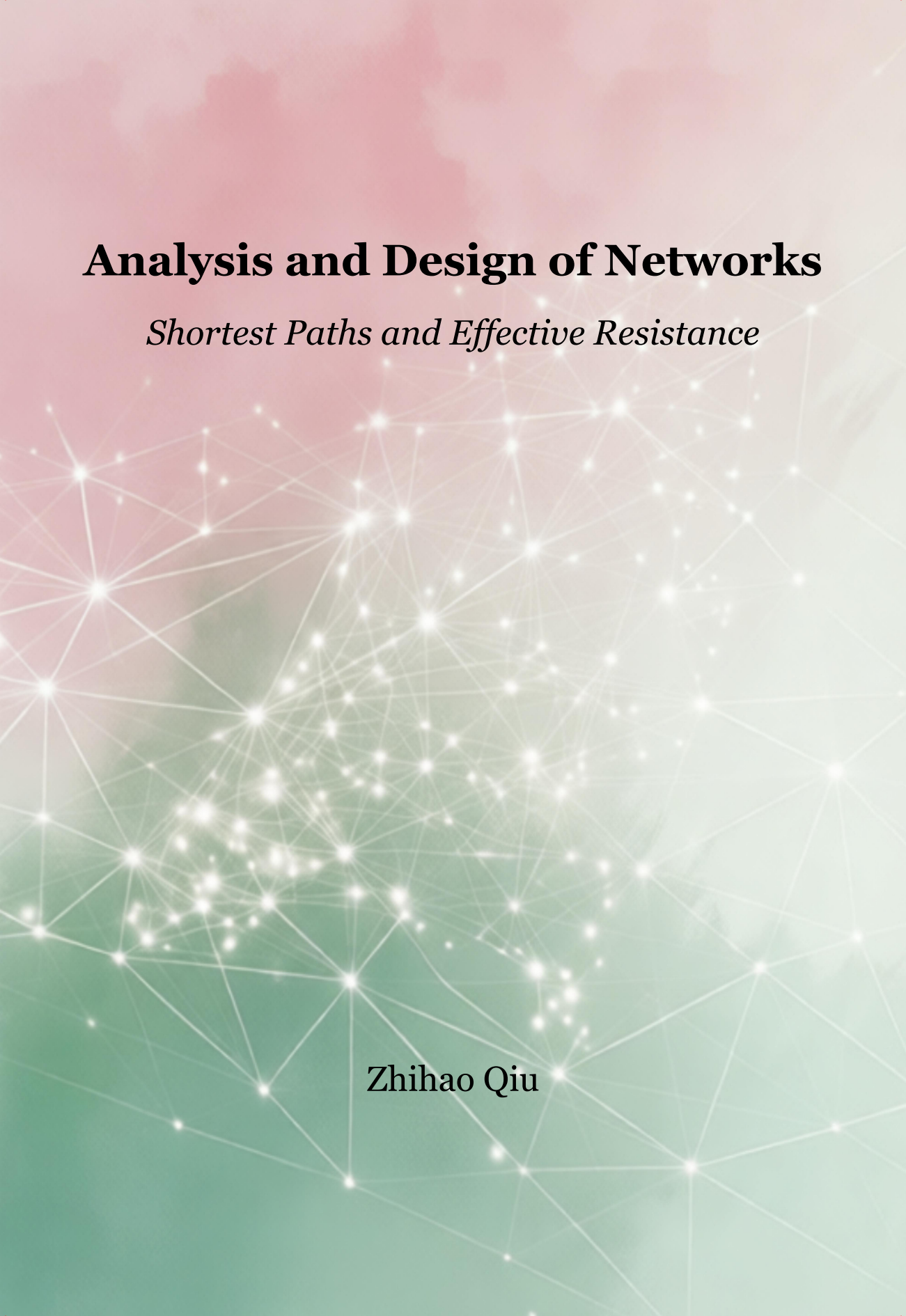
Analysis and Design of Networks

Shortest Paths and Effective Resistance



Zhihao Qiu

Analysis and Design of Networks

Shortest paths and effective resistance

ANALYSIS AND DESIGN OF NETWORKS

SHORTEST PATHS AND EFFECTIVE RESISTANCE

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus,
Prof. dr. ir. H. Bijl,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
Dinsdag, 7 April 2026, 12:30 uur

door

Zhihao QIU

Dit proefschrift is goedgekeurd door de

promotor: Prof. dr. ir. P.F.A. Van Mieghem
promotor: Dr. M.A. Kitsak

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. P.F.A. Van Mieghem	Technische Universiteit Delft
Dr. M.A. Kitsak	Technische Universiteit Delft

Onafhankelijke leden:

Prof. dr. M.E. Warnier	Technische Universiteit Delft
Dr. ir. E. Smeitink	KPN & Technische Universiteit Delft
Prof. dr. ir. S. Tang	Soochow University
Prof. dr. ir. R.E. Kooij	Technische Universiteit Delft
Dr. J.L.A. Dubbeldam	Technische Universiteit Delft
Prof. dr. ir. G.N. Gaydadjiev	Technische Universiteit Delft, reservelid

Kernwoorden: Shortest path, Inverse shortest path, Shortest path nodes prediction, Flow network, Power dissipation, Inverse effective resistance
Druk: ProefschriftMaken
Omslag: Z. QIU

ISBN: 978-94-6384-915-9

Een elektronische versie van dit proefschrift is verkrijgbaar op:
http://repository.tudelft.nl/.

To my family and my love Zhilin

CONTENTS

SUMMARY

The function of a network is generally related to the transport of items over the underlying graph. For example, in transportation networks, vehicles, trains, ships and aircraft deliver passengers and cargo to their destinations, while in telecommunication networks, IP packets are transmitted to enable data change between devices or users. To enhance the performance of item transmission in networks, such as reducing latency in telecommunication networks, we need to deepen the understanding of how a network supports efficient transport between two nodes and how transmission paths can be determined. Another important research topic concerns the design or construction of networks to meet users' evolving demands, either by developing new networks from scratch or by modifying existing ones. In this dissertation, the analysis and design of networks for item transmission are considered in three distinct areas.

We first explore the item transmission in networks where the item travels along a single path, usually the shortest path in most practical scenarios. In Chapter 2, we focus on the geometric properties of shortest paths in soft random geometric graphs (SRGGs) in 2-dimensional Euclidean space. We demonstrate that shortest paths are aligned along geodesic curves connecting shortest path endpoints. To quantify the strength of the alignment, we introduce two metrics: the average distance to the geodesic from shortest path nodes and the average path stretch. Both metrics decrease in larger SRGGs with short-range connections. Additionally, we find that the alignment strength is nonmonotonic with respect to the average degree of the SRGG and identify network properties maximizing shortest path alignment. Based on these observations, we establish that the alignment of shortest paths may be sufficiently strong to allow the identification of shortest path nodes based on their proximity to geodesic curves. Extensive simulations confirm that our geometric-based approach predicts shortest path nodes even when node positions in the latent space are not precisely known.

The second part of this dissertation investigates inverse problems of the shortest paths. Given predetermined demands on the shortest path weight, such as upper bounds on delay between nodes in telecommunication networks, we focus on how to dimension the network, both topology and link weights, so that transport along the shortest path between any pair of nodes satisfies the given demands. Chapter 3 introduces the inverse all shortest path problem (IASPP), which asks for a weighted adjacency matrix of a graph such that all the elements in the corresponding shortest path weight matrix do not exceed those of the given demand matrix. We propose the Descending Order Recovery (DOR) algorithm, which exactly solves the IASPP. The network provided by DOR minimized the number of links and the sum of link weights across all networks with the same shortest path weight matrix. As a complement to DOR, a second algorithm, Omega-based Link Removal (OLR), leverages effective resistance to solve the IASPP. Chapter 4 further extends IASPP to the modification of existing networks under changing transmission requirements. Inspired by practical scenarios, we study two extensions of IASPP.

The first develops a new graph whose shortest path weight matrix equals a predetermined demand matrix while minimizing the total change in link weights from the original graph and an exact approach is proposed. The second variant considers a more constrained scenario in which the graph topology remains unchanged. We propose an algorithm, "distance basis sum" (DBS), for tree topologies commonly found in domains such as telecommunications and industrial networks. Compared with classical approaches, DBS can improve computational efficiency while constructing a graph close to the optimal solution.

In the third part of the dissertation, we investigate item transmission in networks where items propagate along multiple paths between two nodes. In Chapter 5, we study the expected number of nodes and links involved in transferring items between the source and destination in random graphs. Then, we investigate the power dissipation associated with transportation at link level. Further, we address how to construct a network given a predetermined end-to-end power dissipation, which reduces to the "inverse effective resistance problem" of finding a network whose effective resistance matrix matches a given demand. We propose a heuristic algorithm, "Resistor Gap Pruning" (RGP), which provides sparse networks closely approximating the demand effective resistance and performs consistently across different demand scenarios.

SAMENVATTING

De functie van een netwerk is doorgaans gerelateerd aan het transport van objecten over de onderliggende graaf. Zo leveren in transportnetwerken voertuigen, treinen, schepen en vliegtuigen passagiers en geoderen af op hun bestemmingen, terwijl in telecommunicatienetwerken IP-pakketten worden verzonden om gegevensuitwisseling tussen apparaten of gebruikers mogelijk te maken. Om de uitvoering van het transport van objecten in netwerken te verbeteren, bijvoorbeeld door de latentie in telecommunicatienetwerken te verminderen, is het noodzakelijk beter te begrgpen hoe een netwerk efficiënt transport tussen twee knopen ondersteunt en hoe transmissiepaden kunnen worden bepaald. Een ander belangrijk onderzoeksgebied betreft het ontwerpen of bouwen van netwerken om te voldoen aan de veranderende eisen van gebruikers, hetzij door het ontwikkelen van nieuwe netwerken vanaf nul, hetzij door het aanpassen van bestaande netwerken. In dit proefschrift worden de analyse en het ontwerp van netwerken voor het transport van objecten onderzocht binnen drie afzonderlijke gebieden.

In het eerste deel bestuderen we het transport van objecten in netwerken waarbij het object zich langs één enkel pad verplaatst, doorgaans het kortste pad. In Hoofdstuk 2 richten we ons op de geometrische eigenschappen soft random geometric graphs, SRGG's in een tweedimensionale Euclidische ruimte. Op basis van deze waarnemingen voorspellen we de knooppunten van het kortste pad op basis van hun nabijheid tot geodetische krommen. Daarnaast stellen we vast dat de uitlijningssterkte niet-monotoon is ten opzichte van de gemiddelde graad van de SRGG en identificeren we netwerkeigenschappen die de uitlijning van kortste paden maximaliseren. Op basis van deze waarnemingen concluderen we dat de uitlijning van kortste paden voldoende sterk kan zijn om knopen op het kortste pad te identificeren op basis van hun nabijheid tot geodetische krommen.

Uitgebreide simulaties bevestigen dat onze geometrisch gebaseerde benadering knopen op het kortste pad kan voorspellen, zelfs wanneer de knoopposities in de latente ruimte niet exact bekend zijn.

Het tweede deel van dit proefschrift onderzoekt inverse problemen van kortste paden. Gegeven vooraf vastgestelde eisen aan het gewicht van het kortste pad, zoals bovengrenzen aan vertraging tussen knopen in telecommunicatienetwerken, richten we ons op de vraag hoe het netwerk — zowel de topologie als de booggewichten — kan worden gedimensioneerd zodat het transport langs het kortste pad tussen elk knooppaar aan de gestelde eisen voldoet. Hoofdstuk 3 introduceert het inverse all shortest path problem (IASPP), waarin wordt gevraagd om een gewogen bogenmatrix van een graaf zodanig dat alle elementen in de bijbehorende matrix van kortste-padgewichten niet groter zijn dan die van een gegeven demand matrix. We stellen het Descending Order Recovery (DOR)-algoritme voor, dat het IASPP exact oplost. Het door DOR verkregen netwerk minimaliseert het aantal verbindingen en de som van de booggewichten onder alle netwerken met dezelfde kortste-padgewichtmatrix. Als aanvulling op DOR

maakt een tweede algoritme, Omega-based boog Removal (OLR), gebruik van effectieve weerstand om het IASPP op te lossen. Hoofdstuk 4 breidt het IASPP verder uit naar de aanpassing van bestaande netwerken onder veranderende transmissie-eisen. Geïnspireerd door praktische scenario's bestuderen we twee varianten. De eerste ontwikkelt een nieuw netwerk waarvan de kortste-padges gelijk is aan een vooraf bepaalde demand matrix, terwijl de totale verandering in booggewichten ten opzichte van het oorspronkelijke netwerk wordt geminimaliseerd; hiervoor wordt een exacte aanpak voorgesteld. De tweede variant beschouwt een meer beperkend scenario waarin de netwerktopologie onveranderd blijft. We stellen een algoritme voor, "distance basis sum" (DBS), dat wordt toegepast op boomtopologieën die veel voorkomen in domeinen zoals telecommunicatie- en industriële netwerken. Vergeleken met klassieke benaderingen kan DBS de computationele efficiëntie verbeteren terwijl een netwerk wordt geconstrueerd dat dicht bij de optimale oplossing ligt.

In het derde deel van het proefschrift onderzoeken we het transport van objecten in netwerken waarbij objecten zich langs meerdere paden tussen twee knopen voortplanten. In Hoofdstuk 5 bestuderen we het verwachte aantal knopen en verbindingen dat betrokken is bij het transport van objecten tussen bron en bestemming in willekeurige netwerken. Vervolgens onderzoeken we het vermogensverlies dat gepaard gaat met transport op boogniveau. Daarnaast behandelen we de vraag hoe een netwerk kan worden geconstrueerd gegeven een vooraf bepaalde end-to-end vermogensdissipatie, wat kan worden gereduceerd tot het "inverse effectieve-weerstandsprobleem", waarin wordt gevraagd om een netwerk waarvan de effectieve-weerstandsmatrix overeenkomt met een gegeven vraag. We stellen een heuristisch algoritme voor, "Resistor Gap Pruning" (RGP), dat spaarzame netwerken oplevert die de gevraagde effectieve weerstand nauw benaderen en consistent presteren over verschillende vraagscenario's.

1

INTRODUCTION

Life is riding a bicycle.
To keep your balance,
you must keep moving.

Albert Einstein

1.1. COMPLEX NETWORK AND ITEM TRANSMISSION

Complex networks[1, 2] are ubiquitous in both natural (biological network, molecular interaction network, seismic network, etc.) and artificial systems, such as transportation, telecommunication, power grids, etc. Typically, a network is characterized by the topology of the underlying graph and the functional process of the network [3, 4], while the function of a network is generally related to the transport of items over the underlying graph[5, 6]. Depending on the type of transported items, a network can be classified[7, 8] as either a "path network" in which packets (e.g., IP packets, vehicles) are propagated, or a "flow network", e.g., electrical networks, gas networks, water networks, etc., in which the transported item is a flow.

To enhance the performance of item transmission in networks, one important research direction is to deepen the understanding of how a network supports an efficient transport between two nodes and how the transmission paths can be determined. A complementary direction focuses on the inverse problem: how to design or construct networks, either by developing a new one from scratch or by modifying existing networks, to satisfy the evolving demands on item transmission. In this dissertation, we provide a comprehensive study that encompasses both the analysis of item transmission in networks and the network construction given demands, considering both path networks and flow networks.

1

1.2. PATH NETWORK AND SHORTEST PATH

In a path network, the transport of items between two nodes i and j travels along a single path, defined as an ordered sequence of distinct nodes to traverse from node i and j. In most practical scenarios, such as vehicle routing in transportation networks or the well-known Open Shortest Path First (OSPF) protocol [4, 9] in communication networks, the transport typically follows the *the shortest path*, which minimizes the sum of link weights along the path.

The shortest path has been extensively studied in the literature due to its relevance to numerous real problems in routing [10, 11, 12, 13, 14, 15], transportation [16, 17, 18, 19, 20], spreading processes [21, 22, 23, 24, 25] and search [26, 27]. If the network of interest is simple[1], fully observable and does not have negative link weights, a large array of methods exists that allow solving the shortest path problem in polynomial time [28, 29, 30]. However, finding shortest paths in incomplete networks remains a challenging problem [26].

Compared with the extensive study on the shortest path problem, the research on the inverse direction, i.e., developing a network with given shortest paths and shortest path or shortest path weights, is relatively insufficient. A related challenge, termed the *inverse shortest path problem* has garnered attention in prior studies [31, 32, 33, 34, 35, 36]. The inverse shortest path problem focuses on obtaining networks where the predetermined paths can become the shortest paths. Another related formulation, known as the *Inverse all shortest path problem* [7, 37, 38, 39, 40], considers the network construction problem such that the corresponding shortest path weights in the resulting network satisfy predetermined demands.

1.3. FLOW NETWORK AND EFFECTIVE RESISTANCE

In flow networks, the item, such as current or water, propagates proportionally over all possible paths from the source node to the destination. Besides the cases where the item is flow, the investigation on flow networks also benefits many practical scenarios in which the transported item is a packet but can not be adequately described by the shortest path alone [41, 42, 43, 44]. For example, next-generation 6G communication systems aim to enable information be divided into smaller units and transmitted simultaneously over diverse and integrated paths, including satellite, fiber-optic and wireless links, to maximize coverage, reliability and transmission efficiency [45, 46]. Other examples include the epidemic spreading, where infections can propagate through all available connections. Similarly, the propagation of news or rumors in a social network, which generally does not follow the shortest path from a source node to the destination [42] but rather resembles a random walk process. In this dissertation, we will focus on the flow network using the current-flow analogy. Items are transferred following the principle of current flow through an electrical network, where links are resistors and nodes are junctions between the resistors.

One crucial topological characteristic of the flow networks is the *effective resistance* [6, 47, 48], which was originally defined for resistive electrical circuits as the potential (voltage) measured (resistance = voltage/current) between a pair of nodes (i, j) in the circuit,

[1] A simple graph has no multiple links between a same pair of nodes and also no self-loops

when a unit current is forced between these two nodes [49]. As a metric [6, 49] on a graph, the effective resistance provides a natural notation of distance between two nodes: a high resistance (potential difference) indicates that the two nodes are far away, while low resistance between nodes means that the nodes are closely connected [44]. As a result, smaller effective resistance values correspond to more efficient item transmission in flow networks. The effective resistance also specifies how power is dissipated over the entire network as the current flows between the source node and the destination. Geometrically, the elements in the effective resistance matrix equal the squared distances between vertices in the simplex of the graph [50]. Indeed, the effective resistance is a generalization[6] of the classical series and parallel formulas for the resistance to any graph configuration and has been extensively applied in numerous complex network problems, such as random walk [42, 51, 52], network sparsification [53, 54], robustness measures [6, 48], chemical graph theory [55, 56], etc.

1.4. DOCUMENT STRUCTURE

This dissertation focuses on the analysis and design of networks for item transmission. The dissertation consists of three parts, which are further divided into chapters.

I. Analysis and Inference of Shortest Path Nodes in SRGGs The first part focuses on item transmission along the *shortest paths*. Chapter 2 analyzes the geometric organization of shortest-path nodes in soft random geometric graphs (SRGGs) embedded in two-dimensional Euclidean space and shows that shortest paths align closely with geodesic curves connecting their endpoints. The geometric alignment is then exploited to identify shortest-path nodes, even when node positions in the latent space are uncertain.

II. Design of Networks Given Demands on Shortest Path Weights The second part addresses several network design problems: given demands on shortest path weights (and possibly network topology), how to design or construct a network to satisfy those demands. Chapter 3 introduces the inverse all shortest path problem (IASPP), which asks for a weighted adjacency matrix of a graph such that all the elements in the corresponding shortest path weight matrix do not exceed those of the given demand matrix. Two algorithms are proposed —one based on the structure of the shortest path and the other on effective resistance. We also provide a discussion on how perturbations in the effective resistance matrix affect the corresponding weighted adjacency matrix, potentially yielding negative link weights.

Chapter 4 extends IASPP to the modification of existing networks under changing transmission requirements. The first variant, IASPP that minimizes the adjustment of the sum of link weights (IASPP$_M$), constructs a new network that meets updated demands while minimizing total link weight changes. The second variant, IASPP with fixed adjacency matrix (IASPP$_F$), addresses cases where the network topology remains unchanged, with a specific solution developed for tree networks commonly found in telecommunication and industrial systems.

III. Item Transmission in Flow Networks and Flow Network Design The third part, presented in Chapter 5, studies item transmission in flow networks based on the

current-flow analogy, where items propagate proportionally over all available paths. We develop a framework to compute the expected number of nodes and links involved in transmission and analyze the power dissipation associated with the item transportation at link level. Finally, we propose a novel *Inverse Effective Resistance Problem*, which seeks a flow network whose effective resistance matrix satisfies prescribed power dissipation demands. A heuristic algorithm is introduced to generate sparse networks that closely approximate the effective resistance demand across diverse scenarios.

Finally, the contribution of this dissertation is summarized in Chapter 6.

2

GEOMETRIC ORGANIZATION AND INFERENCE OF SHORTEST PATH NODES IN SOFT RANDOM GEOMETRIC GRAPHS

The shortest path problem is related to many dynamic processes on networks, ranging from routing in communication networks to signaling in molecular interaction networks. When the network is only partially observable, solving the shortest path problem becomes nontrivial. Inspired by the shortest path problem in partially observable networks, we investigate the geometric properties of shortest paths in Euclidean Soft Random Geometric Graphs (SRGGs). We find that shortest paths are aligned along geodesic curves connecting shortest path endpoints. The strength of the shortest path alignment, as quantified by the average distance to geodesic from shortest path nodes and the average path stretch, is smaller for larger SRGGs with short-range connections. In addition, we find that the strength of the shortest path alignment is nonmonotonic with respect to the average degree of the SRGG and identify network properties maximizing shortest path alignment. Based on these observations, we establish that the alignment of shortest paths may be sufficiently strong to allow the identification of shortest path nodes based on their proximity to geodesic curves. Further, we find that the accuracy of geometric path finding, as quantified by precision scores, may exceed that of the network-based Dijkstra algorithm in situations where node positions in the latent space are not precisely known.

This chapter is based on a manuscript that is currently under review [57].

2.1. INTRODUCTION

The shortest path $\mathscr{P}^*_{ij}$ from a node i to a node j in a network G is an ordered sequence of distinct nodes to traverse from i to j, such that the sum of the link weights along the path is minimized. In this work, we consider unweighted undirected networks. In unweighted networks, the minimization of the sum of the link weights reduces to the minimization of the number of links, i.e., the hopcount. Further, in undirected networks, shortest paths are symmetric: $\mathscr{P}^*_{ij} = \mathscr{P}^*_{ji}$.

The shortest path has been extensively studied in the literature due to its relevance to a large array of real problems in routing [10, 11, 12, 13, 14, 15], transportation [16, 17, 18, 19, 20], spreading processes [21, 22, 23, 24, 25] and search [26, 27]. If the network of interest is simple, fully observable and without negative link weights, a large array of methods exists that allow solving the shortest path problem in polynomial time [28, 29, 30]. These *conventional* methods are guaranteed to solve the shortest path problem precisely, provided the entire network is fully observable. This assumption is, unfortunately, not met in many large networks. Indeed, large-scale social, technological, and biological networks are often *substantially* incomplete: the numbers of correctly mapped nodes and links in them tend to be smaller than those of unknown links and nodes [58, 59, 60].

The accuracy of conventional methods quickly decreases as the number of missing network components increases [26], due to the extreme sensitivity of shortest paths: even a single missing (false negative) or a spurious (false positive) link may change the path entirely. The conventional shortest-path methods, including the popular Dijkstra [28, 29] and Bellman-Ford algorithms [28, 30, 61, 62] algorithms rely on the iterative exploration of network topology and are very likely to fail if some of the links or nodes are not known [26].

A recent work proposed a novel method for finding shortest and nearly shortest path nodes, making use of their geometric alignment in hyperbolic embeddings of these networks: shortest path nodes are likely found in the geometric vicinity of geodesic curves connecting shortest path end points [26]. The caveat of the method is that it is limited to networks whose organization is consistent with hyperbolic geometry. It is now understood that large sparse networks characterized by strong clustering coefficient and scale-free degree distributions may be accurately embedded into hyperbolic spaces [63, 64, 65, 66], i.e., spaces with constant negative curvature. Node coordinates obtained as a result of hyperbolic embeddings allow one to draw a geodesic curve $\gamma(A,B)$ between any two points A and B and to find distance $d\big(C,\gamma(A,B)\big)$ from any other point C to $\gamma(A,B)$. Kitsak et al. [26] have demonstrated that hyperbolic embeddings can be used to find shortest paths even in extremely incomplete networks, with as few as 10% of network links present, leading to promising applications in Biomedicine and communication networks.

In our work, we ask if shortest paths in *Euclidean* geometric networks are sufficiently aligned and identifiable with geometric methods. On the one hand, our work is fueled by earlier findings in *Euclidean* random geometric graphs demonstrating a certain degree of alignment of shortest paths [67, 68, 69, 70, 71]. On the other hand, our work is inspired by the observation that many networks of interest are embedded into *Euclidean* spaces. Indeed, nodes of wireless communication networks are base stations placed in various geographic areas and the quality of communication between any two base stations de-

pends on the *Euclidean* geographic distance between them [72, 73, 74, 75, 76]. Another example is transportation networks, where nodes are intersections and road segments connecting them are links [77, 78]. Finally, of relevance to this study are critical infrastructure networks, e.g., power grids consisting of power stations connected with transmission lines [79, 80]. In all these networks, nodes are characterized by their geographic positions and links are typically connecting nearby nodes.

We conduct our studies in the context of the Soft Random Geometric Graph [67, 81, 82], which can be regarded as a special case of the Geometric Inhomogeneous Random Graph (GIRG) model [83, 84]. We find that in SRGGs shortest paths nodes tend to be significantly close to geodesic curves connecting path endpoints than expected by chance. In other words, we establish that shortest paths are geometrically aligned along geodesic curves. These alignments exhibit non-trivial behavior as a function of graph properties such as the average degree and the clustering coefficient. We find that under optimal conditions, the alignment of shortest paths allows one to find them using node coordinates in the latent space. Further, we demonstrate that the geometric path-finding methods have a competitive advantage over conventional path-finding methods in situations when network topology is not completely observable.

The rest of our work is organized as follows. We dedicate Section 2.2 to the problem formation. Here we define Soft Random Geometric Graphs (SRGG), to study the geometric alignment of shortest paths. In this section, we also define the distance to the geodesic method for finding shortest path nodes and path finding accuracy metrics. In Section 2.3.1, we study the alignment of shortest paths in Soft Random Geometric Graphs as a function of their parameters. Relying on the results of Section 2.3.1, we examine the problem of the identification of shortest path nodes in Section 2.3.2. We conclude our work with the discussion and outlook in Section 2.4.

2.2. METHODS

2.2.1. RANDOM GEOMETRIC GRAPH (RGG) AND SOFT RANDOM GEOMETRIC GRAPH (SRGG)

A random geometric graph (RGG) [85, 86] is an undirected graph constructed by randomly placing N nodes into a metric space $\mathscr{M}$ and connecting any two nodes i and j if the distance d_{ij} between them is less than a certain radius r. In our work, we consider RGGs built in a 2-dimensional unit square, where N nodes are placed uniformly at random. If the connection radius $r \ll 1$, the link density in the RGG with L links,

$$p_{\mathrm{RGG}} \equiv \frac{L}{L_{\max}} \approx \pi r^2, \tag{2.1}$$

where $L_{\max} = \frac{1}{2}N(N-1)$, see, e.g., [87].

Thus, ignoring boundary effects, the probability that a node i has exactly k connections is

$$\Pr[D_i = k] = \binom{N-1}{k} \left(p_{\mathrm{RGG}}\right)^k \left(1 - p_{\mathrm{RGG}}\right)^{N-k-1}, \tag{2.2}$$

and the expected degree of a node is

$$E[D_i] = (N-1)\, p_{\mathrm{RGG}}, \tag{2.3}$$

where p_{RGG} is the link density given by Eq. (2.1).

A soft random geometric graph (SRGG) can be regarded as a generalization of the random geometric graph in which links are established independently with probabilities that are functions of distances between the nodes in $\mathscr{M}$. An SRGG can be generated as follows:

1. Sprinkle N nodes with probability density function (pdf) $\rho(x)$ into latent space $\mathscr{M}$.
2. Calculate the distances $d_{\mathscr{M}}(ij)$ between each (i, j) node pair.
3. Connect each (i, j) node pair independently with probability p_{ij}, which is a function of distances $d_{\mathscr{M}}(ij)$ between nodes i and j, $p_{ij} = f\left(d_{\mathscr{M}}(ij)\right)$.

While, in principle, $f : R^+ \to [0,1]$ can be any function, f is often selected to be a decreasing function to model networks where connections over short distances are preferable. One common choice for f is

$$f(d) = \beta e^{-\left(\frac{d}{d_0}\right)^{\eta}}, \tag{2.4}$$

where $\beta \in (0,1]$, $d_0 > 0$, and $\eta > 0$ are tunable parameters. In particular, the $\eta = 1$ case is known as the Waxman graph model [86, 88], and $\eta = \infty$ case reduces the SRGG model to the RGG. The choice of $\eta \in [2, 6.5]$ corresponds to the Rayleigh fading connection function [73, 89] used to model radio signal's amplitude and phase fluctuations in multipath environments.

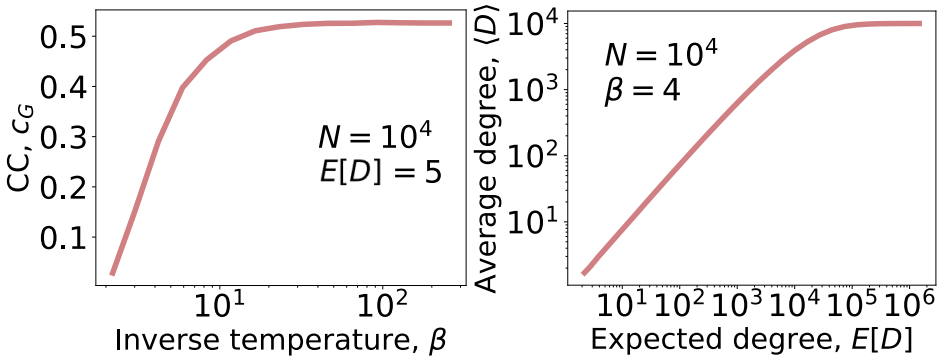
In this work, we study SRGGs with a different connection probability function, which is inspired by the Fermi-Dirac statistics:

$$f(d) = \frac{1}{1 + \left(\frac{d}{d_0}\right)^{\beta}}, \tag{2.5}$$

where $d_0 > 0$ and $\beta > 0$ are again parameters tuning topological properties of resulting graphs. The SRGG model is closely related to the $\mathbb{S}^D$ heterogeneous graph model [90] if node hidden variables in $\mathbb{S}^D$ are constant, and is also closely related to the Geometric Inhomogeneous Random Graph (GIRG) model [91].

An SRGG model with assigned node coordinates can be regarded as a system with quantum states with energy levels ϵ_{ij} are functions of distances d_{ij} between nodes [90]. Links in the SRGG can, therefore, be regarded as fermions occupying corresponding quantum states, at most one fermion per state [90]. Parameter β then corresponds to the inverse temperature $\beta = \frac{1}{\tau}$. At low temperatures (high β values) only quantum states with the lowest energy levels are occupied, i.e., only short-distance connections are possible. Higher temperatures (low β values), on the other hand, allow the occupation of higher energy states, which correspond to longer distance connections in the SRGG. Following this analogy, we refer to β as the inverse temperature parameter throughout the text.

We consider the case of SRGGs built on a unit square, with uniform node density, $\rho(\mathbf{x}) = 1$. Ignoring the boundary effects, one can establish that the expected degree of



(a) Clustering coefficient vs. inverse temperature **(b)** Real average degree vs. input expected degree

Figure 2.1: (a) shows an example of how the clustering coefficient changes with the input inverse temperature in SRGGs. (b) shows an example of how the real average degree changes with the input expected degree in SRGGs. The x-axis is the input expected degree, while the y-axis is the real average degree.

the SRGG model is given by

$$E[D] = E[D_i] \approx \frac{cNd_0^2}{\beta \sin \frac{2\pi}{\beta}}, \tag{2.6}$$

when $\beta > 2$, and the proportionality coefficient $c > 0$, arizing due to finite size effects, can be tuned numerically. Eq. (2.6) indicates that parameter d_0 can be used to tune the expected degree of SRGG graphs. Indeed, greater d_0 values effectively rescale distances in the SRGG, allowing nodes to connect to other nodes over larger distances. Eq. (2.6) holds well for small and moderate expected degree values, Fig. 2.1(b). For large expected degree values, the boundaries of the unit square become non-negligible, making Eq. (2.6) less precise. As a result, in the paper we carefully distinguish the theoretical expected degree $E[D]$ of the SRGG model, from the realized average degree $\langle D \rangle$ of the SRGG, which we measure as $\langle D \rangle \equiv 2L/N$.

While inverse temperature β also affects the average degree, it plays a far more important role by tuning the effective range of connections. When β is large connections are mostly limited to distances $d \leq d_0$. Smaller β values increase the probability of connections at larger distances. As a result, β allows one to tune the average clustering coefficient in SRGGs, see Fig. 2.1(a).

In our work, we parametrize SRGGs by the inverse temperature parameter β and expected degree $E[D]$. To generate an SRGG with desired parameters β and $E[D]$, we use Eq. (2.6) to determine parameter d_0. We then use parameters d_0 and β to determine connection probabilities for all node pairs in the network.

2.2.2. ALIGNMENT OF SHORTEST PATHS ALONG GEODESIC CURVES

A geodesic $\gamma(i, j)$ is the shortest length curve connecting points i and j in space $\mathscr{M}$. The length of the geodesic $\gamma(i, j)$ is the distance d_{ij} between points i and j. In the case of

Euclidean space, the geodesic $\gamma(i,j)$ is a straight line between i and j, while the distance d_{ij} represents the corresponding *Euclidean* distance.

The distance $d(q,\gamma(i,j))$ from a point q to the geodesic $\gamma(i,j)$ is the distance from point q to point $x \in \gamma(i,j)$ on the geodesic, such that distance $d(q,x)$ is minimized.

$$d(q,\gamma(i,j)) \equiv \min d(q,x), \tag{2.7}$$

$$\text{s.t.} \quad x \in \gamma(i,j). \tag{2.8}$$

In *Euclidean* spaces, the distance from the point to a geodesic $d(q,\gamma(i,j))$ is the length of the perpendicular line segment from q to the geodesic line $\gamma(i,j)$.

The hopcount h_{ij} of a path from a node i to a node j is the number of links of that path from node i to node j. Further, we define the path stretch S_{ij} as the total geometric distance accumulated along the path $\mathscr{P}^*_{ij}$:

$$S_{ij} \equiv \sum_{l \in \mathscr{P}^*_{ij}} d_l \tag{2.9}$$

where d_l is the *Euclidean* distance between the two end nodes of link l. Fig. 2.2 (a) depicts an example of a shortest path, a geodesic, the distance from a node on the path to the geodesic, and the shortest path stretch in an SRGG.

In an unweighted SRGG, there can be more than one shortest path $\mathscr{P}_{ij}$ between a node pair (i,j). The set $W_{P^*_{ij}}$ of all nodes constituting shortest paths between i and j are called "shortest path nodes", and $|W_{P^*_{ij}}|$ is the cardinality, i.e., the number of elements in the set.

We quantify the alignment of shortest paths along geodesic curves with two methods. The first method is to compare distances from shortest path nodes to the geodesic with those from randomly selected nodes. The second method is to compute the shortest path stretch, i.e., the distance accumulated along shortest path links. The smaller the stretch the stronger the alignment, Fig. 2.2(a).

To assess the extent of the geometric alignment of shortest path nodes in SRGGs with given parameters, we design the following experiments:

1. Generate an SRGG with the network size N, the expected degree $E[D]$ and the inverse temperature parameter β. We then select a number of node pairs at random.
2. For each node pair (i,j), we obtain all shortest path nodes $W_{\mathscr{P}^*_{ij}}$.
3. Find the geodesic curve γ_{ij} connecting node i and j.
4. Compute the distance to geodesic $d(q,\gamma(i,j))$ for all shortest path nodes $q \in W_{\mathscr{P}^*_{ij}}$ except for node i and j. Record the average, maximum and minimum distance for all shortest path nodes.
5. Randomly select $m \equiv |W_{\mathscr{P}^*_{ij}}| - 2$ nodes from the graph and compute the distance to the geodesic for each node.
6. If there is a single shortest path connecting i and j, compute its stretch. If there are multiple shortest paths connecting i and j, select one shortest path at random and compute its stretch.

2.2.3. PREDICTING SHORTEST PATH NODES USING DISTANCE TO GEODESIC

Kitsak et al. [26] proposed a method to infer shortest path nodes by measuring node proximities to geodesic curves connecting shortest path endpoints in hyperbolic embeddings of networks. We extend this approach to *Euclidean* SRGGs. We refer to this method as the "distance to the geodesic".

To infer shortest path nodes connecting nodes i and j of an SRGG in latent space $\mathscr{M}$, we first find the geodesic $\gamma(i,j)$ connecting the nodes in $\mathscr{M}$. Then, for each network node q excluding i and j, we find the distance to the geodesic $d\big(q,\gamma(i,j)\big)$, and rank all network nodes in increasing order of distance to the geodesic. The closest to the geodesic nodes are then viewed as likely candidates for the shortest path nodes.

As a reference for comparison, we also infer shortest path nodes by first reconstructing network links and then identifying shortest path nodes using the Dijkstra algorithm. We consider two reconstruction strategies, denoted as SRGG + net and RGG + net.

In the SRGG + net experiment, we use known parameters β and $E[D]$ and node coordinates $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$ to rebuild 10^3 random graph instances G_ℓ, $\ell = 1,...,10^3$. For each instance ℓ, we find the set of shortest path nodes $W_{\mathscr{P}^*_{ij}(\ell)}$ using the Dijkstra algorithm on graph G_ℓ. As a result, shortest path node candidates are nodes that appear most frequently in all $W_{\mathscr{P}^*_{ij}(\ell)}$ sets.

The RGG+net experiment is a special case of the SRGG+net experiment when $\beta = \infty$. Since $\beta = \infty$ case corresponds to connecting nodes to geometrically closest nodes, the network reconstruction procedure is equivalent to the reconstruction of random geometric graphs, explaining the name of the experiment. For a specific network size N, expected degree $E[D]$ and node coordinates $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$, there exists a unique RGG realization. Hence, the RGG experiment reconstructs only one graph G. All shortest path nodes $W_{\mathscr{P}^*_{ij}}$ in graph G are predicted as shortest path node candidates by our RGG strategy.

To evaluate the accuracy of our geometry-based method, we use the statistical precision metric [92, 93]. Our evaluation procedure can be summarized as follows:

1. Generate an SRGG G with required parameters N, $E[D]$ and β. Select a number of node pairs in G uniformly at random.

2. For each (i,j) node pair, we find all the shortest path nodes $W_{\mathscr{P}^*_{ij}}$. Nodes in $W_{\mathscr{P}^*_{ij}}$ are regarded as ground-truth set.

3. Obtain the geodesic $\gamma(i,j)$ and compute the distance to the geodesic $d(q,\gamma(i,j))$ for all nodes, excluding nodes i and j.

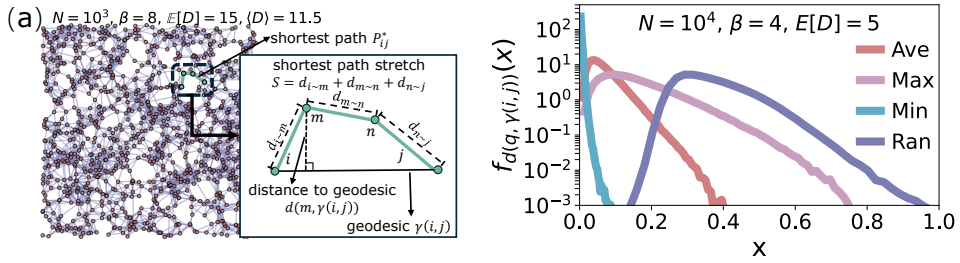
4. Determine the set of $\mathscr{S}$ nodes with the smallest distance to geodesic values, such that $|\mathscr{S}| = |W_{\mathscr{P}^*_{ij}}|$.

5. Compute the Precision $= |\mathscr{S} \bigcap W_{\mathscr{P}^*_{ij}}|/|\mathscr{S}|$.

2.3. RESULTS

2.3.1. QUANTIFYING THE ALIGNMENT OF SHORTEST PATHS ALONG GEODESIC CURVES IN SRGG MODELS

We begin our work with the analysis of the alignment of shortest paths along geodesic curves in Soft Random Geometric Graphs (SRGGs). We generate an instance of a soft random geometric graph (SRGG) of $N = 10^4$ nodes, connected with Fermi-Dirac connection probabilities given by Eq. (2.5) with inverse temperature $\beta = 4$, and parameter d_0 corresponding to the expected degree of $E[D] = 5$. Fig. 2.2(b) depicts the distributions of the maximum, the minimum, and the average distance to geodesic from shortest path nodes for 10^6 randomly selected path endpoints within the SRGG, indicating that all three measures are substantially smaller than expected by chance. Our observation is not specific to the choice of SRGG parameters. We observe similar shortest path alignment properties in SRGGs with different parameters, Fig. A.1, and the same pattern persists when different link connection probability functions are used, Fig. A.4.

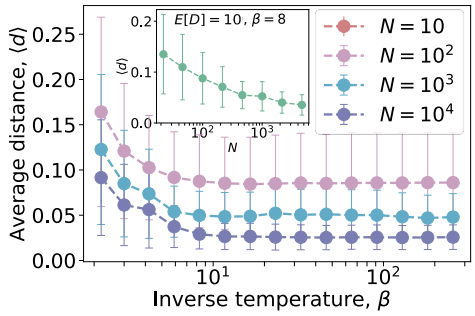


(a) Toy example of an SRGG

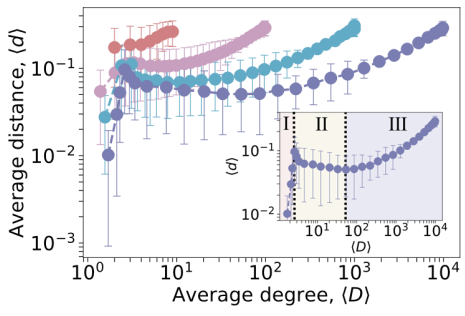
(b) Distribution of distance to geodesic

Figure 2.2: (a) A toy example of a geodesic, distance from a node to the geodesic and a shortest path (highlighted by green) with the corresponding shortest path stretch indicated by the black dashed line along the green path. Henceforth, all experiments are performed for SRGGs of $N = 10^4$ nodes, the expected degree $E[D] = 5$ and inverse temperature $\beta = 4$. (b) The distribution of the average (ave), maximum (max) and minimum (min) distances between shortest path nodes and geodesics connecting shortest path endpoints. We compare the resulting distributions to (ran) those between randomly chosen nodes and the geodesic. Within an SRGG G, we randomly select 10^6 shortest path endpoint pairs and for each (i, j) pair, we find shortest path set $\mathscr{P}^*_{ij}$, geodesic $\gamma(i, j)$. For shortest path nodes in $\mathscr{P}^*_{ij}$, we compute the minimum (Min), maximum (Max) and average (Ave) distance to the geodesic. In addition, for each shortest path set $\mathscr{P}^*_{ij}$, we created a random set $\mathscr{P}^{\text{rand}}_{ij}$ of the same cardinality containing randomly selected nodes. Using $\mathscr{P}^{\text{rand}}_{ij}$, we computed average distances to the geodesic (Ran).

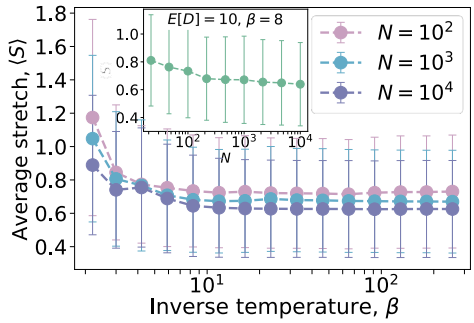
After establishing that shortest path nodes are closer to geodesics than expected by chance, we next investigate how topological properties of SRGGs affect path alignments. We measure the average distance to geodesic $\langle d \rangle$ and the average shortest path stretch $\langle S \rangle$, as described in Section 2.2.2. For each set of SRGG parameters, we generate 100 SRGG realizations if $N \leq 100$ and 1 realization for $N > 100$. For each generated graph G_i, $1 \leq i \leq 100$, we consider 1,000 shortest paths between randomly selected connected node pairs when $N > 100$, and all connected node pairs in graphs with $N \leq 100$.



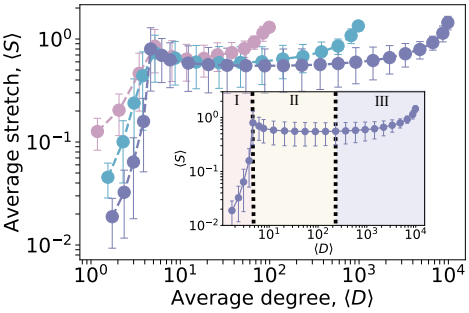
(a) Distance to geodesic vs. inverse temperature



(b) Distance to geodesic vs. average degree



(c) Stretch vs. inverse temperature



(d) Stretch vs. average degree

Figure 2.3: (a) The average distance from shortest path nodes to geodesics $\langle d \rangle$ as a function of inverse temperature β for SRGGs of $E[D] = 10$ and variable size N. The inset displays the average distance from shortest path nodes to geodesics $\langle d \rangle$ as a function of the number of nodes N for SRGGs with expected degree $E[D] = 10$ and inverse temperature $\beta = 8$. (b) The average distance from shortest path nodes to geodesics $\langle d \rangle$ as a function of average degree $\langle D \rangle$ for SRGGs of variable size N and the inverse temperature $\beta = 4$. The inset depicts the three phases discussed in the text. (c) The average shortest path stretch $\langle S \rangle$ as a function of inverse temperature β for SRGGs of $E[D] = 10$ and variable size N. The inset displays the average shortest path stretch $\langle S \rangle$ as a function of the number of nodes N for SRGGs with expected degree $E[D] = 10$ and inverse temperature $\beta = 8$. (d) The average shortest path stretch $\langle S \rangle$ as a function of average degree $\langle D \rangle$ for SRGGs of variable size N and the inverse temperature $\beta = 128$. Each point in all panels corresponds to the average of 1,000 randomly selected shortest paths, while the error bars quantify the standard deviation.

Fig. 2.3 depicts the average shortest path distance to geodesic $\langle d \rangle$ and the average shortest path stretch $\langle S \rangle$ as a function of inverse temperature β and the average degree $\langle D \rangle$.

We observe that the average distance to the geodesic and the average path stretch decrease as β increases, Fig. 2.3(a,c). This observation is expected. Indeed, by design, larger inverse temperature β values favor short-distance connections in the SRGG, see Eq. (2.5). As a result, shortest path nodes are located closer to each other and closer to the geodesic connecting path endpoints, decreasing the average distance to the geodesic and the average shortest path stretch.

We also observe that the average distance to geodesic $\langle d \rangle$ and the average stretch $\langle S \rangle$ decrease as the network size N increases, if the remaining SRGG parameters are held constant. Indeed, the density of nodes in the latent space increases as the network size

N increases. Therefore, distances between connected nodes must decrease on average to preserve the expected degree $E[D]$. It is the decrease of distances between connected nodes that leads to the decrease of the average distance to geodesic $\langle d \rangle$ and the average shortest path stretch $\langle S \rangle$, observed in Figs. 2.3(a,c).

The average distance to geodesics $\langle d \rangle$ and the average path stretch $\langle S \rangle$ exhibit a non-monotonic behavior as a function of the graph average degree $\langle D \rangle$, Fig. 2.3(b,d). Initially, $\langle d \rangle$ and $\langle S \rangle$ increase as as the average degree $\langle D \rangle$ increases, Phase I in Fig. 2.3(b,d). After reaching the local maximum, $\langle d \rangle$ and $\langle S \rangle$ decrease as the average degree increases, Phase II, reaching minimum values $\langle d_{\min} \rangle$ $\langle S_{\min} \rangle$. After reaching local minima, the average distance $\langle d \rangle$ and the average stretch $\langle S \rangle$ resume their growth as a function of average degree in Phase III, see Fig. 2.3(b,d).

We hypothesize that the initial growth of the average distance $\langle d \rangle$ and the average shortest path stretch $\langle S \rangle$ is related to the onset of the giant connected component (GCC) in the SRGGs. Indeed, when the average degree $\langle D \rangle$ is small, the network is fragmented into a large number of small disjoint subgraphs, Fig. A.5(b) in Appendix A.3. As the average degree increases, disjoint subgraphs eventually merge into a giant connected component (GCC) of a size comparable to that of the entire graph. Right above the critical value of $\langle D \rangle = \langle D \rangle_c$ corresponding to the onset of the GCC, most of the nodes in the graph are connected, but the shortest paths between them may be arbitrarily long, see Fig. A.5(a) and Appendix A.3. As a result, the local maximum in the distance to geodesic $\langle d \rangle$ and the shortest path stretch $\langle S \rangle$ at the border of Phases I and II may coincide with the critical value $\langle D \rangle_c$ corresponding to the emergence of the GCC. To check if this is the case, we compared the $\langle D \rangle$ values corresponding to the local maximum of the distance to geodesic $\langle d \rangle$ with those corresponding to the GCC emergence for a rangle of SRGG parameters. As seen in Fig. 2.4(b), the two values are nearly equal, leading to a strong correlation, Pearson $R = 0.99$. The results support our hypothesis that the growth of the distance to geodesic $\langle d \rangle$ and the shortest path stretch $\langle S \rangle$ in Phase I are driven by the formation of the giant connected component in SRGG.

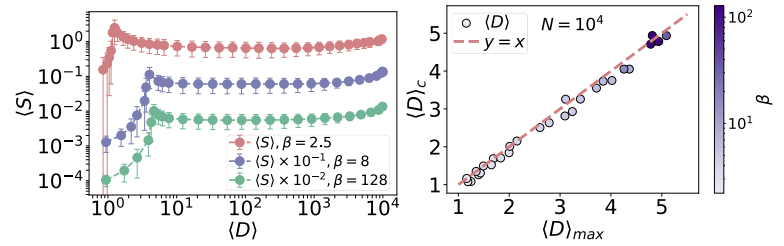
After reaching the local maximum, the average path distance to geodesic $\langle d \rangle$ and the average path stretch $\langle S \rangle$ decrease as the average degree $\langle D \rangle$ increases until they reach their local minima at the border of Phase II and Phase III, Fig. 2.3(b,d).

These non-monotonic behaviors of $\langle S \rangle$ and $\langle d \rangle$ are likely the results of the interplay of two competing effects. On the one hand, as the average degree increases, the average number of hops of a shortest path decreases, improving its alignment along a geodesic. On the other hand, larger average degrees result in larger distances between connected nodes, leading to larger deviations from the geodesic and larger stretch values.

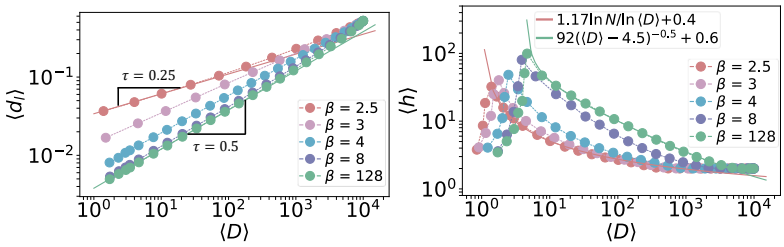
To quantify these effects for the average shortest path stretch $\langle S \rangle$, we approximate it as a product of the average link length $\langle d_l \rangle$ and the average hopcount $\langle h \rangle$,

$$\langle S \rangle \approx \langle d_l \rangle \langle h \rangle. \tag{2.10}$$

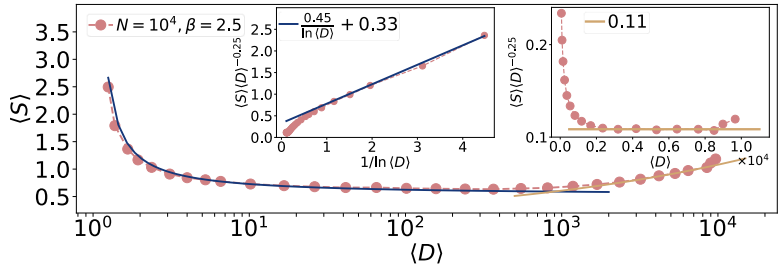
We estimate the average link length $\langle d_l \rangle$ using Bayes' rule. The conditional probability $\rho\left(D_{ij} = x | A_{ij} = 1\right)$ that the distance D_{ij} between two randomly chosen nodes i and j equals x, given that the two nodes are connected, $A_{ij} = 1$, depends on the conditional probability nodes i and j are connected, given that the distance between them equals x, $P\left(A_{ij} = 1 | D_{ij} = x\right)$, the probability density $\rho\left(D_{ij} = x\right)$ for a distance between



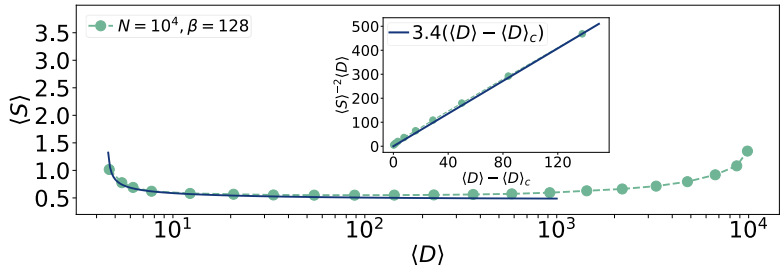
(a) Average stretch vs. average degree **(b)** Critical degree between Phase I and II



(c) Average link length vs. average degree **(d)** Average hopcount vs. average degree



(e) Average stretch vs. average degree for small inverse temperature



(f) Average stretch vs. average degree for large inverse temperature

Figure 2.4: (a) The average shortest path stretch $\langle S\rangle$ as a function of the average degree $\langle D\rangle$ for inverse temperatures $\beta = 1$, $\beta = 8$, and $\beta = 128$. The plots for beta $\beta = 8$ and $\beta = 128$ are scaled by multiplicative factors 10^{-1} and 10^{-2}, respectively, to avoid their overlap. The error bars depict standard deviation values. (b) Critical degree values $\langle D\rangle_c$ corresponding to the onset of the GCC as a function of expected degrees $\langle D\rangle_{\max}$ corresponding to the largest distance to geodesic. The scatter plot includes 20 points corresponding to different inverse temperature values β selected uniformly at random from the $(2,128]$ interval. (c) The average link length $\langle d\rangle_\ell$ as a function of the average degree $\langle D\rangle$ in SRGGs with different inverse temperatures β. The solid lines are power-law curves with exponents $\tau = \min\left(1/2, \beta/2-1\right)$. (d) The average shortest path hopcount $\langle h\rangle$ as a function of the average degree $\langle D\rangle$ in SRGGs with different inverse temperatures β. The solid lines are empirical fits for $\beta = 2.5$ and $\beta = 128$ values. (e,f) The average shortest path stretch $\langle S\rangle$ as a function of the average degree $\langle D\rangle$ for inverse temperatures (e) $\beta = 2.5$ and (f) $\beta = 128$. (e) The left inset depicts $\langle S\rangle\langle D\rangle^{-\tau}$ as a function of $\ln\langle D\rangle^{-1}$. Note that $\langle S\rangle\langle D\rangle^{-\tau} \propto \ln\langle D\rangle^{-1}$, confirming the low-degree behavior of the shortest path stretch $\langle S\rangle$. The right inset depicts $\langle S\rangle\langle D\rangle^{-\tau}$ as a function of $\langle D\rangle$. Note that $\langle S\rangle\langle D\rangle^{-\tau}$ reaches a horizontal asymptote, confirming the power-law scaling of the shortest path stretch for large $\langle D\rangle$ values. (f) The inset depicts $\langle S\rangle^{-2}\langle D\rangle$ as a function of $\langle D\rangle - \langle D\rangle_c$, testing the low-degree behavior of the shortest path stretch. Note that the shortest path stretch $\langle S\rangle$ is nearly independent of $\langle D\rangle$ for a large range of $\langle D\rangle$ values.

two randomly chosen nodes in the unit square, and the probability $P\left(A_{ij}=1\right)$ that two randomly chosen SRGG nodes are connected. Since $P\left(A_{ij}=1|D_{ij}=x\right)$ is nothing else but the connection probability function $f(x)$ prescribed by Eq. (2.5), we have

$$\rho\left(D_{ij}=x|A_{ij}=1\right)=\frac{f(x)\rho\left(D_{ij}=x\right)}{P\left(A_{ij}=1\right)}. \tag{2.11}$$

Then the average link length $\langle d_l\rangle$ can be estimated, see Appendix A.4, as

$$\langle d_l\rangle=\frac{\int x f(x)\,\rho\left(D_{ij}=x\right)\mathrm{d}x}{\int f(x)\,\rho\left(D_{ij}=x\right)\mathrm{d}x}\sim\langle D\rangle^{\tau}, \tag{2.12}$$

where $\tau=1/2$ for $\beta>3$ and $\tau=\beta/2-1$ for $\beta\in(2,3)$, see Fig. 2.4(c).

Fig. 2.4(d) depicts the behavior of the average hopcount $\langle h\rangle$ as a function of the average degree $\langle D\rangle$. For large inverse temperatures β, SRGG is similar to an RGG, and we can approximate the behavior of the average hopcount as

$$\langle h\rangle\approx\frac{\langle\delta\rangle}{[\langle D\rangle-\langle D\rangle_c]^{1/2}}+C,\quad\beta\gg3 \tag{2.13}$$

where $\langle\delta\rangle$ is the average distance between two randomly chosen points in the unit square, $\langle D\rangle_c\approx4.51$ is the critical value of the average degree corresponding to the onset of the GCC [86], and C is a constant. For small inverse temperatures β, see Fig. 2.4(d), SRGG is akin to a configurational model, and

$$\langle h\rangle\approx\frac{\ln N}{\ln\langle D\rangle}+A,\quad\beta\in(2,3), \tag{2.14}$$

where N is the number of nodes and for networks with Binomial degree distribution $A=\frac{1}{2}-\frac{\gamma-2\log(1-1/\langle D\rangle)}{\log\langle D\rangle}-\frac{2\,E[\log W|W>0]}{\log\langle D\rangle}$. Here γ is the Euler constant and W is the almost sure limit of the normalized branching process [4, 94]. For graphs with expected degree $E[D]>1.3$ and a binomial degree distribution, $E[\log W\mid W>0]\in(-0.2,0.2)$ [4]. Since $E[\log W\mid W>0]$ is difficult to measure numerically, in this works we estimate the entire constant A by fitting $\langle h\rangle$ vs $\langle D\rangle$ curves, see Fig. 2.4(d).

The scaling behaviors of $\langle d_l\rangle$ and $\langle h\rangle$ given in Eqs. (2.12), (2.13), and (2.14) account for the observed dependence of the average stretch $\langle S\rangle$ on the average degree $\langle D\rangle$ in phases II and III.

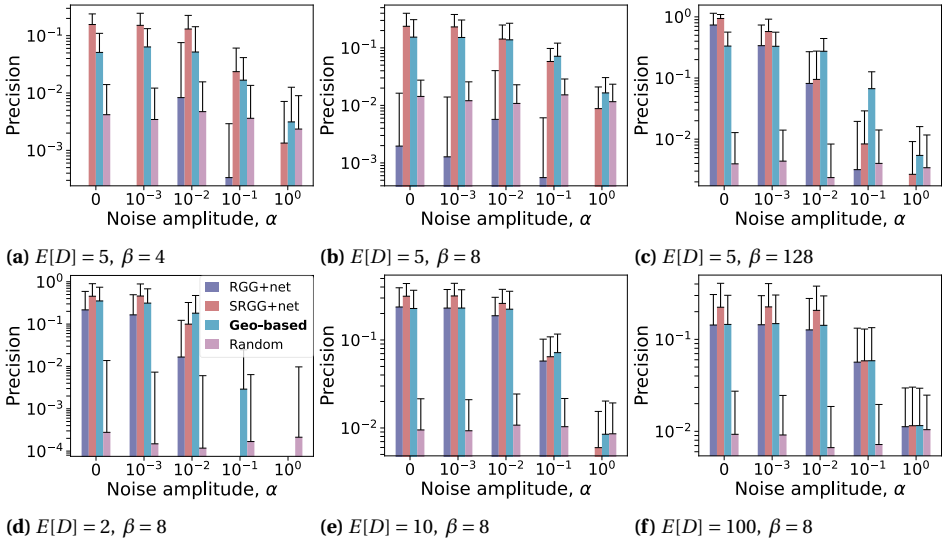
For small inverse temperatures, $2<\beta<3$, the average stretch scales as

$$\langle S\rangle\propto\left(\frac{\ln N}{\ln\langle D\rangle}+C\right)\langle D\rangle^{\frac{\beta}{2}-1}$$

Consequently, at small $\langle D\rangle$ (the onset of phase II), the stretch exhibits the asymptotic behavior $\langle S\rangle\sim\langle D\rangle^{\beta/2-1}/\ln\langle D\rangle$ [Fig. 2.4(e)], whereas at large $\langle D\rangle$ (deep in phase III) the logarithmic correction becomes negligible and $\langle S\rangle\sim\langle D\rangle^{\beta/2-1}$, Fig. 2.4(e).

For large inverse temperatures, $\beta\gg3$, the average stretch follows

$$\langle S\rangle\approx\left(\frac{\langle\delta\rangle}{[\langle D\rangle-\langle D\rangle_c]^{1/2}}+C\right)\langle D\rangle^{1/2}$$



(a) $E[D] = 5,\ \beta = 4$ **(b)** $E[D] = 5,\ \beta = 8$ **(c)** $E[D] = 5,\ \beta = 128$

(d) $E[D] = 2,\ \beta = 8$ **(e)** $E[D] = 10,\ \beta = 8$ **(f)** $E[D] = 100,\ \beta = 8$

Figure 2.5: Precision of shortest path finding using (i) distance to geodesic (Geo-based) and SRGG (SRGG+network) and RGG (RGG+network) reconstructions. Panels display average precision as a function of the coordinate noise amplitude α and panels correspond to different combinations of $E[D]$ and β parameters. All experiments are performed using SRGG networks of $N = 10^4$ nodes. Each bar is the average precision computed for 100 randomly chosen node pairs and the error bars depict standard deviations.

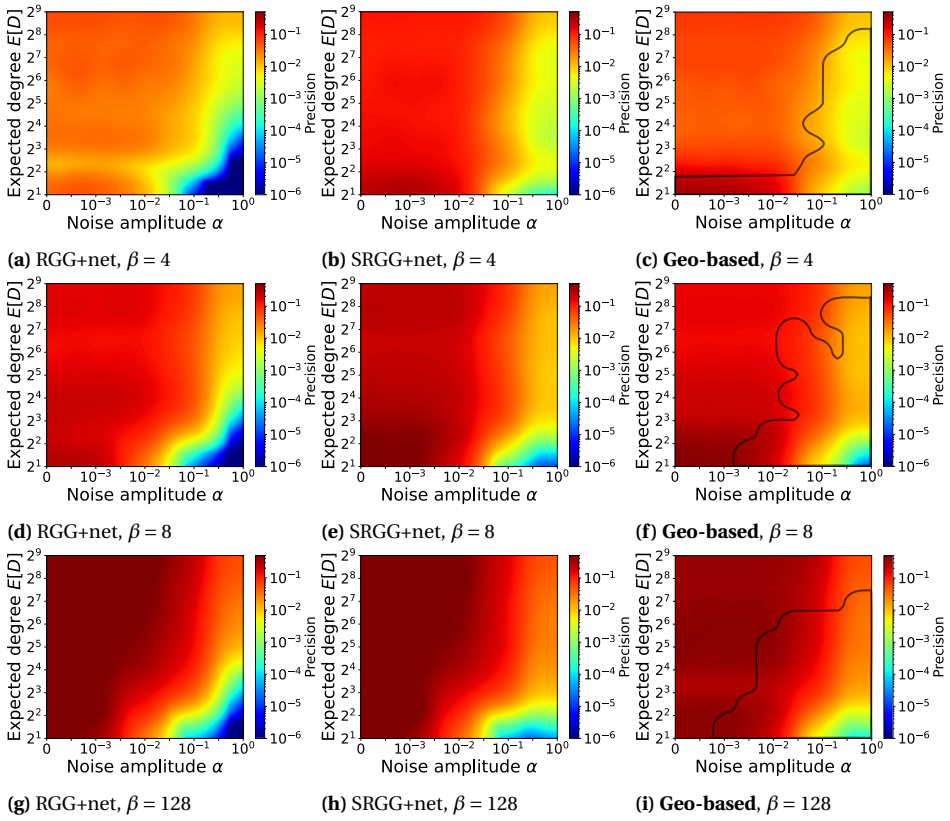
As a consequence, for $\langle D \rangle \gg \langle D \rangle_c$ the dependence on $\langle D \rangle$ is weak, yielding stretch values that are approximately constant, as seen in Fig. 2.4(f).

Overall, our results indicate that shortest paths in Soft Random Geometric Graphs are aligned along geodesic curves connecting shortest path endpoints. The alignment, as quantified by the average distance to geodesic $\langle d \rangle$ and the shortest path stretch $\langle S \rangle$, increases as the inverse temperature β and the SRGG size N increase. In contrast, the dependence of shortest-path alignment on the average degree $\langle D \rangle$ is nonmonotonic, reflecting the competition between two opposing effects: the reduction of the average hop count $\langle h \rangle$ and the concurrent increase of the average link length $\langle d_\ell \rangle$ as connectivity increases.

2.3.2. FINDING SHORTEST PATH NODES IN SOFT RANDOM GEOMETRIC GRAPHS WITH NOISE.

After observing and quantifying the alignment of shortest paths along geodesic curves in soft random geometric graphs, we next ask if shortest path nodes are identifiable based on their proximity to geodesic curves. To identify shortest path nodes connecting an $i - j$ node pair of interest, we draw a geodesic line $\gamma(i, j)$ connecting path endpoints i and j and then for every other network node $\ell \neq i \neq j$ we compute its distance to geodesic $d\big(\ell, \gamma(i, j)\big)$. The smaller the distance, the higher the likelihood for node ℓ to be a shortest path node, see Methods.

Fig. 2.5 displays the results of our path finding experiments for a combination of in-



(a) RGG+net, $\beta = 4$ **(b)** SRGG+net, $\beta = 4$ **(c)** **Geo-based**, $\beta = 4$

(d) RGG+net, $\beta = 8$ **(e)** SRGG+net, $\beta = 8$ **(f)** **Geo-based**, $\beta = 8$

(g) RGG+net, $\beta = 128$ **(h)** SRGG+net, $\beta = 128$ **(i)** **Geo-based**, $\beta = 128$

Figure 2.6: Heatmap of precision. The x-axis is the noise amplitude, while the y-axis represents the expected degree. For each expected degree $E[D]$ and inverse temperature β, a 10^4-node SRGG is generated and the coordinates are respectively blurred with noise amplitude α. The heatmaps consist of 9×9 points, each point corresponding to the average of 100 randomly selected node pairs in a (blurred) SRGG. In the region enclosed by the black line, the distance to geodesic approach (Geo-based) achieves higher precision compared with the SRGG (SRGG+network) and RGG (RGG+network) reconstruction approaches.

verse temperature β and expected degree $E[D]$ parameters. We observe that Precision score for the distance to the geodesic in finding shortest path nodes is significantly larger than expected by chance.

Inspired by this observation, we ask if there are plausible circumstances in which the distance to the geodesic can find shortest-path nodes with accuracy higher than known state-of-the-art methods. Clearly, if network G is fully observable, existing graph theoretic algorithms, e.g., the Dijkstra algorithm, can find shortest path nodes precisely and in polynomial time. A much more challenging situation is when only network nodes and their approximate positions are known in network G. In more precise terms, we assumed that SRGG links are not known. At the same time, all SRGG nodes are known, but their

coordinates are blurred with random noise of amplitude $\alpha > 0$:

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i + \alpha X_i;\ \tilde{\mathbf{y}}_i = \mathbf{y}_i + \alpha Y_i, \tag{2.15}$$

where $i = 1,\ldots,N$, X_i and Y_i are random variables drawn from the uniform distribution $U[-1,1]$.

In this situation, one can still find shortest path nodes using the distance to the geodesic method with blurred node coordinates $\{\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i\}$. To find shortest path nodes with the Dijkstra network-based algorithm, one needs to first reconstruct SRGG links. To this end, we consider two strategies. The SRGG + net strategy presumes that we know the connection probability function $f(x)$ used to generate the original SRGG graph. In this case, one first generates an SRGG instance with the same connection probability function $f(x)$ and then applies the Dijkstra algorithm to the resulting graph to find shortest path nodes. Another strategy, which we call RGG + net, assumes that the average degree $\langle D \rangle$ of the graph is known, but the connection probability function $f(x)$ is not. In this case, we treat the graph at hand as a random geometric graph, we compute its effective connectivity radius R based on the number of nodes and the average degree, and then connect all nodes if their blurred positions are closer than R, see Methods (Section 2.2.3).

By comparing the precision of the distance to geodesics with those of the alternative SRGG + net and RGG + net strategies, we make several observations. When node coordinates are known precisely, $\alpha = 0$, the SRGG + net and RGG + net strategies offer substantially higher Precision score than those of the distance to the geodesic, as shown in Fig. 2.5. This result is expected since precisely known node coordinates allow for the reconstruction of graphs with relatively high accuracy.

Second, both SRGG + net and RGG + net strategies become more accurate in finding shortest path nodes as the inverse temperature β increases, Fig. 2.5(a-c). We explain this observation by the decrease in randomness in SRGG networks: at higher β values, connections are preferentially established at shorter distances, and distinct realizations of reconstructed networks are less different. As a result, reconstructed networks G_ℓ better recreate the original network G, allowing for the identification of original shortest path nodes with higher precision, as shown in Fig. 2.5.

Third, the RGG + net strategy tends to be less accurate than the SRGG + net. However, the relative accuracy of the former seems to increase as the inverse temperature β increases. This result is also expected since larger β values favor shorter links, effectively making SRGGs closer to RGGs.

When comparing the distance to the geodesic approach to the SRGG + net and RGG + net strategies, we find that the precision of the distance to geodesic approach may exceed that of the SRGG + net and RGG + net when the noise amplitude α is large, Fig. 2.6, indicating that the distance to the geodesic approach is more robust to coordinate uncertainities than network reconstruction approaches SRGG + net and RGG + net. In the case of noise magnitude α, distances between nodes may be perturbed by at most 2α and lead to changes in some node rankings with respect to distances to geodesics. The effects of node coordinates on network reconstruction approaches are potentially more disrupting. This is the case since link probabilities in the SRGG depend nonlinearly on distances between the nodes and even small changes in network links may significantly alter shortest paths. As a result, shortest paths in reconstructed networks may signifi-

cantly differ from those in the original network, reducing the accuracy of network reconstruction strategies.

To explore the limits of shortest path finding with the distance to geodesic, we next conduct systematic experiments for ranges of noise amplitude $\alpha \in [0,1]$, average degree $E[D] \in [2, 2^9]$, and different inverse temperature values. As seen in Fig. 2.6, the accuracy of all path finding methods decreases as the average degree $E[D]$ decreases and coordinate uncertainty α increases. At the same time, the accuracy of the distance to the geodesic approach is less sensitive to noise, resulting in low average degree and high noise regions, Fig. 2.6(c), (f), and (i), where the distance to the geodesic method is more accurate than network reconstruction strategies.

2.4. DISCUSSION

In summary, we analyzed the alignment of shortest paths along geodesic lines connecting shortest path endpoints in Soft Random Geometric Graphs (SRGGs). Some of our results are intuitive and expected. In particular, we found that the alignments of shortest paths, as measured by the distance to the geodesic, and the path stretch become stronger in SRGGs as the number of nodes N and the inverse temperature β increase.

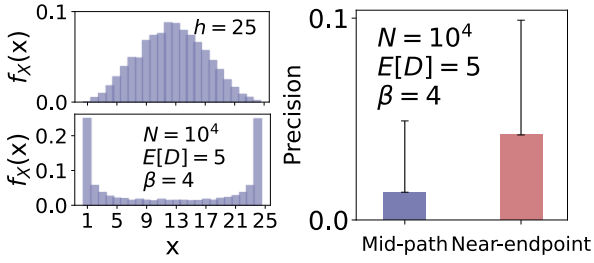
Nevertheless, the non-monotonic behavior of shortest path alignment as a function of the average degree is far from trivial. As the average degree increases, two competing phenomena take place. On the one hand, the hopcount of shortest paths decreases, limiting the extent of *wandering* of shortest paths. On the other hand, larger average degree values can only be achieved by allowing nodes to connect over larger *Euclidean* distances. As a result, distances between adjacent nodes in a shortest path increase, allowing for large distances from shortest path nodes to geodesic curves. The competition between these two effects results in the non-monotonic behavior of the distance to the geodesic as a function of the average degree.

We found that the alignments of shortest paths along geodesic curves in *Euclidean* SRGGs are sufficiently strong to enable shortest path finding, even near criticality where deviation measures such as $\langle S \rangle$ and $\langle d \rangle$ reach their maximum values. Shortest path nodes in *Euclidean* SRGGs can be identified using distance to geodesic, similar to random hyperbolic graphs [26]. The accuracy of such geometric shortest path finding is higher for graphs with a larger inverse temperature β, in agreement with our path alignment experiments.

The distance to the geodesic method uses node coordinates to find nodes closest to the geodesic connecting shortest path endpoints, and can be used in situations when node links are not known. In contrast, to enable traditional network-based methods in this situation, one needs first to reconstruct network links, which can be a highly nontrivial task. We find that the best advantage of the distance to the geodesic method is achieved for small average degree values and larger node coordinate uncertainties. While the geometric alignment of shortest path nodes is not necessarily the strongest in this case, network reconstruction methods are even less accurate, offering better prospects for using network geometry in finding shortest path nodes.

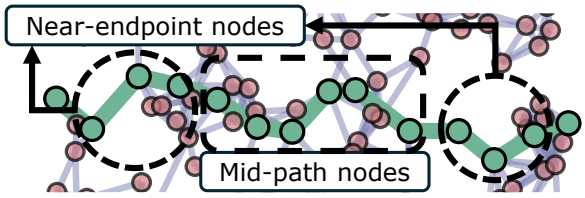
Our results are not specific to the choice of the connection probability function. While most of our analysis is based on SRGGs with the Fermi-Dirac connection probability function, Eq. (2.5), we obtain qualitatively similar results for the alignment of shortest

paths for Waxman SRGGs characterized by the exponentially decaying connection probabilities, Eq. (2.4), see Fig. A.4.



(a) Distribution of node position

(b) Predicting Near-endpoint and Mid-path nodes



(c) Example of near-endpoint and Mid-path nodes

Figure 2.7: (a) top depicts a distribution example of most node positions X in the shortest path corresponding to the largest distance to the geodesic. We considered the shortest paths of fixed hopcount $h = 25$. For each shortest path, we found the most remote node and measured its position $X \in 1, ..., 24$. (a) bottom depicts the distribution of most node positions X in the shortest path corresponding to the smallest distance to the geodesic. Additional distributions for SRGGs with different network parameters are provided in Appendix A.2. (b) shows the precision of predicting the shortest path nodes located near the path endpoint (Near-endpoint) and in the middle of the path (Mid-path), by the geometric-based method. (c) shows an example illustrating the division of Near-endpoint nodes and Mid-path nodes on a shortest path.

Our findings invite follow-up questions. One of them, inspired by real networks, is the problem of shortest path finding in geometric networks where nodes are distributed heterogeneously. While the distance to the geodesic can also be used in geometric networks with heterogeneous spatial node distributions, it is not clear if the accuracy of path-finding remains high.

Another open question outside the scope is the geometric properties of shortest paths in weighted geometric networks. While link weights can be functions of links distances, more elaborate models were proposed in the literature that decouple links weights from distances between connected nodes [95, 96, 97].

Finally, the important question is whether all shortest path nodes are equally difficult to find using distance to the geodesic. Our preliminary experiments indicate that shortest path nodes situated closer to path endpoints tend to have a smaller distance to the geodesic and, as a result, are easier to infer, Fig. 2.7. This observation is exciting since shortest path nodes closer to the path endpoints may be easier to manipulate in networks compared to nodes in the middle of the shortest path. Indeed, nodes in the

middle of a shortest path may be shared by other paths and can be involved in many vital processes. As a result, altering their connections or capacity may lead to cascades of changes throughout the network [98, 99, 100, 101].

Figure 2.7 suggests that shortest path nodes closer to path endpoints are easier to find with the distance to the geodesic. *Are nodes in the middle of a shortest path easier to find with network-based methods?* This can indeed be the case since (i) nodes in the middle of a shortest path tend to have degrees larger than those close to its endpoints and (ii) large degree nodes are often characterized by large betweenness centrality values [102, 103]. As a result, nodes in the middle of shortest paths can be identified by prioritizing large-degree nodes. Thus, a fusion approach combining elements of the distance to geodesic with network topology may ultimately be more accurate in finding shortest path nodes than one of the two approaches used separately.

3

INVERSE ALL SHORTEST PATH PROBLEM

The only real valuable thing is intuition.

Albert Einstein

Although resource management schemes and algorithms for networks are well established, we present two novel ideas, based on graph theory, that solve inverse all shortest path problem. Given a symmetric and non-negative demand matrix, the inverse all shortest path problem (IASPP) asks to find a weighted adjacency matrix of a graph such that all the elements in the corresponding shortest path weight matrix are not larger than those of the demand matrix. In contrast to many inverse shortest path problems that are NP-complete, we propose the Descending Order Recovery (DOR) that exactly solves a variant of IASPP, referred to as optimised IASPP. The network provided by DOR minimized the number of links and the sum of the link weights among all the graphs with the same shortest path weight matrix. Our second proposed algorithm, Omega-based Link Removal (OLR), solves the optimised IASPP by utilising the effective resistance from flow networks. The essence of our idea is the applications of properties of flow networks*, such as electrical power grids, to compute the needed resources in* path networks *subject to end-to-end demands, such as telecommunication networks where quality of service constraints specify the end-to-end demands.*

This chapter is based on a published paper. [8].

3.1. INTRODUCTION

The design, dimensioning or operation of networks is often constrained by end-to-end limits. For example, a telephone call requires that the voice packets travel through a telecommunication network with a designated maximum latency; the delay between a source and a destination is limited to about 150ms. However, real-time control of systems over the Internet may require a lower end-to-end delay. Thus, different services (voice, video, ftp, email, etc.) typically require a different end-to-end delay. Usually, a telecom operator can determine the demand matrix D containing the maximum tolerably end-to-end delay d_{ij} between node i and node j in the network. However, given the demand matrix D, a telecom operator is still confronted to dimension the network, both topology and link weights, so that transport along the "best" path between any pair (i, j) of nodes consumes less time than the maximum tolerable end-to-end delay d_{ij}. Here, we focus on finding a solution to the operator's problem, which we call "inverse all shortest path problem"(IASPP). Other applications of IASPP are the design and construction of transportation networks, where the goal entails creating a network that ensures commute times between stations are constrained by specific upper bounds. Similar challenges occur in wireless sensor and actuator networks [104], mobile communication radio access networks [105], etc. An exploration of practical applications is discussed in Section 3.6.

While extensive research has focused on finding the shortest paths in a given graph, limited attention is given to the inverse direction, i.e. obtaining or recovering a graph based on the shortest path weights between each node pair as IASPP. A related challenge, termed the inverse shortest path problem (ISPP), which has garnered attention in prior research [31, 32, 33, 34, 35, 36, 38], is reviewed in Section 3.2. ISPP asks for making a set of predetermined paths in the graph the shortest paths, after modification and/or ensuring the shortest path weights between specific node pairs are bounded by given demands. Applications of the ISPP occur in the design of networks[7, 31], modelling traffic[33] and seismic tomography[31, 32]. However, in many practical scenarios, the topology of the network is unknown, rendering existing ISPP approaches inapplicable. In contrast to ISPP, our IASPP only requires a demand matrix as input. Additionally, the approach we propose in Section 3.3 not only furnishes a graph that satisfies specified demands, but also stands as an effective technique of "network sparsification"[106] and helps to better understand the importance of different links within a network.

Before introducing the inverse all shortest path problem (IASPP) in Section 3.2, we explain the terminology. We consider a graph G that possesses a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L links. The graph G can be represented [6] by an $N \times N$ adjacency matrix A, with element $a_{ij} = 1$ if there is a link in G between node $i \in \mathcal{N}$ and node $j \in \mathcal{N}$, otherwise $a_{ij} = 0$. Each link $l \in \mathcal{L}$ has a weight w_l, which is a positive real number that specifies a property of the link, e.g. the resistance in an electrical graph or the delay when transmitting IP packets over that link. On the graph G, two different types of transport are possible that lead to either "path networks" or "flow networks". In a path network, the transport of items follows a single path $\mathscr{P}_{ij}$ between a node pair (i, j), whereas in a flow network, the transport from node i to node j propagates over all possible paths from node i to node j. Two typical examples are a communication network, where IP packets follow most of the time a single path $\mathscr{P}_{ij}$ from source i to destination j, and a

power grid, where electrical current flows over all possible paths.

The weight $w\left(\mathscr{P}_{ij}\right) = \sum_{l\in\mathscr{P}_{ij}} w_l$ of a path $\mathscr{P}_{ij}$ between a node pair (i,j) consists[4] of the sum of the weights over all links that belong to that path $\mathscr{P}_{ij}$. We will denote by $\mathscr{P}^*_{ij}$ the shortest path between a node pair (i,j). The shortest path $\mathscr{P}^*_{ij}$ minimizes the path weight over all paths $\mathscr{P}_{ij}$ and obeys $w\left(\mathscr{P}^*_{ij}\right) \leq w\left(\mathscr{P}_{ij}\right)$. In most real-world networks, there is only one shortest path $\mathscr{P}^*_{ij}$, but, in general, there can be many shortest paths between the same node pair (i,j), in particular in unweighted graphs, where each link has the same link weight[1], i.e. $w_{ij} = w$ for all elements of the $N \times N$ link weight matrix W. The weighted adjacency matrix is $\widetilde{A} = W \circ A$, where the Hadamard product $\circ$ means a direct elementwise multiplication, $\widetilde{a}_{ij} = w_{ij}a_{ij}$ and we use "tilde" notation for weighted graph matrices[2]. In our setting, $\widetilde{a}_{ij} = 0$ means that there is no link between node i and node j, because we exclude zero link weights, i.e. $w_{ij} > 0$, as in Dijkstra's shortest path algorithm [28, 29, 107] and in order to avoid the complication that a zero weight, i.e. $w_{ij} = 0$, would physically mean that node i and j are the same. The separation between link weights, represented by the link weight matrix W, and underlying graph G, represented by the adjacency matrix A, is obvious in unweighted graphs, where $W = wJ$ and $J = uu^T$ is the all-one matrix and u is the all-one vector. In the unweighted case, the graph is confining. In the other extreme, where link weights are highly variable and where the minimum link weight $w_{\min} > 0$ is orders of magnitude smaller than the maximum link weight $w_{\max}$, the underlying graph G is less confining than the link weight structure[3], which effectively thins out the graph. Indeed, mainly links with small link weights are relevant in a shortest path problem and large link weights may be ignored[4] from the onset, especially if link weights are assigned per link independently of the other links (see also [4, Chapter 16], [108, 109, 110]). In a shortest path setting, links with low link weights are generally more costly than links with high link weights.

Let v_k denote the potential or voltage of node k in the graph G. The effective resistance ω_{ij} between node i and node j equals the voltage difference $\omega_{ij} = \frac{v_i - v_j}{I_c}$ when a unit current $I_c = 1$ Ampere is injected in node i and leaves the network at node j. The $N \times N$ effective resistance matrix Ω with elements ω_{ij}, studied in e.g. [6, 47, 48, 49, 111] and [6, Chapt. 5], is briefly reviewed in Sec. 3.1.2. If the graph G is connected[5], then the effective resistance ω_{ij} as well as the path weight $w\left(\mathscr{P}_{ij}\right)$ is finite for any node pair (i,j) and a shortest path $\mathscr{P}^*_{ij}$ exists between each node pair (i,j). We define the $N \times N$ matrix S, that contains all shortest path weights with element $s_{ij} = w\left(\mathscr{P}^*_{ij}\right)$. If the weighted adjacency matrix $\widetilde{A}$ is known, then the matrix S is readily found via a shortest path algorithm, like Dijkstra's shortest path algorithm. Dijkstra's shortest path computation is

[1] The shortest path does not change if all weights are multiplied by a constant $\alpha > 0$.

[2] The flow network is characterized by the subscript F, i.e. $\widetilde{A}_F$ is the weighted adjacency matrix of a flow network, while $\widetilde{A}$ denotes the weighted adjacency matrix of a path network.

[3] The link weight structure refers to the entire ensemble $\{w_l\}_{l\in\mathscr{L}}$ of all link weights in the graph as one coherent set, possibly generated by a process that takes correlations of weights over links into account. The matrix W can then be considered as one particular realization of the link weight structural process.

[4] If their removal does not disconnect the graph.

[5] The weighted adjacency matrix $\widetilde{A}$ is called irreducible when the graph G is connected (see [4, p. 183]; [6, art. 167 on p. 235]). For a connected graph, the (weighted) Laplacian only has 1 zero eigenvalue and its rank is $N-1$.

very efficient and only requires $O(N \log N)$ elementary operations. Both the effective resistance matrix Ω and the shortest path weight matrix S are distance matrices[6].

In the sequel, we limit ourselves to connected, undirected, simple[7] graphs. Consequently, the $N \times N$ symmetric matrices A, W, $\widetilde{A}$, Ω and S are non-negative with zero diagonal elements.

The main contributions of this work are as follows:

1. We propose a novel problem named "Inverse all shortest path problem" (IASPP) and its variant "the optimised IASPP" (OIASPP). The IASPP asks for a weighted graph whose shortest path weight between each node pair satisfies a given demand.

2. We prove that OIASPP is not NP-complete.

3. We propose the Descending Order Recovery (DOR) algorithm that exactly solves OIASPP. The DOR graph minimizes the number of links and the sum of the link weights among all the graphs with the same shortest path weight matrix.

4. We demonstrate that DOR is also an effective network sparsification algorithm.

5. We propose the Omega-based Link Removal (OLR) algorithm, which solves OIASPP by utilising the effective resistance [6, Chapter 5]. OLR invokes properties of *flow* networks, such as electrical power grids, to compute the needed resources in *path* networks subject to end-to-end demands, such as telecommunication networks.

6. We discuss the applications of IASPP and evaluate the performance of DOR and OLR.

The paper is outlined as follows. In Section 3.1.1 and 3.1.2, we introduce notations to describe IASPP. We formally define IASPP and its variant OIASPP in Section 3.2 and review related problems from literature. In Section 3.3 and Section 3.4, we respectively propose two algorithms, DOR and OLR, to solve the optimised inverse all shortest path problem (OIASPP). Section 3.5 compares and evaluate the proposed algorithms by simulations. Section 3.6 introduces the potential applications of IASPP. Finally, we summarize our results in Section 3.7.

3.1.1. THE LAPLACIAN MATRIX Q

The $N \times 1$ degree vector $d = Au$ contains the degree d_i of each node i and the corresponding diagonal matrix $\Delta = \text{diag}(d)$ has the nodal degrees on its main diagonal. The eigenvalue decomposition of the $N \times N$ Laplacian $Q = \Delta - A$,

$$Q = Z \text{diag}(\mu) Z^T, \tag{3.1}$$

defines the set of N orthogonal $N \times 1$ eigenvectors z_i contained in columns of the $N \times N$ eigenvector matrix Z, and the set of N eigenvalues $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_N$. Due to double

[6] Any element h_{ij} of a distance matrix H is non-negative $h_{ij} \geq 0$, but $h_{ii} = 0$ and h_{ij} obeys the triangle inequality: $h_{ij} \leq h_{ik} + h_{kj}$.

[7] A simple graph has no multiple links between a same pair of nodes and also no self-loops, i.e. $a_{ii} = 0$ for each node $i \in \mathcal{N}$.

orthogonality of the eigenvector matrix Z (i.e. $ZZ^T = I$ and $Z^T Z = I$), where I is the $N \times N$ identity matrix, (3.1) can be transformed into a weighted sum of N outer vector products

$$Q = \sum_{i=1}^{N} \mu_i z_i z_i^T. \tag{3.2}$$

As of any real, symmetric matrix [6], the eigenvalues of Laplacian Q are real and non-negative because Q is a positive semidefinite matrix. From $Qu = 0$, we observe that $\mu_N = 0$ and $z_N = u$ and thus $\det Q = 0$. Consequently, the Laplacian Q is not invertible. However, the pseudoinverse[8] of the Laplacian [47]

$$Q^\dagger = \sum_{i=1}^{N-1} \frac{1}{\mu_i} z_i z_i^T \tag{3.3}$$

obeys $Q^\dagger Q = QQ^\dagger = I - \frac{1}{N} J$. In this work we consider a weighted graph G, where a link l between node i and node j is defined by its weight

$$w_{ij} = w_l = \frac{1}{r_l},$$

with r_l denoting link l resistance and $r_l > 0$, $w_l > 0$.

3.1.2. EFFECTIVE RESISTANCE

The effective resistance ω_{ij} between node i and node j is defined as [6]

$$\omega_{ij} = \left(e_i - e_j\right)^T Q^\dagger \left(e_i - e_j\right), \tag{3.4}$$

where the $N \times 1$ basic vector e_i has only one non-zero element $(e_i)_i = 1$. The effective resistance ω_{ij} quantifies the dissipated power when the current of 1 Ampere is applied between the nodes i and j. The equation in (3.4) can be transformed into a matrix form, defining the $N \times N$ effective resistance matrix

$$\Omega = \zeta u^T + u\zeta^T - 2Q^\dagger, \tag{3.5}$$

where the $N \times 1$ vector $\zeta = \left(Q^\dagger_{11}, Q^\dagger_{22}, \ldots, Q^\dagger_{NN}\right)$ contains the diagonal elements of the pseudoinverse of the Laplacian Q. The effective resistance ω_{ij} between directly connected nodes i and j (i.e. $a_{ij} = 1$), represents the effective resistance of a parallel connection

$$\frac{1}{\omega_{ij}} = \frac{1}{r_{ij}} + \frac{1}{(\omega_{G^*})_{ij}} \tag{3.6}$$

between the resistance of a direct link r_{ij} and the effective resistance $(\omega_{G^*})_{ij}$ between nodes i and j in the graph $G^* = G \setminus l$, where the link $l = i \sim j$ is removed.

[8] We restrict the analysis to connected simple graphs, as the number of zero eigenvalues of Laplacian Q equals the number of connected components in a graph. More precisely, (3.3) does not hold in the case of a disconnected graph.

Lemma 1. *A link $i \sim j \in \mathscr{L}$ of a graph $G(\mathscr{N}, \mathscr{L})$ connects two disconnected sub-graphs G_1 and G_2, i.e. $\mathscr{L}(G_1) \cup \mathscr{L}(G_2) \cup i \sim j = \mathscr{L}(G)$ and $\mathscr{L}(G_1) \cap \mathscr{L}(G_2) = \emptyset$ if and only if it holds*

$$\omega_{ij} = r_{ij}.$$

Proof. When link $i \sim j$ of a graph G connects two disconnected sub-graphs G_1 and G_2, the effective resistance of a graph $G^* = G \setminus i \sim j$ equals $r^*_{ij} = \infty$. Therefore, (3.6) transforms into $\omega_{ij} = r_{ij}$. □

The effective resistance ω_{ij} between adjacent nodes i and j is upper bounded by the resistance r_{ij} of the direct link between them

$$\omega_{ij} = \frac{r_{ij}\,(\omega_{G^*})_{ij}}{r_{ij} + (\omega_{G^*})_{ij}} \leq \min\left(r_{ij}, (\omega_{G^*})_{ij}\right).$$

Otherwise, if $a_{ij} = 0$, then the effective resistance ω_{ij} is upper bounded by the sum of resistances of links forming the shortest path between the nodes. In both cases, if more paths exist connecting two nodes, then there are more possible paths for the current to flow simultaneously and thus, the effective resistance lowers. The sum of all elements of the $N \times N$ effective resistance matrix Ω defines the effective graph resistance [6]

$$R_G = \frac{1}{2} u^T \Omega u = N \sum_{i=1}^{N-1} \frac{1}{\mu_i}. \tag{3.7}$$

3.2. INVERSE ALL SHORTEST PATH PROBLEM (IASPP)

3.2.1. STATEMENTS OF INVERSE ALL SHORTEST PATH PROBLEMS

Problem 1 (Inverse All Shortest Path Problem (IASPP))**.** *Given an $N \times N$ symmetric demand matrix D with zero diagonal elements but positive off-diagonal elements. Determine an $N \times N$ weighted adjacency matrix $\widetilde{A}$, such that the corresponding shortest path weight matrix S obeys*[9] $S \preccurlyeq D$

Since an element in the shortest path weight matrix S can be any positive number by scaling the weighted adjacency matrix, the IASPP generally has infinitely many solutions. Therefore, optimisation criteria such as the minimization of a norm $||D - S||$ are added. An instance [7] of IASPP is the optimised inverse shortest path problem (OIASPP).

Problem 2 (Optimised Inverse All Shortest Path Problem (OIASPP))**.** *Given an $N \times N$ symmetric demand matrix D with zero diagonal elements but positive off-diagonal elements. Determine an $N \times N$ weighted adjacency matrix $\widetilde{A}$, such that the corresponding shortest path weight matrix S obeys $S \preccurlyeq D$ and minimizes a norm $||D - S||$.*

Van Mieghem [7] demonstrated that any demand matrix D can be transformed into a distance matrix D' with $D' \preccurlyeq D$, where D' represents the (modified) demand matrix that is also a distance matrix: If $d_{ik} + d_{kj} < d_{ij}$ for at least one node $k \in \mathscr{N}$ which violates the

[9] The notation $\preccurlyeq$ is used for componentwise inequality, i.e. $S \preccurlyeq D$ means that $s_{ij} \leq d_{ij}$ for each $i = 1, 2, \ldots, N$ and each $j = 1, 2, \ldots, N$.

triangle inequality of a distance matrix, then we can replace $d_{ij} = \min_{1\le k\le N}(d_{ik} + d_{kj})$ and $d_{ji} = d_{ij}$. In the following, we assume that the demand matrix D is also a distance matrix. A complete graph whose weighted adjacency matrix $\widetilde{A} = D$ is a solution of the OIASPP with demand matrix D. Consequently, given a demand matrix D, we can obtain at least one solution of the OIASPP. In 1965, Hakimi and Yau [37] proved that if a weighted graph G is an N-node realization of an $N \times N$ distance matrix D', i.e. the corresponding shortest path weight matrix S of G equals D', and there does not exist three nodes i, j and k such that $w_{ij} > s_{ik} + s_{kj}$, where w_{ab} is the link weight between nodes a and b and s_{ab} denotes the shortest path weight, then G is unique. Hence, if there is only one solution of the OIASPP, then the resulting graph is a complete graph [7] and $w_{ij} \le s_{ik} + s_{kj}$ holds for arbitrary three nodes i, j and k, when the graph size $N \ge 3$. When the demand matrix D is a shortest path weight matrix generated by a tree, Van Mieghem [7] has solved OIASPP exactly as explained in Section 3.2.2.

In this paper, we focus on general underlying graphs rather than trees or complete graphs. We respectively propose two algorithms Descending Order Recovery (DOR) and Omega-based Link Removal (OLR) to solve OIASPP in Section 3.3 and Section 3.4. Since the computational complexity of DOR is polynomial, we have incidentally proved that OIASPP is not NP-complete.

3.2.2. LITERATURE REVIEW OF IASPP

Before investigating IASPP, we explain the related inverse shortest path problem (ISPP). Both ISPP and IASPP are "inverses" of the shortest path problem, that ask for a graph given the shortest paths or shortest path weights between node pairs. However, ISPP requires both the shortest paths (or shortest path weights) *and* the original graph, while IASPP only necessitates a demand matrix, that specifies the maximum shortest path weights, as input.

Problem 3 (Inverse Shortest Path Problem (ISPP))**.** *Given an $N \times N$ weighted adjacency matrix $\widetilde{A}$ with link weight matrix W and a set of paths $\{\mathscr{P}_{ij}\}$. Determine an $N \times N$ non-negative link weight matrix W' and the corresponding graph H such that all the paths $\mathscr{P}_{ij}$ belonging to $\{\mathscr{P}_{ij}\}$ are the shortest paths in the obtained graph H.*

We will introduce several representative generalizations or variants of ISPP below.

In 1992, Burton and Toint[31] proposed a quadratic programming algorithm based on the Goldfarb-Idnani method [112] to solve a variant of ISPP, which we denote by $\text{ISPP}_{\text{Burton}}$:

Problem 4 (Inverse Shortest Path Problem Burton ($\text{ISPP}_{\text{Burton}}$))**.** *Given an $N \times N$ weighted adjacency matrix $\widetilde{A}$ with link weight matrix W and a set of paths $\{\mathscr{P}_{ij}\}$. Determine an $N \times N$ non-negative link weight matrix W' and the corresponding graph H such that all the paths $\mathscr{P}_{ij}$ belonging to $\{\mathscr{P}_{ij}\}$ are the shortest path in the obtained graph H and minimize $||W' - W||$.*

Burton and Toint utilised l_2 norm $||W' - W|| = \sqrt{\sum_i \sum_j (w'_{ij} - w_{ij})^2}$, where w'_{ij} and w_{ij} represent the elements of W' and W respectively. A specialized Goldfarb-Idnani method can then be implied. The approach involves iterative adjustments to the matrix W', leading to the eventual weighted graph H, in which $\mathscr{P}_{ij}$ belonging to the given

path set $\{\mathscr{P}_{ij}\}$ are the shortest paths. The method works in both directed and undirected graphs.

Different variants and modified methods following $\text{ISPP}_{\text{Burton}}$ are discussed in [33, 34, 35, 36]. In 1999, Fekete et al.[38] considered a more general ISPP, where only the shortest path weight between pairs of nodes is given, but not the paths achieving them. Given a graph G with adjacency matrix A and a demand matrix D, $\text{ISPP}_{\text{Fekete}}$ aims to find a "weight function" of the weighted adjacency matrix $\widetilde{A}$ such that the demand matrix D is exactly the shortest path weight matrix S, where the weight function describes all the weighted adjacency matrices whose corresponding shortest path weight matrix $S = D$. The demand matrix D in $\text{ISPP}_{\text{Fekete}}$ must be a distance matrix measuring the shortest path weight between several pairs of nodes in graph G. *Not* all the pairs of nodes in graph G are necessarily included in the demand matrix D. Fekete et al. [38] proved that $\text{ISPP}_{\text{Fekete}}$ is NP-complete by reducing $\text{ISPP}_{\text{Fekete}}$ to a vertex-disjoint paths problem.

All mentioned variants of ISPP require the original weighted adjacency matrix $\widetilde{A}$ or adjacency matrix A. In contrast, Hakimi and Yau [37] investigated a "weighted graph realization" with only an $N \times N$ demand matrix D as input, which is also a distance matrix. Hakimi and Yau [37] presented an algorithm to obtain a graph H on N' nodes by adding $N' \geq N$ nodes into the graph such that the corresponding shortest path weight matrix $S = D$. If we extract the shortest path weights between node pairs that belonging to the first N nodes and form a shortest path weight matrix S, then $S = D$. Since the input in [37] contains all the shortest path weights in a graph, we call the problem "inverse all shortest path problem" (IASPP).

If the given distance matrix D can be realized by a tree t, Van Mieghem[7] proposed an elegant algorithm to recover the tree t from D by exploiting the analogy between flow networks and path networks. For undirected flow networks, Fiedler [113, 114] has presented the following block matrix relation,

$$\begin{pmatrix} 0 & u^T \\ u & \Omega \end{pmatrix}^{-1} = \begin{pmatrix} -2\sigma^2 & p^T \\ p & -\frac{1}{2}\widetilde{Q} \end{pmatrix} \tag{3.8}$$

with $\Omega p = 2\sigma^2 u$, where $\widetilde{Q} = \widetilde{\Delta}_F - \widetilde{A}_F$ is the weighted Laplacian matrix of a flow network and the diagonal matrix $\widetilde{\Delta}_F = \text{diag}(A_F u)$, $\widetilde{A}_F$ is the weighted adjacency matrix of a flow network, the variance $\sigma^2 = \frac{\zeta^T \widetilde{Q} \zeta}{4} + R_G$, where R_G is the effective graph resistance[47] and u is $N \times 1$ the all-one vector. The vector ζ contains the diagonal elements of pseudoinverse $Q^\dagger$ of the Laplacian $\widetilde{Q}$. Specifically, Van Mieghem[7] defined the weight of a link $w_{ij} = r_{ij}$ as the resistance (in Ohm), then the weighted Laplacian $\widetilde{Q}$ has non-zero elements $\widetilde{q}_{ij} = -\frac{1}{r_{ij}}$ and $\widetilde{a}_F = \frac{1}{r_{ij}}$ for $i \neq j$, where $\widetilde{a}_{ij} = r_{ij}$, but $(\widetilde{a}_F)_{ij} = (\widetilde{a})_{ij} = 0$ for $i \neq j$ if there is no link between node i and node j. The diagonal elements $(\widetilde{a}_F)_{ii} = (\widetilde{a})_{ii} = 0$ are always zero. Fiedler's block matrix relation (3.8) indicates a one-to-one relation between the effective resistance matrix Ω and the weighted Laplacian $\widetilde{Q}$ and therefore, also between the effective resistance matrix Ω and the weighted adjacency matrices $\widetilde{A}_F$ and $\widetilde{A}$. By applying block inverse formulae[6] to Fiedler's block matrix relation, it is shown in [7] that

$$2\sigma^2 = \frac{1}{u^T \Omega^{-1} u} \tag{3.9}$$

$$p = \frac{1}{u^T \Omega^{-1} u} \Omega^{-1} u \tag{3.10}$$

and the inverse of the effective resistance matrix is

$$\Omega^{-1} = \frac{1}{2\sigma^2} p p^T - \frac{1}{2} \widetilde{Q} \tag{3.11}$$

Hence, with $\widetilde{Q} = \widetilde{\Delta}_F - \widetilde{A}_F$, the weighted adjacency matrix follows as

$$\widetilde{A}_F = \widetilde{\Delta}_F + 2\Omega^{-1} - \frac{1}{\sigma^2} p p^T. \tag{3.12}$$

If the graph G is a tree, then the shortest path matrix S equals the effective resistance matrix Ω, because there exists exactly one path in a tree between each pair of nodes[48]. The weighted adjacency matrix $\widetilde{A}_F$ can be deduced from (3.12) by replacing Ω by S. Hence, the weighted adjacency matrix $\widetilde{A}$ follows by taking the element-wise inverse of $\widetilde{A}_F$. The zero elements in $\widetilde{A}_F$ should not be inverted, but should instead be transferred to $\widetilde{A}$. Indeed, the obtained $\widetilde{A}$ is an exact solution of the OISPP for any tree: If the given demand matrix D is a distance matrix such as the shortest path weight matrix S, then the weighted adjacency matrix $\widetilde{A}$ can be obtained from (3.12) with $\Omega = S$. We call this method the "flow analogue method".

As explained in Appendix B.2, the algebraic flow analogue method is hard to extend from a tree graph to a general graph. In the sequel, we solve OIASPP for general graphs.

3.3. DESCENDING ORDER RECOVERY ALGORITHM (DOR)

In this section, we propose the Descending Order Recovery algorithm (DOR) that solves OIASPP exactly. If a demand matrix D is the shortest path weight matrix S of an arbitrary graph G, then DOR retrieves the graph H satisfying the norm $||D - S'|| = 0$, where S' is the shortest path weight matrix of H. For a given demand matrix D, the graph H obtained by DOR is unique and reaches a minimum number of links and a minimum sum of the link weights among all OIASPP solutions with the same demand matrix D. The resulting graph H generally has less links than graph G. If graph G is unweighted, the resulting graph H is the same graph as G.

3.3.1. PROPERTIES OF DOR

Our main idea of DOR is:

1. Given a demand matrix D, find the minimum spanning tree of the complete graph G_D, whose weighted adjacency matrix $\tilde{A} = D$;
2. Add a link between two nodes i and j whose shortest path weights $s_{ij} > d_{ij}$, with link weight $w_{ij} = d_{ij}$;
3. Repeat 2 until $s_{ij} \leq d_{ij}$ for each $i, j \in \mathcal{N}$.

If we remove a link l in graph G and obtain graph H, then the link l either (a) belongs or (b) does not belong to the shortest path $\mathscr{P}^*_{ij}$ between two nodes i and j. In case (a), removing link l does not change the shortest path $\mathscr{P}^*_{ij}$ nor the shortest path weight s_{ij}.

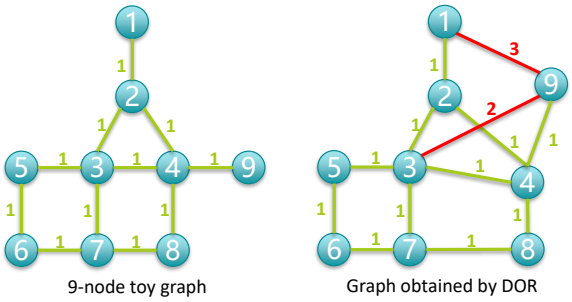


Figure 3.1: Visualization of redundant links in the graph obtained by DOR. We generate a 9-node toy graph and obtain the corresponding shortest path weight matrix as the demand matrix D. A graph is then obtained by DOR with the demand matrix D as input. The redundant links are highlighted.

In case (b), the shortest path between nodes i and j is changed, but the shortest path weight s'_{ij} in H cannot be smaller than s_{ij}, otherwise, the shortest path weight s'_{ij} would also be the shortest path weight in G. Thus, the shortest path weight $s_T(ij)$ between arbitrary nodes i and j in the minimum spanning tree T of a graph G is not smaller than the shortest path weight $s_G(ij)$ in the graph G and step 1 ensures that the lower bound of the shortest path weight matrix of our obtained graph H is D. The upper bound of the shortest path weight matrix of the obtained graph is also D after performing step 2 and 3. DOR obtains a graph H satisfying the norm $||D - S'|| = 0$. We present the pseudo code of DOR initialised with a minimum spanning tree as Algorithm 7 in Appendix B.6.

The graph H obtained by DOR may have more links than the original graph G. In the worst case, the resulting H is a complete graph, whose weighted adjacency matrix equals the demand matrix D. We call $l = i \sim j$ a "redundant" link if we can find another node k, besides i and j, in the graph such that $\widetilde{a}_{ij} = w_{ij}a_{ij} \geq s_{ik} + s_{kj}$. For example, as shown in Fig. 3.1, link l_{19} can be replaced by l_{12}, l_{24} and l_{49} when calculating the shortest path between nodes 1 and 9 in the graph obtained with DOR, since $\widetilde{a}_{19} = s_{14} + s_{49}$. Removing link l_{19} would not change the shortest path weight matrix S nor the connectivity of the original graph.

Algorithm 1 Descending Order Recovery (DOR)

Input: $N \times N$ demand matrix $D = S$: a shortest path weight matrix of a graph G
Output: $N \times N$ weighted adjacency matrix $\widetilde{A}$

```
1: Ã ← D and Ã specifies graph G
2: ∀ positive link weights in G and any node k ≠ i, j
3:     if ã_ij ≥ s_ik + s_kj then
4:         ã_ij ← 0, ã_ji ← 0
5:     end if
6: return Ã
```

If a link $l = i \sim j$ is redundant in a weighted adjacency matrix $\widetilde{A}$ obtained by DOR, i.e. if there exists a node k such that $s_{ik} + s_{kj} \leq \widetilde{a}_{ij}$, we then remove the link between nodes

i and nodes j and let $a_{ij} = 0$. Hakimi and Yau [37] proved that there is only one graph which does not have redundant links among all the graphs with the same shortest path weight matrix S. Therefore, the graph H obtained by DOR is unique for a given demand matrix D after removing all redundant links. Hence, DOR can be further simplified: After removing all redundant links in the complete graph G_D whose weighted adjacency matrix equals the demand matrix D, we obtain the solution graph H, which solves OIASPP exactly. The pseudo code for simplified DOR is shown in Algorithm 1.

Property 1. *Given a demand matrix D, the obtained graph H by DOR reaches a minimum number of links among all the OIASPP solutions.*

Proof. By contradiction: Suppose that there exists a graph H' such that the corresponding shortest path weight matrix $S = D$ and the graph H' has fewer links than the graph H obtained by DOR. The graph H' should have redundant links because both graph H' and graph H have the same shortest path weight matrix S and the graph H does not have redundant links. In that case, we construct a graph H'' by removing redundant links in graph H'. Since removing redundant links does not change the corresponding shortest path weight matrix, graph H'' and graph H have the same shortest path weight matrix S and do not have redundant links, which is impossible because there is only one graph that does not have redundant links among all the graphs with the same shortest path weight matrix S. Hence, the obtained graph H by DOR reaches a minimum number of links among all the solutions to an OIASPP given a demand matrix D. □

Because the graph H obtained by DOR minimizes the number of links among all the graphs with the same shortest path weight matrix S, we can only obtain a graph H' that has the same shortest path weight matrix S by adding redundant links. We thus have:

Property 2. *Given a demand matrix D, the obtained graph H by DOR reaches a minimum sum of the link weights among all OIASPP solutions.*

Given a demand matrix D that is a shortest path weight matrix S of an arbitrary "original" graph G. While the shortest path weight matrix S of the graph H obtained by DOR is identical to the shortest path weight matrix of the original graph G, the two graphs H and G themselves may not be the same. Specifically, when the demand matrix D is computed from an unweighted graph G, fortunately, we can remove all the redundant links by removing links whose weights are larger than 1 in the complete graph G_D. Since all the link weights in unweighted graphs are exactly 1, the shortest path weight s_{ij} between two nodes equals 1 if and only if nodes i and j are neighbours. Thus the adjacency matrix A of the graph after removing redundant links in the complete graph G_D is precisely the same as the adjacency matrix of the original unweighted graph G.

3.3.2. EXAMPLES

In Fig. 3.2, we respectively examine the number of links L_G and L_H of the original graph G and the DOR graph H. For each simulation, we generate an Erdős–Rényi (ER) random graph $G_p(N)$, where N is the number of nodes and p is the probability of connecting two nodes. The link weights of the ER graph $G_p(N)$ are uniformly distributed in $(0,1)$. The $N \times N$ shortest path weight matrix S is calculated and equal to the demand matrix D. For different N and p, 1000 iterations are carried out.

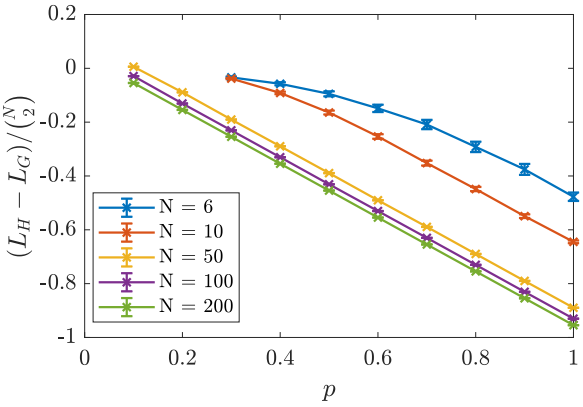


Figure 3.2: The differences of number of links between the original graph and the graph obtained by DOR.

Fig. 3.2 illustrates that DOR produces graphs H with fewer links than the original ER graphs G, provided the link density p is sufficiently large. An interesting phenomenon is that the resulting graph H seems to have a similar number of links, irrespective of the number L_G of links in the original graph. Hence, for a dense original graph G, DOR provides a sparser graph with the same shortest path weight matrix, but with a different adjacency matrix A, which can be regarded as "network sparsification" [106] that preserves all shortest path weights.

An instance of network sparsification is investigated by Simas, et al. [54]. Given a graph G with weighted adjacency matrix $\widetilde{A}$, Simas, et al. [54] focuses on obtaining a graph H, which they call the "distances backbone", with the same shortest path weight matrix S of graph G, but fewer links. The main idea is that the off-diagonal elements of the resulting weighted adjacency matrix $\widetilde{A}'$ are computed by

$$\begin{cases} \widetilde{a}'_{ij} = s_{ij} & \text{if } \widetilde{a}_{ij} = s_{ij} \\ \widetilde{a}'_{ij} = 0 & \text{if } \widetilde{a}_{ij} > s_{ij} \end{cases} \tag{3.13}$$

for $i = 1,2,\ldots,N$, $j = 1,2,\ldots,N$, $j \neq i$. However, the method proposed by Simas et al. [54] always includes the redundant links such that $w_{ij} = s_{ik} + s_{kj}$, where w_{ij} is the weight of link $l = i \sim j$. Thus, DOR can return a sparser graph than the distances backbone.

Van Mieghem and Wang [110] investigated the union of all shortest path trees $G_{\cup_{spt}}$, where the shortest path tree (SPT) rooted at some node is the union of the shortest paths from that node to all the other nodes. If a link $l = i \sim j$ is the shortest path $\mathscr{P}^*_{ij}$ between i and j, then $l = i \sim j$ must belong to the $G_{\cup_{spt}}$, because the $G_{\cup_{spt}}$ is the union of shortest paths between all possible source and destination nodes[110]. All the links in the graph H obtained by DOR belong to at least one shortest path $\mathscr{P}^*_{ij}$ and the graph H thus belongs to the $G_{\cup_{spt}}$. The inverse does not hold, because the union $G_{\cup_{spt}}$ may have redundant links $l = i \sim j$ in which $w_{ij} = s_{ik} + s_{kj}$.

3.3.3. COMPUTATIONAL COMPLEXITY OF DOR

For each possible link $l = i \sim j$, DOR determines whether the link is redundant by comparing the link weight w_{ij} with the sum of the shortest path weights $s_{ik} + s_{kj}$, where $k \in \mathcal{N}$ is a node different from node i and j. Hence, each link $l = i \sim j$ needs to be compared with the sum of the shortest path weights $s_{ik} + s_{kj}$ for $N-2$ nodes k in the worst case. The computational complexity of the worst case of DOR (Algorithm 1) is $O(N^3)$, because the demand matrix $D = O(N^2)$. OIASPP is thus not NP-complete! The main differences between OIASPP and three NP-complete variants of ISPP introduced in Section 3.2 and Appendix B.5 lie in the given constraints. While the three NP-complete variants of ISPP restrict the resulting graph to a predetermined graph topology, OIASPP can be solved by changing both topology and link weights to meet the given constraints about shortest path weights.

3.4. OMEGA-BASED LINK REMOVAL ALGORITHM (OLR)

The Omega-based Link Removal (OLR) algorithm recovers an as sparse as possible graph, with elements of the shortest path weight matrix $s_{ij} \in [bd_{ij}, d_{ij}]$, where d_{ij} is the given demand and $b \in [0,1]$ is an input parameter. OLR leverages information captured by the effective resistance between pairs of nodes. Equation (3.6) enables us to determine the impact on the effective resistance between two neighbouring nodes when the shared link between them, denoted as $l = i \sim j$, is eliminated. By targeting the removal of the link with the highest value of $\frac{1}{(\omega_{G^*})_{ij}}$, we achieve the smallest possible increase in the effective graph resistance R_G of the network. To enhance the efficacy of this approach for solving OIASPP, we introduce a refinement, which involves scaling the quantity $\frac{1}{(\omega_{G^*})_{ij}}$ by the difference between the provided upper bound d_{ij} and the current shortest path weight s_{ij} for the pair of nodes (i, j). This strategic adjustment allows us to combine insights from the effective resistance measurements and the upper bound values supplied by the $N \times N$ demand matrix D.

The shortest path weight between two nodes is the sum of the link weights (i.e. corresponding elements of the weighted adjacency matrix $\tilde{A}$) belonging to that path. On the contrary, a link weight in a "flow network "(defined by the adjacency matrix $\tilde{A}_F$) has a dimension of the inverse of the resistance. Therefore, to utilise the analogy between shortest paths and effective resistance, we additionally define the $N \times N$ link weight matrix $\hat{W}$ containing the inverse link weights[10]

$$\hat{w}_{ij} = \begin{cases} \frac{1}{\tilde{a}_{ij}} & \text{if } \tilde{a}_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.14}$$

where $i, j \in \mathcal{N}$. The corresponding $N \times N$ effective resistance matrix computed with $\hat{W}$ instead of $\tilde{A} = A \circ W$ is denoted as $\hat{\Omega}$.

In Algorithm 2, we propose an iterative algorithm that solves the IASPP problem by invoking the effective resistance between pairs of nodes. The OLR algorithm is initialised in line 1 by the complete graph with the adjacency matrix $A = J - I$, while the link

[10] Link existence overrules the link weight. Equation (3.14) shows that if a link $l = i \sim j$ does not exist in graph G (i.e. $\tilde{a}_{ij} = 0$), than $\hat{w}_{ij} = 0$, although $\frac{1}{\tilde{a}_{ij}} \to \infty$.

Algorithm 2 Omega-based Link Removal (OLR)

Input: $N \times N$ demand matrix $D = S$: a shortest path weight matrix of a graph G; input parameter $b \in [0,1]$
Output: $N \times N$ weighted adjacency matrix $\widetilde{A}$
1: $A_{N\times N} \leftarrow J_{N\times N} - I_{N\times N}$ adjacency matrix of a complete graph
2: $\widetilde{A} \leftarrow b \cdot (A \circ D)$ weighted adjacency matrix
3: $S_{N\times N} \leftarrow$ Shortest path weight matrix of $\widetilde{A}$
4: $\hat{W}_{N\times N} \leftarrow$ Inverse link weight matrix of $\widetilde{A}$
5: **do**
6: $\quad \hat{\Omega}_{N\times N} \leftarrow$ Effective resistance matrix of $\hat{W}$
7: $\quad R \leftarrow \left(\hat{\Omega} - \hat{W}\right) \circ (D - S) \circ A$
8: $\quad (i, j) \leftarrow$ Indices of the maximum element in R
9: $\quad A \leftarrow A - e_i \cdot e_j^T - e_j \cdot e_i^T$
10: $\quad \widetilde{A} \leftarrow b \cdot (A \circ D)$
11: $\quad S_{N\times N} \leftarrow$ Shortest path weight matrix of $\widetilde{A}$
12: $\quad \hat{W}_{N\times N} \leftarrow$ Inverse link weight matrix of $\widetilde{A}$
13: **while** $(S \preccurlyeq D) \wedge (R_{ij} > 0)$
14: $A \leftarrow A + e_i \cdot e_j^T + e_j \cdot e_i^T$
15: $\widetilde{A} \leftarrow b \cdot (A \circ D)$
16: **return** $\widetilde{A}$

weights equal (line 2) the corresponding shortest path weights in the demand matrix $D = S$, scaled by the input parameter b,

$$\widetilde{A} = b \cdot (A \circ D),$$

which ranges between 0 and 1. Link weights are scaled in line 2 for two reasons. Assume the demand matrix D is derived from an original graph. In case $b = 1$, if the proposed OLR algorithm recovers the exact topology as in the original graph G, then the link weights would also be the same. In general, OLR ensures the shortest path weight between directly connected nodes to be equal to the corresponding element of the provided upper bound in D, scaled by the input parameter b. Therefore, $b < 1$ allows OLR to achieve sparser graphs even from the original graph G, at the cost of increased norm [11] of $||D - S||$, still satisfying the bound $S \preccurlyeq D$. To determine which link should be removed in each iteration, in line 7 we compute the $N \times N$ matrix

$$R = \left(\hat{\Omega} - \hat{W}\right) \circ (D - S) \circ A,$$

where the $N \times N$ inverse link weight matrix $\hat{W}$ contains inverse link weights, as defined in (3.14). whose elements are dimensionless and denote the inverse effective resistance $\left(\hat{\Omega} - \hat{W}\right)_{ij}$ between a pair of neighbouring nodes (i.e. $a_{ij} = 1$), in case the direct link between them is removed (as in (3.6)), multiplied by the gap $(d_{ij} - s_{ij})$ between the shortest path weight between them and the given upper bound in D. We remove the existing

[11] For any pair of connected nodes i and j we observe $s_{ij} = b \cdot d_{ij}$. In addition, for non-adjacent nodes m and n we reason $s_{mn} > b \cdot d_{mn}$, because D is a distance matrix. Combining these two observations, we conclude $S \leq b \cdot D$, which yields $||D - S|| < 1 - b$.

link with the highest value in R (line 8), because the adjacent nodes are easily reachable via the rest of the graph when the link is removed, and the margin between the current shortest path weight and the upper bound is relatively high. After updating the adjacency matrix A (line 9), we redistribute the link weights (line 10) as $\widetilde{A} = b \cdot (A \circ D)$ and update (line 11) the $N \times N$ shortest path weight matrix S.

Link removal is performed until at least one shortest path weight in the obtained graph H exceeds the given upper bound in the $N \times N$ demand matrix D. At that point, the last removed link is returned (line 14), while the $N \times N$ weighted adjacency matrix $\widetilde{A}$ is provided as output.

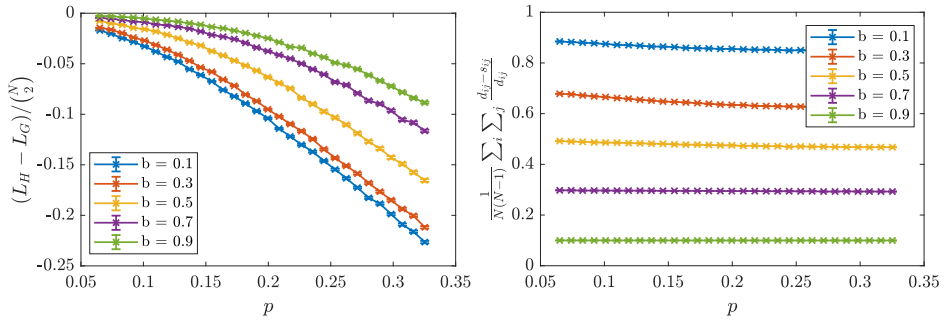
OLR initialises the topology with a complete graph and iteratively removes links until at least one upper bound on the shortest path weight between node pairs is exceeded. In general, OLR can return any connected topology, even a tree. Therefore, there are generally up to $\frac{N\cdot(N-1)}{2} - (N-1)$ iterations. The effective resistance and the shortest path weight between all node pairs are computed within each iteration. Within each iteration in our OLR, the effective resistance and the shortest path weight between any pair of nodes are computed. Both operations require computational complexity $O(N^3)$. In addition, we initialise OLR with a complete graph. The number of iterations in worst case scales as $O(N^2)$. Therefore,the overall complexity of our OLR is $O(N^5)$. Alternatively, DOR can streamline OLR's computational complexity. DOR ensures the retrieval of a graph with the minimum necessary links, accurately aligning the shortest path weight matrix S with the demand matrix D. Instead of initializing OLR with a complete graph, we employ DOR as the initial phase within OLR. Subsequently, we iteratively refine the graph until the shortest path weights fall within a predefined range, as dictated by the input parameter b. Consequently, the number of removed links within OLR reduces significantly, lowering its computational complexity to be $O(N^3 L')$, where L' is the number of links in graph obtained by DOR.

Fig. 3.3 shows the differences between the number of links in the OLR graph H and the original graph G with different b and the norm $||D-S|| = \frac{1}{N(N-1)} \sum_i \sum_j \frac{d_{ij}-s_{ij}}{d_{ij}}$ of the graph obtained by OLR with different input parameter b. For each simulation, we generate a 20-node Erdős–Rényi (ER) random graphs $G_p(20)$ and compute the corresponding shortest path weight matrix as the input demand matrix D, where p is the probability of connecting two nodes (link density). The link weights of the ER graph $G_p(20)$ are uniformly distributed in $(0,1)$. For each link density p, 1000 realizations are carried out. Fig. 3.3a illustrates that a smaller b generates a graph H with fewer links, while Fig. 3.3b shows that a smaller b corresponds to a large norm $||D-S||$.

3.5. PERFORMANCE EVALUATION OF DOR AND OLR

In this section, we evaluate the performance of DOR and OLR [12]. in random graphs and an empirical network. The performance of the DOR and OLR is assessed by three complementary criteria: (i) the number $L_H - L_G$ of additional links in the resulting graph H, (ii) the number $\frac{1}{2L_H} \cdot u^T \cdot (A \circ A_H) \cdot u$ of common links in the original graph G and the resulting graph H and (iii) the norm $||D-S|| = \frac{1}{N(N-1)} \sum_i \sum_j \frac{d_{ij}-s_{ij}}{d_{ij}}$ of the demand matrix

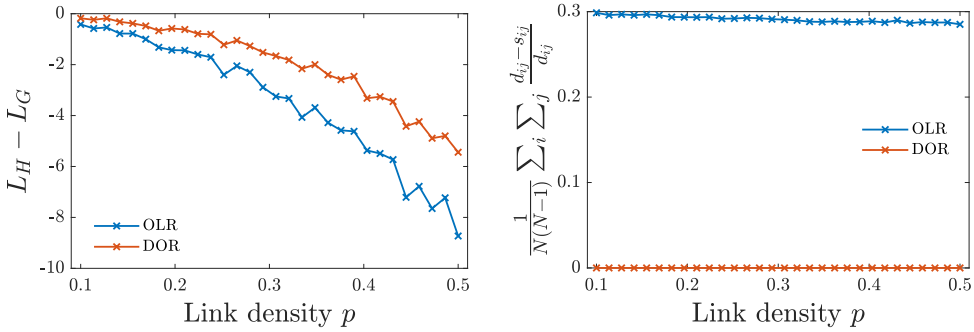
[12] Matlab code is on https://github.com/qzhszl/IASPP.git



(a) Number of additional links in obtained graph H. **(b)** Number of common links in G and H.

Figure 3.3: (a) Differences between number of links in the graph obtained by OLR and the original graph with different input parameter b. The x-axis denotes the link density p of the underlying 20-node ER graphs, while the y-axis is the difference between number of links in the graph H obtained by OLR and in the original graph G. (b) The norm $||D-S||$ of the graph obtained by OLR with different input parameter b. The x-axis denotes the link density p of the underlying 20-node ER graphs, while the y-axis is the norm $||D-S||$.

D and the shortest path weight matrix S.



(a) Number of additional links in obtained graph H. **(b)** Norm $||D-S||$.

Figure 3.4: Performance of the DOR and OLR on ER graphs with $N = 10$ nodes and different link density p. The input parameter $b = 0.7$.

Fig. 3.4 illustrates the results of DOR (red line) and OLR (blue line) in ER graphs $G_p(N)$ with $N = 10$ nodes and different link density p. We uniformly assign a random weight from $(0,1)$ to each link in G, thus defining the weighted adjacency matrix $\widetilde{A}$. For each generated ER graph, we provide the shortest path weight matrix of G as the input demand matrix D to the algorithm DOR and OLR. The input parameter of OLR $b = 0.7$. We then obtain the resulting graph H, whose shortest path weight matrix is denoted as S. For each number N of nodes and different link density p, 100 simulation instances are executed and the average over 100 times of each criterion is computed.

Fig. 3.4a depicts the difference in the number of links $L_H - L_G$ between the obtained

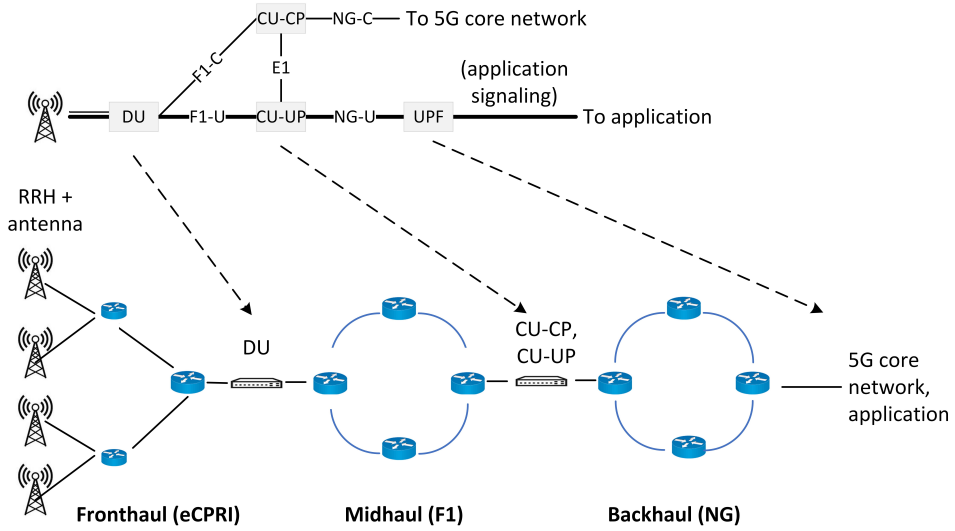


Figure 3.5: A conceptual diagram of a RAN as found in the 5G mobile communications network.

graph H and the original graph G. For a small link density p, the obtained graph H contains almost the same number of links L_H as that of the original graph L_G, while $L_H - L_G$ decreases with the increment of link density p. As for the number of common links in the original graph G and the resulting graph H, our simulation (details are shown in Appendix B.7) shows that $\frac{1}{2L_H} \cdot u^T \cdot (A \circ A_H) \cdot u = 1$ holds for both DOR and OLR with different link density p, which informs us that links of graph H obtained by both DOR and OLR belong to the original graph G. Fig. 3.4b illustrates the norm $||D - S||$, where DOR always returns an exact solution $||D - S|| = 0$ to the OIASPP. In contrast, for OLR, the norm $||D - S||$ is not zero but bounded by $1 - b$.

A similar pattern in performance is visible for a different number of nodes N, as presented in Fig. B.2 for the case $N = 20$ and $N = 50$ in Appendix B.7. The feasibilities of DOR and OLR are also verified in Barabási–Albert (BA) networks[115] with 500, 1000 and 10000 nodes, Watts–Strogatz (WS) small world network[116] with 100, 1000 and 10000 nodes and an empirical network USAir[117]. The details are shown in Appendix B.7.

In summary, the performance of DOR and OLR are stable with arbitrary demands on both small-size and large-size networks. Specifically, our simulation results verify that DOR provides a sparse graph that solves OIASPP exactly, while OLR exhibits a capacity to obtain a graph with fewer links compared with the DOR algorithm, at the cost of increased norm of $||D - S||$. The norm for DOR is always $||D - S|| = 0$, while for OLR $||D - S|| < 1 - b$, where $b \in [0, 1]$ is the input parameter.

3.6. APPLICATION

In this section, we discuss various IASPP applications and present a simulation example to validate the feasibility of our proposed DOR and OLR algorithms.

3.6.1. APPLICATION OF IASPP

The IASPP methodology is useful in Wireless Sensor and Actuator Network (WSAN) [104]. Industrial WSAN (IWSAN) standards such as WirelessHART [118] have gained popularity in process automation, e.g., gas production, electric power generation and smelting plants. An IWSAN consists of a gateway, multiple access points and hundreds of thousands of field devices (i.e., sensors and actuators) that operate at low-power, forming a multi-hop wireless network, where the link weight w_{ij} between node i and node j denotes the latency bound that a link $l = i \sim j$ should provide. In a WSAN network, IASPP considers the end-to-end (E2E) latency as a demand matrix. The WSAN gateway collects network topology and flow demand information [119]. If there is topological change (e.g., node failure, new joining nodes) or change of the traffic pattern that makes current link weight configuration inappropriate[13], then the WSAN gateway can use DOR or OLR to (re-)computes the weighted adjacency matrix $\widetilde{A}$. The updated link weights will then be communicated with devices in the network. With the set of newly computed shortest paths, E2E latency of an arbitrary pair of nodes is guaranteed. A further step is to consider scheduling, power consumption and path redundancy into the problem.

Mobile communication radio access network [105] (RAN) is another application domain. Fig. 3.5 provides a conceptual diagram of a RAN as found in the 5G mobile communications network. The lower part of Fig. 3.5 depicts that the communication between the logical components [105] of the RAN (DU, CU-CP etc.) is formed by IP infrastructure[120]. Data transmission latency between the RAN logical components is bound by demands, i.e. maximum permissible E2E latency. With predetermined E2E latency demands, DOR and OLR can provide guidance in constructing a RAN network, such as installing base stations at different locations of a city.

Transportation networks constitute another potential application domain. For example, urban planners and customers may have demands on the commute time for each pair of bus or train stations. DOR can offer a transportation network such that the commute time between every two nodes (which denotes stations) exactly equals the prescribed demand and reaches a minimum number of links of all the networks with the same shortest path weight matrix. OLR can deal with more specific scenarios. Imagine urban planners have defined maximum allowable travel times as the demands for specific node pairs, accounting for variables like passenger density along these routes. The link weights represent the time needed when travelling between adjacent nodes. These maximum travel time constraints can span from 100% to about 200% of the calculated minimum travel time. OLR can shape the network into an ideal structure, while choosing a relatively small input parameter b value. This strategy seeks to mould the network's topology in a manner that caters to all essential routes while conforming to the stipulated upper travel time limits. By applying the OLR algorithm, we can intelligently eliminate links, while preserving the network's overall connectivity and functionality. This process facilitates the creation of an optimised railway system that ensures both efficiency and adherence to travel time constraints. In the resultant graph generated by OLR, each link signifies the potential introduction of a direct line, further enhancing the network's efficiency and structure.

[13] *Inappropriate* in this context means latency bound violation.

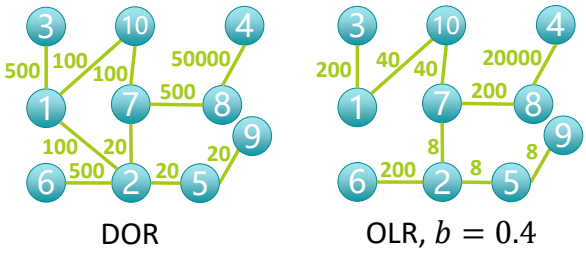


Figure 3.6: Visualization of the graph H obtained by DOR and OLR, respectively. The input E2E demand matrix is (3.16).

3.6.2. SIMULATION ON E2E LATENCY

In this section, we apply our IASPP methods to an E2E latency instance. Since the IASPP methods begin with a demand matrix which is also a distance matrix, the given demand matrix D is required to be modified so that we can imply our algorithm. If the E2E constraint of a node pair (i, j) is not specified, we assume that there is no constraint and that $d_{ij} = \infty$, which means there is no upper bound for the shortest path weight s_{ij} between node i and j. In many practical scenarios, not every pair of nodes necessarily has a demand. We first symmetrize the demand matrix (see explained in Section 3.2.1) following line 2 – 4 of Algorithm 3. We then focus on the triangle inequality of a demand matrix. Consider the following example of a demand matrix:

$$D = \begin{bmatrix} 0 & 1 & \infty & \infty & 1 \\ 1 & 0 & 1 & 1 & \infty \\ \infty & 1 & 0 & 3 & \infty \\ \infty & 1 & 3 & 0 & 5 \\ 1 & \infty & \infty & 5 & 0 \end{bmatrix} \tag{3.15}$$

The demand $d_{34} > d_{32} + d_{24}$ is a typical case of the violation of the triangle inequality. However, the infinite $d_{ij} = \infty$ may lead to complicated cases. Aside from demands that violate the triangle inequality (e.g. d_{34}), there could be other demands that are unattainable. For instance, d_{45} does not breach the triangle inequality as $d_{45} < d_{14} + d_{15}$, $d_{45} < d_{24} + d_{25}$ and $d_{45} < d_{34} + d_{35}$. Nevertheless, d_{45}, d_{42}, d_{21} and d_{15} form a cycle structure and $d_{45} > d_{42} + d_{21} + d_{15}$. Consequently, s_{45} must be smaller than d_{45}, that is, d_{45} is not achievable.

To ensure all the demands are possible to achieve, we modify the demand matrix according to line 5 – 13 of Algorithm 3. Our main idea is to calculate the shortest path weight matrix S of a graph whose weighted adjacency matrix equals the given demand matrix D, because the resulting demand matrix $D' = S$ reserves all the constraints in D except for the E2E demands not achievable. We can now transform an arbitrary non-negative demand matrix to a distance matrix and apply our DOR and OLR algorithms with the demand matrix D' as input.

Algorithm 3 Demand Modification

Input: Demand matrix D whose unspecified demands are represented by ∞
Output: Modified demand matrix D'

```
D' ← D
while symmetry of D' is violated, i.e. d_ij ≠ d_ji do
    d'_ij ← min(d_ij, d_ji), d'_ji ← min(d_ij, d_ji)
end while
while d'_ij = ∞ do
    d'_ij ← 0
end while
G_D' ← Graph whose weighted adjacency matrix equals D'
S ← Shortest path weight matrix of G_D'
while s_ij = ∞ do
    s_ij ← Maximum finite element in S
end while
D' ← S
return D'
```

We present an example in Fig. 3.6. Consider an E2E demand matrix:

$$D = \begin{bmatrix} 0 & 100 & 500 & \infty & \infty & 5000 & \infty & \infty & \infty & 100 \\ 100 & 0 & \infty & \infty & 20 & 500 & 20 & \infty & \infty & \infty \\ 500 & \infty & 0 & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty & 50000 & \infty & \infty \\ \infty & 20 & \infty & \infty & 0 & 1000 & \infty & \infty & 20 & \infty \\ 5000 & 500 & \infty & \infty & 1000 & 0 & \infty & \infty & \infty & \infty \\ \infty & 20 & \infty & \infty & \infty & \infty & 0 & 500 & 100000 & 100 \\ \infty & \infty & \infty & 50000 & \infty & \infty & 500 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty & 20 & \infty & 100000 & \infty & 0 & \infty \\ 100 & \infty & \infty & \infty & \infty & \infty & 100 & \infty & \infty & 0 \end{bmatrix} \tag{3.16}$$

We first transform the given demand matrix D to D' following the method introduced by Algorithm 3. Our DOR and OLR algorithms were applied to the demand matrix D'. Each algorithm produced a graph H and the corresponding shortest path weight matrix S_1 for DOR and S_2 for OLR. These results are depicted in Fig. 3.6, Equation (3.17) and (3.18), respectively.

$$S_1 = \begin{bmatrix} 0 & 100 & 500 & 50620 & 120 & \mathbf{600} & 120 & 620 & 140 & 100 \\ 100 & 0 & 600 & 50520 & 20 & 500 & 20 & 520 & 40 & 120 \\ 500 & 600 & 0 & 51120 & 620 & 1100 & 620 & 1120 & 640 & 600 \\ 50620 & 50520 & 51120 & 0 & 50540 & 51020 & 50500 & 50000 & 50560 & 50600 \\ 120 & 20 & 620 & 50540 & 0 & \mathbf{520} & 40 & 540 & 20 & 140 \\ \mathbf{600} & 500 & 1100 & 51020 & \mathbf{520} & 0 & 520 & 1020 & 540 & 620 \\ 120 & 20 & 620 & 50500 & 40 & 520 & 0 & 500 & \mathbf{60} & 100 \\ 620 & 520 & 1120 & 50000 & 540 & 1020 & 500 & 0 & 560 & 600 \\ 140 & 40 & 640 & 50560 & 20 & 540 & \mathbf{60} & 560 & 0 & 160 \\ 100 & 120 & 600 & 50600 & 140 & 620 & 100 & 600 & 160 & 0 \end{bmatrix} \tag{3.17}$$

$$S_2 = \begin{bmatrix} 0 & 88 & 200 & 20280 & 96 & 288 & 80 & 280 & 104 & 40 \\ 88 & 0 & 288 & 20208 & 8 & 200 & 8 & 208 & 16 & 48 \\ 200 & 288 & 0 & 20480 & 296 & 488 & 280 & 480 & 304 & 240 \\ 20280 & 20208 & 20480 & 0 & 20216 & 20408 & 20200 & 20000 & 20224 & 20240 \\ 96 & 8 & 296 & 20216 & 0 & 208 & 16 & 216 & 8 & 56 \\ 288 & 200 & 488 & 20408 & 208 & 0 & 208 & 408 & 216 & 248 \\ 80 & 8 & 280 & 20200 & 16 & 208 & 0 & 200 & 24 & 40 \\ 280 & 208 & 480 & 20000 & 216 & 408 & 200 & 0 & 224 & 240 \\ 104 & 16 & 304 & 20224 & 8 & 216 & 24 & 224 & 0 & 64 \\ 40 & 48 & 240 & 20240 & 56 & 248 & 40 & 240 & 64 & 0 \end{bmatrix} \tag{3.18}$$

As demonstrated in (3.17), we highlight the shortest path weights which are different from the given specific E2E demands in (3.16). When we use DOR, all the shortest path weights are equal to the given E2E demands except for those that are not achievable. The OLR algorithm necessitates the input parameter b in addition to the demand matrix D', defining the allowed deviation of the norm of $||D-S||$ from 0. For the example illustrated in Fig. 3.6, we adopted $b = 0.4$. Lower values of b necessitate a reduced allocation of resources across the same set of links, culminating in diminished shortest path weights between all conceivable pairs of nodes. This outcome engenders sparser topologies due to the lowered link weights employed. Conversely, higher values of b impose greater link weights, which in turn lead to quicker breaches of the upper shortest path weight bounds provided in D during the iterative process. Therefore, a higher b value results in a higher-density topologies. Our simulations confirmed that reducing b produced sparser graphs but increased the difference between the shortest path weights of the resulting graph H and the given demands. Consequently, selecting the input parameter b represents a compromise between reducing the sparsity of the graph H topology and maximising the corresponding shortest path weights.

3.7. CONCLUSION

This work focuses on inverse all shortest path problem (IASPP), which is a novel problem with promising applications, such as network modelling and design, in transportation networks, wireless sensor and actuator networks, connected vehicle applications, smart factory networks, etc. We present the Descending Order Recovery (DOR) algorithm to solve the optimised inverse all shortest path problem (OIASPP) and prove that OIASPP is not NP-complete. The graph obtained by DOR does not have redundant links and reaches a minimum number of links and a minimum sum of the link weights among all OIASPP solutions given a demand matrix D. DOR can also be regarded as an effective method when solving network sparsification that preserves all shortest path weights. Additionally, we utilise the information captured by the effective resistance between node pairs and propose Omega-based Link Removal (OLR) algorithm that solves the OIASPP. Both DOR and OLR provide solutions to the OIASPP: the solution obtained by DOR has the shortest path weight matrix $S = D$, while OLR focuses on solving OIASPP by providing sparser graphs, at the cost of the norm $||D-S|| > 0$. The ideas of DOR and OLR are different: DOR focuses on the shortest paths and the shortest path weights in a graph, while OLR investigates the shortest path weights from the perspective of the effective resistance.

4

EXTENSIONS OF THE INVERSE ALL SHORTEST PATH PROBLEM

The past cannot be changed,
but the future can still be shaped.

Confucius

The inverse all shortest path problem *(IASPP) seeks to construct a weighted adjacency matrix of a graph such that all the elements in the corresponding shortest path weight matrix are not larger than those of the demand matrix, given that the demand matrix is symmetric and non-negative. This chapter revises IASPP and proposes two extensions with additional constraints, motivated by the fact that network reconfiguration is sometimes infeasible or needs to be performed with economic constraints. The* inverse all shortest path problem that minimizes the adjustment of the sum of link weights *($IASPP_M$) focuses on obtaining a new graph where the shortest path weight matrix S equals a predetermined demand matrix D, meanwhile, minimizing the total adjustment of link weights in the original graph. We have shown that $IASPP_M$ is not NP-hard and propose the* Irreducible Backbone Augmentation *(IBA) algorithm to solve it exactly. If reconfiguration of network connections is not allowed, we study the* inverse all shortest path problem where the adjacency matrix is fixed *($IASPP_F$). A* distance basis sum *(DBS) algorithm is proposed to solve $IASPP_F$ efficiently. When solving resource provisioning problems with end-to-end Quality of Service (QoS) demands, DBS markedly outperforms well-established schemes such as Linear Programming in terms of execution speed.*

This chapter is based on a report [121].

4.1. INTRODUCTION

A fundamental challenge when operating a communication network is to provide end-to-end Quality of Service (QoS) to a number of applications with diverse service requirements. For instance, a traditional voice-over-IP packet requires a maximum delay of 150ms between a source and a destination pair. An interactive cloud virtual/augmented reality (VR/AR) video should be delivered to the user within 15ms to achieve optimal user experience. To ensure real-time control and automation, the network in a smart factory needs to provide ultra-low latency (i.e., less than 1ms) during data transmission [122]. These service requirements are often represented as a demand matrix D, which contains the maximum tolerable end-to-end (E2E) delay d_{ij} between a node pair (i, j). A network operator is asked to dimension the network topology and link weights so that transporting packets over the shortest path between any pair of nodes (i, j) consumes less time than its E2E latency d_{ij}.

The aforementioned problem is formally defined as the *Inverse All Shortest Path Problem* (IASPP) in [7, 8]. IASPP is central to the design and planning of network infrastructure, where the goal is to construct a network that strictly guarantees the specified E2E upper bounds between nodes. During the operation of a network, the demand matrix of the network changes over time due to the introduction of new services and departure of existing flows. Network failures may also occur. These conditions necessitate a recomputation of the topology and link weights with IASPP. It is important to recognize that topological reconfiguration is often impractical or must be strictly constrained with minimal operational expenditure once a network is in production. For instance, network downtime (e.g., due to modification of the existing network) can lead to significant operational losses and safety risks, hence, need to be minimized or completely avoided. This motivates us to revise the IASPP formulation and extend it to two distinct variations to accommodate the deployment challenges described above. The two IASPP extensions are 1) *inverse all shortest path problem that minimized the adjustment of the sum of link weights* (IASPP$_M$) and 2) *inverse all shortest path problem where modification of the adjacency matrix is not allowed* (IASPP$_F$).

Before delving into the formulation of IASPP$_M$ and IASPP$_F$ in Section 4.2, we explain the terminology. Networks[1, 2] are pervasive in both natural (e.g., biological network, molecular interaction network, seismic network) and engineered systems (e.g., transportation, Telecommunications, power grids). A network is characterized by the underlying graph and the functional process associated with it [3, 4], where the function of a network is generally related to the transport of items over the underlying graph[5, 6]. A graph G consists of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L links. In this paper, we limit ourselves to undirected, connected simple [1] graphs. A graph G can be represented by graph related matrices, such as adjacency matrix, Laplacian matrix, incidence matrix, etc. The $N \times N$ adjacency matrix A is defined such that the element $a_{ij} = 1$ if there is a link between node i and node j, otherwise $a_{ij} = 0$. Each link $l = i \sim j \in \mathcal{L}$ between node i and j has a weight w_l (or w_{ij}), which is a positive real number that denotes a property of the link, e.g., the delay when transmitting IP packets over that link. The link weights of all node pair (i, j) compose an $N \times N$ link weight matrix W. We define the $N \times N$ weighted

[1] A graph G is simple if there is at most one link between each pair of nodes and the graph does not contain any self-loops.

adjacency matrix as $\widetilde{A} = W \circ A$, where the Hadamard product $\circ$ defines a direct elemental multiplication $\widetilde{a}_{ij} = w_{ij} a_{ij}$ and "tilde" refers to a weighted graph matrix. In our setting, $\widetilde{a}_{ij} = 0$ means that there is no link between node i and j, because we exclude zero link weights, i.e., $w_{ij} > 0$, as in Dijkstra's shortest path algorithm[29] and in order to avoid the complication that a zero weight $w_{ij} = 0$ would physically mean that node i and j are the same. Since we only consider undirected graphs in this work, the adjacency matrix A is symmetric.

In a weighted graph G, a path[4] from node i to node j with $k-1$ hops or links is a set of links $\mathscr{P}_{ij} = \{n_1 \sim n_2, n_2 \sim n_3, \dots n_{k-1} \sim n_k\}$, where $n_1 = i$ and $n_k = j$. The weight $w(\mathscr{P}_{ij}) = \sum_{l \in \mathscr{P}_{ij}} w_l$ of a path $\mathscr{P}_{ij}$ between a node pair (i, j) consists of the sum of the weights over all links $l = a \sim b$ that belong to that path $\mathscr{P}_{ij}$. Among all possible paths, the shortest path $\mathscr{P}^*_{ij}$ minimizes the sum of the positive weights of its constituent links and $w(\mathscr{P}^*_{ij}) \leq w(\mathscr{P}_{ij})$. The $N \times N$ shortest path weight matrix S contains all shortest path weights with elements $s_{ij} = w(\mathscr{P}^*_{ij})$. Given a weighted adjacency matrix $\widetilde{A}$, the shortest paths between pairs of nodes and the corresponding shortest path weight matrix S can be obtained efficiently by Dijkstra's shortest path algorithm in polynomial time. The shortest path weight matrix S is a distance matrix [2]. We refer to the link $l = i \sim j$ as a "redundant" link if there is another node k, which is different from node i and j, in the network such that $\widetilde{a}_{ij} = w_{ij} a_{ij} \geq s_{ik} + s_{kj}$. The network diameter ρ is defined as the hopcount of the "longest shortest path" in a network.

The paper is outlined as follows. In Section 4.2, we review the inverse all shortest path problem and formally define its two extensions IASPP$_M$ and IASPP$_F$. An overview of related work from literature is provided. Potential applications of IASPP$_M$ and IASPP$_F$ are also discussed. In Section 4.3, we focus on IASPP$_M$ and propose an *Irreducible Backbone Augmentation* algorithm to solve it exactly. In Section 4.4, IASPP$_F$ is examined. Given a tree topology, a *Distance Basis Sum* (DBS) algorithm is desgined to solve the problem with reduced computational complexity compared to well-established solutions such as Linear Programming. The performance of the DBS algorithm is evaluated extensively with simulations in Section 4.5. In Section 4.6, we summarize our results and conclude the paper.

4.2. INVERSE ALL SHORTEST PATH PROBLEM AND ITS EXTENSIONS

4.2.1. STATEMENT OF INVERSE ALL SHORTEST PATH PROBLEM AND ITS EXTENSIONS

In Section 3.2.1, we have introduced the definition of IASPP (Problem 1) and OIASPP (Problem 2). IASPP and OIASPP address the challenge of constructing a graph from scratch, specifying the demand matrix as input. Two extensions of IASPP, formulated as graph modification problems, are investigated as follows. The objective of the two extensions is to adjust the weight adjacency matrix of a graph so that the resulting shortest path weight matrix S satisfies the specified demands D with additional constraints.

[2] Any element h_{ij} of a distance matrix H is non-negative $h_{ij} \geq 0$, but $h_{ii} = 0$ and h_{ij} obeys the triangle inequality: $h_{ij} \leq h_{ik} + h_{kj}$.

Problem 5 (IASPP that minimizes the adjustment of the sum of link weights (IASPP$_M$))**.** *Given an $N \times N$ weighted adjacency matrix $\widetilde{A}$ and an $N \times N$ symmetric demand matrix D. Determine an $N \times N$ weighted adjacency matrix $\widetilde{A}'$, such that the corresponding shortest path weight matrix S' obeys $S' = D$ and minimize $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$.*

In IASPP$_M$, the resulting graph G' is not required to retain the same adjacency matrix A as the original graph. A more restrictive condition is applied to the following extension where construction of additional links and removal of existing links from the original network is not allowed.

Problem 6 (IASPP with fixed adjacency matrix (IASPP$_F$))**.** *Given an $N \times N$ adjacency matrix A and an $N \times N$ symmetric demand matrix D. Determine an $N \times N$ weighted adjacency matrix $\widetilde{A}'$, such that the norm $||D - S||$ between the shortest path weight matrix S and the demand matrix D is minimized.*

Given an adjacency matrix A where no modification on A is allowed, IASPP$_F$ appears to be identical to the Fekete's problem [38] that is reviewed in Section 4.2.2.

4.2.2. LITERATURE REVIEW

EXTENSIONS OF THE INVERSE ALL SHORTEST PATH PROBLEM

In Section 3.2.2, we have reviewed IASPP and OIASPP, which focuses on designing a network with a predetermined demand matrix D as the only input. The extended problems incorporate further constraints where both the shortest path weights and the original graph are known. This is motivated by a number of use-cases (Wide Area Networks or industrial automation networks) where the network operator must reconfigure the network in response to the updated demand matrix, with minimal (or no) modifications on the underlying topology.

In 1999, Fekete et al. [38] considered a graph G with an adjacency matrix A and a symmetric demand matrix D. The goal is to determine the weighted adjacency matrix $\widetilde{A}'$, such that the adjacency matrix remains as A and the corresponding shortest path weight matrix S is exactly the demand matrix D. Fekete et al. [38] showed that the problem is NP-complete by reducing it to the vertex-disjoint paths problem, except for some very restricted cases. One polynomially solvable instance occurs when the demand matrix D specifies shortest path weights for all node pairs in the network.

Hung [39] extended Fekete's problem and demonstrated additional instances that can be solved polynomially. Given an adjacency matrix A and a demand matrix D, Hung proposed several heuristic algorithms to find a weighted graph such that the norm $||S - D||$ between the shortest path weight matrix S and the demand matrix D is minimized, while the adjacency matrix A remains unchanged. Hung's problem is identical to IASPP$_F$ proposed in this paper. A comparative analysis of Hung's scheme and our methodology in solving IASPP$_F$ is presented in Section 4.5.

Based on Fekete's problem, Cui and Hochbaum [40] investigated a more stringent variant called the *inverse shortest path lengths problem*. Given a weighted adjacency matrix $\widetilde{A}$ and demand matrix D, Cui and Hochbaum aim to find the weighted adjacency matrix $\widetilde{A}'$ so that the demand matrix D matches exactly the shortest path weight matrix S. Modification of the adjacency matrix A is not allowed in Cui and Hochbaum's

problem. Adjustment of the link weights should be minimized. In contrast to Cui and Hochbaum's problem, IASPP$_M$ defined in this paper only requires a demand matrix as input. Cui and Hochbaum [40] proved that the inverse shortest path lengths problem is strictly harder than the one studied by Fekete. Inspired by Hung[39], a heuristic algorithm based on a compact convex programming formulation is proposed in [40] to find an approximated solution. Leitão, et al. [123] studied the same problem and developed a solution driven by genetic algorithms. Another variation of the inverse shortest path lengths problem is formulated in Cui and Hochbaum [40] by incorporating lower bounds when calculating the shortest path weights.

4.2.3. APPLICATION OF IASPP$_M$ AND IASPP$_F$

To meet stringent E2E QoS demands, recent research work focuses on selecting an optimal path for each application by minimizing a global cost function such as bandwidth or delay [124]. A flow table is created and updated once a new path has been computed for a service. Unlike the aforementioned *QoS-aware path selection* methodology, the IASPP framework is fully compatible with the existing routing infrastructure. Once the link weights (and the network topology) have been computed, the network executes a shortest path scheme (e.g., Dijkstra's algorithm) for packet forwarding. The above property of IASPP enables its seamless integration with the existing network infrastructure and avoids the overhead of creating and maintaining a separate flow table.

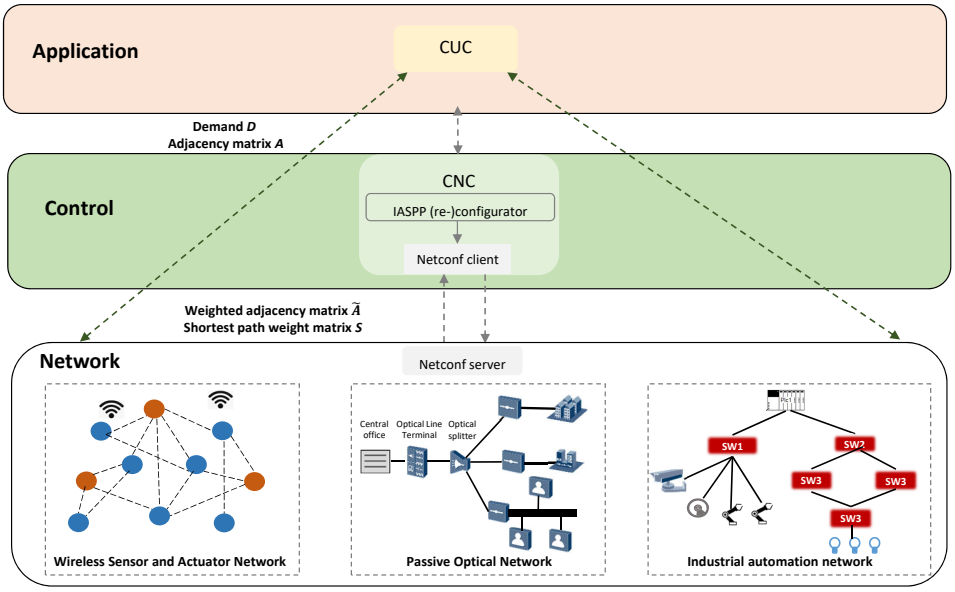


Figure 4.1: Diagram of centralized deployment of the IASPP framework. The IASPP configurator runs different algorithms that solves IASPP.

We consider IASPP$_M$ as a useful tool for network planning and reconfiguration. For example, urban planners may need to reconfigure roads or transit systems in response

to significant changes in the population distribution or traffic volumes over time. Our solution to IASPP$_M$ in Section 4.3 offers an exact algorithm to adjust the network so that the new demands between nodes, e.g., the commute time between two stations, are satisfied. The existing infrastructure is maximally preserving. Another example is an Industrial Wireless Sensor and Actuator Network (IWSAN) in process automation [125, 126]. An IWSAN consists of a gateway and hundreds of thousands of field devices (e.g., sensors and actuators), forming a multi-hop wireless network. The sensors are constantly transmitting information such as temperature, vibration and voltage to a programmable logic controller (PLC), which further sends out control signal to the actuators. E2E latency in the IWSAN network is typically bounded to 10-100ms [127]. A Centralized User Configuration (CUC) module registers a new device with the application running on it. The service requirement (i.e., bandwidth, latency) is also registered with the CUC. In case of topological changes such as new joining nodes, link failure, or device departure, the existing link weights may not be able to accommodate the updated latency demands. The CUC informs the Centralized Network Configuration (CNC)[3] unit about the updated demand matrix and topological changes associated with it. The CNC runs the IBA scheme (see Section 4.3) to (re-)compute the weighted adjacency matrix $\tilde{A}$. The updated weighted adjacency matrix are communicated with devices in the network using network management protocols such as NetConf [128], see Fig. 4.1.

IASPP$_F$, on the other hand, is applied to reconfigure the network where modification of the adjacency matrix A is not allowed. Potential applications are passive optical networks (PON) and networks in industrial automation, where network availability is paramount to ensuring operational continuity. Reconfiguring such networks (e.g., re-wiring links) is prohibitively difficult or physically impossible. Applications running on these networks (e.g., autonomous driving, interactive AR/VR videos, artificial intelligence inference and training, industrial control) may change over time, thus, asking for an update of the shortest path weight matrix S. In the Telecommunications and Industrial Automation sector, the Multiple Spanning Tree Protocol [129, 130] is often used to construct (multiple) loop-free paths between nodes to ensure network availability and reliability [4]. Our solution to IASPP$_F$ therefore focuses on a tree. Deployment of the IASPP$_F$ framework follows the same architecture in Fig. 4.1. Common issues related with a centralized QoS provisioning framework, such as risk of single failure, packet loss due to change of shortest path are beyond the scope of this paper. A further step is to consider scheduling and bandwidth constraints, power consumption and path redundancy into the problem.

In Section 4.3, we demonstrate that IASPP$_M$ is not NP-hard and develop an exact algorithm to solve it. In Section 4.4, we concentrate on solving IASPP$_F$ with a tree topology. The scheme to solve IASPP$_F$ is evaluated in Section 4.5 and compared with the Hung's scheme [5].

[3] CUC and CNC can be deployed as a standalone service or integrated directly in the IWSAN gateway.
[4] Failover from a primary tree to a backup tree is triggered in case of failure.
[5] We refer Hung's scheme to Algorithm 3 in [39], as it outperforms all other algorithms discussed in [39].

4.3. IRREDUCIBLE BACKBONE AUGMENTATION ALGORITHM

In this section, we propose the Irreducible Backbone Augmentation algorithm (IBA) to solve IASPP$_M$ exactly. Among all the graphs with the same shortest path weight matrix S, there is only one graph which does not have redundant links [8, 37]. We refer to such a graph as an *irreducible backbone* for short. Adding any redundant link to an irreducible backbone does not change the shortest path weight matrix. The main idea to solve IASPP$_M$ is to find an irreducible backbone whose shortest path weight matrix matches the given demand and then add redundant links to it, such that the difference between the resulting graph and the original graph is minimized. The process is outlined as follows:

1. Given a demand matrix D and a graph G with weighted adjacency matrix $\widetilde{A}$ and link weight matrix W as input, construct a complete graph K_N with a weighted adjacency matrix $\widetilde{A} = D$.
2. Obtain a graph G' by removing all the redundant links in the complete graph K_N.
3. For each link $l = i \sim j$ in the original graph G but does *not* exist in the graph G', a corresponding link $l = i \sim j$ is added between nodes i and j in G'.
4. If the link weight follows $w_{ij} \geq d_{ij}$ in the input graph G, set the link weight in graph G' to $w'_{ij} = w_{ij}$. If $d_{ij} < 2w_{ij}$, set the link weight in graph G' to $w'_{ij} = d_{ij}$. If $d_{ij} \geq 2w_{ij}$, remove the added link $l = i \sim j$ in G'.
5. Repeat step 3-4 for all links present only in the input graph G but not in G'.

The following analysis restricts to demand matrices D that are distance matrices[6]. In steps 1-2, we construct an irreducible backbone G' whose shortest path weight matrix equals the given demand matrix D by executing the DOR algorithm (see details in [8]). Step 1 ensures that the generated complete graph K_N is an exact solution to OIASPP because each node i can reach another node j via a direct link with weight $w_{ij} = d_{ij}$ in K_N. For the generated graph K_N, the shortest path weight matrix follows $S = D$. The graph G' (i.e., the irreducible backbone) obtained from step 2 has a minimum number of links and a minimum sum of link weights among all graphs with the same shortest path weight matrix [8].

The subsequent steps 3–5 add links to the backbone to minimize the adjustment of the sum of link weights between the resulting graph and the input graph. Each added link $l = i \sim j$ is determined according to two principles: (a) link l is redundant, i.e., adding or removing l does not change the shortest path weight matrix S; and (b) the chosen link weight w'_{ij} minimizes the norm $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$ between the resulting weighted adjacency matrix $\widetilde{A}'$ and the input matrix $\widetilde{A}$, among all possible choices of link weight. Specifically, since each term $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| \geq 0$, the summation $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$ is minimized if each term is minimized. In step 3, we add a link $l = i \sim j$ to the resulting graph G' only if link $l = i \sim j$ exists in the input graph G but not in G'. Hence, for any node pairs (i, j) that is not connected directly in both the input graph G and the resulting graph G', the corresponding term $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = 0$.

[6] As discussed in Section 2.1, any shortest path weight matrix S of a graph is a distance matrix.

For each added link $l = i \sim j$ in step 3, we then compare the weight w_{ij} of link l in the input graph with the given demand $d_{ij} = s_{ij}$, see step 4. To ensure the shortest path weight matrix S remains unchanged, the weight of the added link l in the resulting graph G' follows $w'_{ij} \geq d_{ij}$. If the weight of link l in the input graph $w_{ij} \geq d_{ij}$, we set the weight of the added link to $w'_{ij} = w_{ij}$ in the resulting graph G', which ensures $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = 0$. The shortest path weight matrix S remains unchanged because $w'_{ij} = w_{ij} \geq d_{ij}$.

When the link weight $w_{ij} < d_{ij}$, we can not simply let the weight of the added link to $w'_{ij} = w_{ij}$ in the resulting graph G' because link l will not be redundant in G'. To keep shortest path weight matrix S remains unchanged, there are two possible cases: (a) to ensure link l is redundant, set the weight of the added link to $w'_{ij} \geq d_{ij}$ in the resulting graph G', thereby the term $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = |w'_{ij} - w_{ij}| \geq d_{ij} - w_{ij}$ holds; and (b) remove the added link l in step 3 from G', and we have $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = w_{ij}$. Thus, when $d_{ij} - w_{ij} < w_{ij}$, i.e., $d_{ij} < 2w_{ij}$, we set the weight of the added link to $w'_{ij} = d_{ij}$. Otherwise, we remove the added link $l = i \sim j$ from the resulting graph G'. If $d_{ij} = 2w_{ij}$, removing the added link $l = i \sim j$ or keeping the added link with link weight $w'_{ij} = d_{ij}$ in the resulting graph minimizes $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = w_{ij}$ without changing the shortest path weight matrix S. After adding links in step 3 – 5, we obtain a graph G' as a solution to IASPP$_M$. The meta code is shown in Algorithm 4.

Algorithm 4 Irreducible Backbone Augmentation algorithm

Input: $N \times N$ demand matrix D: a distance matrix;
$N \times N$ weighted adjacency matrix $\widetilde{A}$ of an original graph G
Output: $N \times N$ weighted adjacency matrix $\widetilde{A}'$

$K_N \leftarrow$ a graph with weighted adjacency matrix $\widetilde{A}_K = D$
graph G' and its weighted adjacency matrix $\widetilde{A}' \leftarrow$ remove all redundant links in K_N
$\forall$ link $l = i \sim j$ in G but not in G'
    $a'_{ij} \leftarrow 1, a'_{ji} \leftarrow 1$
    if $w_{ij} \geq d_{ij}$ **then**
        $w'_{ij} \leftarrow w_{ij}, w'_{ji} \leftarrow w_{ji}, \widetilde{a}'_{ij} \leftarrow w'_{ij}a'_{ij}, \widetilde{a}'_{ji} \leftarrow w'_{ji}a'_{ji}$
    else if $|d_{ij} - w_{ij}| < w_{ij}$ **then**
        $w'_{ij} \leftarrow d_{ij}, w'_{ji} \leftarrow d_{ji}, \widetilde{a}'_{ij} \leftarrow w'_{ij}a'_{ij}, \widetilde{a}'_{ji} \leftarrow w'_{ji}a'_{ji}$
    else
        $a'_{ij} \leftarrow 0, a'_{ji} \leftarrow 0$
    end if
return $\widetilde{A}'$

With a weighted adjacency matrix $\widetilde{A}$ and a demand matrix D as input, IBA generates a graph that is exact and unique to most IASPP$_M$ instances (see details in Appendix C.2). In rare circumstances, exceptions may occur. Consider a node pair (i, j) with link weight $\widetilde{a}_{ij} > 0$, the link $l = i \sim j$ between node i and j does not present in the irreducible backbone. Suppose the demand between node i and j follows $d_{ij} = 2\widetilde{a}_{ij}$. The difference of the link weight on link $l = i \sim j$ between graph G and G' is $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = \widetilde{a}_{ij}$. This leads to two outcomes: 1) adding the link $l = i \sim j$ with weight $w'_{ij} = d_{ij} = 2\widetilde{a}_{ij}$ to graph G'

or 2) exclude link $l = i \sim j$ in graph G'. Hence, the graph G' obtained by Algorithm 4 is no longer unique. Given m of the aforementioned node pairs, the total number of non-unique solution graphs is 2^m. To ensure IBA returns a unique solution G', we exclude link $l = i \sim j$ in graph G' whenever $d_{ij} = 2\widetilde{a}_{ij}$ occurs (see lines 7-10 in Algorithm 4).

The computational complexity of IBA is mainly from two parts: (1) Remove redundant links from a complete graph in Step 2 and (2) Add new links to the irreducible backbone in Steps 3-4. As shown in [8], removing all redundant links from a complete graph requires $O(N^2)$ computational iterations. During Steps 3-4, at most $\binom{N}{2} - (N-1)$ links may need to be considered and the worst case is where the irreducible backbone is a tree with $N-1$ links. Thus, Steps 3-4 also incurs $O(N^2)$ computational iterations, leading to the overall computational complexity of IBA as $O(N^2)$.

In summary, the IBA algorithm provides an exact solution to solve IASPP$_M$ with polynomial time complexity. To our knowledge, an exact solution to IASPP$_M$ has not been examined in prior work. Given an arbitrary graph, the IBA algorithm enables efficient reconstruction of the graph that satisfies the E2E demands, with minimum changes to the topology.

4.4. DISTANCE BASIS SUM ALGORITHM

In this section, we focus on solving IASPP$_F$ on a tree graph. Construction of a *spanning tree* or *minimum spanning tree* is a common practice in designing resource-efficient networks, e.g., transport networks, airline routes planning and Telecommunication networks. As discussed in Section 4.2.3, once in operation, the network topology is often fixed. The operator can reconfigure network parameters (e.g., the shortest path weight matrix) to satisfy demands that change over times.

Define the norm $||D-S|| = \frac{1}{2}\sum_{i,j}|d_{ij}-s_{ij}|$ as the difference between the shortest path weight matrix and the demand. We consider a *Linear Programming with link weights as variables* (LPLW) approach to solve IASPP$_F$ on a tree (see Appendix C.3). LPLW solves a linear programming problem with two types of variables. The first type consists of L link weight variables ($L = N-1$ in a tree) that determines the topological property of the underlying graph. The second type includes $\frac{N(N-1)}{2}$ slack variables, representing the difference between the shortest path weight s_{ij} and the demand d_{ij}. In total, LPLW has maximally $\frac{N(N-1)}{2} + L = \frac{(N+2)(N-1)}{2}$ variables and $N(N-1)+L = (N+1)(N-1)$ constraints[7]. Since the computational cost to solve a linear program problem is positively related with the size of its variable and constraint space, we propose the *Distance Basis Sum* (DBS) algorithm to reduce the IASPP$_F$ variables and constraints. The main idea of DBS is:

1. Given a demand matrix D and an adjacency matrix A, find a shortest path weight matrix S such that the shortest path distance matrix S can be derived from a graph with the adjacency matrix A, where $||D-S||$ is minimized.

2. Obtain the weighted adjacency matrix $\widetilde{A}$ by executing DOR [8] with S as input.

[7] The constraints are derived by enforcing positivity on the link weights and linearizing the absolute differences using the slack variables.

Before introducing the steps to find a proper shortest path weight matrix S in DBS, we explain a property of the shortest path weight matrices in tree graphs. Given a tree with adjacency matrix A, the shortest path $\mathscr{P}^*_{ij}$ between any pair of nodes (i,j) remains unchanged despite of the changes of the link weight matrix W, because there is only one path between a node pair (i,j). Let S_1 and S_2 be two shortest path weight matrices computed from two trees with the same adjacency matrix A but different link weight matrices W_1 and W_2. The sum of the shortest path weight between a node pair (i,j) follows

$$s_{ij} = (s_1)_{ij} + (s_2)_{ij} = \sum_{(a\sim b)\in\mathscr{P}^*_{ij}} (w_1)_{ab} + \sum_{(a\sim b)\in\mathscr{P}^*_{ij}} (w_2)_{ab} = \sum_{(a\sim b)\in\mathscr{P}^*_{ij}} ((w_1)_{ab} + (w_2)_{ab}),$$

where $(s_1)_{ij}, (s_2)_{ij}, (w_1)_{ab}, (w_2)_{ab}$ represent the element in S_1, S_2, W_1, W_2 respectively. This is to say that the sum $S = S_1 + S_2$ of two shortest path weight matrices of two trees, which have the same adjacency matrix A but different link weight matrices W_1 and W_2, is also the shortest path weight matrix of a tree with adjacency matrix A.

Assume that there is a tree graph T whose shortest path distance matrix S equals the given demand matrix D. The demand matrix can be decomposed into m shortest path weight matrices, i.e., $D = \sum_{k=1}^{m} \epsilon_k S_k$, where ϵ_k is a nonnegative scaling factor, S_k is the $k-$th shortest path weight matrix after decomposition. The m shortest path distance matrices share an identical adjacency matrix A. We define the shortest path weight matrix S_k that constitutes a demand matrix (or a shortest path weight matrix that is close to the demand matrix) as a *distance basis matrix* and m as the number of distance basis matrices. With a given adjacency matrix A, we can generate different distance basis matrices S_k by varying the link weight matrix. Assume we have m distance basis matrices S_k for a given adjacency matrix A and demand matrix D, IASPP$_F$ asks for a set of $m \times 1$ scaling factors $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_m\}$ such that:

$$\begin{aligned} \min \quad & \sum_{i,j} |d_{ij} - s_{ij}| \\ \text{s.t.} \quad & s_{ij} = \sum_{k=1}^{m} \epsilon_k (s_k)_{ij}, \quad \forall i < j, \\ & \epsilon_k \geq 0, \forall k. \end{aligned} \tag{4.1}$$

where $(s_k)_{ij}$ denotes the elements of the distance basis matrix S_k. By introducing an auxiliary variables $z_{ij} \geq d_{ij} - s_{ij}$ and $z_{ij} \geq s_{ij} - d_{ij}$:

$$\begin{aligned} \min \quad & \sum_{i,j} z_{ij} \\ \text{s.t.} \quad & z_{ij} \geq d_{ij} - \sum_{k=1}^{m} \epsilon_k (s_k)_{ij}, \quad \forall i < j, \\ & z_{ij} \geq \sum_{k=1}^{m} \epsilon_k (s_k)_{ij} - d_{ij}, \quad \forall i < j, \\ & \epsilon_k \geq 0, \forall k. \end{aligned} \tag{4.2}$$

As shown in (4.2), the DBS scheme has $\frac{N(N-1)}{2} + m$ variables and $N(N-1) + m$ constraints (where $2 \leq m \leq L$). If $m = L$, the complexity of DBS converges to that of LPLW. After solv-

ing Eq. (4.2), we obtain m scaling factor ϵ_k associated with the distance basis matrices S_k. The obtained shortest path weight $S = \sum_{k=1}^{m} \epsilon_k S_k$ is a shortest path weight matrix of a graph with adjacency matrix A by minimizing the norm $||D - S||$. A solution graph G' can be further obtained by using DOR with the obtained shortest path weight matrix S as input. Because the graph G' obtained by DOR excludes all redundant links and there is only one graph without any redundant link for all graphs that have the same shortest path weight matrix, the adjacency matrix of graph G' equals the given adjacency matrix A of the tree T. The pseudocode for DBS is introduced in Algorithm 5.

Algorithm 5 Distance basis sum algorithm (DBS)

Input: $N \times N$ demand matrix D: a distance matrix;
$N \times N$ adjacency matrix A of an original tree graph T;
m: Number of distance basis matrices, $m \leq L$; α: An insignificant positive real number.
Output: $N \times N$ weighted adjacency matrix $\widetilde{A}'$.

1: $\widetilde{A} \leftarrow \alpha A$
2: $S_1 \leftarrow$ shortest path weight matrix of T
3: $S_2 \leftarrow$ shortest path weight matrix from A
4: $\forall k \in \mathbb{Z}, 3 \leq k \leq m$
5: $\quad \widetilde{A}_k \leftarrow \widetilde{A}$
6: $\quad i, j \leftarrow$ end nodes of the k-th link in T
7: $\quad (\widetilde{a}_k)_{ij} \leftarrow 1, (\widetilde{a}_k)_{ji} \leftarrow 1$
8: $\quad S_k \leftarrow$ shortest path weight matrix of $\widetilde{A}_k$
9: $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_m\} \leftarrow$ solve Eq. 4.2 with $\{S_1, S_2, \ldots, S_k\}$ as input
10: $S \leftarrow \sum_{k=1}^{m} \epsilon_k S_k$
11: $\widetilde{A}' \leftarrow$ execute DOR with S as input
12: **return** $\widetilde{A}'$

In Algorithm 5 line 4-8, we construct a distance basis matrix S_k as followings. We first generate a weighted adjacency matrix $\widetilde{A} = \alpha A$ by multiplying the adjacency matrix with an insignificant positive α, e.g. 0.001. To generate a distance basis matrix, we randomly select a link l that is different from any selected links in other distance basis matrices and let the link weight $w_l = 1$. We then obtain the shortest path weight matrix S_k as the k-th distance basis matrix from the weighted adjacency matrix $\widetilde{A}$. The performance of DBS is evaluated in Section 4.5.

4.5. PERFORMANCE EVALUATION OF DBS

This section evaluates the performance of DBS on an empirical tree network as well as tree networks derived from random graphs[8]. The performance of the DBS is assessed by three criteria: (i) The norm $||D - S|| = \frac{u|D-S|u^T}{uDu^T}$ between the demand matrix D and the shortest path weight matrix S, where u represents the $N \times 1$ all-one vector. (ii) The computation time t that it takes to find a solution. (iii) The number of instances that

[8] The IBA algorithm is an exact solution to solve with polynomial time complexity, thus, is not evaluated with simulation.

can be successfully found within a pre-defined time limit. We take LPLW and the Hung's method as the benchmark schemes throughout the evaluation.

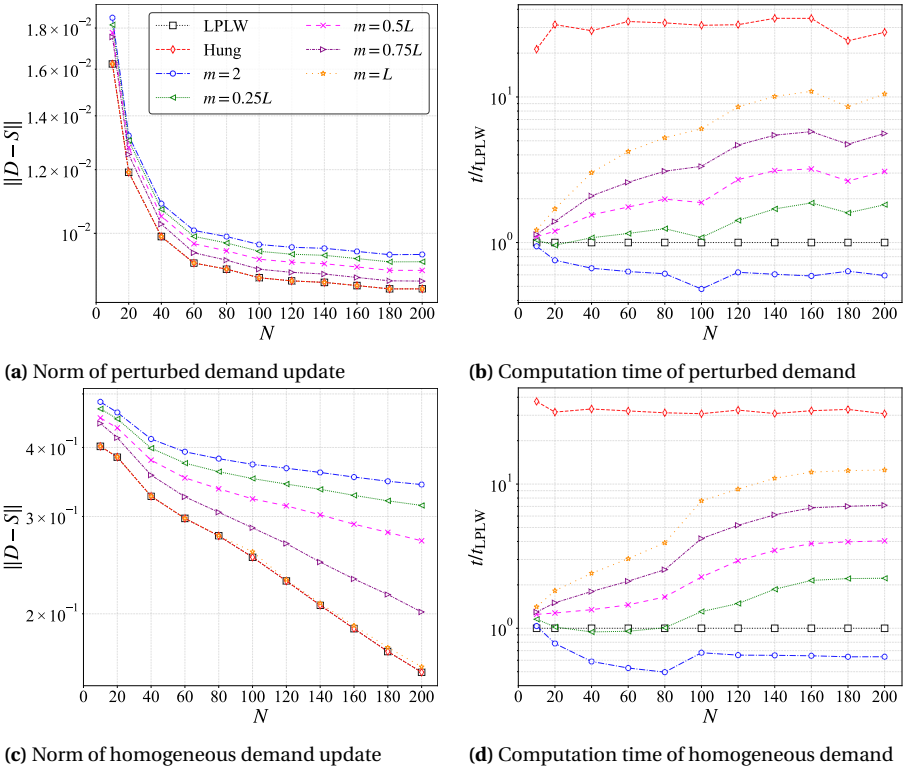
The DBS algorithm is evaluated with different numbers of distance basis, i.e., $m = 2$, $m = 0.25L$, $m = 0.5L$, $m = 0.75L$, and $m = L$, where $L = N - 1$ denotes the number of links in the tree network. The insignificant positive constant in DBS is set to $\alpha = 0.01$. Both LPLW and DBS are implemented in MATLAB [9] and evaluated across three settings. The first configuration (Section 4.5.1) simulates a system with homogeneous service demands, i.e., the demand matrix fluctuates marginally. The second setting models a system in which heterogeneous service requirements need to be accommodated (Section 4.5.2). The performance of DBS on tree networks with different diameters and widths is examined in Section 4.5.3. Unless otherwise specified, we construct minimum spanning trees T from an Erdős–Rényi graph (ER), whose link weights are integers following the uniform distribution in $[1, 10]$. An empirical tree network is also considered in Section 4.5.3.

4.5.1. PERFORMANCE OF DBS WITH HOMOGENEOUS SERVICE DEMAND

In this section, we employ two schemes to generate the demand matrix. The first scheme constructs demand updates based on the statistical fluctuation of the current shortest path weights. It is designed to test the performance of DBS against minor modifications in service requirements. By perturbing the shortest path weight matrix S of the generated tree T with random noise θ uniformly distributed in $U(0, 1)$, we obtain the demand matrix D whose element follows $d_{ij} = d_{ji} = s_{ij} + \theta$. We refer to the first scheme as *perturbed demand update*. In the second scheme, an update of the demand matrix D is generated as a symmetric random distance matrix with zero diagonal elements and non-diagonal elements drawn uniformly from $[1, 10]$. It corresponds to a network characterized with moderate E2E delay demands. For instance, applications running on the network can be voice or video calls, interactive VR/AR services that need to be delivered to the user within 1 to 10ms. We denote the second scheme as *homogeneous demand update*. Due to the randomness when generating d_{ij}, the resulting demand matrix D does not possess any structural regularity. High and low demand entries may be arbitrarily generated in the demand matrix D. We modify the demand matrix D to a distance matrix according to the method introduced in Section 3.2.1. That is, in case of violation of the triangle inequality in a distance matrix (if $d_{ik} + d_{kj} < d_{ij}$ for at least one node $k \in \mathcal{N}$), we replace $d_{ij} = \min_{1 \le k \le N}(d_{ik} + d_{kj})$ and $d_{ji} = d_{ij}$ to make sure the demand matrix D is a distance matrix.

Figs. 4.2a and 4.2c illustrate the norm $\|D - S\|$ of the graph obtained by LPLW, Hung's scheme and DBS with different numbers of distance basis m. We highlight the main observations as follows. First of all, Hung's method generates the same norm as LPLW for both demand update schemes. The norm obtained by DBS with $m = L$ overlaps with that of LPLW and Hung's scheme. Decreasing the distance basis m leads to larger difference between the shortest path weight matrix and the demand. Second, the norm $||D - S||$ decreases with the increment of graph size N in both demand update schemes. Com-

[9] The experiments are conducted on a system with Intel(R) Core(TM) i5-10505 CPU and 8GB RAM for network sizes $N \le 100$ and on a system with Intel XEON E5-6448Y 32-Core 2.1GHz CPU and 16GB RAM for network sizes $N > 100$.



(a) Norm of perturbed demand update

(b) Computation time of perturbed demand

(c) Norm of homogeneous demand update

(d) Computation time of homogeneous demand

Figure 4.2: Performance of DBS with perturbed demand update in (a) and (b) and homogeneous demand update scheme in (c) and (d). The x-axis indicates the graph size. The y-axis is the norm $||D-S||$ averaged over all simulations in (a)(c). The y-axis in (b)(d) plots the computation time t/t_{LPLW} normalized by LPLW. Simulations are repeated 1000 times. LPLW is normalized against itself, hence, remains constant across all graph sizes.

pared to the homogeneous update scheme (Fig. 4.2c), the norm $\|D-S\|$ obtained by the perturbed update scheme (Fig. 4.2a) is smaller and converges faster as N increases. This is because the homogeneous update scheme handles arbitrarily generated high and low demands, which produces significant differences in $|d_{ij}-s_{ij}|$ and leads to a larger norm $||D-S||$. On the other hand, the perturbed demand update scheme constructs its demand matrix by adding small perturbations to an existing *valid* shortest path weight matrix. The new demand matrix preserves, to a significant extent, the structural consistency of an exact shortest-path metric. Hence, the perturbed demand update scheme exhibits substantially fewer conflicts between demands and shortest path weights[10] compared

[10] Recall that the norm $\|D-S\|$ is a summation of the difference $|d_{ij}-s_{ij}|$ between the given demands and shortest path weights for all node pairs (i,j). Consider a three-node graph consisting of nodes i, j and k connected by link $i\sim j$ and $j\sim k$, with demands $d_{ik}=5$, $d_{ij}=15$ and $d_{jk}=2$ respectively. The link weights are assigned as $w_{ij}=3$ and $w_{jk}=2$ (the shortest path weight $s_{ik}=w_{ij}+w_{ik}=5$) to satisfy the demand requirement between node i and k. A large difference of $|d_{ij}-s_{ij}|=12$ is observed on link $i\sim j$. In the special case where $d_{ik}=d_{ij}+d_{jk}$, the shortest path weights between link $i\sim j$ and $j\sim k$ can be directly

to the homogeneous update scheme. The standard deviation of the norm $||D-S||$ is evaluated against different graph sizes N in Appendix C.4.

Figs. 4.2b and 4.2d plot the average computation time t (normalized to the computation time t_{LPLW} of LPLW) to solve an IASSP$_F$ instance with DBS and Hung's scheme. As we can see, the computation time of Hung's method (the red dashed curve with diamond markers) is approximately 10 times that of LPLW (the black dashed line with square markers), even though both schemes produce the same norm $||D-S||$. For graph size $N > 20$, DBS with $m = 2$ (the blue dashed curve with circle markers) consistently achieves more than 40% reduction in the computation time relative to LPLW at the cost of increased norm $||D-S||$. In general, increasing the distance basis reduces the norm, but incurs a higher computational cost. Unless otherwise specified, we focus on the comparative analysis between DBS ($m = 2$) with LPLW in the sequel. Hung's scheme and DBS with distance basis other than $m = 2$ are not discussed due to their higher computational overhead.

In Figs. 4.3a and 4.3c, we plot the ratio of the maximum time t_f of DBS to LPLW when scheduling the same IASPP$_F$ instance. As we can see, DBS ($m = 2$) demonstrates more than 20% reduction in terms of the maximum time to schedule an instance compared to LPLW. In certain cases, e.g., $N = 120$ with perturbed demand update and $N = 80$ with homogeneous demand update, the reduction of the maximum computation time can reach to 80% and 70% respectively. Figs. 4.3b and 4.3d examine the ratio of successfully scheduled instances as a function of the simulation time t for graph size $N = 200$. DBS ($m = 2$) is able to solve all instances around $t = 25$ and $t = 28$ for the perturbed and homogeneous demand update scheme respectively, see Figs. 4.3b and 4.3d. LPLW, on the other hand, schedules all instances approximately at $t = 55$ and $t = 48$ simulation time.

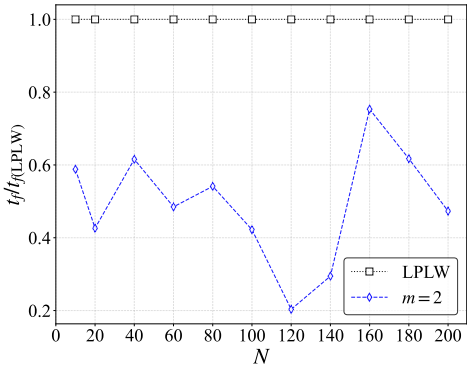
4.5.2. PERFORMANCE OF DBS UNDER HETEROGENEOUS SERVICE DEMAND

This section evaluates the performance of DBS in a network that supports diverse service demands from the end users. We roughly classify the demands into three categories as follows. 1) Mission-critical machine services (machine control loop, interactive haptic control) require extreme low latency in the range of $[100\mu s, 1ms]$. 2) Cloud VR/AR rendering, cloud gaming and V2X communication with low to moderate latency of 1ms to 50ms. 3) Applications such as voice-over-IP, smart metering, that can tolerant E2E latency up to a hundred milliseconds. According to the heterogeneous service demands described above, the demand between each pair of nodes (i, j) is generated uniformly at random from the following three sets:

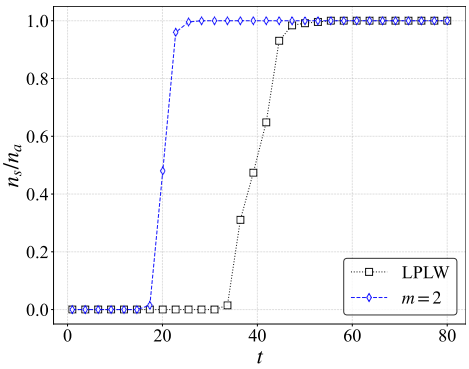
- Set 1: {100μs, 500μs, 1ms, 2ms}
- Set 2: {5ms, 10ms, 15ms, 20ms}
- Set 3: {50ms, 100ms, 500ms, 1s}

Four types of demand matrices D are considered, each parameterized by a different percentage of demand sets. As shown in Table 4.1, in each category, a fixed proportion of

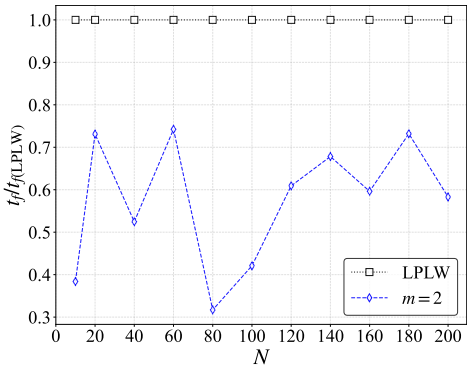
assigned by their demands, resulting in a zero norm.



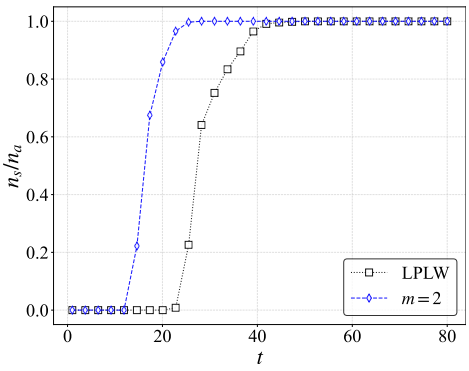
(a) Maximum computation time (perturbed demand)



(b) Ratio of scheduled instance (perturbed demand)



(c) Maximum computation time (homogeneous demand)



(d) Ratio of scheduled instance (homogeneous demand)

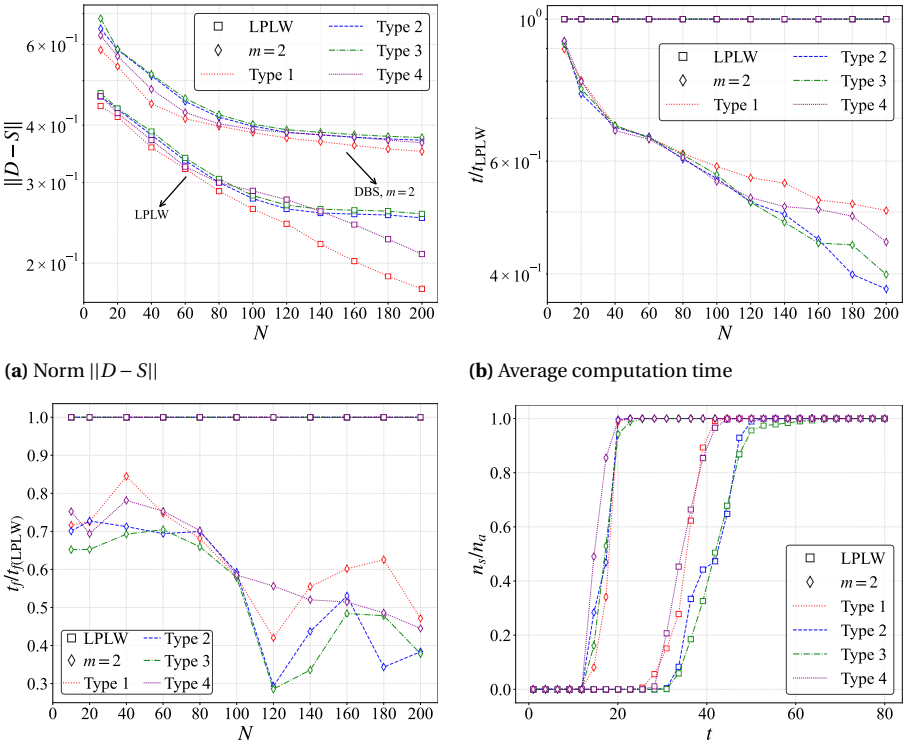
Figure 4.3: Performance of DBS with perturbed demand update in (a) and (b) and homogeneous demand update scheme in (c) and (d). (a) and (c) plot the ratio of the maximum computation time t_f of DBS to LPLW when scheduling the same IASPP$_F$ instance as a function of graph size N. (c) and (d) demonstrate the ratio of successfully scheduled instances $\frac{n_s}{n_a}$ as a function of simulation time t, where n_a is the total number of simulations. The graph size presented in (c) and (d) is $N = 200$. All simulations are repeated $n_a = 1000$ times.

node pairs (i, j) is selected with demand d_{ij} $(i < j)$ randomly drawn from a demand set i $(i = 1,2,3)$. The four types of demand matrices represent different amounts of services that to be accommodated in the network. For example, with demand Type 1, 50% of services require extremely low latency, the remaining 50% of services are less sensitive (demand set 2) or relaxed (demand set 3) to the E2E latency that the network can provide. Since the shortest path weight matrix is a distance matrix, the generated demand matrix D is modified to be a distance matrix, following the rules introduced in [7] and Section 3.2.1).

Together with the generated demand and a minimum spanning tree generated from

Table 4.1: Four types demand matrices

Demand matrix D	Demand set 1	Demand set 2	Demand set 3
Type 1	50%	25%	25%
Type 2	25%	50%	25%
Type 3	25%	25%	50%
Type 4	34%	33%	33%



(a) Norm $||D-S||$

(b) Average computation time

(c) Maximum computation time

(d) Scheduled instance number

Figure 4.4: Performance of DBS ($m = 2$) with heterogeneous demand matrix. Four types of demand matrices, Type 1 to Type 4 as defined in Table 4.1, are evaluated. (a) illustrates the difference between the given demand matrix and the shortest path weight matrix of the graph obtained by LPLW and DBS. The x-axis represents the graph size, and the y-axis is the norm $||D-S||$. In (b), the average computation time of DBS normalized by LPLW is demonstrated as a function of graph size N. (c) presents the maximum computation time t_f normalized by the maximum computation time $t_{f(\text{LPLW})}$ of LPLW as a function of graph size N. (d) plots the number of successful scheduled instances n_s normalized by the simulation number $n_a = 1000$ within a time limit t. n_s is evaluated with graph size $N = 200$. All the simulations in (a)–(d) are repeated 1000 times.

an ER graph as input, we evaluate the performance of DBS with $m = 2$. Similar as Figs. 4.2a and 4.2c, the norm $||D-S||$ obtained by DBS ($m = 2$) in Fig. 4.4a is slightly higher than

that of LPLW. The norm of Type 1 traffic (red dashed line with diamond markers) approximates the results of DBS ($m = 2$) in Fig. 4.2c. This is attributed to the transformation of the demand matrix into a distance matrix in the presence of a high proportion of delay-sensitive traffic (i.e., Type 1 in Table 4.1)[11]. The modified demand matrix of Type 1 traffic in Fig. 4.4a resembles the homogeneous random demand matrix in Fig. 4.2c, thus, results in similar performance of the norm $||D - S||$.

In Figs. 4.4b, 4.4c and 4.4d, we verify the computational efficiency of DBS (with $m = 2$). The results exhibit similar behavior as of the homogeneous service demand update scheme in Section 4.5.1. With graph size $N > 50$, DBS ($m = 2$) achieves approximately 50% reduction in both the average computation time (Fig. 4.4b and the maximum time required to schedule all simulation instances (Fig. 4.4c) compared to LPLW (for all demand types). Fig. 4.4d plots the ratio of successfully scheduled instances as a function of the simulation time t. As we can see, DBS is able to solve all instances around $t = 22$ for all demand types, while LPLW needs about $t = 42$ simulation time to compute the instances for Type 1& 4 traffic, and $t = 60$ for Type 2 and 3 traffic. The similar performance between Types 1 and 4 demands, as well as Types 2 and 3 demands is due to the transformation of the demand matrix described before.

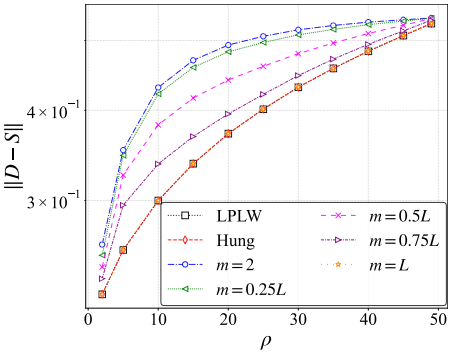
4.5.3. PERFORMANCE OF DBS IN TREES WITH VARYING DIAMETERS

This section evaluates the performance of DBS in different tree graph topologies. Three types of tree graphs, characterized by graph diameters[12] ρ are examined: (i) a star graph of N nodes, with diameter $\rho = 2$; (ii) a path graph of N nodes, with diameter $\rho = N-1$; (iii) a spanning tree generated from ER graphs, with diameter $2 < \rho < N-1$; (iv) a real-world tree topology.

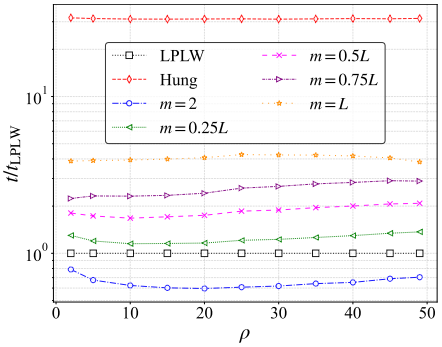
For each IASPP$_F$ instance, we generate a tree graph of $N = 50$ nodes with varying diameter ρ. The link weights are generated uniformly in $[1, 10]$ as integers. The input demand matrix D is obtained as homogeneous demand, in order to evaluate the performance of DBS where the demand matrix does not possess any structural regularity As shown in Fig. 4.5a, LPLW, Hung's method and DBS with $m = L$ produce the same norm $||D - S||$, which is consistent with the observation in Section 4.5.1. The norm $||D - S||$ obtained by LPLW and DBS exhibits distinct behavior in different tree topologies. In a star graph with diameter $\rho = 2$, the demand on each shortest path is determined by maximally two link weights w_{ij}, either from the root to a leaf or between two leaf nodes via the root. Since each shortest path consists of only one or two links, a star graph exhibits minimal mutual dependency (thus, the smallest norm) among shortest paths compared to other tree topologies with the same number of nodes. As the diameter of tree increases, the mutual dependency between shortest path weight s_{ij} increases. This means that the link weight w_{ij} on a path between node pair (i, j) may influence multiple shortest path weights s_{ij}, making it more difficult to approximate the shortest path weight matrix S with the demand matrix D. Hence, a higher norm $||D - S||$ is observed for $2 < \rho < N-1$. In a path graph ($\rho = 49$), DBS shows comparable performance to LPLW,

[11] Recall the triangle inequality discussed in Section 3.2.1,we replace the demand by $d_{ij} = \min_{1 \le k \le N}(d_{ik} + d_{kj})$ and $d_{ji} = d_{ij}$ if the inequality is violated for any node $k \in \mathcal{N}$. Demands from delay-tolerant flows (i.e., from set 2 and set 3) is therefore removed from the demand matrix.

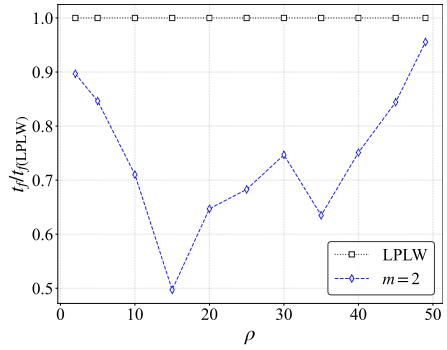
[12] The diameter of a graph G equals the length of the longest shortest path in G.



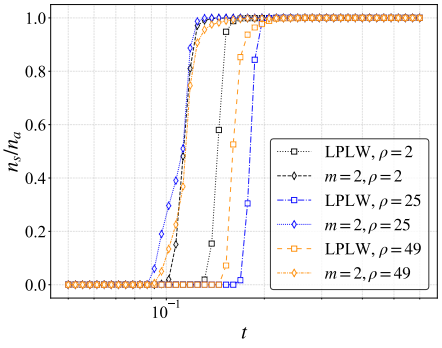
(a) Norm $||D-S||$



(b) Computation time



(c) Maximum computation time



(d) Scheduled instance number

Figure 4.5: Performance of DBS in tree graphs with $N = 50$ nodes and varying diameters ρ, where $\rho = 2$ represents the star graph, $\rho = 49$ is the path graph, and $2 < \rho < 49$ are spanning trees randomly generated from an ER graph with 50 nodes. m denotes the number of distance basis matrices of DBS. (a) illustrates the norm $||D-S||$ as a function of diameter ρ. (b) plots the average computation time of DBS and Hung's method normalized to that of LPLW for the same IASPP$_F$ instance. (c) presents the ratio of the maximum computation time t_f of DBS to LPLW when scheduling the same IASPP$_F$ instance. (d) demonstrates the percentage of successfully scheduled instances n_s across all simulations as a function of time limit t, where n_a denotes the number of simulations. All results are repeated 1000 times.

regardless of the distance basis m as it has the strongest mutual dependencies among all three types of trees.

Analogous to the results presented in Section 4.5.1, DBS with $m = 2$ outperforms LPLW in terms of computation efficiency across all tree topologies. For instance, DBS ($m = 2$) shows approximately 40% reduction in the average computation time relative to LPLW in general graphs (e.g., $10 < \rho < 40$) in Fig. 4.5b. Moreover, as plotted in Fig. 4.5d, LPLW computes faster on star and path graphs than on general trees. In a star graph, the hopcount of any shortest path is either 1 or 2, leading to simple and localized constraints in Eq. (C.2). In a path graph, although many shortest paths share common segments, the overlap is highly localized along the path and the constraints in Eq. (C.2) exhibits a

high degree of nesting (i.e., $\mathscr{P}^*_{i,j} \subset \mathscr{P}^*_{i,j+1}$). This allows LPLW solves the linear programming problem (C.2) efficiently. In contrast, a general tree topology induces complex, non-structured interactions among shortest path constraints. Multiple paths overlap across various subsets of links. The complex structure of constraints then slows down the LPLW [13]. Finally, compared to LPLW, DBS demonstrates consistent superior performance across tree graphs with different diameters in terms of the time to successfully schedule all simulation instances, see Fig. 4.5d.

The performance of DBS is also evaluated on a real-word factory automation network [132, 133]. Details of the network are introduced in Appendix C.6. Compared to LPLW, DBS with $m = 2$ shows approximately 30% reduction in average computation time and 40% reduction in maximum computation time, at the cost of increasing the norm $\|D - S\|$ around 0.08.

In summary, DBS demonstrates consistent superior performance in terms of computational efficiency over LPLW across different IASPP$_F$ instances with varying demands and tree topologies. Increasing the distance basis m in DBS decreases the difference between the obtained shortest path weight matrix and the given demand, at the cost of longer computation time. Compared to LPLW, DBS with $m = 2$ achieves a significant reduction in average computation time by constructing a graph that is close to the optimal solution, i.e., an increase in the norm $||D - S||$.

4.6. CONCLUSION

This work focuses on two extensions of the inverse all shortest path problem (IASPP), i.e., IASPP with minimum adjustment of the sum of link weights (IASPP$_M$) and IASPP given a fixed adjacency matrix (IASPP$_F$). Given a graph G and an arbitrary demand D, IASPP$_M$ addresses the challenge of obtaining a new graph G' with $S = D$ and minimizing changes of link weight. We present the Irreducible Backbone Augmentation (IBA) algorithm to solve IASPP$_M$ exactly. We also prove that the graph G' obtained by IBA minimizes the sum of link weight changes between the resulting weighted adjacency matrix $\widetilde{A}'$ and the input matrix $\widetilde{A}$.

The second extension IASPP$_F$ considers a more demanding scenario where modification of the input adjacency matrix A is not allowed (i.e., the modified graph must have the same adjacency matrix as the original one). Inspiring by practical deployment in Telecommunications and industrial automation networks, where (multiple) spanning trees are constructed to ensure network availability and reliability, we focus on a tree graph when solving IASPP$_F$. A Distance Basis Sum (DBS) algorithm is proposed to reduce the variable and constraint space associated with the IASPP$_F$ framework. Through extensive simulations, we have shown that DBS with distance basis $m = 2$ reduces approximately 30% $\sim$ 50% of the computation time compared to the classical Linear Programming approach, while constructing a graph that is close to the optimal solution.

Both IASPP$_M$ and IASPP$_F$ show promising applications during network planning and network (re-)configuration where accommodating practical constraints (e.g., minimiz-

[13] In our experiments, we employ the simplex method [131], to solve the linear programming problem formulated by LPLW or DBS. The complex structure of the constraints can significantly slow down the simplex method, as it may require a large number of pivot iterations to reach the optimal solution.

ing network downtime and operational cost due to reconfiguration, difficulty of rewiring cables) is of significant relevance. We have demonstrated a potential deployment of IASPP$_M$ and IASPP$_F$ and conclude their applicability in real-world networks such as transportation networks, wireless sensor and actuator networks, Telecommunications and industrial automation networks.

5

Flow Subgraphs and Flow Network Design under End-to-End Power Dissipation Constraints

If we knew what it was we were doing,
it would not be called research,
would it?

Albert Einstein

Though the transport of items following the shortest path in a network has been extensively studied, many practical scenarios involve transmission over multiple paths. In this work, we focus on the transportation in flow networks with the current-flow analogy. We investigate how network topology supports an effective flow between a source node and a destination node and propose a framework to compute the expected number of nodes and links that contributed to transferring items in random networks. We then address how to construct a network given predetermined end-to-end power dissipation, which can be reduced to the "inverse effective resistance problem" that asks for a network whose effective resistance matrix equals a predetermined demand matrix. We propose a heuristic algorithm, "Resistor Gap Pruning" (RGP), which provides sparse networks closely approximating the demand effective resistance and shows stable performance across different demand scenarios.

This chapter is based on a manuscript that is currently under review [134].

5.1. INTRODUCTION

The function of a network is generally related to the transport of items over the underlying graph[5, 6, 8]. Depending on the type of transported items, a network G can be classified as either a "path network" in which packets (e.g., IP packets, vehicles) are propagated, or a "flow network", e.g., electrical networks, gas networks, water networks, etc, in which the transported item is a flow.

In a path network, the transport of items between two nodes (i, j) travels along a single path $\mathscr{P}_{ij}$, defined as a succession of links [4, 6] of the form $\mathscr{P}_{ij} = \{n_1 \sim n_2, n_2 \sim n_3, \ldots, n_{k-1} \sim n_k\}$, where node label $n_1 = i$, $n_k = j$ and $n_a \neq n_b$ for each index a and b. In most practical scenarios, the transport follows the "the shortest path" $\mathscr{P}^*_{ij}$, which minimizes the sum of all link weights along the path. A well-known example is the Open Shortest Path First (OSPF) protocol[4, 9] in data communication networks, whereby routers in an IP network maintain a list of IP destinations, with each IP destination (or range of IP destinations) having the "next hop" associated with it. The next hop is the network connection that shall be followed in order to route an IP message over the shortest path towards the destination. Indeed, the computations, behaviors (such as the hopcount) and applications of the shortest path have been extensively studied in the literature [4, 15, 19, 26, 28, 29, 30, 61, 135, 136, 137].

Many practical scenarios, however, can not be adequately described by the shortest path alone [2, 41, 42, 43, 44]. Multiple alternative paths, or even random paths, are considered when the shortest paths are unknown or unavailable. For example, next-generation 6G communication systems aim to enable information to be divided into smaller units and transmitted simultaneously over diverse and integrated paths, including satellite, fiber-optic and wireless links, to maximize coverage, reliability and transmission efficiency [45, 46]. Other examples include the epidemic spreading [4, 138, 139], where infections can propagate through all available connections. Similarly, the propagation of news or rumors in a social network, which generally does not follow the shortest path from a source node to the destination [42, 140, 141, 142] but rather resembles a random walk process. In cases where the shortest path is not sufficient, the flow network provides a complementary perspective, since the transport propagates proportionally over all possible paths from node i to node j in a flow network.

In this work, we will focus on the flow network using the current-flow analogy. Items are transferred following the principle of current flow through an electrical network, where links are resistors and nodes are junctions between the resistors. We first investigate how the flow transmits through a flow network. Specifically, we study how the topology of a network supports an effective flow between the source node and destination node. We build a framework to compute the expected number of nodes and links that contribute to transferring items in random networks.

The flow over nodes and links in networks also affects the "power dissipation" by the links in the network. Power dissipation is taking here an analogy to electrical networks, where the power dissipation in a connection is determined by the current through that connection and the resistance of that connection. The investigation into the power dissipation is inspired by communication networks, where higher data transmission through a link generally leads to higher cost for that connection [143, 144]; the higher cost may be installation cost, e.g., fibre optic cable, as well as operational cost, e.g., energy consump-

tion of routers or repeaters. Similar examples can be observed in social networks [145, 146] and transportation networks [147, 148]. We demonstrate that transferring items in accordance with the "resistance" of each path between the source node and the destination node provides a more evenly spread load on the links in the network and may lead to a reduction in power consumption. The power dissipated on each link in ER graphs holds a power law decay with respect to the network size and link density. Finally, we propose a heuristic algorithm for constructing networks such that the total power dissipation of the flow network approximately equals a predetermined demand with different source and destination node pairs.

The outline of the paper is as follows: In Section 5.2, we first introduce terminology and background knowledge on flow networks. In Section 5.3, we investigate the links and nodes that contribute to item transmission in flow networks. A framework is proposed to compute the expected number of nodes and links that contributed to transferring items in random networks. In Section 5.4, we consider the inverse problem of power dissipation, which asks for constructing a network given predetermined demands on the end-to-end power dissipation. A heuristic approach named Resistor Gap Pruning (RGP) is proposed and tested for different demand matrices. Finally, we summarize our results in Section 5.5.

5.2. TERMINOLOGY

In this chapter, we introduce the terminology used in this paper. A summary notation list is presented in Appendix D.1.

5.2.1. GRAPH

In this work, we limit ourselves to connected, undirected, simple graphs[1] [6]. We introduce an $N \times N$ adjacency matrix A to represent a graph, with element $a_{ij} = 1$ if there is a link between node i and node j, otherwise $a_{ij} = 0$. The link between nodes i and j is represented by $l = i \sim j$. Each link $l \in \mathscr{L}$ has a positive weight w_l and the $N \times N$ link weight matrix is denoted by W. The $N \times N$ weighted adjacency matrix is defined as $\widetilde{A} = W \circ A$, where the Hadamard product $\circ$ defines a direct elementwise multiplication $\widetilde{a}_{ij} = w_{ij} a_{ij}$ and the "tilde" refers to a weighted graph matrix. In our setting, $\widetilde{a}_{ij} = 0$ means that there is no link between nodes i and j, because we exclude zero link weights, i.e. $w_{ij} > 0$, in order to avoid the complication that a zero weight $w_{ij} = 0$ would physically mean that nodes i and j are the same.

By the definition of the adjacency matrix, the degree d_i of node i equals to the row sum of A, i.e. $d_i = \sum_{k=1}^{N} a_{ik}$, which represents the number of neighbors of node i. The excess degree d_i' of node i is defined [1, 149] as the number of remaining links after reaching node i by a random link l. In other words, for node i with degree d_i, the excess degree $d_i' = d_i - 1$, like the example shown in Fig. 5.1. Suppose a graph has degree distribution $\Pr[D = k]$. The degree probability generating function (pgf) is defined as

$$\varphi_D(z) = E[z^D] = \sum_k \Pr[D = k]\, z^k,$$

[1] A simple graph has no multiple links between a same pair of nodes and the graph does not contain any self-loops, i.e. $a_{ii} = 0$ for each node $i \in \mathscr{N}$.

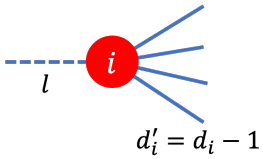


Figure 5.1: Illustration of the excess degree d_i'. When node i is reached by following a randomly chosen link l, the remaining number of neighbors of node i is $d_i' = d_i - 1$.

which encodes the full distribution of D into an analytic function [4, Chapter 2]. The excess degree distribution [1] is then

$$\Pr\left[D' = k\right] = \frac{k+1}{E[D]} \Pr\left[D = k+1\right],$$

where $E[D]$ is the expected degree of the graph. The pgf of the distribution of excess degree D' is

$$\varphi_{D'}(z) = E\left[z^{D'}\right] = \sum_k \Pr[D' = k] z^k = \frac{\varphi_D'(z)}{\varphi_D'(1)}$$

A directed graph can also be represented by an $N \times L$ incidence matrix B with elements

$$b_{il} = \begin{cases} 1 & \text{if link } l = i \rightarrow j \\ -1 & \text{if link } l = j \rightarrow i \\ 0 & \text{otherwise} \end{cases}$$

where link $l = i \rightarrow j$ denotes the direction from node i to node j. The sum of the columns in an incidence matrix equals zero, i.e. $u^T B = 0$, where u represents the $N \times 1$ all-one vector. An undirected graph can be represented by an $N \times (2L)$ incidence matrix, where each link $i \sim j$ is counted twice, once for direction $i \rightarrow j$ and once for direction $j \rightarrow i$. In that case, the degree of each node is just doubled [6].

For an undirected graph, the $N \times N$ Laplacian matrix Q reveals the relation [6] between the adjacency matrix A and the $N \times L$ incidence matrix B:

$$Q = BB^T = \Delta - A \tag{5.1}$$

where the $N \times N$ degree diagonal matrix $\Delta = \text{diag}(d)$ contains node degrees on its main diagonal and the $N \times 1$ degree vector $d = Au$ contains the degree of each node, with d_i denoting the node i degree. Details of the Laplacian and Pseudoinverse Laplacian are provided in Section 3.1.1, [6] and [47].

5.2.2. ELECTRICAL RESISTOR NETWORK AND FLOW SUBGRAPH

In this work, the flow network is modeled analogously to the electrical resistor network. Consider a weighted electrical resistor network G with N nodes and L links. Each link $l = i \sim j$ between the nodes i and j possesses a resistor with resistance $r_l = r_{ij} > 0$ and the weight of link $w_l = w_{ij} = \frac{1}{r_l}$. Let v_k denote the potential or voltage potential of node k in the network circuit and the current $y_l = y_{ij}$ through the resistor of link l between

node i and node j, which is directed (even in undirected graphs) so that $y_{ij} = -y_{ji}$. According to the law of Ohm $v_i - v_j = r_{ij} y_{ij}$, the voltage and current are related by

$$y = \text{diag}\left(\frac{1}{r_l}\right) B^T v, \tag{5.2}$$

where y is the $L \times 1$ link current vector, v is the $N \times 1$ vector with nodal voltages and the $L \times L$ matrix $\text{diag}(\frac{1}{r_l})$ has diagonal elements $(\frac{1}{r_1}, \ldots, \frac{1}{r_2}, \ldots, \frac{1}{r_L})$ where $r_l = r_{ij}$ is the resistance of link $l = i \sim j$.

Define x_i as the external current injected into node i. If the current leaves from node i, then $x_i < 0$. The Kirchhoff's current law shows that

$$x = By, \tag{5.3}$$

where x is the $N \times 1$ node external current vector containing the external current injected into each node x_i. Substituting Eq. (5.2) into the Kirchhoff's conservation law (5.3), the external current and voltage are related by

$$x = B\text{diag}(\frac{1}{r_{ij}}) B^T v = \widetilde{Q} v \tag{5.4}$$

where the weighted Laplacian

$$\widetilde{Q} = B\text{diag}(\frac{1}{r_{ij}}) B^T \tag{5.5}$$

is defined similarly to the unweighted Laplacian decomposition in Eq. (5.1).

Though matrix relation $x = \widetilde{Q} v$ (5.4) cannot be inverted due to $\det \widetilde{Q} = 0$, Van Mieghem [6] shows that

$$v = \widetilde{Q}^{\dagger} x, \tag{5.6}$$

by choosing the average voltage equaling 0 in the network. Substituting the voltage v in Eq. (5.2) with Eq. (5.6), we couple the external current and current on each link:

$$y = Cx, \tag{5.7}$$

where the $L \times N$ matrix $C = \text{diag}\left(\frac{1}{r_l}\right) B^T \widetilde{Q}^{\dagger}$.

Inject a unit current $I_c = 1$ Ampere at node (source) s and the current flows out of the network G at node (destination) t, the current flow through links l are then

$$y = C(e_s - e_t) \tag{5.8}$$

We define the flow subgraph G^*_{st} as the subgraph through which the current I_c propagates. Specifically, the flow subgraph G^*_{st} comprises all links with nonzero current $y_l \neq 0$ and nodes adjacent to those links.

In an electrical resistor network G, each link $l = i \sim j$ possesses a resistor with resistance $r_{ij} = \frac{1}{w_l}$. Ohm's law $v_i - v_j = r_{ij} y_{ij}$ states that the resistance is the proportionality constant or ratio between the potential differences $v_i - v_j$ and the current I_c injected at node i and leaving at node j. The ratio $\frac{v_i - v_j}{I_c}$ thus measures the resistance of a subgraph over which the injected current I_c in node i spreads towards node j and w_{ij} is

called "effective" resistance [6] between node i and node j. The details of the effective resistance are introduced in [6, 47] and Section 3.1.2. Here, we recall upper bound of the effective resistance [8]. The effective resistance ω_{ij} between two adjacent nodes i and j (i.e. $a_{ij} = 1$), represents the effective resistance of a parallel connection

$$\frac{1}{\omega_{ij}} = \frac{1}{r_{ij}} + \frac{1}{(\omega_{G'})_{ij}} \tag{5.9}$$

between the resistance r_{ij} of a link $l = i \sim j$ and the effective resistance $(\omega_{G'})_{ij}$ between nodes i and j in the graph $G' = G \setminus l$, where the link $l = i \sim j$ is removed. The effective resistance ω_{ij} between adjacent nodes i and j is upper bounded by the resistance r_{ij} of the direct link between them

$$\omega_{ij} = \frac{r_{ij}\,(\omega_{G'})_{ij}}{r_{ij} + (\omega_{G'})_{ij}} \leq \min\Big(r_{ij}, (\omega_{G'})_{ij}\Big)$$

Otherwise, if $a_{ij} = 0$, then the effective resistance ω_{ij} is upper bounded by the sum of resistances of links forming the shortest path between the nodes. In both cases, if more paths exist connecting two nodes, then there are more possible paths for the current to flow simultaneously and thus, the effective resistance lowers.

Inject a current I_c in a network from source node s and let it out from destination node t. The power P_l, the energy per unit time (in watts), dissipated on each link is defined as $P_l = y_l^2 r_l$. The power dissipation P_G in the network is the sum of the power dissipated on each link,

$$P_G = \sum_{l \in \mathscr{L}} P_l = \sum_{l \in \mathscr{L}} y_l^2 r_l = I_c^2 \omega_{st}, \tag{5.10}$$

which also equals the product of the square of the current and the effective resistance.

5.3. SIZE OF THE FLOW SUBGRAPH

Given a flow network G and a pair of nodes (i, j), the items transported on the network only propagate along the flow subgraph G^*_{ij}, Figure 5.2. Indeed, the number of nodes and links in a flow subgraph is generally determined by the network topology. For example, for a tree network, the flow subgraph G^*_{ij} between nodes (i, j) is the same as the corresponding (shortest) path $\mathscr{P}^*_{ij}$. In a 2-d lattice graph G, the flow subgraphs G^*_{ij} are identical to the graph G for every pair of nodes (i, j). In this section, we propose a framework "self-consistent approach" to compute the expected number of nodes and links comprising the flow subgraphs G^*_{ij}. The main idea of the self-consistent approach is inspired by the property:

Property 3. *A node k, $k \neq i, j$, in the flow subgraph G^*_{ij} between nodes (i, j) has at least two neighbors belonging to the flow subgraph G^*_{ij}.*

Proof. For an arbitrary node k that belongs to the flow subgraph G^*_{ij}, the current I_k flowing through node k is not zero, whereas nodes outside G^*_{ij} carry no current. With Kirchhoff's current law, the total current flowing into node k equals the total current leaving node k. Hence, node k must have at least two neighbors belonging to the flow subgraph G^*_{ij}. □

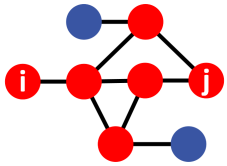


Figure 5.2: A schematic representation of the flow subgraph G^*_{ij}, which is highlighted by red, between source node i and destination node j.

The self-consistent approach can be applied to all random graphs with a known probability generating function (pgf) of node degree D. In Section 5.3.1, we present an analysis using Erdős–Rényi (ER) random graphs as an illustrative example.

5.3.1. NODE NUMBER OF THE FLOW SUBGRAPH IN ERDŐS–RÉNYI (ER) RANDOM GRAPHS

We first analyze how the flow subgraph G^*_{ij} is composed. For ER graph $G_p(N)$ with a small expected degree $E[D] < 1$, all connected components are small [1, 150] with size of order $O(\log(N))$. Hence, in this regime, the relative size of the flow subgraph $\rho_N = |G^*_{ij}|/N$ is also negligible, where $|G^*_{ij}|$ denotes the number of nodes in the flow subgraph G^*_{ij}. As the expected degree increases, a giant component (GC) emerges [1]. Outside the GC, all connected components have a vanishing relative size and thus the relative flow subgraph size ρ_N remains negligible when either (or both) i and j lie outside the GC. Therefore, the average size of the flow subgraph G^*_{ij} is dominated by node pairs (i, j) that both belong to the GC.

We then decompose the GC into a backbone subgraph $\mathscr{B}$ and branches.

1. Backbone $\mathscr{B}$: The maximal induced subgraph of the (giant) component in which every node has at least two neighbors within $\mathscr{B}$ (nodes highlighted with green outlines in Fig. 5.3). Denote by $b = |\mathscr{B}|/N$ the relative size of the backbone $\mathscr{B}$, where $|\mathscr{B}|$ denotes the cardinality of the set $\mathscr{B}$.

2. Branches: the connected components of $U := GC \setminus \mathscr{B}$ (nodes without green outlines in Fig. 5.3). Each branch T_k (with index $k = 1, \ldots, M$) is a finite subgraph attached to $\mathscr{B}$ at a single node, where M denotes the total number of branches. Let $\theta := |U|/N$ denote the total relative size of the union of all branches.

RELATIVE SIZE OF THE BACKBONE

Consider a flow subgraph G^*_{ij} with source node i and destination j. A node either belongs to the backbone $\mathscr{B}$ or to its complement $\mathscr{B}^c$ with probability b and $1-b$, respectively. By definition, any node m, different from node i or j, can be in $\mathscr{B}$ only if it has at least two neighbors that also lie within the backbone $\mathscr{B}$.

Let $\Pr[D = k]$ denote the degree distribution of the graph, the pgf of the excess-degree distribution (see details in Section 5.2.1) is then given by $\varphi_{D'}(z) = \varphi'_D(z)/\varphi'_D(1)$, where $\varphi_D(z) = \sum_{k\geq 0} \Pr[D = k]\, z^k$ is the degree pgf of graph $G_p(N)$.

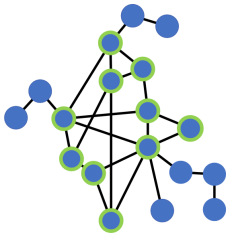


Figure 5.3: Backbone–branch decomposition: Backbone nodes (each node has ≥ 2 neighbors within the backbone) are highlighted with green outlines; other nodes are branches (finite subgraph) attached to the backbone.

Follow a uniformly random chosen link to one of the endpoints and remove the link. Let p_b be the probability that the reached node m is "supported" by at least one of its remaining neighbors, i.e., the reached node has at least one neighbor that also belongs to the backbone $\mathscr{B}$. For each remaining neighbor node n of node m, the probability that node n can *not* support node m is $1-p_b$. Independence across the remaining neighbors yields the self-consistent equation

$$\begin{aligned} p_b &= 1 - \Pr\left[\text{No remaining neighbor belongs to } \mathscr{B}\right] \\ &= 1 - \sum_k \Pr[D' = k](1-p_b)^k \\ &= 1 - \varphi_{D'}(1-p_b) \end{aligned} \tag{5.11}$$

Eq. (5.11) indicates that the probability p_b is numerically identical to the probability that a node belongs to the GC of a random graph [3, 149, 151].

Given the probability p_b and a node w with degree $D = k$, let X denote the number of supporting neighbors of w, i.e., neighbors that also belong to the backbone $\mathscr{B}$. Conditioned on $D = k$, the variable X follows a binomial distribution and

$$\Pr[X = j \mid D = k] = \binom{k}{j} p_b^j (1-p_b)^{k-j}$$

Since a node belonging to the backbone $\mathscr{B}$ has at least two supporting neighbors,

$$\begin{aligned} &\Pr[w \in \mathscr{B} \mid D = k] \\ &= 1 - \Pr[X = 0 \mid D = k] - \Pr[X = 1 \mid D = k] \\ &= 1 - (1-p_b)^k - k p_b (1-p_b)^{k-1} \end{aligned} \tag{5.12}$$

Applying the law of total probability [4], the overall fraction b of nodes that have at least

two supporting neighbors and thus belong to the backbone $\mathscr{B}$ is

$$\begin{aligned} b &= \sum_{k=1}^{N-1} \Pr[D=k]\,\Pr[w \in \mathscr{B} \mid D=k] \\ &= \sum_{k=1}^{N-1} \Pr[D=k]\left[1-(1-p_b)^k - k p_b (1-p_b)^{k-1}\right] \\ &= 1-\varphi_D(1-p_b) - p_b \varphi_{D'}(1-p_b) \end{aligned} \tag{5.13}$$

For an ER graph $G_p(N)$ with mean degree $E[D] = (N-1)p = \lambda$, the degree distribution is binomial [4] with probability generating function

$$\varphi_D^{(\mathrm{Bin})}(z) = (1-p(1-z))^{N-1}$$

In the sparse regime where $N \to \infty$ and λ is fixed, this pgf admits the expansion [4]

$$\varphi_D^{(\mathrm{Bin})}(z) = e^{-\lambda(1-z)}\left(1 - \frac{\lambda^2(1-z)^2}{2(N-1)} + O\left(\frac{1}{N^2}\right)\right)$$

In the sparse limit $N \to \infty$ with fixed mean degree $E[D] = \lambda$, the degree and excess–degree probability generating functions of an ER graph are given by $\varphi_D(z) = (1-p(1-z))^{N-1}$ and $\varphi_{D'}(z) = (1-p(1-z))^{N-2}$, respectively, which admit the binomial–to–Poisson expansion $\varphi_D(z), \varphi_{D'}(z) = e^{-\lambda(1-z)} + O(1/N)$. Substituting this expansion into the general self-consistency equation (5.11) yields

$$p_b = 1 - e^{-E[D]\,p_b}, \tag{5.14}$$

where the omitted terms introduce corrections of order $O(1/N)$ for ER graphs.

Eq. (5.14) can be solved in closed form. Rewrite

$$p_b = 1 - e^{-E[D]p_b} \quad \Longleftrightarrow \quad 1 - p_b = e^{-E[D]p_b}.$$

Let $y = E[D](1-p_b)$. Then

$$(-y)e^{-y} = -E[D]\,e^{-E[D]} \tag{5.15}$$

Here $W(x)$ denotes the Lambert–W function [152], defined implicitly by $W(x)e^{W(x)} = x$. Eq. (5.15) is of the canonical form $ue^u = x$ with $u = -y$ and $x = -E[D]e^{-E[D]}$, and therefore yields

$$p_b = 1 + \frac{1}{E[D]}\,W_0\big(-E[D]\,e^{-E[D]}\big), \tag{5.16}$$

where W_0 is the principal branch of the Lambert–W function [152]. For $E[D] \geq 0$, this branch gives the unique solution satisfying $p_b \in [0,1]$, while the remaining real branch leads to non-physical solutions and is discarded.

Using the same Poisson approximation in the backbone-size expression (5.13) gives

$$b \;=\; 1 - e^{-E[D]\,p_b}\big(1 + E[D]\,p_b\big), \tag{5.17}$$

while the binomial-to-Poisson expansion above implies that the exact ER value of b differs from (5.17) by a relative error of order $O(1/N)$ in the sparse regime.

Eq. (5.14) always has the trivial solution $p_b = 0$; for the expected degree $E[D] \leq 1$, the trivial solution is the only fixed point, which implies $b = 0$ from (5.17). When the expected degree $E[D] > 1$, a positive fixed point $p_b > 0$ appears and increases with $E[D]$, and the backbone fraction b given by (5.17) grows accordingly.

RELATIVE SIZE OF THE BRANCHES

For the the union of all branches $U = GC \setminus \mathscr{B}$, the relative size $\theta = \frac{|U|}{N} = p_b - b$, while the relative size of each branch T_k is defined as $\tau_k = \frac{|T_k|}{N}$, where $|T_k|$ is the size of a single branch.

For the ER graph or a configuration model with fixed degree distribution possessing a finite second moment, each branch can be regarded as a subcritical Galton–Watson tree [1, 149, 151] with offspring mean, denoted by $\xi < 1$. Let p_T denote the extinction probability of the corresponding Galton–Watson process and p_T satisfies $p_T = \varphi_{D'}(p_T)$, where $\varphi_{D'}(z)$ is the pgf of the excess-degree distribution of the graph. The average offspring [1, 149, 151] is then given by $\xi = (1 - p_T)\varphi'_{D'}(p_T)$. In the specific case of an ER graph, the average offspring specializes to $\xi = E[D]p_T(1 - p_T) < 1$.

Since a single branch behaves as a subcritical Galton–Watson tree, the expected size of the branch T_k is constant and independent of the size of the graph N: $E[|T_k|] = \frac{1}{1-\xi}$. The branch size distribution has an exponentially decaying tail [1, 149, 151], implying that $|T_k| = O(1)$ and therefore

$$\tau_k = \frac{|T_k|}{N} = O\left(\frac{1}{N}\right)$$

Since the total branch mass is $|U| = \theta N$ and a single branch has $E[|T_k|] = \frac{1}{1-\xi}$, the expected number of branches is $E[M] \approx \frac{|U|}{E[|T_k|]} = \theta N(1 - \xi)$.

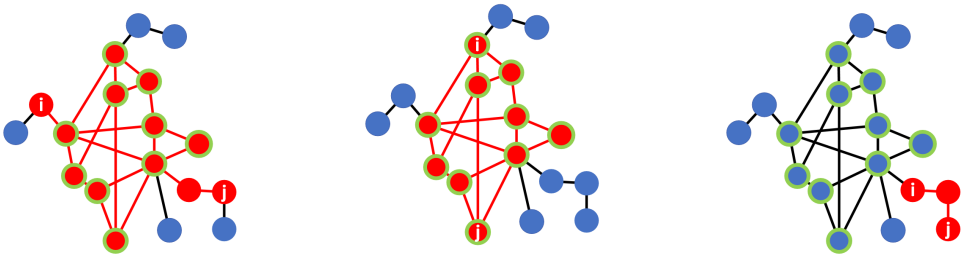
RELATIVE SIZE OF THE FLOW SUBGRAPH

Conditional on source-destination node pair $(i, j) \in$ GC, three different cases are considered in Fig. 5.4:

1. Case (i): If node i and j lie in different branches (or one is in a branch and the other in the backbone $\mathscr{B}$), then this case occurs with probability $O(1)$, since the total relative size of all branches is $O(1)$ whereas each individual branch has $O(1)$ size. Hence, two random GC nodes fall in different branches (or one in a branch and one in $\mathscr{B}$) with a non-vanishing probability as $N \to \infty$. The normalized coverage within the backbone converges to b, while the branch parts of the flow subgraph G^*_{ij} contain only $O(1)$ nodes and therefore contribute $O(1/N)$ after normalization. This follows from the fact that each branch behaves as a subcritical Galton–Watson tree whose expected size is bounded and independent of N (see discussion above).

2. Case (ii): If both node i and j lie within the backbone $\mathscr{B}$, then this event also has probability $O(1)$ under the same conditioning. In this case, the flow remains entirely inside the backbone, and the normalized size of the visited backbone portion converges to b.

3. Case (iii): If both node i and j belong to the same branch, then the flow subgraph G^*_{ij} is contained inside a single finite tree. Since a branch is a subcritical Galton–Watson tree of constant expected size (independent of N), we have the size of flow subgraph $|G^*_{ij}| = O(1)$ and therefore the relative size of the flow subgraph $|G^*_{ij}|/N = O(1/N)$. The probability of this event is $O(1/N)$, so its contribution to the expected normalized size of the flow subgraph is $O(1/N) \cdot O(1/N) = O(1/N^2)$, which is negligible in expectation.



(a) Case (i) **(b)** Case (ii) **(c)** Case (iii)

Figure 5.4: Toy examples for different source–destination pairs. Nodes i and j are the source and destination of the flow. Red nodes and links indicate the components of the flow subgraph. Backbone nodes are highlighted with green node outlines.

In Section 5.3.1, we have shown that the probability that a randomly chosen node i belongs to the giant component GC is $\Pr[i \in \mathrm{GC}] = p_b$ and therefore the probability that both nodes i and j belong to the GC is $\Pr[i,j \in \mathrm{GC}] = p_b^2$. Conditioning on this event, Cases (i) and (ii) imply that the normalized size of the flow subgraph restricted to the GC satisfies

$$E\left[\frac{|G^*_{ij}|}{N}\,\middle|\, i,j \in \mathrm{GC}\right] = b + O\left(\frac{1}{N}\right),$$

where the $O(1/N)$ term comes from the branches on the GC. Thus,

$$E\left[\frac{|G^*_{ij}|}{N}\,\middle|\, i,j \in \mathrm{GC}\right] p_b^2 = b\,p_b^2 + O\left(\frac{1}{N}\right)$$

Next, nodes outside the GC occur with probability $1-p_b^2$ and generate a flow subgraph of size $O(1)$, since every connected component outside the GC is a finite tree of bounded size. Their total contribution to $E[\rho_N]$ is therefore also $O(1/N)$. Consequently, the expected normalized size of the flow subgraph satisfies

$$E[\rho_N] = b\,p_b^2 + O\left(\frac{1}{N}\right) \tag{5.18}$$

For ER graph $G_p(N)$ with expected degree $E[D] = Np$, the probability p_b solves $p_b = 1 - e^{-E[D]p_b}$ and the relative backbone size is $b = 1 - e^{-E[D]p_b}(1 + E[D]p_b)$.

Fig. 5.5 shows that when the expected degree $E[D]$ is small (e.g., $E[D] = 0.5$), the two curves $y = p$ and $y = 1 - e^{-E[D]p}$ intersect only at $p = 0$, Eq. (5.14) has only the solution 0, so the normalized size b of the backbone $\mathscr{B}$ is zero. When the expected degree $E[D]$ exceeds the critical value $(E[D])_c = 1$, a second intersection emerges, indicating that a giant flow subgraph starts to form. As the expected degree $E[D]$ increases further (e.g., $E[D] = 2{,}5$), the fixed point p and the corresponding size b both grow and eventually approach 1.

Fig. 5.6 compares our analytical prediction in (5.18) with simulations on ER graphs. For small network sizes N, a noticeable gap appears between the theoretical curve and

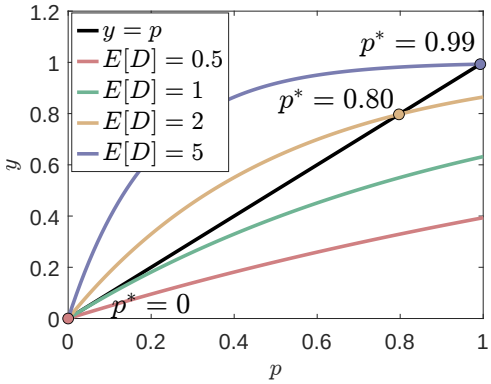
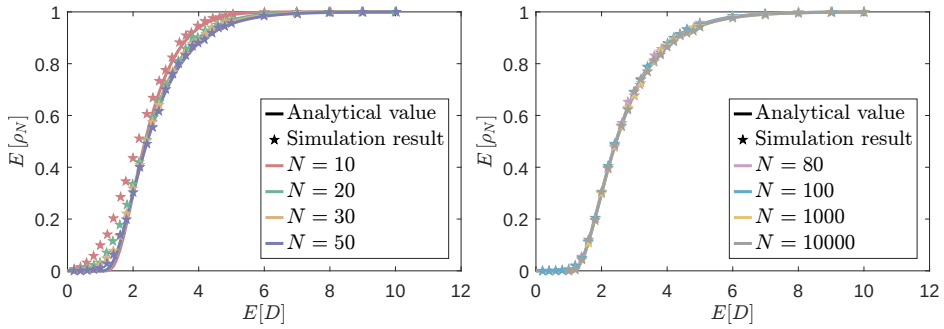


Figure 5.5: Intersections of $y = p$ (black) and $y = 1 - e^{-E[D]p}$ (colored) for different expected degrees $E[D]$. When the expected degree $E[D] \leq 1$, only the trivial solution $p = 0$ exists. For $E[D] > 1$, a non-trivial fixed point appears and increases with $E[D]$.

5

the simulation results, which is consistent with the $O(1/N)$ finite-size deviation established in our analysis. As N increases, this finite-size effect diminishes, and the analytical prediction in (5.18) converges to the simulation results. When the graph is sufficiently large, the $O(1/N)$ correction becomes negligible, and the agreement between theory and simulation is nearly exact.



(a) N from 10 to 50 **(b)** N from 80 to 10000

Figure 5.6: The average relative size $E[\rho_N]$ of the flow subgraph as a function of the expected average degree $E[D]$ in ER graphs with different numbers of nodes. The analytical results in (5.18) are derived using the binomial degree distribution of the ER graph.

5.3.2. NUMBER OF LINKS OF THE FLOW SUBGRAPH

A link belongs to the flow subgraph G^*_{ij} only if the link contributes to the current flowing from the source node i to the destination node j. Consider an undirected link $l = m \sim n$ in a weighted or unweighted graph and let v_m and v_n denote the electrical potentials of nodes m and n, respectively. By Ohm's law, the current flowing through link l is propor-

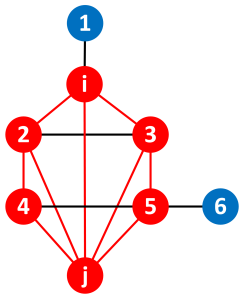
tional to the potential difference $v_m - v_n$. Hence,

$$y_l \neq 0 \iff v_m \neq v_n$$

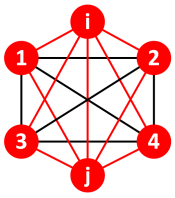
The node set $\mathcal{N}(G^*_{ij})$ of the flow subgraph G^*_{ij} consists exactly of those nodes through which the injected current propagates, that is, nodes with non-zero total current, i.e., $y_m = \sum_{n \in N_G(m)} y_{mn} > 0$, where $\mathcal{N}_G(m)$ denotes the set of all neighbours of node m. For any node $m \notin \mathcal{N}(G^*_{ij})$, the total current $y_m = 0$ and Kirchhoff's current law then implies that all its incident link currents must be zero: $y_{mn} = 0$ for $n \in \mathcal{N}_G(m)$. Hence, no link adjacent to a node outside the node set of the flow subgraph $\mathcal{N}(G^*_{ij})$ can belong to the flow subgraph. Every link in the flow subgraph G^*_{ij} must have both endpoints in $\mathcal{N}(G^*_{ij})$. Among the links belonging to G^*_{ij}, the membership condition reduces to

$$l = m \sim n \in \mathcal{L}(G^*_{ij}) \iff y_l \neq 0 \iff v_m \neq v_n$$

Therefore, when identifying the links in the flow subgraph, we only need to consider links whose endpoints both lie in the node set of the flow subgraph $\mathcal{N}(G^*_{ij})$ and have different electrical potentials.



(a) Flow subgraph G^*_{ij} with $N = 6$ nodes and $L = 9$ links.



(b) Flow subgraph G^*_{ij} of a complete graph with $N = 6$ nodes.

Figure 5.7: Schematic illustration of links in flow subgraphs of two unweighted toy graphs. The flow subgraph is highlighted in red. In (a), nodes 2, 3, 4, and 5 belong to the flow subgraph. Links $1 \sim i$ and $5 \sim 6$ are not included because nodes 1 and 6 do not belong to the flow subgraph. Links $2 \sim 3$ and $4 \sim 5$ are also excluded, since the node pairs $(2,3)$ and $(4,5)$ have equal potentials. In (b), all links between nodes 1, 2, 3, and 4 are not flow subgraph links.

We define the set of *candidate* links connecting nodes in the flow subgraph G^*_{ij},

$$\mathcal{L}_{\mathrm{FS}}(G^*_{ij}) := \{l = m \sim n \in \mathcal{L}(G) \mid m, n \in \mathcal{N}(G^*_{ij})\},$$

Since a node pair (m, n) belonging to the flow subgraph may have the same potential, a link $l = m \sim n$ may belong to $\mathcal{L}_{\mathrm{FS}}(G^*_{ij})$ but does not belong to $\mathcal{L}(G^*_{ij})$, Fig. 5.7a. Thus, the link set of the flow subgraph satisfies

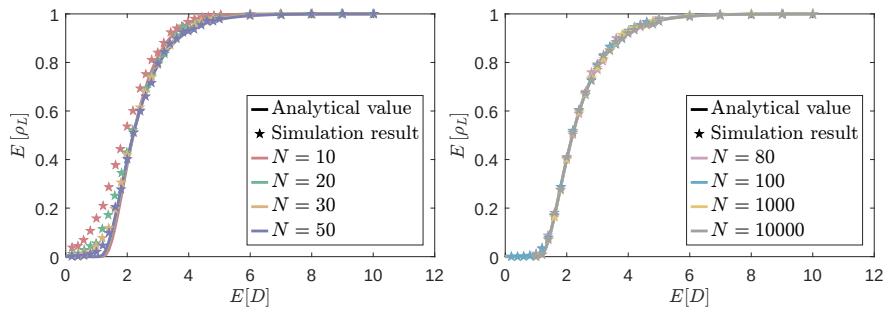
$$\mathcal{L}(G^*_{ij}) \subseteq \mathcal{L}_{\mathrm{FS}}(G^*_{ij}) \tag{5.19}$$

In weighted ER graphs with randomly generated link weights (e.g., the link weights are uniformly distributed), the event that link $l = m \sim n$ belongs to $\mathscr{L}_{\text{FS}}(G^*_{ij})$ but does not belong to $\mathscr{L}(G^*_{ij})$ has a probability zero, since node potentials are continuous functions of the independent link weights. Hence, $\mathscr{L}(G^*_{ij}) = \mathscr{L}_{\text{FS}}(G^*_{ij})$ almost surely. In unweighted ER graphs, equipotential nodes may occur only when two nodes are structurally indistinguishable, i.e., when they lie in the same orbit of a graph automorphism [153]. For ER graphs with moderate expected degree, such nontrivial automorphisms occur [154] with probability $O(1/N)$, giving

$$|\mathscr{L}_{\text{FS}}(G^*_{ij})| - |\mathscr{L}(G^*_{ij})| = O(1), \quad \frac{|\mathscr{L}(G^*_{ij})|}{|\mathscr{L}_{\text{FS}}(G^*_{ij})|} = 1 - O(1/N)$$

However, when the expected degree $E[D]$ becomes very large (the graph approaches a dense or even complete graph), many nodes acquire almost identical neighborhoods, making them structurally indistinguishable. The number of such equipotential node pairs increase rapidly with p, and a positive fraction of links in $\mathscr{L}_{\text{FS}}(G^*_{ij})$ becomes degenerate (zero potential drop). Consequently, the deviation $|\mathscr{L}_{\text{FS}}(G^*_{ij})| - |\mathscr{L}(G^*_{ij})|$ is no longer $O(1)$ but grows proportionally to the number of these symmetric node pairs.

In the extreme case where the graph becomes a complete graph K_N (as in Fig. 5.7b), all non-terminal nodes have exactly the same neighbors and therefore attain the same electrical potential under a unit $i \to j$ current injection. Hence every link whose endpoints do not include i or j has zero potential difference and carries no current. The only links with nonzero potential differences are: (i) the link $l = i \sim j$ and (ii) the $2(N-2)$ links on the two-hop paths $\mathscr{P}_{ij} = \{i \sim k, k \sim j\}$, k is different from node i and j. Thus, almost all links of K_N lie outside the flow subgraph.



(a) N from 10 to 50 **(b)** N from 80 to 10000

Figure 5.8: The average link fraction ρ_L of the flow subgraph as a function of the expected average degree $E[D]$ in weighted ER graphs with different numbers of nodes.

In Section 5.3.1, we have shown that as the size of the graph N increases, for a fraction $1 - p_b^2$ of source-destination node pairs (i, j), which lie outside GC, the relative size ρ_N of the flow subgraph satisfies

$$\rho_N = \frac{|G^*_{ij}|}{N} \longrightarrow 0,$$

which further indicates that the expected fraction $\frac{|\mathscr{L}_{\mathrm{FS}}(G^*_{ij})|}{L}$ of candidate links $\mathscr{L}_{\mathrm{FS}}(G^*_{ij})$, normalized by the total number of links L of the graph, is 0 in this case. For the remaining fraction p_b^2 of node pairs, the relative size of the flow subgraph approaches the size of the backbone $\mathscr{B}$, i.e.,

$$\rho_N \longrightarrow \frac{|\mathscr{B}|}{N}$$

Following one endpoint of a uniformly chosen link, the probability that the reached node belongs to (i.e., is supported within) the backbone $\mathscr{B}$ is p_b. Hence, a link belongs to the candidate link set $\mathscr{L}_{\mathrm{FS}}(G^*_{ij})$ with probability p_b^2. In other words, the expected fraction $\frac{|\mathscr{L}_{\mathrm{FS}}(G^*_{ij})|}{L}$ of candidate links $\mathscr{L}_{\mathrm{FS}}(G^*_{ij})$, normalized by the total number of links L of the graph, is p_b^2 in this case.

Averaging over all source–destination node pairs gives the overall expected fraction of links in the flow subgraph as

$$\begin{aligned} E[\rho_L] &= E\left[\frac{|\mathscr{L}_{\mathrm{FS}}(G^*_{ij})|}{L}\right] + O\left(\frac{1}{N}\right) \\ &= p_b^2 \times p_b^2 + (1-p_b^2) \times 0 + O\left(\frac{1}{N}\right) \\ &= p_b^4 + O\left(\frac{1}{N}\right) \end{aligned} \tag{5.20}$$

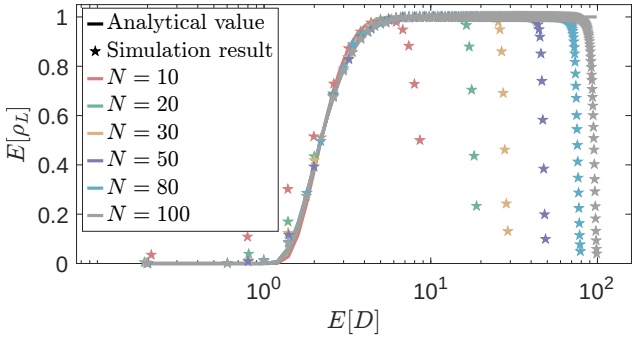


Figure 5.9: The average fraction $E[\rho_L]$ of flow subgraph as a function of the expected average degree $E[D]$ in unweighted ER graphs with different numbers of nodes.

Fig. 5.8 compares our analytical results by Eq. (5.20) with simulations on weighted ER graphs. The link weights are uniformly distributed from $(0,1)$. A similar pattern to Fig. 5.6 is visible. When the graph size N is small, the branches lead to a gap between the analytical results and the simulation. The gap decreases and approaches 0 as N increases. For unweighted ER graphs, our Eq. (5.20) provides accurate predictions as long as the expected degree is not too large, Fig 5.9. In the regime of large expected degrees, however, the number of links in the flow subgraph decreases due to Eq. (5.19). In the

extreme case where the unweighted graph is complete, the size of the flow subgraph reaches $|G^*_{ij}| = 1 + 2(N-2)$.

5.4. GRAPH CONSTRUCTION WITH END-TO-END DEMANDS ON POWER DISSIPATION

In this section, we focus on the flow network design problem, where we draw an analogy to electrical resistor networks. In an electrical resistor network G, the current flowing through each link leads to power dissipation because of the resistors. For any given resistor network, the power P_G dissipated when transmitting a current between two nodes i and j can be directly computed by (5.10): $P_G = \sum_{l\in\mathscr{L}} P_l = \sum_{l\in\mathscr{L}} y_l^2 r_l = I_c^2 \omega_{st}$. In Appendix D.2, we show that for unweighted ER graphs or ER graphs with uniformly distributed link weights, the power dissipated on a link exhibits a power law decay with respect to the size of the graph N and link density p. However, the inverse problem remains a non-trivial challenge: how to construct a graph that satisfies predetermined graph power dissipation demands (E2E constraints on power dissipation budget) d_{ij}, i.e., the required transportation cost between a node pair (i, j) in a flow network.

As shown by (5.10), the power dissipation is determined by the injected current I_c and the resistance r_l of links carrying transported current. Since the demand d_{ij} can be normalized to the graph power dissipation P_G with unit current injected in the graphs, we consider $I_c = 1$ in the following analysis. In a path network where the item travels along the shortest path and the link weight w_l denotes the "resistance" or transportation cost on link l during the transportation, the power dissipation P_G then numerically equals the shortest path weight s_{ij} between node i and j. Hence, the problem reduces to constructing a graph such that the corresponding shortest path weight matrix S equals the given demands D, which is solved by the inverse all shortest path problem [8].

Similar to the path network case, when unit currents are considered, the inverse power dissipation problem on a flow network reduces to constructing a network that satisfies prescribed end-to-end effective resistance demands:

Problem 7 (Inverse Effective Resistance Problem (IERP))**.** *Given an $N \times N$ symmetric demand matrix D with zero diagonal elements but positive off-diagonal elements. Determine an $N \times N$ weighted adjacency matrix $\widetilde{A}$ for the flow network, such that the norm $\|D - \Omega\|$ between the demand matrix and the effective resistance matrix is minimised.*

5.4.1. INVERSE EFFECTIVE RESISTANCE PROBLEM

Since the effective resistance matrix is a distance matrix[2], we assume that the demand matrix D is also a distance matrix in the sequel. Indeed, Van Mieghem [7] has proposed a method to transform an arbitrary demand matrix D into a distance matrix: If $d_{ik} + d_{kj} < d_{ij}$ for at least one node $k \in \mathscr{N}$ which violates the triangle inequality of a distance matrix, then we can replace $d_{ij} = \min_{1\le k\le N}(d_{ik} + d_{kj})$ and $d_{ji} = d_{ij}$.

When the given demand matrix D can be realized by a graph, i.e., there exists at least one graph whose effective resistance matrix $\Omega = D$, the IERP can be solved elegantly by

[2] Any element h_{ij} of a distance matrix H is non-negative $h_{ij} \ge 0$, but $h_{ii} = 0$ and h_{ij} obeys the triangle inequality: $h_{ij} \le h_{ik} + h_{kj}$.

Fiedler's block matrix relation [7, 113, 114],

$$\begin{pmatrix} 0 & u^T \\ u & \Omega \end{pmatrix}^{-1} = \begin{pmatrix} -2\sigma^2 & p^T \\ p & -\frac{1}{2}\widetilde{Q} \end{pmatrix} \tag{5.21}$$

with $\Omega p = 2\sigma^2 u$, where $\widetilde{Q} = \widetilde{\Delta} - \widetilde{A}$ is the weighted Laplacian matrix of a flow network and the diagonal matrix $\widetilde{\Delta} = \mathrm{diag}(\widetilde{A}u)$, $\widetilde{A}$ is the weighted adjacency matrix of a flow network, the variance $\sigma^2 = \frac{\zeta^T \widetilde{Q} \zeta}{4} + R_G$, where R_G is the effective graph resistance and u is $N \times 1$ the all-one vector. The vector ζ contains the diagonal elements of pseudoinverse $\widetilde{Q}^\dagger$ of the Laplacian $\widetilde{Q}$. Fiedler's block matrix relation (5.21) indicates a one-to-one relation between the effective resistance matrix Ω and the weighted Laplacian $\widetilde{Q}$ and therefore, also between the effective resistance matrix Ω and the weighted adjacency matrices $\widetilde{A}$. By applying block inverse formulae [6] to Fiedler's block matrix relation, the inverse of the effective resistance matrix is

$$\Omega^{-1} = \frac{1}{2\sigma^2} p p^T - \frac{1}{2}\widetilde{Q} \tag{5.22}$$

where $2\sigma^2 = \frac{1}{u^T \Omega^{-1} u}$ and the vector $p = \frac{1}{u^T \Omega^{-1} u} \Omega^{-1} u$. Hence,

$$\widetilde{Q} = \frac{2}{2\sigma^2} p p^T - 2\Omega^{-1} \tag{5.23}$$

and with $\widetilde{Q} = \widetilde{\Delta} - \widetilde{A}$, the weighted adjacency matrix follows as

$$\widetilde{A} = \widetilde{\Delta} + 2\Omega^{-1} - \frac{1}{\sigma^2} p p^T \tag{5.24}$$

We refer to the method that substitutes the effective resistance matrix Ω in (5.24) with a given demand D as the "Fiedler approach". Given a demand matrix D, if there exists at least one graph whose effective resistance matrix $\Omega = D$, then the Fiedler approach can provide an exact solution to the IERP instance.

5.4.2. GRAPH REALIZABILITY OF PERTURBED EFFECTIVE RESISTANCES

We call a demand matrix D graph-realizable if there exists at least one graph G whose effective resistance matrix $\Omega = D$. Constructing a demand matrix that is guaranteed to be graph realizable remains, to the best of our knowledge, a challenging problem. The main difficulty lies in the dependence among the effective resistances of different node pairs. To demonstrate the difficulty, we now consider adding a perturbation to the effective resistance Ω, which is motivated by the study on the sum of effective resistances in tree graphs in [8].

Consider a graph G with weighted Laplacian $\widetilde{Q}$ and effective resistance Ω. Let K be a symmetric perturbation matrix with zero diagonal elements $k_{ii} = 0$ and whose off-diagonal elements are arbitrary real numbers. Assume that there exists a graph G' whose effective resistance satisfies $\Omega' = \Omega + \varepsilon K$, where ε is an arbitrary insignificant positive real number.

By substituting the perturbed Ω' for Ω in (5.23) and ignoring terms of order ε^2 and higher, the weighted Laplacian $\widetilde{Q}'$ of graph G' can be expressed as:

$$\widetilde{Q}' \approx \widetilde{Q} + \varepsilon \Delta \widetilde{Q} \tag{5.25}$$

where

$$\Delta\widetilde{Q} \approx 2M + 2(u^T M u)pp^T - \frac{1}{\sigma^2}\left(Mpp^T + pp^T M\right), \tag{5.26}$$

the vector $p = \frac{1}{u^T\Omega^{-1}u}\Omega^{-1}u$ and $M = \Omega^{-1}K\Omega^{-1}$. The details are provided in Appendix B.3.

Consider two nodes i and j that are not connected directly by a link and the corresponding element $\widetilde{q}_{ij} = 0$ of the weighted Laplacian. The element $\widetilde{q}_{ij} = 0$ of the perturbed Laplacian is then determined by the first-order perturbation term $\varepsilon\Delta\widetilde{Q}$ only. Since $\varepsilon > 0$, the sign of $\Delta\widetilde{q}_{ij}$ determines the sign of $\widetilde{q}_{ij}$. Eq. (5.26) shows that the first-order perturbation $\Delta\widetilde{q}_{ij}$ depends on the inverse of the effective resistance matrix Ω^{-1} and the perturbation matrix K. Since the inverse of the effective resistance matrix Ω^{-1} is fully determined by the topology and link weights of graph G and the perturbation matrix K can take arbitrary values, even an arbitrarily small perturbation can yield a positive $\Delta\widetilde{q}_{ij}$, thereby producing a positive $\widetilde{q}'_{ij} > 0$, which implies, since $\widetilde{q}_{ij} = -\widetilde{a}_{ij}$, a negative weighted adjacency matrix element $\widetilde{a}_{ij}$. The appearance of a positive off-diagonal $\widetilde{q}'_{ij}$ or, equivalently, a negative off-diagonal $\widetilde{a}'_{ij}$ indicates that the addition of an extremely small, but non-zero perturbation in Ω cannot always lead to a graph-realizable effective resistance Ω'.

A similar situation, where even small perturbations in the effective resistance matrix can render the matrix not graph-realizable, appears in the network version of Calderón's inverse problem [155, 156, 157], which is known to be a severely ill-posed problem. Consider an electrical network G whose link weight represent a conductance and nodes are classified as boundary nodes and interior nodes. Given the adjacency matrix A of network G, the prescribed electric potential on the boundary nodes and the currents injected into (or extracted from) the boundary nodes, Calderón's inverse problem asks for obtaining the conductance of every link such that all potential and current constraints are satisfied. Carmona et al. [156] demonstrated that a small perturbation on the given potentials or currents may result in extremely large variations in the recovered conductances and in some cases may even produce negative conductances. Moreover, the big discrepancies appear on links that are far away from the boundary nodes in terms of hopcount. The severe ill-posedness arises because the boundary measurements on currents and voltages are only weakly sensitive to variations of conductances located deep inside the network, so that small perturbations at the boundary input may correspond to large and non-unique changes in the interior conductances.

Our investigation of small perturbations on effective resistance highlights the difficulty of constructing a graph-realizable demand matrix: even an insignificant perturbation of a valid effective resistance matrix may destroy its graph realizability; therefore, generating demands by perturbing an existing effective resistance matrix or by directly specifying a resistance matrix is unreliable. Hence, the Fiedler approach, which directly executes (5.24) by substituting the effective resistance matrix Ω with the given demand matrix D, loses feasibility in practical scenarios where measure accuracy is limited. To solve the IERP for arbitrary demands D, we propose an algorithm called "Resistor Gap Pruning (RGP)" (Algorithm 6), by leveraging resistor and parallel circuit rules, in Section 5.4.3.

5.4.3. RESISTOR GAP PRUNING ALGORITHM

The Resistor Gap Pruning (RGP) algorithm aims to obtain a graph whose effective resistance matrix Ω is approximately equal to an arbitrary predetermined demand matrix D. Consider a flow network G with each link $l = i \sim j$ possessing a resistor $r_{ij} = \frac{1}{w_{ij}}$ equaling the reciprocal of the link weight. In a path network, Qiu et al. [8] demonstrated that a link with small link weight w_{ij}, which corresponds to a high resistor r_{ij} in a flow network, is unlikely to be included in a shortest path. Utilizing this principle, Qiu et al. [8] proposed the Omega Link Removal algorithm to construct a graph such that the corresponding shortest path weight matrix S approximately equals a given demand matrix D. Inspired by [8], we now propose our RGP algorithm for designing a flow network.

Algorithm 6 Resistor Gap Pruning (RGP)

Input: $N \times N$ demand matrix D
Output: $N \times N$ weighted adjacency matrix $\widetilde{A}$ for flow network

$A \leftarrow J - I$
$W \leftarrow$ Element-wise reciprocal of D over nonzero elements
$\widetilde{A} \leftarrow A \circ W$
$\Omega \leftarrow$ Effective resistance matrix of $\widetilde{A}$
$\epsilon \leftarrow ||D - \Omega||$
do
    $\epsilon' \leftarrow \epsilon$
    $\hat{\Omega} \leftarrow$ Element-wise reciprocal of Ω over nonzero elements
    $\Gamma \leftarrow \left(\hat{\Omega} - \widetilde{A}\right) \circ (D - \Omega) \circ A$
    $(i, j) \leftarrow$ Indices of the maximum element in Γ
    $A \leftarrow A - e_i e_j^T - e_j e_i^T$
    $\widetilde{A} \leftarrow A \circ W$
    $\Omega \leftarrow$ Effective resistance matrix of $\widetilde{A}$
    $\epsilon \leftarrow ||D - \Omega||$
while $\epsilon < \epsilon'$
$A \leftarrow A + e_i e_j^T + e_j e_i^T$
$\gamma_{ij} \leftarrow d_{ij}/\omega_{ij}$ for each non-diagonal indices (i, j) in Ω
$\alpha \leftarrow$ mean of all γ_{ij}
$\widetilde{A} \leftarrow \frac{1}{\alpha} A \circ W$
return $\widetilde{A}$

The main mechanism of RGP is based on the parallel circuit rules and (5.9). The parallel circuit rules show that the effective resistance in the circuit is always less than or equal to the smallest resistance (see [8]). In other words, for two adjacent nodes i and j, the effective resistance always satisfies $\omega_{ij} \leq r_{ij}$, with equality holding only when the direct link $i \sim j$ is the unique path connecting nodes i and j. Hence, for an arbitrary node pair (i, j), adding links in G can only decrease the effective resistance ω_{ij} or leave ω_{ij} unchanged, while removing links in G *cannot* decrease the effective resistance ω_{ij}. Furthermore, (5.9) indicates that removing a link with a high resistor[3] r_{ij} has an insignif-

[3] A high resistor means small current through the link by the law of Ohm; hence, a small proportion of traffic. In

icant effect on the effective resistance ω_{ij}. The link $l = i \sim j$ with a high resistor r_{ij} is then regarded as a "redundant effective resistance link". The main idea of RGP is to remove redundant effective resistance links from a complete graph so that the effective resistance matrix Ω approaches the target demand D:

1. Construct a complete graph whose weighted adjacency matrix $\widetilde{A} = D$.

2. Iteratively remove links whose removal has an insignificant impact on the graph effective resistance.

The details are shown by Algorithm 6.

The RGP algorithm is initialised in line 1 by the complete graph with the adjacency matrix $A = J - I$, while the link weight matrix W is element-wise reciprocal of D over nonzero elements, line 2. Lines 3 and 4 then obtain the weighted adjacency matrix $\widetilde{A}$ of the flow network and the corresponding effective resistance matrix Ω. The difference between the demand matrix D and the effective resistance matrix Ω is computed in line 5 as ϵ. Specifically, the difference is measured by $L1$ norm $\|D - \alpha\Omega\| = \sum_{ij} |d_{ij} - \alpha\omega_{ij}|$. Assume there exists a graph G whose effective resistance matrix Ω_G exactly equals the demand matrix D. The effective resistance matrix $\Omega < \Omega_G$ because the resistance of each link $r_{ij} = d_{ij}$ and there are multiple paths between node i and j. We expect that the difference ϵ will shrink as the links are iteratively removed until the obtained graph is close to the graph G whose effective resistance matrix Ω_G exactly equals the demand matrix D. To determine which link should be removed, in line 9, we compute the $N \times N$ heuristic score matrix

$$\Gamma \leftarrow \left(\hat{\Omega} - \widetilde{A}\right) \circ (D - \Omega) \circ A,$$

where the $N \times N$ matrix $\hat{\Omega}$ contains the element-wise reciprocal of the effective resistance on the off-diagonal elements, with the diagonal elements remaining zero. The heuristic score consists of three components:

1. The first factor $\left(\hat{\Omega} - \widetilde{A}\right)$ of the heuristic score matrix Γ denotes the reciprocal of effective resistance between a pair of adjacent nodes (i.e. $a_{ij} = 1$), in case the direct link between them is removed (as in (5.9)). Specifically, from (5.9), we have

$$\frac{1}{(\omega_{G\setminus l})_{ij}} = \frac{1}{\omega_{ij}} - \frac{1}{r_{ij}}$$

   A smaller $(\omega_{G\setminus l})_{ij}$, equivalently a larger $\frac{1}{(\omega_{G\setminus l})_{ij}}$), may imply a smaller contribution of $\frac{1}{r_{ij}}$ and thus a larger r_{ij}, indicating that link $l = i \sim j$ can be regarded as redundant. Moreover, a smaller $(\omega_{G\setminus l})_{ij}$ indicates that, after removing link $l = i \sim j$, nodes i and j may remain connected through multiple paths acting in parallel. In other words, the adjacent nodes i and j are easily reachable via the rest of the graph when the link is removed.

2. The second factor $(D - \Omega)$ quantifies the gap between the predetermined demand and the obtained effective resistance in the current iteration.

the extreme case where the resistor $r_{ij} \to \infty$ implying that almost no current flows through the link $l = i \sim j$.

3. The third factor is the adjacency matrix, which encodes the graph topology at the current iteration and ensures that only existing links are considered for removal.

By jointly accounting for the link redundancy, the reachability between two nodes after link removal and the difference between the current effective resistance and the target demand, we remove the existing link with the highest value in Γ (lines 10 and 11). If multiple links have the same heuristic score, one link is randomly removed; therefore, the solution provided by RGP may not be unique. The adjacency matrix A, the weighted adjacency matrix $\widetilde{A}$ of the flow network, the corresponding effective resistance matrix Ω and the difference ϵ between the demand and the effective resistance matrix are then updated respectively in lines 12-14. We perform the link removal until the updated ϵ is larger than the difference ϵ' in the previous iteration. At that point, the last removed link is returned (line 16).

In lines 17-19, we further scale the link weight matrix W according to the norm $\|D-\Omega\|$. Since the effective resistance Ω is sensitive even to insignificant perturbations [8], we cannot independently scale each element of the effective resistance Ω with a separate parameter. Fortunately, scaling the entire effective resistance matrix Ω, by a positive scaling parameter α yielding $\alpha\Omega$, guarantees that the corresponding weighted adjacency matrix remains nonnegative: From (3.4): $\omega_{ij} = (e_i - e_j)^T Q^\dagger (e_i - e_j)$, we have

$$\alpha\omega_{ij} = \alpha (e_i - e_j)^T Q^\dagger (e_i - e_j) \tag{5.27}$$

$$= (e_i - e_j)^T \left(\frac{1}{\alpha} Q\right)^\dagger (e_i - e_j), \tag{5.28}$$

which indicates that $\alpha\Omega$ is the effective resistance of a graph with Laplacian $\frac{1}{\alpha}\widetilde{Q}$ and weighted adjacency matrix $\frac{1}{\alpha}\widetilde{A}$. The remaining task is then to obtain the scaling parameter α that minimizes the norm $\|D-\Omega\|$, e.g., $L1$ norm, $\|D-\alpha\Omega\| = \sum_{ij}|d_{ij}-\alpha\omega_{ij}|$. Because both the demand d_{ij} and effective resistance ω_{ij} are positive, each term $|d_{ij}-\alpha\omega_{ij}|$ can be expressed piecewise as

$$|d_{ij}-\alpha\omega_{ij}| = \begin{cases} d_{ij}-\alpha\omega_{ij}, & \text{if } \alpha \le \dfrac{d_{ij}}{\omega_{ij}}, \\ \alpha\omega_{ij}-d_{ij}, & \text{if } \alpha > \dfrac{d_{ij}}{\omega_{ij}}, \end{cases}$$

which is a convex piecewise-linear function, minimized at $\alpha = \dfrac{d_{ij}}{\omega_{ij}}$. Hence, the norm $\|D-\alpha\Omega\| = \sum_{ij}|d_{ij}-\omega_{ij}|$ is also a convex piecewise linear function and the minimum occurs at one of the breakpoints, corresponding to a $\dfrac{d_{ij}}{\omega_{ij}}$ for a (or some) node pair (i,j). In line 18, instead of searching over all possible $\dfrac{d_{ij}}{\omega_{ij}}$ to obtain the minimum of norm $\|D-\alpha\Omega\|$, we use α equaling the mean of $\dfrac{d_{ij}}{\omega_{ij}}$ for all possible (i,j), which can reduce the computational complexity. In line 19, the weighted adjacency matrix $\widetilde{A}$ is updated

by scaling the link weight matrix W with the factor $\frac{1}{\alpha}$. Finally, in line 20, the RGP returns the resulting weighted adjacency matrix $\widetilde{A}$.

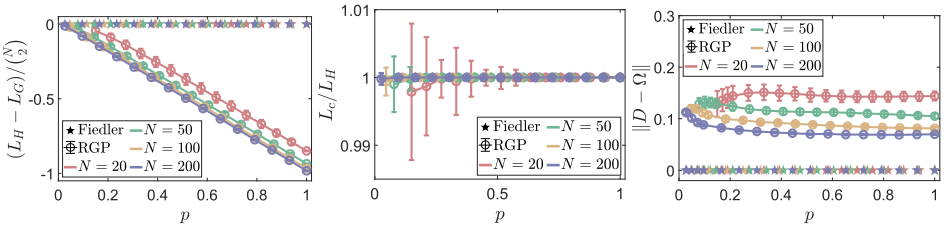
5.4.4. PERFORMANCE OF RGP ALGORITHM

In this section, we evaluate the performance of the RGP algorithm. We introduce a benchmark approach "Fiedler approach", which executes (5.24) by substituting the effective resistance matrix Ω with an input demand matrix D, as a reference. Since the Fiedler approach [4] is only feasible when the demand matrix is graph-realizable, our experiment is conducted given demand $D = \Omega_G$, where Ω_G is the effective resistance matrix of a randomly generated graph G. Specifically, for each experiment trial, we first generate a "baseline" graph G and compute the corresponding effective resistance matrix Ω_G. With the demand matrix $D = \Omega_G$ as input, RGP and the Fiedler approach are respectively executed to obtain the resulting graph H and the corresponding effective resistance matrix Ω. We then measure how the resulting graph H, produced by RGP, differs from the benchmark approach. The performance of RGP is then assessed by three complementary criteria: (i) the normalized number $\frac{2}{N(N-1)}(L_H - L_G)$ of additional links in the resulting graph H, (ii) the number of common links $L_c = \frac{1}{2}u^T(A \circ A_H)u$ shared by the baseline graph G and the resulting graph H and (iii) the norm $||D - \Omega|| = \frac{1}{N(N-1)}\sum_i \sum_j \frac{|d_{ij} - \omega_{ij}|}{d_{ij}}$ of the demand matrix D and the effective resistance matrix Ω.

We first consider the demand $D = \Omega_G$ derived from the sparsest graph, tree. Given a tree graph G, the RGP algorithm can generally recover a graph $H = G$ with $D = \Omega_G$ as input, see Fig. D.3 in Appendix D.3. In a tree graph, there exists a unique path $\mathscr{P}_{ij}$ between an arbitrary node pair (i, j) and the effective resistance ω_{ij} is the sum of the resistances r_l of links along the path $\mathscr{P}_{ij}$. Starting from the initial complete graph (lines 1-3 in Algorithm 6) by RGP, whose link weights W are the element-wise reciprocal of the demand D over nonzero elements, any link $l = i \sim j$ that is absent in the baseline tree G has a relatively high resistance r_l, as node i and j can already connect with each other via the path $\mathscr{P}_{ij}$ in the baseline tree G. Moreover, according to parallel series rules, the effective resistance ω_{ij} of a link $l = i \sim j$ that exists in the complete graph but not in the baseline tree G is significantly smaller than the demand d_{ij}. The RGP algorithm can then leverage both the high resistance r_{ij} and large difference $d_{ij} - \omega_{ij}$ to remove redundant effective resistance links, ultimately recovering a tree graph H that matches the given tree graph G.

We then extend the performance evaluation using demands $D = \Omega_G$ computed from ER graphs with different link densities p, Fig. 5.10. Specifically, for each size of the graph N and link density p, 1000 ER graphs $G_p(N)$ with link weight uniformly distributed in $(0, 1)$ are generated. As shown in Fig. 5.10, the Fiedler approach can exactly recover the given baseline graph $G_p(N)$, which is expected. Compared to the benchmark Fiedler approach, RGP generally produces graphs H with fewer links (Fig. 5.10a). The normalized number of additional links of resulting graphs H by RGP decreases linearly as the link density p increases. Fig. 5.10b presents the common links L_c shared by graph H and G normalized by link number L_H. The ratio $\frac{L_c}{L_H}$ is approximately 1, indicating that nearly

[4] Solving the IERP with arbitrary demands, to the best of our knowledge, is a novel problem and there is no scheme to compare with.



(a) Normalized number of additional links

(b) Normalized number of common links

(c) Norm difference

Figure 5.10: Performance evaluation of the RGP algorithm (RGP) compared with the Fiedler approach (Fiedler). (a) Normalized number of additional links in the resulting graph H compared with the baseline G. (b) Normalized number of common links in the baseline graph G and the resulting graph H. (c) shows the norm between the demand matrix D and the resulting effective resistance matrix Ω. The x-axis of all three panels represents the connection probability p of ER graphs $G_p(N)$. For each graph size N and probability p, 1000 ER graphs with uniformly distributed link weights are generated as baseline graphs. Error bars indicate the standard deviation.

5

all the links in the reconstructed graph H also exist in the given graph G. In Fig. 5.10c, we further investigate the difference between the effective resistance matrix Ω of the resulting graph by RGP and the demand D. The RGP algorithm performs better in graphs with a larger size N.

The feasibility of RGP is also examined for demands derived from ER graphs with different link weight distributions. A similar pattern as Fig. 5.10 is visible in Fig. D.4 in Appendix D.3, where the link weights of the given ER graphs exhibit an exponential distribution with an average equaling 0.5. Further, in Table D.2 in Appendix D.3, we demonstrate the consistent performance of RGP across demands generated from different empirical networks [117, 158].

The main computational complexity of the RGP algorithm comes from updating the effective resistance (line 13), which requires computational complexity $O(N^3)$ in each iteration. Since the RGP algorithm initialises the topology with a complete graph and iteratively removes redundant links until, in the worst case, the graph becomes a tree graph with $N-1$ links, there are up to $\frac{N(N-1)}{2} - (N-1)$ iterations. Therefore, the overall computational complexity for the worst case of the RGP algorithm is $O(N^5)$. To streamline the computational complexity, one possible strategy is to employ the Sherman-Morrison equation [159], which provides an efficient way to compute the inverse of a matrix after a rank-one modification. In each iteration of RGP, we remove one link from the graph, which corresponds to a rank-one update to the Laplacian Q. Hence, after the first iteration, the Sherman-Morrison equation can eliminate the need to compute the pseudoinverse of the Laplacian $Q^\dagger$ and effective resistance Ω from scratch, which reduces the computational complexity of updating Ω from $O(N^3)$ to $O(N^2)$. Accordingly, the overall computational complexity of the RGP can be reduced to $O(N^4)$.

5.5. CONCLUSION

Motivated by practical scenarios in 6G communication and information spreading on social platforms, where transport cannot be accurately described by a single shortest path, we study flow networks in which transport distributes proportionally across all available paths. We first analyze the size of the flow subgraph in flow networks. We decompose the giant component (GC) into a backbone subgraph $\mathscr{B}$ and branches. Using the requirement that every non-terminal node in the flow subgraph must have at least two neighbors within the flow subgraph, we establish a self-consistent formulation. From this formulation, we obtain the probability p_b that a node has at least one neighbor that belongs to the backbone and the backbone fraction b. Both probability p_b and backbone fraction b remain zero when the expected degree $E[D] < 1$ and become positive once the expected degree exceeds one. Based on these quantities, we characterize the expected size of the flow subgraph. Conditioned on both nodes lying in the giant component, the normalized size of the flow subgraph converges to $b\,p_b^2$ with an $O(1/N)$ finite-size deviation. A similar reasoning applies to the number of links in the flow subgraph, which converges to p_b^4 up to the same correction order. Simulations agree with these results and reproduce both the large-scale behavior and the finite-size deviations.

Finally, we consider the challenge of constructing a graph with predetermined end-to-end power dissipation, which can be reformulated as the *inverse effective resistance problem* (IERP) that asks for a graph such that the corresponding effective resistance matrix equals a given demand matrix. Inspired by circuit rules and effective resistance computation, we propose a heuristic algorithm, "Resistor Gap Pruning" (RGP), which provides sparse graphs closely approximating the demand effective resistance and shows stable performance across different demand scenarios. Further, our simulations reveal that nearly all links belonging to the sparse graphs H obtained by RGP also exist in the baseline ER graph G, indicating that the links in graph H exhibit significant influence on the effective resistance. Indeed, for an unweighted ER graph, previous studies [41, 51, 160, 161, 162] have shown that the effective resistance can be approximated as $\omega_{st} \propto \frac{1}{d_s} + \frac{1}{d_t}$, stating that the transport is dominated by the links adjacent to node s and t, while the remaining graph is practically a perfect conductor. Our findings then provide a complementary perspective on those "critical" links/nodes that determine effective resistance in a weighted flow network: the effective resistance is likewise dominated by a small number of links that interconnect the network. Moreover, the proposed RGP algorithm can also be utilized for a type of "network sparsification", which provides a sparser network while preserving a similar effective resistance matrix.

Although our analysis is formulated using a flow network analogy to electrical networks, the results are not limited to the electrical network domain. Rather, the IERP can be applied to different domains, including telecommunication networks, water distribution networks, social networks, and transportation networks, where transport processes can be effectively modeled using random walks. For example, in telecommunication networks, our RGP can provide a scheme for the design of low-power Internet of Things (IoT) networks [163], where the effective resistance captures energy consumption under unit data transmission. More generally, IERP is not confined to scenarios in which physical power dissipation itself is the demand or constraint. The demand can represent a general end-to-end cost of transporting a unit of items between two nodes. For

instance, in a telecommunication network, the resistance of a link l can represent the operational and maintenance costs associated with transmitting a unit of IP packets on link l. Similarly, in a social network, the demand may quantify the cost of propagating information from different source node to destination nodes, while the resistance of each link corresponds to the cost incurred when information passes through that link.

6

CONCLUSION

Time and stupidity are your biggest enemies.

Piet Van Mieghem

This dissertation aims to enhance the understanding of how a network supports an effective item transmission between the source and destination and how to construct a network given requirements on item transmission. Our exploration investigates both path networks, where the item transmission typically follows the shortest path and flow networks, in which the item propagates over proportionally all possible paths. For path networks, we have studied the geometric organization and identification of the shortest path nodes on 2-dimensional Euclidean space. Additionally, we developed several approaches for different network design problems motivated by practical scenarios. For flow networks, we provided an analysis of the subgraph that contributed to item transmission in random networks and the power dissipation associated with transporting items. Finally, we studied a flow network design problem. In this chapter, the main contribution of this dissertation and directions for future work are presented.

6.1. MAIN CONTRIBUTION AND REFLECTIONS

This dissertation aims to deepen the understanding of how complex networks support item transmission between sources and destinations and how such networks can be systematically constructed to meet prescribed transmission requirements.

Chapter 2 investigated the geometric organization and the identification of the shortest path nodes in soft random geometric graphs (SRGGs) in 2-dimensional Euclidean space. We have demonstrated that the shortest paths are aligned along geodesic curves connecting shortest path endpoints. To quantify the strength of the alignment, we introduce two metrics: the average distance to the geodesic from shortest path nodes and the average path stretch. We showed that both metrics decrease in larger SRGGs with short-range connections, while exhibiting a nontrivial, nonmonotonic behavior with respect to the average degree. We have also analyzed the hopcount of the shortest path and the geodesic length between adjacent nodes in SRGGs. Furthermore, we established that the alignment of shortest paths may be sufficiently strong to allow the identification of shortest path nodes based on their proximity to geodesic curves, even when the precise node positions in the latent space are unknown.

In Chapter 3, we introduced the inverse all shortest path problem (IASPP), which asks for a weighted adjacency matrix of a graph such that all the elements in the corresponding shortest path weight matrix do not exceed those of the given demand matrix. We have proposed the Descending Order Recovery (DOR) algorithm, which exactly solves an optimized variant of IASPP in polynomial time, thereby proving that IASPP is not NP-hard. The network constructed by DOR minimizes both the number of links and the total link weight among all networks that share the same shortest-path weight matrix. Moreover, as a complementary approach, we developed Omega-based Link Removal (OLR), which leverages effective resistance to solve the optimized IASPP. The essence of OLR lies in solving a path network problem from the perspective of flow networks, thereby bridging two crucial metrics: the shortest path weight and effective resistance. Additionally, while an effective resistance matrix is a distance matrix, we have demonstrated that not every distance matrix admits a corresponding effective resistance matrix realization. Even a minor perturbation of the effective resistance matrix can significantly affect the corresponding weighted adjacency matrix, potentially resulting in negative link weights. Thus, for any distance matrix, there exists at least one simple graph with positive link weights whose shortest path weight matrix equals the given distance matrix; however, no such guarantee holds for a graph whose effective resistance matrix equals the distance matrix.

We further extended the IASPP in Chapter 4 to the modification of existing networks under changing item transmission requirements. Inspired by practical scenarios, we study two variants. The first variant named IASPP that minimizes the adjustment of the sum of link weights (IASPP$_M$)), which aims to construct a network such that the resulting corresponding shortest path weights meet the updated demands while minimizing the total change in link weights relative to the original network. An exact algorithm for IASPP$_M$ was proposed and its potential application domains were discussed. The second variant, the IASPP with a fixed adjacency matrix (IASPP$_F$), considers a more constrained scenario in which the network topology remains unchanged. For IASPP$_F$, we develop an approach applicable to tree networks, which are commonly found in domains such as

telecommunication and industrial networks. Compared with classical approaches, the DBS can improve computational efficiency while constructing a network that is close to the optimal solution. We have shown examples of how the IASPP$_F$ scheme can be used to manage resources in an industrial network.

In Chapter 5, we have studied the item transmission in flow networks. We established a framework to compute the expected number of nodes and links involved in transferring items between the source and destination in random networks. For weighted Erdős–Rényi networks, we showed a phase transition in the flow subgraph size occurring when the expected degree $E[D]$ exceeds 1. We then discussed the power dissipation associated with the transportation at the link level. We have demonstrated that the power dissipated on each link in ER networks exhibits a power-law decay as a function of network size or link density, with exponents of -3 and -2, respectively. Moreover, we addressed how to construct a network given predetermined end-to-end power dissipation, which can be reduced to the "inverse effective resistance problem" that asks for a network whose effective resistance matrix matches a given demand. We proposed a heuristic algorithm, "Resistor Gap Pruning" (RGP), which provides sparse networks closely approximating the demand effective resistance and performs consistently across different demand scenarios. The proposed RGP algorithm can also be utilized for a type of "network sparsification", which provides a sparser network while preserving a similar effective resistance matrix.

Across all chapters, the transmission of items in both path and flow networks tends to follow principles that lower the overall "cost" or "resistance", even though their transmission rules are substantially different. In path networks, the shortest paths in soft random geometric graphs are aligned along geodesic curves, which minimize the geometric distance between two endpoints, Chapter 2. Meanwhile, in flow networks, the effective resistance serves as the distance metric, is proportional to the minimization of electrical power, Chapter 5 and Ref [6]. Regarding network design, in Chapter 3 and 5, we have shown that removing links with high resistance values, i.e., links with large weights in path networks or small weights in flow networks (the reciprocal relation), can approximately preserve both the shortest path distance matrix and the effective resistance matrix in a dense network. This finding indicates that the links with low resistance play crucial roles in the function of networks. Though establishing a direct mathematical bridge between different item-transmission metrics (e.g., the shortest path weight and effective resistance) can be challenging (Chapter 3), our work has addressed item-transmission problems by integrating insights from multiple metrics. For example, we predicted shortest path nodes based on their geometric proximity in Chapter 2 and constructed networks that satisfy prescribed shortest path demands from the perspective of effective resistance in Chapter 3.

6.2. FUTURE WORK

Based on the methods and results in this thesis, we raise some promising future directions.

In Chapter 2, our findings on the shortest path nodes open avenues for follow-up questions. One of them is the problem of shortest path finding in geometric networks where nodes are distributed heterogeneously. This is the case for transportation, infras-

tructure and communication networks where the densities of nodes may vary from one geographic region to another. Another important question is whether all shortest path nodes are equally difficult to find. In many networks, node degree centrality is correlated with betweenness centrality, implying that nodes with large degree values are more likely to belong to a shortest path and are typically found in the middle of a shortest path. This knowledge that large degree nodes are likely to be on a shortest path has limited practical importance since large degree nodes are harder to manipulate and altering the properties of these nodes may lead to cascades of changes throughout the network. In contrast, our preliminary experiments indicate that shortest path nodes situated closer to path endpoints tend to have a smaller distance to the geodesic and as a result, are easier to infer. Thus, a fusion approach combining elements of the distance to geodesic with network topology may ultimately be more accurate in finding shortest path nodes than one of the two approaches used separately.

In Chapter 3 and Chapter 4, we have established several frameworks for general network design problems given prescribed shortest path weight requirements. A remaining open question is how to extend the proposed algorithm for the IASPP with a fixed adjacency matrix to more general network topologies beyond trees. Moreover, practical network design problems often involve multiple simultaneous requirements. For example, the transmission of IP packets in telecommunication networks may be constrained by both delay and bandwidth. Thus, the inverse all shortest path problem on multiple layer networks may represent a promising direction for future research and applications.

In Chapter 5, we have shown that constructing an effective resistance matrix whose corresponding weighted adjacency matrix is guaranteed to be non-negative remains an open problem. One potential avenue is to exploit spectral properties of graphs, as Van Mieghem [164] recently addressed a problem "Given the spectrum (i.e., all eigenvalues and eigenvectors) of the Laplacian matrix of a graph, find the spectrum of the effective resistance matrix of that graph" and of the reversed problem. Building on these spectral relationships may provide valuable insights into ensuring the realizability of the effective resistance and further benefit effective-resistance-based network design.

A

APPENDIX OF CHAPTER 2

A

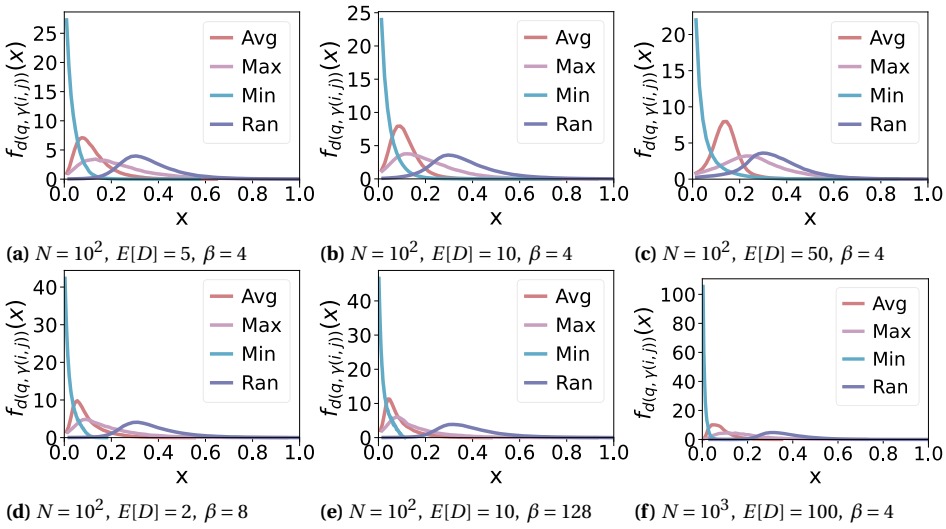
A.1. NOTATION FOR CHAPTER 2

Table A.1: Notation

Symbol	Definition
G	Graph or network
$\mathcal{N}$	Set of nodes
N	Number of nodes
$\mathcal{L}$	Set of links
L	Number of links
$l = i \sim j$	Link between node i and node j
$c_G(i)$	Clustering coefficient of node i
C_G	Clustering coefficient of network G
$E[D]$	Expected degree
$\langle D \rangle$	Real average degree of a network
$\mathscr{P}_{ij}$	Path from node i to node j
$\mathscr{P}^*_{ij}$	Shortest path between node i to node j
h	Hopcount of the shortest path $\mathscr{P}^*_{ij}$
$\gamma(i,j)$	Geodesic between nodes i and j
d_{ij}	Geometric distance between nodes i and j
d_l	Geometric distance between two end nodes of a link l
$d(q,\gamma(i,j))$	Geometric distance from node q to the geodesic $\gamma(i,j)$
$\langle d \rangle$	Average distance of shortest path nodes to the geodesic
r	Radius of an RGG or effective radius of an SRGG
S	Shortest path stretch

A.2. DISTANCE TO GEODESIC

Figure A.1 depicts the distributions of the distance to the geodesic in SRGGs for a variety of expected degree $E[D]$ and inverse temperature β values. Figure A.2 demonstrates the distribution of the node positions X in the shortest path corresponding to the largest distance to the geodesic. Figure A.3 demonstrates the distribution of the node positions X in the shortest path corresponding to the minimum distance to the geodesic.



(a) $N = 10^2$, $E[D] = 5$, $\beta = 4$ **(b)** $N = 10^2$, $E[D] = 10$, $\beta = 4$ **(c)** $N = 10^2$, $E[D] = 50$, $\beta = 4$

(d) $N = 10^2$, $E[D] = 2$, $\beta = 8$ **(e)** $N = 10^2$, $E[D] = 10$, $\beta = 128$ **(f)** $N = 10^3$, $E[D] = 100$, $\beta = 4$

Figure A.1: Distribution of the deviation in SRGGs. The x-axis represents the deviation of nodes that from the geodesic, while the y-axis is the deviation's probability density function. For each set of SRGG parameters, the number of nodes N, expected degree $E[D]$ and temperature parameter β, we generated 10^3 SRGG realization for $N \leq 100$ and 1 SRGG for $N \geq 1000$. For each graph G_i, $i = 1, 2 \ldots, 100$, we consider 10^5 shortest paths between randomly node pairs (we consider all the connected node pairs when the network size $N \leq 100$). For each (i, j) node pair, we find shortest path set $\mathscr{P}^*_{ij}$, geodesic $\gamma(i, j)$ and compute the minimum (Min), maximum (Max) and the average (Avg) distance to geodesic from all shortest path nodes. For each (i, j) node pair, we also randomly select a set of nodes equal in number to those on the shortest path $\mathscr{P}^*_{ij}$ and compute the average distance to the geodesic (Ran).

In this section, we also study the distributions of the distance to the geodesic for Waxman SRGGs with the connection probability function

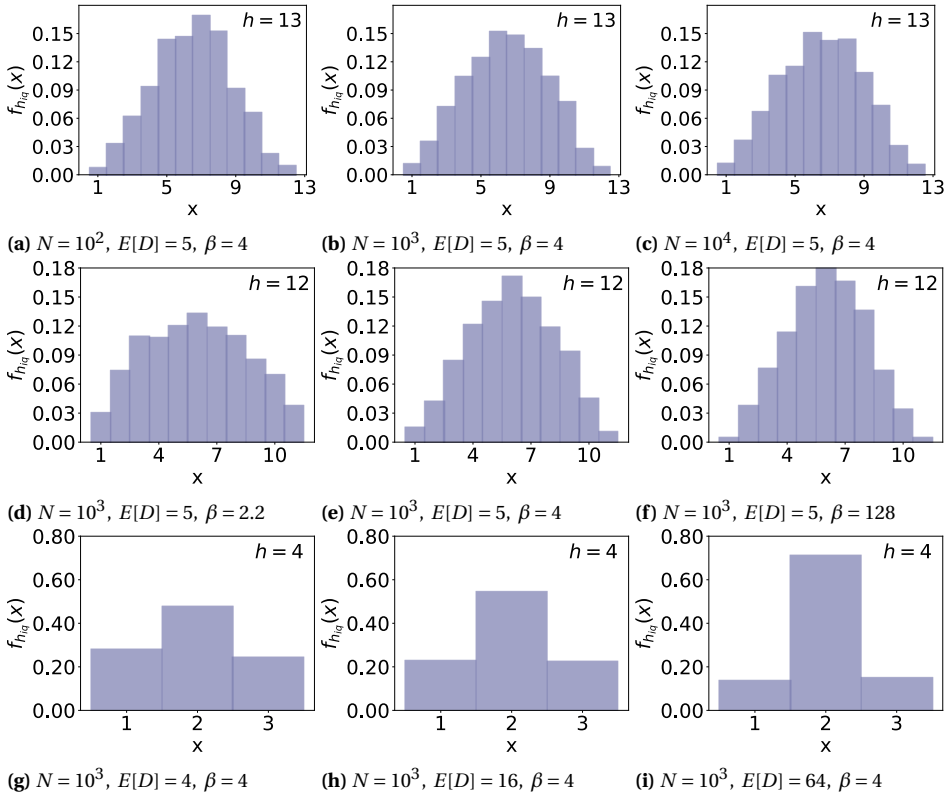
$$f(d) = \beta e^{\left(-\frac{d}{d_0}\right)^\eta}. \tag{A.1}$$

Waxman SRGGs of N nodes built on the unit square with uniform node density are characterized by the expected degree

$$E[D] \approx 2\pi N \beta \frac{d_0^2}{\eta} \Gamma\left(\frac{2}{\eta}\right). \tag{A.2}$$

A

Figure A.4 depicts the distributions of the distance to the geodesic for a variety of expected degree $E[D]$ and parameter η values. Note that the results for the Waxman SRGGs are similar to SRGGs with the Fermi-Dirac connection probability function.

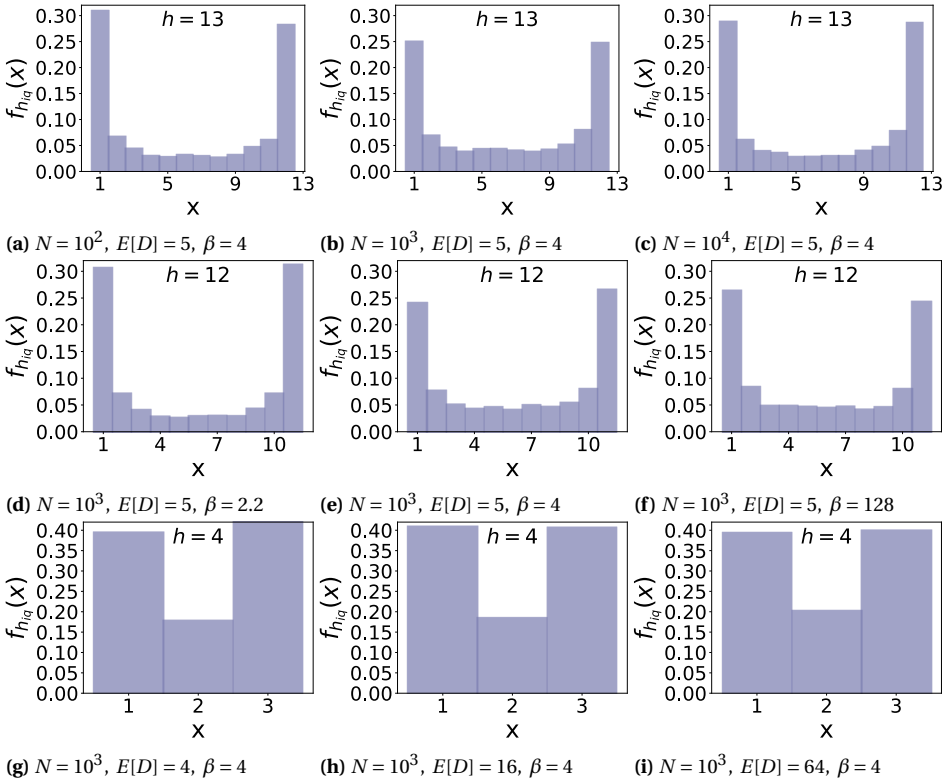


(a) $N = 10^2$, $E[D] = 5$, $\beta = 4$ **(b)** $N = 10^3$, $E[D] = 5$, $\beta = 4$ **(c)** $N = 10^4$, $E[D] = 5$, $\beta = 4$

(d) $N = 10^3$, $E[D] = 5$, $\beta = 2.2$ **(e)** $N = 10^3$, $E[D] = 5$, $\beta = 4$ **(f)** $N = 10^3$, $E[D] = 5$, $\beta = 128$

(g) $N = 10^3$, $E[D] = 4$, $\beta = 4$ **(h)** $N = 10^3$, $E[D] = 16$, $\beta = 4$ **(i)** $N = 10^3$, $E[D] = 64$, $\beta = 4$

Figure A.2: Distribution of the node positions X in the shortest path corresponding to the largest distance to the geodesic in different SRGGs. We consider the shortest paths of fixed hopcount h. For each shortest path, we found the most remote node and measured its position $X \in 1,...,h$. For each set of SRGG parameters, we generated 100 SRGG realizations if $N \leq 100$ and 1 realization for $N > 100$. For each generated graph G_i, $i = 1,2\ldots,100$, we considered 10^5 shortest paths between randomly selected connected node pairs, in case $N > 100$. We considered all connected node pairs in graphs of $N \leq 100$.



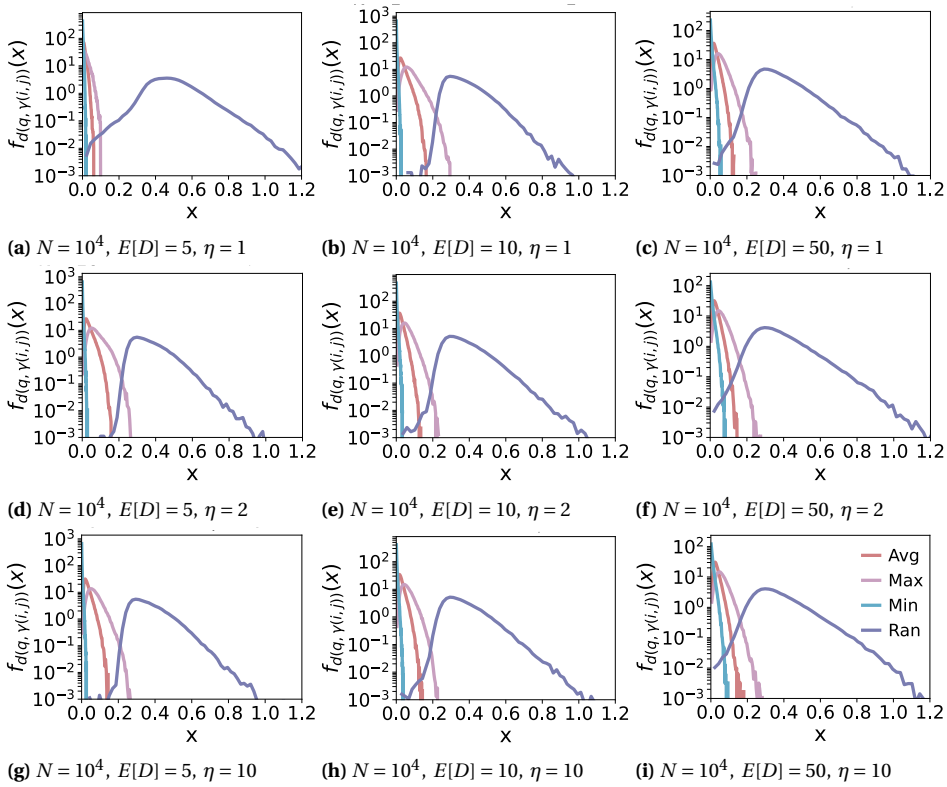
(a) $N = 10^2$, $E[D] = 5$, $\beta = 4$ **(b)** $N = 10^3$, $E[D] = 5$, $\beta = 4$ **(c)** $N = 10^4$, $E[D] = 5$, $\beta = 4$

(d) $N = 10^3$, $E[D] = 5$, $\beta = 2.2$ **(e)** $N = 10^3$, $E[D] = 5$, $\beta = 4$ **(f)** $N = 10^3$, $E[D] = 5$, $\beta = 128$

(g) $N = 10^3$, $E[D] = 4$, $\beta = 4$ **(h)** $N = 10^3$, $E[D] = 16$, $\beta = 4$ **(i)** $N = 10^3$, $E[D] = 64$, $\beta = 4$

Figure A.3: Distribution of the node positions X in the shortest path corresponding to the smallest distance to the geodesic in different SRGGs. We consider the shortest paths of fixed hopcount h. For each shortest path, we found the most remote node and measured its position $X \in 1,...,h$. For each set of SRGG parameters, we generated 100 SRGG realizations if $N \leq 100$ and 1 realization for $N > 100$. For each generated graph G_i, $i = 1,2\ldots,100$, we considered 10^5 shortest paths between randomly selected connected node pairs, in case $N > 100$. We considered all connected node pairs in graphs of $N \leq 100$.

A



(a) $N = 10^4$, $E[D] = 5$, $\eta = 1$ **(b)** $N = 10^4$, $E[D] = 10$, $\eta = 1$ **(c)** $N = 10^4$, $E[D] = 50$, $\eta = 1$

(d) $N = 10^4$, $E[D] = 5$, $\eta = 2$ **(e)** $N = 10^4$, $E[D] = 10$, $\eta = 2$ **(f)** $N = 10^4$, $E[D] = 50$, $\eta = 2$

(g) $N = 10^4$, $E[D] = 5$, $\eta = 10$ **(h)** $N = 10^4$, $E[D] = 10$, $\eta = 10$ **(i)** $N = 10^4$, $E[D] = 50$, $\eta = 10$

Figure A.4: The distribution of the average (avg), maximum (max), and minimum (min) distances between shortest path nodes and geodesics connecting shortest path endpoints in Waxman SRGGs with the connection probability function given by Eq. (2.4). For comparison, we also plot the distribution of distances to geodesic from randomly selected nodes (ran). The network size $N = 10^4$ and the parameter $\beta = 1$. For each expected degree $E[D]$ and parameter η, we generated an SRGG and considered 10^6 random shortest paths.

A.3. ONSET OF THE GIANT CONNECTED COMPONENT IN THE SRGG AS THE EXPECTED DEGREE INCREASE

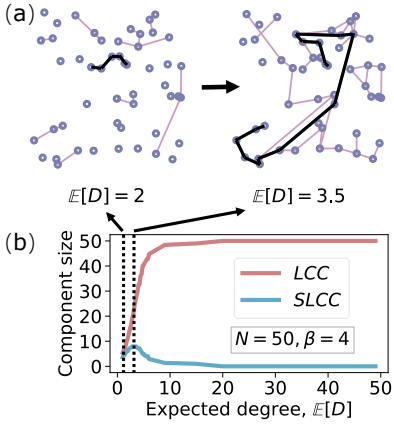


Figure A.5: (a) Visualizations of toy SRGGs with $N = 50$ nodes and inverse temperature $\beta = 4$ for increasing expected degree, from left to right: $E[D] = 2$, $E[D] = 3.5$. In each network, a representative shortest path is highlighted in black. (b) Average sizes of the largest connected component (LCC) and the second largest connected component (SLCC) for SRGGs with $N = 50$ and $\beta = 4$ as a function of the expected degree $E[D]$. The emergence of a giant connected component occurs at the critical expected degree $D_c = 3.5$, identified by the maximum of the SLCC. All LCC and SLCC values are averaged over 100 independent realizations. Accordingly, the network visualizations in panel (a) correspond to the disconnected and critical connected regimes.

A.4. THE AVERAGE LINK LENGTH

To derive the expression for the average link length $\langle d_l \rangle$ in SRGG models, we first define $\rho\left(D_{ij} = x | A_{ij} = 1\right)$ as the conditional probability density that the distance D_{ij} between two randomly chosen nodes in the SRGG equals x, provided these nodes are connected, $A_{ij} = 1$. The average link length is then given by $\langle d_l \rangle = \int_0^{\sqrt{2}} x \rho\left(D_{ij} = x | A_{ij} = 1\right) \mathrm{d}x$.

Using the Bayes' rule, the thought probability density function can be expressed as

$$\rho\left(D_{ij} = x | A_{ij} = 1\right) = \frac{P\left(A_{ij} = 1 | D_{ij} = x\right) \rho\left(D_{ij} = x\right)}{P\left(A_{ij} = 1\right)}, \tag{A.3}$$

where $P\left(A_{ij} = 1 | D_{ij} = x\right)$ is nothing else but the connection probability function $f(x)$, prescribed by Eq. (2.5), and $\rho\left(D_{ij} = x\right)$ is the distribution of distances between node pairs in a unit square. $P\left(A_{ij} = 1\right)$ is the probability that a randomly chosen node pair is connected, $P\left(A_{ij} = 1\right) = \frac{2L}{N(N-1)} = \frac{\langle D \rangle}{N-1}$.

The average link length is then given by

$$\langle d_l \rangle = \frac{\int_0^{\sqrt{2}} x f(x) \rho\left(D_{ij} = x\right) \mathrm{d}x}{\int_0^{\sqrt{2}} f(x) \rho\left(D_{ij} = x\right) \mathrm{d}x}. \tag{A.4}$$

A

We approximate the distribution of distances between node pairs in a unit square as $\rho\left(D_{ij}=x\right)=\frac{2x}{R^2}$, where R is the size of a unit square, $R=\mathcal{O}(1)$, obtaining

$$\langle d_l\rangle \approx d_0 \frac{\int_0^{(\sqrt{2}/d_0)^\beta} \frac{u^{3/\beta-1}}{1+u}\mathrm{d}u}{\int_0^{(\sqrt{2}/d_0)^\beta} \frac{u^{2/\beta-1}}{1+u}\mathrm{d}u}. \tag{A.5}$$

The closed-form expression for $\langle d_l\rangle$ in Eq. (A.5) is given by the ratio of two hypergeometric functions. It is useful, however, to infer its leading order behavior as a function of $\langle D\rangle$. To this end, we recall that $\frac{1}{d_0}\sim N^{1/2}\gg 1$ for large N. This follows from Eq. (2.6) for sparse SRGGs.

Since $\beta>2$, the integral in the denominator of Eq. (A.5) can be approximated as $\int_0^\infty \frac{u^{2/\beta-1}}{1+u}\mathrm{d}u=\pi/\sin\left(\frac{2\pi}{\beta}\right)$. Further, if $\beta>3$, the integral in the numerator of Eq. (A.5) can be approximated in a similar manner as $\int_0^\infty \frac{u^{3/\beta-1}}{1+u}\mathrm{d}u=\pi/\sin\left(\frac{3\pi}{\beta}\right)$. As a result,

$$\langle d_l\rangle \approx d_0 \frac{\sin\left(\frac{2\pi}{\beta}\right)}{\sin\left(\frac{3\pi}{\beta}\right)} \sim \langle D\rangle^{1/2}, \quad \beta>3. \tag{A.6}$$

In case $\beta\in(2,3)$, we approximate the integral in the numerator of Eq. (A.5) as $\int_0^{(\sqrt{2}/d_0)^\beta} u^{3/\beta-2}\mathrm{d}u$, which results in

$$\langle d_l\rangle \approx \frac{\pi\beta 2^{\frac{3-\beta}{2}}\sin\left(\frac{2\pi}{\beta}\right)}{3-\beta} d_0^{\beta-2} \sim \langle D\rangle^{\beta/2-1}, \quad 2<\beta<3. \tag{A.7}$$

B

APPENDIX OF CHAPTER 3

B.1. NOTATION FOR CHAPTER 3

Table B.1: Notation

B

Symbol	Definition
G	Graph
$\mathscr{N}$	Set of nodes
N	Number of nodes
$\mathscr{L}$	Set of links
L	Number of links
$\mathscr{P}_{ij}$	Path from node i to node j
$\mathscr{P}^*_{ij}$	Shortest path from node i to node j
D	Demand matrix
G_D	Complete graph whose weighted adjacency matrix equals the demand matrix D
A	Adjacency matrix
W	Link weight matrix
$\widetilde{A}$	Weighted adjacency matrix
$\widetilde{A}_F$	Weighted adjacency matrix of a flow network
Ω	Effective resistance matrix
r_l	Link l resistance
R_G	Effective graph resistance
S	Shortest path weight matrix
Q	Laplacian matrix
$Q^\dagger$	Pseudoinverse Laplacian matrix
$\widetilde{\Delta}$	Weighted degree matrix
$\widetilde{\Delta}_F$	Weighted degree matrix of a flow network
d_i	Degree of node i
u	All-one vector
J	All-one matrix
I	Identity matrix
e_i	$N \times 1$ basic vector has only one non-zero element $(e_i)_i = 1$

B.2. FLOW ANALOGUE METHOD FOR GENERAL GRAPHS

As demonstrated in Section 3.2.2, Van Mieghem[7] proposed the "flow analogue method" to recover the weighted adjacency matrix $\widetilde{A}$ from a demand matrix D, which is also a shortest path weight matrix S, by exploiting the analogy between flow and path networks, when the given shortest path weight matrix S derived from a tree graph. The crux of the "flow analogue method" lies in the strategic employment of the equivalence existing between the shortest path matrix S and the effective resistance matrix Ω. Since the equality between the shortest path weight matrix S and the effective resistance matrix Ω only holds for the tree graph, the given demand matrix must be transformed into an effective resistance matrix first.

Unfortunately, it remains an open question how to construct an effective resistance matrix whose corresponding weighted adjacency matrix is non-negative. In Appendix B.3, we try to extend the flow analogue method utilizing perturbation theory. We first obtain a perturbed effective resistance matrix of a graph by adding a noise matrix. The resulting perturbed effective resistance matrix is then imported into (3.8) and we can obtain a weighted Laplacian and the corresponding weighted adjacency matrix $\widetilde{A}$. Appendix B.3 demonstrates that even an extremely tiny perturbation of an effective resistance matrix of a tree graph may lead to negative off-diagonal elements in the obtained weighted adjacency matrix $\widetilde{A}$ computed with (3.12). Hence, the sum of two effective resistance matrices is not necessarily an effective resistance matrix.

The difficulty of constructing an effective resistance matrix with a non-negative corresponding weighted adjacency matrix can also be explained from the perspective of the inverse simplex[50] of a graph. The details of simplex and inverse simplex are provided in Appendix B.4. Any undirected, weighted graph G can be uniquely represented by a simplex $\mathcal{V}$ or an inverse simplex $\mathcal{V}^+$ in the $N-1$ dimensional Euclidean space [50]. The reverse also holds: Every simplex with non-obtuse angles, i.e. smaller than or equal to 90 degrees, between all pairs of facets is the inverse simplex $\mathcal{V}^+$ of a connected, undirected graph with positive link weights[50]. Fiedler [50, 114] demonstrated that the angle ϕ^+_{ij} between two facets $\mathscr{F}^+_i$ and $\mathscr{F}^+_j$ in an inverse simplex $\mathcal{V}^+$ is related to the graph G by

$$\cos(\phi^+_{ij}) = -\frac{\widetilde{q}_{ij}}{\sqrt{\widetilde{q}_{ii}\widetilde{q}_{jj}}} \tag{B.1}$$

where $\widetilde{q}_{ii}$ and $\widetilde{q}_{ij}$ are the diagonal and off-diagonal elements of the weighted Laplacian matrix $\widetilde{Q}$. As shown in (B.1), if there is no link between node i and j, then $\tilde{q}_{ij} = 0$ and the dihedral angle ϕ^+_{ij} exactly equals 90 degrees. For an inverse simplex of tree graph, there are $\frac{N(N-1)}{2} - (N-1)$ 90-degree dihedral angles. Moreover, a positive off-diagonal element $\widetilde{q}_{ij}$, which leads to a negative link weight, corresponds to an obtuse dihedral angle with more than 90 degrees. When we perturb the effective resistance matrix of a tree, the corresponding inverse simplex has a high probability of having obtuse dihedral angles because each 90-degree angle may become obtuse even with a tiny perturbation. This explains why a tiny perturbation on an effective resistance matrix of a tree graph may lead to negative off-diagonal elements in the obtained weighted adjacency matrix $\widetilde{A}$.

Fiedler[114] proposed that the effective resistance matrix Ω of a weighted graph G equals the squared Euclidean distance matrix D_E, for which $d_E(ij) = ||d_E(i) - d_E(j)||^2$ for the vertices coordinates $\{v_1, v_2, \ldots, v_N\}$ of the corresponding inverse simplex $\mathcal{V}^+$ in the Euclidean space. Constructing an effective resistance matrix whose corresponding weighted adjacency matrix is non-negative can thus be regarded as a problem in a graph: "Obtain the criterion for the edges in a simplex that does not have obtuse angles between arbitrary two facets", which is complicated.

B

B.3. FIRST-ORDER PERTURBATION ON EFFECTIVE RESISTANCE MATRIX

Consider a perturbed effective resistance $\Omega' = \Omega + \varepsilon K$, where Ω is an effective resistance of a graph G, matrix K is a symmetric noise matrix whose diagonal elements k_{ii} are zero and the off-diagonal elements are arbitrary real numbers and ε is an arbitrary insignificant positive real number.

The inverse Ω'^{-1} of the effective resistance matrix is then:

$$\begin{aligned}\Omega'^{-1} &= (\Omega + \varepsilon K)^{-1} = \left(\Omega\left(I + \varepsilon\Omega^{-1}K\right)\right)^{-1} \\ &= \left(I + \varepsilon\Omega^{-1}K\right)^{-1}\Omega_1^{-1}\end{aligned}$$

where $(AB)^{-1} = B^{-1}A^{-1}$ (see e.g. [165, p. 93]) is used. Invoking $(I+\varepsilon R)^{-1} = \sum_{k=0}^{\infty}(-1)^k\varepsilon^k R^k$ for sufficiently small $\varepsilon < \frac{1}{\lambda_{\max}(R)}$,

$$\Omega'^{-1} = \Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1} + O\left(\varepsilon^2\right)$$

and ignoring higher terms $O\left(\varepsilon^2\right)$ yields [8]

$$\Omega'^{-1} \approx \Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}$$

Recall (3.11): $\widetilde{Q} = \frac{2}{2\sigma^2}pp^T - 2\Omega^{-1}$, where $2\sigma^2 = \frac{1}{u^T\Omega^{-1}u}$ and the vector $p = \frac{1}{u^T\Omega^{-1}u}\Omega^{-1}u$, we then have

$$\widetilde{Q} = \frac{2}{u^T\Omega^{-1}u}\left(\Omega^{-1}uu^T\Omega^{-1}\right) - 2\Omega^{-1} \tag{B.2}$$

After substituting Ω^{-1} in (B.2) with Ω'^{-1}, we have (B.3).

$$\widetilde{Q}' \approx \frac{2}{u^T\left(\Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}\right)u}\left[\left(\Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}\right)uu^T\left(\Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}\right)\right] - 2\left(\Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}\right) \tag{B.3}$$

The left part of the first factor of (B.3) can be rewritten:

$$\begin{aligned}\frac{2}{u^T\left(\Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}\right)u} &= \frac{2}{u^T\Omega^{-1}u - \varepsilon u^T\Omega^{-1}K\Omega^{-1}u} \\ &= \frac{2}{u^T\Omega^{-1}u}\frac{1}{1 - \frac{\varepsilon u^T\Omega^{-1}K\Omega^{-1}u}{u^T\Omega^{-1}u}} \\ &\approx \frac{2}{u^T\Omega^{-1}u}\left(1 + \frac{\varepsilon u^T\Omega^{-1}K\Omega^{-1}u}{u^T\Omega^{-1}u}\right),\end{aligned} \tag{B.4}$$

where the approximation utilizes $\frac{1}{1-x} \approx 1 + x$ for small x.

The right part of the first factor of (B.3) is then (B.5):

$$\begin{aligned}&\left(\Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}\right)uu^T\left(\Omega^{-1} - \varepsilon\Omega^{-1}K\Omega^{-1}\right) \\ &= \Omega^{-1}uu^T\Omega^{-1} - \varepsilon\left(\Omega^{-1}K\Omega^{-1}uu^T\Omega^{-1} + \Omega^{-1}uu^T\Omega^{-1}K\Omega^{-1}\right) + O(\epsilon^2) \\ &\approx \Omega^{-1}uu^T\Omega^{-1} - \varepsilon\left(\Omega^{-1}K\Omega^{-1}uu^T\Omega^{-1} + \Omega^{-1}uu^T\Omega^{-1}K\Omega^{-1}\right)\end{aligned} \tag{B.5}$$

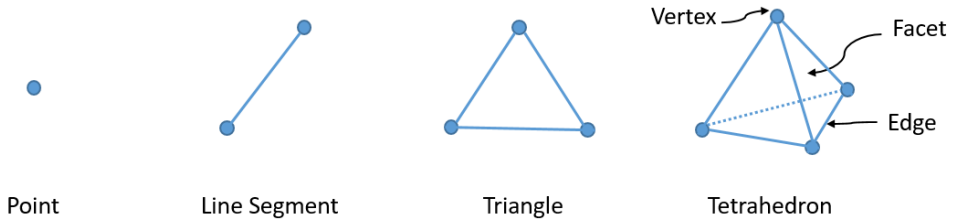


Figure B.1: Examples of low-dimensional simplices

With (B.4) and (B.5), (B.3) is then:

$$\widetilde{Q}' \approx \widetilde{Q} + \varepsilon\Delta\widetilde{Q} + O(\varepsilon^2) \approx \widetilde{Q} + \varepsilon\Delta\widetilde{Q}, \tag{B.6}$$

where $\Delta\widetilde{Q}$ is given by

$$\begin{aligned}\Delta\widetilde{Q} \approx\ & 2\Omega^{-1}K\Omega^{-1} + \frac{2u^T\Omega^{-1}K\Omega^{-1}u}{(u^T\Omega^{-1}u)^2}(\Omega^{-1}uu^T\Omega^{-1}) \\ & - \frac{2}{u^T\Omega^{-1}u}(\Omega^{-1}K\Omega^{-1}uu^T\Omega^{-1} + \Omega^{-1}uu^T\Omega^{-1}K\Omega^{-1}) \\ =\ & 2\Omega^{-1}K\Omega^{-1} + 2(u^T\Omega^{-1}K\Omega^{-1}u)pp^T - \frac{1}{\sigma^2}\left(\Omega^{-1}K\Omega^{-1}pp^T + pp^T\Omega^{-1}K\Omega^{-1}\right)\end{aligned} \tag{B.7}$$

B.4. SIMPLEX AND INVERSE SIMPLEX

Besides the adjacency matrix A, the Laplacian matrix Q and the effective resistance matrix Ω, which represent a graph in the topology domain, an undirected graph can be represented in the geometric domain by its corresponding simplex or inverse simplex [50]. Any undirected, weighted graph G can be uniquely represented by a simplex $\mathcal{V}$ or an inverse simplex $\mathcal{V}^+$ in the $N-1$ dimensional Euclidean space [50]. Both simplex $\mathcal{V}$ and inverse simplex $\mathcal{V}^+$ are geometric objects that generalizes triangles and tetrahedrons to any dimension [50]. For example, 0−simplex, 1−simplex, 2−simplex and 3−simplex respectively correspond to a point, a line segment, a triangle and a tetrahedron, as shown in Fig. B.1. The vertex matrix $V^=[v_1, v_2, \ldots, v_N]$, where v_i is a $N-1$ dimensional vector including the coordinates of the vertex i in an simplex $\mathcal{V}^+$, can be obtained from $\widetilde{Q} = V^TV$. In an inverse simplex $\mathcal{V}^+$, the vertex matrix $V^\dagger = [v_1^+, v_2^+, \ldots, v_N^+]$, where v_i^+ is a $N-1$ dimensional vector including the coordinates of the vertex i, can be calculated by $\widetilde{Q}^\dagger = V^{\dagger T}V^\dagger$, where $\widetilde{Q}^\dagger$ is the weighted pseudoinverse Laplacian matrix of G. The surface or boundary of a simplex $\mathcal{V}$ (or an inverse simplex $\mathcal{V}^+$) is called a face, which is a lower dimensional simplex[50]. A face in a simplex (or an inverse simplex) is denoted by $\mathcal{F}_\mathcal{E}$ (or $\mathcal{F}^+_\mathcal{E}$), where vector $\mathcal{E}$ is a subset of vertex indices of a simplex $\mathcal{V}$ (or an inverse simplex $\mathcal{V}^+$). If we use $\bar{\mathcal{E}}$ as the complementary set of vertex indices, $\mathcal{F}_\mathcal{E}$ and $\mathcal{F}_{\bar{\mathcal{E}}}$ are then called complementary faces. For example, in a 3-dimensional simplex (known as a tetrahedron), a 0-dimensional face corresponds to a vertex, while its complementary face is a triangle (2-dimensional face). Furthermore, a $N-2$ dimensional face is called

a facet and the dihedral angles between two facets of a simplex $\mathcal{V}$ (or an inverse simplex $\mathcal{V}^+$) are denoted by ϕ (or ϕ^+).

B

B.5. THREE NP-COMPLETE ISPP

B.5.1. A GENERAL INVERSE SHORTEST PATH PROBLEM STUDIED BY FEKETE ET AL.

In 1999, Fekete et al.[38] consider a general ISPP ($\text{ISPP}_{\text{Fekete}}$), where only the shortest path weight between pairs of nodes are given, but not the paths achieving them.

A general inverse shortest path problem studied by Fekete et al. ($\text{ISPP}_{\text{Fekete}}$): Given a N-node graph G with adjacency matrix A and a $N \times N$ symmetric and non-negative demand matrix D. Determine the link weight matrix W and the weighted adjacency matrix $\widetilde{A}$ such that the corresponding shortest path weight matrix S equals D.

In $\text{ISPP}_{\text{Fekete}}$, the demand matrix D does not necessarily include all the shortest path weights of the pairs of nodes in graph G. Fekete et al. [38] prove that $\text{ISPP}_{\text{Fekete}}$ is NP-complete by reducing $\text{ISPP}_{\text{Fekete}}$ to a vertex-disjoint paths problem, which is NP-complete. Specifically, $\text{ISPP}_{\text{Fekete}}$ is polynomial solvable if the distance graph $G_d(V, E_d, w_d)$ is a star (in which all the links are incident to a node) or the union of complete star (which is a star include all the nodes in the graph). The problem becomes NP-complete if slightly more complicated distance graphs $G_d(V, E_d, w_d)$ are considered.

B.5.2. FORTH PATH CUT PROBLEM

B. A. Miller et al. [166] investigated an ISPP aiming to obtain a graph, such that a given path is the shortest path, by removing links of the given graph. This problem is referred as Forth Path Cut problem (FPCP).

Forth path cut problem: Given a weighted graph G with link weight matrix W and a path $\mathscr{P}_{ij}$ for node i and j, determine a new weighted graph G' by removing links, such that $\mathscr{P}$ is the shortest paths in the graph and the sum of the weights of removed links does not exceed a given limitation b. B. A. Miller et al. [166] proved that the Force path cut problem is an NP-complete problem by reducing FPCP to 3-Terminal Cut problem, which is NP-complete. Given a graph G with weights matrix W and three terminal nodes, the 3-Terminal cut problem asks for removing a set of links $\mathscr{L}_{\text{R}} \in \mathscr{L}$ such that the terminals are disconnected (there is no path connecting any two terminals) in the resulting graph G' and the sum of weights of removed links $l \in \mathscr{L}_{\text{R}}$ does not exceed a given limitation b. The success condition of Force Path Cut is that all paths from nodes i to j aside from $\mathscr{P}$ must be strictly longer than $\mathscr{P}$, which is an example of the Weighted Set Cover problem. In Force Path Cut, the elements of the universe to cover are the paths and the sets represent links: each link corresponds to a set containing all paths from s to t on which it lies. Including this set in the cover implies removing the edge, thus covering the elements (i.e., cutting the paths). While the weighted cover is computationally intractable, there are established approximation algorithms. B. A. Miller et al.[166] then proposed an algorithm called path attack, which uses a natural oracle to generate only those constraints needed to execute the approximation algorithm.

B.5.3. ISPP WITH UPPER BOUND

Burton et al.[32] posed an ISPP with upper bound constraints, i.e. the weight of the shortest path $s_{ij} = w(\mathscr{P}^*_{ij}) \leq d_{ij}$, where d_{ij} is the element of the demand matrix D between nodes i and j, which is called "upper bound". The problem is described as

The inverse shortest path problem with upper bound constraints($\text{ISPP}_{\text{upperbound}}$): Given a graph G with link weight matrix W and an $N \times N$ symmetric demand matrix D with zero diagonal elements, but positive off-diagonal elements. Determine a new link weight matrix W', such that the corresponding shortest path weight matrix S obeys $S \preccurlyeq D$ and minimizes a norm $||W' - W||$.

In this case, the "upper bound" d_{ij} are allowed to be infinite, i.e. the constraint of "upper bond" is not necessary for all the shortest path weights. Burton et al.[32] choose l_2 norm and consider $\text{ISPP}_{\text{upperbound}}$ as a least squares problem

$$\min_{w'_l \in \mathscr{L}'} \frac{1}{2} \sum_{l=1}^{L} (w'_l - w_l)^2 \tag{B.8}$$

where w_l and w'_l denote the weight of link $l \in \mathscr{L}$ and $l \in \mathscr{L}'$ respectively, subject to

$$w_l \geq 0, l \in \mathscr{L} \tag{B.9}$$

and the bound constraints on the shortest paths,

$$\sum_{l \in \mathscr{P}^*_{ij}} w_l \leq d_{ij} \tag{B.10}$$

Equation (B.10) only consider the shortest path weight and the definition of the path $\mathscr{P}^*_{ij}$ is implicit. For example, as the link weight matrix W is modified, the nodes and links belonging to the shortest path $\mathscr{P}^*_{ij}$ between nodes i and nodes j may vary. Burton et al. [32] illustrated the non-convex nature of the "upper bound" constraints and the least squares problem describing $\text{ISPP}_{\text{upperbound}}$ is proved to be non-convex. The quadratic programming algorithm introduced in [31] thus cannot be utilized directly to solve $\text{ISPP}_{\text{upperbound}}$.

Burton et al.[32] then proved that finding a global solution of $\text{ISPP}_{\text{upperbound}}$ is NP-complete through a decision problem ISP: Given an $\text{ISPP}_{\text{upperbound}}$ with a bound k, does there exists a solution with objective value at most k? Burton et al. demonstrate that (a) the decision question ISP is in NP. (b) a transformation from 3-SAT problem [167] to ISP can be constructed (c) the transformation is proved to be polynomial. The decision question ISP is thus NP-complete and $\text{ISPP}_{\text{upperbound}}$ is NP-hard.

Since the $\text{ISPP}_{\text{upperbound}}$ is non-convex and NP-hard, Burton et al.[32] proposed a local optimum algorithm to solve the shortest path problem. A feasible starting point is computed first. Then at each iteration, with a modified weight matrix W', the explicit definition of the shortest path constraints(B.10) is revised and the resulting convex problem is solved by the algorithm introduced in [32], which leads a new modified weight matrix W'. The calculation stops when no further progress can be obtained. The algorithm terminates in a finite number of iterations.

B.6. PSEUDOCODE OF DOR INITIALISED WITH A TREE

In this section, we supplement the pseudocode of DOR algorithm initialised with a tree, which is shown in Algorithm 7.

Algorithm 7 Descending Order Recovery (DOR) initialised with a tree graph

Input: $N \times N$ demand matrix $D = S$: a shortest path weight matrix of a graph G
Output: $N \times N$ weighted adjacency matrix $\widetilde{A}$

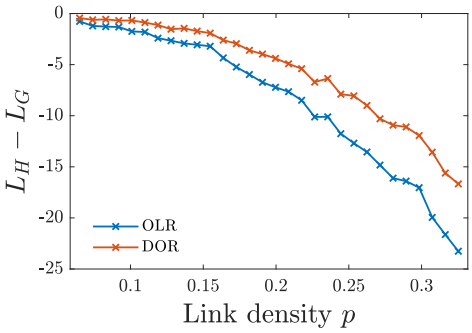
$\widetilde{A}_D \leftarrow D$
$G_D \leftarrow$ Complete graph whose weighted adjacency matrix is $\widetilde{A}_D$
$T_D \leftarrow$ Minimum spanning tree of G_D
$\widetilde{A} \leftarrow$ Weighted adjacency matrix of T_D
$S' \leftarrow$ Shortest path weight matrix of T_D
while $C \leftarrow D - S'$ has at least one negative element, i.e. $(c_{ab})_{\max} > 0$ **do**
    $(i, j) \leftarrow$ Indices of the maximum element in C
    $\widetilde{a}_{ij} \leftarrow d_{ij}$, $\widetilde{a}_{ji} \leftarrow d_{ij}$
    $G_A \leftarrow$ Graph whose weighted adjacency matrix is $\widetilde{A}$
    $S' \leftarrow$ Shortest path weight matrix of G_A
end while
return $\widetilde{A}$

B.7. PERFORMANCE OF THE DOR AND OLR ALGORITHM

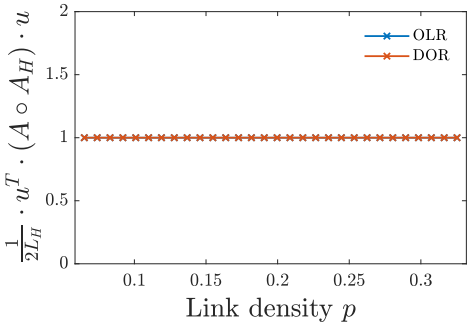
In this section, we supplement the performance of DOR and OLR in ER random network $G_p(N)$, Barabási–Albert (BA) network[115], Watts–Strogatz(WS) small world network[116] and an empirical network USAir[117]. The performance of the DOR and OLR is assessed by three complementary criteria: (i) the number $L_H - L_G$ of additional links in the resulting graph H, (ii) the number $L_C = \frac{1}{2L_H} \cdot u^T \cdot (A \circ A_H) \cdot u$ of common links in the original graph G and the resulting graph H and (iii) the norm $||D-S|| = \frac{1}{N(N-1)} \sum_i \sum_j \frac{d_{ij}-s_{ij}}{d_{ij}}$ of the demand matrix D and the shortest path weight matrix S.

Fig. B.2 illustrates the results for ER graphs with different network sizes. We uniformly assign a random weight from $(0,1)$ to each link in G, thus defining the $N \times N$ weighted adjacency matrix $\widetilde{A}$. For each generated ER graph, we provide the $N \times N$ shortest path weight matrix of G as the input demand matrix D to the algorithm DOR and OLR. The input parameter of OLR $b = 0.7$. We then obtain the resulting graph H, whose $N \times N$ shortest path weight matrix is denoted as S. For each number N of nodes and different link density p, 100 simulation instances are executed and the average over 100 times of each criterion is computed.

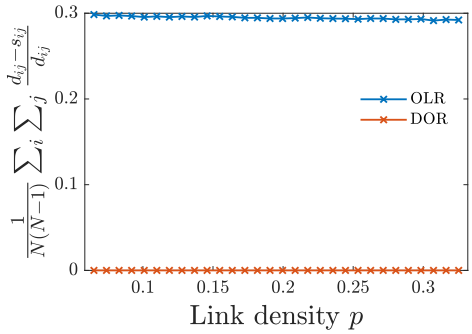
Table B.2 and Table B.3 respectively show the performances of DOR and OLR in BA network, WS network and an empirical network USAir. The node number N and link number L are shown in the tables. For each graph, the shortest path weight matrix are computed as the input demand matrix D for DOR and OLR. The input parameter of OLR $b = 0.3$. For BA networks, the degree of node $\Pr[\mathrm{d} = \mathrm{k}] \propto \mathrm{k}^{-\gamma}$, where $\gamma = 2.75, 2.36, 2.48$ and 2.37 in BA1, BA2, BA3 and BA4, respectively. The link weights are uniformly distributed in $(0,1)$. We generate WS networks as follows:



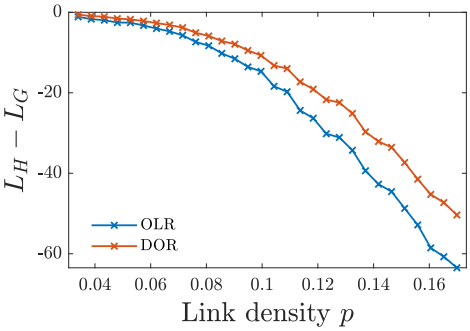
(a) Number of additional links in obtained graph H ($N = 20$).



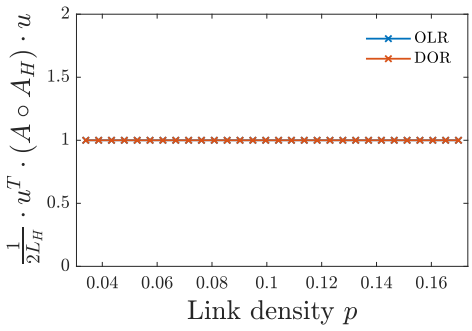
(b) Number of common links in G and H ($N = 20$).



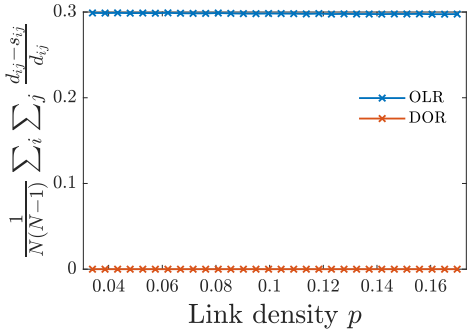
(c) Norm $||D - S||$ ($N = 20$).



(d) Number of additional links in obtained graph H ($N = 50$).



(e) Number of common links in G and H ($N = 50$).



(f) Norm $||D - S||$ ($N = 50$).

Figure B.2: Performance of the DOR and OLR on ER graphs with network size $N = 20$ (panels (a)–(c)) and $N = 50$ (panels (d)–(f)) with respect to link density p. The input parameter $b = 0.7$.

B

Table B.2: Performance of the DOR

Network	N	L_G	$L_H - L_G$	L_C	$\|\|D-S\|\|$
BA1	500	994	-72	1	0
BA2	500	1979	-750	1	0
BA3	1000	1993	-76	1	0
BA4	10000	19989	0	1	0
WS1	100	200	-9	1	0
WS2	100	300	-65	1	0
WS3	1000	3000	-244	1	0
WS4	10000	30000	0	1	0
USAir	332	2126	-279	1	0

Table B.3: Performance of the OLR

Network	N	L_G	$L_H - L_G$	L_C	$\|\|D-S\|\|$
BA1	500	994	-181	1	0.6736
BA2	500	1979	-853	1	0.6938
BA3	1000	1993	-140	1	0.6961
BA4	10000	19989	-36	1	0.6939
WS1	100	200	-39	1	0.6865
WS2	100	300	-157	1	0.6642
WS3	1000	3000	-995	1	0.6850
WS4	10000	30000	-7	1	0.6999
USAir	332	2126	-1658	1	0.6481

1. Create a ring lattice with N nodes of the mean degree $2k$.
2. Each node is connected to its k nearest neighbors on either side.
3. For each edge in the graph, rewire the target node with probability $p = 0.5$

For WS1, WS2, WS3 and WS4, the corresponding $k = 2, 3, 3$ and 3, respectively. The link weights are uniformly distributed in $(0, 1)$.

C

APPENDIX OF CHAPTER 4

C.1. NOTATION FOR CHAPTER 4

Table C.1 summarizes symbols used in this paper.

Table C.1: Symbols

Symbol	Definition
G	Graph/network
$\mathscr{N}$	Set of nodes
N	Number of nodes
$\mathscr{L}$	Set of links
L	Number of links
$l = i \sim j$	Link between node i and node j
D	Demand matrix
A	Adjacency matrix
W	Link weight matrix
$\widetilde{A}$	Weighted adjacency matrix
$\widetilde{Q}$	Weighted Laplacian matrix
$Q^{\dagger}$	Pseudoinverse Laplacian matrix
$\widetilde{\Delta}$	Weighted degree matrix for path graph
d_i	Degree of node i
$\mathscr{P}_{ij}$	Path from node i to node j
$\mathscr{P}^*_{ij}$	Shortest path from node i to node j
Ω	Effective resistance matrix
S	Weighted shortest path weight matrix

C.2. IBA AS AN EXACT SOLUTION TO IASPP$_M$

In this section, we prove that the obtained graph G' by IBA is an exact solution to IASPP$_M$. We start with two properties of the irreducible backbone:

Property 4. *Each link $i \sim j$ belonging to the irreducible backbone G_b must be included in any graphs G whose shortest path weight matrix is identical to that of the irreducible backbone. The weight of each link in graph G is identical to its weight in G_b.*

Proof. We prove Property 1 by contradiction. Suppose that there is a backbone link $l = i \sim j$ that does not exist in a graph G which has the same shortest path matrix S as the backbone. The backbone G_b contains no redundant links. For each link $l = i \sim j$ in backbone G_b, the link weight is $(w_b)_{ij} = s_{ij}$. Since link $l = i \sim j$ does not exist in graph G, there is at least one node k in G such that $s_{ik} + s_{kj} = s_{ij}$. Because graph G and backbone G_b have the identical shortest path weight matrices, the equality $s_{ik} + s_{kj} = s_{ij}$ also holds in G_b, implying that link $l = i \sim j$ is redundant in the irreducible backbone G_b, which is impossible.

Furthermore, if the weight of a link w_{ij} in G differs from $(w_b)_{ij}$, two possibilities shall occur. If the link weight $w_{ij} < (w_b)_{ij}$ in G, then the shortest path weight $s_{ij} > w_{ij}$,

which is impossible. If $w_{ij} > (w_b)_{ij}$, there must exist at least one node k in G such that $s_{ik} + s_{kj} = s_{ij}$, which, as shown above, can not occur. Therefore, all links forming the irreducible backbone must exist in any graph that has the same shortest path weight matrix and their weights remain unchanged. □

From Property 4, we can conclude:

Property 5. *The links constituting the irreducible backbone are "not" redundant in an arbitrary graph G that has the same shortest path weight matrix as the irreducible backbone.*

If a link in the irreducible backbone is redundant, there would be at least one node k in graph G such that $s_{ik} + s_{kj} = s_{ij}$. However, the existence of such a node k has been proved to be impossible from Property 4. We therefore close the proof for Property 5 and introduce Property 6.

Property 6. *Given a weighted adjacency matrix $\widetilde{A}$ and the demand matrix D, the obtained graph G' by IBA obeys the shortest path weight matrix $S' = D$ and has minimum sum of link weights adjustment $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$.*

Proof. The proof consists of two parts: the graph G' obtained by IBA should (i) obey the shortest path weight matrix $S' = D$ and (ii) have minimum sum of link weights adjustment $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$. The first part is relatively straightforward: By design, IASPP$_M$ obtains the graph G' by adding redundant links to an irreducible backbone G_b. The corresponding shortest path weight matrix $S = D$ of the irreducible backbone [8] and adding redundant links does not change the shortest path weight matrix. The shortest path distance matrix thus satisfies $S' = D$ for the obtained graph G'.

Next, we prove $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$ is minimized in graph G' by contradiction. The sum $\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$ reaches its minimum when $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$ is minimized for each node pair (i, j), since $|\widetilde{a}'_{ij} - \widetilde{a}_{ij}| \geq 0$ for all (i, j). Suppose that there exists a graph G'' such that the corresponding shortest path weight matrix $S'' = D$ and the sum of link weight adjustment $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$ is smaller compared to G'. Then there must be at least one node pair (i, j) such that $|\widetilde{a}''_{ij} - \widetilde{a}_{ij}| < |\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$, which implies that $\widetilde{a}'_{ij} \neq \widetilde{a}_{ij}$. There are 4 possible cases:

1. $\widetilde{a}'_{ij} > 0$ and $\widetilde{a}_{ij} = 0$: In this case, link $l = i \sim j$ exists in the obtained graph G' but not in the original graph G, which implies that link l belongs to the irreducible backbone and is *not* a redundant link in G' (Property 5). Hence, $\widetilde{a}'_{ij} = d_{ij}$. If $|\widetilde{a}''_{ij} - \widetilde{a}_{ij}| < |\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$, then $\widetilde{a}''_{ij} < \widetilde{a}'_{ij} = d_{ij}$. This is impossible because a link weight w''_{ij} can not be smaller than the shortest path weight s''_{ij} in graph G''.

2. $\widetilde{a}'_{ij} = 0$ and $\widetilde{a}_{ij} > 0$: In this case, link $l = i \sim j$ exists in the original graph G but not in the obtained graph G', as determined by lines 8-9 in Algorithm 4 and the link weight satisfies $2w_{ij} < d_{ij}$. If $|\widetilde{a}''_{ij} - \widetilde{a}_{ij}| < |\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = \widetilde{a}_{ij}$, then $\widetilde{a}''_{ij} \in (0, 2\widetilde{a}_{ij}) < d_{ij} = s''_{ij}$, which is not possible.

3. $\widetilde{a}'_{ij} > \widetilde{a}_{ij} > 0$: In this case, link $l = i \sim j$ exists in both the original graph G and the obtained graph G'. The link $l = i \sim j$ in G' either belongs to the irreducible

backbones or is added according to lines 6-7 in Algorithm 4. In both situations, the corresponding weighted adjacency matrix element $\widetilde{a}'_{ij} = d_{ij} > \widetilde{a}_{ij}$. If $|\widetilde{a}''_{ij} - \widetilde{a}_{ij}| < |\widetilde{a}'_{ij} - \widetilde{a}_{ij}| = d_{ij} - \widetilde{a}_{ij}$, then $\widetilde{a}''_{ij} < d_{ij} = s''_{ij}$, which is not possible.

4. $\widetilde{a}_{ij} > \widetilde{a}'_{ij} > 0$: In this case, link $l = i \sim j$ exists in both the original graph G and the obtained graph G'. Since $\widetilde{a}_{ij} > \widetilde{a}'_{ij}$, link $l = i \sim j$ in G' belongs to the irreducible backbones. According to Property 4, such a backbone link $l = i \sim j$ exists in all graphs that have the same shortest path weight matrix S and the link weight w_l remains unchanged, implying that finding a $\widetilde{a}''_{ij} \neq \widetilde{a}'_{ij}$ is impossible.

C

All 4 cases are proved to be impossible. This concluded that $\frac{1}{2}\sum_{i,j}|\widetilde{a}'_{ij} - \widetilde{a}_{ij}|$ is minimized in graph G' □

C.3. SOLVING IASPP$_F$ WITH CLASSICAL LINEAR PROGRAMMING

Consider an undirected N-node tree graph. For a tree, there is only one path, i.e., the shortest path, between an arbitrary pair of nodes. Since the adjacency matrix is given, for each node pair (i, j), we can obtain the constraint of the shortest path distance $s_{ij} = w(\mathscr{P}^*_{ij}) = \sum_{(a\sim b)\in\mathscr{P}^*_{ij}} w_{ab}$, which is determined by the weights of links belonging to the shortest path P^*_{ij}. IASPP$_F$ then becomes an optimization problem of finding a link weight matrix W while minimizes the difference between the shortest path weight matrix S and the given demand D:

$$\begin{aligned} \min \quad & \sum_{i,j} |d_{ij} - s_{ij}| \\ \text{s.t.} \quad & w_{ab} > 0, \forall (a \sim b) \in \mathscr{L}, \\ & s_{ij} = \sum_{(a\sim b)\in\mathscr{P}_{ij}} w_{ab}, \quad \forall i < j. \end{aligned} \tag{C.1}$$

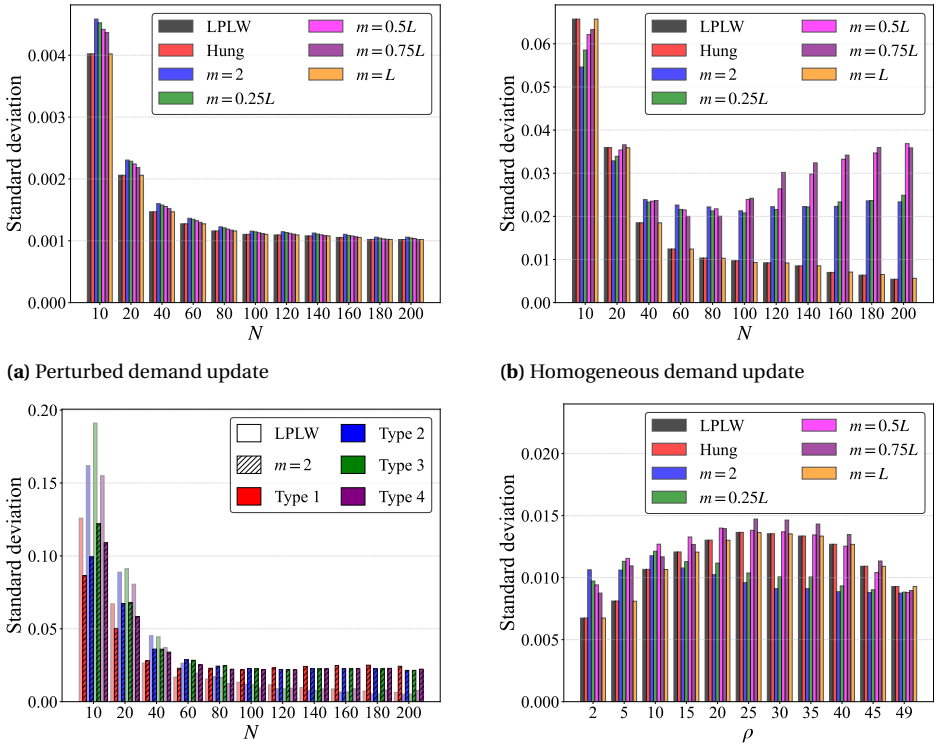
We rewrite Eq. (C.1) in linear programming (LP) standard form by introducing auxiliary variables $z_{ij} \geq d_{ij} - s_{ij}$ and $z_{ij} \geq s_{ij} - d_{ij}$:

$$\begin{aligned} \min \quad & \sum_{i,j} z_{ij} \\ \text{s.t.} \quad & z_{ij} \geq d_{ij} - \sum_{(a,b)\in\mathscr{P}^*_{ij}} w_{ab}, \quad \forall i < j, \\ & z_{ij} \geq \sum_{(a,b)\in\mathscr{P}^*_{ij}} w_{ab} - d_{ij}, \quad \forall i < j, \\ & w_{ab} > 0, \quad \forall (a \sim b) \in L. \end{aligned} \tag{C.2}$$

According to Eq. C.2, IASPP$_F$ is a classical LP problem with $N(N-1)+L = (N+1)(N-1)$ constraints and $\frac{N(N-1)}{2} + L = \frac{(N+2)(N-1)}{2}$ variables. We refer to the method for solving IASPP$_F$ using Eq. C.2 as "linear programming with link weights as variables (LPLW)". If all link weights are required to be integers, then IASPP$_F$ introduced by Eq. (C.2) becomes a classic integer programming (IP) problem.

C.4. STANDARD DEVIATION OF THE NORM FOR VARYING DEMAND TYPES

In Figs C.1a, C.1b and C.1c, we plot the standard deviation of the norm in ER graphs for the three demand types described in Section 4.5. In Fig. C.1d, the standard deviation of the norm in tree graphs are analyzed for the heterogeneous demand update scheme.



(a) Perturbed demand update

(b) Homogeneous demand update

(c) Heterogeneous demand update

(d) Tree with different diameter

Figure C.1: Standard deviation associated with Figs. 4.2a, 4.2c, 4.4a, and 4.5a. (a)–(c) plot the standard deviation of $\|D-S\|$ as a function of (ER) graph size N, for the perturbed demand update scheme, homogeneous demand update scheme, and heterogeneous demand update scheme, respectively. Four types of heterogeneous demand matrices (Type 1–4), as defined in Table 4.1, are evaluated in (c). Hung's scheme is excluded since it produces the same results as LPLW. (d) presents the standard deviation of $\|D-S\|$ as a function of the diameter ρ of the given tree graph. The graph size is $N=50$. All simulations in (a)–(d) are repeated 1000 times.

In Figs. 4.2a, 4.2c, we've seen that the norm $\|D-S\|$ of LPLW, Hung's scheme and DBS decrease as the graph size N increases, and that DBS with $m=2$ has the largest difference between the shortest path weight matrix and the demand. The standard deviation associated with $\|D-S\|$ shows a similar pattern for the perturbed demand update scheme in Fig. C.1(a). First of all, DBS with $m=2$ exhibits a higher standard deviation compare to LPLW and Hung's scheme. This is because, to improve computational efficiency, DBS

($m = 2$) reduces the number of optimization variables and thus the degrees of freedom available in the optimization process. The reduction in the degrees of freedom results in larger difference in $\|D - S\|$, thereby, leading to a higher standard deviation. Furthermore, the standard deviation of DBS, LPLW and Hung's scheme decrease as the size of the graph grows. This is because in small graphs (e.g., $N < 50$), paths with extremely high or low demand d_{ij} may produce significant differences in $|d_{ij} - s_{ij}|$, leading to a larger norm $||D-S||$. The existence of such paths amplifies the variability across different IASPP$_F$ instances, i.e., an instance in a small graph may contain several aforementioned paths, while another instance of the same size may contain none. Thus, the standard deviation appears to be larger in small graphs.

In Fig. C.1b, we see that the standard deviation (of the norm $||D - S||$) of the DBS variants behaves noticeably differently with the homogeneous demand update scheme. DBS with $m = 2$ and $0.25L$ decreases as the graph size grows and stabilizes after $N > 40$. The standard deviation of DBS with $m = 0.5L$ and $0.75L$ starts to increase as the graph size scales, although the average value of $||D - S||$ decreases as N grows, see Fig. 4.2c. This interesting observation could be attributed to the fact that the homogeneous demand update scheme handles arbitrarily generated high and low demands. Therefore, increases the likelihood of obtaining either very small or relatively large norms. As described in Section 4.5, DBS with $m = L$ consistently generates solutions close to the benchmark scheme of LPLW. The standard deviation associated with it is also close to that of LPLW and Hung's scheme.

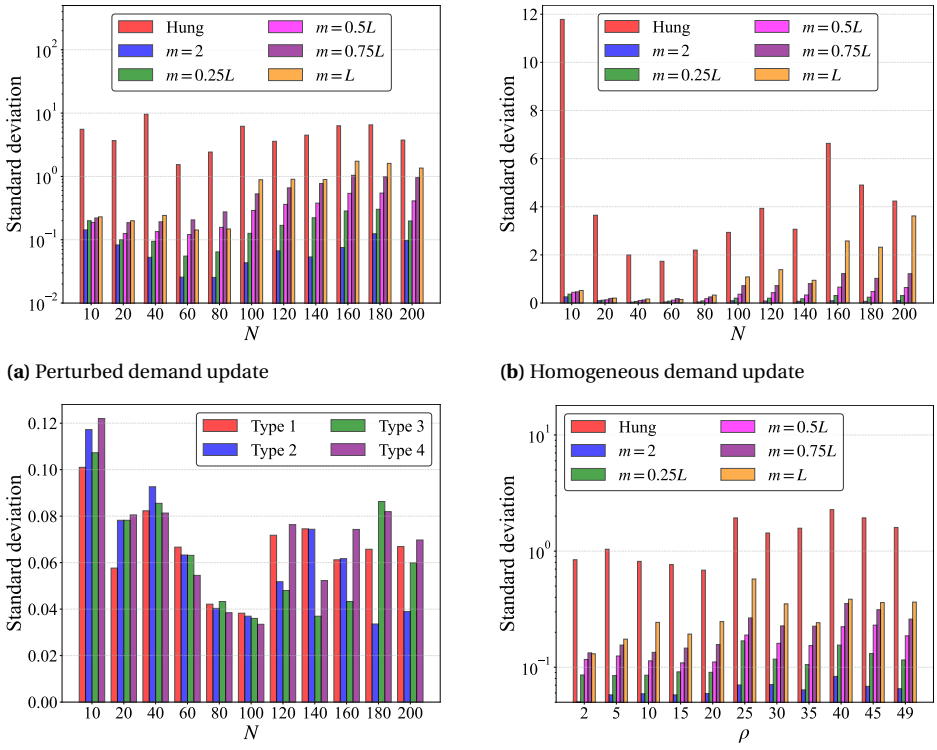
In Fig. C.1c, we compare the standard deviation of $\|D - S\|$ with respect to LPLW and DBS ($m = 2$) for the four types of demand defined in Section 4.5.2. Together with Fig 4.4a, we see that LPLW minimizes both the norm and its standard deviation for all types of demand in a larger graph (e.g., $N > 40$). Fig. C.1d plots the standard deviation of the norm $||D - S||$ as a function of the diameter ρ of a tree. When the diameter is small (i.e., $\rho = 2$ in a star graph) or approaches $N - 1$ (i.e., a path graph), the shortest path structures possess a higher structural regularity compared to those of general trees ($2 < \rho < N-1$). For a fixed N, all star graphs (or all path graphs) are mutually isomorphic. The structural similarity for graphs with either very small or very large diameter suppresses the fluctuations of different IASPP$_F$ instances, resulting in a small standard deviation.

C.5. STANDARD DEVIATION OF THE AVERAGE COMPUTATION TIME FOR VARYING DEMAND TYPES

This section evaluates the standard deviation of the (normalized) average computation time for DBS and Hung's scheme. LPLW has a standard deviation of zero since its computation time is normalized against its own average in Section 4.5.1. We, therefore, do included LPLW in Fig. C.2(a)-(d). As shown in Fig. C.2a, C.2b and C.2d, Hung's algorithm exhibits the highest standard deviation among all schemes. DBS ($m = 2$) achieves 10x reduction of the standard deviation compared to Hung's algorithm, see Fig. C.2a and C.2d. As the distance bases m increases, the variability of DBS also scales. This is attributed to the complexity of m introduced in Eq. (4.2), i.e., a larger m makes Eq. (4.2) more complicated, thus, incurs higher computation time. Together with the observations in Section 4.5.1, 4.5.2 and 4.5.3, we confirm the computational efficiency and sta-

tistical stability of DBS ($m = 2$) across all simulation settings.

In Fig. C.2c, we focus on the standard deviation of the computation time of DBS ($m = 2$), for the four types of demand defined in Table 4.1. Hung's scheme is omitted from the figure, since it exhibits a significantly larger standard deviation compared to DBS ($m = 2$). For a given graph size N, DBS ($m = 2$) demonstrates comparable performance across different types of demands. This indicates that the performance of DBS ($m = 2$) is insensitive to demand patterns, thus, is able to deliver stable performance under heterogeneous scenarios,



(a) Perturbed demand update

(b) Homogeneous demand update

(c) Heterogeneous demand update

(d) Tree with different diameter

Figure C.2: Standard deviation associated with Figs. 4.2b, 4.2d, 4.4b, and 4.5b. (a)–(c) plot the standard deviation of the average computation time t/t_{LPLW} normalized by LPLW as a function of graph size N, for the perturbed demand update scheme, homogeneous demand update scheme, and heterogeneous demand update, respectively. In (c), DBS ($m = 2$) is evaluated for four types of heterogeneous demand matrices (Type 1–4), as defined in Table 4.1. For graphic clarity, Hung's scheme is not included. (d) presents the standard deviation of the average computation time t/t_{LPLW} normalized by LPLW as a function of the diameter ρ of the given tree graph. The standard deviation of LPLW remains zero across all simulation scenarios and is therefore not presented. The graph size is $N = 50$. All simulations in (a)–(d) are repeated 1000 times.

C.6. PERFORMANCE EVALUATION OF DBS ON A REAL-WORLD TREE NETWORK

This section evaluates the performance of DBS in a factory automation network [132, 133]. The network consists of Ethernet switches (SW) and end devices (EN). End devices are connected to switches that forward the messages from a sending end device, either to the receiving end device, or to another switch in closer proximity to the receiving device [132]. Switches may be connected to each other, forming an arbitrary physical network topology (e.g., a line, a ring, or a mesh network), A logical tree can be created on top of the physical network topology by executing network protocols such as the spanning tree protocol. The tree network used for testing our DBS algorithm consists of 10 Ethernet switches and 20 end devices, see Fig. C.3.

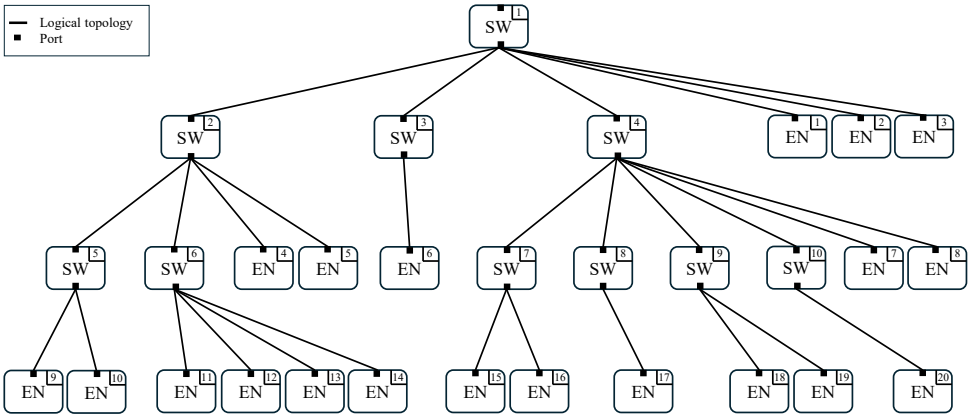


Figure C.3: A logical tree topology generated with a real-world factory automation network.

Table C.2: Comparison of $\|D-S\|$ and computation time under different methods

Method	$\|D-S\|$	Average computation time	Maximum computation time
LPLW	**0.3094**	0.0369	0.0859
Hung	0.3094	1.3102	1.7418
$m=2$	0.3906	**0.0259**	**0.0495**
$m=0.25L$	0.3718	0.0401	0.0818
$m=0.5L$	0.3487	0.0569	0.1141
$m=0.75L$	0.3329	0.0708	0.1754
$m=L$	0.3095	0.0967	0.1600

We generate 1000 IASPP$_F$ instances (i.e., 1000 demand matrices) on the Ethernet network. The input demand matrices D are the homogeneous demand introduced in Section 4.5.1. For each instance, we execute LPLW, Hung's approach and DBS with different numbers of distance bases. The results are summarized in Table C.2 and Fig. C.4.

Compared to LPLW, DBS with $m=2$ shows approximately a 30% reduction in average computation time and a 40% reduction in maximum time, at the cost of an increased

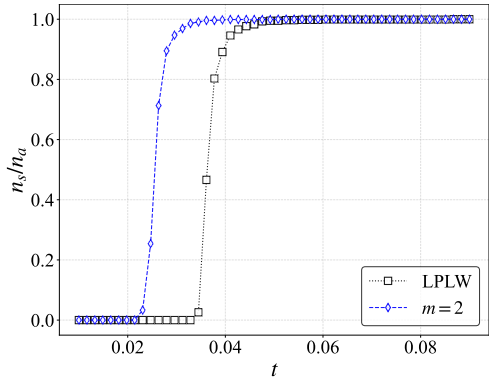


Figure C.4: Percentage of successfully scheduled instances n_s/n_a as a function of time limit t, where $n_a = 1000$ denotes the number of simulations.

norm $\|D-S\|$ by around 0.08. As plotted in Fig. C.4, DBS can finish all 1000 IASPP$_F$ instances within 0.05 seconds, while LPLW requires around 0.086 seconds to finish all 1000 IASPP$_F$ instances. These findings align with observations from various network topologies and demand update strategies presented in Section 4.5 and further demonstrate the superior performance of DBS in a real-world network.

D

APPENDIX OF CHAPTER 5

D.1. NOTATION FOR CHAPTER 5

Table D.1: Notation

Symbol	Definition
G	A network or graph
$\mathscr{N}$	Set of nodes
N	Number of nodes
$\mathscr{L}$	Set of links
L	Number of links
$l = i \sim j$	a link connecting node i and j
$\mathscr{P}_{ij}$	Path from node i to node j
$\mathscr{P}^*_{ij}$	Shortest path from node i to node j
A	Adjacency matrix
W	Link weight matrix
$\widetilde{A}$	Weighted adjacency matrix
B	Incidence matrix
Q	Laplacian matrix
$Q^\dagger$	Pseudoinverse Laplacian matrix
Ω	Effective resistance matrix
R_G	Effective graph resistance
r_l	Resistance of link l
S	Shortest path weight matrix
d_i	Degree of node i
$\widetilde{\Delta}$	Weighted degree matrix
u	All-one vector
J	All-one matrix
I	Identity matrix
e_i	$N \times 1$ basic vector has only one non-zero element $(e_i)_i = 1$
x_i	External current injected into node i.
v_i	Potential or voltage potential of node i
y_l	Current through link l
$y_l(s,t)$	Current through link l with node s as source and t as destination
P_G	Power dissipation in a network G
P_l	Power dissipated on link l
b_i	Betweenness of node i
c_i	Current-flow betweenness of node i
G^*_{ij}	Flow subgraph with (i,j) as source-destination node pair
$\mathscr{B}$	Backbone subgraph in the flow subgraph
T_k	k-th brach in the flow subgraph
ρ_N	Relative size of the flow subgraph
ρ_L	Fraction of link number of the flow subgraph normalized by link number of the graph

D.2. POWER DISSIPATION ON ER GRAPHS

In an electrical circuit network, the current flowing through each link leads to power dissipation because of the resistors. Similarly, the propagation of information across the links in a social network also requires "cost" or "power" consumption. In this section, we focus on the power dissipation problem in flow networks, where we draw an analogy to electrical circuit networks.

Recall Eq. (5.10), the power dissipation P_G equals the product of the square of the current and the effective resistance. For simplicity, the injected current I_c is considered to be unity in the following analysis.

According to the parallel circuit rules, the upper bound of the effective resistance ω_{ij} is the weight of the shortest path $\mathscr{P}^*_{ij}$, where the equality holds if there is only one path between node i and j. Hence, compared with the path network, where the transportation of items generally follows the shortest path, the flow network can transfer a unit of items with less total energy.

We now investigate the power dissipation on each link $l = i \sim j$ given node pair (s,t) as the source and destination node. Substituting the current y_l with Eq. (5.8), the link level power dissipation $P_l = y_l^2 r_l$ then becomes

$$\begin{aligned} P_l &= \frac{1}{r_l}[(e_i - e_j)^T \widetilde{Q}^\dagger (e_s - e_t)]^2 \\ &= \frac{1}{r_l}\left(\widetilde{q}^\dagger_{it} + \widetilde{q}^\dagger_{js} - \widetilde{q}^\dagger_{is} - \widetilde{q}^\dagger_{jt}\right)^2 \end{aligned}$$

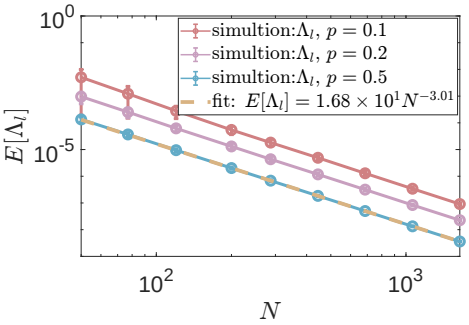
We can further relate the power dissipated on each link with the effective resistance by substituting the corresponding elements of the pseudoinverse Laplacian $\widetilde{Q}^\dagger$ using Eq. (3.4):

$$\begin{aligned} P_l &= \frac{1}{r_l}\left(\widetilde{q}^\dagger_{it} + \widetilde{q}^\dagger_{js} - \widetilde{q}^\dagger_{is} - \widetilde{q}^\dagger_{jt}\right)^2 \\ &= \frac{1}{4r_l}\left(\omega_{it} + \omega_{js} - \omega_{is} - \omega_{jt}\right)^2 \end{aligned} \tag{D.1}$$

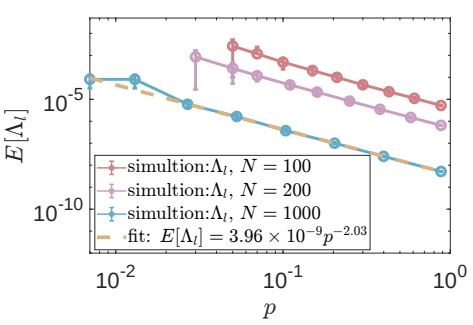
However, determining a closed-form formula for the distribution of the power dissipated on a link is mathematically challenging, as it involves dependencies among the effective resistance $\omega_{it}, \omega_{js}, \omega_{is}$ and ω_{jt} in Eq. (D.1), even in unweighted networks. We therefore consider the expected link power dissipation $E[P_l]$ in unweighted ER graphs.

To further investigate the expected dissipation $E[P_l]$, we execute simulations as follows:

1. Generate an ER graph $G_p(N)$.
2. For a node pair (s,t), we inject a unit current $I_c = 1A$ from node s (source) and let it out from node t (destination).
3. Compute the energy dissipation P_l for each link l and the expected energy dissipation $E[P_l]$.
4. Repeat steps 2-3 for all possible node pairs (i,j)



(a) Link power dissipation vs network size



(b) Link power dissipation vs link connection probability

Figure D.1: (a) The expected link power dissipation as a function of the network size N for different link connection probability p of unweighted ER networks. (b) The expected link power dissipation as a function of the link connection probability p of ER networks for different network size N.

The simulation is repeated 100 times for $N < 1000$ and 1 time for $N \geq 1000$. As shown in Figure D.1a and Figure D.1b, the expected link power dissipation $E[P_l]$ decreases with increasing probability p and network size N, following approximately power-law scalings with exponents close to -2 and -3, respectively.

The expected link power dissipation can be computed as the graph power dissipation divided by the number of links:

$$E[P_l] = \frac{P_G}{L} = \frac{\omega_{st}}{L} \tag{D.2}$$

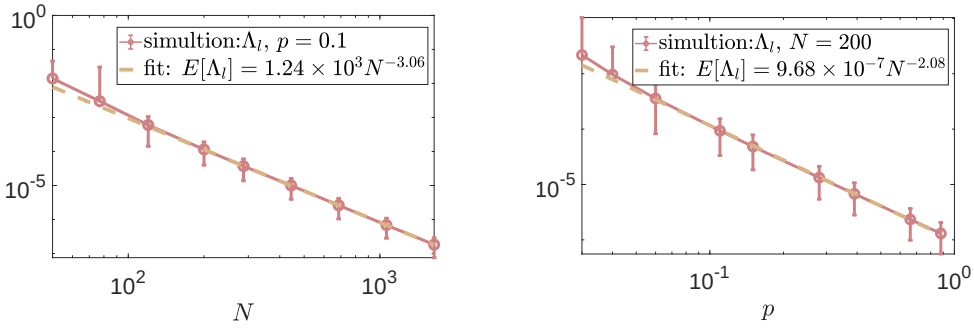
In an ER graph, the effective resistance [41, 51, 160, 161, 162] can be approximated as $\omega_{st} \approx \frac{1}{d_s} + \frac{1}{d_t}$ when the network is dense enough (the expected degree $E[D]$ is comparable or larger than $\log N$). The degrees d_i in such a dense ER network are concentrated around the expected degree $E[D]$. According to the permutation symmetry of the ER ensemble and mean field theory, one can obtain $\omega_{st} \approx \frac{2}{E[D]}$, so that Eq. (D.2) can be approximated as

$$E[P_l] \sim \frac{1}{E[D]^2 N} \sim \frac{1}{N^3 p^2} \tag{D.3}$$

which explains the power-law decrease of the expected link power dissipation $E[P_l]$ as a function of network size N or the average degree $E[D]$.

The power-law relation between the expected dissipation $E[P_l]$ and network size N or expected degree $E[D]$ can be further extended to weighted ER networks with link weights uniformly distributed in $[a, b]$. The effective resistance $\omega_{st} \approx \frac{1}{d_s} + \frac{1}{d_t}$ states that the effective resistance and the transport in flow networks are mainly determined by the links adjacent to the source node s and destination t, while the rest of the network is practically a perfect conductor. Since the link weights w_l are drawn independently and uniformly, the network excluding the source and destination can still be regarded as a perfect conductor, which leads to the effective resistance $\omega_{st} \approx E[w_l](\frac{1}{d_s} + \frac{1}{d_t})$, where the link weight w_l is the reciprocal of the corresponding resistance in the flow network. In

figure D.2, we supplement simulations for the power-law scaling of the expected dissipation $E[P_l]$ with respect to the network size N or expected degree $E[D]$.



(a) Weighted ER networks $G_{0.1}(N)$ **(b)** Weighted ER networks $G_p(200)$

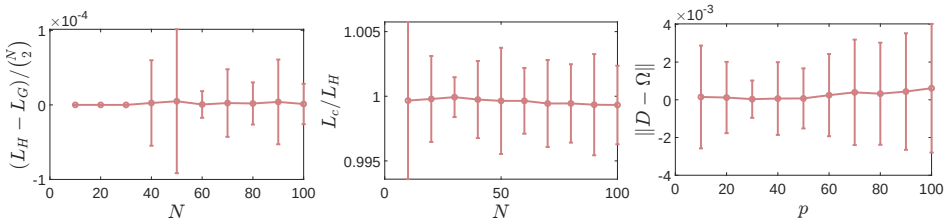
Figure D.2: (a) The expected link power dissipation as a function of the network size N for ER networks $G_{p=0.1}(N)$. (b) The expected link power dissipation as a function of the probability p for ER networks $G_p(N = 200)$. For each simulation, we generate an ER graph $G_p(N)$ with link weights uniformly distributed in $(0, 1)$. We compute the link power dissipation for all possible node pairs and obtain the expected value $E[\Lambda_l]$. The simulation is repeated 100 times for $N < 1000$ and once for $N \geq 1000$. The power dissipated on each link in weighted Erdős–Rényi networks exhibits a power-law decay as a function of network size or link density, with exponents of -3 and -2, respectively.

D.3. PERFORMANCE OF RGP ALGORITHM

Table D.2: Performance of RGP with empirical networks as baseline

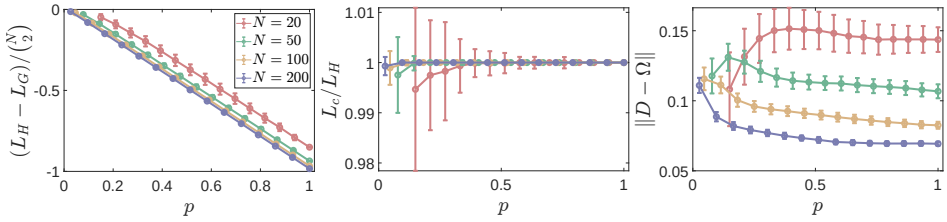
Network	N	L_G	$L_H - L_G$	L_c/L_H	$\|\|D-S\|\|$
karate	34	78	-35	1	0.1533
dolphins	62	159	-79	1	0.1273
NewSpainTravelMap	224	241	-13	1	0.0491
wiki-vote	889	2914	-1745	1	0.1045

In this section, we supplement the performance of RGP in tree graphs, ER random graphs $G_p(N)$ with exponentially distributed link weights and four empirical networks [117, 158]. We use the demand matrix $D = \Omega_G$ as input, where Ω_G is the effective resistance of a randomly generated ER graph or an empirical network G. The performance of RGP is tested by (i) the normalized number $\frac{2}{N(N-1)}(L_H - L_G)$ of additional links in the resulting graph H, (ii) the number of common links $L_c = \frac{1}{2} u^T (A \circ A_H) u$ shared by the baseline graph G and the resulting graph H and (iii) the norm $||D - \Omega|| = \frac{1}{N(N-1)} \sum_i \sum_j \frac{|d_{ij} - \omega_{ij}|}{d_{ij}}$ of the demand matrix D and the effective resistance matrix Ω. Figs. D.3, D.4 and Table D.2 demonstrate the results for tree graphs, ER graphs and empirical networks, respectively.



(a) Normalized number of additional links **(b)** Normalized number of common links **(c)** Norm difference

Figure D.3: Performance evaluation of RGP algorithm with tree baseline graphs. (a) Normalized number of additional links in the resulting graph H compared with the baseline G. (b) Normalized number of common links in the baseline graph G and the resulting graph H. (c) shows the norm between the demand matrix D and the resulting effective resistance matrix Ω. The x-axis of all three panels represents the number of nodes N of tree graphs. For each graph size N, 1000 trees are generated as baseline graphs, with integer link weights independently and uniformly drawn from $[1, 10]$. Error bars denote the standard deviation.



(a) Normalized number of additional links **(b)** Normalized number of common links **(c)** Norm difference

Figure D.4: (a) Normalized number of additional links in the resulting graph H compared with the baseline G. (b) Normalized number of common links in the baseline graph G and the resulting graph H. (c) Norm between the demand matrix D and the resulting effective resistance matrix Ω. The x-axis of all three panels represents the connection probability p of ER graphs $G_p(N)$. For each graph size N and probability p, 1000 ER graphs are generated as baseline graphs, each with link weights drawn from an exponential distribution with mean 0.5. Error bars indicate the standard deviation.

BIBLIOGRAPHY

[1] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010. ISBN: 9780199206650.

[2] A.-L. Barabási. "Network Science". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1987 (2013), p. 20120375.

[3] M. E. J. Newman. "The Structure and Function of Complex Networks". In: *SIAM Review* 45.2 (2003), pp. 167–256.

[4] P. Van Mieghem. *Performance Analysis of Complex Networks and Systems*. Cambridge, U.K.: Cambridge University Press, 2014.

[5] P. Van Mieghem et al. "A Framework for Computing Topological Network Robustness". In: *Delft University of Technology, Report 20101218* (2010), pp. 1–15.

[6] P. Van Mieghem. *Graph Spectra for Complex Networks*. Second. Cambridge University Press, 2023.

[7] P. Van Mieghem. "A Tree Realization of a Distance Matrix: The Inverse Shortest Path Problem with a Demand Matrix Generated by a Tree". In: *Delft University of Technology, Report 20211012* (2021), pp. 1–15.

[8] Z. Qiu et al. "Inverse All Shortest Path Problem". In: *IEEE Transactions on Network Science and Engineering* 11.3 (2024), pp. 2703–2714.

[9] B. Fortz and M. Thorup. "Internet traffic engineering by optimizing OSPF weights". In: *Proceedings IEEE INFOCOM 2000*. Vol. 2. IEEE. 2000, pp. 519–528.

[10] Z. Wang and J. Crowcroft. "Analysis of Shortest-Path Routing Algorithms in a Dynamic Network Environment". In: *ACM SIGCOMM Computer Communication Review* 22.2 (1992), pp. 63–71.

[11] K. Magzhan and H. M. Jani. "A Review and Evaluations of Shortest Path Algorithms". In: *International Journal of Scientific and Technological Research* 2.6 (2013), pp. 99–104.

[12] C. W. Ahn and R. S. Ramakrishna. "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations". In: *IEEE Transactions on Evolutionary Computation* 6.6 (2002), pp. 566–579.

[13] D. Bertsekas. "Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks". In: *IEEE Transactions on Automatic Control* 27.1 (1982), pp. 60–74.

[14] D. B. Johnson. "Efficient Algorithms for Shortest Paths in Sparse Networks". In: *Journal of the ACM* 24.1 (1977), pp. 1–13.

[15] R. M. Gomathi and J. M. L. Manickam. “Energy Efficient Shortest Path Routing Protocol for Underwater Acoustic Wireless Sensor Networks”. In: *Wireless Personal Communications* 98 (2018), pp. 843–856.

[16] L. Fu, D. Sun, and L. R. Rilett. “Heuristic Shortest Path Algorithms for Transportation Applications: State of the Art”. In: *Computers & Operations Research* 33.11 (2006), pp. 3324–3343.

[17] S. Pallottino and M. G. Scutella. “Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects”. In: *Equilibrium and Advanced Transportation Modelling*. Berlin, Heidelberg: Springer, 1998, pp. 245–281.

[18] D. Van Vliet. “Improved Shortest Path Algorithms for Transport Networks”. In: *Transportation Research* 12.1 (1978), pp. 7–20.

[19] D. Zhang, Y. Shou, and J. Xu. “A MapReduce-Based Approach for the Shortest Path Problem in Road Networks”. In: *Journal of Ambient Intelligence and Humanized Computing* (2024), pp. 1–9.

[20] W. B. Du, Z. X. Wu, and K. Q. Cai. “Effective Usage of Shortest Paths Promotes Transportation Efficiency on Scale-Free Networks”. In: *Physica A: Statistical Mechanics and Its Applications* 392.17 (2013), pp. 3505–3512.

[21] M. Kitsak et al. “Identification of Influential Spreaders in Complex Networks”. In: *Nature Physics* 6.11 (2010), pp. 888–893.

[22] N. E. Friedkin. “Theoretical Foundations for Centrality Measures”. In: *American Journal of Sociology* 96.6 (1991), pp. 1478–1504.

[23] C. F. Moukarzel. “Spreading and Shortest Paths in Systems with Sparse Long-Range Connections”. In: *Physical Review E* 60.6 (1999), R6263.

[24] S. Pei and H. A. Makse. “Spreading Dynamics in Complex Networks”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2013.12 (2013), P12002.

[25] R. K. Karunakaran, S. Manuel, and E. N. Satheesh. *Spreading Information in Complex Networks: An Overview and Some Modified Methods*. IntechOpen, 2017.

[26] M. Kitsak et al. “Finding Shortest and Nearly Shortest Path Nodes in Large Substantially Incomplete Networks by Hyperbolic Mapping”. In: *Nature Communications* 14.1 (2023), p. 186.

[27] B. Y. Chen et al. “Reliable Shortest Path Finding in Stochastic Networks with Spatially Correlated Link Travel Times”. In: *International Journal of Geographical Information Science* 26.2 (2012), pp. 365–386.

[28] T. H. Cormen et al. *Introduction to Algorithms*. MIT Press, 2022.

[29] E. W. Dijkstra. “A Note on Two Problems in Connexion with Graphs”. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[30] R. Bellman. “On a Routing Problem”. In: *Quarterly of Applied Mathematics* 16.1 (1958), pp. 87–90.

[31] D. Burton and Ph. L. Toint. “On an Instance of the Inverse Shortest Paths Problem”. In: *Mathematical Programming* 53.1 (1992), pp. 45–61.

[32] D. Burton, W. R. Pulleyblank, and Ph. L. Toint. "The Inverse Shortest Paths Problem with Upper Bounds on Shortest Paths Costs". In: *Network Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 156–171. ISBN: 978-3-642-59179-2.

[33] J. Zhou, F. Yang, and K. Wang. "An Inverse Shortest Path Problem on an Uncertain Graph". In: *Journal of Networks* 9.9 (2014), p. 2353.

[34] D. Burton and Ph. L. Toint. "On the Use of an Inverse Shortest Paths Algorithm for Recovering Linearly Correlated Costs". In: *Mathematical Programming* 63.1 (1994), pp. 1–22.

[35] J. Zhang, Z. Ma, and C. Yang. "A Column Generation Method for Inverse Shortest Path Problems". In: *Zeitschrift für Operations Research* 41.3 (1995), pp. 347–358. ISSN: 1432-5217.

[36] S. Xu and J. Zhang. "An Inverse Problem of the Weighted Shortest Path Problem". In: *Japan Journal of Industrial and Applied Mathematics* 12.1 (1995), pp. 47–59. ISSN: 1868-937X.

[37] S. L. Hakimi and S. S. Yau. "Distance Matrix of a Graph and Its Realizability". In: *Quarterly of Applied Mathematics* 22.4 (1965), pp. 305–317.

[38] S. P. Fekete et al. "The Complexity of an Inverse Shortest Path Problem". In: *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*. Vol. 49. American Mathematical Society, 1999, pp. 113–127.

[39] C.-H. Hung and J. Sokol. "On the Inverse Shortest Path Length Problem". AAI3110418. PhD thesis. USA: Georgia Institute of Technology, 2003.

[40] T. Cui and D. S. Hochbaum. "Complexity of Some Inverse Shortest Path Length Problems". In: *Networks* 56.1 (2010), pp. 20–29.

[41] P. Akara-pipattana, T. Chotibut, and O. Evnin. "Resistance Distance Distribution in Large Sparse Random Graphs". In: *Journal of Statistical Mechanics: Theory and Experiment* 2022.3 (Mar. 2022), p. 033404.

[42] M. E. J. Newman. "A Measure of Betweenness Centrality Based on Random Walks". In: *Social Networks* 27.1 (2005), pp. 39–54. ISSN: 0378-8733.

[43] L. C. Freeman, S. P. Borgatti, and D. R. White. "Centrality in Valued Graphs: A Measure of Betweenness Based on Network Flow". In: *Social Networks* 13.2 (1991), pp. 141–154. ISSN: 0378-8733.

[44] E. Bozzo and M. Franceschet. "Resistance Distance, Closeness, and Betweenness". In: *Social Networks* 35.3 (2013), pp. 460–469. ISSN: 0378-8733.

[45] F. A. Dicandia et al. "Space–Air–Ground Integrated 6G Wireless Communication Networks: A Review of Antenna Technologies and Application Scenarios". In: *Sensors* 22.9 (2022), p. 3136.

[46] Z. Zhang et al. "6G Wireless Networks: Vision, Requirements, Architecture, and Key Technologies". In: *IEEE Vehicular Technology Magazine* 14.3 (2019), pp. 28–41.

[47] P. Van Mieghem, K. Devriendt, and H. Cetinay. “Pseudoinverse of the Laplacian and Best Spreader Node in a Network”. In: *Physical Review E* 96.3 (2017), p. 032311.

[48] W. Ellens et al. “Effective Graph Resistance”. In: *Linear Algebra and Its Applications* 435.10 (2011), pp. 2491–2506.

[49] K. Devriendt. “Effective Resistance Is More than Distance: Laplacians, Simplices and the Schur Complement”. In: *Linear Algebra and Its Applications* 639 (2022), pp. 24–49. ISSN: 0024-3795.

[50] K. Devriendt and P. Van Mieghem. “The Simplex Geometry of Graphs”. In: *Journal of Complex Networks* 7.4 (2019), pp. 469–490.

[51] J. Sylvester. “Random Walk Hitting Times and Effective Resistance in Sparsely Connected Erdős–Rényi Random Graphs”. In: *Journal of Graph Theory* 96.1 (2021), pp. 44–84.

[52] P. Tetali. “Random Walks and the Effective Resistance of Networks”. In: *Journal of Theoretical Probability* 4.1 (1991), pp. 101–109.

[53] D. A. Spielman and N. Srivastava. “Graph Sparsification by Effective Resistances”. In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 2008, pp. 563–568.

[54] T. Simas, R. B. Correia, and L. M. Rocha. “The Distance Backbone of Complex Networks”. In: *Journal of Complex Networks* 9.6 (2021).

[55] D. J. Klein. “Resistance-Distance Sum Rules”. In: *Croatica Chemica Acta* 75.2 (2002), pp. 633–649.

[56] W. Xiao and I. Gutman. “Resistance Distance and Laplacian Spectrum”. In: *Theoretical Chemistry Accounts* 110.4 (2003), pp. 284–289.

[57] Z. Qiu et al. “Geometric organization and inference of shortest path nodes in soft random geometric graphs”. In: *Physical Review Research, under review* (2025).

[58] Katja Luck et al. “A reference map of the human binary protein interactome”. In: *Nature* 580.7803 (Apr. 2020), pp. 402–408. ISSN: 0028-0836.

[59] Nadeem Ahmed et al. “A Survey of COVID-19 Contact Tracing Apps”. In: *IEEE Access* 8 (2020), pp. 134577–134601. ISSN: 2169-3536.

[60] K. C. Claffy and D. Clark. “Challenges in Measuring the Internet for the Public Interest”. In: *Journal of Information Policy* 12 (2022), pp. 195–233.

[61] A. Schrijver. “On the History of the Shortest Path Problem”. In: *Documenta Mathematica* 17 (2012), pp. 155–167.

[62] L. R. Ford. “Network Flow Theory”. In: *RAND Corporation Paper* (1956).

[63] M. Kitsak, I. Voitalov, and D. Krioukov. “Link Prediction with Hyperbolic Geometry”. In: *Physical Review Research* 2.4 (2020), p. 043113.

[64] D. Krioukov et al. “Hyperbolic Geometry of Complex Networks”. In: *Physical Review E* 82.3 (2010), p. 036106.

[65] K. Zuev et al. “Emergence of Soft Communities from Geometric Preferential Attachment”. In: *Scientific Reports* 5.1 (2015), p. 9421.

[66] F. Papadopoulos et al. “Popularity versus Similarity in Growing Networks”. In: *Nature* 489.7417 (2012), pp. 537–540.

[67] A. P. Kartun-Giles, M. Barthelemy, and C. P. Dettmann. “Shape of Shortest Paths in Random Spatial Networks”. In: *Physical Review E* 100.3 (2019), p. 032315.

[68] J. Díaz et al. “On the Relation Between Graph Distance and Euclidean Distance in Random Geometric Graphs”. In: *Advances in Applied Probability* 48.3 (2016), pp. 848–864.

[69] R. B. Ellis, J. L. Martin, and C. Yan. “Random Geometric Graph Diameter in the Unit Ball”. In: *Algorithmica* 47.4 (2007), pp. 421–438.

[70] M. Bradonjić et al. “Efficient Broadcast on Random Geometric Graphs”. In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2010, pp. 1412–1421.

[71] T. Friedrich, T. Sauerwald, and A. Stauffer. “Diameter and Broadcast Time of Random Geometric Graphs in Arbitrary Dimensions”. In: *Algorithmica* 67 (2013), pp. 65–88.

[72] F. Voss et al. “Distributional Properties of Euclidean Distances in Wireless Networks Involving Road Systems”. In: *IEEE Journal on Selected Areas in Communications* 27.7 (2009), pp. 1047–1055.

[73] M. Raftopoulou, R. Litjens, and P. Van Mieghem. “Fréchet Distribution in Geometric Graphs for Drone Networks”. In: *Physical Review E* 106.2 (2022), p. 024301.

[74] Z. Hu and B. Li. “Fundamental Performance Limits of Wireless Sensor Networks”. In: *Ad Hoc and Sensor Networks* 81101 (2004).

[75] R. Hekmat and P. Van Mieghem. “Connectivity in Wireless Ad-Hoc Networks with a Log-Normal Radio Model”. In: *Mobile Networks and Applications* 11 (2006), pp. 351–360.

[76] Y. Zhang et al. “Connectivity Analysis for Vehicular Ad Hoc Networks Based on the Exponential Random Geometric Graphs”. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 993–998.

[77] B. Boyacı, T. H. Dang, and A. N. Letchford. “Vehicle Routing on Road Networks: How Good Is Euclidean Approximation?” In: *Computers & Operations Research* 129 (2021), p. 105197.

[78] S. Buczkowska, N. Coulombel, and M. De Lapparent. “A Comparison of Euclidean Distance, Travel Times, and Network Distances in Location Choice Mixture Models”. In: *Networks and Spatial Economics* 19.4 (2019), pp. 1215–1248.

[79] S. Ahmed et al. “Covert Cyber Assault Detection in Smart Grid Networks Utilizing Feature Selection and Euclidean Distance-Based Machine Learning”. In: *Applied Sciences* 8.5 (2018), p. 772. ISSN: 2076-3417.

[80] U. Islambekov et al. “Role of Local Geometry in Robustness of Power Grid Networks”. In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 2018, pp. 885–889.

[81] M. D. Penrose. “Connectivity of Soft Random Geometric Graphs”. In: *The Annals of Applied Probability* 26.2 (2016), pp. 986–1028.

[82] G. Mao and B. D. O. Anderson. “Connectivity of Large Wireless Networks under a General Connection Model”. In: *IEEE Transactions on Information Theory* 59.3 (2012), pp. 1761–1772.

[83] K. Bringmann, R. Keusch, and J. Lengler. “Geometric Inhomogeneous Random Graphs”. In: *Theoretical Computer Science* 760 (2019), pp. 35–54.

[84] T. Bläsius et al. “Efficiently Generating Geometric Inhomogeneous and Hyperbolic Random Graphs”. In: *Network Science* 10.4 (2022), pp. 361–380.

[85] J. Dall and M. Christensen. “Random Geometric Graphs”. In: *Physical Review E* 66.1 (2002), p. 016121.

[86] M. Penrose. *Random Geometric Graphs*. Vol. 5. Oxford, U.K.: Oxford University Press, 2003.

[87] Piet Van Mieghem. “Paths in the simple random graph and the Waxman graph”. In: *Probability in the Engineering and Informational Sciences* 15.4 (2001), pp. 535–555.

[88] B. M. Waxman. “Routing of Multipoint Connections”. In: *IEEE Journal on Selected Areas in Communications* 6.9 (2002), pp. 1617–1622.

[89] V. Erba et al. “Random Geometric Graphs in High Dimension”. In: *Physical Review E* 102.1 (2020), p. 012306.

[90] Dmitri Krioukov et al. “Hyperbolic geometry of complex networks”. In: *Physical Review E* 82.3 (Sept. 2010), p. 036106. ISSN: 1539-3755.

[91] Marián Boguñá et al. “Small worlds and clustering in spatial networks”. In: *Physical Review Research* 2.2 (Apr. 2019), p. 023040. ISSN: 2643-1564.

[92] D. M. W. Powers. “Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation”. In: *arXiv preprint arXiv:2010.16061* (2020).

[93] Ž. Vujović et al. “Classification Model Evaluation Metrics”. In: *International Journal of Advanced Computer Science and Applications* 12.6 (2021), pp. 599–606.

[94] G Hooghiemstra and P Van Mieghem. “On the mean distance in scale free graphs”. In: *Methodology and Computing in Applied Probability* 7.3 (2005), pp. 285–306.

[95] Filippo Simini et al. “A universal model for mobility and migration patterns”. In: *Nature* 484.7392 (2012), pp. 96–100.

[96] Antoine Allard et al. “The geometric nature of weights in real complex networks”. In: *Nature Communications* 8.1 (2017), p. 14103.

[97] Ivan Voitalov et al. “Weighted hypersoft configuration model”. In: *Physical Review Research* 2.4 (2020), p. 043157.

[98] M. Kitsak et al. “Stability of a Giant Connected Component in a Complex Network”. In: *Physical Review E* 97.1 (Jan. 2018), p. 012309.

[99] Y. Yang, T. Nishikawa, and A. E. Motter. “Small Vulnerable Sets Determine Large Network Cascades in Power Grids”. In: *Science* 358.6365 (2017), eaan3184.

[100] L. Lü et al. “Vital Nodes Identification in Complex Networks”. In: *Physics Reports* 650 (2016), pp. 1–63.

[101] Z. Qiu et al. “Identifying Vital Nodes by Achlioptas Process”. In: *New Journal of Physics* 23.3 (2021), p. 033036.

[102] L. C. Freeman. “A Set of Measures of Centrality Based on Betweenness”. In: *Sociometry* (1977), pp. 35–41.

[103] M. Kitsak et al. “Betweenness Centrality of Fractal and Nonfractal Scale-Free Model Networks and Tests on Real Networks”. In: *Physical Review E* 75.5 (May 2007), p. 056115.

[104] A. Nayak and I. Stojmenovic. *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. Wiley Online Library, 2010.

[105] S. K. Singh, R. Singh, and B. Kumbhani. “The Evolution of Radio Access Network Towards Open-RAN: Challenges and Opportunities”. In: *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2020, pp. 1–6.

[106] A. M. Mercier, S. V. Scarpino, and C. Moore. “Effective Resistance for Pandemics: Mobility Network Sparsification for High-Fidelity Epidemic Simulation”. In: *arXiv preprint arXiv:2111.02449* (2021).

[107] P. Van Mieghem. *Data Communications Networking*. Third. ISBN 978-94-91075-01-8. Piet Van Mieghem, 2018.

[108] P. Van Mieghem and S. M. Magdalena. “Phase Transition in the Link Weight Structure of Networks”. In: *Physical Review E* 72.5 (2005), p. 056138.

[109] P. Van Mieghem and S. van Langen. “Influence of the Link Weight Structure on the Shortest Path”. In: *Physical Review E* 71.5 (2005), p. 056113.

[110] P. Van Mieghem and H. Wang. “The Observable Part of a Network”. In: *IEEE/ACM Transactions on Networking* 17.1 (2009), pp. 93–105.

[111] A. Ghosh, S. Boyd, and A. Saberi. “Minimizing Effective Resistance of a Graph”. In: *SIAM Review* 50.1 (2008), pp. 37–66.

[112] D. Goldfarb and A. Idnani. “A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs”. In: *Mathematical Programming* 27.1 (1983), pp. 1–33. ISSN: 1436-4646.

[113] M. Fiedler. “Some Characterizations of Symmetric Inverse M-Matrices”. In: *Linear Algebra and Its Applications* 275–276 (1998), pp. 179–187. ISSN: 0024-3795.

[114] M. Fiedler. *Matrices and Graphs in Geometry*. Cambridge, U.K.: Cambridge University Press, 2011.

[115] R. Albert and A.-L. Barabási. “Statistical Mechanics of Complex Networks”. In: *Reviews of Modern Physics* 74.1 (2002), pp. 47–97.

[116] D. J. Watts and S. H. Strogatz. “Collective Dynamics of ‘Small-World’ Networks”. In: *Nature* 393.6684 (1998), pp. 440–442.

[117] Ryan Rossi and Nesreen Ahmed. “The Network Data Repository with Interactive Graph Analytics and Visualization”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Mar. 2015).

[118] “Industrial Communication Networks—Wireless Communication Network and Communication Profiles—WirelessHART™”. In: *IEC Standard 62591: Geneva, Switzerland* (2010), p. 944.

[119] J. Schönwälder, M. Björklund, and P. Shafer. “Network Configuration Management Using NETCONF and YANG”. In: *IEEE Communications Magazine* 48.9 (2010), pp. 166–173.

[120] S. Rommel, R. T. Raddo, and I. T. Monroy. “The Fronthaul Infrastructure of 5G Mobile Networks”. In: *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2018, pp. 1–6.

[121] Z. Qiu, S. Tang, and P. Van Mieghem. “Inverse all shortest path problem with given network topology”. In: *In preparation* (2025).

[122] International Telecommunication Union. *Framework and overall objectives of the future development of IMT for 2030 and beyond*. Recommendation M.2160-0. Accessed: Jan. 2, 2026. Geneva, Switzerland: ITU, 2023.

[123] A. Leitão et al. “A Genetic Algorithms Approach for Inverse Shortest Path Length Problems”. In: *International Journal of Natural Computing Research (IJNCR)* 4.4 (2014), pp. 36–54.

[124] Yogita Pimpalkar et al. “A Novel E2E Path Selection Algorithm for Superior QoS and QoE for 6G Services”. In: *IEEE Transactions on Network and Service Management* 22.2 (Apr. 2025), pp. 1174–1187.

[125] A. N. Kim et al. “When HART Goes Wireless: Understanding and Implementing the WirelessHART Standard”. In: *Proceedings of the 2008 IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2008, pp. 899–907.

[126] Saleem Raza, Muhammad Faheem, and Mesut Guenes. “Industrial wireless sensor and actuator networks in industry 4.0: Exploring requirements, protocols, and challenges—A MAC survey”. In: *International Journal of Communication Systems* 32.15 (2019), e4074.

[127] S. Senk et al. “Flexible Measurement Testbed for Evaluating Time-Sensitive Networking in Industrial Automation Applications”. In: *Proceedings of the 2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 402–410.

[128] R. Enns et al. *Network Configuration Protocol (NETCONF)*. RFC 6241. 2011.

[129] Tibor Cinkler et al. “Optimizing QoS aware ethernet spanning trees”. In: *2005 1st International Conference on Multimedia Services Access Networks, 2005. MSAN’05.* IEEE. 2005, pp. 30–34.

[130] K. Ishizu, M. Kuroda, and K. Kamura. “SSTP: an 802.1s extension to support scalable spanning tree for mobile metropolitan area network”. In: *IEEE Global Telecommunications Conference, 2004. GLOBECOM ’04*. Vol. 3. 2004, 1500–1504 Vol.3.

[131] George B. Dantzig. *Linear Programming and Extensions*. Princeton Landmarks in Mathematics and Physics. New Jersey: Princeton University Press, 2016, p. 656. ISBN: 9781400884179.

[132] Wilfried Steiner et al. “Next generation real-time networks based on IT technologies”. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2016, pp. 1–8.

[133] Marek Vlk et al. “Enhancing Schedulability and Throughput of Time-Triggered Traffic in IEEE 802.1Qbv Time-Sensitive Networks”. In: *IEEE Transactions on Communications* 68.11 (2020), pp. 7023–7038.

[134] Z. Qiu et al. “Flow Subgraphs and Flow Network Design under End-to-End Power Dissipation Constraints”. In: *Physical Review E, under review* (2025).

[135] F. B. Zhan and C. E. Noon. “Shortest Path Algorithms: An Evaluation Using Real Road Networks”. In: *Transportation Science* 32.1 (1998), pp. 65–73.

[136] F. Begtaševiü and P. Van Mieghem. “Measurements of the Hopcount in Internet”. In: *Proceedings of the Workshop on Passive and Active Measurement (PAM)*. 2001, pp. 183–190.

[137] Y.-X. Qiu et al. “Efficient Shortest Path Counting on Large Road Networks”. In: *Proceedings of the VLDB Endowment* (2022).

[138] R. Pastor-Satorras et al. “Epidemic Processes in Complex Networks”. In: *Reviews of Modern Physics* 87.3 (2015), pp. 925–979.

[139] C. Nowzari, V. M. Preciado, and G. J. Pappas. “Analysis and Control of Epidemics: A Survey of Spreading Processes on Complex Networks”. In: *IEEE Control Systems Magazine* 36.1 (2016), pp. 26–46.

[140] E. Bozzo and M. Franceschet. “Approximations of the Generalized Inverse of the Graph Laplacian Matrix”. In: *Internet Mathematics* 8.4 (2012), pp. 456–481.

[141] N. Masuda, M. A. Porter, and R. Lambiotte. “Random Walks and Diffusion on Networks”. In: *Physics Reports* 716–717 (2017). Random walks and diffusion on networks, pp. 1–58. ISSN: 0370-1573.

[142] K. Pearson. “The Problem of the Random Walk”. In: *Nature* 72.1867 (1905), pp. 342–342.

[143] G. Guennebaud and A. Bugeau. “Energy Consumption of Data Transfer: Intensity Indicators Versus Absolute Estimates”. In: *Journal of Industrial Ecology* 28.4 (2024), pp. 996–1008.

[144] David Costenaro and Anthony Duer. “The megawatts behind your megabytes: going from data-center to desktop”. In: *Proceedings of the 2012 ACEEE Summer Study on Energy Efficiency in Buildings, ACEEE, Washington* (2012), pp. 13–65.

D

[145] Petter Holme and Gourab Ghoshal. “The Diplomat’s Dilemma: Maximal Power for Minimal Effort in Social Networks”. In: *Adaptive Networks: Theory, Models and Applications*. Understanding Complex Systems. Berlin, Heidelberg: Springer, 2009, pp. 269–288.

[146] Matthew O. Jackson. “The Study of Social Networks in Economics”. In: *The Missing Links: Formation and Decay of Economic Networks*. Ed. by James E. Rauch and Alessandra Casella. New York: Russell Sage Foundation, 2007, pp. 1–24.

[147] Avinatan Hassidim et al. “Network utilization: The flow view”. In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 1429–1437.

[148] David J. Aldous. “Optimal spatial transportation networks where link costs are sublinear in link capacity”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2008.03 (2008), P03006.

[149] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. “Random Graphs with Arbitrary Degree Distributions and Their Applications”. In: *Physical Review E* 64.2 (2001), p. 026118.

[150] P. Erdős and A. Rényi. “On the Evolution of Random Graphs”. In: *Publ. Math. Inst. Hungar. Acad. Sci.* 5 (1960), pp. 17–61.

[151] D. S. Callaway et al. “Network Robustness and Fragility: Percolation on Random Graphs”. In: *Physical Review Letters* 85.25 (2000), pp. 5468–5471.

[152] Robert M. Corless et al. “On the Lambert W function”. In: *Advances in Computational Mathematics* 5.1 (1996), pp. 329–359.

[153] Chris Godsil and Gordon F Royle. *Algebraic graph theory*. Vol. 207. Springer Science & Business Media, 2013.

[154] Paul Erdos and Alfréd Rényi. “Asymmetric graphs”. In: *Acta Math. Acad. Sci. Hungar* 14.295-315 (1963), p. 3.

[155] Giovanni Alessandrini. “Stable determination of conductivity by boundary measurements”. In: *Applicable Analysis* 27.1-3 (1988), pp. 153–172.

[156] Ángeles Carmona et al. “Stable recovery of piecewise constant conductance on spider networks”. In: *International Journal of Computer Mathematics* 101.11 (2024), pp. 1237–1254.

[157] Hannes Gernandt and Jonathan Rohleder. “A Calderón type inverse problem for tree graphs”. In: *Linear Algebra and its Applications* 646 (2022), pp. 29–42. ISSN: 0024-3795.

[158] Anonymous. *Plano del Arzobispado de México*. Accessed 9 June 2018. Mexico, 2018.

[159] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

[160] U. Von Luxburg, A. Radl, and M. Hein. “Hitting and Commute Times in Large Random Neighborhood Graphs”. In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1751–1798.

[161] S. Carmi et al. “Transport Between Multiple Users in Complex Networks”. In: *The European Physical Journal B* 57.2 (2007), pp. 165–174.

[162] S. Carmi et al. “Transport in Networks with Multiple Sources and Sinks”. In: *Europhysics Letters* 84.2 (2008), p. 28005.

[163] C.-F. Chiasserini and M. Garetto. “An Analytical Model for Wireless Sensor Networks with Sleeping Nodes”. In: *IEEE Transactions on Mobile Computing* 5.12 (2006), pp. 1706–1718.

[164] P. Van Mieghem. “Eigenvectors and Eigenvalues of the Effective Resistance Matrix of a Graph”. In: *Delft University of Technology, Report 20250616* (2025), pp. 1–32.

[165] L. Mirsky. *An Introduction to Linear Algebra*. Dover Publications, Inc., 1982.

[166] B. A. Miller et al. “PATHATTACK: Attacking Shortest Paths in Complex Networks”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 532–547.

[167] M. R. Garey and D. S. Johnson. *Computers and Intractability*. New York: W. H. Freeman and Company, 1979. ISBN: 0-7167-1044-7.

ACKNOWLEDGEMENTS

Completing my PhD has been a deeply rewarding and transformative journey. I am profoundly grateful to everyone who supported me over the past four years.

Specifically, I would like to thank my promotor and supervisor, Prof. Piet Van Mieghem, for granting me the opportunity to study and conduct research within NAS, our research group. I am very thankful for his encouragement to focus on the *Inverse all shortest path problem* (IASPP). IASPP has proven to be an extraordinary topic, and more than half of this thesis has grown from the original seed planted by Piet. I also appreciate the time and meticulous feedback on my NAS reports. His guidance was instrumental in improving my academic writing. I would extend my deepest gratitude to my daily supervisor, Dr. Maksim Kitsak. Maksim taught me how to model complex systems with mathematical elegance and how to synthesize and package our research into a powerful narrative. Through his own example and his careful attention to the smallest details, Maksim taught me what it means to be a rigorous researcher. I am also grateful for the many life lessons he shared, which have been as valuable as his academic guidance.

Next, I would like to thank all my amazing collaborators. It would not be possible for me to finish all my work without them. I would like to thank Professor Siyu Tang for her time and effort in providing feedback on our joint work, which greatly improved its quality. I would also like to thank Ivan, Rogier, Xinhan and Samu for their insightful discussions and valuable contributions to our joint work. My appreciation then extends to all my doctoral committees: Prof. M.E.Warnier, Dr. E. Smeitink, Prof. R. Kooij, Dr. J. L. A. Dubbeldam and Prof. G.N. Gaydadjiev.

Now, I would like to thank my office roommates: Massimo, Yingyue, Maria, Benedetta and Federico. It has been a great pleasure to share a room and have nice discussions on both academic and non-academic topics. Thank you all for your constant support.

My appreciation also extends to all my colleagues at NAS: Caterina, Edgar, Remco, Scott, Sergey, Yuxuan, Robert(J), Long, Peng, Gabriel, Liza, Matteo(D), David, Yongding, Roberto, Robert(A), Matteo(C). I truly cherish the time we spent together. I would also like to extend my deep and sincere thanks to our supporting staff: Paul, Francis and Helen.

I would like to thank my colleagues in the MMC group: Huijuan, Tianrui and Maosheng. I am also grateful to my colleagues Naqi and Erbing, who are from other groups but have provided me with a lot of help. I would further like to thank my badminton teammates: Li(W), Wei, Yun, Jeroen and Xuliang.

Now, it is time to thank my friends, who are also my traveling/hotpot/mahjong/beer *dazi*: Fenghua, Li(W), Li(Z), Tianqi, Yingyue, Brian and Robin. Our gatherings brought me so much relaxation and joy amid the demands of research. I am deeply grateful for your friendship throughout this journey, which made me feel far from alone. A special thanks to my shixiong Shilun and dajie Jingjing for guiding and supporting me from

UESTC to TU Delft. I treasure every moment we spent together—whether celebrating festivals, cycling, or most importantly, discussing how to face the challenges before us.

I would like to sincerely thank the Chinese and Dutch governments for their support, which made it possible for me to pursue my PhD.

I would like to express my deepest gratitude to my parents for their unconditional love, support and understanding. Their encouragement and care have always been a constant source of strength for me.

I would like to extend my heartfelt thanks to my love Zhilin for standing by me throughout this journey, offering unwavering companionship, understanding and encouragement during both challenging and joyful times.

Finally, I would also like to thank myself for the perseverance and courage throughout this journey.

To all who have been part of this journey, thank you very much. I wish each of you a happy life and continued success in all your future endeavors.

Curriculum Vitæ

Zhihao Qiu

Zhihao Qiu was born in Shandong, China, on June 29, 1996. He received his B.Sc. degree in Electronic Information Science and Technology from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2018 and his M.Sc. degree in Information Physics from UESTC in 2021. Since the end of 2021, he has been a PhD candidate at the Network Architecture and Services (NAS) group at Delft University of Technology, under the supervision of Prof.dr.ir. Piet Van Meighem and Dr. M.A. Kitsak.

His main research interests include network science, shortest path, inverse shortest path problem, geometric shortest path, inverse effective resistance problem, vital node identification and science of science.

LIST OF PUBLICATIONS

5. **Qiu Z**, Balogh S.G., Liu X., et al. Geometric organization and inference of shortest path nodes in soft random geometric graphs[J]. Physical Review Research, under review.

4. **Qiu Z**, Liu, X., Noldus R., et al. Flow Subgraphs and Flow Network Design under End-to-End Power Dissipation Constraints[J]. Physical Review E, under review.

3. Ma L, **Qiu Z**, Van Mieghem P, et al. Reporting delays: A widely neglected impact factor in COVID-19 forecasts[J]. PNAS nexus, 2024, 3(6).

2. **Qiu Z**, Jokić I, Tang S, et al. Inverse all shortest path problem[J]. IEEE Transactions on Network Science and Engineering, 2023, 11(3): 2703-2714.

1. **Qiu Z**, Fan T, Li M, et al. Identifying vital nodes by Achlioptas process[J]. New Journal of Physics, 2021, 23(3): 033036.