, 32.

[2] Nino Boccara. *Modeling Complex Systems*. Graduate Texts in Physics. Springer New York, New York, NY, 2010.

[3] Grégoire Nicolis and Catherine Nicolis. *Foundations of complex systems: nonlinear dynamics, statistical physics, information and prediction*. World Scientific, Hackensack, NJ, 2007.

[4] Guanwen Zeng, Daqing Li, Shengmin Guo, Liang Gao, Ziyou Gao, H. Eugene Stanley, and Shlomo Havlin. Switch between critical percolation modes in city traffic dynamics. *Proceedings of the National Academy of Sciences*, 116(1):23–28, January 2019.

[5] Lukas Ambühl, Monica Menendez, and Marta C. González. Understanding congestion propagation by combining percolation theory with the macroscopic fundamental diagram. *Communications Physics*, 6(1):26, February 2023.

[6] Meead Saberi, Homayoun Hamedmoghadam, Mudabber Ashfaq, Seyed Amir Hosseini, Ziyuan Gu, Sajjad Shafiei, Divya J. Nair, Vinayak Dixit, Lauren Gardner, S. Travis Waller, and Marta C. González. A simple contagion process describes spreading of traffic jams in urban networks. *Nature Communications*, 11(1):1616, April 2020.

[7] Boris S. Kerner. *Introduction to Modern Traffic Flow Theory and Control: The Long Road to Three-Phase Traffic Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[8] Daqing Li, Bowen Fu, Yunpeng Wang, Guangquan Lu, Yehiel Berezin, H. Eugene Stanley, and Shlomo Havlin. Percolation transition in dynamical traffic network with evolving critical bottlenecks. *Proceedings of the National Academy of Sciences*, 112(3):669–672, January 2015.

[9] Adolf D .May. Traffic flow fundamentals.

[10] Takashi Nagatani. The physics of traffic jams. *Reports on Progress in Physics*, 65(9):1331–1386, September 2002.

[11] Robin J. Wilson. *Introduction to graph theory*. Prentice Hall, Harlow Munich, 4. ed., [nachdr.] edition, 2009.

[12] Darij Grinberg. An introduction to graph theory, August 2023. arXiv:2308.04512 [math].

[13] Giancarlo Bongiovanni and Ivano Salvo. Algoritmo di Dijkstra, 2019.

[14] E. Andreotti, A. Bazzani, S. Rambaldi, N. Guglielmi, and P. Freguglia. Modeling traffic fluctuations and congestion on a road network. *Advances in Complex Systems*, 18(03n04):1550009, May 2015.

[15] Choong Nyoung Kim, Kyung Hoon Yang, and Jaekyung Kim. Human decision-making behavior and modeling effects. *Decision Support Systems*, 45(3):517–527, June 2008.

[16] T. Abrahamsson. Estimation of Origin-Destination Matrices Using Traffic Counts - A Literature Survey, May 1998. Num Pages: 37 Place: IIASA, Laxenburg, Austria Publisher: IR-98-021.

[17] Armando Bazzani and Lorenzo Di Meco. Random walks on graphs: finite transport effects and congestion transition. 2024.

[18] De Oliveira Luciana Renata. *Master Equation: Biological Applications and Thermodynamic Description*. PhD thesis, Alma Mater Studiorum - Universita di Bologna, 2014.

[19] Marco Uguccioni. *Introduzione alla meccanica statistica dei random walk su network*. PhD thesis.

[20] Jiwon Kim and Hani S. Mahmassani. Compound Gamma representation for modeling travel time variability in a traffic network. *Transportation Research Part B: Methodological*, 80:40–63, October 2015.

[21] Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Professional., 1st edition, December 2001.